Incorporating Theoretical Concepts and Models from Population Genetics into Machine Learning Methods

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
M. Sc. Klara Elisabeth Burger
aus Neuenburg am Rhein

Tübingen
2024

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	14.02.2025
Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Dr. Franz Baumdicker
2. Berichterstatterin:	Prof. Dr. Ulrike von Luxburg
3. Berichterstatter:	Dr. Matteo Fumagalli

Für meine Eltern.

Abstract

Machine learning is an ever-growing scientific field with increasing impact on our lives and has already revolutionized areas such as speech recognition, natural language processing and image classification. Machine learning is also of great interest to population genetics, especially as next-generation sequencing methods provide ever-larger genomic data sets that challenge traditional model-based estimators. In addition, new simulation software allows efficient generation of training data. While machine learning is already widely applied in population genetics, this emerging methodology also poses challenges, particularly with respect to robustness and interpretability. A promising strategy to address these challenges is to incorporate theoretical concepts and models from population genetics into machine learning methods. In this thesis, we present two approaches and demonstrate their advantages: First, by using a neural network to estimate the scaled mutation rate, we present a concept of how well-established model-based estimators can be integrated into the loss functions of supervised methods. Second, we incorporate key population genetic concepts such as the fixation index and Hardy-Weinberg equilibrium into an unsupervised hierarchical soft clustering method to infer population ancestry. These methods demonstrate that combining theoretical insights with data-driven learning not only enables processing of large data sets, capturing and exploiting underlying data dynamics, but also improves robustness and interpretability. With the ever-increasing availability of genetic data, such approaches have enormous potential to significantly deepen our understanding of genetic variability and evolution.

Zusammenfassung

Maschinelles Lernen ist ein stetig wachsendes Forschungsgebiet mit zunehmendem Einfluss auf unser Leben und hat bereits Bereiche wie Spracherkennung, linguistische Datenverarbeitung und Bilderkennung revolutioniert. Maschinelles Lernen ist auch für die Populationsgenetik von großem Interesse, da Next-Generation-Sequencing-Methoden immer größere genomische Datensätze erzeugen, die herkömmliche modellbasierte Schätzer vor Herausforderungen stellen. Darüber hinaus ermöglicht neue Simulationssoftware die effiziente Generierung von Trainingsdaten. Obwohl maschinelles Lernen in der Populationsgenetik bereits weit verbreitet ist, bringt diese neue Methodik auch Herausforderungen mit sich, insbesondere in Bezug auf Robustheit und Interpretierbarkeit. Eine vielversprechende Strategie zur Überwindung dieser Herausforderungen besteht darin, theoretische Konzepte und Modelle der Populationsgenetik in maschinelle Lernmethoden zu integrieren. In dieser Dissertation präsentieren wir zwei Ansätze und demonstrieren deren Vorteile: Erstens stellen wir anhand der Schätzung der skalierten Mutationsrate mit einem neuronalen Netz ein Konzept vor, wie etablierte modellbasierte Schätzer in die Verlustfunktionen überwachter Methoden integriert werden können. Zweitens integrieren wir zentrale populationsgenetische Konzepte wie den Fixationsindex und das Hardy-Weinberg-Gleichgewicht in eine unüberwachte hierarchische Soft-Clustering-Methode, um auf die Abstammung von Populationen zu schließen. Diese Methoden zeigen, dass die Kombination theoretischer Erkenntnisse mit datengesteuertem Lernen nicht nur die Verarbeitung großer Datensätze ermöglicht, wobei zugrundeliegende Datendynamik erfasst und genutzt werden kann, sondern auch die Robustheit und Interpretierbarkeit verbessert. Angesichts der stetig zunehmenden Verfügbarkeit von genetischen Daten haben solche Ansätze ein enormes Potenzial, unser Verständnis der genetischen Variabilität und Evolution erheblich zu vertiefen.

Acknowledgements

First, I would like to thank Franz for his support and guidance over the past four years. His enthusiasm for both population genetics and machine learning, as well as the regular, stimulating discussions that kept our projects moving forward, have been invaluable. I also appreciate his flexibility in making the most of every situation, both during and after Covid's prime, as well as the freedom he gave me to approach the challenges I was presented with in the best way possible.

I would also like to thank Ulrike for her willingness to be my co-supervisor, for the great collaboration that led to the second publication, and for her honest and reliable advice, especially during the final stages of my PhD. Likewise, I am grateful to Matteo for serving as my external reviewer and for his valuable feedback on my projects. Special thanks also go to Solveig for the excellent collaboration on the tangleGen project; it has always been a true pleasure to work with her and I appreciate her advice regarding the PhD process and related topics. I would also like to thank my colleagues Johannes, Hannah, and especially Axel, for the many coffee breaks and our conversations about both science and life in general.

I am deeply grateful to my parents and my siblings, Hannah and Friedrich, for their unwavering support throughout my PhD, and for always being there for me unconditionally. I am especially thankful to Vera, who has been an incredible friend and constant companion from the first semester of our mathematics studies until today, as we are finishing our PhDs almost simultaneously. Finally, I would like to thank my friends for accompanying me on this journey; it would not have been the same without them. The support of the people closest to me has played a crucial role in bringing me to where I am today.

Contents

Chapter 1

Introduction

> *"The capacity to blunder slightly is the real marvel of DNA. Without this special attribute, we would still be anaerobic bacteria and there would be no music."*
>
> Lewis Thomas, 1977

Genetic diversity is the foundation of evolution, enabling species to develop, adapt to environmental changes and ultimately to survive. The earliest traces of life on Earth date back at least 3.77 billion years [Dodd et al., 2017]. From these early life forms, modern eukaryotes, such as animals, plants and fungi, developed from single-celled prokaryotic organisms [Weiss et al., 2016]. Among these eukaryotes, *Homo rudolfensis*, who lived around 2.4 million years ago, is considered one of the earliest ancestors of *Homo sapiens* [Schrenk et al., 1993]. Today's *Homo sapiens* is the result of millions of years of evolution, characterized by important developments such as bipedalism and the ability to sweat - both of which were essential for persistence hunting and thus survival [Folk and Semken, 1991]. Adaptations to specific environments, such as high altitude or low solar radiation, have also significantly shaped human life as we know it today [Muehlenbein, 2010]. Even in the present, genetic variation remains central to the survival of species. This is particularly evident in the conservation of endangered species. If genetic variation decreases, as it inevitably does as populations decline, the risk of extinction increases, for example, because of the increased risk of inbreeding and, therefore, of rare hereditary diseases. Finally, the future will also be affected by genetic variation. Climate change alone is expected to drastically alter living conditions on Earth [IPCC, 2023], and the ability to adapt will be essential for the survival of many species, including our own.

