Scalable approaches to inference and analysis of genome-wide genealogies



Árni Freyr Gunnarsson

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

21.12.2023

Abstract

The genealogical history of a set of genomes is encoded by its ancestral recombination graph (ARG), a graph-like structure that represents ancestral recombination and coalescence events. ARGs have proven a useful tool for a range of tasks in statistical and population genetics, but in many cases the inference of an ARG from observed genotypes presents a strict computational bottleneck. In this text we present Threads, a highly scalable and accurate ARG inference algorithm. We show through extensive simulations that Threads is significantly more computationally efficient than other ARG inference methods, while achieving high accuracy across a range of metrics. Using Threads, we explore new ARG-based applications in genotype compression, phasing, imputation, and association studies, in each case observing unique benefits obtained through the use of ARGs. Together, our results present a strong argument in favor of the ARG as a standard feature of the statistical geneticist's toolkit, and with Threads, efficient and accurate inference of ARGs is made feasible for modern, biobank-scale datasets.

Contents

1 Introduction **1**

2 Background **6**

2.1 The coalescent and ancestral recombination graphs 6

2.1.1 The coalescent 6

2.1.2 ARG inference and threading 10

2.1.3 Tree-based metrics of inference accuracy 12

2.2 The positional Burrows-Wheeler transform 17

2.2.1 The dynamic PBWT 18

2.3 The Li-Stephens haplotype matching model 19

2.3.1 FastLS 21

3 Threads **22**

3.1 The Threads algorithm 22

3.1.1 Haplotype matching with the PBWT 23

3.1.2 A memory-efficient Li-Stephens path 26

3.1.3 Dating path segments 34

3.1.3.1 Maximum likelihood estimators 36

3.1.3.2 Bayesian estimators 36

3.1.3.3 Piecewise-constant demography 37

3.1.4 The limits of identity-by-descent modelling 39

3.2 The Threads-fastLS algorithm 45

3.2.1 Proof of correctness 48

3.2.2 Controlling the number of active states 51

3.2.3 Comparison with Threads 53

3.3 Diploid Threads-fastLS 55
3.3.1 Proof of correctness 60
3.4 An efficient diploid fastLS 63
3.5 ARG inference, extensions and heuristics 64
3.5.1 Setting the Li-Stephens parameters 65
3.5.2 Sparse sites 66
3.5.3 Threading small sample sizes 68
3.5.4 Adaptive Threads-fastLS 69
3.5.5 Seeded inference 70
3.5.6 Mutation-consistent threading 71

4 Inference accuracy and computational performance **74**
4.1 Accuracy metrics 74
4.2 Simulation parameters 76
4.3 Simulation results 77
4.3.1 Genotyping arrays 78
4.3.2 Robustness to reduced mutation rate 80
4.3.3 Robustness to genotyping errors 80
4.3.4 Imputed data 83
4.3.5 Seeded ARG inference 86
4.4 Computational performance 88

5 Genotype compression **94**
5.1 Genotypes as threading instructions 95
5.2 Simulation studies 96
5.2.1 Genotype dosages 102
5.3 Real data 103
5.3.1 1000 Genomes 103

5.3.2 UK Biobank genotyping arrays . . . 104
5.3.3 UK Biobank imputed variants . . . 105

6 Genotype phasing and imputation **107**
6.1 Background . . . 108
6.1.1 The limits of haplotype copying . . . 109
6.2 Imputation via threading . . . 110
6.2.1 Placing mutations on ARGs . . . 112
6.3 Simulation studies . . . 114
6.3.1 Association studies . . . 116
6.4 Imputation using the 1000 Genomes . . . 116
6.5 ARG-based genotype phasing . . . 117
6.6 Discussion . . . 120

7 Genealogy-wide association studies **123**
7.1 Background . . . 123
7.1.1 Runtime improvements with Threads . . . 126
7.2 GeWAS of African and Central/South Asian samples from the UK Biobank . . . 127

8 Discussion **133**

Bibliography **139**

Chapter 1

Introduction

The complete genealogical history of a set of genomes can be captured by its ancestral recombination graph (ARG). Due to randomly occurring recombination events in the past, the ARG is not a simple genealogical tree, but a graph-like structure with nodes representing ancestral genomes and edges representing lineages [21, 22, 28, 67]. This graph may contain a different genealogy for each site in the genome, allowing us to trace a unique history for each genetic variant.

Inference of ARGs from observed genetic variation has been made possible by statistical modelling of historical recombination and mutation events and access to accurately inferred genome-wide genealogies has allowed for the development of ARG-based applications in statistical and population genetics. These include ARG-based genetic association studies and complex trait analysis [60, 99, 103], inference of demographic structure and natural selection [25, 84, 95], genotype compression [39] and polygenic prediction of human traits [65].

However, in many of these applications, inference of the ARG is a computational bottleneck. Current methods do not scale to biobank-sized, whole-genome sequenced data sets and although biobank-scale ARGs have been inferred previously [39, 99], these have relied on the sparse set of common variants used for

genotyping arrays. By omitting most of the rare variation in the genome we are left with less signal to accurately infer recent history in the sample, which can affect downstream applications.

In this work, we describe Threads, a highly scalable and accurate ARG inference method. We use Threads to infer an ARG for 487,407 samples from the UK Biobank [10] using 10 million high-quality imputed variants and through extensive simulation and benchmarking, we verify that Threads obtains high accuracy across a range of ARG- and tree-based metrics.

Threads relies on the algorithmic paradigm of "threading", whereby new genomes are sequentially added to a partial ARG. The idea of threading new samples to an ARG was previously implemented in ARGWeaver [73] and ARG-Needle [99]. We achieve significant computational improvements over both methods through the use of several techniques, some of which were developed in other domains of statistical genetics. These include a haplotype refinement step using the positional Burrows-Wheeler transform [17], a low-memory implementation of the Li-Stephens model [48] and a likelihood-based approximation for coalescent times. Related algorithmic ideas have appeared in methods for genotype phasing and imputation [8, 9, 27, 76].

Using Threads, we explored how ARGs could be used to improve upon or complement traditional approaches to four familiar tasks in statistical genetics. First, we explored how the genealogical inference in Threads can be used to compress genotype data on disk. We observed up to 94% compression relative to traditional genotype compression formats, while storing not just the genotypes but their complete genealogical history. This compression is obtained by noting that the "threading instructions" inferred by Threads are sufficient to uniquely recover the ARG and the genotypes used to infer it. In both simulations and real-data experiments, Threads outperformed tsinfer [39], another ARG-based method for genotype compression.

Second, we studied whether ARGs might be used to improve accuracy in genotype imputation. Genotype imputation is the process of filling in missing genotypes from sparsely sequenced haplotypes using patterns of linkage disequilibrium and haplotype sharing with a densely sequenced reference panel. Imputation is an important first step for association studies and other applications [10]. ARG-based genotype imputation was previously attempted by Zhang et al. in [99] without improvement over widely used methods based on haplotype copying using the Li-Stephens model [48] such as Beagle 5.4 and IMPUTE5 [9, 76]. Our experiments were motivated by the observation that imputed genotype dosages of ultra-rare variants are systematically inflated in traditional models by not accounting for the possibility that the sparsely sequenced target sequence might coalesce with its closest genealogical cousin at a time in the past greater than the age of the mutation being imputed [81]. We hypothesized that by leveraging the ARG of the reference panel and inferring the age of mutations, we might improve the imputation quality of ultra-rare variants. In our simulations and in experiments on the 1000 Genomes Project [3, 11], we found that using the ARG provides significant improvement in imputation quality for variants up to minor allele count of around 10 in the panel, independent of reference panel size. These results are in concordance with our estimates of which variants are most affected by the dosage inflation.

Third, we developed a way to use ARGs for rare variant phasing. Genotype phasing is the process of decomposing an observed diploid genome into its two parental haplotypes. Motivated by the success of our ARG-based imputation approach, we developed an ARG-based method for the phasing of rare variants. Modern phasing algorithms are usually composed of a two-step approach, whereby a high-quality scaffold of phased common variants is built and then used to inform the phasing of rarer variants. For our approach, we build common-variant scaffolds using SHAPEIT5 [27] and use the inferred ARGs of the scaffolds to inform the

phasing of rare variants. In experiments on the 1000 Genomes Project, we observed similar accuracy to SHAPEIT5. However, our ARG-based method underperformed in simulations, suggesting the need for additional methodological development.

Fourth, we studied whether the ARG may be used to improve association studies of rare variants in poorly imputed populations. The ARG can uncover links between phenotypes and variants that are either poorly imputed or unobserved in reference panels [99]. Here, we focused on non-European samples from the UK Biobank. We performed a genealogy-wide association study (GeWAS) of seven quantitative traits, finding that compared with imputation, the ARG improves tagging of a rare, African-specific missense variant in GPLD1 associated with alkaline phosphatase levels. Our results also show that more independent association signals are uncovered using an ARG based on a dense set of imputed variants than using only genotyping array variants, emphasizing the importance of using high-quality ARGs inferred from dense marker sets. Moreover, we show that the minimal ARG representation obtained by Threads can speed up the GeWAS by close to an order of magnitude compared with ARG-Needle [99] since the less fragmented graph structure allows for more efficient ARG traversal.

The structure of the rest of this text is as follows. In Chapter 2, we present the theoretical background used to develop our inference algorithms. This chapter is divided into three main topics, the coalescent and ARGs, the positional Burrows-Wheeler transform, and the Li-Stephens haplotype matching model. Chapter 3 describes Threads, our ARG inference algorithm, and its variant, Threads-fastLS, in detail, with extensive computational benchmarking and accuracy measurements in simulations provided in Chapter 4. In Chapter 5, we give a detailed discussion on how threading instructions may be used to compress genotype data. Chapter 6 presents our imputation and phasing algorithms, with benchmarks in simulations and real data, and Chapter 7 presents our results for genealogy-wide association

studies of non-European samples in the UK Biobank. Finally, Chapter 8 provides a brief discussion of the results and methods presented in the previous chapters.

Chapter 2

Background

In this chapter we introduce useful concepts and results from statistical and population genetics that we use to develop our inference algorithms for ancestral recombination graphs in Chapter 3 and to discuss our applications in Chapters 5-7. In Chapter 2.1 we introduce the coalescent with recombination, the standard approach for genealogical modelling. We also define the ancestral recombination graph and discuss inference approaches and accuracy metrics. Chapter 2.2 discusses the positional Burrows-Wheeler transform and its dynamic extension, the d-PBWT. Finally, in Chapter 2.3 we introduce the Li-Stephens model, along with an efficient variant, the fastLS algorithm.

2.1 The coalescent and ancestral recombination graphs

2.1.1 The coalescent

The coalescent is a stochastic process that describes the common ancestry of a set of genomes. Starting with a number of observed individuals at a fixed genomic site, the process moves backwards in time; pairs of lineages, which represent the ancestors of the original individuals, coalesce at a population-wide coalescence rate,

merging to form a single lineage. The result of this process is a tree with nodes representing coalescence events and edges representing lineages, together giving a genealogy for the observed alleles [24, 42].

The rate at which each pair of lineages coalesces follows an exponential distribution with rate $\gamma = 1/N_e$, where N_e is called the *effective population size*, giving a tree-wide coalescence rate of $\binom{K(t)}{2} \cdot \gamma$, where $K(t)$ is the number of open lineages at time t. Here, we adopt the convention of using N_e to represent haploid population size, rather than diploid. The coalescence rate is often rescaled; in the *standard coalescent*, as defined by Kingman [42], it is assumed to be $\gamma = 1$. In reality, population sizes are not constant through time and the coalescent can similarly be defined with a variable-sized demography $N_e(t)$, where t represents time in the past [47, 80]. However, many properties of the standard coalescent also hold for arbitrary demography, up to some monotonic scaling of time. In this work, we often refer to the function $N_e(t)$ as *demographic history* or *demography*, although we note that an idealized demographic history may be more general, e.g. it may involve multiple populations.

Denote by $S_k \sim \text{Exp}\binom{k}{2}$ the random waiting time for the $(N-k+1)$-th coalescence event. We can then obtain the total expected tree volume V_N for a standard coalescence tree on N leaves as

$$E[V_N] = \sum_{k=2}^{N} kS_k = \sum_{k=2}^{N} \frac{k}{\binom{k}{2}} = \sum_{k=1}^{N-1} \frac{2}{k} \approx 2\log(N),$$

tending to infinity as N grows. The total height of a coalescence tree, H_N, satisfies

$$E[H_N] = \sum_{k=2} NE[S_k] = \sum_{k=2} N\frac{2}{k(k-1)} = 2\left(1 - \frac{1}{N}\right),$$

tending to 2 as N grows.

We may also inductively obtain the standard coalescent on $N + 1$ leaves by adding a new lineage at time 0 to the coalescent on N leaves, coalescing at rate $K(t)$ to any of the open lineages in the existing tree. Denote the time until its first coalescence by L. At each time t the total coalescence rate becomes the minimum of the exponential variables L and $S_{K(t)}$. This gives an exponential variable with rate $K(t) + \binom{K(t)}{2} = \binom{K(t)+1}{2}$, recovering the coalescent on $N + 1$ leaves and an expected length of

$$E[L] = E[V_{N+1}] - E[V_N] = \frac{2}{N}$$

for the new lineage.

Together, these observations emphasize the fact that as N grows, the volume of coalescence trees concentrates towards the bottom, resulting in a rapid burst of coalescence events in the recent past, gradually slowing as fewer lineages are left in the tree in the deeper past. Thus, to analyze the ancestry of large genotype sets in high resolution from observed genotypes we must use methods that effectively model the recent past.

Extending the coalescent from a single locus to whole sequences we must also account for recombination. If the coalescent is a pure death process going backwards in time, we may think of the coalescent with recombination as a birth-death process, with lineages merging under coalescence and splitting with recombination. Here, we will only focus on Hudson's model of recombination [28], which does not consider non-crossover recombination events. After the recombination event, the genetic material to the left and right of the breakpoint can be traced to different ancestors. This process terminates when only one lineage is left and the topmost node represents the *grand most recent common ancestor* (GMRCA) of the whole sample. A realization of this process is a complex and possibly cyclic graph, called the *ancestral recombination graph* (ARG) of the sample [21, 22, 24]. Marginally at each site, it simplifies to a unique coalescence tree describing the local ancestry.

The ARG defined as a realization of the coalescent with recombination as described above means it may contain both nodes and lineages carrying no material ancestral to the sample, and the process may continue for a long time after a most recent common ancestor has been found at each site. Removing this information, which is redundant for most downstream analyses, results in a minimal structure called Hudson's graph [24, 28]. ARGs described in this work further adopt the widely used convention that coalescence nodes represent ancestral genomes and that nodes representing recombination events are identified with the closest ancestral genome below them [38, 67, 84, 99]. This simplification does not affect the distribution of observable genotypes under our modelling assumptions for mutations arising along the ARG.

The coalescent with recombination provides another viewpoint of the inference of ancestry in large sample sizes. Human mutation rates are not sufficiently high to ensure that every lineage in the ARG carries a non-ancestral mutation. Modelling of recombination, however, allows us to leverage additional information for genealogical inference as recombination events lead to patterns of haplotype sharing that can be used to inform ARG inference. An example of these useful patterns is the presence of tracts of genome carried by two or more sequences and inherited from the same ancestor, sometimes referred to as identical-by-descent (IBD) haplotypes. The length of these tracts can be shown be roughly inversely proportional to their age [68, 77], a relationship that can be used in genealogical inference. Leveraging this kind of long-range haplotypic pattern requires the development of ARG-inference algorithms that scale to long tracts of genome, which is computationally challenging.

2.1.2 ARG inference and threading

Inference of ancestral recombination graphs from observable genotypes is a highly complex computational task. The search space for ARG topologies is vast and all but the simplest ARG-related likelihoods are prohibitively expensive to compute. To overcome these difficulties, all widely used ARG inference methods rely on a set of inference heuristics and simplifications that make the problem of ARG inference more tractable [39, 84, 95, 99].

One such approach, called *threading*, was first introduced by Rasmussen et al. with ARGweaver [73] and used for de-novo ARG inference by Zhang et al. with ARG-needle [99]. Threading proceeds by iteratively adding new samples to a partial ARG. More formally, an ARG over a genomic region $[L_1, L_2]$ may be uniquely determined by a set of *threading instructions* [99]. For each sample $n = 1, \ldots, N$, threading instructions consist of a map $[L_1, L_2] \to \{1, \ldots, n-1\} \times \mathbb{R}_+$, specifying for each site within the region a *threading target* and a *coalescence time*. The threading target is the index of an individual, which we sometimes refer to as a *closest cousin*, that is already in the ARG and is most closely related to the target sample. Multiple such individuals may exist. The coalescence time refers to the value of the time at which the target individual and its closest cousin coalesce, i.e., the genealogical age of their most recent common ancestor. Once threading instructions have been inferred, a set of new branches is grafted onto the ARG along the unique lineages between the threading targets and the root at height determined by the coalescence time (Figure 2.1).

Threading has proved to be an efficient ARG inference technique, especially for large sample sizes, as it scales well to the long genomic regions required to infer recent ancestry. Threading inherently relies on pairwise comparisons of input sequences which can lead to lower accuracy than methods that jointly model all sequences. However, Zhang et al. showed that ARGs inferred using threading can

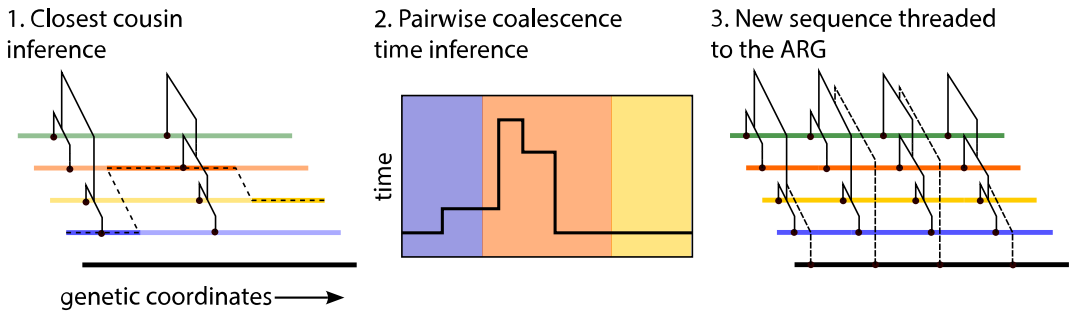


Figure 2.1: Diagram showing the steps involved in threading a new sample to the ARG. Inference of a genealogical closest cousin and pairwise coalescence give threading instructions which determine where new lineages are added to the ARG. Here, the sequence being threaded to a pre-existing ARG on 4 leaves is shown in black.

nevertheless achieve similar or greater accuracy to other methods and that the quality of the inferred ARG is robust to the ordering of samples [99].

Marginally at each site, we can think of threading as a variant of single-linkage clustering [61] applied to pairwise times to the most recent common ancestor (TMRCAs). However, because the threading order is predetermined, there is no need to keep track of a sequence of $N \times N$ pairwise TMRCA matrices in memory. Moreover, by exploiting patterns of shared haplotypes within the data, we can avoid ever evaluating the full $N \times N$ matrix, achieving a time and memory cost of approximately $O(N)$. By contrast, relying on marginal clustering algorithms (as done in, e.g. the ASMC-clust algorithm [99]) can induce a quadratic cost in time and memory, while also producing more fragmented ARGs [29, 84, 99].

2.1.3 Tree-based metrics of inference accuracy

To evaluate the accuracy of ARG inference methods, we need informative measures of similarity between a simulated true ARG and an inferred ARG. Most existing ARG metrics are based on direct comparison of marginal trees and in this chapter we review five such metrics. We introduce a novel ARG-based metric, the $\max r^2$-score, in chapter 4.1.

In what follows, we consider two ultrametric trees (phylogenies) T_1, T_2 on leaves $\{1, \ldots, N\}$. In its classical form, the Robinson-Foulds metric (RF) [75] is a tree-based symmetric difference metric. If $\mathcal{A}_1$ and $\mathcal{A}_2$ are the partitions of the leaf set defined by trees T_1 and T_2 respectively, then RF is defined as the cardinality of the symmetric difference $\mathcal{A}_1 \triangle \mathcal{A}_2$, often normalized to take a maximum value of 1. The RF only considers the topology of the input trees, ignoring branch lengths. It is both sensitive to off-by-one errors in leaf partitions and offers poor discrimination within tree space [49, 82]. However, it is still widely used as a measure of phylogenetic accuracy. The RF can be evaluated in time linear in N.

The tree total variation (TV) is a metric of probability distributions inspired by the coalescent. Our observed data at each site are bitsets $g \in \{0,1\}^N$, where we assume $g_i = 1$ if and only if sample i carries the non-ancestral allele at that site. Under the coalescent, the number of mutations that occur on a lineage corresponding to a bitset g follows a Poisson distribution with rate μl_g, where μ is the population mutation rate and l_g the length of the lineage corresponding to g, with $l_g = 0$ if the lineage does not exist. Assuming at most one mutation occurs per site, each ultrametric tree gives a probability distribution P_T over $\{0,1\}^N$, with

$$P_T(g) = \begin{cases} e^{-\mu \sum_g l_g} & \text{if } g = (0, \ldots, 0); \\ (1 - e^{\mu l_g}) e^{-\mu \sum_{h \neq g} l_h} & \text{otherwise.} \end{cases}$$

The tree total variation is defined as the total variation between the probability distributions P_{T_1} and P_{T_2},

$$\mathrm{TV}(T_1, T_2) \coloneqq \mathrm{TV}(P_{T_1}, P_{T_2}) = \sum_{g \in \{0,1\}^N} |P_{T_1}(g) - P_{T_2}(g)|,$$

where we have suppressed the mutation rate μ in the notation. The tree total variation metric introduced in [99] is the metric obtained in the limit $\mu \to \infty$, and in this limit it is closely related to branch length-aware extensions of the Robinson-Foulds [45]. Like the RF, the TV is also sensitive to off-by-one errors in the observed leaf partitions. Although the definition of the TV includes a sum over $\{0,1\}^N$, all but $O(N)$ terms in the summation are 0, allowing linear-time evaluation.

Although not explored here, the RF and TV have natural extensions to whole ARGs. In the case of RF, this is obtained by taking a symmetric difference of all leaf-set partitions throughout the entire ARG and for TV, by using not branch lengths, but branch *volumes*, defined as the height of a branch multiplied by the number of base-pairs it persists through the ARG. However, these can be costly to evaluate across large ARGs, and in practice estimates based on a subset of marginal trees provide a good approximation to the ARG-wide values.

Next, we introduce three vector-based tree metrics, which all proceed by mapping trees to a real vector space and then relying on the Euclidean metric in that space. First, the time to the most recent common ancestor (TMRCA) metric [99] compares the ultrametrics defined on the leaf space $\{1, \ldots, N\}$ by T_1 and T_2. Each such tree T gives a metric $d_T(i,j) = 2 \cdot \mathrm{TMRCA}_T(i,j)$ on leaves by computing the length of the shortest path between leaves i and j within the tree T. Let $\mathbf{d}(T) \coloneqq \big(d_T(1,2), d_T(1,3), \ldots, d_T(N-1,N)\big)$. The TMRCA metric then defined as the Euclidean distance

$$\mathrm{TMRCA}(T_1, T_2) = \|\mathbf{d}(T_1) - \mathbf{d}(T_2)\|.$$

The Kendall-Colijn metric (KC) is a weighted sum of discrete and continuous vector-based tree metrics [40]. First, we establish the time of the deepest coalescence event, $\mathrm{Root}(T)$, in each tree, and the length of each *pendant* branch, i.e., the distance from each leaf to its most recent ancestor, denoted p_i. For a fixed tree T, let $M_{i,j} \coloneqq \mathrm{Root}(T) - \mathrm{TMRCA}(i,j)$, we then construct a vector

$$\mathbf{M}(T) = \big(M_{1,2}, M_{1,3}, \ldots, M_{N-1,N}, p_1, \ldots, p_N\big).$$

The discrete KC is defined by letting $m_{i,j}$ represent the number of edges between the common ancestor of leaves i, j and the root. We let

$$\mathbf{m}(T) = \big(m_{1,2}, m_{1,3}, \ldots, m_{N-1,N}, 1, \ldots, 1\big),$$

padding with ones such that $\mathbf{m}(T)$ and $\mathbf{M}(T)$ have the same length. Finally, let $\mathbf{v}_\lambda(T) = (1-\lambda)\mathbf{m}(T) + \lambda\mathbf{M}(T)$ for $\lambda \in [0,1]$ and define the KC as the Euclidean distance between two such vectors,

$$\mathrm{KC}(T_1, T_2) \coloneqq \|\mathbf{v}_\lambda(T_1) - \mathbf{v}_\lambda(T_2)\|_2.$$

In our experiments we only consider the case $\lambda = 0$. It has been observed that the KC is highly dependent on tree balance, and can provide poor discrimination between trees [82].

Finally, for a fixed coalescence tree T, denote by e_{ij} the number of leaves subtended by the most recent common ancestor of leaves i and j, and let $\mathbf{e}(T) \coloneqq (e_{1,2}, e_{1,3}, \ldots, e_{N-1,N})$. The the split-size vector (SV) metric is then defined as

$$\mathrm{SV}(T_1, T_2) = \|\mathbf{e}(T_1) - \mathbf{e}(T_2)\|_2.$$

The vector-based metrics TMRCA, KC, and SV all take time quadratic in N to evaluate and we thus only include them in our results for our smaller ($N \leq 4{,}000$) simulations. Another, more subtle, drawback of the vector-based metrics is their root bias. The quantities considered by all three metrics rely on the quantity $\mathrm{TMRCA}(i, j)$ for all pairs $1 \leq i < j \leq N$. However, as N grows, around half of all elements in the tree-derived vectors refer the root. This observation is formalized in the following proposition.

Proposition 1. *The probability that two randomly chosen leaves in a coalescent tree on N leaves share a most recent common ancestor at the root is $\frac{N-1}{N}\frac{2^{N-2}}{2^{N-1}-1}$.*

Proof. The root node defines a partition of the N leaves into non-empty sets A, B. Under the coalescent, all tree topologies are equally likely, so

$$P(|A| = k) = P(|B| = k) = \frac{\binom{N}{k}}{2^N - 2},$$

for $k = 1, \ldots, N_1$. Now pick two leaves at random. They will share a most recent common ancestor at the root if and only if one belongs to A and the other to B. Call this event AB. Then,

$$\begin{aligned}
P(AB) &= \sum_{k=1}^{N-1} P(AB \,|\, |A| = k) P(|A| = k) \\
&= \sum_{k=1}^{N-1} \frac{2k(N-k)}{N^2} \frac{\binom{N}{k}}{2^N - 2} \\
&= \frac{1}{N^2} \frac{2}{2^N - 2} \sum_{k=0}^{N} k(N-k) \binom{N}{k} \\
&= \frac{1}{N^2} \frac{1}{2^{N-1} - 1} 2^{N-2} (N-1) N \\
&= \frac{N-1}{N} \frac{2^{N-2}}{2^{N-1} - 1},
\end{aligned}$$

Accuracy metric	**Topology-only**	**Root bias**	**Evaluation complexity**	**Comments**	**Citation**
Robinson-Foulds (RF)	Yes	No	$O(N)$	Sensitive to off-by-one errors	[75]
Total variation (TV)	No	No	$O(N)$	Sensitive to off-by-one errors	[99]
Kendall-Colijn (KC)	No	Yes	$O(N^2)$	Sensitive to tree balance	[40]
TMRCA MSE	No	Yes	$O(N^2)$	Strong root bias	[99]
Split-size vector (SV)	Yes	Yes	$O(N^2)$		[82]

Table 2.1: Table to test captions and labels.

as claimed. □

Informally, Proposition 1 implies that TMRCA, KC and SV are all quadratically biased towards the root, and that as N grows, accuracy of recent coalescence events has a diminishingly small impact on their value.

By contrast, RF is likely to be poor estimator of deep-time inference accuracy, instead giving a measure of topological accuracy in recent time. This follows from the facts that all tree branches are equally weighed in the evaluation of RF, and that most of the branches represent recent coalescence events. The TV attempts to balance recent and deep time by weighing branches by their length. However, the TV is also sensitive to off-by-one errors, whereby a single misplaced leaf or clade may dramatically increase the observed metric. We summarize some important properties of the five accuracy metrics in Table 2.1.

2.2 The positional Burrows-Wheeler transform

The positional Burrows-Wheeler transform (PBWT) [17] is a binary matrix transformation that, when applied to a set of haplotypes, allows efficient querying of long matches. It has been successfully applied to genotype imputation [52, 76] and the detection of IBD segments [63, 102] and is a fundamental tool in the ARG inference methods we introduce in Chapter 3. In what follows, we briefly describe the PBWT using notation and definitions from [78].

For a set of N haplotypes $x_1, \ldots, x_N$ across M sites, let $x_n[k_1, k_2)$ denote the substring starting at $x[k_1]$ and ending at $x[k_2 - 1]$. The PBWT consists of four $M \times N$ matrices. The *prefix matrix* a contains the lexicographical sortings a_k of the reversed prefixes $x[0, k)$ for $k = 1, \ldots, M$. The prefix matrix will implicitly group sequences that share long matches together.

The *divergence matrix* d tracks maximal matches between reversed prefixes that are adjacent in the prefix matrix. Letting $y_i^k = x_{a_k[i]}$ be the i-th sequence in the k-th ordering, $d_k[i]$ is the smallest number $j \geq 0$ such that $y_i^k[j, k) = y_{i-1}^k[j, k)$. Thus, while the prefix matrix clusters sequences with long local matches, the divergence matrix gives us the exact lengths of those matches.

The final two matrices u, v together form the *extension function* w. The matrix u contains, at $u_k[i]$, the index in prefix sorting a_{k+1} of the first sequence below $a_k[i]$ containing a 0 at k. That is, if $u_k[i] = g$, then $a_{k+1}[g] = a_k[j]$, where $j = \min\left\{i \leq o < N \mid y_o^k[k] = 0\right\}$. The matrix v contains the same information for the fixed value 1 instead of 0. The extension function $w : \{1, \ldots, N\} \times \{0, 1\} \to \{1, \ldots, N\}$ is then defined at site k as

$$w_k(i, h) = \begin{cases} u_k[i], & \text{if } h = 0; \\ v_k[i], & \text{if } h = 1. \end{cases}$$

While the prefix array gives local clusters of similar sequences and the divergence array measures the length of those matches, the extension function allows us to efficiently extend sets of matching sequences along the genome.

2.2.1 The dynamic PBWT

While the PBWT data structure allows efficient queries for long matches, it takes $O(NM)$ time and memory to construct, which can be prohibitively expensive. Moreover, to exploit the PBWT for threading new sequences to an ARG, we require for each sample n the PBWT of sequences $1, \ldots, n-1$, giving, for the complete inference process, a time complexity of $O(N^2M)$. To address this quadratic term, we can use the dynamic-PBWT (d-PBWT) [78], a variant which allows for $O(M)$ time insertion and deletion of sequences from the PBWT data structure.

The d-PBWT is constructed as a sequence of M linked lists of length N, allowing dynamic operations that are not feasible with the array-based PBWT data structure. The i-th element of the k-th list is a node object n, that has prefix, divergence and extension properties satisfying

$$n.\text{ID} = a_k[i], \quad n.d = d_k[i], \quad n.u = u_k[i], \quad n.v = v_k[i].$$

Moreover, it has pointers to the nodes above and below it, denoted $n.\text{above}$ and $n.\text{below}$.

The d-PBWT extends to the most widely used matching algorithms applied to the PBWT [78]. However, the linked list structure does not allow for the same efficient indexing operations as the array-based version, as traversing the k-th linked list down to its i-th element is an $O(N)$ time operation. A comparable operation in the array-based data structure simply reads from the desired array in constant time.

2.3 The Li-Stephens haplotype matching model

The Li-Stephens model [48] is a foundational model of linkage disequilibrium and haplotype matching in statistical and population genetics, with applications in genotype phasing and imputation [8, 9, 27, 76], inference of recombination rates and local ancestry [71, 94] and many other fields [83]. It is a hidden Markov model (HMM) [72] that accepts an input genotype and attempts to model it as a mosaic of a set of reference haplotypes, minimizing the number of heterozygous sites and recombination events. When used with standard HMM algorithms, the output can variously be an optimal path through the reference panel, a posterior probability over individual reference sequences, or path probabilities.

In what follows, suppose $\mathcal{H}$ is an $M \times N$ binary genotype matrix for a phased reference panel, with $\mathcal{H}_{i,j}$ representing the allele carried by sample j at site i. Let g be the M-dimensional binary genotype vector we wish to trace through the panel. A *path* through the reference panel is a string $\pi = \pi_1 \cdots \pi_M$, with each $\pi_n \in \{1, \ldots, N\}$.

In the standard HMM setting, the ingredients required to compute the probability of a path π are an *initial probability* p for π_0, *transition probabilities* $p(\pi_{i+1}|\pi_i)$, and *emission probabilities* $p(\mathcal{H}_{i,\pi_i}|g_i)$. The initial probability distribution is usually taken to be uniform over $\{1, \ldots, N\}$, the transition probabilities are based on genetic distance between adjacent sites, and the emission probabilities are based on population-wide mutation rates.

The exact parameters can be optimized using numerical methods, but good estimates can also be obtained using coalescent modelling [33, 48]. Usually, these estimates rely on the assumption that mutation and recombination events can be modelled as Poisson processes. Under a demography of constant size N_e and a per-site, per-generation mutation rate of μ, we can approximate the probability of observing a mismatch in a region of length l base-pairs and obtain emission prob-

abilities

$$p(h|g_i) \begin{cases} e^{-2l\mu \frac{N_e}{N}} & \text{if } h = g_i; \\ 1 - e^{-2l\mu \frac{N_e}{N}} & \text{if } h = 1 - g_i. \end{cases}$$

independent of i. This approximation follows from the fact that the input sequence has expected coalescence time N_e/N into the panel and holds well if l is sufficiently small and N is sufficiently large. Here, both N_e and N count haplotypes, not diplotypes. Similarly, we can obtain transition probabilities

$$p(\pi_{i+1} = \beta | \pi_i = \alpha) = \begin{cases} (1 - r_i) + \frac{r_i}{N}, & \text{if } \beta = \alpha; \\ r_i/N, & \text{otherwise,} \end{cases}$$

where r_i is the probability of recombining between sites i and $i + 1$, given by

$$r_i = 1 - e^{-2d_i \frac{N_e}{N}}$$

and d_i is the distance between sites i and $i + 1$ measured in Morgans.

The two algorithms most commonly applied to this model are the forward-backward algorithm and the Viterbi algorithm [72]. The forward-backward algorithm outputs a full posterior matrix $P \in [0, 1]^{N \times M}$, where $P_{i,j}$ is the probability of haplotype i being the best match with g at site j, equivalent to averaging the probabilities of every path π through $\mathcal{H}$. The Viterbi algorithm outputs a single path π of highest probability, satisfying $P(\pi) = \max_{\gamma \in \{1,\ldots,N\}^M} P(\gamma)$, sometimes called the *Viterbi path*.

The Li-Stephens also has a diploid variant, in which the input genotype $g \in \{0, 1, 2\}^M$ is unphased. Here, a solution path $\pi = \pi_1 \cdots \pi_N$ consists of pairs of states, with each $\pi_n \in \{1, \ldots, N\} \times \{1, \ldots, N\}$. In the diploid case, the transition and emission probabilities are computed per haplotype identically to the above haploid algorithm across the N^2-sized search space.

2.3.1 FastLS

While the Li-Stephens allows for efficient modelling of linkage disequilibrium using the forward-backward and Viterbi algorithms, their $O(NM)$ time and memory complexity can become too expensive. In [55], Lunter introduced fastLS, an algorithm that under some minor simplifications can find a Viterbi path through an $N \times M$ panel in $O(M)$ time, independent of sample size. This improvement is achieved by exploiting the PBWT of the panel and extends to the diploid Li-Stephens as well as the haploid.

The algorithm is a branch-and-bound algorithm [46], using properties of the Li-Stephens model along with the PBWT extension function to significantly reduce the path search space. We do not give a complete description of the algorithm here. However, our derivation of one of our ARG inference algorithms, Threads-fastLS, in Chapter 3.2, closely follows Lunter's original algorithm and provides an introduction to its most important ideas.

Although fastLS achieves excellent scalability in inference time, it has several drawbacks. First, although inference can be performed in $O(M)$ time, construction of the PBWT takes $O(NM)$ time, so unless done a priori, the PBWT construction time may overwhelm inference time for large panels. If the panel is frequently used, the PBWT can be stored on disk and loaded into memory for inference. However this will induce an $O(MN)$ memory footprint. For large reference panels, neither option is feasible. Second, while fastLS is an excellent Viterbi solver, it cannot give a full posterior like the forward-backward algorithm, which can lead to a loss of accuracy in many applications. Third, the modelling simplifications that fastLS adopts mean it cannot exploit prior information about recombination rates. The algorithms we introduce in Chapters 3 and 6 address all of these concerns either in part or whole, while also enjoying the benefits of fast inference using a fastLS-like branch-and-bound approach.

Chapter 3

Threads

In this chapter we present two complementary approaches to ARG-inference, Threads and Threads-fastLS. We give a detailed description of the algorithms with proofs of their correctness in Chapters 3.1 through 3.3. Chapter 3.4 makes a brief digression away from ARG inference to describe how our algorithmic advances can be used to scale up diploid Li-Stephens models. In Chapter 3.5 we discuss practical challenges of ARG inference and provide extensions and heuristics to the core algorithms that significantly increase accuracy.

3.1 The Threads algorithm

In this chapter we describe Threads, a highly scalable ARG inference algorithm. Threads takes as input a set of phased genotypes and outputs as set of "threading instructions" for each sample. More precisely, for a genomic region $[L_1, L_2]$ and a set of genotypes indexed $1, \ldots, N$, Threads infers for each $n \leq N$ a map $[L_1, L_2] \to \mathbb{R}_+ \times \{1, \ldots, n-1\}$, representing, for each site, the coalescence time of sample n and a threading target from among the previously threaded samples.