Understanding the complex patterns of genetic variation is the aim of population genetics [Etheridge, 2012]. Genetic differences between individuals and populations are based on variations in the genome, with genetic information for most species being encoded in the double helix of nucleotides known as deoxyribonucleic acid (DNA). Individual traits are determined by segments of DNA, genes, which can exist in different variants called alleles. There are many factors that influence genetic variation within a species, not only factors that affect the genes themselves, but also epigenetic effects that influence the expression of genes. The main forces that generate genetic variation include mutation, the introduction of new genetic variants; recombination, the rearrangement of existing genetic material; and migration, the introduction of variants from other populations [Wakeley, 2009]. Other factors, such as natural selection or genetic drift, subsequently influence genetic variation by changing the frequency of certain genetic variants within a population. These processes alter the genetic composition of a population over time — a phenomenon known as evolution [Dobzhansky, 1951].

Traditionally, population genetics has been a theory-driven field that relies heavily on mathematical models that assume idealized conditions. The advent of next-generation sequencing methods has made increasingly large genomic data sets available [Koboldt et al., 2013], providing researchers with a tremendous opportunity to gain new insights while simultaneously challenging established model-based methods in population genetics. Machine learning, on the other hand, offers techniques that can effectively handle large data sets and account for the underlying data dynamics. Often described as one of the disruptive technologies of the future [Jordan and Mitchell, 2015], machine learning is already having a significant impact on our daily lives. For example, image classification has been revolutionized by machine learning and is now widely used in various fields, including security and healthcare. Digital assistants such as Siri and Alexa are used by many people on a daily basis, as is ChatGPT, which has recently received considerable attention. In population genetics, machine learning has been used for nearly three decades. However, challenges remain, with two of the most crucial being the interpretability and robustness of these methods [Huang et al., 2024; Korfmann et al., 2023]. The highly theoretical nature of population genetics is thereby often underutilized, and a promising approach to tackle these challenges is to integrate the rich theoretical insights of population genetics into machine learning.

This thesis presents novel machine learning methods for population genetics that incorporate established population genetic concepts and methods. Chapters 1.1 and 1.2 provide the fundamental concepts of population genetics and machine

learning that form the foundation of this thesis. Chapter 1.3 then outlines how the two methods presented in this thesis contribute to more explainable and robust machine learning in population genetics.

1.1 Population Genetics

"All models are wrong but some are useful."
George Box, 1978

To unravel the complex patterns of genetic variation, population geneticists have long relied on mathematical models to provide a workable framework for analyzing evolutionary dynamics. Tree structures that model inheritance processes have been central to our understanding since Darwin [Darwin, 1837]. Although these models inevitably oversimplify, they remain important tools for analyzing factors influencing genetic variation. The following section introduces the core concepts and models of population genetics that are essential to this thesis. For a comprehensive introduction, see [Ewens, 2004; Wakeley, 2009] or [Durrett, 2008].

In most organisms, the genetic information necessary for life is encoded in DNA, which consists of strands of nucleic acid with four possible nucleobases [Wakeley, 2009]. Depending on the species, this genetic information may be stored in one (haploid), two (diploid) or more (polyploid) sets of chromosomes. A classic approach to analyzing DNA sequences in population genetics is to represent each sequence as, for example, a binary sequence (zeros and ones) for haploid species or a ternary sequence (zeros, ones and twos) for diploid species. Typically, the ancestral state is assumed to be known for each site of the DNA sequence and the differences from that state are counted at each site. Assuming the sites are biallelic, that is each site can have only two alleles, the maximum difference is one for haploid species and two for diploid species. Sites that match the ancestral state across all samples are usually excluded from further analysis, with each remaining site being considered a single-nucleotide polymorphism (SNP). For a sample consisting of DNA sequences from multiple individuals, such a representation in matrix form is also called **genotype matrix**. An example of a genotype matrix is added in Fig. 1.1 C.

In population genetics, particular attention is paid to the process of inheritance describing the evolution of DNA sequences in a population over time. Modeling this process is complex as it can be influenced by various factors, such as differences in fitness among individuals that lead to different expected numbers of offspring.

One of the most commonly used population models for inheritance, albeit a very simplistic one, is the **Wright-Fisher model** [Wakeley, 2009; Wright, 1931]. For a population of N haploid individuals, this model assumes that all individuals die at the end of each generation and are replaced by N offspring, resulting in non-overlapping generations. Each generation is formed by random sampling with replacement from the previous generation, leading to some individuals contributing no offspring to the next generation, while others contribute multiple offspring. The genome is always inherited from the parent and the population size is constant. In fact, as the genomes of each generation only depend on the genomes of the previous one, the Wright-Fisher model also forms a Markov-Chain [Ewens, 2004]. For a population of N diploid individuals, the model assumes that each of the $2N$ genomes is inherited from a single parent.

In reality, the assumptions of the Wright-Fisher model, like those of most models in population genetics, are rarely met by actual populations. For example, random mating, non-overlapping generations and constant population size do not occur in humans. To account for these discrepancies, population genetics often uses the concept of **effective population size** (N_e) [Ewens, 2004]. This concept aligns the model with reality by adjusting only a single parameter, N_e, representing the size of an idealized population that would have the same characteristics as the real population for a particular trait of interest. In this context, an ideal population is one in which each individual has an equal chance of passing on its genes to the next generation. In humans, for example, an ideal population might be one with random mating, a one-to-one ratio of females to males, a constant population size, and non-overlapping generations. In reality, most populations are not ideal, and thus the effective population size is usually much smaller than the census population size.

Given a sample of n DNA sequences from a subset of a population, we can trace the series of ancestors back through time, defining a **genealogical tree** or **genealogy** with n leaves. Starting from one leaf, we follow the ancestral lineage upwards back in time, and as soon as a common ancestor of two DNA sequences is found, their ancestral lineages coalesce into one. This process continues until all lineages converge at a single common ancestor. An exemplary genealogical tree for $n = 4$ is shown in Fig. 1.1 A. While the tree topology is unique for very small sample sizes (e.g. $n = 2$ or $n = 3$), the number of possible topologies grows rapidly as n increases. Fig. 1.2 displays two possible tree topologies for $n = 4$. For a constant effective population size (N_e), this genealogical process is modeled by **Kingman's coalescent** [Kingman, 1982], which serves as a limit of the Wright-Fisher model for large populations [Durrett, 2008]. Kingman's coalescent, or coalescent theory

in general, is a major achievement in population genetics because it provides a mathematical framework for analyzing the past of DNA sequences [Berestycki, 2009]. The value of this theory rests mainly on four features that greatly increase its applicability: it is a sample rather than a population-based theory, it is a mathematically elegant and, above all, efficient approach, and it is particularly suitable for DNA sequence data [Fu and Li, 1999; Hudson et al., 1990]. However, in reality the genealogical tree is generally unknown and needs to be inferred.

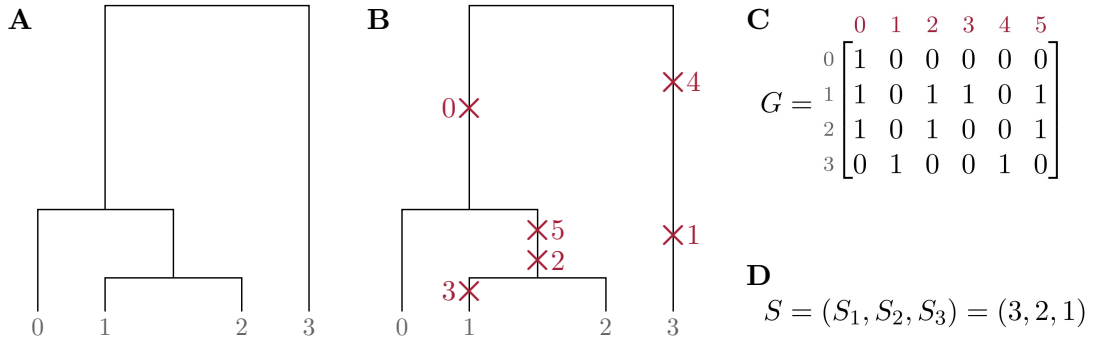


Figure 1.1: **A: Genealogical tree** for set of 4 samples. **B: Genealogical tree with exemplary mutations**, marked as red crosses, for a set of 4 samples. **C: Genotype matrix** corresponding to the genealogical tree in B, with individuals/samples in the rows and mutations in the columns. **D: Site frequency spectrum** corresponding to the genealogical tree in B. Individuals are assumed to be haploid.

Mutations are random and undirected changes in the genetic material and are central to genetic variation. They can occur, for example, through errors in DNA replication and there are various types of mutations that differ in location, extent and impact on the protein sequence. Modeling these processes is complex. For example, mutations between the nucleotides adenine and guanine (purine class) or between cytosine and thymine (pyrimidine class) occur more frequently than mutations between the two classes, which can be taken into account when modeling mutations [Pardoux, 2024]. In this thesis, as is common practice, we make several simplifying assumptions. We consider point mutations, which are mutations that change only a single nucleotide base or site. Furthermore, we assume these mutations to be neutral, that is they do not affect the fitness of individuals. In addition, we allow only two possible states for each site: ancestral (unmutated) and derived (mutated). These assumptions are fairly reasonable given the large size of genomes (about 3 billion nucleotide base pairs in humans), with only a small fraction of these bases (about 1-2% in humans) being protein-coding,

and the fact that most mutations are indeed point mutations [Siegel et al., 1999].

A commonly used model in population genetics for mutation is the **infinite sites model** [Kimura, 1969], which assumes that the DNA sequence has an infinite number of sites, such that each mutation occurs at a previously unmutated site. Given the length of a DNA sequence and the magnitude of mutation rates (in humans estimated to be about 10^{-8} per base pair per generation [Scally and Durbin, 2012]), this assumption is both practical and robust. Therefore, under these assumptions, each mutation is irreversible, is inherited by subsequent generations, and results in a new variant at the specific site. For this reason, the genealogical tree and the mutations can then be considered as two independent processes. An exemplary genealogical tree with mutations is shown in Fig. 1.1 B.

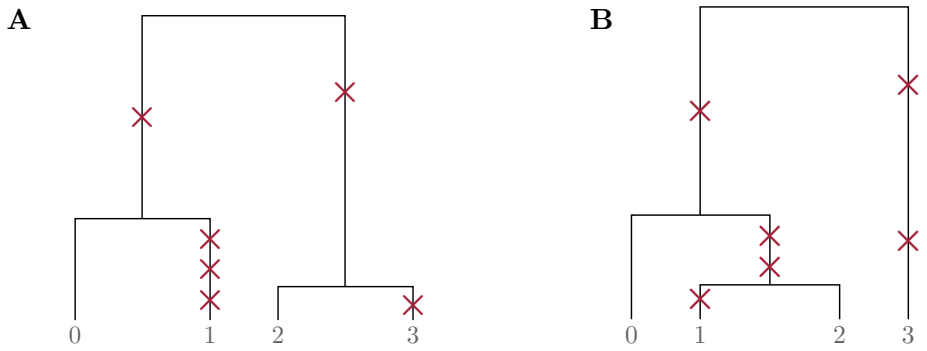


Figure 1.2: **Minimal example illustrating the link between tree topology and SFS without recombination**, $n = 4$. In the absence of recombination, a more balanced tree forces more zero entries in the site frequency spectrum, revealing information about the underlying tree topology. This effect is particularly visible at higher mutation rates. **A:** Balanced genealogical tree with an SFS of $S = (S_1, S_2, S_3) = (4, 2, 0)$. **B:** Unbalanced genealogical tree with an SFS of $S = (3, 2, 1)$. The non-zero SFS entry $S_3 > 0$ indicates an unbalanced tree topology. A balanced genealogical tree for $n = 4$ cannot have a branch leading to three individuals.

Genetic variation is increased not only by mutation but also, to a considerable extent, by **recombination**, that is the rearrangement of genetic material that results in a new combination of alleles being passed on to the offspring. In sexually reproducing species, this process occurs, for example, through crossover during meiosis, where homologous chromosomes exchange segments, resulting in new combinations of genes. When we think of recombination in the context of population genetics, we think of recombination events on the DNA sequence, or

the chromosome. Suppose the chromosome has a total length of 1, and there is exactly one recombination event at site x, $x \in (0, 1)$. The recombination event then leads to a rearrangement of the genetic material left and right of x, in the sense that the left part $(0, x)$ may now have a different ancestor and therefore a different genealogy than the right part $(x, 1)$. For sexually reproducing species, this means that although each zygote (fertilized egg) initially contains one set of chromosomes from the mother and father, it may include genes from all four grandparents instead of just two, which greatly increases genetic variability. Therefore, multiple recombination events lead to a series of dependent genealogies along the chromosome. In the limit of high recombination rates, all sites have their own genealogy which can all be considered independent. Thus, with recombination, we do not consider a single genealogy for a sample of DNA sequences. Instead, we examine a sequence of dependent genealogies along the chromosome, referred to as the **ancestral recombination graph** (**ARG**). Fig. 1.3 shows an example ARG.

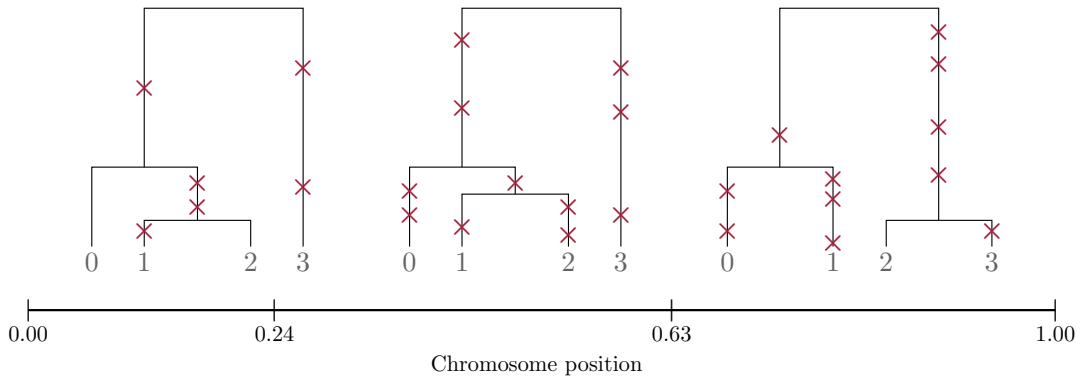


Figure 1.3: **Ancestral recombination graph** for $n = 4$. Recombination events at site 0.24 and 0.63 lead to a series of dependent genealogical trees along the chromosome. Exemplary mutations are marked with red crosses on the corresponding genealogical trees.