The threading instructions are obtained in two steps, first inferring the closest genealogical relative at each site and then the coalescence time. In what follows,

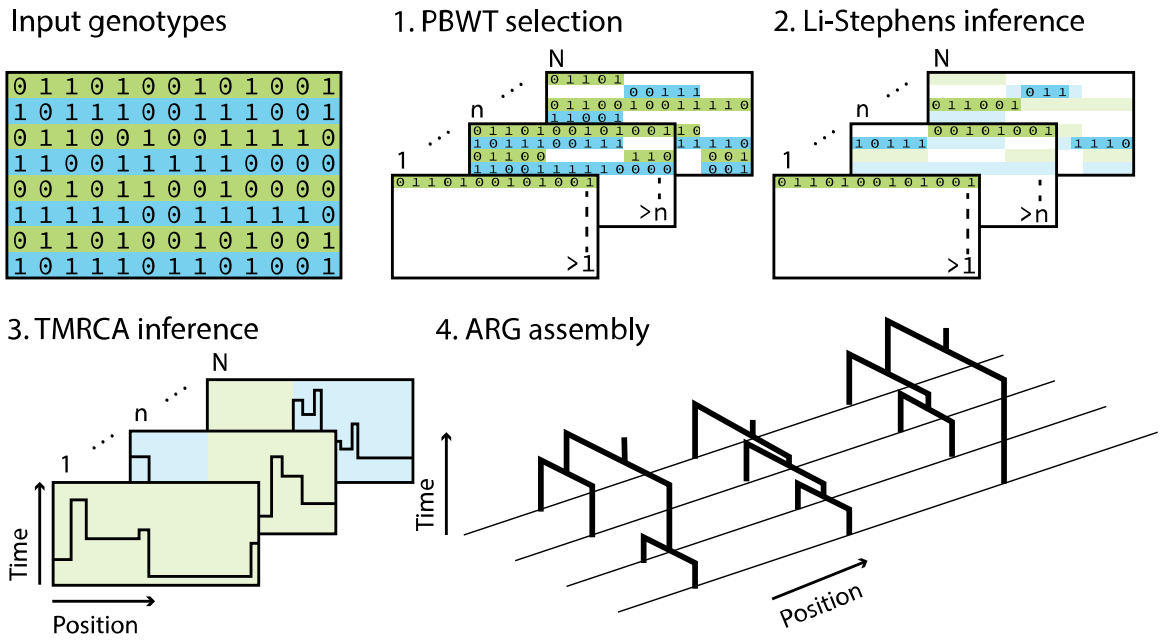


Figure 3.1: Diagram showing the main steps of the Threads algorithm. Threading instructions are inferred in steps 1 through 3. These uniquely determine an ARG which is assembled in the fourth and final step.

we will refer to the closest genealogical relative both as the "closest cousin" and the "threading target". Closest cousin inference is itself split into two sub-routines, a haplotype matching algorithm based on the positional Burrows-Wheeler transform (PBWT) [17] followed by a memory-efficient and perfectly parallelizable implementation of the Viterbi algorithm under the Li-Stephens model [48]. Coalescence times are estimated using a segment-based dating method described in Chapter 3.1.3. A diagram of the main algorithm steps is provided in Figure 3.1 and in what follows we describe each step in detail and provide pseudocode for the relevant subroutines.

3.1.1 Haplotype matching with the PBWT

The Li-Stephens model is a useful tool for closest cousin inference but a direct application to large-scale ARG inference by threading quickly becomes computationally

unfeasible. As the sample size grows, each sample n must search through $n-1$ previously threaded sequences for close matches, resulting in a time complexity of $O(MN^2)$ for the whole ARG inference process, where M is the number of sites and N the total number of samples.

To reduce the search space for the Li-Stephens algorithm, Threads first constructs a PBWT of the input genotypes [17] to find a small set of candidate matches for each sample being threaded. This strategy is similar to haplotype matching methods employed by modern phasing and imputation algorithms to reduce the state space of a Li-Stephens model but with some crucial differences [8, 9, 27, 76].

First, Threads partitions the input region into small chunks and estimates candidate haplotype matches independently within each chunk. This allows the algorithm to maintain a locally optmized Li-Stephens state space which is important for inference over large genomic regions. The default genomic chunk size for this step is 0.5cM. This preliminary subdivision step is novel to Threads and we denote the number of genomic chunks thus obtained by M_{chunk}.

The haplotype matching then proceeds by sorting the input haplotypes at each site into a PBWT prefix array. At regular intervals, the prefix array is queried for the L neighbours around each sequence n from among the sequences $1, \ldots, n-1$, keeping track of how many times each sequence has been chosen as a candidate match. More precisely, if sequence n has prefix index i at site j, i.e. $a_{i,j} = n$, then the first $L/2$ sequences $a_{k,j}$ immediately above and below $a_{i,j}$ in the prefix array satisfying $a_{k,j} < n$ are considered as candidate matches. If fewer than $L/2$ such indices $k > i$ or $k < i$ are found then the search window is expanded in the opposite direction until L matches have been found or until all sequences have been considered. The default interval between these queries is 0.01cM and the default window size is $L = 4$. We denote by M_{query} the number of sites queried for haplotype matches. Threads does not keep track of the PBWT divergence array. Instead it

heuristically assumes neighbouring sequences in the PBWT prefix array share long tracts that are identical by state. A similar heuristic is employed by IMPUTE5 [76] for efficient genotype imputation.

The querying operation is implemented by sequentially inserting the prefix-array index of each sequence $n = 1, \ldots, N$ at the query site into a red-black tree, as implemented by the C++ container `std::set`, and querying the L-sized neighbourhood around the insertion index in the tree. The time complexity for the complete matching step thus becomes $O(MN + M_{\text{query}} N \log N)$. Pseudocode is provided in Algorithm 1. In practice, we do not include singleton variants in this step of the analysis as they tend to have higher phasing and genotyping errors than other variants [8, 27] and may interfere with identification of long stretches of identical-by-state regions.

Finally, once all sites within a chunk have been processed, the candidate sequences are filtered by the number of times they were matched within the chunk. By default, we discard any matches observed in fewer than 4 queries. If no matches remain after this filter, the threshold is decreased by 1 until at least one sequence is matched. For $n < 100$, we include all sequences observed in the querying step and for $n \geq 10,000$, we double the number of matches required for inclusion, to heuristically limit the number of candidates and save on memory usage. Moreover, we include the top 4 matched sequences from adjacent chunks, to account for the possibility that the chunk breakpoint lies close to a recombination event.

At no point is the full PBWT or the full genotype matrix stored in memory. Instead, genotypes are streamed through the algorithm in small chunks, only keeping track of the match candidates for each sequence. For each sample, the maximum possible number of matches found per chunk is $LM_{\text{query}}/M_{\text{chunks}}$, although in practice it is usually much lower. Thus, the complete haplotype matching step can be

performed using only a single pass through the data, with a total memory footprint of $O(NLM_{\text{query}})$.

3.1.2 A memory-efficient Li-Stephens path

In this section we present the memory-efficient Viterbi algorithm used in Threads for inference of threading targets. Recall from Chapter 2.3 that for a reference set $\mathcal{H}$ of N haplotypes over M sites and a query haplotype g over the same sites, the Li-Stephens model [48] uses prior information on mutation and recombination rates in the reference sample to model g as a mosaic of haplotypes from $\mathcal{H}$, minimizing the number of allelic mismatches and recombinations. This is achieved by assigning a probability $P(\pi)$ to each path $\pi \in \{1, \ldots, N\}^M$ through the reference panel. A path through $\mathcal{H}$ of maximum probability can be found in $O(NM)$ time and space using the Viterbi algorithm and we call such a path a *Viterbi path*.

In classical implementations of the Viterbi algorithm, a full per-site, per-sample matrix of probabilities is constructed before a traceback step is performed to extract a best path. This approach quickly becomes computationally prohibitive, as genetic data sets can contain hundreds of thousands of samples and millions of sites. The number of haplotypes under consideration can be reduced significantly by applying steps such as the haplotype matching described in Chapter 3.1.1, but even then the memory requirements for long genomic tracts are high. In what follows we describe an algorithm that builds a Viterbi path using $O(NM)$ runtime and, on average, $O(N)$ memory. A detailed discussion of the exact Li-Stephens parameters used in Threads is given in Chapter 3.5.1. Here, we focus only on the memory-efficient Viterbi algorithm.

Our implementation of a memory-efficient Viterbi is a branch-and-bound algorithm based on two key observations. First, recombination events in the output of the Viterbi are rare, especially for large N, resulting in paths that consist of few

Algorithm 1 Threads haplotype matching for a single genotype chunk

Input: A genotype matrix $X \in \{0,1\}^{M \times N}$ and query sites Q.
Output: For each $n = 1, \ldots, N$, haplotype matches from $\{1, \ldots, n-1\}$.

$C \Leftarrow [\{\}, \ldots, \{\}]$ ▷ Empty multi-set for each $n = 0, \ldots, N-1$
$sort_0 \Leftarrow [0, \ldots, N-1]$
$sort_1 \Leftarrow [0, \ldots, N-1]$
for $i = 0, \ldots, M-1$ **do**
    $ac \Leftarrow \sum_j X[i,j]$
    $c_0, c_1 \Leftarrow 0$
    for $j = 0, \ldots, N-1$ **do** ▷ Compute the next PBWT sorting
        if $X[i,j] = 1$ **then**
            $sort_1[N - ac + c_1] \Leftarrow sort_0[i]$
            c_1++
        else
            $sort_1[c_0] \Leftarrow sort_0[i]$
            c_0++
        end if
    end for
    $sort_0 \Leftarrow sort_1$
    if $i \in Q$ **then** ▷ Query site for matches
        $p = [\,]$ ▷ $p[j]$ is the index of sample j in $sort_0$
        for $j = 0, \ldots, N-1$ **do**
            $p[sort_0[j]] \Leftarrow j$
        end for
        Initialize an empty red-black tree t
        t.insert($p[0]$)
        for $j = 1, \ldots, N-1$ **do**
            $k \Leftarrow t$.insert($p[j]$) ▷ k is the insertion index of $p[j]$ in t
            $matches \Leftarrow L$-sized neighborhood around k in t
            for $l \in matches$ **do**
                $C[j]$.insert($s_0[l]$)
            end for
        end for
    end if
end for
return C

but long segments. In Viterbi paths obtained using a reference panel of 2,251 unrelated sequences from the 1000 Genomes Project [3, 11, 19], we find that the average segment length is over 200 kilobases, and in genotyping array data from unrelated white British samples in the UK Biobank (N=337,464), the length of an average segment is well over a megabase. This observation allows us to store whole Viterbi paths using a negligible amount of memory, by only noting the segment breakpoints and the associated threading target. This observation is of key importance for the ARG-based compression approach discussed in Chapter 5.

Second, under the Li-Stephens model, recombination events are perfectly symmetric, that is,

$$p(\pi_{i+1} = \beta | \pi_i = \alpha, \text{ recombination between } i \text{ and } i+1 \text{ }) = \frac{1}{N}.$$

This property allows us to drastically reduce the search space for possible Viterbi paths. We formalize this property in the following Proposition.

Proposition 2. *Suppose $\pi^{(i)}$ is a Viterbi path for the subset of the panel $\mathcal{H}$ containing sites 1 through i. If there exists a Viterbi path through $\mathcal{H}$ that recombines between sites i and $i+1$, there exists such a path π satisfying $\pi_i = \pi_i^{(i)}$.*

More succinctly, at site i we need only consider recombination events from a sequence of highest probability given all observations up to i. We now describe the Threads-Viterbi algorithm in detail. The algorithm consists of alternating "branch" and "bound" steps, with a single "traceback" step at the end. Branch-and-bound implementations for the Li-Stephens model have been previously implemented in Eagle2 [52, 53] and fastLS [55]. Our implementation is partly based on algorithmic innovations presented in fastLS, which is discussed in more detail in Chapter 3.2. The output of the branch-and-bound steps is a set Ω of *path segments* ω, each consisting of

- a start site $m_\omega \in \{1, \ldots, M\}$,
- a threading target $n_\omega \in \{1, \ldots, N\}$,
- if $m_\omega > 0$, a *traceback* segment $\omega' \in \Omega$ satisfying $m_{\omega'} < m_\omega$.

A full path through $\mathcal{H}$ can thus be constructed by starting at any path segment and following the traceback segments until a segment with $m_\omega = 0$ is found. We denote a path ending with ω as $\mathcal{P}(\omega)$ and by $s(\omega)$ the negative log-likelihood (penalty) of that path. The penalty for the partial path up to site m is denoted by $s(\omega, m)$.

Moreover, the set Ω contains exactly N path segments $\omega_1, \ldots, \omega_N$ with $n_{\omega_n} = n$ classified as *active*, with the rest denoted *inactive*. If the active segments all satisfy the property that $\mathcal{P}(\omega_n)$ is a Li-Stephens-optimal path through $\mathcal{H}$ conditional on the path ending at n, the segment set is said to be *complete*. In this case, the path ω_n with minimum penalty from among the active segments is a Viterbi path through $\mathcal{H}$.

Given a genotype vector h_{m+1} for site $m+1$ and a complete segment set $\Omega^{(m)}$ for the partial panel $\mathcal{H}^{(m)}$ consisting of sites 1 through m, the branching step constructs a segment set $\Omega^{(m+1)}$ containing $\Omega^{(m)}$ that is complete for the partial panel $\mathcal{H}^{(m+1)}$. By contrast, the bounding step takes a complete segment set $\Omega^{(m)}$ and gives a smaller set $\Omega^{(m)}_* \subseteq \Omega^{(m)}$ that is also complete. Following the terminology of [55], we can say each segment in $\Omega^{(m)}$ is *undercut* by a segment in $\Omega^{(m)}_*$. The set $\Omega^{(0)}$ containing only N active segments $\omega_1, \ldots, \omega_N$, all with starting site 0 and penalty 0 is trivially complete for $\mathcal{H}^{(0)}$, so by inductively applying the branching step for each site, we eventually arrive at a full Viterbi path. These steps are described in the following two theorems and in Figure 3.2.

Theorem 1 (Branch)**.** *Suppose $\Omega^{(m)}$ is a complete segment set for $\mathcal{H}^{(m)}$ and $\omega' \in \Omega^{(m)}$ a path segment such that $\mathcal{P}(\omega')$ is a Viterbi path through $\mathcal{H}^{(m)}$. Denote by ρ and ρ_c the penalty for recombination and no recombination between sites m and $m+1$, respectively.*

Finally, let

$$\Omega' \coloneqq \{\omega(m+1, n, \omega') \mid s(\omega_n, m) + \rho_c > s(\omega', m) + \rho\},$$

where ω_n are the active segments in $\Omega^{(m)}$. Then the segment set

$$\Omega^{(m+1)} \coloneqq \Omega^{(m)} \cup \Omega'$$

obtained by replacing the active segments of $\Omega^{(m)}$ with those from Ω' where possible is a complete segment set for $\mathcal{H}^{(m)}$.

Proof. Fix $n \in \{1, \ldots, N\}$ and let s_{m+1} be the penalty of a Li-Stephens-optimal path through $\mathcal{H}^{(m+1)}$ ending in n. Let μ be the match/mismatch penalty for the target haplotype g and h_n at $m+1$. Then, $s_{m+1} \leq \min(\{s(\omega_n) + \rho_c + \mu,\, s(\omega') + \rho + \mu\})$ by definition. If $s_{m+1} < s(\omega_n) + \rho_c + \mu$, then any path of penalty s_{m+1} ending at haplotype n must recombine between m and $m+1$. By Proposition 2, $s_{m+1} = s(\omega') + \rho + \mu$. Similarly, if $s_{m+1} < s(\omega') + \rho + \mu$, then any path of score s_{m+1} ending at haplotype n cannot recombine between sites m and $m+1$, giving $s_{m+1} = s(\omega_n) + \rho_c + \mu$. From these two observations it follows that $s_{m+1} = \min(\{s(\omega_n) + \rho_c + \mu,\, s(\omega') + \rho + \mu\})$, but this is exactly the penalty for the active state for haplotype n in $\Omega^{(m+1)}$. □

Starting from $\Omega^{(0)}$, sequential application of this branching step creates N different tree structures, with segments for nodes and traceback pointers as edges. At each step, the complete segment set grows by at most N elements, leading to a time and memory complexity of $O(NM)$. However, new information is only stored in case of inferred recombination events and by our earlier observation, recombination events in Viterbi paths through realistic genetic data are very rare. Thus, in practice, the memory usage grows substantially slower than $O(MN)$. In the case of fast growth in memory usage or other constraints, even this memory usage may not be computationally feasible. For this reason we include a step to prune the

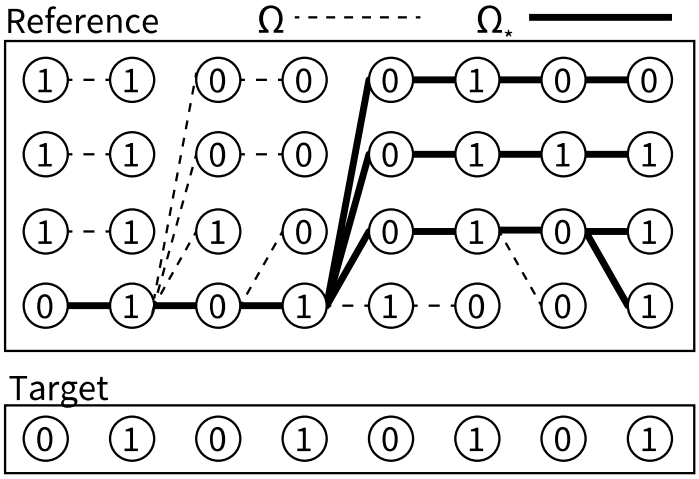


Figure 3.2: Diagram of the Threads-Viterbi algorithm. The full set of expanded segments Ω (both dashed and thick lines) is replaced with the subset Ω_* (thick lines). In this instance, $|\Omega| = 13$ and $|\Omega_*| = 5$. A unique path of lowest penalty given the target sequence can be traced back for each of the four reference sequences.

complete segment set and limit the number of segments that need to be stored in memory at each time.

Theorem 2 (Bound)**.** *Let Ω be a complete segment set through $\mathcal{H}$ with active segments $\omega_1, \ldots, \omega_N$. Let $\Omega_* \subseteq \Omega$ be the segment set $\bigcup_{n=1}^{N} \mathcal{P}(\omega_n)$ containing those segments that are contained in a traceback path from one of the active segments of Ω. Then Ω_* is also a complete segment set through $\mathcal{H}$.*

Proof. By Theorem 1, it is sufficient to expand only the active segments in the branching step, and replace them with new segments. This means the completeness condition only depends on the active segments. By the definition of the segment sets, and to allow for Viterbi paths to be traced back, we must also include those segments that lie on the traceback paths. The set Ω_* contains exactly those segments. □

We apply the bounding step at regular intervals throughout the inference process, either at fixed intervals or following a heuristic that limits the growth rate of the total number of segments. In the latter case, we fix a starting value (in Threads,

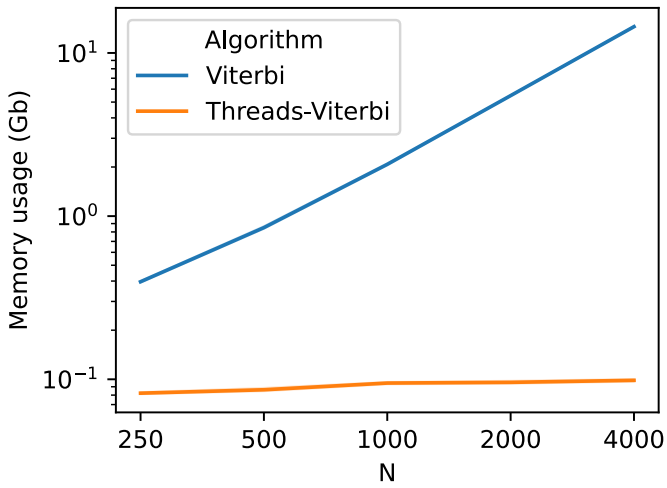


Figure 3.3: Memory usage of the Threads-Viterbi algorithm and the classical Viterbi algorithm on simulated genotype data. Sequences of 5MB length were simulated under a realistic European demography. Here, we apply the bounding step of Threads-Viterbi every 10 sites.

we use $B_0 := 10 \cdot N$), and then, if $|\Omega^{(m)}| > B_0$, we prune $\Omega^{(m)}$ to $\Omega_*^{(m)}$. If again $|\Omega^{(m')}| > B_0$ for some $m' > m$, we continue with a new threshold of $B_1 := 2B_0$ if $m' - m < 30$, otherwise we continue with the threshold of B_0. This procedure heuristically balances the number of pruning operations against the risk of memory spikes due to rapid accumulation of segments. At no point do we need to store either the full genotype matrix in memory or any $N \times M$ probability matrix. In sum, we obtain an algorithm with $O(NM)$ runtime and, on average, $O(N)$ memory usage. Pseudocode is provided in Algorithm 2 and a diagram of the algorithm is given in Figure 3.2. In practice the penalty function $s(\cdot)$ is never evaluated on a full or partial path, instead penalties are updated as attributes of the active segments. In Algorithm 2 we use the penalty function s only to simplify notation. A comparison between memory usage of the Threads-Viterbi algorithm and the classical Viterbi is shown in Figure 3.3.

The algorithm we have described thus far finds the Viterbi path through a single genotype matrix of dimensions $N \times M$. To compute a full set of threading instruc-

Algorithm 2 Threads-Viterbi

Input: Genotype matrix $X \in \{0,1\}^{M \times N}$, target genotype g, genetic distances c, physical distances d, and expected coalescence height t.
Output: A Viterbi path through X for g.

$\Omega \Leftarrow \{\omega_0, \ldots, \omega_{N-1}\}$ ▷ Initialize the segment set with empty segments
$\omega^* \Leftarrow \omega_0$ ▷ Initialize the best segment
for $m = 0, \ldots, M-1$ **do**
  $\mu_c \Leftarrow 2 * \mu * d_m * t$ ▷ Mismatch penalties for site m
  $\mu \Leftarrow -\log(1 - e^{-\mu_c})$
  $\rho_c \Leftarrow 0.02 * c_m * t$ ▷ Recombination penalties for site m
  $\rho \Leftarrow -\log(1 - e^{-\rho_c})$
  $\omega' \Leftarrow \omega^*$
  $s' \Leftarrow s(\omega^*, m-1) + \mu + \rho$
  for $i = 0, \ldots, N-1$ **do** ▷ Perform the branching step
    $\mu_i \Leftarrow \mu$ if $X_{m,i} \neq g_m$, else μ_c
    $s' \Leftarrow \min(s(\omega_i, m-1) + \rho_c, s(\omega^*, m-1) + \rho) + \mu_i$
    if $s(\omega_i, m-1) + \rho_c > s(\omega^*, m-1) + \rho$ **then**
      Add new segment $\omega(m, i, \omega^*)$ to Ω.
      Set the new segment to be the active segment ω_i for i.
    end if
    if $s' > s(\omega_i, m)$ **then**
      $\omega' \Leftarrow \omega_i$
      $s' \Leftarrow s(\omega_i, m)$
    end if
  end for
  $\omega^* \Leftarrow \omega'$
  if $|\Omega| > B$ **then** ▷ Perform the bounding step if $|\Omega|$ exceeds a threshold
    $\Omega' \Leftarrow \{\omega_0, \ldots, \omega_{N-1}\}$
    for $i = 0, \ldots, N-1$ **do**
      $\omega \Leftarrow \omega_i$
      while ω has starting site > 0 **do**
        $\omega \Leftarrow \text{traceback}(\omega)$
        Add ω to Ω'
      end while
    end for
    $\Omega \Leftarrow \Omega'$
  end if
end for
$\mathcal{P} \Leftarrow \{\omega^*\}$ ▷ Final traceback step
$\omega \Leftarrow \omega^*$
while ω has starting site > 0 **do**
  $\omega \Leftarrow \text{traceback}(\omega)$
  Add ω to $\mathcal{P}$
end while
return $\mathcal{P}$

tions, however, we must infer such a path for each sample n using a reference panel built from samples $1, \ldots, n-1$. Using the haplotype matching step described in Chapter 3.1.1, we obtain a panel of size approximately $L \times M$ for each sample n, for some $L \ll N$. Crucially, the output of each hidden Markov model doesn't depend on any of the other hidden Markov models, meaning that we can safely run the N instances of Threads-Viterbi in parallel. In practice, we divide the N HMMs evenly among available CPU cores. This means we can complete the Viterbi inference step for all samples using only a single pass through the data per core.

Given L haplotype matches per sample, we can thus obtain threading targets in $O(MLN/N_{\text{CPU}})$ time and $O(LN)$ memory on average while streaming the genotypes from disk only once for the haplotype matching and once for the Threads-Viterbi algorithm. In comparison, $O(MN^2)$ time and memory are required to run the same inference without haplotype matching and using a classical HMM for the Li-Stephens inference. Even so, the Threads-Viterbi step is still the most computationally intensive part of the full Threads algorithm.

The Viterbi paths may be interpreted as, for each $n \leq N$, a map $[L_1, L_2] \to \{1, \ldots, n-1\}$, representing a threading target for each sample at each site. To complete the ARG inference, a third and final pass through the genotypes is required to assign coalescence times to each of these inferred threading targets. The next chapter describes and motivates the segment-based coalescent modelling used in the last step of Threads.

3.1.3 Dating path segments

In this chapter we describe the identity-by-descent (IBD) modelling used to assign coalescence times to each of the segments inferred by the Threads-Viterbi algorithm. A detailed discussion of the assumptions and simplifications of this modelling decision is provided in Chapter 3.1.4.

It is convenient to think of the pairwise Viterbi segments inferred in Chapter 3.1.2 as IBD segments, since whole ARGs may be assembled from a collection of such segments. Moreover, this approach allows us to assign coalescence times for each sequence in close to constant time. Together with the Threads-Viterbi step, our dating method forms a change-point detection algorithm, partitioning a stretch of genome into segments of constant height and threading target. In addition, we assume that the segment heights are independent random variables. Such models are sometimes called product partition models [5].

Recall that for a pair of sequences, we define a genomic region delimited by $[L_1, L_2]$ to be *identical by descent* (IBD) for those sequences if their most recent common ancestor is constant along the region. We say $[L_1, L_2]$ is an *IBD segment* if it cannot be extended in either direction while remaining IBD. IBD segments have a fixed *height*, or *age*, measured by the age of the common ancestor in generations.

Given the length and the number of heterozygous sites along the IBD segment, we can use these modelling assumptions to estimate the segment age. Moreover, if we have access to a demographic history for the two sequences as measured by effective population size $N_e(t)$ or, equivalently, by pairwise coalescence rates, we have a powerful prior estimate of the segment age. This is the approach taken by Threads. In the following, we derive the formulas used to estimate coalescence times.

To simplify notation in what follows, we fix an IBD segment of length l_{cM} in centiMorgans and l_{bp} in base-pairs. We assume a constant coalescence rate γ between the two sequences. For now, we assume a fixed mutation rate c and a constant coalescence rate γ. We then write $\mu = 2 \cdot c \cdot l_{\mathrm{bp}}$ and $\rho = 2 \cdot 0.01 \cdot l_{\mathrm{cM}}$, such that the number of heterozygous sites along the segment follows as Poisson distribution with rate μ and, given the starting coordinate of the segment, its length follows an exponential distribution with rate ρ. Here, we implicitly assume that the segment terminates at

the first recombination event, similar to the the widely used sequentially Markovian coalescent (SMC) [59]. We now derive several estimates of increasing complexity for the segment age t.

3.1.3.1 Maximum likelihood estimators

First, we ignore the coalescence rate γ and focus on likelihood functions for t using ρ and μ. Ignoring mutations for now, the length (in Morgans) of an IBD segment follows an exponential distribution with rate of $1/2t$ [68]. We obtain a likelihood $\ell(\rho|t) = 2te^{-t\rho}$, which has derivative $2e^{-t\rho}(1-t\rho)$ w.r.t. t and a maximum at $t = 1/\rho$. This estimator was used in [77] to date IBD segments.

To include mutations in our likelihood function, we denote the number of heterozygous sites on the segment as m and write

$$\ell(\rho, m|t, c) = \ell(\rho|t, c)\ell(m|\rho, t, c) = 2te^{-t\rho}\frac{(t\mu)^m}{m!}e^{-t\mu}.$$

Differentiating w.r.t. t, we find that this expression has a maximum at $t = \frac{m+1}{\rho+\mu}$. We call these estimators of t the *maximum likelihood estimators*. In what follows, we will drop the mutation rate c from our notation.

3.1.3.2 Bayesian estimators

We now place a prior π on t and compute $\mathbb{E}[t|\rho, m]$. Throughout, we will assume $\pi(t) \sim \mathrm{Exp}(\gamma)$. This prior is distinct from the commonly used length-biased prior $\underline{\pi}(t) \propto t\pi(t)$ [12, 18, 41, 68] which models coalescence at each site and must thus account for the inverse correlation between segment height and length. Here, we model IBD segments directly meaning we do not need to account for imbalanced site-based sampling.

As above, we start with a point estimate for t that ignores mutations. We observe that

$$p(t|\rho) \propto \pi(t)p(\rho|t) = 2\gamma t e^{-t(\rho+\gamma)},$$

an Erlang-2 distribution with rate $\rho + \gamma$ and mean $\frac{2}{\rho+\gamma}$ [18, 69]. Adding mutations, we can similarly write

$$p(t|\rho, m) \propto \pi(t)p(m|\rho, t)p(\rho|t) = 2\frac{\gamma\mu^m}{m!}t^{m+1}e^{-t(\rho+\mu+\gamma)}$$

to obtain an Erlang-$(m+1)$ distribution with rate $\rho + \mu + \gamma$ and mean $\frac{m+2}{\rho+\mu+\gamma}$. We call the estimators $\mathbb{E}[p(t|\rho, m)]$ and $\mathbb{E}[p(t|\rho)]$ the *Bayesian estimators*.

3.1.3.3 Piecewise-constant demography

We now extend the estimators above to account for variable demography. In problems dealing with variable demography, a standard simplification is to assume the effective population size $N_e(t)$ is piecewise constant [47, 70]. Under this assumption, we write $0 = T_0 < T_1 < \cdots < T_K = \infty$ and assume $N_e(t) \equiv N_e^{(k)}$ for all $t \in [T_k, T_{k+1})$. Similarly, we obtain piecewise-constant coalescence rates $\gamma_k = 1/N_e^{(k)}$ within each interval. Writing $\Delta_k = T_{k+1} - T_k$ for $k = 0, \ldots, K-1$, our prior becomes $\pi(t) = \gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j - (t-T_k)\gamma_k}$ for $t \in [T_k, T_{k+1})$. In this case there is no compact description of the distributions $p(t|\rho)$ and $p(t|\rho, m)$, so instead we compute the expectations directly.

As before, we start by ignoring the number of heterozygous sites. This gives us the ratio of integrals

$$\mathbb{E}[t|\rho] = \int_0^\infty tp(t|\rho)\mathrm{d}t = \frac{\int_0^\infty t\pi(t)p(\rho|t)\mathrm{d}t}{\int_0^\infty \pi(t)p(\rho|t)\mathrm{d}t}.$$

The numerator has solution

$$\begin{aligned}
\int_0^\infty t\pi(t)p(\rho|t)\mathrm{d}t &= \sum_{k=0}^{K-1}\int_{T_k}^{T_{k+1}} t\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j-(t-T_k)\gamma_k}2te^{-t\rho}\mathrm{d}t \\
&= \sum_{k=0}^{K-1}\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j+T_k\gamma_k}2\int_{T_k}^{T_{k+1}} t^2e^{-t(\rho+\gamma_k)}\mathrm{d}t \\
&= \sum_{k=0}^{K-1}\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j+T_k\gamma_k}\frac{4}{(\rho+\gamma_k)^3} \\
&\quad\times\Big[P(3,(\rho+\gamma_k)T_{k+1})-P(3,(\rho+\gamma_k)T_{k+1})\Big],
\end{aligned}$$

where

$$P(a,z)=\frac{\gamma(a,z)}{\Gamma(a)}=\frac{1}{\Gamma(a)}\int_0^z t^{a-1}e^{-t}\mathrm{d}t$$

is the normalized lower incomplete gamma function. The denominator has a similar expression, together giving the point estimate

$$\mathbb{E}[t|\rho]=\frac{\sum_{k=0}^{K-1}\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j+T_k\gamma_k}\frac{2}{(\rho+\gamma_k)^3}\Big[P\big(3,(\rho+\gamma_k)T_{k+1}\big)-P\big(3,(\rho+\gamma_k)T_{k+1}\big)\Big]}{\sum_{k=0}^{K-1}\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j+T_k\gamma_k}\frac{1}{(\rho+\gamma_k)^2}\Big[P\big(2,(\rho+\gamma_k)T_{k+1}\big)-P\big(2,(\rho+\gamma_k)T_{k+1}\big)\Big]}. \tag{3.1}$$

Using the same approach, it is straightforward to add mutations to the model to get the estimate

$$\mathbb{E}[t|\rho,m]=\frac{\sum_{k=0}^{K-1}\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j+T_k\gamma_k}\frac{\mu^m(m+2)}{\lambda_k^{m+3}}\Big[P\big(m+3,\lambda_kT_{k+1}\big)-P\big(m+3,\lambda_kT_k\big)\Big]}{\sum_{k=0}^{K-1}\gamma_k e^{-\sum_{j=0}^{k-1}\Delta_j\gamma_j+T_k\gamma_k}\frac{\mu^m}{\lambda_k^{m+2}}\Big[P\big(m+2,\lambda_kT_{k+1}\big)-P\big(m+2,\lambda_kT_k\big)\Big]}, \tag{3.2}$$

where $\lambda_k=\rho+\mu+\gamma_k$. This is the estimate used by Threads to infer coalescence times. In inference from sparse sequences such as genotyping arrays, we rely on the mutation-free formula in Equation 3.1.

3.1.4 The limits of identity-by-descent modelling

In this chapter we discuss several caveats to IBD-based dating of Viterbi segments. First of all, Viterbi segments do not always coincide with IBD segments. In Figure 3.4, we show in simple toy examples how Viterbi paths, even when inferred perfectly, may both over- and underestimate the length of the IBD segments they reflect. We can quantify the frequency of these events in a simple simulation as follows. Fix a leaf l in a simulated ARG, with most recent ancestor a_1 in a neighborhood left of some fixed site x and a different most recent ancestor a_2 to the right of x. The closest cousins of l at either side of x belong to different IBD segments, and we want to quantify to what extent these coincide with Viterbi segments. If the leaf sets subtended by a_1 and a_2 (excluding l) have a non-empty intersection, then this recombination event is not detectable by the Viterbi algorithm and we overestimate the length of the underlying IBD segment. This case is illustrated in Figure 3.4c. If the leaf sets are disjoint, but a_2 is an ancestor of a_1 in a neighborhood to the left of x or a_1 an ancestor of a_2 to the right of x, the IBD segments overlap and we underestimate their length by choosing a single breakpoint. We call these events "left" and "right" overlaps and give an illustration in Figures 3.4a and 3.4b. If none of these conditions are satisfied we say the Viterbi segment coincides with the IBD segments.

To quantify the frequency of each of these events we traverse simulated ARGs of varying size, noting recombination events directly above a fixed leaf. The results are presented in Figure 3.5. We first note that the rate of overestimates tends to 0 as N grows. This is due to the fact that overestimates can only occur when a recombination event results in the same genealogical closest cousin belonging to two adjacent IBD segments. As the ARG grows, the proportion of volume taken up by the clade subtending the fixed leaf l and its closest cousins shrinks, making this event less likely. We also note that the rate of Viterbi segments coinciding with

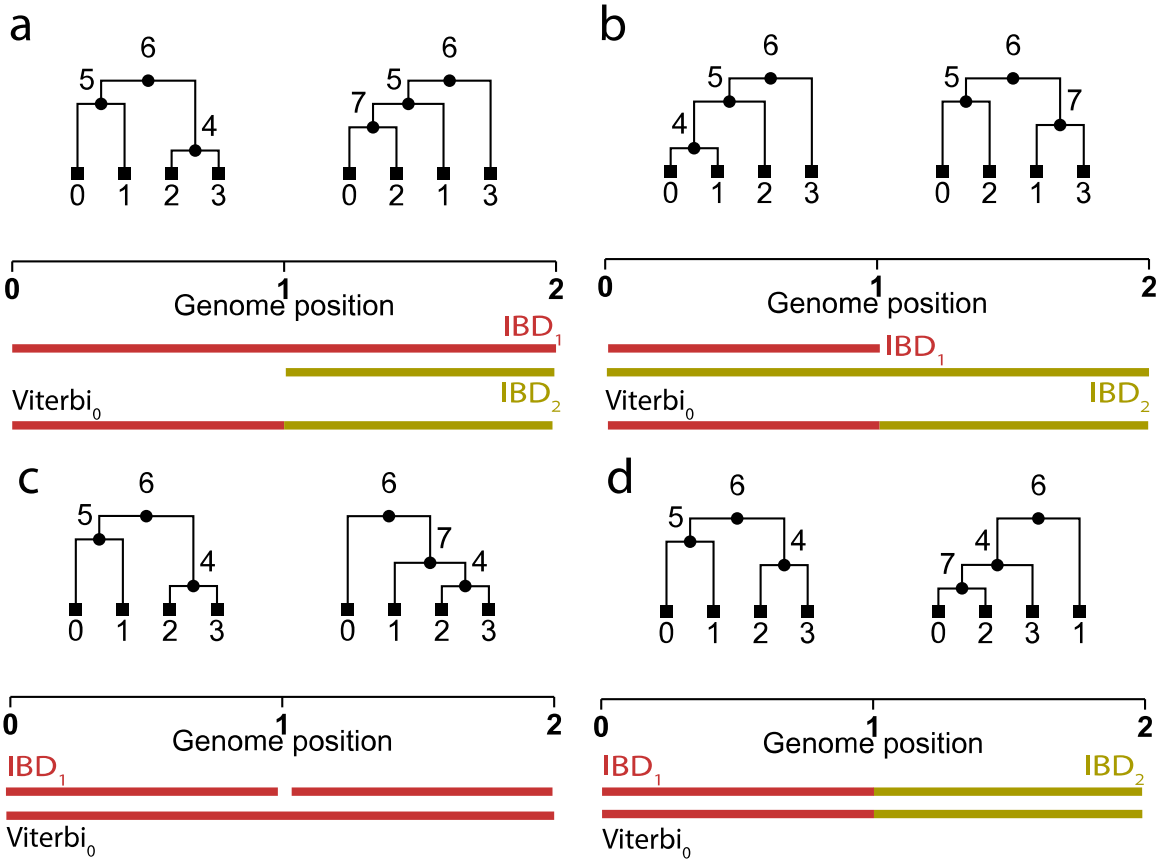


Figure 3.4: Illustration of how Viterbi segments might differ from IBD segments. Colored segments show Viterbi and IBD segments for the leaf labeled 0. In each panel, a unique Viterbi path can be found for leaf 0 through leaves 1 and 2. In panels **a** and **b**, the IBD segment subtended by the ancestor labeled 5 overlaps the next or the previous IBD segment. We call these events "right" and "left" overlaps, respectively. In panel **c**, a single, observed Viterbi segment overlaps two distinct IBD segments, overestimating the actual segment length. Finally, panel **d** shows a scenario where the IBD and Viterbi segments coincide.