Finally, we introduce some summary statistics and concepts used to assess genetic variability within and between populations. One of the most common summary statistics to measure genetic variation is the **site frequency spectrum** (**SFS**) $S = S_1, \ldots, S_{n-1}$ with S_i being the number of mutations that occur on a branch that is ancestral to exactly i out of n individuals. The SFS can also be computed directly from the genotype matrix. An example of an SFS is shown in Fig. 1.1 D and Fig. 1.2. Many model-based estimators are based on the SFS, and we will use it as input to a neural network in the first publication. Fig. 1.2 shows that although the

underlying genealogical tree is generally unknown, this straightforward statistic can already provide some insight into the genealogical history of the sample [Wakeley, 2009].

Two important concepts for assessing genetic variability within and between populations are the Hardy-Weinberg equilibrium and the fixation index. The **Hardy-Weinberg equilibrium** (**HWE**) assumes a randomly mating diploid population free from evolutionary forces and describes genotype frequencies within populations [Gillespie, 1998; Hardy, 1908; Weinberg, 1908]. Given two alleles, A with frequency p and a with frequency q (where $p + q = 1$), the genotype frequencies are described by the equation

$$(p + q)^2 = p^2 + 2pq + q^2 = 1,$$

where p^2 is the frequency of the homozygous genotype AA, $2pq$ is the frequency of the heterozygous genotype Aa, and q^2 is the frequency of the homozygous genotype aa. Despite its simplifying assumptions, the model often holds remarkably well for large populations [Coop, 2020]. It is therefore commonly used to calculate expected genotype frequencies when allele frequencies p and q are known, or to examine whether populations are well mixed by assessing the deviation from an assumed HWE. In this thesis, we use the HWE to evaluate the plausibility of inferred populations.

To assess the relevance of a SNP to a population structure, we will use the **fixation index** ($\boldsymbol{F_{ST}}$) for ancestry inference in this thesis. The fixation index is a commonly used measure of genetic differentiation that compares genetic variability within and between populations [Wright, 1943, 1949]. It is defined as [Coop, 2020]

$$F_{ST} = 1 - \frac{H_S}{H_T} = 1 - \frac{2p_S(1 - p_S)}{2p_T(1 - p_T)},$$

where H_S is the expected heterozygosity within a subpopulation S, and H_T is the expected heterozygosity in the total population T, with $p_S \in [0, 1]$ and $p_T \in (0, 1)$ being the corresponding allele frequencies. The F_{ST} value ranges from 0 to 1, where 0 indicates no genetic differentiation between S and T at a given SNP, and values close to 1 indicate strong genetic differentiation. For example, if all individuals in population S are homozygous for a particular SNP (that is $p_S = 0$ or $p_S = 1$), the F_{ST} value for that SNP will be 1. On the other hand, if the allele frequencies are equal in both S and T ($p_S = p_T$), the F_{ST} value will be 0.

1.2 Machine Learning

"What we want is a machine that can learn from experience."
Alan Turing, 1947

Population genetics is a theory-driven field, and model-based estimators have been the standard approach to inference for decades. While these model-based estimators provide valuable insights, they often assume idealized situations, such as no recombination or infinite recombination scenarios, which rarely reflect reality. As a result, they often cannot account for complex dynamics underlying the data. In addition, many model-based methods are not designed to handle large data sets.

In the 21st century, research capabilities within population genetics have changed significantly: With the advent of next-generation sequencing methods, increasingly large genomic datasets are becoming available as the cost of sequencing DNA continues to decrease [Koboldt et al., 2013]. This represents an enormous opportunity for researchers to gain new insights, but it also highlights the need for new methodological approaches. To take advantage of the available data, we need methods that can efficiently process large amounts of genomic data while recognizing and exploiting the underlying dynamics within the data.

Unlike model-based methods in population genetics, machine learning methods benefit from large data sets and have the flexibility to adapt to complex underlying data dynamics. The approach to estimation in machine learning is fundamentally different from that of model-based estimators: Instead of developing a specific model for a particular application that best approximates our domain knowledge, machine learning starts with coarse methods with many parameters. These methods are then trained on application-specific data, adjusting their parameters to optimize performance in terms of a quality measure, such as a loss or cost function [Alpaydin, 2020]. This allows machine learning methods to discover and exploit hidden patterns in the data. The use of these methods in population genetics is further facilitated by the availability of powerful population genetic simulators, such as *ms* and *msprime* [Baumdicker et al., 2022; Hudson, 2002], that can efficiently simulate large training data sets.

With the availability of ever larger data sets and computing power, machine learning has revolutionized many areas of research [Jordan and Mitchell, 2015], including speech recognition [Hinton et al., 2012], natural language processing [Devlin et al., 2019], and image classification [Krizhevsky et al., 2012]. At its core, according to [Alpaydin, 2020], machine learning is defined as the programming of

computers to optimize a performance criterion based on data, and it can be broadly categorized into three areas: supervised (or predictive) learning, unsupervised (or descriptive) learning, and, less commonly, reinforcement learning [Bishop, 2006]. For a comprehensive introduction, see [Alpaydin, 2020; Bishop, 2006; Goodfellow, 2016].

In **supervised learning**, the goal is to learn a function that maps the input data to the corresponding output label. The input data consists of tuples, each containing a data point (e.g. the SFS for a set of samples) and its corresponding target label (the underlying scaled mutation rate). These labels are used in the loss function to evaluate the neural network's output. After training, the learned function can be used to make predictions on unseen, unlabeled data.