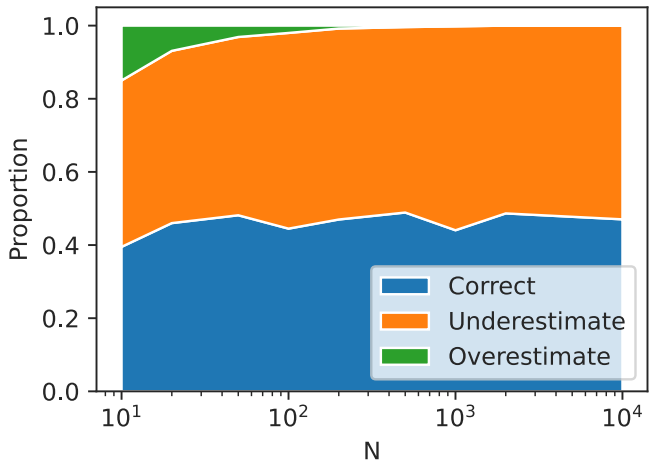


Figure 3.5: Concordance of IBD segments and Viterbi segments in simulated ARGs under a typical European demography. Here, we fix a leaf and trace it along the genome until the first change in most recent common ancestor. We repeat this experiment, varying the random seed and the total number of leaves (N) from 10 to $10{,}000$. We then count the number of times we observe each of the recombination patterns from Figure 3.4, labelled "underestimate" (Fig 3.4a-b), "overestimate" (Fig 3.4c) and "correct" (Fig 3.4d) to obtain the different probabilities.

IBD remains stable at a low rate of around 40%. Together these results suggest that IBD-based dating of Viterbi segments might underestimate the segment length and thus overestimate their height.

In practice we cannot infer perfect Viterbi segments from observed mutations. Instead, the algorithm smooths out underlying recombination events, resulting in segments that are longer and fewer in number than the IBD segments they model, especially so for smaller sample sizes. This is shown in simulation in Figure 3.6, where for small sample sizes, Threads outputs less than half the true number of segments for ARGs on tens of samples, with the difference shrinking as sample sizes approach the tens of thousands. Inferring fewer segments means we also infer longer segments, leading to underestimates of segment height. The interaction between overestimated segment lengths due to smoothing over imperfect data and the underestimates due to overlaps of the underlying IBD segments is complex. We

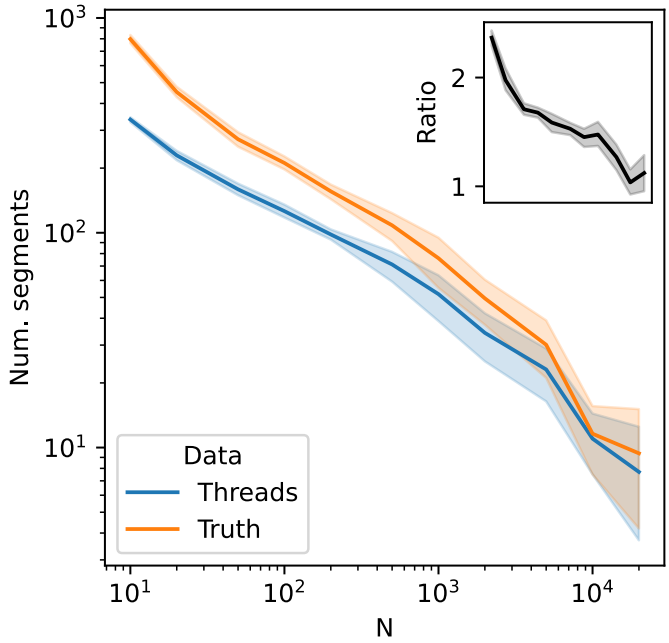


Figure 3.6: The number of segments inferred by Threads compared with the true number of underlying IBD segments. Here, we simulate ARGs for N ranging from 10 to $20{,}000$ under a typical European demography with fixed mutation rate $\mu = 1.65 \times 10^{-8}$ across a 20 mega-base region for 10 random seeds. We infer an ARG from the observed genotypes using Threads and count the number of distinct segments for the last sample to be threaded. We compare this with the true number of segments form the simulated ARGs for that sample.

discuss some ways to improve segment dating in the small sample regime in Chapter 3.5.3, but leave more sophisticated methods to model Viterbi segment lengths to future studies.

The second main caveat to pairwise IBD-based estimates of coalescence times is the coalescence time prior. In the segment age estimates derived in Chapter 3.1.3 it is assumed that the two sequences coalesce with a pairwise population-based rate of γ. However, this approach ignores the fact that at each site the IBD segments we aim to date represent the first coalescence event of the sequence being threaded with any of the $n-1$ sequences that are already present in the ARG. A more correct model of the first coalescence time of sample n comes from using a variable coa-

lescence rate of $\gamma(t) \cdot L_{n-1}(t)$, where $L_{n-1}(t)$ counts the number of lineages in the ARG on $n-1$ leaves at time t. The distribution of the random variable $L_{n-1}(t)$ has a closed-form expression obtained by marginalizing over its possible values, but takes $O(n)$ time to evaluate.

Using the exact prior, evaluating an estimator like the one in Equation 3.2 would take $O(CN)$ time, where C is the demography discretization factor, giving a time complexity of $O(CN^2)$ for the whole ARG-inference process, a prohibitively expensive amount of computation. In Threads, we only rely on the pairwise coalescence rate $\gamma(t)$ and leave the derivation of more appropriate, easily evaluated priors as an open problem.

In support of our use of $\gamma(t)$ as a coalescence prior, we evaluate the accuracy of estimation of the time to the most recent common ancestor (TMRCA) between the sample being threaded and its inferred closest cousin, and of the actual first coalescence time in Figure 3.7 for ARGs on 100, $1{,}000$, and $10{,}000$ samples. Although estimates of linear correlation (Pearson's r) are sensitive to outliers in our simulations and fluctuate significantly, we achieve robust rank correlation (Spearman's r). ARG inference through threading relies on the relative timing of segments to estimate topology and a high rank correlation of estimated coalescence times suggests good topological accuracy. Extensive accuracy benchmarking and comparisons with other ARG inference methods are provided in Chapter 4.

The final simplifying assumption that we discuss here is the independence of segment heights. In SMC frameworks it is standard to model the coalescence times of adjacent sites using a conditional distribution $p(t_1|t_2)$, where the adjacent coalescence times t_1 and t_2 are not independent and the literature contains many suggestions for the exact form of $p(t_1|t_2)$ [26, 56, 59]. Threads is not an SMC model, instead, it is more appropriate to think of Threads as a change point detection al-

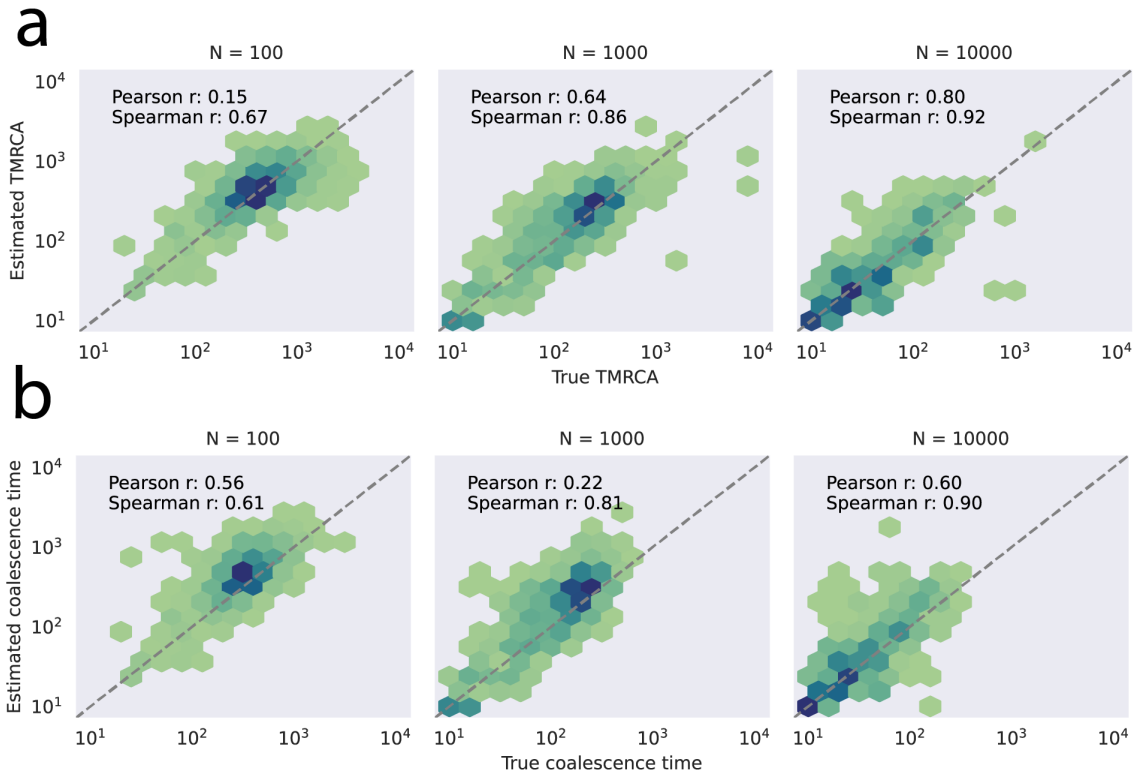


Figure 3.7: Accuracy of TMRCA inference using segment-based dating. Here, we simulate ARGs for N ranging from 100 to $10{,}000$ under a typical European demography with fixed mutation rate $\mu = 1.65 \times 10^{-8}$ across a 20 megabase region for 10 random seeds. We then infer ARGs for the observed genotypes using Threads. In **a**, we compare the coalescences times of the last 5 samples to be threaded to the ARG to the actual time to the most recent common ancestor (TMRCA) of that sample and the closest cousin as inferred by Threads. Coalescence times are observed at 20 evenly spaced points between coordinates $5{,}000{,}000$ and $15{,}000{,}000$, allowing sufficient burn-in for segment dating and closest cousin inference. In **b**, we compare the inferred coalescence times of the last 5 samples to be threaded with their actual coalescence times into the ARG at the same base-pair coordinates as in panel **a**.

gorithm, where the change points are inferred jointly in the Viterbi step, but their age inferred independently in a second step.

Each of the simplifications discussed above is in some way inconsistent with the coalescent with recombination, but brings substantial computational benefits. Applying segment-based dating instead of the site-based dating used in ARG-Needle [99] to estimate the age of threading segments means we can estimate coalescence times in time only loosely dependent on the number of sites M. This approach also leads to more sparsely represented ARGs, containing fewer edges and nodes, saving memory and disk space as we discuss further in Chapter 4.4.

The simplified demographic prior means coalescence time inference is independent of the underlying ARG, allowing us to estimate the height of each segment in $O(C)$ time, where C is the demography discretization factor. The independence of segment heights for different samples also allows us to parallelize computation in the same way as described in Chapter 3.1.2.

Finally, as we discuss in more detail in Chapter 4, Threads achieves comparable or better accuracy than other ARG inference methods across a range of metrics despite our simplifications while scaling to much larger sample sizes, suggesting that the computational benefits gained by applying our modelling simplifications do not result in a substantial loss of accuracy.

3.2 The Threads-fastLS algorithm

In this chapter, we describe a variant of the first two steps of Threads, i.e. the haplotype matching and Viterbi inference, based on fastLS [55] and the dynamic positional Burrows Wheeler (d-PBWT) transform [78]. We call this variant Threads-fastLS. Although Threads-fastLS is less computationally efficient than Threads, it is still an efficient ARG inference algorithm, and more importantly, supports sev-

eral useful operations that the Threads algorithm does not, such as arbitrary insertion and removal of samples from the ARG and PBWT-like queries for haplotype matches. In the following, we use the notation from Chapter 2.2, where a haplotype x on M sites has genotypes $x[k] \in \{0, 1\}$, for $k = 0, \ldots, M-1$ and where $x[k_1, k_2)$ represent the substring $x[k_1]x[k_1+1]\cdots x[k_2-1]$ of length $k_2 - k_1$.

Recall that the PBWT of a genotype matrix X consists of four arrays, the *prefix array* a, the *divergence array* d and the *extension arrays* u and v. In the d-PBWT, the vectors a_k, d_k, u_k, v_k are replaced with a single linked list starting at a "head" node of index k. The first and last elements of this linked list are denoted $\mathrm{heads}[k]$ and $\mathrm{tails}[k]$, respectively. Other elements of the linked list are nodes n that encode the data in the four PBWT arrays. The i-th node in the k-th linked list, $n_{i,k}$, satisfies the properties

$$n.\mathrm{ID} = a_k[i], \quad n.d = d_k[i], \quad n.u = u_k[i], \quad n.v = v_k[i].$$

Alternatively, we write $n.w[0] = n.u$ and $n.w[1] = n.v$ for the extension function. Insertion and deletion algorithms for the d-PBWT are given in Algorithms 1 and 5 of [78], respectively.

In the d-PBWT insertion algorithm, the extension function specifies the node above which the new haplotype should be inserted. We use the same mechanism to traverse the dynamic haplotype panel for Threads-fastLS. For a node n at coordinates (i, k) in the dynamic panel and a genotype $h \in \{0, 1\}$, we define *the extension of n by h* as the unique node

$$\mathrm{ext}(n_{i,k}, h) = \begin{cases} \mathrm{heads}[k].\mathrm{below}.w[1], & \text{if } h = 0 \text{ and } n.w[h] = \mathrm{tails}[i+1] \\ n.w[h], & \text{otherwise.} \end{cases}$$

In the former case, n lies below any of the sequences extending to 0, and we accordingly define the extension as the topmost sequence carrying a 1-allele.

To obtain a Li-Stephens Viterbi path through the dynamic panel, Threads-fastLS uses a branch-and-bound approach similar to fastLS [55]. The algorithm is implemented using a tree-like structure of states representing haplotype prefixes and traceback states representing recombination events that at each step in a single iteration across the genome are either extended to states at the next site or pruned. The state objects s carry a pointer $s.\mathrm{below}$ to a node in the haplotype panel, a score $s.\mathrm{score} \geq 0$ and a pointer to a previous state $s.\mathrm{traceback}$. To abbreviate, we write $s.b = s.\mathrm{below}$, $s.b.a = s.\mathrm{below.above}$ and $s.t = s.\mathrm{traceback}$.

For any input genotype $g \in \{0, 1\}$, we can obtain an extension of a state s at site k by applying $\mathrm{ext}(s.b, g)$. We say this extension *exists* and that the state s *extends to* $n \coloneqq \mathrm{ext}(s.b, g)$ *by* g if the genotypes for sequence $s.b.\mathrm{ID}$ exactly match $n.\mathrm{ID}$ on the interval going back to the previous recombination event. More precisely, we require $x_{s.b.a.\mathrm{ID}}[s.t.\mathrm{site}, k+1) = x_{n.a.\mathrm{ID}}[s.t.\mathrm{site}, k+1)$. In practice, we evaluate the extensibility of states by nodes by going through the following criteria, only directly comparing genotypes when all other conditions have been considered.

1. $n.a$ is not $\mathrm{heads}[k+1]$;
2. $x[k] = x_{n.a.\mathrm{ID}}[k]$;
3. One of the following holds:
   (a) $s.b.a.\mathrm{ID} = n.a.\mathrm{ID}$;
   (b) $x_{s.b.a.\mathrm{ID}}[s.t.\mathrm{site}, k) = x_{n.a.\mathrm{ID}}[s.t.\mathrm{site}, k)$.

Similar to Threads, recombination events represent branching events in the branch-and-bound algorithm. Whenever a state s is hit by a recombination event at site k, a new state s' is created with $s'.\mathrm{below} = \mathrm{tails}[k+1]$ and $s'.\mathrm{traceback} = s$. The

bounding conditions described in Theorem 3 guarantee that tracing back the state of lowest penalty gives a Viterbi path. We provide pseudocode for Threads-fastLS in Algorithms 3 and 4.

3.2.1 Proof of correctness

Throughout this section we will borrow terminology and notation from [55]. We represent paths through the reference panel with strings $\pi = a_1 r_r a_2 r_2 \cdots a_M r_M$, with $a_i \in \{0, 1\}$ representing genotypes and $r_i \in \{R, R_c\}$ representing recombination events (R) or no-recombination events (R_c) for all i. Using the criteria above, for any state s and any genotype $a \in \{0, 1\}$, we can check whether s is extensible by a and if so, whether the extension is unique. Starting from a state s_0 with $s_0.\text{below} = \text{tails}[0]$ and empty traceback, a sequence of the form $\pi_k = a_1 r_r a_2 r_2 \cdots a_k r_k$ defines a unique sequence of transformations of s_0 resulting in a new state s_k. We will denote by $\pi[k] = s_k.b.a.\text{ID}$ the haplotype index in the panel obtained by applying the extensions and recombination events $a_1 r_r a_2 r_2 \cdots a_k r_k$ to s_0.

For an input genotype x and a reference panel $x_1, \ldots, x_N$ our aim is to find a path π through the panel that minimizes the objective function

$$s_x(\pi) := \sum_{m=1}^{M} \mu_c[m] \cdot \delta(x[m], x_{\pi[m]}[m]) + \mu[m] \cdot (1 - \delta(x[m], (x_{\pi[m]}[m])) \\ + \rho_c[m]\delta(\pi[m+1], \pi[m]) + \rho[m](1 - \delta(\pi[m+1], \pi[m]))$$

where μ, μ_c are negative mismatch log-likelihoods, ρ, ρ_c are negative recombination log-likelihoods, and $\delta(\cdot, \cdot)$ is the Kronecker delta function. We write $s(\cdot) = s_x(\cdot)$ when only one input sequence x is being considered.

We say a set of paths S_l is a *full prefix set* for x on $[0, l)$ if for any sequence x' such that $x[0, l) = x'[0, l)$, there is is a path minimizing $s_{x'}(\cdot)$ whose $[0, l)$-prefix lies in S_l. We adopt the convention that each prefix ends with either the symbol R or

Algorithm 3 Threads-fastLS

Input: Haplotype $x \in \{0,1\}^M$, d-PBWT panel, recombination penalties ρ, ρ_c, mutation penalties μ, μ_c.

Output: A path π through the haplotype panel minimizing $s_x(\pi)$

traceback $\Leftarrow [(0, -1, \text{Null})]$ ▷ Traceback states contain "site", "prevID" and "prevTraceback"
states $\Leftarrow$ [(tails[0], 0, traceback.back())] ▷ States contain "below"-node, "score" and "traceback"
$z_{\min} \Leftarrow 0$, $t \Leftarrow 1$
for $i = 0, \ldots, M-1$ **do** ▷ Iterate over all sites
    $z'_{\min} \Leftarrow z_{\min} + \mu[i] + \rho_c[i]$
    newStates $\Leftarrow [\,]$
    for s in states **do** ▷ Iterate over all states on the stack
        if s.score + $\rho_c[i] + \mu_c[i] < z'_{\min} + \rho[i] - \rho_c[i]$ **then** ▷ Extensions by $x[i]$
            $t \Leftarrow$ extension of $s.below$ by $x[i]$
            if s extends to t **then**
                append $(t, s.\text{score}+\mu_c[i] + \rho_c[i], s.\text{traceback})$ to newStates
                $z'_{\min} \Leftarrow \min(z'_{\min}, s.\text{score}+\mu_c[i] + \rho_c[i])$
            end if
        end if
        if s.score + $\rho_c[i] + \mu[i] < z'_{\min} + \rho[i] - \rho_c[i]$ **then** ▷ Extensions by $1 - x[i]$
            $t \Leftarrow$ extension of $s.below$ by $1 - x[i]$
            if s extends to t **then**
                append $(t, s.\text{score}+\mu[i] + \rho_c[i], s.\text{traceback})$ to newStates
                $z'_{\min} \Leftarrow \min(z'_{\min}, s.\text{score}+\mu[i] + \rho_c[i])$
            end if
        end if
    end for
    bestExtension $\Leftarrow \text{argmin}_{s \in \text{newStates}}(s.\text{score})$ ▷ Recombination events
    append $(i+1,$ bestExtension.below.above.ID, bestExtension.traceback$)$ to traceback
    append (tails$[i+1]$, $z'_{\min} - \rho_c[i] + \rho[i]$, traceback.back()) to newStates
    $z'_{\min} \Leftarrow \min(z'_{\min}, z_{\min} - \rho_c[i] + \rho[i])$
    states $\Leftarrow$ newStates
    $z_{\min} \Leftarrow z'_{\min}$
end for
path $\Leftarrow [\,]$ ▷ Best path traceback
bestState $\Leftarrow \text{argmin}_{s \in \text{states}}(s.\text{score})$
return traceback(bestState) ▷ See Algorithm 4

Algorithm 4 Threads-fastLS traceback

Input: Traceback state s and a d-PBWT of the genotype matrix X
Output: A Viterbi path through X

```
matchID ⇐ s.below.above.ID
tb ⇐ s.traceback
while tb is not Null do
    segStart ⇐ traceback.site
    append (segStart, matchID) to path
    matchID ⇐ tb.prevID
    tb ⇐ tb.prevTraceback
end while
return path.reverse()
```

R_c. In the following theorem, we generalize Lunter's Theorem 1 to the dynamic haplotype panel and to variable recombination rates, by relaxing the conditions on prefix extensions.

We observe that if a full prefix set S on $[0, l)$ contains a path π ending with R, then it does not need to contain any path π' such that $s(\pi') \geq s(\pi)$, since every extension of π' can then be obtained as an extension of π but with a lower or equal score. In this case we will say π' is *undercut* by π and write $\pi \leq \pi'$.

Theorem 3 (Threads-fastLS)**.** *Let S be a full prefix set on $[0, l)$ and S' a set of $[0, l+1)$-prefixes. Write $s' = \min_{\pi' \in S, a' \in \{0,1\}} s(\pi' a')$. Then S' is a full prefix set on $[0, l+1)$ if the following conditions hold:*

1. *For all $\pi \in S$ and all $a \in \{0, 1\}$ such that πa exists and $s(\pi a R_c) < s' + \rho[l]$, $\pi a R_c \in S'$.*

2. *S' contains a path of the form $\pi a R$, where $s(\pi a) = s'$.*

Correctness of the algorithm then follows by induction starting from the empty path.

Proof. We follow a similar strategy as the proof for Lunter's Theorem 1. The full prefix set S on $[0, l)$ can be trivially extended to a full prefix set on $[0, l+1)$ by

taking all possible extensions of paths in S. Explicitly, we define

$$S_R = \{\pi a R \,|\, \pi \in S, a \in \{0,1\}, \pi a \text{ exists}\}$$

and

$$S_{R_c} = \{\pi a R_c \,|\, \pi \in S, a \in \{0,1\}, \pi a \text{ exists}\},$$

giving a full prefix set $S_R \cup S_{R_c}$ on $[0, l+1)$.

It is sufficient to show that each $\sigma \in (S_R \cup S_{R_c}) \setminus S'$ is undercut by a path in S'. Fix a path $\pi' \in S'$ of the form $\pi a R$, where $s(\pi a) = s'$. Such a path must always exist since the full prefix set S must be non-empty.

Now, if $\sigma \in S_{R_c} \setminus S'$, then $s(\sigma) \geq s' + \rho[l]$ by assumption. Thus $s(\sigma) \geq s(\pi')$ so σ is undercut by π'. Similarly, for each $\sigma \in S_R \setminus S'$, $s(\sigma) \geq s(\pi')$ by definition. Again, choose $\pi \in S$ and $a \in \{0,1\}$ such that $s(\pi a) = s'$. Thus, each $\sigma \in (S_R \cup S_{R_c}) \setminus S'$ is undercut by a path in S', completing the proof. □

3.2.2 Controlling the number of active states

Following [55], we can extend the notion of prefixes undercutting one another and use this to prune the set of states active at each iteration. Note that there is a one-one correspondence between *states* at site l and prefixes on $[0, l+1)$, but because Algorithm 1 stores states and not whole prefixes, we will work with states here instead.

We define a relation between states s_1 and s_2 that either point to the same haplotype, i.e. $s_1.\text{below.ID} = s_2.\text{below.ID}$, or one of which is recombinant. In this case, we say s_1 *undercuts* s_2 if $s_1.\text{traceback.site} \geq s_2.\text{traceback.site}$ and $s_1.\text{score} \leq s_2.\text{score}$. This may be intuited from the fact that a state representing a path that has more recently recombined has greater capacity to be extended through the sequences in

the panel. This operation can be performed in $O(S \log(S))$ time, where S is the number of states.

Algorithm 5 Prefix pruning

Input: A set of states $\mathcal{S}$
Output: A set of states $\mathcal{S}' \subseteq \mathcal{S}$ that undercuts $\mathcal{S}$

Partition $\mathcal{S} = \bigcup_i \mathcal{S}_i$ such that $s.\text{ID} = i \Leftrightarrow s \in \mathcal{S}_i$
for each $\mathcal{S}_i$ **do**
    $\mathcal{S}'_i = \{\,\}$
    Sort $\mathcal{S}$ in descending order lexicographically by $(s.\text{traceback.site}, -s.\text{score})$ into a list L.
    $z \Leftarrow \infty$
    for each $s \in L$ **do**
        if $s.\text{score} < z$ **then**
            $z \Leftarrow s.\text{score}$
            Add s to $\mathcal{S}'$
        end if
    end for
end for
return $\bigcup_i \mathcal{S}'_i$

By first sorting by the traceback site, Algorithm 5 can assume successive states in L have greater distance to the most recent recombination event than the ones that precede them and can only be admitted to $\mathcal{S}'_i$ if they have a strictly lower score than all previously admitted states. By only comparing states that point to the same node in the dynamic panel we may be missing some states that could otherwise safely be omitted. This is a limitation inherent to the linked-list implementation of the d-PBWT compared with the array-based transform.

In practice the heuristic bounds in the Threads-LS algorithm are very efficient, and the extra overhead of pruning states is recommended only in cases where the search space grows to thousands of states.

3.2.3 Comparison with Threads

By Theorem 3, Threads-fastLS is guaranteed to give an exact Viterbi path through the reference panel, while the base Threads algorithm relies on a preliminary haplotype matching step to reduce computational time. Threads-fastLS thus gives an upper bound on the accuracy of Threads.

The most crucial difference between the two algorithms lies in their memory usage. For Threads-fastLS to run, it must store the whole $N \times M$-sized d-PBWT in memory, whereas Threads can run without storing more than one site in memory at a time, in addition to the Viterbi paths and haplotype matches described in Chapters 3.1.1 and 3.1.2. Moreover, requiring that the whole panel is kept in memory means Threads-fastLS cannot be parallelized to the same extent that Threads can, since each thread would need to maintain a copy of the panel in memory.

Regardless of memory usage, the two algorithms also have different runtimes, with Threads being optimized for whole-ARG inference, and Threads-fastLS more flexible, allowing Viterbi inference for individual sequences given the d-PBWT of the panel. Threads-fastLS does not achieve the $O(M)$ runtime claimed in [55] for fastLS for a panel of M sites and N samples. This is due to two main differences between fastLS and Threads-fastLS. First, Threads-fastLS is built on top of the linked-list based d-PBWT and so cannot use the same array-based indexing tricks to speed up inference as fastLS does. In particular, Threads-fastLS must regularly compare long stretches of genome to evaluate the extensibility of states. Second, Threads-fastLS has been built in a way that handles arbitrary recombination maps, while fastLS assumes all sites are equally spaced in terms of both genetic and physical distance. This means the branch-and-bound algorithm in Threads-fastLS must consider more branches (i.e., recombination events) than fastLS. With these changes, we observe runtimes of approximately $O(M \log N)$ for Threads-fastLS for Viterbi inference of a single sequence for a panel of size $N \times M$ (Figure 3.8).

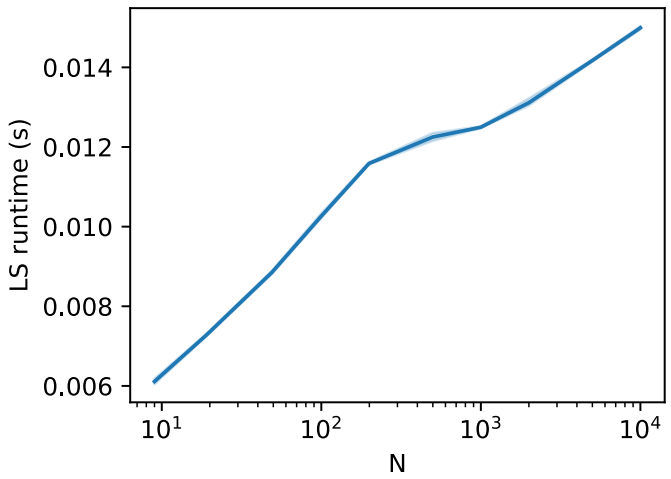


Figure 3.8: Runtimes for Li-Stephens inference of a single sequence using Threads-fastLS over $5{,}000$ sites. We simulate N sequences under a constant $N_e = 10{,}000$ demography with a fixed recombination rate of $\rho = 1.2 \times 10^{-8}$ and a mutation rate of $\mu = 1.65 \times 10^{-8}$. Sequences are subsampled to the first $5{,}000$ sites and Threads-fastLS is run for a single sequence for panels of varying size.

ARG inference using Threads is run site-by-site, whereas Threads-fastLS runs sample-by-sample. This difference has a big practical impact when it comes to I/O usage, as most conventional genotype file formats are compressed, indexed, and optimized for site-wise queries. Threads is written to operate directly on PLINK2 pgen files [14], while Threads-fastLS requires an extra pre-processing step to index genotypes sample-wise.

However, keeping the d-PBWT in memory also has benefits. The possibility of removing samples from the d-PBWT means we can sequentially thread and then remove samples in a manner similar to a Gibbs sampler, giving a weaker dependency on the order in which samples are threaded. This approach is discussed in more detail in Chapter 3.5.3. Furthermore, access to the divergence values allows us to query the reference panel to obtain, for each segment along the inferred path, all haplotypes that are identical to the threading target along that path. This ex-

tra information can be extracted with little extra cost by modifying Algorithm 4 as shown in Algorithm 6.

Algorithm 6 Threads-fastLS full traceback

Input: Traceback state s and a d-PBWT of the genotype matrix X
Output: A Viterbi path through X

```
matchID ⇐ s.below.above.ID
tb ⇐ s.traceback
segEnd ⇐ M
while tb is not Null do
    segStart ⇐ traceback.site
    segIDs ⇐ {matchID}
    tmpNode ⇐ tb.State    ▷ Look up the state corresponding to this traceback
    while tmpUp.divergence at segEnd ≤ segStart and tmpNode.ID ≥ 0 do
        tmpNode ⇐ tmpNode                          ▷ Traverse d-PBWT upwards
        add tmpNode.ID to segIDs
    end while
    tmpNode ⇐ tb.State
    while tmpUp.below.divergence at segEnd ≤ segStart
            and tmpNode.below.ID ≥ 0 do
        tmpNode ⇐ tmpNode.below                    ▷ Traverse d-PBWT downwards
        add tmpNode.ID to segIDs
    end while
    append (segStart, segIDs) to path
    matchID ⇐ tb.prevID
    tb ⇐ tb.prevTraceback
    segEnd ⇐ segStart
end while
return path.reverse()
```

3.3 Diploid Threads-fastLS

In this chapter we describe an extension of Threads-fastLS that infers threading instructions of unphased genotypes by running the diploid Li-Stephens model. As in Chapter 3.2, the implementation draws heavily from fastLS [55] but is extended to allow for a dynamic reference panel and arbitrary recombination maps.

Recall from Chapter 2.3 that the diploid Li-Stephens algorithm makes the same assumptions as the haploid algorithm, but that it does not require phased input sequences. Instead, an unphased sequence $g \in \{0, 1, 2\}^M$ of length M is modelled as the sum of two paths through a phased reference panel of N sequences. Traditional implementations of the diploid Viterbi and forward-backward algorithms have state space $\{1, \ldots, N\} \times \{1, \ldots, N\}$, requiring $O(N^2M)$ time and space to compute. For most modern data sets, this computational complexity is prohibitive.

In [55], Lunter proposed an algorithm that solves the diploid Viterbi problem in time independent of N by using the PBWT of the reference panel. The cost of building and storing the PBWT is $O(NM)$ time and memory, both a great improvement over the traditional approach. However, the constraint that the PBWT be stored in memory is still very limiting, as we saw in our discussion of Threads and Threads-fastLS.

In what follows, we give a high-level description of the diploid Threads-fastLS algorithm and a discussion of its limitations with further details and a proof of correctness provided in Chapter 3.3.1. The core steps of the diploid Threads-fastLS are identical to the haploid algorithm. It is a branch-and-bound algorithm and employs a score-based heuristic to limit the number of active branches. Extensibility of states is checked using the same criteria as those that were described in Chapter 3.2.

Recall that the leaves of the search tree for the haploid algorithm represented *states*, each comprised of a *below*-node, a *score*, and a *traceback* state. To account for pairs of paths, the diploid algorithm employs *state-pairs* comprising a pair of states and a combined score. More precisely, each pair p represents two states a and b, with nodes $p.a$.below and $p.b$.below, traceback states $p.a$.traceback and $p.b$.traceback and a score $p.s$. Once the algorithm has processed the whole data set, the trace-

back states in the state-pair with lowest score can be used to recover a pair of paths through the reference panel of lowest score.

In the haploid Threads-fastLS algorithm, we omit possible extensions of states if the path score we obtain by extending is higher than the one we would get by recombining from the state of lowest score. In the diploid algorithm, we can omit an extension of a state-pair p if there is another pair p' with $p.a = p'.a$, such that the score obtained by extending both $p.a$ and $p.b$ is higher than extending $p'.a$ and recombining $p'.b$, and similarly for any such state p' with $p.b = p'.b$. Finally, we may also omit extensions of p if there is any other state p^* such that recombining both $p^*.a$ and $p^*.b$ results in a lower score than extending $p.a$ and $p.b$. In practice, this observation means we must keep track of not just a single, best state as in the haploid algorithm, but also, for each state c we must know the state-pair p' that has lowest score among all state-pairs with $p.a = c$ or $p.b = c$. We provide simplified pseudocode in Algorithm 7.

To account for variable recombination maps as in the haploid algorithm, we must consider recombination events for each state-pair at each site, instead of deferring recombination like fastLS. This can result in a very high number of active state-pairs and slow run-times. However, we observe favorable results for run-times and memory usage when compared with the traditional diploid Li-Stephens Viterbi algorithm.

We note two other limitations of the diploid Threads-fastLS. First, the threading instructions inferred by a diploid Li-Stephens model do not account for coalescence between the two lineages being inferred, only giving threading targets from the reference panel. For ARG inference in a large, panmictic population this is not likely to be problematic, but in practice long runs of homozygosity are frequently observed in real data, implying recent coalescence between the two parental haplotypes [13].

Algorithm 7 Diploid Threads-fastLS

Input: Genotype matrix X and unphased genotype vector g, mutation penalties μ, μ_c, recombination penalties ρ, ρ_c

Output: output: A pair of Viterbi paths through X

tracebacks $\Leftarrow [(0, -1, \text{Null})]$
pairs $\Leftarrow [(\text{tails}[0], \text{traceback.back}(), \text{tails}[0], \text{traceback.back}(), 0)]$
bestPair $\Leftarrow$ pairs.back()
$z_{\min} \Leftarrow 0$, $t \Leftarrow 1$
for $i = 0, \ldots, M-1$ **do** ▷ Iterate over all sites
    newPairs $\Leftarrow [\,]$
    $\rho_\Delta \Leftarrow \max(0, \rho[i] - \rho_c[i])$
    $z_{\min} \Leftarrow$ minimum extension cost for bestPair
    $Z_{\text{local}}, Z'_{\text{local}}, \text{Ext}_0, \text{Ext}_1 \Leftarrow \{\,\}, \{\,\}, \{\,\}, \{\,\}$ ▷ Empty maps indexed by states
    for p in pairs **do**
        for c in $[p.a, p.b]$ **do**
            if c is not a key in Ext_0 or Ext_1 **then**
                Compute the 0- and 1-extensibility of c and cache in $\text{Ext}_0, \text{Ext}_1$
            end if
            $z_c \Leftarrow$ cheapest extension/recombination cost for c
            if $Z_{\text{local}}[c]$ is empty or $z_c < Z_{\text{local}}[c]$ **then**
                $Z_{\text{local}}[c] = z_c$
            end if
        end for
    end for
    for p in pairs **do** ▷ Main extension loop
        $z_a \Leftarrow Z_{\text{local}}[a]$
        $z_b \Leftarrow Z_{\text{local}}[b]$
        $R = \min(z_a, z_b, z_{\min} + \rho_\Delta) + \rho_\Delta$
        if p.score $+2(\rho_c[i] + \mu_c[i]) \leq R$ **then**
            Get double extension events, update Z'_{local} and newPairs
        end if
        if p.score $+2\rho_c[i] + \mu[i] + \mu_c[i] + \leq R$ **then**
            Get extension/mutation events, update Z'_{local} and newPairs
        end if
        if p.score $+2(\rho_c[i] + \mu[i]) + \leq R$ **then**
            Get double mutation events, update Z'_{local} and newPairs
        end if
    end for ▷ Algorithm continued on next page

recPairs ⇐ { } ▷ Recombined state-pairs
recombinedStates ⇐ { } ▷ Recombined states
for p in newPairs **do** ▷ Single recombination events
$r = p$.score - $\rho_c[i] + \rho[i]$
for c in $[a, b]$ **do**
if $p.c$ not in recombinedStates and p.score $= Z'_{\text{local}}[c]$ **then**
add c to recombinedStates
tracebacks.append($i + 1, p.c$.below.above, $p.c$.traceback)
$d \Leftarrow a$ if $c = b$ else b
recPairs.append($p.c$.below, $p.c$.traceback, tails[$i + 1$], tracebacks.back(), r)
end if
end for
end for
bestPair ⇐ $\text{argmin}_{p \in \text{newPairs}} p.s$ ▷ Double recombination event
$r_{\text{double}} \Leftarrow z_{\min} + 2(\rho[i] - \rho_c[i])$
tracebacks.append($i + 1$, bestPair.a.below.above, bestPair.a.traceback)
tracebacks.append($i + 1$, bestPair.b.below.above, bestPair.b.traceback)
add recPairs to newPairs
newPairs.append(tails[$i + 1$], tracebacks.back(), tails[$i + 1$], tracebacks[tracebacks.size()-2], r_{double})
bestPair ⇐ $\text{argmin}_{p \in \text{newPairs}} p.s$
$z_{\min} \Leftarrow$ bestPair.s
pairs ⇐ newPairs
end for
bestPair ⇐ $\text{argmin}_{p \in \text{pairs}} p.s$
return traceback(bestPair.a), traceback(bestPair.b) ▷ See Algorithm 4

Second, for large-scale analyses, storing a full PBWT for the reference panel in memory is not feasible. To apply memory-saving techniques such as those employed by Threads for haploid genotypes, it is necessary to first perform preliminary phasing of the input sequences. However, if the input diplotypes can readily be phased, it may be argued that a diploid algorithm is no longer necessary, since simpler, haploid algorithms may be applied instead. This limits the usability of diploid Threadsl-fastLS to data sets that fit in memory and for which no accurate phasing is available.