Neural networks are a common example of supervised learning. While they can also be trained unsupervised, we will focus on the supervised case since the first publication considers a supervised neural network. A neural network is structured in layers of neurons or nodes, generally divided into input, hidden and output layers, and the architecture determines the type of function on which the estimator is based. A fundamental architecture is the dense feedforward neural network, where each layer is a function of the previous one and every neuron is connected to all neurons of the subsequent layer. The input layer consists mainly of the input features, and the first hidden layer $h^{(1)}$ can then be constructed, for example, as follows [Goodfellow, 2016]

$$h^{(1)} = g^{(1)}\left(W^{(1)\top}x + b^{(1)}\right)$$

where g is an activation function, W are the weights, x are the input features and b is a bias node. The activation function is usually non-linear and is chosen based on the data and the task. The second hidden layer $h^{(2)}$ is then constructed as

$$h^{(2)} = g^{(2)}\left(W^{(2)\top}h^{(1)} + b^{(2)}\right)$$

and so on, with the last layer being the output layer, which is closely linked to the task. The number of nodes and the activation function can vary from layer to layer and must be chosen by the user. During training, the parameters — weights and biases — of the neural network are optimized with respect to a loss function, such as the mean squared error. Training is typically performed in batches of training samples using stochastic gradient decent: backpropagation is used to compute the gradient of the loss function with respect to the network parameters, which are then updated in the direction of the steepest descent for each batch [Alpaydin, 2020]. After a complete pass through the entire training data set,

called an epoch, the error is monitored on a separate validation data set and used, for example, for the regularization method known as early stopping, a criterion for stopping training if the validation error does not improve for a specified number of epochs. Such regularization methods reduce the risk of overfitting and improve generalization. Finally, the error of the trained neural network on a previously unseen test data set is computed as an estimate of the generalization error. While a dense feedforward neural network of sufficient size can theoretically approximate any function (universal approximation theorem [Hornik, 1991; Hornik et al., 1989]), there is no guarantee that the neural network will learn the desired function, since the optimization algorithm may not find it, may choose another function due to overfitting or the data may not capture it [Goodfellow, 2016]. In addition, the required complexity of the neural network, such as the number of layers and parameters, may exceed the available computing infrastructure. To achieve more efficient training and better generalization, other types of hidden layers may be preferred, depending on the application and data structure. For example, in a convolutional neural network [LeCun et al., 1989, 1998], convolutional and pooling layers are designed to capture local structures in grid-like data, such as correlations between nearby pixels in an image. This approach improves pattern recognition and significantly reduces the number of weights that need to be learned through weight sharing [Alpaydin, 2020]. Training a neural network with multiple hidden layers is also known as *deep learning*. In the first publication, Chapter 2.1, we use a dense feedforward neural network to estimate the scaled mutation rate from the SFS for different recombination scenarios.

In **unsupervised learning**, the training data has no explicit labels and the goal is often to discover patterns or structures within the data. Cluster methods that group data points, such as individuals, based on similarities are a classic example of unsupervised learning [Xu and Tian, 2015]. Among these methods, k-means is one of the most well known. It divides data points into k clusters in order to minimize the within-cluster variance [MacQueen et al., 1967]. Hereby, k-means requires the user to specify the number of clusters, k, in advance, which may not always be straightforward. In contrast, other methods, such as Tangles [Klepper et al., 2023], a clustering method leveraging graph theoretical concepts, do not require such a specification in advance. Instead, Tangles identifies the optimal number of clusters as part of its hierarchical clustering process. Tangles form the basis of the second publication, Chapter 2.2, for the inference of population ancestry. A detailed introduction to the Tangles clustering method can be found in Chapter 2.2.

The use of **machine learning in population genetics** spans nearly three decades, with the first known application of neural networks to the field dating back to 1996 [Cornuet et al., 1996]. Methods such as principal component analysis (PCA) [Patterson et al., 2006; Price et al., 2006], k-means clustering [Jombart et al., 2010], or hidden Markov models [Boitard et al., 2009; Kern and Haussler, 2010; Mailund et al., 2011] were also used early on. Later, methods like support vector machines [Ronen et al., 2013; Schrider and Kern, 2015] or boosting [Lin et al., 2011; Pybus et al., 2015] were used. The first application of deep learning was achieved by [Sheehan and Song, 2016], and has been widely used ever since. Commonly used methods are convolutional neural networks (CNNs) [Chan et al., 2018; Flagel et al., 2019; Mo and Siepel, 2023; Sanchez et al., 2021; Smith and Kern, 2023; Torada et al., 2019], feedforward neural networks [Sanchez et al., 2022; Sheehan and Song, 2016], neural network autoencoders [Dominguez Mantes et al., 2023], and graph networks [Korfmann et al.]. Some of these methods already include strategies to adapt to phylogenetic data, such as the permutation invariance in the CNN described by [Chan et al., 2018]. This CNN takes into account the fact that when a genotype matrix is viewed as a black-and-white image, the rows can be shuffled without loss of information, unlike classical image data. However, theoretical knowledge from population genetics is often not exploited.

Key challenges in the widespread application of machine learning methods in population genetics, as elsewhere, include interpretability and robustness [Huang et al., 2024; Korfmann et al., 2023]. However, unlike many other application areas, population genetics has the unique advantage of offering theoretical concepts and models. This naturally raises the question of how to combine the strengths of both machine learning and population genetics: On the one hand, the ability to handle large data sets and the flexibility to recognize and exploit the underlying dynamics of the data; on the other hand, the integration of existing theoretical knowledge about evolutionary factors and well-established models to enhance not only interpretability but also robustness. Investigating this question forms the backbone of this thesis. To this end, we focus on two machine learning approaches applied to classical population genetics tasks: a supervised neural network for mutation rate estimation and an unsupervised hierarchical clustering method for population ancestry inference. The following section summarizes how the two publications, produced as part of this dissertation, contribute to more explainable and robust machine learning in population genetics by leveraging existing knowledge from the field.

1.3 Scope and Contributions

This thesis is based on two publications, both presenting novel machine learning approaches for population genetics that incorporate established population genetic concepts and methods.

> Klara Elisabeth Burger, Peter Pfaffelhuber, and Franz Baumdicker. "Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown." PLOS Computational Biology 18.8, 2022.

In the first publication, we introduce a concept for incorporating well-established model-based estimators into a loss function. We demonstrate this approach using a supervised dense feedforward neural network designed to estimate the scaled mutation rate from the SFS. Estimating the scaled mutation rate, or equivalently the effective population size, is a common task in population genetics. For low or high recombination, the optimal model-based estimation methods, in terms of minimum variance or mean squared error, are known and well understood. For intermediate recombination rates, the computation of optimal estimators is more involved. We investigate two neural network architectures: a simple linear network with no hidden layers, and an adaptive network with one hidden layer of 200 nodes.

The key innovation of the adaptive network is its ability to incorporate optimal model-based estimators into the training process through the loss function. The training data is thereby adaptively reweighted, which also helps when simulated training data does not sufficiently represent real-world data [Mo and Siepel, 2023]. Training is performed on simulated data over multiple iterations, with each iteration involving training of a neural network itself. Between iterations, the cost function is updated by increasing the weight of those parts of the training data where the performance of the neural network is lagging behind that of the best model-based estimator. The training process is considered complete when an iteration results in no further weight updates, meaning that the adaptive neural network performs as well as or better than the model-based estimators over the entire training domain. This adaptive approach can be thought of as a form of boosting. The neural networks are trained on both, data sets with fixed and variable recombination rates.