3.3.1 Proof of correctness

In this chapter we give a formal proof that for a fixed haploid reference panel and fixed input diplotype x, Algorithm 7 finds a pair of paths π_a, π_b through the panel minimizing the objective function

$$\begin{aligned} s_x(\pi_a, \pi_b) \coloneqq \sum_{m=1}^{M} & \mu[m]|x[m] - (x_{\pi_a[m]}[m] + x_{\pi_b[m]}[m])| \\ & + \mu_c[m]\big(2 - |x[m] - (x_{\pi_a[m]}[m] + x_{\pi_b[m]}[m])|\big) \\ & + \rho[m]\big(2 - \delta(\pi_a[m], \pi_a[m+1]) - \delta(\pi_b[m], \pi_b[m+1])\big), \\ & + \rho_c[m]\big(\delta(\pi_a[m], \pi_a[m+1]) + \delta(\pi_b[m], \pi_b[m+1])\big), \end{aligned}$$

where, as in Chapter 3.2.1, ρ, ρ_c are recombination penalties and μ, μ_c are mutation penalties. To simplify the theorem and its proof, we make an extra assumption that $\rho[l] > \rho_c[l]$ for all l. In general, recombination events are rare and genetic markers are dense, so in most realistic applications this is a reasonable assumption.

As in Chapter 3.2.1, each path π is a string $a_1 r_1 a_2 r_2 \cdots a_M r_M$ with $a_i \in \{0, 1\}$ and $r_i \in \{R, R_c\}$ for each i, with a_i specifying the allele carried by the haplotype at site i and R, R_c specifying recombination or no recombination, respectively. Using

the extensibility criteria defined in Chapter 3.2, each such string either defines a unique path through the reference panel or is incompatible with the panel. Instead of working with sets of paths, as in the haploid case, here we consider sets of path-pairs $\{\pi_a, \pi_b\}$. We say a path π belongs to a path-pair set $\mathcal{P}$ if it is a member of any of the path-pairs in $\mathcal{P}$.

We call a set of path-pairs $\mathcal{P}_l$ a *full prefix set* for x on $[0, l)$ if for any diploid sequence x' such that $x[0, l) = x'[0, l)$, there is a path minimizing $s_{x'}(\cdot, \cdot)$ whose $[0, l)$-prefix lies in $\mathcal{P}_l$. As in the proof for Theorem 3, it is sufficient to show that the extension and recombination rules defined by Algorithm 7 extend a full prefix set $\mathcal{P}$ on $[0, l)$ to a full prefix set $\mathcal{P}'$ on $[0, l + 1)$.

Define the *tail* of a path $\pi = a_1 r_1 a_2 r_2 \cdots a_M r_M$ as the haplotype index at M and the index of the last recombination event, denoted

$$t(\pi) \coloneqq (\pi[M], \max\{l \mid r_l = R \text{ or } l = 0\}).$$

In the haploid setting, it is easy to see that if two paths π, π' satisfy $t(\pi) = t(\pi')$ and $s(\pi) \leq s(\pi')$, we can discard π'. Here, a path-pair $\{\pi_a, \pi_b\}$ *undercuts* a path-pair $\{\pi'_a, \pi'_b\}$ if $t(\pi_a) = t(\pi'_a)$, π_b ends with R, and $s(\{\pi_a, \pi_b\}) \leq s(\{\pi'_a, \pi'_b\})$.

For a set $\mathcal{P}$ of path-pairs, we say that the path-pair $\{\pi_a, \pi_b\}$ is a *local minimum* for a path $\pi \in \mathcal{P}$ if $t(\pi) \in \{t(\pi_a), t(\pi_b)\}$ and

$$s(\pi_a, \pi_b) = \min_{\{\pi'_a, \pi'_b\} \in \mathcal{P},\, t(\pi) \in \{t(\pi'_a), t(\pi'_b)\}} s(\pi'_a, \pi'_b).$$

We say $\{\pi_a, \pi_b\}$ is a *global minimum* for $\mathcal{P}$ if

$$s(\pi_a, \pi_b) = \min_{\{\pi'_a, \pi'_b\} \in \mathcal{P}} s(\pi'_a, \pi'_b).$$

Under our assumption that $\rho[l] > \rho_c[l]$ for all l, any sequence ending in R_c will have a local minimum also ending in $R_c c$. We define $\rho_\Delta[l] \coloneqq \rho[l] - \rho_c[l]$.

Theorem 4 (Diploid Threads-fastLS)**.** *Let $\mathcal{P}$ be a full prefix set on $[0, l)$ and*

$$\mathcal{P}^* = \big\{\{\pi_a u_a r_a, \pi_b u_b r_b\} \mid \{\pi_a, \pi_b\} \in \mathcal{P},\ u_a, u_b \in \{0,1\},\\ r_a, r_b \in \{R, R_c\},\ \pi_a u_a, \pi_b u_b \textit{ exist}\big\},$$

be its trivial extension to a full prefix set on $[0, l+1)$. Let $s = \min_{\{\pi_a,\pi_b\}\in\mathcal{P}^*} s(\{\pi_a, \pi_b\})$ be the penalty of the global minimum for $\mathcal{P}^*$ and, for each path $\pi \in \mathcal{P}^*$, $z(\pi)$ be the penalty for its local minimum. Then a set $\mathcal{P}'$ of $[0, l+1)$-prefixes is a full prefix set if the following conditions hold:

1. *For all $\{\pi_a, \pi_b\} \in \mathcal{P}^*$ with π_a, π_b both ending in R_c such that*

$$s(\pi_a, \pi_b) < \min(s' + 2\rho_\Delta[l], z(\pi_a) + \rho_\Delta[l], z(\pi_b) + \rho_\Delta[l]),$$

   we have $\{\pi_a, \pi_b\} \in \mathcal{P}'$.

2. *For each path $\pi u R_c$ in $\mathcal{P}'$, $\mathcal{P}'$ also contains a path-pair $\{\pi_a u_a R_c, \pi_b u_b R\}$ satisfying $t(\pi u R_c) = t(\pi_a u_a R_c)$ such that the path-pair $\{\pi_a u_a R_c, \pi_b u_b R_c\}$ is a local minimum for $\pi u R_c$.*

3. *$\mathcal{P}'$ contains a path of the form $\{\pi_a u_a R, \pi_b u_b R\}$ such that $\{\pi_a u_a R_c, \pi_b u_b R_c\}$ is a global minimum for $\mathcal{P}'$.*

Proof. We follow the same outline as in the proof of Theorem 3, showing that each path-pair in $\mathcal{P}^* \setminus \mathcal{P}'$ is undercut by a path-pair in $\mathcal{P}'$. Let $P = \{\pi_a u_a r_a, \pi_b u_b r_b\}$ be any path-pair in $\mathcal{P}^* \setminus \mathcal{P}'$.

First, suppose $r_a = r_b = R_c$. Then, by definition,

$$s(\pi_a u_a r_a, \pi_b u_b r_b) \geq \min(s' + 2\rho_\Delta[l], z(\pi_a) + \rho_\Delta[l], z(\pi_b) + \rho_\Delta[l]).$$

If $s(P) \geq s' + 2\rho_\Delta[l]$, then P is undercut by $\{\pi'_a u'_a R, \pi'_b u'_b R\} \in \mathcal{P}'$, where the pair $\{\pi'_a u'_a R_c, \pi'_b u'_b R_c\}$ is a global minimum for $\mathcal{P}^*$. If $s(P) \geq z(\pi_a) + \rho_\Delta[l]$, then P is undercut by a path-pair $\{\pi'_a u'_a R_c, \pi'_b u'_b R\} \in \mathcal{P}'$, where $\{\pi'_a u'_a R_c, \pi'_b u'_b R_c\}$ is a local minimum for $\pi_a u_a r_a$. The case $s(P) \geq z(\pi_b) + \rho_\Delta[l]$ follows similarly.

Next, suppose $r_a = R_c$ and $r_b = R$. If $\mathcal{P}'$ contains a path π' with $t(\pi') = t(\pi_a u_a R_c)$, it also contains a path-pair $\{\pi'_a u'_a R_c, \pi'_b u'_b R\}$ such that $\{\pi'_a u'_a R_c, \pi'_b u'_b R_c\}$ is a local minimum for $\pi_a u_a R_c$, undercutting P. If $\mathcal{P}'$ contains no such path, then in particular

$$\begin{aligned} s(P) &= s(\pi_a u_a R_c, \pi_b u_b R_c) + \rho_\Delta[l] \\ &\geq \min(s' + 2\rho_\Delta[l], z(\pi_a) + \rho_\Delta[l], z(\pi_b) + \rho_\Delta[l]) + \rho_\Delta[l] \\ &\geq s' + 2\rho_\Delta[l], \end{aligned}$$

since $s' \leq z(a), z(b)$. Thus, P is undercut by a double recombination on the global minimum. By symmetry in the indices a, b, the same argument holds for the case $r_a = R$ and $r_b = R_c$. Finally, if $r_a = r_b = R$, P is undercut by a double recombination on the global minimum. □

3.4 An efficient diploid fastLS

While the chief advantage of the diploid fastLS algorithm compared with traditional diploid Li-Stephens solvers is the improved inference speed, the memory footprint is also decreased from $O(N^2M)$ to $O(NM)$. In this chapter, we depart from ARG inference to show how our modelling and algorithmic advances in Chapters 3.1 and 3.3 allow us to construct an efficient, scalable, generalized variant of the diploid fastLS.

Our improvement relies on the simple observation that the inference step of the diploid fastLS (Algorithm 6 in [55]) only requires one site in memory at a time, while the traceback algorithm (Algorithm 7 in [55]) relies on the full PBWT being available.

To obtain a diploid Li-Stephens Viterbi solver, we re-implemented the diploid Threads-fastLS algorithm, using an array-based PBWT and keeping only column of the PBWT in memory at a time. As in the main Threads algorithm, we do this in a way that infers the Viterbi paths of multiple input sequences in parallel, using only one pass through the data and keeping track of inferred segments in a way that removes the need for a separate pass through the PBWT for the traceback step as in fastLS. We thus obtain a memory footprint of $O(N_{\text{reference}} + MN_{\text{target}})$, where $N_{\text{reference}}$ is the number of sequences in the reference panel and N_{target} is the number of input sequences. For very long sequences, the memory usage may be improved even further by addition of a pruning step as described in Chapter 3.1. Figure 3.9 compares time and memory usage of a classical diploid Viterbi algorithm with our streaming fastLS (SFLS) version.

3.5 ARG inference, extensions and heuristics

The Threads algorithm is designed to infer threading instructions for whole-genome sequences of large sample sizes and some of the modelling assumptions may be inappropriate for smaller sample sizes and for sparsely genotyped sequences such as those obtained using genotyping arrays. In this chapter we describe some of the practical concerns that arise during de-novo ARG-inference using Threads and Threads-fastLS and discuss the extra steps and heuristics we adopt to address them.

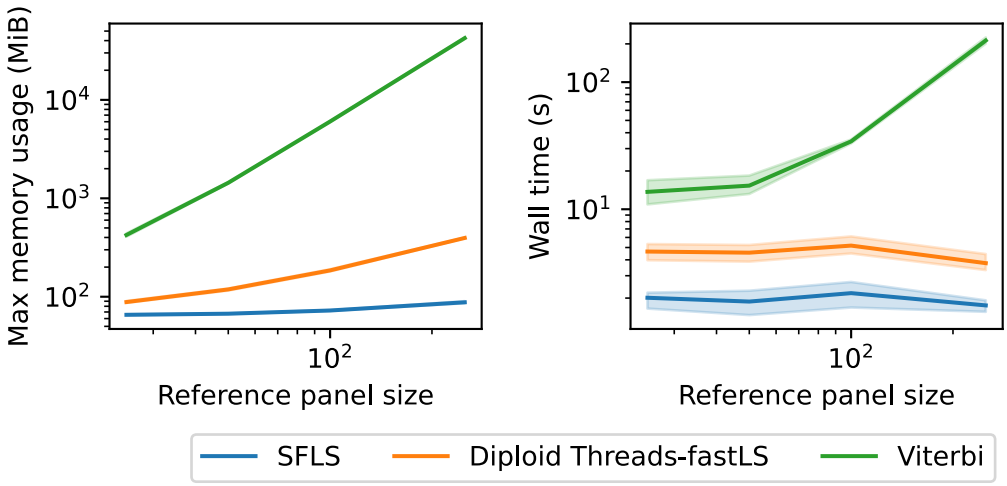


Figure 3.9: Computational performance of different diploid Viterbi path solvers for a single unphased input sequence. Reference panels of size varying from 25 to 250 diploids were simulated over a 1-megabase region under a constant-sized demography. **a**. Peak memory usage for each algorithm. Non-linear growth in Diploid Threads-fastLS and SFLS is due to larger simulations also having more mutations per megabase. **b**. Runtime in seconds is near-constant for Diploid Threads-fastLS and SFLS at this small scale. However, both algorithms have an $O(NM)$ time complexity due to the PBWT construction.

3.5.1 Setting the Li-Stephens parameters

As discussed in Chapter 2.3, mismatch and recombination rates for the Li-Stephens model are usually either estimated numerically or modelled using coalescent arguments that rely on panel size, empirical recombination and mutation rates, and demographic history. For Threads and Threads-fastLS, the panel size changes at each iteration, making numerical estimates unfeasible and instead we rely on coalescent modelling. Definitions and proofs relating of relevant results relating to the coalescent were introduced in Chapter 2.1.

Coalescent estimates of Li Stephens parameters usually rely on the fact that for a tree on N leaves under the standard coalescent, the expected time to the first coalescence event for a given leaf is $2/N$. In Threads, we always assume that the

coalescence times of the N-th sequence being threaded and the $N-1$ sequences previously threaded are jointly distributed as a coalescent tree on N leaves. In a constant-sized population of N_e haploids, the expected coalescence time becomes $2N_e/N$. It is usually assumed that, between two sites at a distance of l centiMorgans, recombination events follow a Poisson process with rate $0.01l \cdot 2N_e/N$.

We extend this model to piecewise-constant demographies, which are equivalent to a piecewise linear monotonic re-scaling of time. Any variable-sized demography $N_e(t)$ gives a mapping from absolute time, as measured in generations, to standard coalescence time via $t \mapsto \int_0^t N_e(s)^{-1}\mathrm{d}s$.

For a piecewise-constant demography with time breakpoints $0 = t_1 < t_2 < \cdots < t_K$ and effective haploid population sizes $N_1, N_2, \ldots, N_K$ we obtain the expected coalescence times as follows. We first map each of the breakpoints to standard coalescent time using the map $t_i \mapsto \tau_i \coloneqq \tau_{i-1} + (t_i - t_{i-1})/N_{i-1}$ for $i > 1$, with $\tau_1 = 0$. Then, to get the expected coalescence time for N leaves under this demography, we linearly interpolate between the values τ_i and then change coordinates back to time in generations using the inverse map. In practice, this means finding the highest i such that $\tau_i \leq 2/N$, giving the expected coalescence time $T_N \coloneqq t_i + (2/N - \tau_i)N_i$.

Together, these results give us a recombination probability of $1 - e^{-0.01l \cdot 2T_N/N}$ for sites at a distance of l centiMorgans under a tree on N leaves and a mismatch probability of $1 - e^{-\mu l' 2T_N/N}$ on a tract of genome of length l' base-pairs under a constant per-site-per-generation mutation rate of μ.

3.5.2 Sparse sites

In many settings we do not have access to whole sequences from which to infer ARGs. In these cases it is normal to impute the missing variants using a reference panel of whole genomes. However, imputation may not always be available, and when it is available it may have poor quality for rare variants when the source pop-

ulation is poorly represented in the reference panel. In [99], Zhang et al. showed how ARGs inferred directly from genotyping arrays may be used for analysis of rare variation in complex traits. Here, we focus on inference from genotyping arrays using Threads.

Missing genotype status results in misspecification of the emission probabilities for the Li-Stephens HMM. Sites in genotyping arrays are not sampled at random and have a bias towards common variants. Accurate modelling of the effects of SNP ascertainment is highly complex. A good approximation can be made by using the conditional site frequency spectrum (CSFS) [89], which is leveraged by ASMC [70] for inference of coalescence times from genotyping arrays.

In Threads, we opt for a simpler, heuristic approach by artificially inflating the rate of recombination events. Recall that if the probability of recombination between two sites in the Li-Stephens model is p, then the transition penalty in the HMM is $-\log(p/N)$. When modelling sparse sites in Threads, we instead use a transition penalty of $-\log(p)$. This heuristic is designed for cases where common variants are expected to be overrepresented in the subset of variants that have been ascertained. It is based on the intuition that as sample sizes increase, expected coalescence times drop, but the rare variants used to distinguish closely related sequences are missing from the panel. The lower transition penalty encourages recombination in this low-signal regime. Mutation penalties in the sparse model are kept identical to the full whole-genome model.

This model differs from other array-based Li-Stephens implementations, such as Beagle 5.4 [9] and IMPUTE5 [76] that use a fixed mismatch probability of 0.0001 and recombination rates identical to the full model. However, these rates are optimized for biobank-scale panel sizes and not guaranteed to generalize to the whole process of ARG inference. In our experiments we find that Threads is relatively robust to the choice of Li-Stephens parameters, with slightly more accurate results

observed using the heuristic sparse-model parameters described above. However, we caution that the exact choice of parameters is always highly dataset-specific and no single set of modelling assumptions or heuristics will fit all data sets.

3.5.3 Threading small sample sizes

As discussed in Chapter 3.1.3, segments from Viterbi paths through a reference panel do not necessarily match underlying IBD segments and the difference is especially pronounced for small sample sizes. Accurate threading of small sample sizes is crucial to get high-quality inference of ancient coalescence events, since samples after the first few hundred are likely to coalesce into the ARG in recent time, only helping to resolve recent ancestry but contributing little information for deep coalescence events.

To improve threading accuracy in small sample sizes, we explored two approaches. First, we implemented a Viterbi algorithm based on the pairwise sequentially Markovian coalescent (PSMC) [47] that we apply to any haplotype matching segment with more than 5 heterozygous sites inferred while threading the first 1,000 sequences. The algorithm identifies the most likely path through a set of discrete time-states, and we interpret each jump in the PSMC path as a recombination event defining a new IBD segment. This operation breaks the input segment into one or more sub-segments that we each date independently using the approach described in Chapter 3.1.3.

Second, by using the dynamic reference panel in Threads-fastLS, we have implemented a "cycling" algorithm. Here, we infer ARGs as normal, then, for the first 100 sequences added to the ARG, we remove them from the dynamic-PBWT data structure, and re-infer threading instructions from the remaining sequences. A similar strategy was previously described in the context of ARG inference in [73].

In our experiments, we found that the addition of the PSMC step increased accuracy in inference of deep coalescence times when inferring ARGs from dense marker sets but found limited gains for inference from genotyping arrays. By default, this step is included in inference from dense marker sets but omitted when Threads is run in genotyping array mode. We observed limited gains from using the cycling algorithm. Using the modular nature of the threading instructions it is also possible to *seed* Threads ARG inference with an independently-inferred ARG built using a small batch of sequences using methods which are less scalable but that rely on more accurate modelling of deep-time genealogical processes. However, different calibration of coalescence times between different models can result in sub-optimal inference results. We discuss seeded inference in more detail in Chapter 3.5.5.

3.5.4 Adaptive Threads-fastLS

To overcome the memory bottleneck incurred by storing the whole d-PBWT in memory during ARG inference using Threads-fastLS, we save memory by incrementally pruning away the rarest sites. We call this approach "adaptive Threads-fastLS".

Adaptive Threads-fastLS infers threading instructions as usual for the first 1,000 samples, then discards the d-PBWT, rebuilds it without singletons and then continues inference using only filtered haplotypes. Generally, once $k \times 1000$ haplotypes have been added to the panel, it is trimmed to discard sites with minor allele count $\leq k$. We repeat this operation up to $k = 5$. The time spent on inference of threading instructions far exceeds the time spent on panel-building, so re-building the panel for the first few thousand sequences has little impact on runtime. Instead, subsequent sequences have significantly fewer sites to process, leading to improvements in both runtime and memory usage.

However, while adaptive Threads-fastLS can reduce the memory footprint of Threads-fastLS by more than half, depending on the data set, there is significant cost to accuracy. Rare variation is essential to discriminate between closely related haplotypes, and a coarser genetic map makes accurate inference of segment boundaries more difficult.

3.5.5 Seeded inference

Recall that the output of Threads is a set of threading instructions. For a genomic region $[L_1, L_2]$ and a set of genotypes indexed $1, \ldots, N$, these consist of a map $T_n : [L_1, L_2] \rightarrow \mathbb{R}_+ \times \{1, \ldots, n-1\}$ for each $n = 1, \ldots, N$, representing coalescence times and threading targets. Any set of threading instructions uniquely identifies an ARG, meaning we can combine threading instructions from different sources.

This modularity is especially useful in settings where only a subset of sequences have been whole-genome sequenced, but the rest genotyped using genotyping arrays. In this case, it is possible to build a high-quality ARG from the full sequences, and then subsequently thread the sparser array sequences. Alternatively, when all genomes have been fully sequenced, we may apply a less scalable method that jointly models all sequences, such as ARGWeaver [73] or Relate [84], to build a scaffold ARG from a small batch of initial sequences, to which the rest of the cohort is subsequently threaded. We call this approach *seeded threading* and it is the motivation behind the imputation algorithm described in Chapter 6.

In simulations, we found that seeding ARG inference from arrays with a small number of whole-genome sequences greatly improved inference in deep time, although we observed imperfect calibration of coalescence times between the two methods. This lack of calibration was even more exacerbated when combining more than one ARG inference methods, as we show in Chapter 4.3.5, suggesting

more research is required into optimal ways to combine ARGs inferred through different means.

3.5.6 Mutation-consistent threading

The threading algorithms we have described in this chapter do not guarantee that the mutations used in for inference are perfectly represented as edges in the resulting ARG. In general, it is preferable to avoid overfitting inferred ARGs to error-prone genetic sequences, but in applications such as allele age estimation it is useful to have a one-to-one correspondence between input mutations and a subset of edges on the ARG, a property we call *mutation consistency*. In this chapter we describe a threading algorithm that guarantees consistency with input mutations and discuss why such an approach is incompatible with Threads.

Suppose we want to estimate the TMRCA t of two sequences at a biallelic site in the genome where one or both sequences carry the non-ancestral allele and suppose we know the age a of the mutation. If only one sequence carries the mutation, then $t \geq a$, and if both carry the mutation, $t \leq a$. Applying methods based on the sequentially Markovian coalescent such as PSMC [47] or ASMC [70] to the neighbouring region of genome, we can estimate a posterior distribution $p(t)$. By masking and reweighing the transition probabilities in the underlying SMC model, we can also obtain a posterior $p(t|a)$, that satisfies $p(t > a|a) = 0$ if both sequences carry the mutation and $p(t < a|a) = 0$ if only one sequence carries the mutation (Figure 3.10). If neither sequence carries a mutation, we make no changes to the SMC model.

To apply this idea to ARG inference, suppose we have an ARG on sequences $1, \ldots, n$ that is consistent with the observed mutations carried by those sequences, and we wish to thread the $n + 1$-th sequence in a way that maintains mutation consistency. In the partial ARG, each mutation uniquely maps to an edge in a marginal

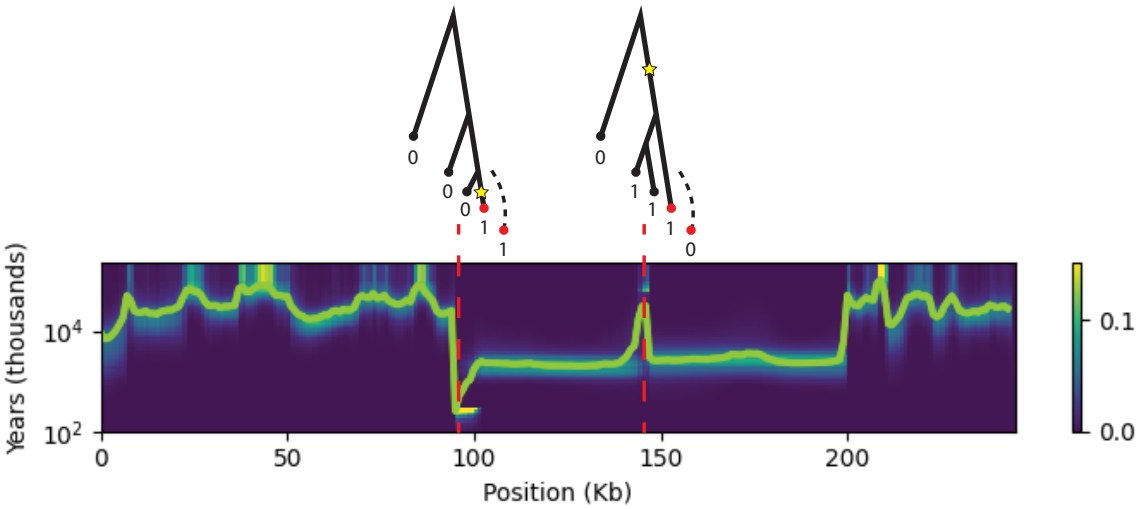


Figure 3.10: Output from a toy example of a mutation-constrained SMC algorithm with two constraints. The heat map shows the full posterior of the forward-backward algorithm and the green line shows the expected pairwise coalescence time at each site. The marginal trees show the sample being threaded and the threading target with red dots and the constraining mutations with stars. In the first case a shared recent mutation constrains the coalescence time downwards, and in the second case a heterozygous deep-time mutation constrains coalescence upwards. Here, we used a modified, mutation-consistent version of ASMC [70].

tree that goes from t_1 to t_2 generations in the past, with $t_1 < t_2$. The numbers t_1, t_2 give a lower and upper bound for the age of the mutation, respectively. After identifying the threading target for the new sequence, to maintain mutation consistency, we must guarantee that the the threading time t at a fixed mutation satisfies $t < t_2$ if both sequences carry the mutation and $t > t_1$ if only one sequence carries the mutation. This can be achieved by modifying the SMC posteriors as described above.

However, this mutation-consistent approach is incompatible with Threads for two main reasons. First, Threads relies on segment-based TMRCA estimates instead of site-based, so that in practice it can have contradictory constraints on TMRCA estimation within the same segment. Second, Threads does not maintain an internal track of marginal topologies and mutation ages, which would lead to significant computational overhead.

In experiments replacing the ASMC subroutines used by ARG-Needle [70, 99] with a mutation-consistent and ARG-aware variant of ASMC, we observe minimal gains in inference accuracy. Weighed against the increased complexity and computational overhead of tracking the edge associated with each mutation on the ARG during inference, we found the simpler non-consistent threading model to be preferable to our mutation-consistent variant.

Chapter 4

Inference accuracy and computational performance

In this chapter we measure the accuracy and efficiency of the Threads inference algorithm in simulations and compare against three other inference methods, Relate [84], ARG-Needle [99], and tsinfer combined with tsdate [39, 95]. Using simulated ground-truth ARGs, we generated artificial mutations which we used to infer ARGs and measure inference accuracy in a range of different testing modalities.

4.1 Accuracy metrics

Before describing the simulations and their results in more detail, we briefly review the accuracy metrics used to evaluate the ARG inference algorithms. Here, we focus mainly on three scalable metrics, the Robinson-Foulds (RF) [75], the total variation (TV) [99] and the $\max r^2$-score, which we describe below.

We evaluate three further tree-based metrics, namely the TMRCA-metric [99], the topological Kendall-Colijn metric (KC) [40], and the split-size vector metric (SV) [82]. However, as the TMRCA, KC and SV require time and memory that is quadratic in tree size to evaluate, they are only evaluated in simulations of sam-

ple size up to 4,000. Chapter 2.1.3 gives a detailed description of the RF, TV, TMRCA, KC and SV metrics and of their limitations. In particular, it explains how the TMRCA, KC and SV are quadratically biased towards the root node which can result in limited informativeness for larger trees. The metrics RF, TV, TMRCA, KC and SV were evaluated across 200 randomly sampled marginal trees using the `arg_needle_lib` software [99]. Tsinfer and tsdate are the only methods that output polytomous trees, so to ensure a fair comparison, we have randomly broken polytomies when evaluating KC and RF, following the approach used in [99].

We also evaluate a novel, stochastic accuracy metric, which we call the $\max r^2$-score. The $\max r^2$-score is the only ARG accuracy metric that is not based on comparisons of marginal trees and is designed to give an estimate of ARG performance for genealogy-wide association studies (GeWAS) [99]. Given a ground-truth ARG A_0 and an inferred ARG A_1, we generate M_0 and M_1 mutations on those ARGs, respectively, by uniformly distributing them along the ARG volume. Then, for each mutation m_0 on A_0 we compute $\max_{m_1 \in A_1} r^2(m_0, m_1)$, finding the mutation that is most closely correlated with m_0 from among the M_1 mutations on A_1. Here, we identify mutations with the bitsets they subtend. The $\max r^2$-score is defined as the mean over all such maximum correlations.

$$\max r^2(A_0, A_1) \coloneqq \frac{1}{M_0} \sum_{m_0 \in A_0} \max_{m_1 \in A_1} r^2(m_0, m_1).$$

Given that a disease-causing mutation arose on A_0 sometime in the past, the $\max r^2$-score estimates how well we can expect to capture that mutation in a GeWAS study using the inferred ARG A_1. In our experiments we use $M_0 = M_1 = 10{,}000$ and report the $\max r^2$-error, $1 - \max r^2(A_0, A_1)$, so that a lower value reflects better performance for all accuracy metrics.

4.2 Simulation parameters

We evaluated accuracy under two different demographic models, an inferred demography for Northern Europeans from Utah (CEU) in the Thousand Genomes Project [3, 11] inferred using SMC++ [85, 89] and a constant demography of effective population size $N_e = 10{,}000$ diploids. All simulations were performed using msprime (version 1.2.0) [7].

In all simulations we used a fixed recombination rate of 1.3×10^{-8} per site per generation. This number was derived by averaging recombination rates across whole chromosome arms as estimated by the HapMap project [20]. The precise per-site-per-generation genome-wide human mutation rate is unknown, with estimates ranging between 1.2×10^{-8} and 1.65×10^{-8} [43, 62, 69, 79, 90, 91]. For our simulations, we have chosen a fixed genome-wide mutation rate of 1.4×10^{-8}. In each case, we simulated sequences of 15MB in length, with sample sizes ranging from $N = 250$ to $N = 16{,}000$. All accuracy metrics were evaluated only on the central 5MB region, discarding the 5MB to the left and right as burn-in. For each algorithm, we only scaled N up to what could by performed using reasonable computational resources, setting a runtime limit of around 4 hours for each method. We found this limit sufficient to give a clear picture of the behaviour of the different ARG inference metrics under the various metrics and simulation configurations. A detailed comparison of computational efficiency for all algorithms is given in Chapter 4.4. Each experiment was repeated for 5 different random seeds.

The two demographies we used lead to drastically different data. Figure 4.1 shows site frequency spectra and the number of segregating sites derived from each. Under the CEU demography, around a third of the simulated mutations are singletons, with proportionally fewer common mutations, while the spectrum for the constant demography is more balanced. The CEU demography also gives a

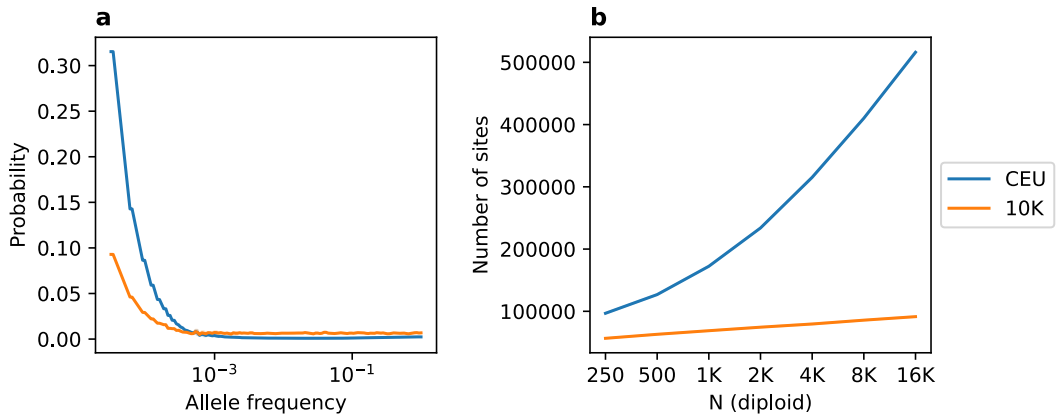


Figure 4.1: **a.** Site frequency spectra for the simulated European demography (CEU) and the constant-sized 10,000 diploid demography (10K). The frequencies were evaluated in a single simulation of a 15MB region for 16,000 diploid samples. **b.** The number of sites observed in each simulation as a function of sample size.

higher total number of segregating sites, with the largest simulations containing around 500,000 sites in the 15MB simulated region.

4.3 Simulation results

In this chapter we present results for inference accuracy and computational performance in simulations. Results for each of the four methods evaluated, Relate (version 1.2.1) [84], ARG-Needle (version 1.0.2) [99], tsinfer (version 0.3.1) and tsdate (version 0.1.5) [39, 95], and Threads, are truncated such that all experiments remain within a reasonable runtime while running on 8 CPU cores. For whole-genome sequences, we ran Relate up to 2,000 samples, tsinfer+tsdate up to 8,000 samples and Threads up to 16,000 samples. ARG-Needle is optimized for inference from genotyping arrays and was only evaluated in that setting.

Results for simulated whole-genome sequences simulated under constant and CEU demographies are presented in Figure 4.2. Under both constant and CEU demographies, Relate obtains the best performance on RF and SV, suggesting high

inference accuracy of tree topology. However, Threads outperforms Relate and tsinfer+tsdate on TV and on the $\max r^2$-score under a CEU demography. These results suggest that Relate may obtain more accurate tree topologies, but that Threads is better calibrated for recent coalescence time inference. Under the topological KC, tsinfer+tsdate outperforms the other two methods, while for the TMRCA, Threads is slightly outperformed by both Relate and tsinfer+tsdate.

Together, these results paint a nuanced picture where no single method outperforms the others in all measures. Our simulations suggest Threads performs well for ARG inference in recent time, while slightly lagging behind the other methods when it comes to deeper coalescence events. This observation is consistent with the difference in methodologies. ARG inference by threading as implemented by Threads and ARG-Needle relies on only a small batch of initial sequences to infer the deepest coalescence events, while Relate and tsinfer+tsdate jointly model all sequences. By contrast, the Li-Stephens approach employed by Threads shows excellent performance in capturing the long-range genetic relationships required to infer relatedness in recent time.

4.3.1 Genotyping arrays

ARGs inferred directly from genotyping arrays may be used in a range of applications in statistical genetics, including association studies, construction of genetic relatedness matrices and genotype imputation [99]. To compare the performance of Threads on genotyping array data to other inference methods, we subsampled the observed mutations in each simulation to match in frequency with array data from the UK Biobank and build ARGs directly from these simulated arrays. This subsampling procedure has previously been described in [70] and [99]. Out of the four methods evaluated, only ARG-Needle and Threads have purpose-made array settings. Tsinfer has a sparse-mutation variant which we evaluated on array data

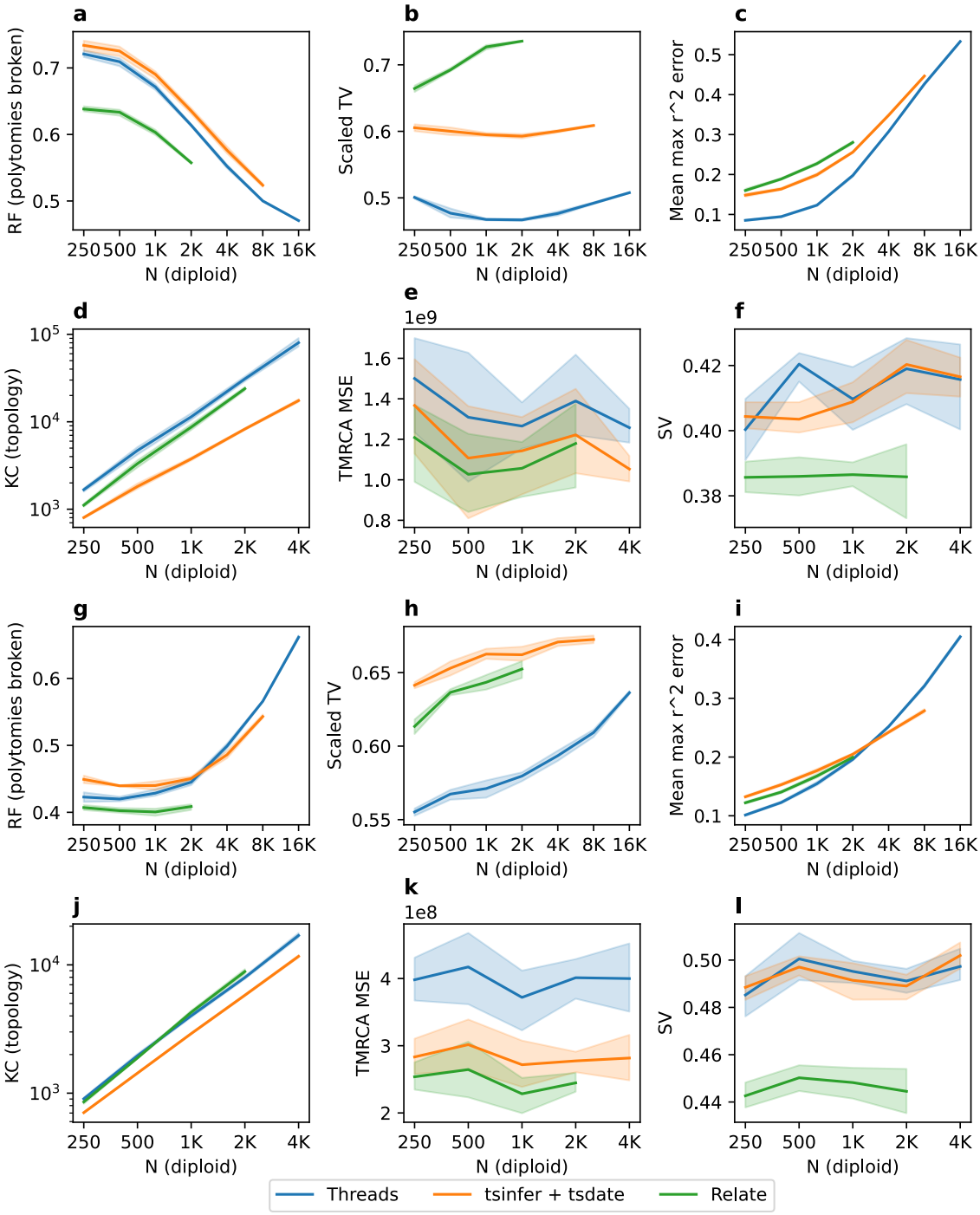


Figure 4.2: ARG-inference accuracy on simulated whole genome sequences. Results are truncated to remain within a reasonable runtime. Shaded regions show 95% confidence intervals across 5 random seeds. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized ($N_e = 10{,}000$) demography.

instead of the regular algorithm. Relate was run using the same parameters for both whole genomes and arrays. Results are presented in Figure 4.3.