We apply the methods to simulated data for fixed recombination scenarios, as well as to the human chromosome 2 recombination map, providing a realistic setting in which local recombination rates may vary or remain unknown. Remarkably,

when trained with the adaptive loss function and variable recombination rates, only one hidden layer is required to obtain a single estimator that performs nearly as well as the model-based estimators for both low and high recombination rates, while offering a superior estimation method for intermediate recombination rates. Compared to more sophisticated machine learning methods, such as CNNs, the adaptive neural network remains the preferred choice for all recombination scenarios considered. Furthermore, the comparison with model-based estimators provides validation, as the neural network consistently mirrors the behavior of the established model-based estimator in every scenario where such an estimator exists. We show that the linear neural network reproduces the optimal estimators for low and high recombination rates, while the adaptive neural network weights the site frequency entries according to the optimal model-based estimator for each scenario. This holds even for the adaptive neural network trained with variable recombination rates, without requiring the recombination rate as an additional input. In addition, the adaptive neural network effectively helps when the simulated training data under-represents certain parameter ranges, which is especially observed at small mutation rates. Overall, this highlights the potential of using model-based estimators to refine training processes to improve robustness.

Klara Elisabeth Burger, Solveig Klepper, Ulrike von Luxburg, and Franz Baumdicker. "Inferring ancestry with the hierarchical soft clustering approach tangleGen." Accepted by Genome Research, 2024.

In the second publication, we present tangleGen, a hierarchical soft clustering tool that adapts the unsupervised machine learning framework Tangles [Klepper et al., 2023] to infer population ancestry by integrating key concepts from population genetics. Understanding the genetic ancestry of populations is central to many scientific and societal fields, including human evolutionary history, personalized medicine, forensic science, and genealogical research.

The presented method tangleGen is a hierarchical soft clustering method designed to infer ancestral relationships of populations from genomic data by incorporating fundamental population genetic concepts such as the fixation index or the Hardy-Weinberg equilibrium. The foundation of this approach is the use of bipartitions based on biallelic SNPs, which divide diploid individuals into two groups according to the presence or absence of at least one derived allele. tangleGen then aggregates information about the structure of the dataset from many weaker, imperfect bipartitions that provide only local insight, and combines them into an expressive clustering. To create a meaningful hierarchical structure, tangleGen prioritizes

bipartitions according to their discriminative power, which is achieved through a cost function based on the fixation index, a classic method for distinguishing populations. In addition, tangleGen accounts for the reliability of bipartitions by incorporating an estimate of the number of misclassified individuals according to an assumed Hardy-Weinberg equilibrium.

We demonstrate the capabilities and advantages of tangleGen for inference of ancestral relationships using both simulated data and data from the 1000 Genomes Project. tangleGen performs comparably to established methods in terms of clustering and ancestry inference, with a key advantage being its interpretability. The hierarchical perspective of tangleGen on the composition and structure of populations improves the interpretability of inferred ancestral relationships. In addition, tangleGen adds a new level of explainability by allowing the identification of the SNPs responsible for the clustering structure. Unlike many other methods, tangleGen is robust, providing deterministic and consistent results across different numbers of populations. Instead of requiring the number of independent populations as a pre-specified input, it identifies the number of related populations through its hierarchical clustering approach. With the ability to customize cuts, cost function, soft clustering and hyperparameters, tangleGen can be tailored to a wide range of research questions and is a versatile tool for investigating genetic diversity and ancestral relationships between populations.

Chapter 2

Publications

2.1 Neural Networks for Self-Adjusting Mutation Rate Estimation when the Recombination Rate is Unknown

Klara Elisabeth Burger, Peter Pfaffelhuber, and Franz Baumdicker.

Published in *PLOS Computational Biology*, 18.8, 2022 [Burger et al., 2022].

PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown

Klara Elisabeth Burger[1], **Peter Pfaffelhuber**[2], **Franz Baumdicker**[1,3]*

1 Cluster of Excellence "Machine Learning: New Perspectives for Science", University of Tübingen, Tübingen, Germany, **2** Department of Mathematical Stochastics, University of Freiburg, Freiburg, Germany, **3** Cluster of Excellence "Controlling Microbes to Fight Infections", Mathematical and Computational Population Genetics, University of Tübingen, Tübingen, Germany

* franz.baumdicker@uni-tuebingen.de

OPEN ACCESS

Citation: Burger KE, Pfaffelhuber P, Baumdicker F (2022) Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown. PLoS Comput Biol 18(8): e1010407. https://doi.org/10.1371/journal.pcbi.1010407

Editor: Luis Pedro Coelho, Fudan University, CHINA

Received: September 7, 2021

Accepted: July 18, 2022

Published: August 3, 2022

Peer Review History: PLOS recognizes the benefits of transparency in the peer review process; therefore, we enable the publication of all of the content of peer review and author responses alongside final, published articles. The editorial history of this article is available here: https://doi.org/10.1371/journal.pcbi.1010407

Data Availability Statement: All results in this manuscript are reproducible using code available at https://github.com/fbaumdicker/ML_in_pop_gen. This includes the simulation of training data, as well as the adaptive training procedure for the

Abstract

Estimating the mutation rate, or equivalently effective population size, is a common task in population genetics. If recombination is low or high, optimal linear estimation methods are known and well understood. For intermediate recombination rates, the calculation of optimal estimators is more challenging. As an alternative to model-based estimation, neural networks and other machine learning tools could help to develop good estimators in these involved scenarios. However, if no benchmark is available it is difficult to assess how well suited these tools are for different applications in population genetics.

Here we investigate feedforward neural networks for the estimation of the mutation rate based on the site frequency spectrum and compare their performance with model-based estimators. For this we use the model-based estimators introduced by Fu, Futschik et al., and Watterson that minimize the variance or mean squared error for no and free recombination. We find that neural networks reproduce these estimators if provided with the appropriate features and training sets. Remarkably, using the model-based estimators to adjust the weights of the training data, only one hidden layer is necessary to obtain a single estimator that performs almost as well as model-based estimators for low and high recombination rates, and at the same time provides a superior estimation method for intermediate recombination rates. We apply the method to simulated data based on the human chromosome 2 recombination map, highlighting its robustness in a realistic setting where local recombination rates vary and/or are unknown.

Author summary

- single-layer feedforward neural networks learn the established model-based linear estimators for high and low recombination rates

PLOS COMPUTATIONAL BIOLOGY Neural networks for mutation rate estimation when the recombination rate is unknown

neural network estimators. Furthermore, the optimal coefficients can be computed with a python script available in the same repository.

Funding: KB and FB are funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy EXC 2064/1 Project number 390727645, and EXC 2124 Project number 390838134. PP is supported in part by the Freiburg Center for Data Analysis and Modeling. We acknowledge support by Open Access Publishing Fund of University of Tübingen. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

- neural networks learn good estimators for intermediate recombination rates where computation of model-based optimal estimators is hardly possible
- a single neural network estimator can automatically adapt to a variable recombination rate and performs close to optimal
- this is advantageous when recombination rates vary along the chromosome according to a recombination map
- using the known estimators as a benchmark to adapt the training error function improves the estimates of the neural networks

Introduction

The development of machine learning methods for population genetics faces multiple specific challenges [1]. Nonetheless, it is meanwhile clear that machine learning is a promising technique to build more powerful inference tools. Especially for problems that are hard to tackle down with classical methods they might offer a new approach. In contrast, in population genetics, many theoretical results have been obtained within the last decades and enabled us to identify the best inference technique for specific scenarios. For example, the variance of estimators of the mutation rate, or equivalently the effective population size, is well understood, at least if the rate is constant and recombination is low or high.