Under the European demography, Threads and ARG-Needle outperform Relate and tsinfer+tsdate under the RF, TV and $\max r^2$ metrics. Threads and tsinfer+tsdate achieve the best results under KC and ARG-Needle outperforms other methods under the TMRCA and SV metrics. Under the constant-sized demography, Threads outperforms ARG-Needle under the RF and also achieves the best results under the KC. ARG-Needle still outperforms other methods under the SV. Under the $\max r^2$-metric no one method is significantly better than all others across all ARG sizes.

Together, these experiments suggest both ARG-Needle and Threads are strong candidates for array-based inference of ARGs, with either one or the other achieving the best results under each metric we evaluated.

4.3.2 Robustness to reduced mutation rate

To further test for robustness against data with little signal we repeated the experiments above with an artificially low mutation rate of 1.4×10^{-9}. Results are provided in Figure 4.4. Comparing with Figure 4.2, we see decreased accuracy across all measures, as expected due to the reduced signal in this setting. Consistent with its high performance on sparse genotyping array data, Threads appears highly robust to the low mutation rate compared with other methods, especially for the constant-sized demography.

4.3.3 Robustness to genotyping errors

To test for robustness to the presence of genotyping errors, we randomly flipped each entry in the simulated genotype matrices using a realistic error rate of 0.1% [93]. For CEU simulations where the minor allele count of most mutations is $<$

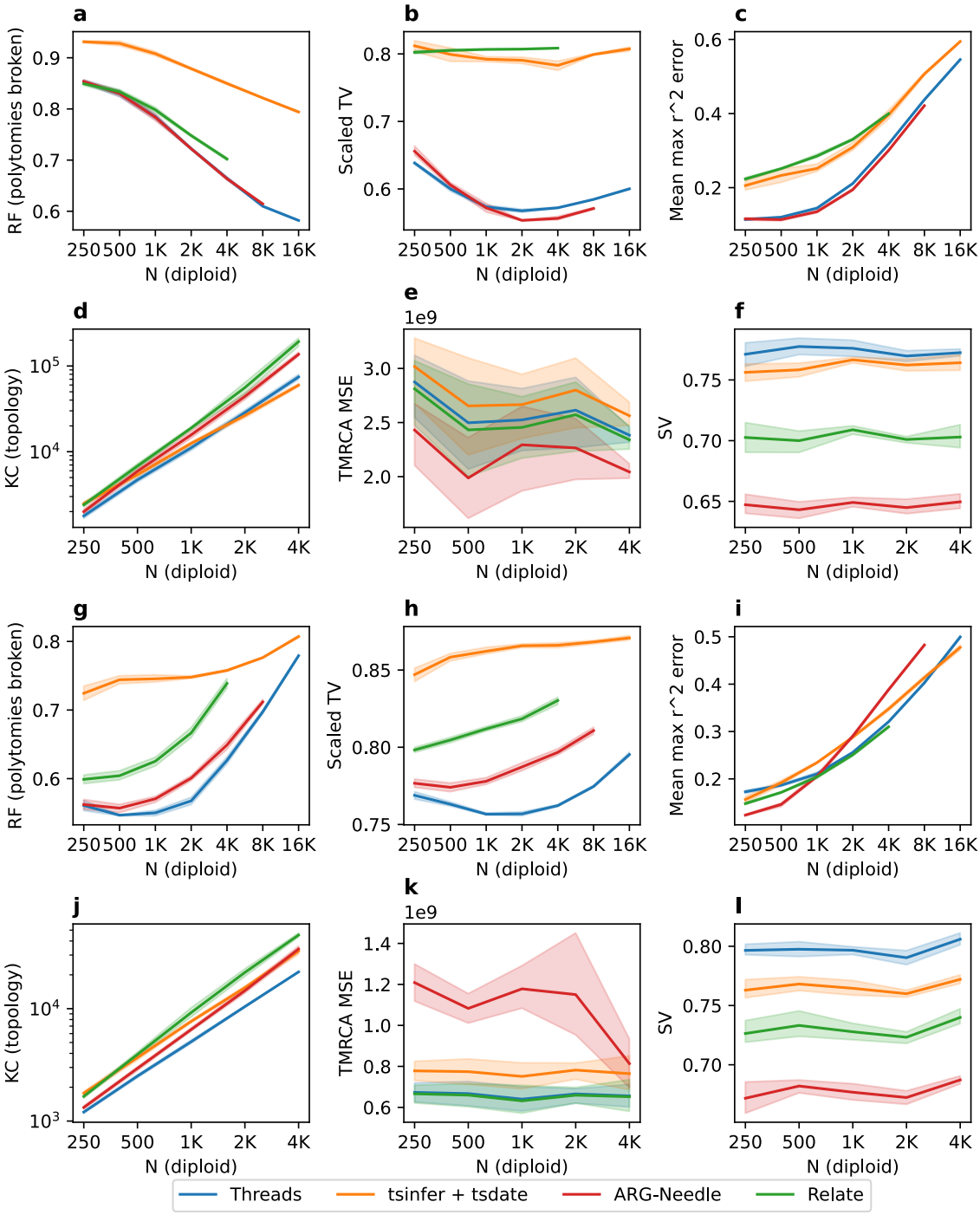


Figure 4.3: ARG-inference accuracy on simulated genotyping arrays. Shaded regions show 95% confidence intervals across 5 random seeds. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized ($N_e = 10{,}000$) demography.

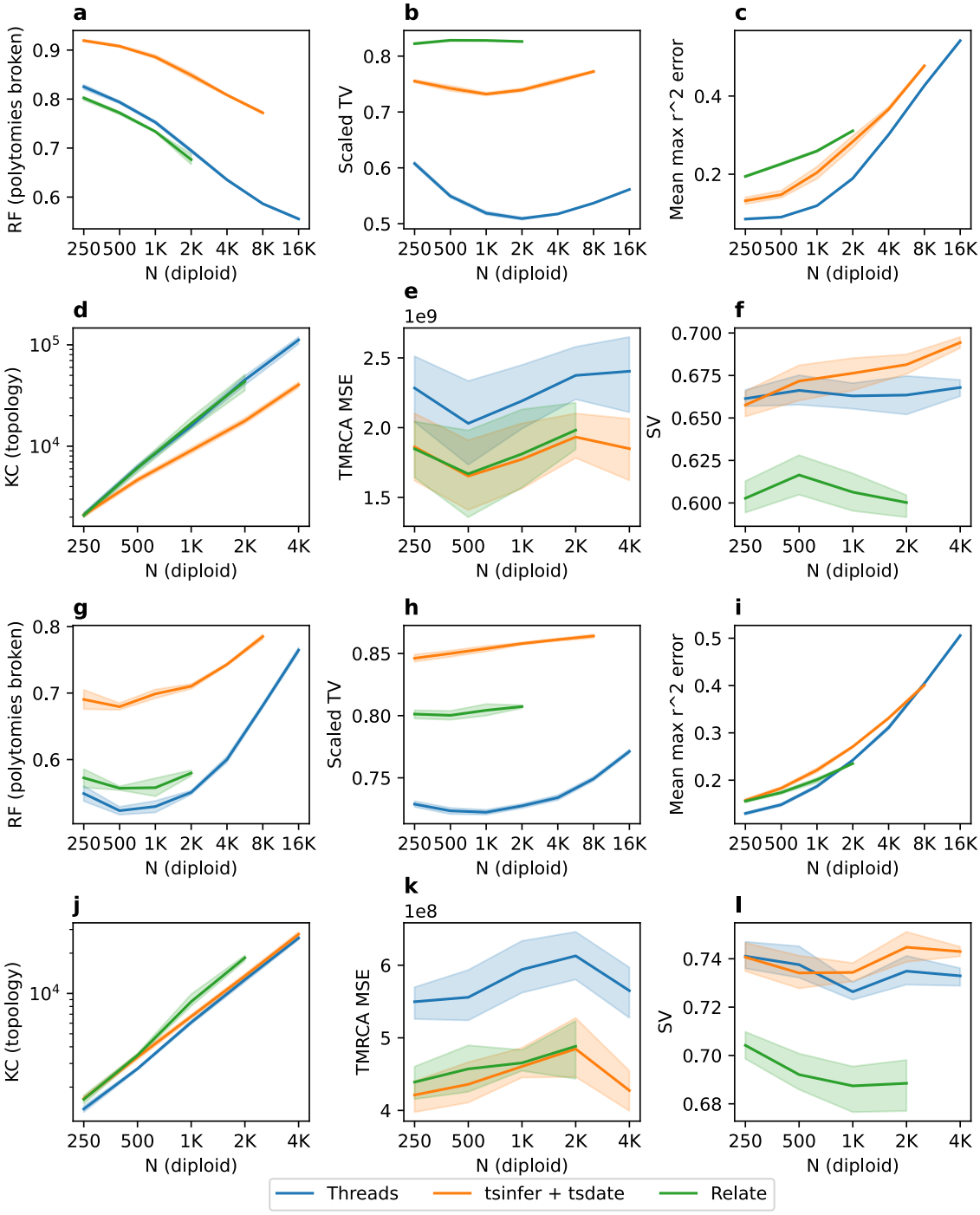


Figure 4.4: ARG-inference accuracy on simulated whole genome sequences with an artificially low mutation rate of $\mu = 1.4 \times 10^{-9}$ per site per generation. Shaded regions show 95% confidence intervals across 5 random seeds. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized ($N_e = 10{,}000$) demography.

10, this had a drastic effect, but less so for the constant-sized demography which has a more uniform site frequency spectrum. Genotype matrices derived from the simulated genotyping arrays were relatively unaffected. Results are given in Figure 4.5 for whole genomes and in Figure 4.6 for arrays.

Out of all methods, Relate was the most robust to genotyping errors, obtaining comparable results on most metrics with and without errors. For whole genomes, Threads achieved greater or equal accuracy to tsdate+tsinfer for all metrics except the KC under the CEU demography and the TMRCA under the constant-sized demography. For genotyping arrays, accuracy measurements with and without errors did not change much, though Threads proved slightly more robust to errors than ARG-Needle, achieving significantly better accuracy under the RF and TV metrics under both demographic models. These experiments indicate Threads achieves greater robustness to genotyping errors than ARG-Needle and tsinfer+tsdate.

4.3.4 Imputed data

We also evaluated the accuracy of Threads on imputed sequences. Using the simulation parameters from Chapter 4.2, we simulated genotyping array data for $4{,}000$ samples. We imputed these samples by applying IMPUTE5 [76] to simulated in-sample reference panels of size ranging from 100 to $30{,}000$ diploids. After imputing, we discarded variants with INFO score < 0.95 and rounded all remaining genotype dosages to the closest integer.

Surprisingly, we found that for under the CEU demography, we obtained lower accuracy under the RF, KC and TV with small imputation panels ($N \leq 3{,}000$) than using genotyping arrays only, while the TMRCA, SV and max-r^2 error monotonically improved with panel size (Figure 4.7a-f). This was not the case under the constant $N_e = 10{,}000$ demography, where we only observed performance degradation for the smallest panel and only under the KC metric (Figure 4.7g-l). These

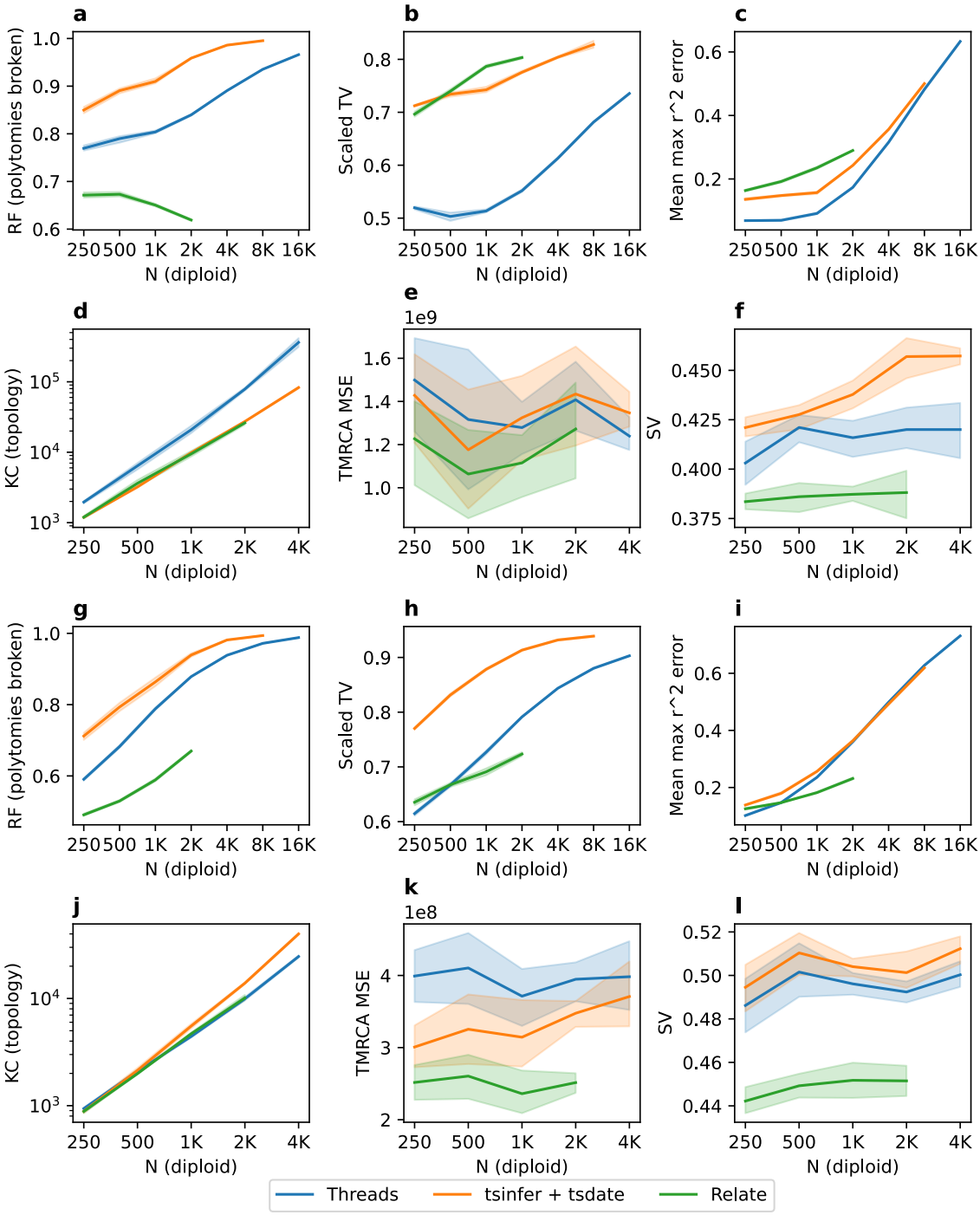


Figure 4.5: ARG-inference accuracy on simulated whole genome sequences with a 0.1% error rate. Shaded regions show 95% confidence intervals across 5 random seeds. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized ($N_e = 10{,}000$) demography.

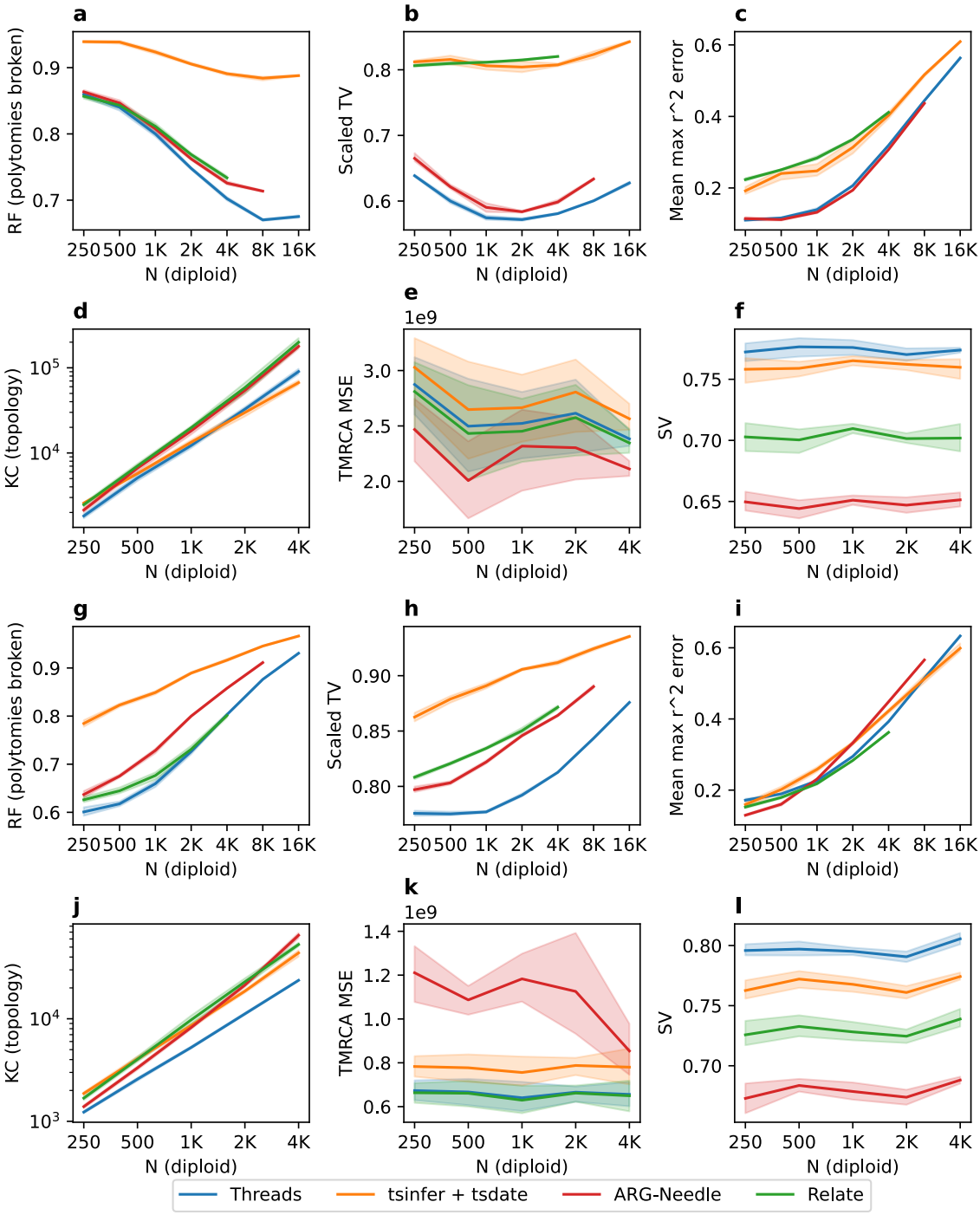


Figure 4.6: ARG-inference accuracy on simulated genotyping arrays with a $0,1\%$ error rate. Shaded regions show 95% confidence intervals across 5 random seeds. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized ($N_e = 10{,}000$) demography.

observations likely reflect the different site frequency spectra of the two simulation settings. Simulations under the realistic European demography contain a considerably higher proportion of ultra-rare variants that are more difficult to impute correctly (Figure 4.1a).

For the largest reference panel sizes, all metrics but the RF and TV converged to accuracy levels observed using ground-truth whole genomes. The RF and TV focus on accuracy in recent time which requires precise observations of rare variants. Our results suggest even larger reference panels than the ones simulated here are needed to obtain the same level of accuracy in recent-time inference as using ground-truth whole genomes, but that ARG inference from imputed variants using Threads is nevertheless a viable strategy, especially when large reference panels are available.

4.3.5 Seeded ARG inference

Based on the high accuracy achieved by Threads in recent-time ARG inference and accurate deep time inference using Relate, we hypothesized that we could leverage the modular nature of the threading instructions inferred by Threads to combine both methods and achieve high accuracy across the whole coalescence time spectrum. Threads can be combined with any other ARG inference method through the process of seeding, whereby an ARG inferred using one method for a subset of the data set is used as a scaffold to which the remaining samples are threaded. A more detailed discussion of seeding is provided in Chapter 3.5.5.

Here, we again focused on the simulations described in Chapter 4.2, applying Relate to the first 50, 100, 200, 500, and 1,000 samples to infer seed ARGs, threading the remaining samples to those ARGs using threading instructions inferred by Threads and then evaluating accuracy compared with the simulated ground truth.

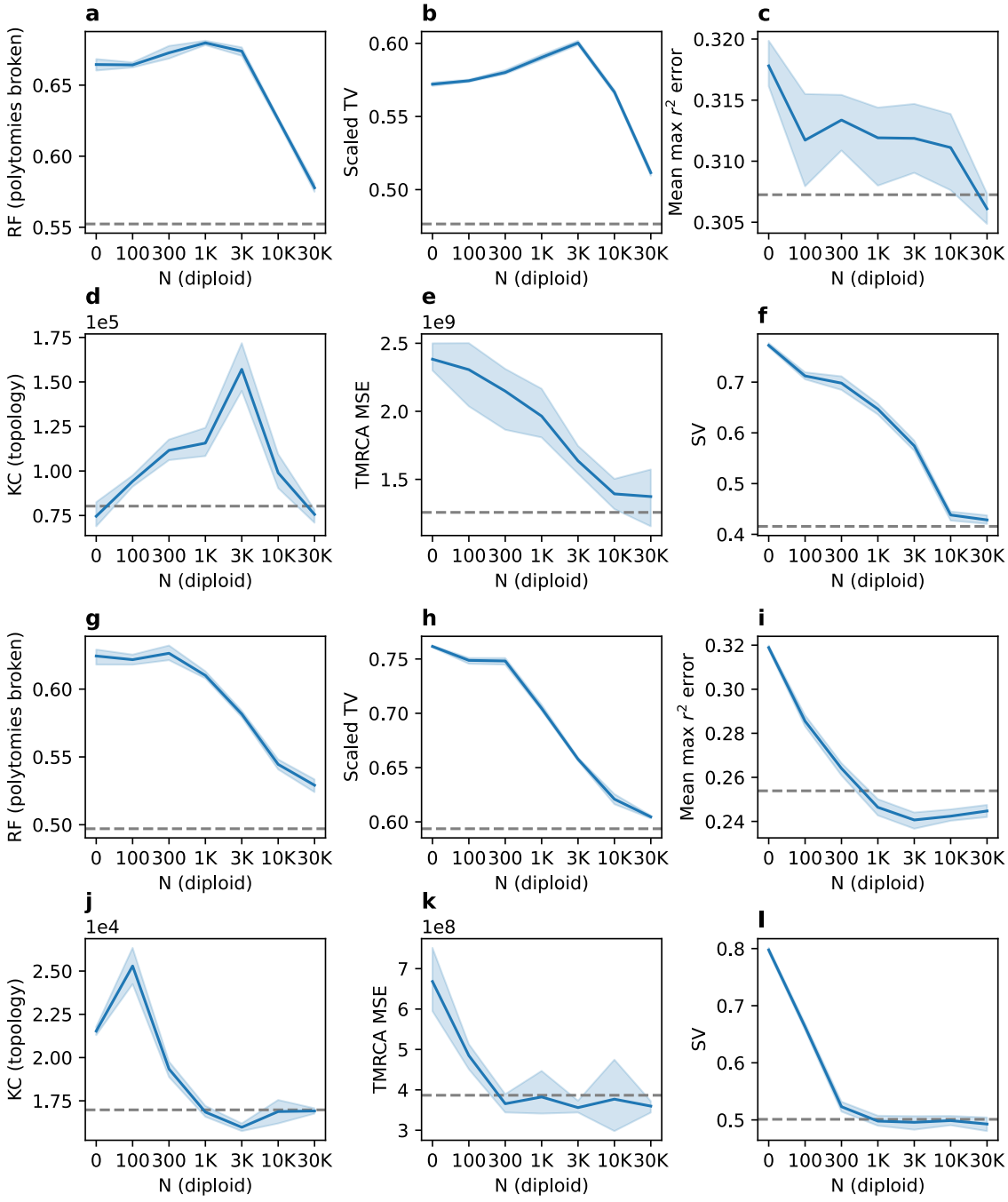


Figure 4.7: Inference accuracy of Threads using 4,000 simulated imputed sequences. The x-axis shows the reference panel sizes. The case $N = 0$ indicates inference directly from genotyping arrays. Dotted lines represent accuracy obtained from inference of ground-truth whole genomes. Shaded regions show 95% confidence intervals across 5 random seeds. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized ($N_e = 10{,}000$) demography.

Our results, in Figure 4.8, suggest further methodological development is required before combining the two methods, as in many cases we found the seeded inference to perform worse than either Relate or Threads. It is possible that the poor performance of the seeded ARGs is due to different marginal distributions of coalescence times between the two methods, meaning the threading instructions are not correctly calibrated for the seed ARG. In this case, an additional normalization step, e.g. as described in [99], might improve concordance between the two methods when applied to each in turn. However, further research is required to determine the causes of the poor accuracy we observe in seeded ARGs and to ascertain if and how they may be improved.

4.4 Computational performance

In all simulations described above, we also measure computational efficiency by comparing wall time, maximum memory usage and disk space required to write results. Threads, tsinfer+tsdate and Relate allow for efficient parallelization, while ARG-Needle does not currently support multi-threading. All methods were given 8 CPU cores. Each algorithm uses a distinct file format to store the inference results. For ARG-Needle, these are `.argn` files, while Relate uses `.anc` and `.mut` files and tsinfer+tsdate use the `.tsz` format. Threads uses its own `.threads` format, which we describe in more detail in Chapter 5. Disk space usage is evaluated by measuring the size of each of these output files.

In Figure 4.9, we compare computational performance on simulated whole genomes without genotyping errors. Here, Threads achieves speed-ups of up to an order of magnitude compared with Relate and tsinfer+tsdate, while achieving significantly better scaling with memory and consuming around five times less disk space to store results. With the presence of genotyping errors, Threads achieves

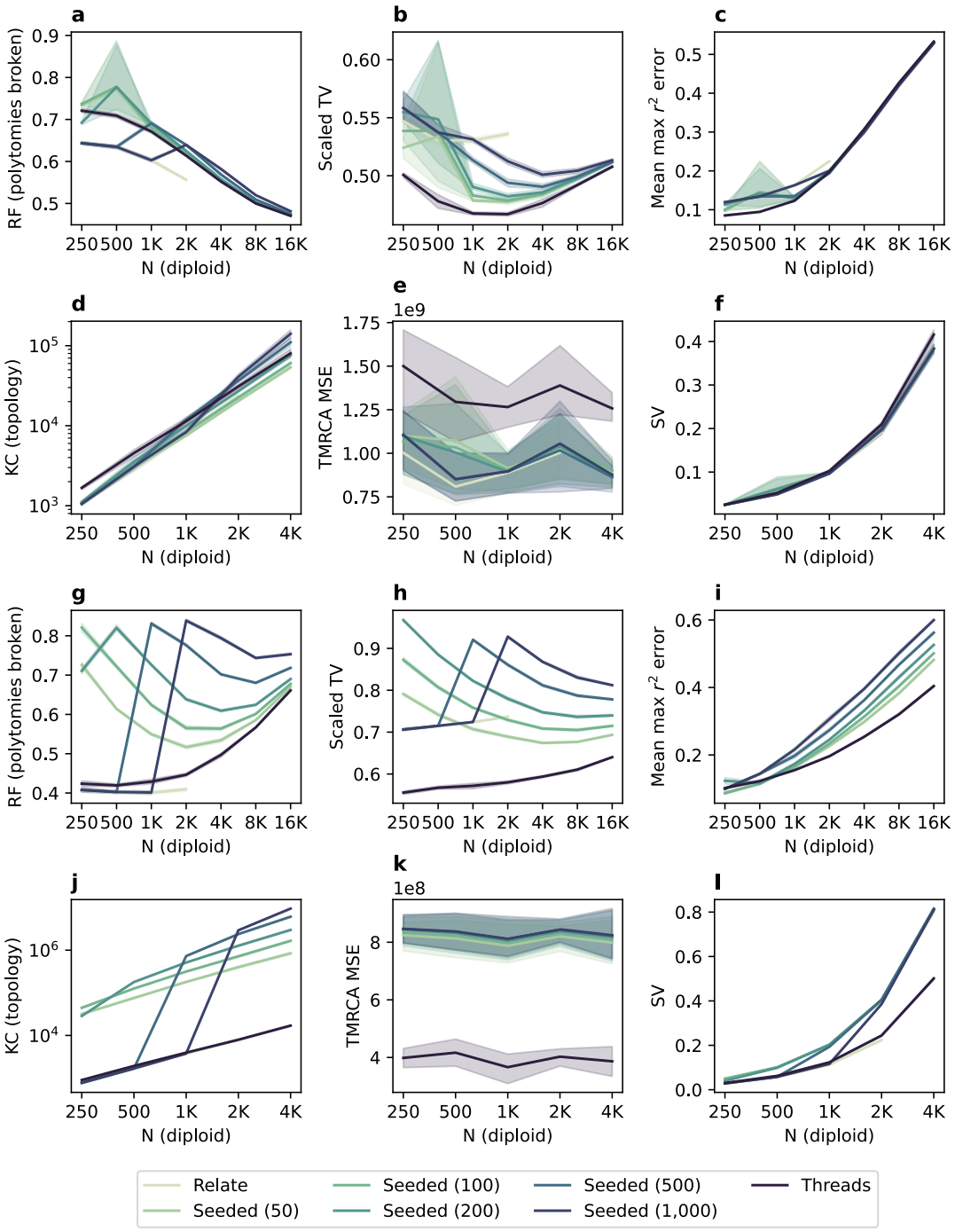


Figure 4.8: Accuracy of seeded ARG inference. "Seeded (x)" refers to seeded ARG inference whereby a scaffold ARG for the first x sequences is inferred using Relate and the remaining samples are added using Threads. Shaded regions show 95% confidence intervals across 5 random seeds. Relate is truncated at 2,000 samples due to computational constraints. **a-f.** Results for a European (CEU) demography. **g-l.** Results for a constant-sized $(N_e = 10{,}000)$ demography.

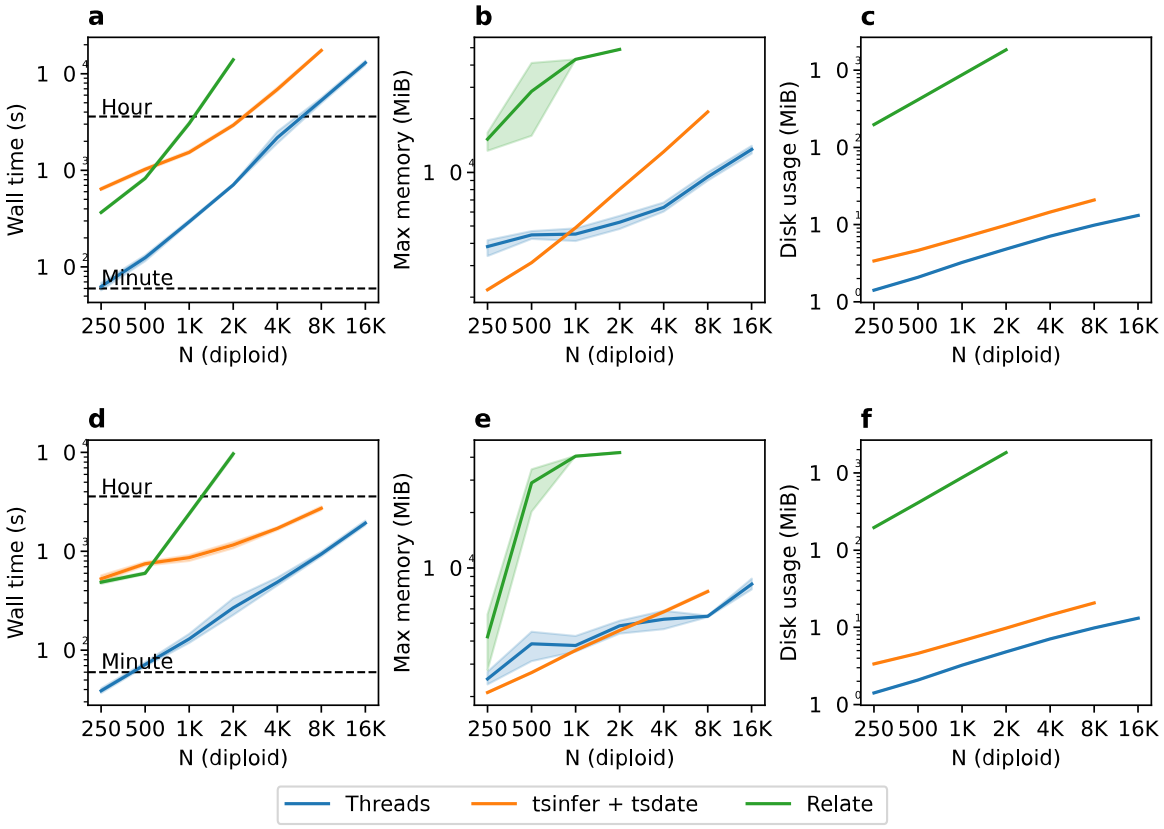


Figure 4.9: Computational performance on simulated whole genome sequences. All methods were given 8 threads. Shaded regions show 95% confidence intervals across 5 random seeds. **a-c.** Results for a European (CEU) demography. **d-e.** Results for a constant-sized ($N_e = 10{,}000$) demography.

even greater speed-ups compared to Relate and tsinfer+tsdate, consuming a similar amount of memory as tsinfer+tsdate and significantly less disk space (Figure 4.10). These results do not include temporary files written by Relate, which can use up to an order of magnitude more disk space than the final results.

As shown in Figure 4.11, Threads again achieves speed-ups of up to an order of magnitude on genotyping arrays compared with tsinfer+tsdate, while consuming a similar amount of memory and disk space. Some of the advantage in inference speed achieved by Threads compared to tsinfer+tsdate is due to its efficient parallelization. Figure 4.12 shows computational performance using only a single thread for each method. In this setting, Threads still obtains a significant improvement in

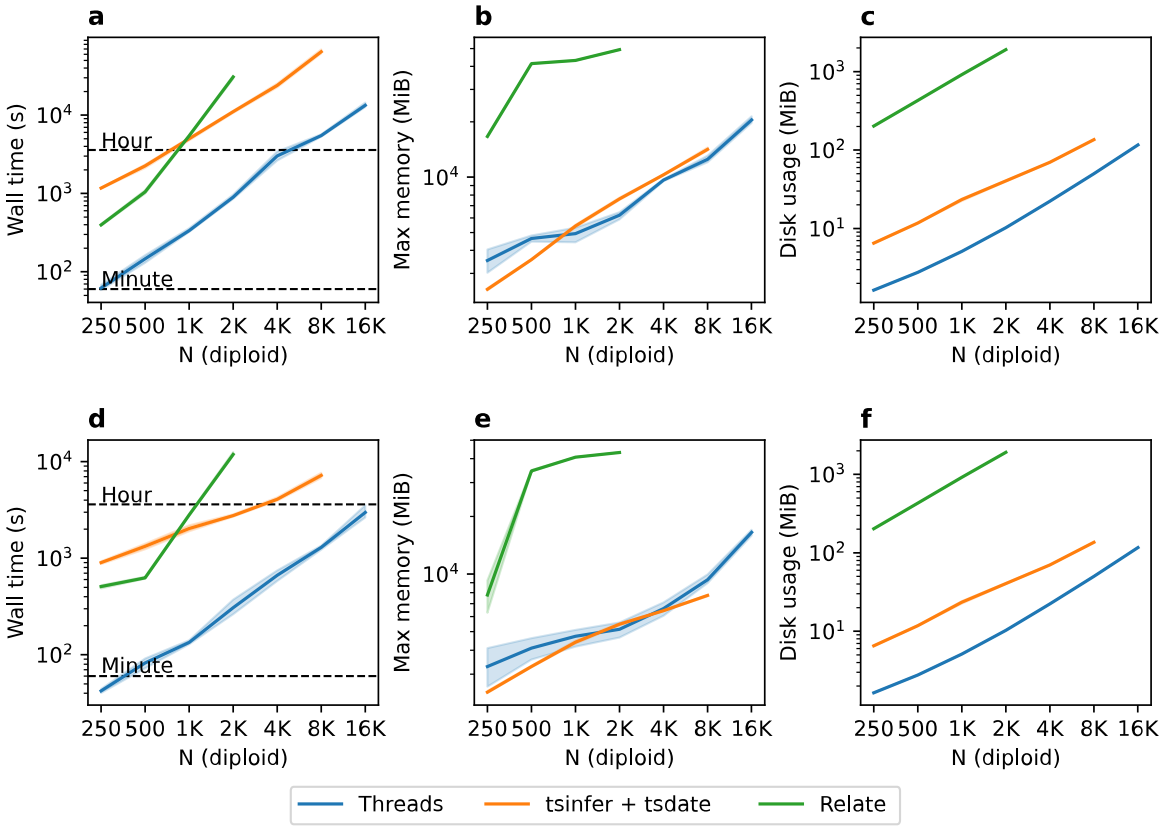


Figure 4.10: Computational performance on simulated whole genome sequences with a 0.1% error rate. All methods were given 8 threads. Shaded regions show 95% confidence intervals across 5 random seeds. **a-c.** Results for a European (CEU) demography. **d-e.** Results for a constant-sized ($N_e = 10{,}000$) demography.

both inference time and memory usage across all simulation configurations. These results, together with the accuracy benchmarks in Chapter 4.3 show Threads is an efficient, versatile, and accurate tool for ARG inference, running up to an order of magnitude faster than other methods without compromising inference accuracy.