Estimating the effective population size or mutation rate

Estimating the scaled mutation rate, usually denoted by $\theta = 4N_e\mu$, is a fundamental task in population genetics. If the per generation mutation rate μ is known, estimating the scaled mutation rate corresponds to estimating the effective population size N_e, which is often of primary interest and correlates with the genetic diversity in a given population [2]. More precisely, in populations with a small effective population size, genetic drift, i.e. frequency changes due to random sampling, is stronger. Knowledge about N_e allows thus to assess the relative importance of selection, mutation, migration, and other evolutionary forces compared to the influence of genetic drift. The effective population size is often significantly lower than the census population size and varies among populations [3], but also in time [4] and along the genome [5]. The same variation is also observed for the actual mutation rate μ [6–8]. Consequently, estimating θ, i.e. the mutation rate or N_e, is of great interest in many evolutionary fields including conservation genetics, breeding, and population demographics [9].

Many estimators of θ are developed within the coalescent framework without recombination, where mutations arise along a genealogy given by Kingman's coalescent [10]. However, if recombination is included into the framework [11], estimators of θ often require an estimate of the recombination rate [12, 13]. One of the most common estimators is Watterson's estimator [14]. It is an easy to compute, unbiased and asymptotically consistent estimator, which has a low variance for high recombination rates, when compared to alternative model-based, unbiased estimators. For Watterson's estimator the only necessary input is the total number of segregating sites in the sample. However, for low recombination rates Watterson's estimator, while still unbiased and consistent, usually has a high variance, when compared to alternative model-based, unbiased estimators. Multiple estimators have been developed based on the site frequency spectrum (SFS), which is the number of segregating sites that occur in k out of n individuals of the sample for $k = 1, \ldots, n - 1$. In particular, without recombination, and if θ is

known, the coefficients of an optimal unbiased linear estimator based on the SFS have been computed by Fu [15] and the linear estimator that minimizes the mean squared error (MSE) has been identified by Futschik et. al. [16]. Knowledge of the model-based estimators most suitable without and with high recombination rates enables us here to assess and improve the overall performance of artificial neural network estimators which have been trained either for exactly these scenarios or with unknown intermediate recombination rates.

Machine learning in population genetics

Artificial neural networks are popular in various scientific fields and often used in cases where theoretical models are very complex or hard to analyze [17–19]. Interestingly, machine learning approaches in population genetics, including support vector machines, neural networks, random forests, and approximate Bayesian computation (ABC) often use summary statistics of the genetic data borrowed from the theoretical literature as input data [1, 20], such as the site frequencies. However, there is an increasing number of studies and methods that do not rely on summary statistics. One example is a recent publication by Flagel et al. [21] who showed that effective population genetic inference can also be reached with deep learning structures like convolutional neural networks (CNNs) without precomputed summary statistics. Instead of a set of summary statistics they used the full genotype matrix as input for their CNN. The CNNs performed surprisingly good in different tasks from population genetics, including inferring historic population size changes and selection pressures along the genome. The genotype matrix has also been used as input data in other studies using deep learning in population genetics [22, 23]. As, in contrast to image data, the rows of the genotype matrix can be shuffled without loosing information, an adaptation of the network architecture or presorting the genotype matrix is often beneficial. One approach is suggested in [24] introducing an exchangeable neural network for population genetic data that makes the ordering of the samples invisible to the neural network. Besides summary statistics and the genotype matrix, tools based on inferred gene trees and ancestral recombination graphs are emerging [25]. Recently, Sanchez et al. [23] showed that combining deep learning structures using the genotype matrix and approximate Bayesian computation provides an effective method to reconstruct the effective population size through time for unknown recombination rates. Thus, artificial neural networks are a promising tool in more involved scenarios where theoretical insights are harder to obtain, although the often complex architectures impair the comprehension of the underlying learning process.

Here, we take a different perspective and consider the estimation of the mutation rate from single nucleotide frequencies. If data is generated within a coalescent framework, the optimal estimator (which is a linear map of the SFS) is known for no recombination when θ is known, and for high recombination rates. In particular, we consider a dense feedforward neural network with at most one hidden layer, which already suffices to achieve almost the performance of the optimal linear estimators and at the same time to provide a superior estimator for variable recombination rates.

Materials and methods

Model-based estimators of mutation rate and population size

In this section, we recall the properties of known estimators for θ, that are linear in the site frequency spectrum. More precisely, we consider mutations as modeled by a neutral Wright Fisher model with infinitely many sites model along Kingman's coalescent. The model-based estimators presented below were proposed and analyzed by Watterson [14], Fu [15] and

PLOS COMPUTATIONAL BIOLOGY Neural networks for mutation rate estimation when the recombination rate is unknown

Futschik et al. [16]. In order to give a self-contained presentation, we now recall some basics on the coalescent and give details of the above estimators:

To obtain Kingman's coalescent for a sample of size n we trace back the series of ancestors through time. Therefore, we define an ancestral tree for this sample by repeatedly coalescing each pair of two lineages at a rate 1, such that, when k is the number of remaining lineages, at rate $\binom{k}{2}$ two randomly chosen lineages coalesce. The resulting random tree is called Kingman's coalescent. We denote the random duration the coalescent spends with exactly k lineages by $T_k \sim \mathrm{Exp}\left(\binom{k}{2}\right)$. Neutral mutations are independently added upon this tree.

More precisely, for a given genealogical tree, mutations can happen everywhere along the branches at rate $\frac{\theta}{2}$, see Fig M in S1 Text. Given the length ℓ of a branch, the number of mutations on this branch is $\mathrm{Poi}\left(\frac{\theta}{2}\ell\right)$ distributed. Consequently there are $M \sim \mathrm{Poi}\left(\frac{\theta L}{2}\right)$ mutations along the tree, if $L = \sum_{k=2}^{n} kT_k$ is the total length of the tree. We consider the infinitely many sites model, where each mutation hits a new site. Thus the frequency of the derived allele in the sample population is given by the number of descendants of the branch where the mutation occurred. We define the site frequency spectrum as $S = (S_1, \ldots, S_{n-1})$, where S_i is the number of mutations that occur on a branch that is ancestral to exactly i out of n individuals in the sample.

In the following, we are looking for estimators of the form

$$\hat{\theta} = \sum_{i=1}^{n-1} a_i S_i.$$

which are uniquely defined by the vector $a = (a_1, \ldots, a_{n-1})$. We will see that optimal choices for a frequently depend on θ, which will lead to an iterative estimation procedure.

Unbiased linear estimators of the mutation rate θ. Watterson's estimator is given by setting $a_1 = \cdots = a_{n-1} = \mathbb{E}[L]^{-1}$, i.e.

$$\hat{\theta}_{\mathrm{W}} \coloneqq \frac{M}{h_n} = \sum_{i=1}^{n-1} \frac{S_i}{h_n} \quad \text{with} \quad h_n \coloneqq \sum_{i=1}^{n-1} \frac{1}{i}$$

being the n-th harmonic number and $S = (S_1, \ldots, S_{n-1})$ the site frequency spectrum.