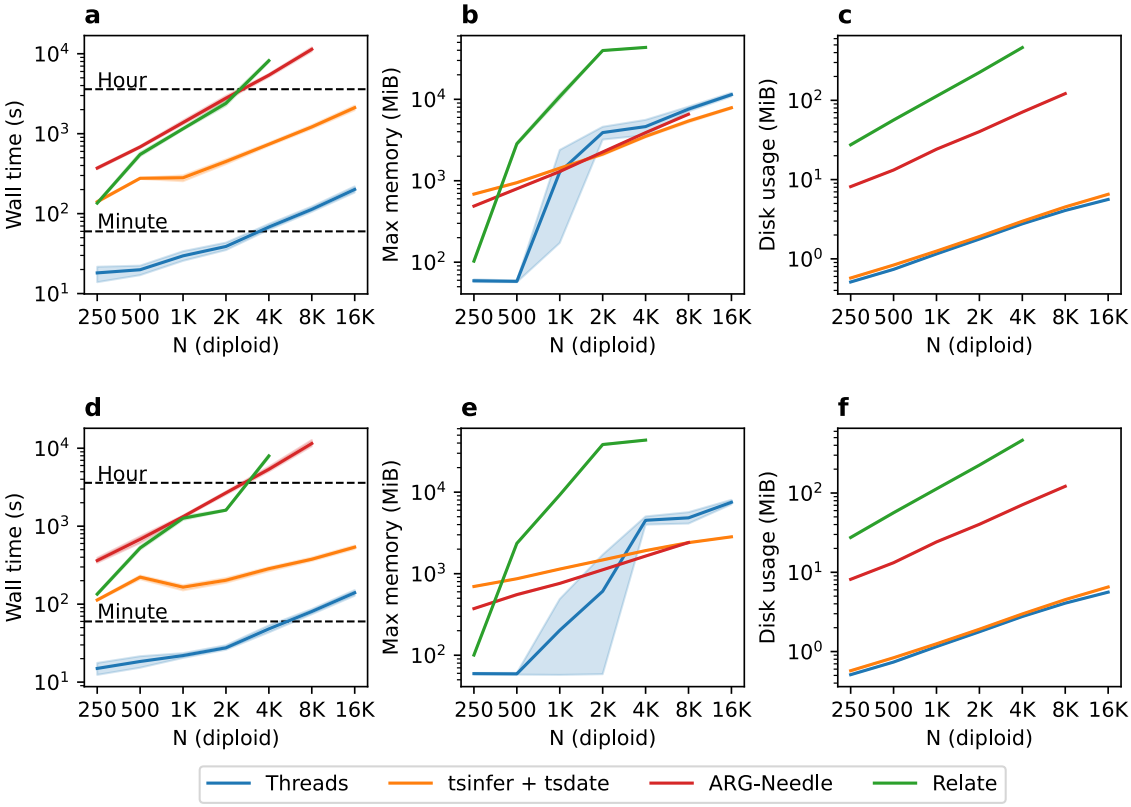


Figure 4.11: Computational performance on simulated genotyping arrays. All methods were given 8 threads. Shaded regions show 95% confidence intervals across 5 random seeds. **a-c.** Results for a European (CEU) demography. **d-e.** Results for a constant-sized ($N_e = 10{,}000$) demography.

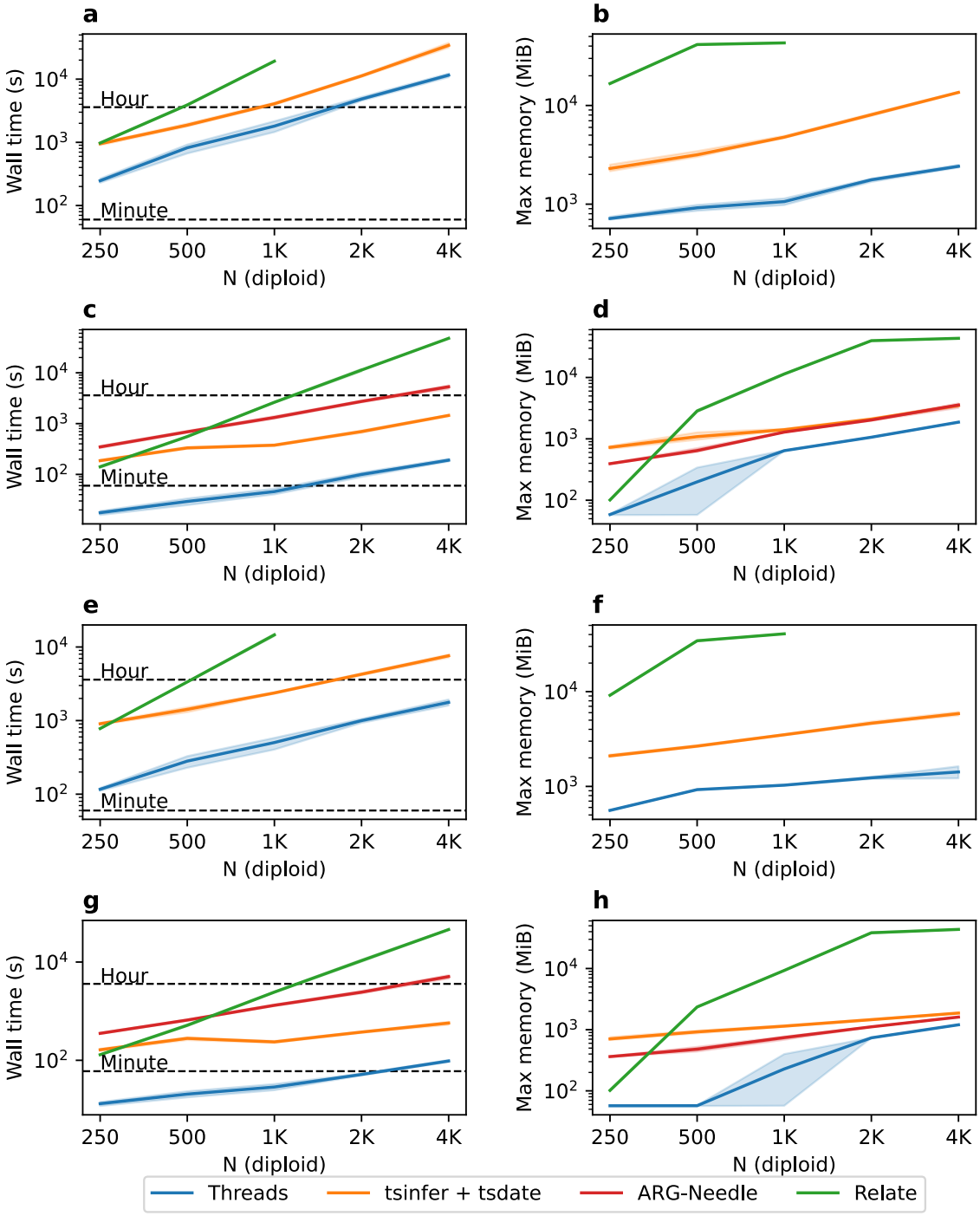


Figure 4.12: Computational performance using a single thread. Shaded regions show 95% confidence intervals across 5 random seeds. **a-b.** European (CEU) demography on whole genome sequences. **c-d.** CEU demography on genotyping arrays. **e-f.** Constant-sized ($N_e = 10{,}000$) demography demography on whole genome sequences. **g-h.** Constant-sized demography on genotyping arrays.

Chapter 5

Genotype compression

In Chapter 4 we showed how Threads achieves excellent computational performance on ARG inference tasks, including low usage of disk space to store the inferred ARGs. This storage efficiency opens up the door to applications in compression of both genotypes and their associated ARGs. The use of ARGs to compress genotype data was first explored with tsinfer by Kelleher et al. in [39]. The approach is based on the observation that genotypes in population-scale data sets can be efficiently represented by mutation events on edges within a compactly structured ARG, reducing the computational space and memory required to store large genotype data sets. However, genotyping and phasing errors were observed to lead to an excess of branches in ARGs inferred using tsinfer. Thus, while achieving good compression rates on simulated data, tsinfer was not shown to improve upon traditional genotype compression methods such as the compressed VCF [15] in real data.

In this chapter we show how Threads achieves greater compression rates than both standard genotype formats and ARGs inferred using tsinfer, storing not only the genotypes but also their complete genealogical history. We show that our thread-

ing-based compression of biallelic SNPs is robust to genotyping errors and that it outperforms traditional compression formats on both real and simulated data.

5.1 Genotypes as threading instructions

Recall that the output of ARG inference via threading consists of threading instructions. For each $n = 1, \ldots, N$, these consist of a map $[L_1, L_2] \to R_+ \times \{1, \ldots, n-1\}$, assigning to each site in a genomic region $[L_1, L_2]$ a threading target and a coalescence time. When inferred using Threads, these maps are piece-wise constant with few breakpoints and may be stored efficiently by only noting the breakpoints and their associated threading targets and coalescence times.

By construction, the threading target chosen within each such segment minimizes the number of mismatches between the input sequence and its targets. As the number of samples grows, mismatches become rarer and matches longer. This intuition motivates the idea of storing mutations as part of the threading instructions by simply noting each site where the input sequence and its threading target are heterozygous. More formally, for sequence n with inferred threading instructions T_n, we can store the tuple (T_n, C_n), where $C_n \subseteq [L_1, L_2]$ satisfies the property that for each $l \in C_n$, sequence n and its threading target at l are heterozygous at l.

Setting C_1 to simply be the set of sites where the first sequence differs from the null sequence $(0, \ldots, 0)$, the collection $(T_1, C_1), \ldots, (T_N, C_N)$ thus uniquely encodes an ARG and the complete set of mutations used to infer it. However, as discussed in chapter 3.5.6, the mutations are not guaranteed to map to a unique branch in the inferred ARG. These data, once serialized and compressed using HDF5-compression [44], make up the contents of the `.threads` files introduced in Chapter 4.4 and are used to store all output from Threads.

The threading instructions along with the sets of heterozygous sites define a topological structure similar to an ARG. Marginally at each site, they give an oriented graph with nodes representing the N sequences in the sample and an edge from i to j if and only if j is the threading target of i at the fixed site, giving a tree structure rooted at 0. Finally, the edges carry a "mutation" if sequences i and j are heterozygous at the site. To read out the genotype of sequence i, we trace the graph from node i back to 0, which we store as a reference genome, keeping track of the number of mutations along the path. An illustration of this structure, which we call a *threading tree*, is given in Figure 5.1. Threading trees such as the one in Figure 5.1 can persist over long genomic regions, implying that the genotype vectors at multiple sites can efficiently be derived using a single tree traversal.

The threading tree data structure also allows for efficient partial queries of the genotype matrix. For example, the genotypes of a single sample may be reconstructed by tracing all of its threading instructions back to the root sequence. By contrast, traditional genotype formats such as VCF [15], pgen [14], or bgen [4] files are indexed by site, which can make sample-wise queries less efficient.

5.2 Simulation studies

In this chapter we attempt to quantify the efficiency of the Threads compression on simulated genotypes. We compare compression rates and the time required to parse genotypes with `.tszip` files generated by tsinfer and tsdate and with the PLINK2 `.pgen` format. We use the same simulation parameters as in Chapter 4.2, with both a typical European (CEU) and a constant-sized ($N_e = 10{,}000$) demography on simulated whole genome sequences and genotyping arrays with and without genotyping errors. In experiments with genotyping errors, we used a realistic error rate of 0.1% [93]. For whole genomes under the CEU demography, we found

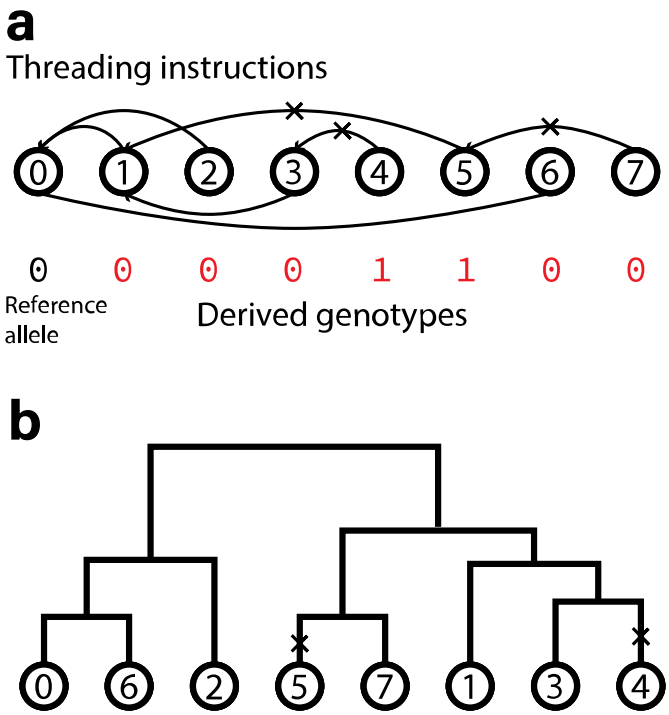


Figure 5.1: **a**. Toy example of the tree structure used to derive allele states for eight sequences from threading instructions at a fixed site. Circles depict indexed sequences in threading order and arrows depict threading target relations. The $\times$ symbols indicate the two sequences are heterozygous at the site. Only the 0-allele for sequence 0 is stored in memory, the rest are obtained by traversing the tree. Coalescence times, which are also stored as part of the threading instructions, are not represented in this figure. **b** A coalescence tree representing the same data as **a**.

tsinfer+tsdate could only scale to 8,000 samples within the runtime limits we set in Chapter 4 and so results are truncated to that scale. Chapter 4.4 gives a more detailed discussion of the computational performance of the ARG-inference algorithms.

For each simulation setting and for each software/file format combination we run three benchmarks. First, we compare the disk space required to store genotypes using `.threads`, `.tszip` and `.pgen` formats. The first two also store the inferred genealogical history of the sample, while the `.pgen` files only store the observed genotypes (Figure 5.2). Our results show the ARG-based formats achieving equal or better scaling than the `.pgen` compression in all settings. In Figure 5.2b, depict-

ing compression rates of realistic whole genome sequences with a high error rate, we replicate the findings from [39] that tree-sequence based compression is outperformed by traditional formats. However, `.threads` compression consistently outperforms both `.tszip` and `.pgen` at the largest scales. Compression with `.threads` may be made even more efficient by discarding information on coalescence times, keeping only the ARG topology and mutations. We have chosen to include coalescence times as most relevant applications of ARGs include both coalescence times and the topology.

Second, we measure the time needed to load the compressed objects into memory. For `.threads` files, we measure both the time needed to load threading instructions into memory to construct the threading trees and the time required to assemble the threading instructions into a full ARG using the threading algorithm implemented in `arg_needle_lib` [99]. For `.tszip` files, we measure the time required to build a full `tskit`-readable tree sequence from the compressed data using the `tszip` Python library. Results, in Figure 5.3, suggest that the sequential initialization algorithms used for threading trees and ARGs from `.threads` files are not as efficient as the initialization of tree sequences. The `arg_needle_lib` threading algorithm in particular struggles with whole genomes with high error rates.

Finally, we measure the time needed to parse the full genotype matrix into a Python object. This operation is applied directly on the threading trees and the tree sequences in the cases of `.threads` and `.tszip` files. For `.pgen` files, we use the `pgenlib` library. In Figure 5.4 we observe how `pgenlib` consistently achieves best or tied-for-best performance. The `tskit` genotype matrix construction is in almost all cases more efficient than the Threads construction, but is disproportionately affected by genotyping errors. This may be because genotyping errors increase the number of edges in the tree sequence, slowing down traversal.

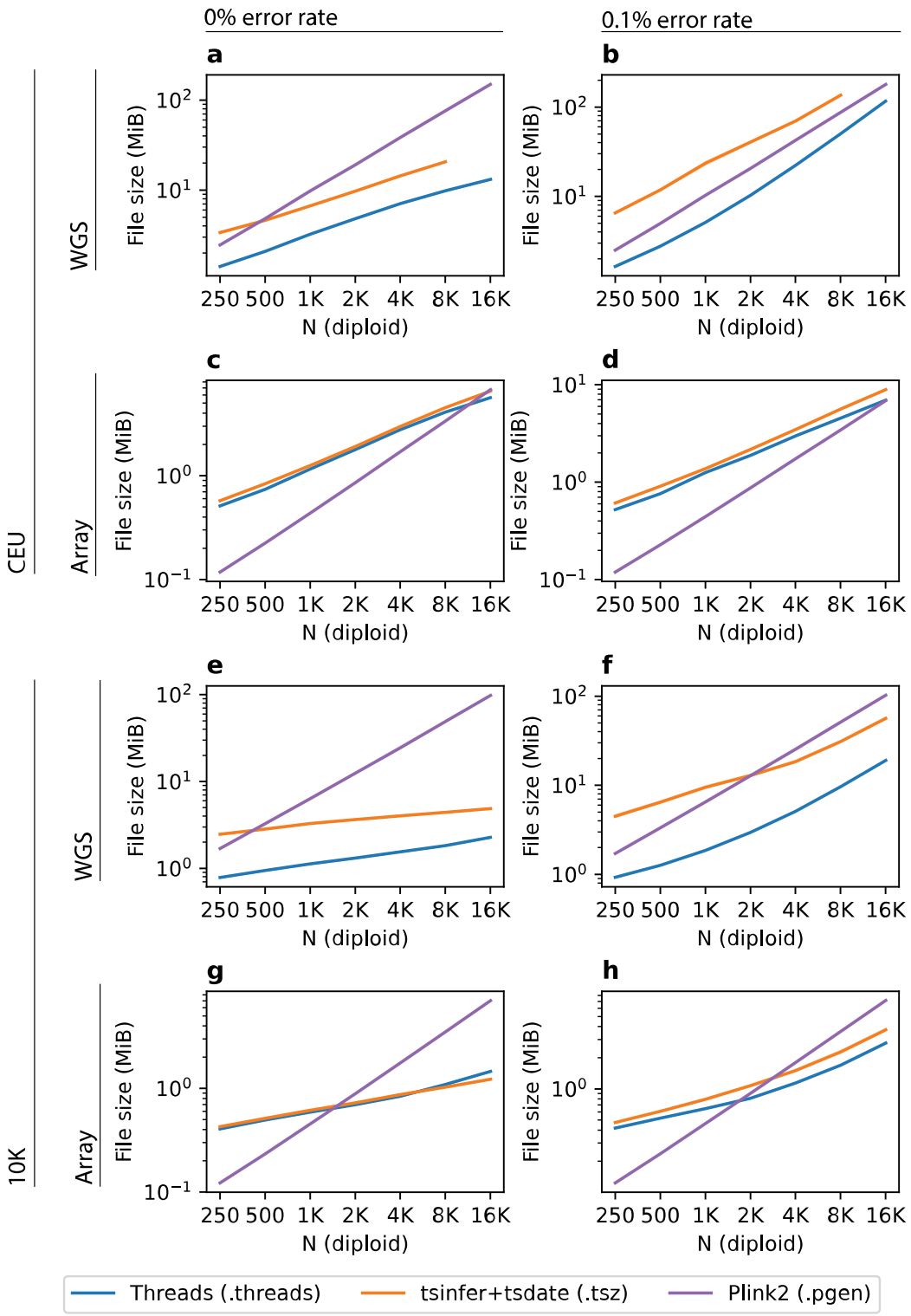


Figure 5.2: Disk space required to store a simulated 15MB genomic tract using Threads, tsinfer+tsdate and PLINK2 compressions.

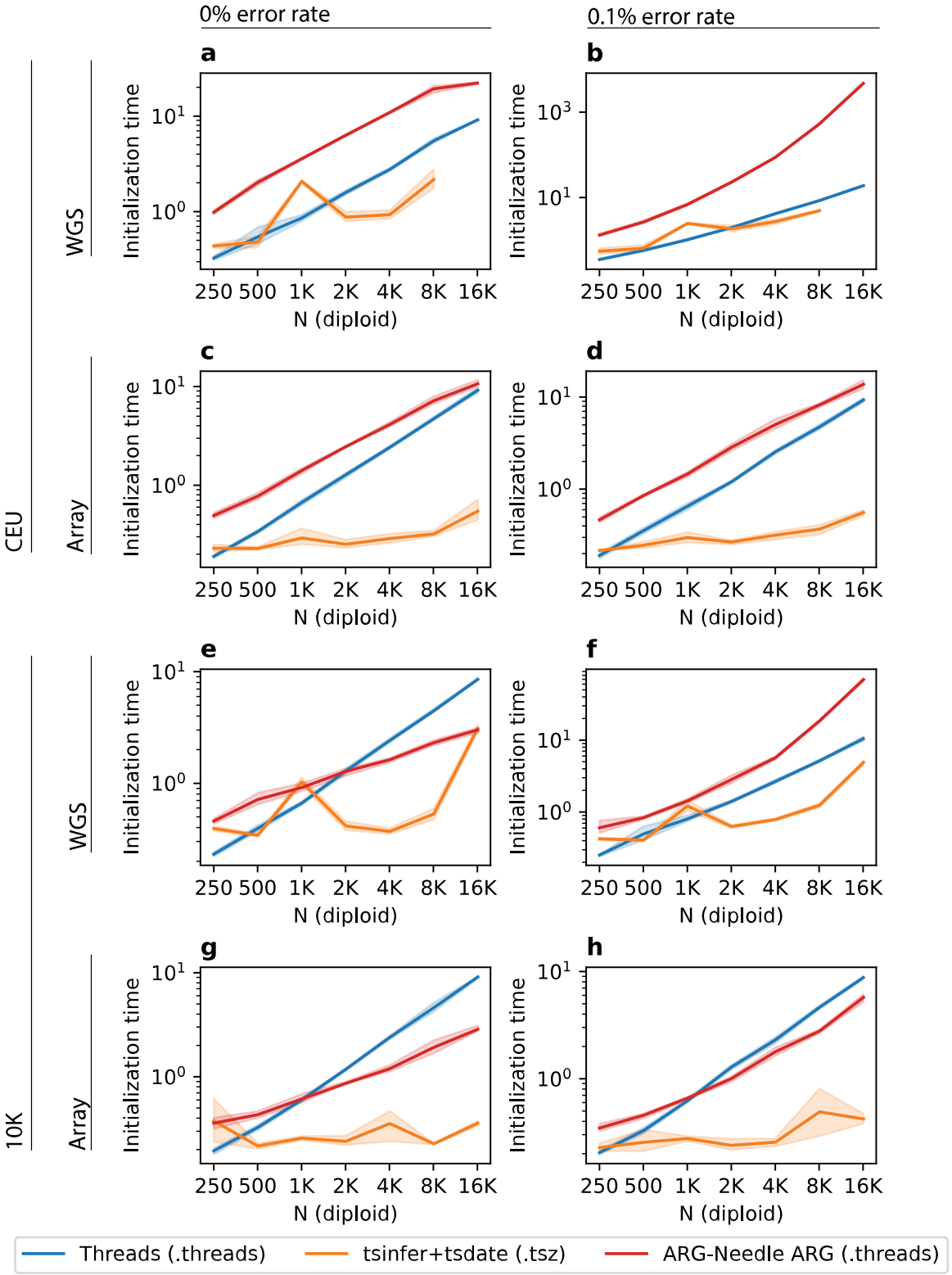


Figure 5.3: Time required to initialize tree sequences, threading trees and full ARGs. Shaded regions show 95% confidence intervals across 5 random seeds.

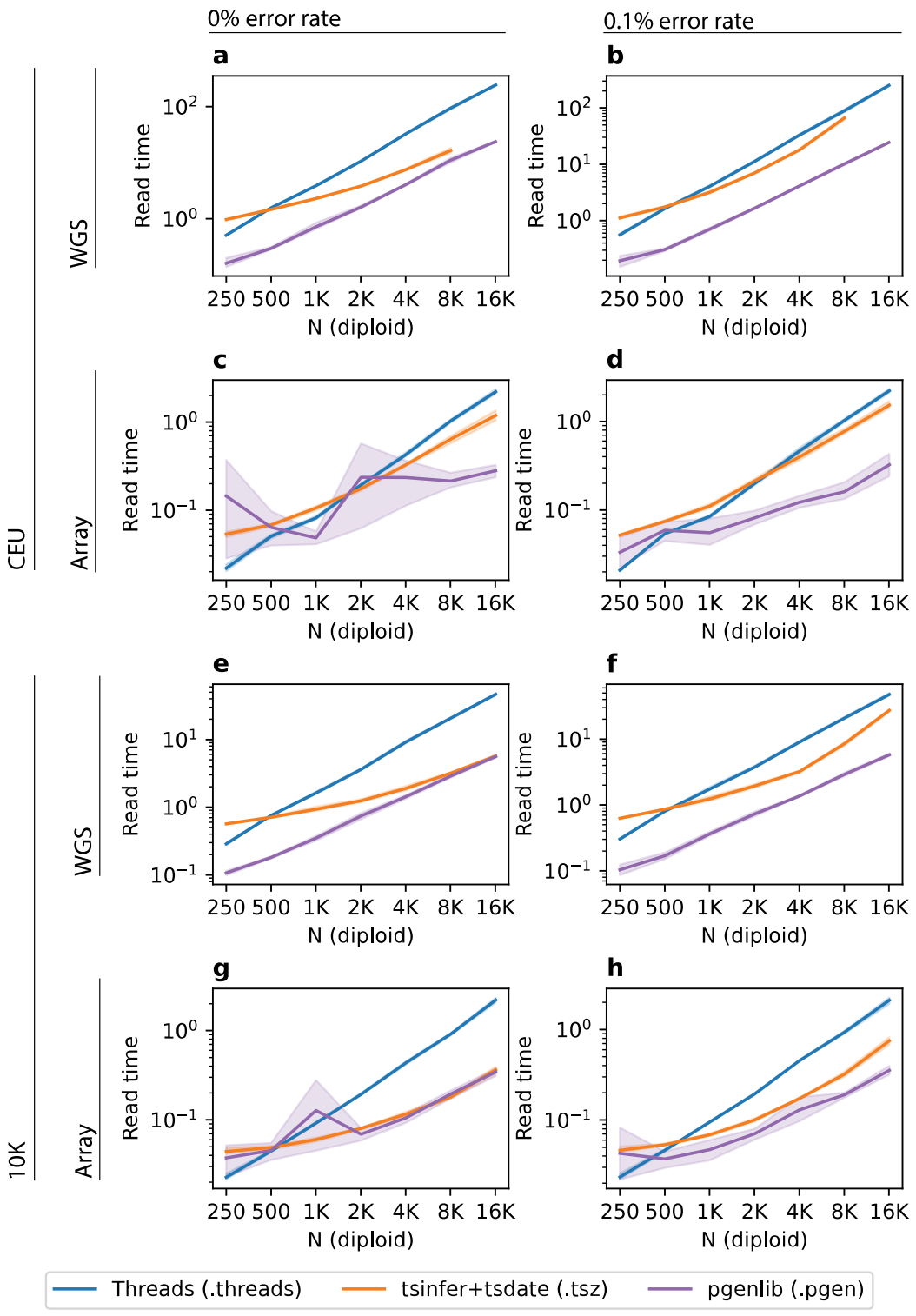


Figure 5.4: Time (in seconds) required to read the full genotype matrix from threading trees, tree sequences and directly from a pgen-compressed file. Shaded regions show 95% confidence intervals across 5 random seeds.

Together, our simulation studies suggest that Threads-inferred threading instructions are a robust and efficient compression of binary SNPs, compressing better than both tree sequences and traditional formats, with some compromises for the time required to initialize and read genotypes.

5.2.1 Genotype dosages

Thus far we have only evaluated compression of simulated binary SNPs, but many widely-used data sets rely on imputed genotype dosages, i.e., alleles that may consist of floating point numbers. To extend the `.threads` format to account for imputed dosages, we take the full, phased dosage matrix and round each entry to the nearest integer and apply Threads to the result. We then store the difference between this rounded matrix and the true dosage matrix as a sparse matrix of floating-point numbers.

This work-around achieves higher compression rates than traditional formats and includes an approximate genealogy for each variant. To evaluate the compression rates we simulated 5,000 diploid samples using the parameters described in Chapter 4.2. Out of these, we used 1,000 samples as a reference panel to impute parts or the whole of the remaining 4,000 using IMPUTE5 [76]. We compressed the output into `.vcf.gz` [15], `.pgen` [14], and `.bgen` [4] files, in addition to the `.threads` compression. Figure 5.5 shows how Threads achieves the greatest compression rates for imputed data. However, higher compression rate comes at a cost to read speeds, as observed in Figures 5.4 and 5.3, which for many applications is more important than the level of compression.

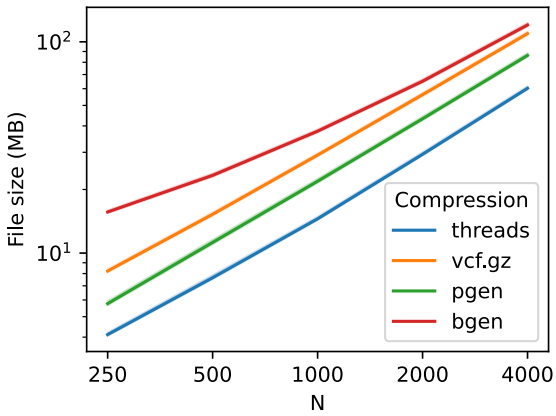


Figure 5.5: Compression of genotype dosages using traditional genotype formats as well as Threads compression.

5.3 Real data

In [39], Kelleher et al. show that tsinfer achieves good compression rates on simulated data but observe that it struggles to improve over traditional formats on real data, which suffer from genotyping and phasing errors. In this chapter we replicate those results and show that unlike tsinfer, the compression gains observed in Threads extend to real data. We do this using two publicly available data sets, the 1000 Genomes [3, 11] and the UK Biobank [10].

5.3.1 1000 Genomes

The 1000 Genomes data set [3, 11] contains 2,504 high-quality whole genome sequences of diverse ancestry. In this section we use these sequences as a test set for compression rates achieved by ARG inference methods and traditional genotype formats on whole genome sequences.

First, we inferred an ARG for chromosome 20 using the full data set using both Threads and tsinfer+tsdate. We then compared the sizes of the resulting `.threads` and `.tsz` files, each encoding both the full genotype matrix and an inferred ge-

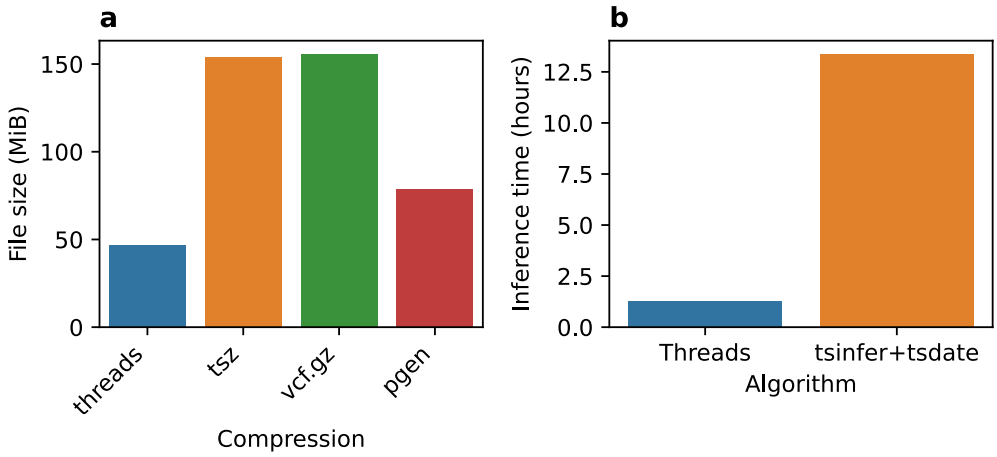


Figure 5.6: **a**. File sizes for 1,227,802 biallelic SNPs from chromosome 20 for 2,504 samples from the 1000 Genomes Project using two traditional genotype compression formats as well as tree sequences inferred using tsinfer+tsdate and threading instructions inferred using Threads. **b**. Inference time for the ARG of chromosome 20 in the 1000 Genomes data set. The two chromosome arms were inferred separately and inference times added together. Each method had access to 8 CPUs.

nealogical history of each variant, to traditional compression using `.vcf.gz` and `.pgen` files generated using bcftools [16] and PLINK2 [14], respectively. Our results, in Figure 5.6, show that Threads achieves greater compression of genotypes and ARGs than any other format, while also inferring ARGs around an order of magnitude faster than tsinfer+tsdate.

Here, both methods were run using a constant-sized demography of $N_e = 20{,}000$ and a mutation rate of 1.4×10^{-8} per site per generation. Threads used a genetic map based on HapMap 3 [20]. Tsinfer+tsdate does not accept an explicit genetic map so instead we used a constant recombination rate of 1.3×10^{-8} per site per generation.

5.3.2 UK Biobank genotyping arrays

To evaluate compression on genotyping arrays, we used Threads to infer ARGs directly from genotyping arrays for 337,464 unrelated white British samples from

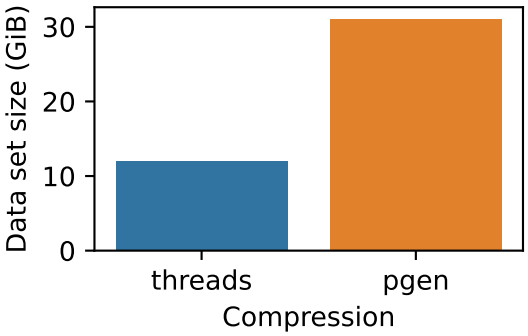


Figure 5.7: File sizes for genome-wide genotyping array data for 337,464 unrelated white British samples from the UK Biobank compressed using the Plink2 and Threads compression methods.

the UK Biobank [10], phased using Beagle 5.1 [8]. We partitioned the genome into 200 chunks of approximately equal size and inferred a genome-wide genealogy for the 337,464 samples. This data set was previously used for ARG-based analyses in [99]. File sizes for `.threads` and `.pgen` compression are compared in Figure 5.7, with Threads reducing disk usage by around 60%, while encoding both the full genotype matrix and an inferred ARG.

5.3.3 UK Biobank imputed variants

Finally, we tested Threads using SNPs from the UK Biobank imputed using the Haplotype Reference Consortium (HRC) panel [58]. We included only high quality SNPs of INFO score greater than 0.95, minor allele frequency greater than 0.0001 and missingness ratio lower than 0.1. We rounded dosages within 0.1 of the nearest integer and set other dosages to missing. This filtering procedure left in 9,992,478 high-quality and highly informative variants for for 487,409 samples. These variants were phased using SHAPEIT5 [27] and passed to Threads for ARG inference in 137 chunks each of around 30cM in length.

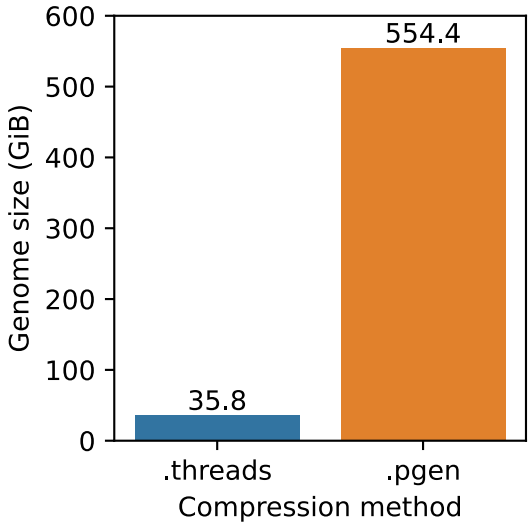


Figure 5.8: File sizes for 9,992,478 high-quality, hard-called imputed variants for 487,409 samples from the UK Biobank compressed using the PLINK2 and Threads compression methods.

We found that at this scale, Threads achieved excellent compression rates, using only 6% of the disk space used by `.pgen` files to store the same genotypes. The high compression rate is achieved primarily due to the density of variants efficiently stored on each Viterbi segment, as well as the presence of relatedness in the UK Biobank.

Together with our observations from simulations and from the 1000 Genomes data set, these results show Threads provides a robust, scalable, and widely-applicable approach for storage of genetic data. Distribution of large data sets as Threads-inferred threading instructions could lead to significant savings in disk space while also making a range of applications in statistical and population genetics more accessible, since an ARG inference step is a computational bottleneck for many possible downstream analyses.

Chapter 6

Genotype phasing and imputation

Genotype imputation is the process of inferring unobserved or missing genotypes, usually by exploiting patterns of linkage disequilibrium and an external panel of densely genotyped sequences. In the most common setting, the study sample has been sparsely genotyped using SNP microarrays and a whole-genome sequenced reference panel is used to impute variants that are observed in the panel but not included on the array. Genotype phasing is the process of decomposing an observed diplotype into its two parental haplotypes. Both phasing and imputation are important first steps for many downstream analyses.

Here, we describe ARG-based approaches to phasing and imputation. This chapter is structured as follows. In Chapter 6.1 we give a brief discussion of how the Li-Stephens model is used for genotype imputation and why haplotype-copying methods like the Li-Stephens are not optimal for imputation of ultra-rare variants. In Chapter 6.2 we show how ARGs may be used to improve imputation of variants with a low derived allele count in the reference panel. We confirm this with both simulations and experiments using data from the 1000 Genomes Project [3, 11] and we show that our ARG-based imputation method can lead to increased associative power in simulated burden tests. ARG-based imputation has been explored

conceptually [81, 99], but empirical studies have been limited by the scalability of ARG inference. In Chapter 6.5 we present an ARG-based approach to rare-variant phasing and show that it obtains similar accuracy to traditional algorithms for a trio-phased sample from the 1000 Genomes Project, while underperforming in simulations.

6.1 Background

The gold-standard statistical model for genotype imputation is the Li-Stephens hidden Markov model [48], solved using the forward-backward algorithm. This is the approach employed by widely used imputation algorithms such as IMPUTE5 [76] and Beagle 5.4 [8], each providing a different set of computational heuristics to very similar underlying models. The Li-Stephens model is introduced and discussed in more detail in Chapter 2.3.

In a typical imputation study, the reference panel is subsampled down to the microarray sites and the Li-Stephens forward-backward algorithm is run for each sample using the subsampled reference panel. For each site on the genotyping array, this gives a posterior over the reference sequences for a haplotype copying path. To extend this posterior to the sites to be imputed, the posterior is linearly interpolated between the array sites, giving a full posterior for each reference site. To impute site j for sample i, we can then use the expectation $E[g_{i,j}] = \mathbf{p}_{i,j} \cdot \mathbf{g}_j^{\text{ref}}$, where $g_{i,j}$ is the genotype we wish to impute, $\mathbf{p}_{i,j}$ is the interpolated posterior for target sample i at site j, and $\mathbf{g}_j^{\text{ref}}$ contains the reference genotypes at site j. The genotype obtained using this approach is not guaranteed to be an integer and is usually called a *genotype dosage*.

6.1.1 The limits of haplotype copying

Imputation using the Li-Stephens model has been widely observed to achieve near-perfect accuracy for common variants, but struggles at the rarest end of the frequency spectrum. By directly copying the reference sequences, Li-Stephens-based imputation methods implicitly assume that the age of any mutation carried by the reference sequences is greater than the coalescence time of the sequence being imputed into the reference panel. This assumption leads to a systematic inflation of genotype dosages for rare variants [81]. When the underlying ARG of the reference panel and threading instructions for the target sequence are available, we can identify mutations with lineages in the coalescent tree and verify whether the target sequence is descended from the lineage on which the mutation arose, as demonstrated by a toy example in Figure 6.1a. Dosage inflation is primarily a function of allele count in the reference panel, with only a weak dependence on panel size [81]. Consequently, imputed variants of a fixed, low allele count in the reference will have similar levels of inflation across a range of panel sizes.

Although it is possible to derive an analytical expression for the probability, under the standard coalescent, that an alternative allele is imputed with high confidence when the sequence being imputed carries the reference allele [81], no such expression is available for arbitrary demography or for the imputed dosage inflation. However, it can be estimated empirically. In Figures 6.1b and 6.1c we attempt to estimate the imputation inflation due to haplotype copying in simulations under a realistic European demography for trees on 100 and 1,000 leaves, respectively. We find that when imputing a variant that is a singleton in the reference and carried by a closest cousin of the sequence being imputed, the average imputed dosage is 2.55 and 2.83 times higher than the truth for trees on 100 and 1,000 leaves, respectively. Imputing singletons using the ARG results in an inflation factor of 1.38 for both tree sizes. As the allele count grows, coalescence of the target sequence is increasingly

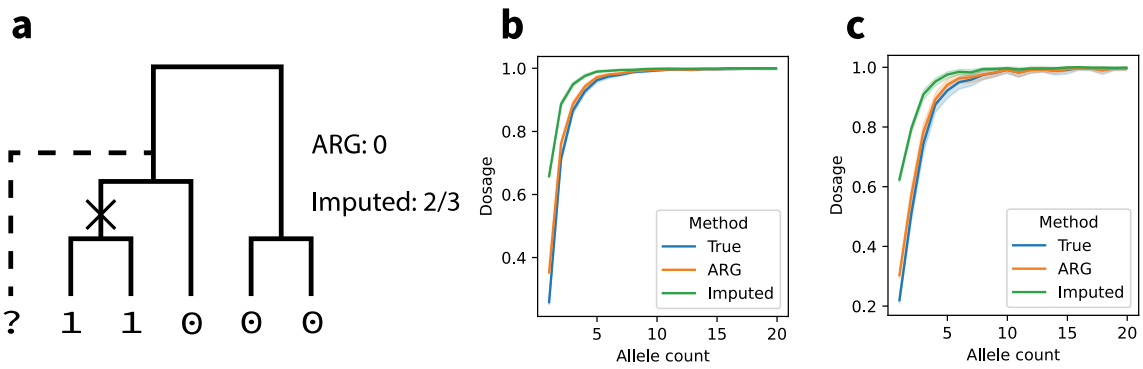


Figure 6.1: **a**. A toy example of how haplotype copying can inflate genotype dosages. The coalescent tree of the reference panel and threading instructions for the target sequence are shown, with the mutation being imputed marked by "$\times$". Reading directly from the ARG gives the true dosage of 0, while haplotype copying ("imputed"), which averages over the genotypes of all closest cousins, gives a dosage of $2/3$. **b**, **c**. Average imputed dosages by allele count for panel sizes of 100 and 1,000 haplotypes, respectively. For each tree size, we simulate 500,000 random coalescence trees and sample a mutation randomly from any lineage ancestral to the target sequence or any of its closest cousins. Using the true, simulated ARG, we then derive the true genotype of the target sequence as well as the haplotype copying dosage and the ARG dosage. The ARG dosage is equal to the true dosage except when the target coalesces directly to the lineage carrying the mutation, in which case we assume the mutation is uniformly distributed along that lineage and define the ARG dosage as the proportion of the lineage below the coalescence event.

likely to lie below the mutation being imputed and the scenario in Figure 6.1a is less likely to occur, with the inflation levelling out for variants of minor allele count around 10 in the panel.

6.2 Imputation via threading

As discussed in Chapter 6.1 and in [99] and [81], we can significantly improve imputation accuracy for ultra-rare variants if we have access to an ARG of the reference panel and threading instructions for the target sequences. In this chapter we describe how these ideas can be applied to inferred ARGs and threading instructions.

For now, assume we have inferred an ARG for the whole-genome sequenced reference panel and have mapped each variant to a set of edges on marginal coalescence trees such that the leaves subtended by those edges exactly match the carriers of the variant. Most ARG inference algorithms do not provide guarantees that input mutations exactly match edges in the inferred ARG and we discuss how mutations can be mapped to the ARG post-hoc in Chapter 6.2.1. To obtain threading instructions for the target sequences, we subsample the reference panel to match the target sequences and run Threads.

While these threading instructions are sufficient to perform ARG-based imputation, we can further improve accuracy by accounting for uncertainty in coalescence height, threading target, and segment boundaries. First, to account for uncertainty in coalescence height, we note that since we are not iteratively building an ARG but only performing imputation, there is no need to keep a fixed point estimate for coalescence height. If the threading target carries a non-ancestral allele arising at some time t_m, we instead compute the probability $p(t \leq t_m)$ that the coalescence time t is below the mutation. We can model the coalescence time using an Erlang-2 distribution (see Chapter 3.1.3) with parameter λ, a function of segment length and effective population size. Setting t_m to be the midpoint of the edge on which the mutation lies, we get

$$p(t \leq t_m) = 1 - e^{-\lambda t_m}(1 + \lambda t_m). \tag{6.1}$$

In our applications, we assume a constant demographic size of $N_e = 20{,}000$.

Next, to account for uncertainty in the threading target, we use divergence values in the PBWT to find sequences in the reference panel that are identical to the threading target on the inferred Viterbi-segment. These can be found without excessive computation using the divergence values stored in the d-PBWT data structure in Threads-fastLS or using a separate PBWT pass through the data in Threads. For each such sample, we gather the ARG dosages from Equation 6.1 into a matrix

A, with $A_{i,j}$ equal to the ARG dosage for reference sample i at site j if i is identical to the threading target on the segment spanning i, but 1 otherwise.

Third, to account for uncertainty in segment boundaries, and to further increase resolution in threading target uncertainty, we also run a forward-backward algorithm to get a full posterior matrix P. This step is also necessary to obtain good accuracy for common variants, as for all but the most recent mutations the forward-backward algorithm is a more powerful tool for imputation than the Viterbi algorithm. In practice, we fully rely on the forward-backward posterior for variants of allele frequency greater than 0.1% and of minor allele count greater than 20.

Finally, to obtain the full genotype dosage for each site, we take each of our three $N_{\text{ref}} \times M_{\text{ref}}$ matrices, the ARG dosages A, the forward-backward posterior P, and the reference panel genotypes G, and compute the M_{ref}-dimensional genotype vector

$$\mathbf{g} := \mathbf{1} \cdot (A \odot P \odot G),$$

where $\odot$ is the element-wise product and $\mathbf{1}$ is the $1 \times N_{\text{ref}}$-dimensional vector with each element equal to 1.

6.2.1 Placing mutations on ARGs

In the imputation-by-threading algorithm described above, it is assumed that all mutations carried by the reference panel can be identified with edges in marginal coalescence trees of the inferred ARG. However, most ARG inference methods, including Threads, do not guarantee the existence of edges exactly subtending the carriers of each mutation. In Chapter 3.5.6 we discussed the difficulties of developing a mutation-consistent threading algorithm. Here we focus instead on post-hoc placement of mutations on inferred ARGs. Throughout, we assume a binary mutation model and that the ancestral allele is known.

An effective way of mapping mutations to ARGs is by assigning one mutation each to the smallest number of edges that exactly subtend the carriers of the mutation. This type of heuristic strategy is often referred to as *parsimony*. The naive implementation in Algorithm 8 runs in $O(N)$ time in the worst case but $O(N_{\text{carriers}})$ on average. While both simple and effective, the parsimony approach has two downsides. First, errors in the inferred ARG can mean a variant that perfectly maps to a real lineage needs to be broken up into many recurrent mutations, each subtending a lower allele count than the true variant, and thus biased downwards in time. Second, this approach relies on perfect knowledge of ancestral allele states.

Algorithm 8 Parsimonious mapping of mutations to coalescence trees

Input: A coalescent tree T on N leaves and a set X of carriers.
Output: A smallest set of branches $\mathcal{B}$ subtending exactly X.

$\mathcal{B} \Leftarrow \{\}$
while X is non-empty **do**
    Choose an element x from X
    Let B be the branch satisfying leaves$(B) = \{x\}$
    while leaves(parent$(B)) \subseteq X$ **do**
        $B \Leftarrow$ parent(B)
    end while
    Add B to $\mathcal{B}$
    $X \Leftarrow X \setminus$ leaves(B)
end while
return $\mathcal{B}$

Another method of mapping mutations to coalescent trees that heuristically accounts for both errors in the inferred ARG and an unknown ancestral state was introduced by Speidel et al. with Relate [84]. Briefly, for an observed variant with C_o carriers and N_o non-carriers, and a mutation (equivalently, branch) subtending C_m carriers and N_m non-carriers, we say the mutation *maps*, in the Relate sense, to the variant if it satisfies

$$\frac{|C_o \cap C_m|}{\max\{|C_o|, |C_m|\}} > 0.7 \quad \text{and} \quad \frac{|N_o \cap N_m|}{\max\{|N_o|, |N_m|\}} > 0.7.$$

For each branch mapping in this way to the variant, we also obtain the fraction of misclassified haplotypes as

$$\frac{|N_m \cap C_o| + |N_o \cap C_m|}{N}.$$

This operation is repeated after flipping the observed variant and interpreting carriers as non-carriers, thus checking both possible ancestral states. Finally, the branch with the lowest fraction of misclassified variants is chosen if that fraction is lower than 0.03.

Both approaches are computationally expensive, especially for common variants. In our applications we adopt the Relate mapping approach, but as we only use the ARG for imputation of rare variants, we adopt an extra set of heuristics. Firstly, we only attempt to map mutations of allele frequency $\min\{0.02, 20/N_{\text{hap}}\}$. Second, we assume the major allele is ancestral, which is usually a safe assumption for the very rare variants we target here. Finally, we only apply the Relate mapping for variants of minor allele count ≥ 5, relying on parsimony for variants of minor allele count < 5. For variants that do not map to the ARG, we rely on the Li-Stephens forward-backward posterior for imputation.

6.3 Simulation studies

To evaluate the accuracy of our ARG-based imputation algorithm, we first simulate whole genome sequences and genotyping arrays using the European demography and simulation parameters from Chapter 4.2. We use reference panel sizes ranging from 1,000 to 30,000 diploids and 10 target diploids, in each case simulating a 20MB region and discarding the outermost 5MB on both ends as burn-in. We repeated each experiment for 10 random seeds. Accuracy was evaluated using the aggregate r^2 between imputed genotypes and the true, simulated genotypes [9, 76, 99].

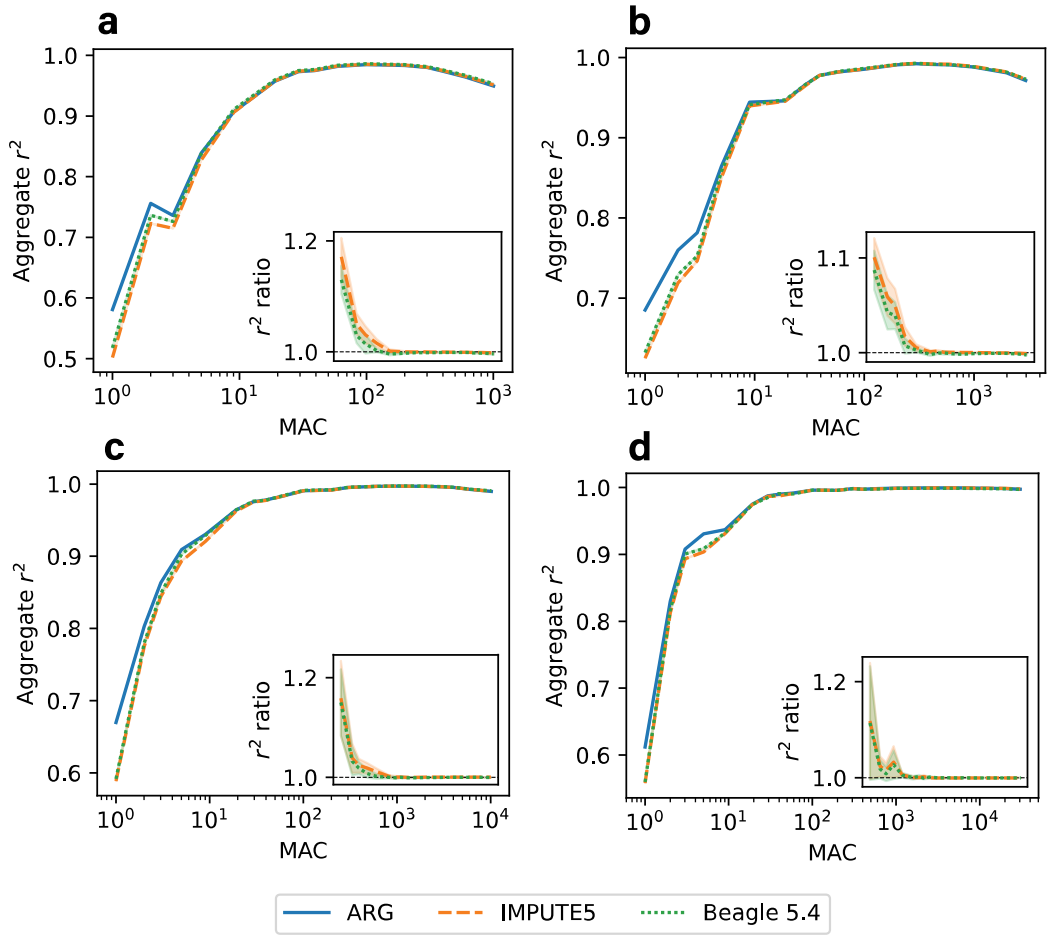


Figure 6.2: Imputation accuracy using reference panels of size 1,000, 3,000, 10,000 and 30,000 in panels **a**-**d**, respectively. Values shown are averaged across 10 simulation seeds. Inset plots show the ratio between the aggregate r^2 obtained by our ARG-based imputation method and the other methods, with shaded regions showing 95% confidence intervals across all random seeds.

Our results, in Figure 6.2, show a marked improvement over IMPUTE5 and Beagle 5.4 in imputation accuracy of ultra-rare variants for all panel sizes. To verify that these gains are statistically significant, we compared the ratio of aggregate r^2 between ARG-based imputation and IMPUTE5 and Beagle 5.4 for each seed, finding an improvement over IMPUTE5 and Beagle 5.4 of 10-15% in r^2 for singletons, with the accuracy gains levelling out for variants of allele frequency around 10.

6.3.1 Association studies

A principal goal of genotype imputation is to increase the power of association studies. To explore whether our improved imputation of ultra-rare variants could translate into improved association power, we set up a simulated burden test scenario.

Using the same simulation parameters as above, but with a panel size of 3,000 and a target sample size of 2,000, we chose 10 evenly distributed 5-kilobase regions as simulated genes. For each region in turn, we designated each variant of allele frequency $< 1\%$ as causal, with dominant, non-additive effects. We simulated a quantitative phenotype by assigning an effect of $\beta = 1.0$ to carriers and drawing environmental effects from a standard normal distribution. We then computed the imputed burden by summing over the dosages of causal variants within the region, imposing a maximum value of 1.

We computed χ^2-statistics for association between the simulated phenotypes and the burden obtained using each of the three imputation methods, also comparing with the test statistic obtained using the true, simulated burden. We report the effective sample size relative to the true burden as $\chi^2_{\text{imputed}}/\chi^2_{\text{true}}$ for each of the three methods in Figure 6.3. This analysis was repeated for 5 different random seeds. A pairwise test of χ^2-statistics for our ARG-based imputation against IMPUTE5 and Beagle 5.4 confirms that the increase in power is significant (paired, two-sided t-test, $p = 8.6 \times 10^{-9}$ for IMPUTE5, $p = 3.6 \times 10^{-6}$ for Beagle 5.4).

6.4 Imputation using the 1000 Genomes

To evaluate imputation performance on real data, we inferred genome-wide genealogies for the 1000 Genomes Project data set [3, 11] using Threads. We used a subset of 2,261 unrelated samples, first designating 10 samples of European ances-

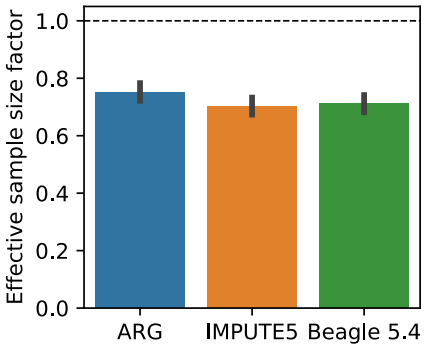


Figure 6.3: Estimated effective sample size multiplier when using burden tests on data imputed using three methods compared with using the true genotypes.

try (2 from each of 5 subpopulations) as target samples and the remaining 2,251 as the reference panel, and then repeating the experiment for 7 samples of African ancestry (1 from each of 7 subpopulations). We then subsampled all samples down to sites matching the UK Biobank Axiom array sites and performed genotype imputation using IMPUTE5, Beagle 5.4 and our ARG-based method.

The results show slightly lower gains in accuracy than in simulations for ultra-rare variants, with an increase in r^2 of 5-6% for singletons in both populations, with accuracy levelling off around an allele count of 10 (Figure 6.4). We also observed a small performance degradation in common variants, that while marginal, is significant, and is likely either due to different parameter choices for the Li-Stephens forward-backward algorithm that underlies all three algorithms or different choices of computational heuristics.

6.5 ARG-based genotype phasing

Genotype phasing is the process of decomposing a diplotype $g = \{0, 1, 2\}^M$ into parental haplotypes $g_1, g_2 \in \{0, 1\}^M$, such that $g_1 + g_2 = g$. Phasing is a pre-requisite step for ARG-building with Threads and is an essential step in other applications,

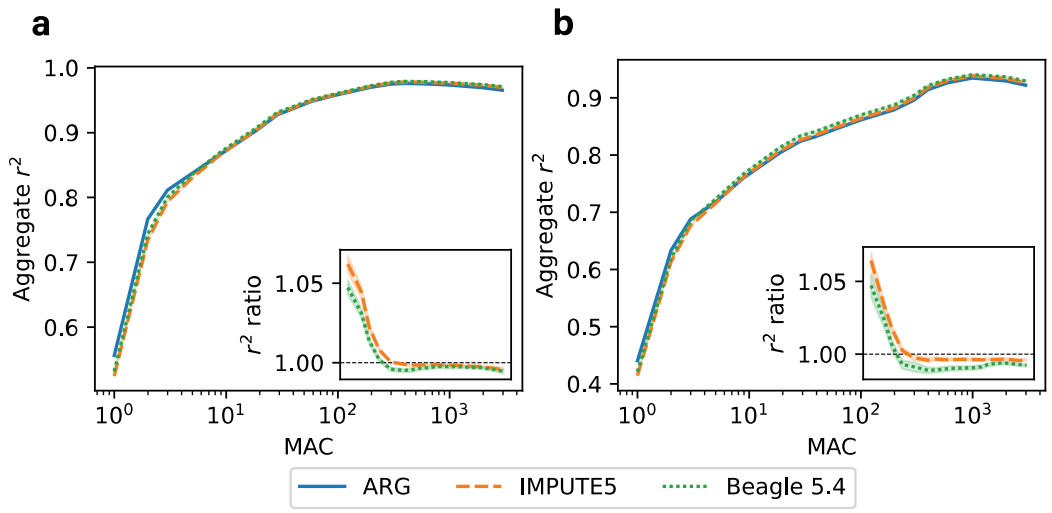


Figure 6.4: **a**. Imputation accuracy of 10 European samples from the 1,000 Genomes Project, 2 from each of 5 subpopulations, using 2,251 unrelated samples as the reference panel. Inset shows the ratio between the aggregate r^2 obtained by our ARG imputation method and the other methods, averaged across the whole genome. Shaded regions show 95% confidence intervals. **b**. The same experiment repeated for 7 African samples, 1 from each of 7 subpopulations.

such as the identification of compound heterozygotes [27]. Modern phasing algorithms proceed in two distinct steps, first obtaining a high-confidence scaffold of phased common variants that is then used to inform the phasing of rarer variants [8, 27]. Here, we present an ARG-based approach to phasing, leveraging the phased common variant scaffold to infer an ARG that is used to give a simple phasing algorithm for rare variants.

Both Beagle 5.4 and SHAPEIT5, two widely used phasing algorithms, rely on a Li-Stephens model to perform rare variant phasing [8, 27], with a rare-variant threshold of 0.1-0.2%. In the case of SHAPEIT5, the rare-variant procedure is very similar to the inference of threading instructions using Threads, a haplotype selection step using the PBWT followed by a Li-Stephens algorithm, and so it is instructive to compare how the two methods perform in phasing.

The ARG-based rare-variant algorithm starts from each of the two possible haplotypes carrying the unphased variant, traversing the marginal coalescence tree up-

wards until a common ancestor of at least one haplotype of all heterozygous carriers has been found and both haplotypes of all homozygotes. The average tree-based distance to all such carriers gives the *phasing penalty* $d_{\text{phase}}(\cdot)$ for the haplotype. We then place the variant on the haplotype of lower phasing penalty and repeat the analysis for all heterozygous carriers of the mutation. More formally, if for each unphased carrier c, c_1 and c_2 are its two haplotypes, then the phasing penalty can be written as

$$d_{\text{phase}}(c'_a) = \sum_{c \text{ heterozygous}} \min(d(c'_a, c_1), d(c'_a, c_2)) + \sum_{c \text{ homozygous}} d(c'_a, c_1) + d(c'_a, c_2)$$

for $a = 1, 2$, where $d(\cdot, \cdot)$ is the tree-based distance as measured in generations. If c' is the only carrier of the mutation, we set $d_{\text{phase}}(c'_a)$ to the negative of the age of its most recent common ancestor. The singleton case prioritizes the haplotype that is least related to the rest of the sample and more likely to carry a singleton mutation. A similar step based on identical-by-state tracts is performed by SHAPEIT5 [27]. For all ARG-based analyses, we rely on common-variant scaffolds inferred using SHAPEIT5 with a threshold of 0.1%.

In Figure 6.5 we show results for simulations for $N = 3{,}000$ and $N = 10{,}000$ samples simulated under a realistic European demography using simulation parameters described in Chapter 4.2. We assume nothing is known about the phase a priori and that there is no access to a phased reference panel. Accuracy was estimated using the switch error rate [27] as implemented in SHAPEIT5. Threads and SHAPEIT5 perform similarly in the phasing of singletons and both outperform Beagle 5.4, which is the only method that does not specifically model singletons. However, for non-singleton rare variants, Threads obtains lower accuracy than SHAPEIT5 and Beagle 5.4. Singletons account for around half of the variants in our simulations, and poor performance in singleton phasing also affects accuracy

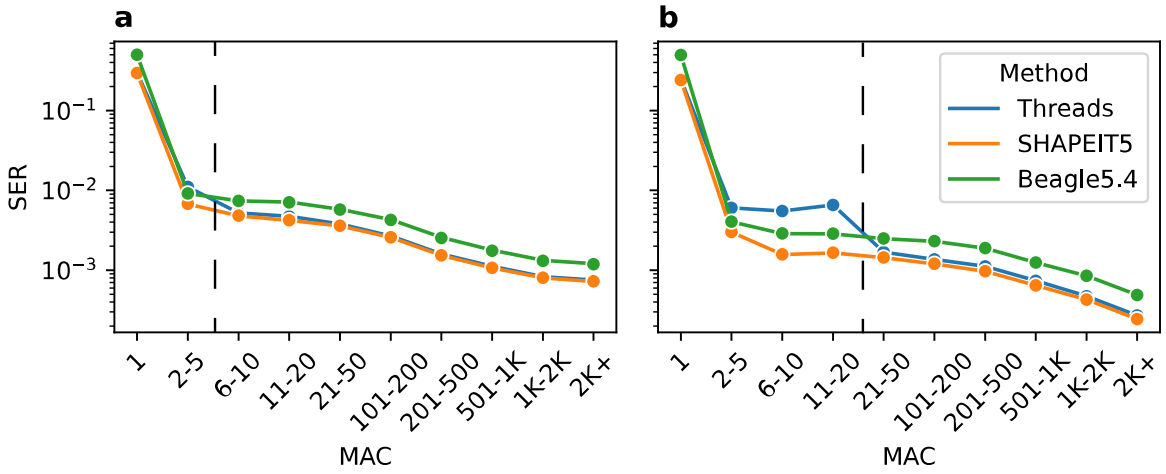


Figure 6.5: Switch error rate (SER) binned by minor allele count (MAC) for Threads, SHAPEIT5 and Beagle 5.4 in simulations of size $N = 3{,}000$ and $N = 10{,}000$ in panels **a** and **b**, respectively. Dashed vertical lines show the cut-off between common and rare variant phasing routines.

estimates for adjacent mutations, partly explaining the lower accuracy of Beagle 5.4 for common variants.

We also tested the performance of SHAPEIT5 and Threads in the 1000 Genomes Project [3, 11] using a single, trio-phased sample, NA12878. We removed the phase from 2,261 unrelated samples ran both phasing algorithms, evaluating accuracy only on NA12878. Due to the small sample size, only variants of minor allele count up to 4 remain after the common-variant scaffold. The results show a similar performance across all frequencies for both Threads and SHAPEIT5 (Figure 6.6).

6.6 Discussion

In this chapter, we presented an ARG-based method that improves over state-of-the-art imputation approaches that are based on the Li-Stephens forward-backward algorithm. We've shown robust improvement in imputation accuracy for ultra-rare variants (MAC ≤ 10) in both simulated and real data and discussed why we should not expect gains for more common variants, irrespective of reference panel size.

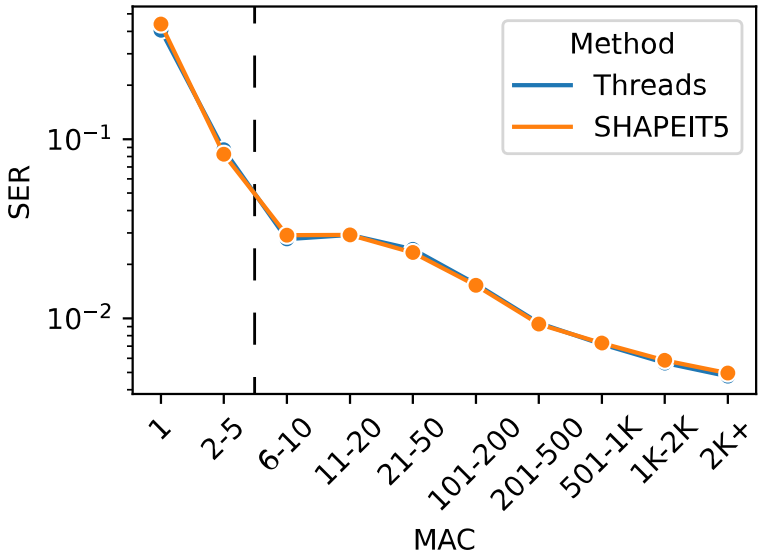


Figure 6.6: Genome-wide switch error rate (SER) binned by minor allele count (MAC) for Threads and SHAPEIT5 for a single trio-phased sample from the 1000 Genomes Project (NA12878). Dashed vertical line shows the cut-off between common and rare variant phasing routines.

Our simulations suggest that this threading-based approach to genotype imputation may increase the power to detect disease-causing genes through burden tests.

This improvement was made possible by two innovations. First, our highly scalable ARG inference algorithm, Threads, allows us to construct accurate genealogies of large-scale reference panels, giving estimates of mutation ages. Second, the probabilistic threading model described in Chapter 6.2 allows us to leverage uncertainty in threading instructions to get accurate genotype dosage estimates. Together with our results on genotype compression in Chapter 5, these results suggest our ARG-based models could be effectively applied to imputation servers, where they could yield improved genotype imputation results, require less disk space, and provide an estimate of the genealogical history of the imputed samples.

Because our ARG-based imputation approach runs the forward-backward algorithm as a sub-routine, it will struggle to outperform IMPUTE5 and Beagle 5.4 computationally. However, the costly extra steps of ARG inference and mutation

mapping need only be performed once for the reference panel. Inference of threading instructions and ARG querying can be performed very efficiently, once the reference panel has been pre-processed. The exact choice of which computational heuristics and optimizations would best suit the ARG-based imputation remains an open question.

We have also presented a new ARG-based approach to rare-variant phasing, which obtained promising results for a trio-phased sample from the 1000 Genomes Project, but underperformed in simulations. We hypothesize that the poor performance observed in simulations may be due to two factors. First, the ARG-based phasing method does not account for uncertainty to the same extent as other phasing methods. By comparison, our ARG-based imputation algorithm accounts for uncertainty across all three axes of ARG inference; the position of breakpoints along the genome, the index of the threading target, and the coalescence time. Second, the ARGs inferred from common-variant scaffolds are likely of lower quality than the ones used in our imputation method, where whole genomes of the reference panel are available for the inference. More work is required to further explore if and how ARGs can be used to provide consistent improvement in rare-variant phasing.

Chapter 7

Genealogy-wide association studies

In Chapter 6 we showed how ARGs may be used to improve imputation accuracy for ultra-rare variants. But no matter how large our reference panels or how accurate our algorithms, large, imputed cohorts will always carry a significant amount of genetic variation that is absent from the reference panel and thus impossible to analyze while relying on genotype imputation. This phenomenon of "unobserved variation" motivated the development of a strategy referred to as genealogy-wide association study (GeWAS) in Zhang et al. [99], whereby ARG clades that may or may not carry unobserved mutations are tested for association with observed phenotypes. This approach was observed in [99] to increase association power at rare and ultra-rare variants for unrelated white British samples in the UK Biobank. In this chapter we show how ARGs inferred using Threads can improve upon ARG-Needle in both speed and power, focusing on samples of non-European origin.

7.1 Background

The linear mixed model (LMM) is the most widely used approach for genome-wide association studies (GWAS), giving strong associative power and robustness to population structure [31, 32, 51, 54, 57, 100]. The GeWAS test statistic used in

[99] is also derived from the LMM and in what follows we give details of the LMM model and how it is used in GeWAS testing. As in [99], we focus on the case of quantitative traits and consider a vector of phenotypic values $\mathbf{y}$ of dimension $N \times 1$. Under the LMM, we model the trait using the model

$$\mathbf{y} = \mathbf{W}\beta + \mathbf{g} + \mathbf{e},$$

where $\mathbf{W}$ are covariates with fixed effects β, $\mathbf{g} \sim N(0, \sigma_g^2 \mathbf{K})$ is a random effect representing total genetic effects under the $N \times N$ kinship matrix $\mathbf{K}$, and $\mathbf{e} \sim N(0, \sigma_e^2 \mathbf{I}_N)$ is residual environmental variance [31, 36, 50, 51, 98, 100, 101]. The kinship matrix $\mathbf{K}$ is usually unknown, so instead we approximate it using the $N \times N$ genetic relatedness matrix (GRM)

$$\mathbf{A} \coloneqq \frac{1}{M}\mathbf{G}\mathbf{G}^{\mathrm{T}},$$

where $\mathbf{G}$ is the $N \times M$ genotype matrix, allowing us to write

$$\mathbf{V} \coloneqq \mathrm{Var}(\mathbf{y}) = \sigma_g^2 \mathbf{A} + \sigma_e^2 \mathbf{I}_N.$$

To test for association of a variant $\mathbf{x}_{\text{snp}}$ of dimension $N \times 1$ using the LMM, we model

$$\mathbf{y} = \mathbf{x}_{\text{snp}}\beta_{\text{snp}} + \mathbf{W}\beta + \mathbf{g} + \mathbf{e}$$

and test the null hypothesis $\beta_{\text{snp}} = 0$ using the test statistic [2]

$$\chi^2_{\text{LMM}} = \frac{\left(\mathbf{x}_{\text{snp}}^{\mathrm{T}} \mathbf{V}^{-1} \mathbf{y}\right)^2}{\mathbf{x}_{\text{snp}}^{\mathrm{T}} \mathbf{V}^{-1} \mathbf{x}_{\text{snp}}}.$$

Applicability of this test statistic in large data sets follows from two observations. First, the factor $\mathbf{V}^{-1}\mathbf{y}$ in the numerator does not depend on which variant is being tested, and can be efficiently approximated with some $\tilde{\mathbf{y}} \approx \mathbf{V}^{-1}\mathbf{y}$ a priori using

a subset of the observed variants [35]. Second, under the GRAMMAR-Gamma assumption, $\mathbf{x}_{\text{snp}}^{\text{T}}\mathbf{V}^{-1}\mathbf{x}_{\text{snp}} \approx c \cdot \mathbf{x}_{\text{snp}}^{\text{T}}\mathbf{x}_{\text{snp}}$, for some constant c [86]. This leads to the test statistic

$$\chi^2_{\text{LMM}} \approx \frac{\left(\mathbf{x}_{\text{snp}}^{\text{T}}\tilde{\mathbf{y}}\right)^2}{c \cdot \mathbf{x}_{\text{snp}}^{\text{T}}\mathbf{x}_{\text{snp}}}, \tag{7.1}$$

which is used to test for association in the GeWAS [99]. Although suppressed in the notation here, the test statistic is computed using a leave-one-chromosome-out (LOCO) scheme, in which variants on one chromosome are tested using a model that has been fitted using variants from all other chromosomes, preventing proximal contamination which may reduce association power [50, 98].

To adjust for covariates, we save computation by applying an approximate covariate adjustment approach. Under this approximate adjustment, only the phenotype $\mathbf{y}$ is adjusted, but the genotype $\mathbf{x}_{\text{snp}}$ is used without any adjustment [31, 99]. For the vast majority of SNPs, there will be no meaningful interaction with covariates and this approximation is justified. In fastGWA-GLMM [32], the full, adjusted test statistic is then computed for variants with a χ^2-statistic greater than 4, but in the GeWAS, no such second pass is performed.

Under the infinite sites model for point mutations, each variant that arises in the study population can be uniquely mapped to an edge in the underlying ARG by identifying its carriers with leaves subtended by that edge. This assumption does not account for large structural variants and recurrent mutations that may not map perfectly or uniquely to the underlying ARG. We can test for association with observed and unobserved variation by traversing the ARG and computing the test statistic in Equation 7.1 for each edge. However, this is a costly operation that can result in an excessive number of tests. Instead, Zhang et al. [99] suggest traversing the ARG and testing edges with probability $1 - e^{-\mu \cdot v}$, where μ is some pre-determined sampling rate and v is the edge volume, defined as the edge height in generations multiplied by the number of base-pairs it persists along the genome.

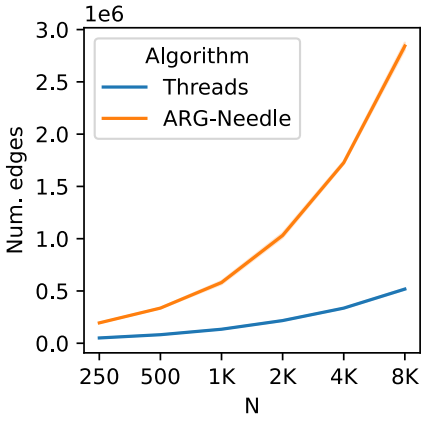


Figure 7.1: The number of edges in ARGs inferred using Threads and ARG-Needle on simulated genotyping arrays under a typical European demography.

In our experiments, we use $\mu = 1 \times 10^{-5}$. This procedure results in a significantly higher number of tests than a traditional GWAS, and significance thresholds are derived using permutation tests [34, 99].

7.1.1 Runtime improvements with Threads

The GeWAS analysis requires an edge-by-edge traversal of the input ARG to test for association with a trait genealogy-wide. To limit the number of tests and to reduce overall runtime, it is important that the input ARG is efficiently represented. Because of the segment-based approach used to infer threading targets and coalescence times in Threads, inferred ARGs have significantly fewer edges than ARG-Needle ARGs, making ARG traversal more efficient. Figure 7.1 shows the number of edges in ARGs inferred from simulated genotyping arrays using the simulation parameters described in Chapter 4.2, with ARG-Needle using more than five times as many edges as Threads to represent ARGs inferred from the same data set in the largest simulations.

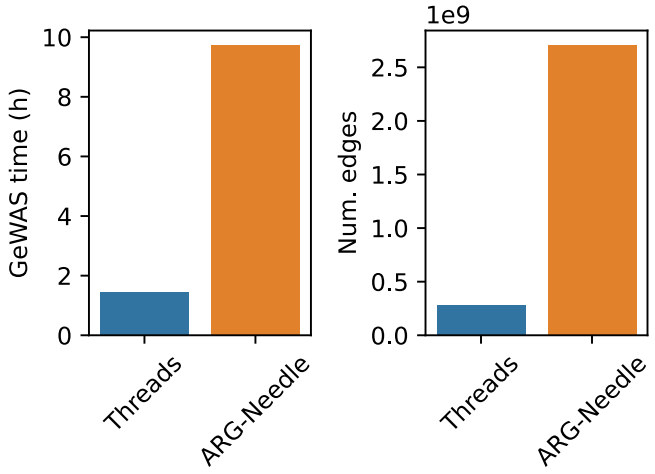


Figure 7.2: Wall time required to run a GeWAS for ARGs built using genotyping arrays from 6,747 samples of African origin from the UK Biobank using both Threads and ARG-Needle and the number of edges in those ARGs.

This parsimonious representation of the ARG translates into more efficient ARG traversal and lower memory usage. In Figure 7.2 we compare the number of edges used to represent genome-wide ARGs for genotyping arrays from 6,747 samples of African ancestry in the UK Biobank and the time required to perform a single genealogy-wide association study using both Threads and ARG-Needle ARGs. In both instances we obtain an improvement of close to an order of magnitude. Accuracy benchmarks in Chapter 4.3 suggest that this sparser ARG structure does not result in a significant loss of accuracy.

7.2 GeWAS of African and Central/South Asian samples from the UK Biobank

In this chapter, we evaluate the power of genealogy-wide association studies using Threads-inferred ARGs in populations that are under-represented in reference panels. Using ancestry definitions from the Pan-UK Biobank Project [88], we focus on

the two largest non-European ancestry groups from the UK Biobank, 6,747 samples of African (AFR) ancestry and 9,009 samples of Central/South Asian (CSA) ancestry, imputed using the Haplotype Reference Consortium (HRC) reference panel, which consists of 65,000 haplotypes of predominantly European ancestry [10, 58]. We hypothesized that a genealogy-wide association study might identify clades associated with quantitative traits that would otherwise not be discovered using a GWAS, due to lack of representation in the reference panel.

First, we inferred genome-wide genealogies for both ancestry groups, using high-quality markers from the HRC imputation data set. We filtered out variants of with INFO score < 0.95, with missingness greater than 10%, setting to missing all genotype dosages further than 0.1 from the nearest integer, otherwise rounding to hard integer calls. We omitted singletons but kept all other variants passing the filters above. This left 8,911,042 variants for AFR and 8,488,525 for CSA. By comparison, the Axiom genotyping arrays used to infer genealogies by Zhang et al. used only 727,130 markers genome-wide [10, 99]. We did not perform any chunking of the genome, inferring ARGs for whole chromosome arms at a time. We ran genealogy-wide association studies on both AFR and CSA for height and six blood-related traits (mean platelet volume, levels of alkaline phosphatase, aspartate aminotransferase, high-density lipoprotein cholesterol, low-density lipoprotein cholesterol, total bilirubin), all of which have high phenotyping rates. We then compared the results with GeWAS analyses using ARGs based only the array SNPs and inferred using Threads, and with GWAS performed using BOLT-LMM [51] directly on the imputed data set. All traits were adjusted for age, sex, age squared, interactions between sex and age, and 20 within-population principal components. The imputed markers used for GWAS were filtered on INFO score > 0.3, minor allele count ≥ 5, and a missingness rate of $< 5\%$. All GeWAS experiments used a sampling rate of $\mu = 10^{-5}$. To maintain a consistent false positive rate across all

Testing modality	AFR threshold	CSA threshold
Axiom array ARGs	$9.38\ (7.50; 11.19)\times10^{-10}$	$6.04\ (4.44; 8.26)\times10^{-10}$
Imputation	$2.39\ (1.44; 3.90)\times10^{-9}$	$1.69\ (7.29; 3.16)\times10^{-9}$
Threads HRC ARGs	$5.48\ (4.29; 7.21)\times10^{-10}$	$5.40\ (3.46; 6.84)\times10^{-10}$

Table 7.1: Significance thresholds for genome- and genealogy-wide association studies with 95% confidence intervals obtained using permutation testing as described in [99].

experimental modalities, we performed permutation testing as described in [99] to evaluate genome- and genealogy-wide significance thresholds, which are shown in Table 7.1.

To ascertain the number of independent association signals we applied a stringent two-step filtering routine. First, we applied genotype clumping as implemented in PLINK (version v1.90b6.26) [14] to the summary statistics produced by each association testing method, keeping only lead SNPs passing the thresholds in Table 7.1, and removing SNPs with r^2 greater than 0.01 with the lead SNPs within a 2MB window. Next, we extracted conditionally and jointly independent signals from the clump leads using the GCTA command `--cojo-slct`(version v1.94.1) [96, 97], setting `--cojo-p 1e-7` but otherwise using default parameters. Because the GeWAS does not produce the effect size estimates used by GCTA, we used approximate effect size estimates obtained running linear regression using the PLINK2 `--glm` command (version v2.00a3.7LM) [14], adjusting for the same covariates as in the GeWAS and GWAS studies. These filtering steps resulted in a total of 38 independent association signals for the GeWAS on array ARGs, 48 for the GeWAS on the HRC-inferred ARGs, and 60 signals for BOLT-LMM applied to the imputed genotypes (Figure 7.3).

The higher total number of associated loci discovered using imputation than under the GeWAS model is expected mainly due to two factors. First, high accuracy in the imputation of common variants even in the presence of small reference

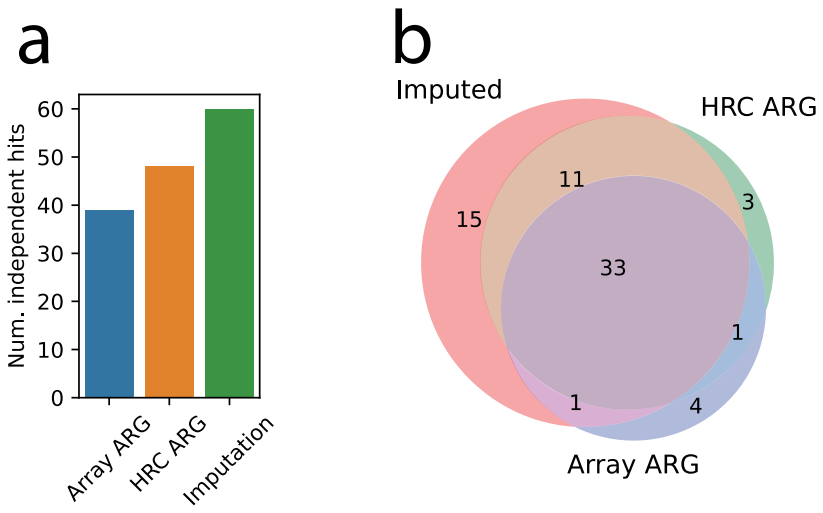


Figure 7.3: **a**. The total number of independently associated signals obtained for Africans and Central/South Asians for seven quantitative traits using three different methods. **b** Venn diagram of the independent association signals. Significantly associated markers from across study modalities were interpreted as overlapping if their respective genotypes satisfied $r^2 > 0.1$. Here, "imputation" refers to BOLT-LMM applied to the HRC imputation of the UK Biobank, "HRC ARG" refers to GeWAS analyses using ARGs inferred from high-quality HRC-imputed variants and "Array ARG" refers to GeWAS analyses using ARGs inferred using genotyping arrays only.

panels means that common causal variation is usually more accurately tagged by imputation than by an inferred ARG. Second, the high amount of rare variation tested under the ARG results in a more stringent significance threshold than for imputation. These two observations emphasize the role of the GeWAS primarily as a tool to analyze and detect rare variation in the genome. However, we did not find enrichment for rare variation among the independent HRC ARG variants compared with those uncovered using imputation (Figure 7.4), which may be due to the small sample sizes considered here.

Although the GeWAS study does not reveal a higher total number of associated loci, it can improve upon imputation where tagging of rare variants is lacking. Here, we highlight one such example, a rare clade overlapping GPLD1 as-

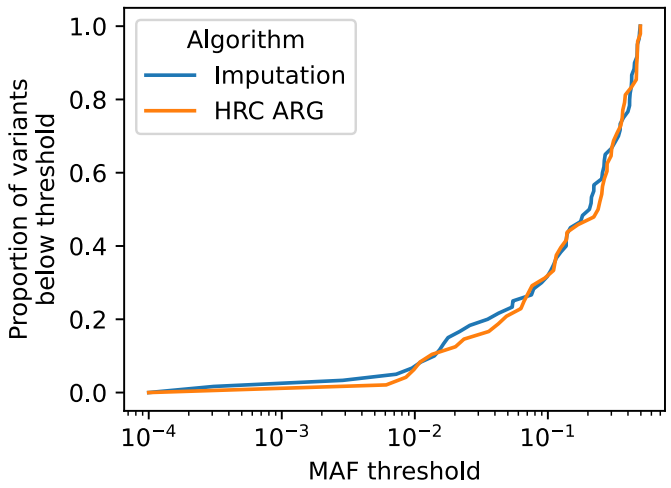


Figure 7.4: Distribution of allele frequencies for 48 independent association signals uncovered using the HRC ARG and 60 independent association signals uncovered using imputation. A higher line indicates a higher proportion of rare variants.

sociated with alkaline phosphatase levels in Africans. GPLD1 has previously been shown to be associated with alkaline phosphatase levels [6] and within the locus we find 2 independently associated GeWAS clades. One of these clades (MAF=0.01, $p = 3.7 \times 10^{-10}$) was, upon verification with exome data, found to be in moderate LD ($r^2 = 0.33$) with an AFR-specific missense variant. This clade was found to be a part of larger haplotype carrying variants in high LD with the variant (maximum $r^2 = 0.75$, $p = 3.4 \times 10^{-9}$, Figure 7.5).

The variant, rs146351134, has been annotated as deleterious under both SIFT [64] and Polyphen [1]. Moreover, it carries a CADD score of 33, indicating high deleteriousness [74], suggesting it is likely to be driving the signal. Variant annotations were retrieved using the Genome Aggregation Database (gnomAD) [37]. The variant was not tagged strongly by any imputed SNP in our data set and no other non-synonymous mutation was found to be in LD with the associated SNPs and ARG clades. The highest LD between rs146351134 and an imputed SNP passing all filtering criteria was found to be in a variant with $r^2 = 0.41$ and $p = 8.9 \times 10^{-4}$.

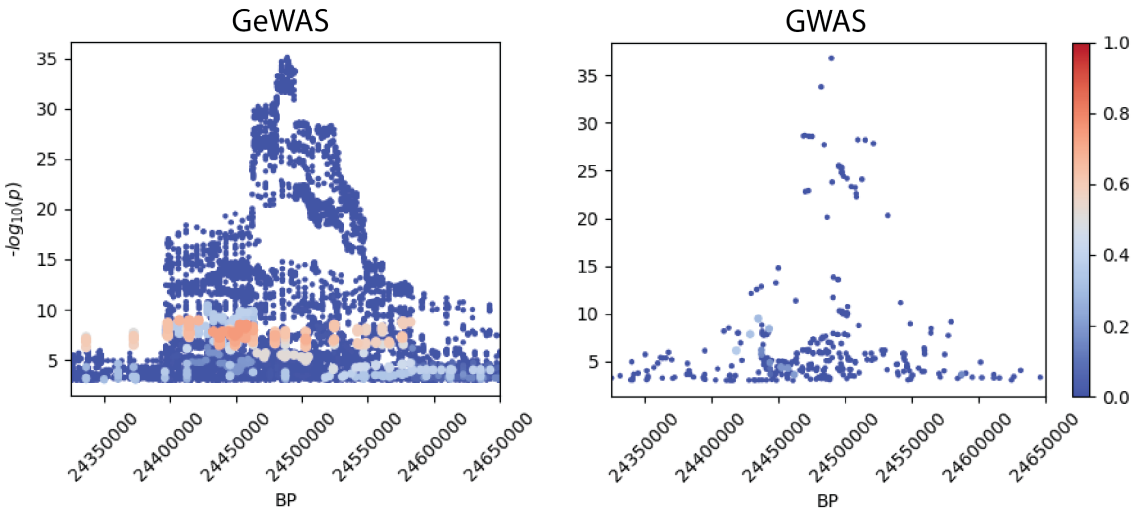


Figure 7.5: Locus zoom of association statistics for alkaline phosphatase in Africans over GPDL1 on chromosome 6. The left panel shows the HRC-ARG GeWAS and the right panel shows BOLT-LMM statistics. Colors represent correlation with rs146351134.

Together, our results show that the GeWAS is a viable tool to complement GWAS in the analysis of causal rare variation in small cohorts that have not been imputed using a large in-sample reference panel. The higher discovery rate observed using our HRC ARGs compared with ARGs built from genotyping arrays highlights the importance of using high-quality and densely inferred genealogies. Our simulations in Chapter 4 showed that the largest gains in accuracy of ARG inference using imputed variants compared with genotyping arrays are obtained in the presence of large in-sample reference panels, indicating that as panel sizes increase, so will the quality of inferred ARGs and their power to detect rare causal variants. Our computational benchmarks suggest Threads is well-equipped to process these larger data sets.

Chapter 8

Discussion

In this text we've introduced a new algorithm to infer ancestral recombination graphs (ARGs) from observed genotypes and presented several promising applications to familiar tasks in statistical genetics. We showed that our algorithm, Threads, is more scalable than other currently available ARG inference algorithms, while also achieving good accuracy across a range of metrics and simulation configurations. We've developed versions of Threads that can be applied to phased and unphased data, whole genomes and genotyping arrays, and we have successfully built large-scale ARGs from imputed sequences.

Nevertheless, Threads has some limitations which may affect its applicability in certain settings and we emphasize two of those here. First, Threads may achieve relatively poor accuracy in the inference of deep-time coalescence events compared with other methods. For large ARGs, new sequences tend to coalesce in recent time and in this case, threading and likelihood-based, segment-wise coalescence time estimation is an efficient approximation. Conversely, deep time is largely established by the time the first few hundred sequences have been threaded to the ARG. If deep time suffers from poor inference quality those problems may propagate to the rest of the inference process. ARG inference by threading does not model the

initial sequences jointly to infer topologies and coalescence times, which can result in lower accuracy than in less scalable methods that perform joint modelling. In Chapter 3.5 we introduced some helpful heuristics to improve deep-time inference, such as a PSMC-like step to improve the segmentation of threading instructions, but these alone are not enough to compensate for the lack of joint modelling. However, due to the modular nature of the threading instructions inferred by Threads, the threading instructions for the first few hundred sequences in a sample can be replaced with a "seed ARG" inferred using a more accurate method applied to those same sequences. The possibility of benefiting from both the scalability of Threads and the deep-time accuracy of slower methods through ARG seeding is an exciting direction for ARG inference which has not yet been fully explored.

Second, although the threading instructions inferred by Threads provide good compression rates and allow for efficient querying of genotypes, common operations applied directly to the threading instructions can be inefficient compared with fully assembled ARG data structures such as the `tskit` TreeSequence [38] and the `arg_needle_lib` ARG [99]. In our experiments we have observed that in some cases, especially with noisy data, assembly of the threading instructions into a full ARG can take as long as the inference process itself. However, this step depends on an external threading algorithm and it remains an open question whether more efficient assembly of threading instructions into an ARG data structure can be achieved.

Whole-genome sequenced biobanks that contain tens to hundreds of thousands of genomes provide an unprecedented opportunity to infer and analyse the fine-grained genealogical histories of large samples [23, 30, 87, 92] and the scalability of Threads paves the way towards these analyses. To explore ARG inference at this scale, we applied Threads to 10,000,000 high-quality imputed variants for 487,409 samples from the UK Biobank. This large-scale genome-wide genealogy captures

a wealth of information on the sample and is currently being used in several downstream analyses that are beyond the scope of this text.

Threading instructions also provide an efficient new way to compress genotype data, compressing the genotypes used to infer the large-scale UK Biobank ARG by up to 94% compared with the PLINK2-compressed input files [14], while also encoding an estimated genealogical history. The combination of ARG-based imputation and compression is an exciting prospect for imputation servers, which could improve accuracy in rare variants and deliver results containing both genotypes and inferred ARGs while consuming fewer resources than using traditional methods. Initial results for threading-based encoding of genotype dosages in Chapter 5.2.1 showed promising results, but more work is required to explore how ARGs can be used to store genotype information beyond phased, binary hard-calls.

The ARG-driven improvements over traditional methods for genotype imputation introduced in Chapter 6 highlight the value of accurate inference of recent ancestry in applications outside population genetics. In particular, we've demonstrated how traditional methods result in systematically inflated dosages for ultra-rare variants. This phenomenon occurs when the sequence being imputed coalesces into the reference panel in time more recent than the age of a mutation carried by its genealogical closest cousins, in which case the ARG can be used to improve imputation quality. This can be helpful as high-impact variants often undergo negative selection and tend to be rare [66].

One important observation from our experiments with ARG-based genotype imputation is that it is crucial to apply probabilistic modelling where possible and to keep in mind that the inferred ARG is only a point estimate in the large mathematical space of possible ARGs. This observation may in part explain why our ARG-based rare-variant phasing algorithm underperforms compared with traditional methods, even though it benefits from access to a reference ARG. The algo-

rithm does not account for uncertainty in estimation of coalescence times, threading targets or genomic position of segment break-points. Instead, it proceeds by simply reading coalescence times from an inferred point-estimate ARG of a common-variant scaffold. By comparison, the ARG-based imputation approach leverages uncertainty across all three dimensions of ARG inference via threading: coalescence times, genetic positions and threading targets.

In a final experiment, we studied whether the ARG might be used to increase the power to detect rare variants in association studies of populations where no large-scale imputation panel was available. To this end, we focused on samples of African and Central/South-Asian ancestry in the UK Biobank, the two largest non-European ancestry groups, imputed using the Haplotype Reference Consortium (HRC) panel [58]. Promising results from genealogy-wide association studies (GeWAS) of 337,000 White British samples had been previously demonstrated by Zhang et al. in [99] and we aimed to investigate whether those results might replicate in a smaller, non-European sample.

The strength of the ARG lies in uncovering associations between phenotypes and ultra-rare variants that are either unobserved in reference panels or poorly imputed. In our experiments, we did not find evidence that the GeWAS uncovers a high amount of variation missed by imputation, although in some cases the ARG provided better tagging of underlying likely-causal exome variants than imputation, such as the deleterious missense variant on GPLD1 described in Chapter 7.2. However, the lack of rare-variant association is likely due to the small sample size and larger studies are required to further assess the power of the ARG to uncover rare variation in samples of African and Central/South Asian ancestry.

Our non-European GeWAS also found a lower total number of independently associated loci than were uncovered using imputation. This is expected, as common variants are well imputed even when using a reference imputation panel that

is small or not population-specific. Thus, association studies of well-imputed common variants yields more accurate results than those of inferred ARGs. Although it remains possible that an ARG inferred using common variants may help tag high-frequency genomic variants that are poorly tagged by available markers, such as structural variants or indels, there is as of yet no evidence to support that hypothesis.

In our analyses of GeWAS associations detected for common variants, we found that even under variant clumping at the $r^2 = 0.1$-level (Zhang et al. [99] performed clumping at the $r^2 = 0.5$-level) combined with a conditional and joint analysis (COJO) [97], the ARG found a high number of non-independent association signals compared with imputation. This motivated us to use a highly stringent threshold of $r^2 = 0.01$. For common variants, the ARG is more likely to achieve imperfect tagging than imputation. If the lead SNP at an associated locus imperfectly tags the underlying causal variant, there is a higher risk that the locus contains other variants that weakly tag the causal variant but remain orthogonal to the lead SNP, leading to false positive association signals. As we found in our experiments, the GeWAS is especially susceptible to this type of signal duplication, due to the high sampling rate along the ARG and the imperfect tagging of common variants. Methods developed to obtain approximately independent association in genotyped or imputed variants, such as COJO [97], may thus not be as effective in this regime. Applying a highly stringent clumping, however, significantly reduced signal duplication in our experiments. Overall, these analyses highlight the importance of performing stringent post-processing of common-variant association signals detected using GeWAS.

A comparison of GeWAS studies using ARGs built directly from genotyping array data on one hand and imputed variants on the other emphasized the importance of using dense marker sets for ARG inference. Notably, the imputed-variant

ARGs uncovered around 25% more independent association signals than the genotyping array ARGs. These densely-inferred ARGs were only made possible by the development of our highly scalable ARG inference algorithm, Threads. At the same time, ARGs inferred using Threads contained a lower total number of edges than ARG-Needle, which had previously been applied in GeWAS studies [99], making ARG traversal more efficient.

The applications summarized above emphasize the versatility of ARGs for applications across a range of tasks in statistical genetics. Together with traditional applications in population genetics and other applications from the literature, the ARG is poised to become a standard feature of the statistical geneticist's toolbox. We believe that with Threads we have developed a tool that makes inference of ARGs simple and efficient, taking a further step towards the development of ARG-driven statistical genetics.

References

[1] Ivan Adzhubei et al. "Predicting Functional Effect of Human Missense Mutations Using PolyPhen-2". *Current Protocols in Human Genetics* 76 (1 2013), pp. 7.20.1–7.20.41.

[2] P Armitage. "Tests for Linear Trends in Proportions and Frequencies". *Biometrics* 11 (3 1955), pp. 375–386.

[3] Adam Auton et al. "A global reference for human genetic variation". *Nature* 526 (7571 2015), pp. 68–74.

[4] Gavin Band and Jonathan Marchini. "BGEN: a binary file format for imputed genotype and haplotype data". *bioRxiv* (2018), p. 308296.

[5] Daniel Barry and J A Hartigan. "Product Partition Models for Change Point Problems". *The Annals of Statistics* 20 (1 1992), pp. 260–279.

[6] Alison R Barton et al. "Whole-exome imputation within UK Biobank powers rare coding variant association and fine-mapping analyses". *Nature Genetics* 53 (8 2021), pp. 1260–1269.

[7] Franz Baumdicker et al. "Efficient ancestry and mutation simulation with msprime 1.0". *Genetics* 220 (3 2022).

[8] Brian L Browning et al. "Fast two-stage phasing of large-scale sequence data". *The American Journal of Human Genetics* 108 (10 2021), pp. 1880–1890.

[9] Brian L. Browning et al. "A One-Penny Imputed Genome from Next-Generation Reference Panels". *American Journal of Human Genetics* 103 (3 2018), pp. 338–348.

[10] Clare Bycroft et al. "The UK Biobank resource with deep phenotyping and genomic data". *Nature* 562 (7726 2018), pp. 203–209.

[11] Marta Byrska-Bishop et al. “High-coverage whole-genome sequencing of the expanded 1000 Genomes Project cohort including 602 trios”. *Cell* 185 (18 2022), 3426–3440.e19.

[12] Shai Carmi et al. “A renewal theory approach to IBD sharing”. *Theoretical Population Biology* 97 (2014), pp. 35–48.

[13] Francisco C. Ceballos et al. “Runs of homozygosity: Windows into population history and trait architecture”. *Nature Reviews Genetics* 19 (4 2018), pp. 220–234.

[14] Christopher C Chang et al. “Second-generation PLINK: rising to the challenge of larger and richer datasets”. *GigaScience* 4 (1 2015), s13742-015-0047–8.

[15] Petr Danecek et al. “The variant call format and VCFtools”. *Bioinformatics* 27 (15 2011), pp. 2156–2158.

[16] Petr Danecek et al. “Twelve years of SAMtools and BCFtools”. *GigaScience* 10 (2 2021), giab008.

[17] Richard Durbin. “Efficient haplotype matching and storage using the positional BurrowsWheeler transform (PBWT)”. *Bioinformatics* 30 (9 2014), pp. 1266–1272.

[18] Romain Fournier et al. “Haplotype-based inference of recent effective population size in modern and ancient DNA samples”. *Nature Communications* 14 (1 2023), p. 7945.

[19] Steven Gazal et al. “High level of inbreeding in final phase of 1000 Genomes Project”. *Scientific Reports* 5 (1 2015), p. 17453.

[20] Richard A Gibbs et al. “The International HapMap Project”. *Nature* 426 (6968 2003), pp. 789–796.

[21] R C Griffiths and P Marjoram. "Ancestral Inference from Samples of DNA Sequences with Recombination". *Journal of Computational Biology* 3 (4 1996), pp. 479–502.

[22] Robert C Griffiths and Paul Marjoram. "An ancestral recombination graph". *Progress in population genetics and human evolution* (1997), pp. 257–270.

[23] Bjarni V Halldorsson et al. "The sequences of 150,119 genomes in the UK Biobank". *Nature* 607 (7920 2022), pp. 732–740.

[24] Jotun Hein et al. *Gene Genealogies, Variation and Evolution. A Primer in Coalescent Theory*. Oxford University Press, 2004.

[25] Hussein A Hejase et al. "A Deep-Learning Approach for Inference of Selective Sweeps from the Ancestral Recombination Graph". *Molecular Biology and Evolution* 39 (1 2022), msab332.

[26] Asger Hobolth and Jens Ledet Jensen. "Markovian approximation to the finite loci coalescent with recombination along multiple sequences". *Theoretical Population Biology* 98 (2014), pp. 48–58.

[27] Robin J Hofmeister et al. "Accurate rare variant phasing of whole-genome and whole-exome sequencing data in the UK Biobank". *Nature Genetics* 55 (7 2023), pp. 1243–1249.

[28] Richard R Hudson. "Properties of a neutral allele model with intragenic recombination". *Theoretical Population Biology* 23 (2 1983), pp. 183–201.

[29] Anastasia Ignatieva et al. "The distribution of branch duration and detection of inversions in ancestral recombination graphs". *bioRxiv* (2023).

[30] Brynjar O Jensson et al. "Actionable Genotypes and Their Association with Life Span in Iceland". *New England Journal of Medicine* 389 (19 2023), pp. 1741–1752.

[31] Longda Jiang et al. “A resource-efficient tool for mixed model association analysis of large-scale data”. *Nature Genetics* 51 (12 2019), pp. 1749–1755.

[32] Longda Jiang et al. “A generalized linear mixed model association tool for biobank-scale data”. *Nature Genetics* 53 (11 2021), pp. 1616–1621.

[33] Yifan Jin and Jonathan Terhorst. “The solution path of the Li-Stephens haplotype copying model”. *bioRxiv* (2022), p. 2022.08.03.502674.

[34] Masahiro Kanai et al. “Empirical estimation of genome-wide significance thresholds based on the 1000 Genomes Project data set”. *Journal of Human Genetics* 61 (10 2016), pp. 861–866.

[35] Hyun Min Kang et al. “Efficient Control of Population Structure in Model Organism Association Mapping”. *Genetics* 178 (3 2008), pp. 1709–1723.

[36] Hyun Min Kang et al. “Variance component model to account for sample structure in genome-wide association studies”. *Nature Genetics* 42 (4 2010), pp. 348–354.

[37] Konrad J Karczewski et al. “The mutational constraint spectrum quantified from variation in 141,456 humans”. *Nature* 581 (7809 2020), pp. 434–443.

[38] Jerome Kelleher et al. “Efficient pedigree recording for fast population genetics simulation”. *PLOS Computational Biology* 14 (11 2018).

[39] Jerome Kelleher et al. “Inferring whole-genome histories in large population datasets”. *Nature Genetics* 51 (9 2019), pp. 1330–1338.

[40] Michelle Kendall and Caroline Colijn. “Mapping Phylogenetic Trees to Reveal Distinct Patterns of Evolution”. *Molecular Biology and Evolution* 33 (10 2016), pp. 2735–2743.

[41] Caleb Ki and Jonathan Terhorst. "Exact Decoding of a Sequentially Markov Coalescent Model in Genetics". *Journal of the American Statistical Association* (2023), pp. 1–14.

[42] J F C Kingman. "The coalescent". *Stochastic Processes and their Applications* 13 (3 1982), pp. 235–248.

[43] Augustine Kong et al. "Rate of de novo mutations and the importance of fathers age to disease risk". *Nature* 488 (7412 2012), pp. 471–475.

[44] Sandeep Koranne. *Handbook of Open Source Tools*. Ed. by Sandeep Koranne. Springer US, 2011, pp. 191–200.

[45] M K Kuhner and J Felsenstein. "A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates." *Molecular Biology and Evolution* 11 (3 1994), pp. 459–468.

[46] A H Land and A G Doig. "An Automatic Method of Solving Discrete Programming Problems". *Econometrica* 28 (3 1960), pp. 497–520.

[47] Heng Li and Richard Durbin. "Inference of human population history from individual whole-genome sequences". *Nature* 475 (7357 2011), pp. 493–496.

[48] Na Li and Matthew Stephens. "Modeling Linkage Disequilibrium and Identifying Recombination Hotspots Using Single-Nucleotide Polymorphism Data". *Genetics* 165 (4 2003), pp. 2213–2233.

[49] Y Lin et al. "A Metric for Phylogenetic Trees Based on Matching". *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 9 (4 2012), pp. 1014–1022.

[50] Jennifer Listgarten et al. "Improved linear mixed models for genome-wide association studies". *Nature Methods* 9 (6 2012), pp. 525–526.

[51] Po-Ru Loh et al. “Efficient Bayesian mixed-model analysis increases association power in large cohorts”. *Nature Genetics* 47 (3 2015), pp. 284–290.

[52] Po-Ru Loh et al. “Fast and accurate long-range phasing in a UK Biobank cohort”. *Nature Genetics* 48 (7 2016), pp. 811–816.

[53] Po-Ru Loh et al. “Reference-based phasing using the Haplotype Reference Consortium panel”. *Nature Genetics* 48 (11 2016), pp. 1443–1448.

[54] Po-Ru Loh et al. “Mixed-model association for biobank-scale datasets”. *Nature Genetics* 50 (7 2018), pp. 906–908.

[55] Gerton Lunter. “Haplotype matching in large cohorts using the Li and Stephens model”. *Bioinformatics* 35 (5 2019), pp. 798–806.

[56] Paul Marjoram and Jeff D Wall. “Fast "coalescent" simulation”. *BMC Genetics* 7 (1 2006), p. 16.

[57] Joelle Mbatchou et al. “Computationally efficient whole-genome regression for quantitative and binary traits”. *Nature Genetics* 53 (7 2021), pp. 1097–1103.

[58] Shane McCarthy et al. “A reference panel of 64,976 haplotypes for genotype imputation”. *Nature Genetics* 48 (10 2016), pp. 1279–1283.

[59] Gilean A.T. McVean and Niall J. Cardin. “Approximating the coalescent with recombination”. *Philosophical Transactions of the Royal Society B: Biological Sciences* 360 (1459 2005), pp. 1387–1393.

[60] Mark J Minichiello and Richard Durbin. “Mapping Trait Loci by Use of Inferred Ancestral Recombination Graphs”. *The American Journal of Human Genetics* 79 (5 2006), pp. 910–922.

[61] Daniel Müllner. “Modern hierarchical, agglomerative clustering algorithms”. *arXiv* (2011).

[62] Vagheesh M Narasimhan et al. “Estimating the human mutation rate from autozygous segments reveals population differences in human mutational processes”. *Nature Communications* 8 (1 2017), p. 303.

[63] Ardalan Naseri et al. “RaPID: ultra-fast, powerful, and accurate detection of segments identical by descent (IBD) in biobank-scale cohorts”. *Genome Biology* 20 (1 2019), p. 143.

[64] Pauline C Ng and Steven Henikoff. “SIFT: predicting amino acid changes that affect protein function”. *Nucleic Acids Research* 31 (13 2003), pp. 3812–3814.

[65] Pouria Salehi Nowbandegani et al. “Extremely sparse models of linkage disequilibrium in ancestrally diverse association studies”. *Nature Genetics* 55 (9 2023), pp. 1494–1502.

[66] Luke J O’Connor et al. “Extreme Polygenicity of Complex Traits Is Explained by Negative Selection”. *The American Journal of Human Genetics* 105 (3 2019), pp. 456–476.

[67] Pier Francesco Palamara. “ARGON: fast, whole-genome simulation of the discrete time Wright-fisher process”. *Bioinformatics* 32 (19 2016), pp. 3032–3034.

[68] Pier Francesco Palamara et al. “Length Distributions of Identity by Descent Reveal Fine-Scale Demographic History”. *The American Journal of Human Genetics* 91 (5 2012), pp. 809–822.

[69] Pier Francesco Palamara et al. “Leveraging Distant Relatedness to Quantify Human Mutation and Gene-Conversion Rates”. *The American Journal of Human Genetics* 97 (6 2015), pp. 775–789.

[70] Pier Francesco Palamara et al. “High-throughput inference of pairwise coalescence times identifies signals of selection and enriched disease heritability”. *Nature Genetics* 50 (9 2018), pp. 1311–1317.

[71] Alkes L Price et al. “Sensitive Detection of Chromosomal Segments of Distinct Ancestry in Admixed Populations”. *PLOS Genetics* 5 (6 2009), e1000519–.

[72] L R Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. *Proceedings of the IEEE* 77 (2 1989), pp. 257–286.

[73] Matthew D Rasmussen et al. “Genome-Wide Inference of Ancestral Recombination Graphs”. *PLOS Genetics* 10 (5 2014).

[74] Philipp Rentzsch et al. “CADD: predicting the deleteriousness of variants throughout the human genome”. *Nucleic Acids Research* 47 (D1 2019), pp. D886–D894.

[75] D F Robinson and L R Foulds. “Comparison of phylogenetic trees”. *Mathematical Biosciences* 53 (1 1981), pp. 131–147.

[76] Simone Rubinacci et al. “Genotype imputation using the Positional Burrows Wheeler Transform”. *PLoS Genetics* 16 (11 2020).

[77] Juba Nait Saada et al. “Identity-by-descent detection across 487,409 British samples reveals fine scale population structure and ultra-rare variant associations”. *Nature Communications* 11 (1 2020).

[78] Ahsan Sanaullah et al. “d-PBWT: dynamic positional BurrowsWheeler transform”. *Bioinformatics* 37 (16 2021), pp. 2390–2397.

[79] Aylwyn Scally and Richard Durbin. “Revising the human mutation rate: implications for understanding human evolution”. *Nature Reviews Genetics* 13 (10 2012), pp. 745–753.

[80] Stephan Schiffels and Ke Wang. “MSMC and MSMC2: The Multiple Sequentially Markovian Coalescent”. Ed. by Julien Y Dutheil. Springer US, 2020, pp. 147–166.

[81] Yichen Si et al. “Why are rare variants hard to impute? Coalescent models reveal theoretical limits in existing algorithms”. *Genetics* 217 (4 2021), iyab011.

[82] Martin R Smith. “Robust Analysis of Phylogenetic Tree Space”. *Systematic Biology* 71 (5 2022), pp. 1255–1270.

[83] Yun S Song. “Na Li and Matthew Stephens on Modeling Linkage Disequilibrium”. *Genetics* 203 (3 2016), pp. 1005–1006.

[84] Leo Speidel et al. “A method for genome-wide genealogy estimation for thousands of samples”. *Nature Genetics* 51 (9 2019), pp. 1321–1329.

[85] Jeffrey P Spence and Yun S Song. “Inference and analysis of population-specific fine-scale recombination maps across 26 diverse human populations”. *Science Advances* 5 (10 2023), eaaw9206.

[86] Gulnara R Svishcheva et al. “Rapid variance componentsbased method for whole-genome association analysis”. *Nature Genetics* 44 (10 2012), pp. 1166–1170.

[87] Daniel Taliun et al. “Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program”. *Nature* 590 (7845 2021), pp. 290–299.

[88] Pan-UKB team. “Pan-UK Biobank”. *https://pan.ukbb.broadinstitute.org* (2020).

[89] Jonathan Terhorst et al. “Robust and scalable inference of population history from hundreds of unphased whole genomes”. *Nature Genetics* 49 (2 2017), pp. 303–309.

[90] Xiaowen Tian et al. “Estimating the Genome-wide Mutation Rate with Three-Way Identity by Descent”. *The American Journal of Human Genetics* 105 (5 2019), pp. 883–893.

[91] Xiaowen Tian et al. “Estimating the genome-wide mutation rate from thousands of unrelated individuals”. *The American Journal of Human Genetics* 109 (12 2022), pp. 2178–2184.

[92] The National Genomics Research Library v5.1. “Genomics England” (2020).

[93] Jeffrey D. Wall et al. “Estimating genotype error rates from high-coverage next-generation sequence data”. *Genome Research* 24 (11 2014), pp. 1734–1739.

[94] Daniel Wegmann et al. “Recombination rates in admixed individuals identified by ancestry-based inference”. *Nature Genetics* 43 (9 2011), pp. 847–853.

[95] Anthony Wilder Wohns et al. “A unified genealogy of modern and ancient genomes”. *Science* 375 (6583 2023), eabi8264.

[96] Jian Yang et al. “GCTA: A Tool for Genome-wide Complex Trait Analysis”. *The American Journal of Human Genetics* 88 (1 2011), pp. 76–82.

[97] Jian Yang et al. “Conditional and joint multiple-SNP analysis of GWAS summary statistics identifies additional variants influencing complex traits”. *Nature Genetics* 44 (4 2012), pp. 369–375.

[98] Jian Yang et al. “Advantages and pitfalls in the application of mixed-model association methods”. *Nature Genetics* 46 (2 2014), pp. 100–106.

[99] Brian C. Zhang et al. “Biobank-scale inference of ancestral recombination graphs enables genealogical analysis of complex traits”. *Nature Genetics* 55 (5 2023), pp. 768–776.

[100] Wei Zhou et al. “Efficiently controlling for case-control imbalance and sample relatedness in large-scale genetic association studies”. *Nature Genetics* 50 (9 2018), pp. 1335–1341.

[101] Xiang Zhou and Matthew Stephens. “Genome-wide efficient mixed-model analysis for association studies”. *Nature Genetics* 44 (7 2012), pp. 821–824.

[102] Ying Zhou et al. “A Fast and Simple Method for Detecting Identity-by-Descent Segments in Large-Scale Data”. *The American Journal of Human Genetics* 106 (4 2020), pp. 426–437.

[103] Sebastian Zöllner and Jonathan K Pritchard. “Coalescent-Based Association Mapping and Fine Mapping of Complex Trait Loci”. *Genetics* 169 (2 2005), pp. 1071–1092.