From a theoretical point of view, only the cases of no recombination or in the limit of high recombination (leading to independence between loci) can easily be treated. Watterson's estimator is unbiased for θ for all recombination rates and if there is no recombination the variance, as found by Watterson [14], is given by

$$\mathbb{V}_\theta\left[\hat{\theta}_{\mathrm{W}}\right] = \theta\frac{1}{h_n} + \theta^2\frac{g_n}{h_n^2} \quad \text{where} \quad g_n \coloneqq \sum_{i=1}^{n-1} \frac{1}{i^2}.$$

In the limit of high recombination (unlinked loci) we get that $S_k \sim \mathrm{Poi}\left(\frac{\theta}{k}\right)$ and $(S_1, \ldots, S_{n-1})$ are independent, such that the variance reduces to

$$\mathbb{V}_\theta\left[\sum_{i=1}^{n-1} \frac{S_i}{h_n}\right] = \frac{\theta}{h_n}.$$

In this case, using standard theory on exponential families, $\sum_{i=1}^{n-1} S_i$ is a complete and sufficient statistic for θ, and it follows from the Lehmann–Scheffé theorem that Watterson's estimator is a unique Uniformly Minimum Variance Unbiased Estimator (Uniformly MVUE) [26]. However, this only holds for unlinked loci.

If no recombination is included the MVUE estimator was found by Fu [15]. In this case, the best linear unbiased estimator of θ from the site frequency spectrum $S = (S_1, \ldots S_{n-1})$ is given in matrix notation by

$$f_\theta(S) = \frac{\alpha^\top (D_\alpha + \theta\Sigma)^{-1}}{\alpha^\top (D_\alpha + \theta\Sigma)^{-1}\alpha} S,$$

whereby

$$\alpha = (\alpha_1, \ldots, \alpha_{n-1}) = \left(1, \frac{1}{2}, \ldots, \frac{1}{n-1}\right), \quad D_\alpha = diag(\alpha_1, \ldots, \alpha_{n-1}) \quad \text{and}$$
$$\Sigma = \{\sigma_{ij}\}, \qquad i, j = 1, \ldots, n-1,$$

symmetric with

$$\sigma_{ii} := \begin{cases} \beta_n(i+1) & \text{if } i < \frac{n}{2}, \\ 2\frac{h_n - h_i}{n-i} - \frac{1}{i^2} & \text{if } i = \frac{n}{2}, \\ \beta_n(i) - \frac{1}{i^2} & \text{if } i > \frac{n}{2}, \end{cases}$$

and for $i > j$

$$\sigma_{ij} := \begin{cases} \frac{\beta_n(i+1) - \beta_n(i)}{2} & \text{if } i + j < n, \\ \frac{h_n - h_i}{n-i} + \frac{h_n - h_j}{n-j} - \frac{\beta_n(i) + \beta_n(j+1)}{2} - \frac{1}{ij} & \text{if } i + j = n, \\ \frac{\beta_n(j) - \beta_n(j+1)}{2} - \frac{1}{ij} & \text{if } i + j > n, \end{cases}$$

where

$$\beta_n(i) := \frac{2n(h_{n+1} - h_i)}{(n-i+1)(n-i)} - \frac{2}{n-i}.$$

We note that the optimal coefficients $a(\theta) = (a_1(\theta), \ldots, a_{n-1}(\theta))$ depend on the real θ. In practice, therefore, the estimation of θ depends on an iterative procedure that approximates Fu's estimator as described below. The iterative version is neither linear nor unbiased, but close to the non-iterative version (Fig H in S1 Text).

General linear estimators of the mutation rate θ. So far we only considered unbiased estimators and minimized their variance. Allowing for a potential bias of the estimator can decrease the MSE when compared to the unbiased estimators. The SFS based estimator with minimal MSE in the absence of recombination was found by Futschik and Gach [16]:

The linear estimator of θ from the site frequency spectrum $S = (S_1, \ldots S_{n-1})$ with minimal MSE is given by

$$\tilde{f}_\theta(S) = \alpha^\top \left(\frac{D_\alpha}{\theta} + \Sigma + \alpha\alpha^\top\right)^{-1} S,$$

where α, D_α, and Σ are as above.

Note that Futschik and Gach introduced multiple variants for the estimation of θ. The estimator $\tilde{f}_\theta(S)$ used here corresponds to formula (26) in Futschik and Gach [16]. Another variant

from Futschik and Gach is a modification of Watterson's estimate that minimizes the MSE for estimates based on the number of segregating sites $\sum_{i=1}^{n-1} S_i$ in the scenario without recombination (Fig O in S1 Text). For positive recombination rates further variants depend on estimates of the recombination rate.

Iterative estimation of θ. The estimators of Fu and Futschik depend on the true but unknown θ. In practice, we thus have to build an iterative estimator, which will yield a θ-independent estimator and approximate the variance minimizing or MSE-minimizing estimators. Starting with some $\hat{\theta}_0$, e.g. $\hat{\theta}_0 = \hat{\theta}_W$, Wattersons estimator, we set

$$\hat{\theta}_{k+1}(S) \coloneqq f_{\hat{\theta}_k}(S)$$

which usually converges quickly. For example, for $n = 40$ and $\theta = 40$ it takes about 5 iterations until the estimate of θ is obtained up to 3 decimal places.

We call the resulting estimator $\hat{\theta}_{ItV}$ for Fu's estimator f_θ, and $\hat{\theta}_{ItMSE}$ for Futschik's estimator $\tilde{f}_\theta$. Note that due to the iteration the estimators are no longer explicitly linear or unbiased, but do not depend on θ.

Estimating the mutation rate with a dense feedforward neural network

We trained dense feedforward neural networks with zero or one hidden layer to perform the estimation task. Note that the architecture of the neural networks determine the type of functions the neural network can approximate. For example, if no hidden layer and no bias is included, the architecture ensures that the resulting estimator is a linear function in $S = (S_1, \ldots, S_{n-1})$.

Simulation of training data. In contrast to model-based estimators, the estimators first have to be trained in order to optimize parameters within the neural network. Therefore, we rely on simulations to train the neural network and then evaluate the resulting estimators. Five training data sets were generated with the software msprime by Kelleher et al. [27] for various parameters. For each training data set with $2 \cdot 10^5$ independent site frequency spectra, the haploid sample size is given by $n = 40$ and the recombination rate ρ was either set to 0 (no recombination), 20, 35 (moderate recombination), or 1000 (high recombination). Hereby, the recombination rate is scaled by N_e, such that $\rho = rL4N_e$, where r is the per generation per site recombination rate and L is the length of the simulated sequence. Within all training data sets the mutation rate θ is chosen uniformly in (0, 100). In addition, we combined the data sets with no recombination and high recombination with a data set with a uniformly chosen recombination rate in (0, 50). This creates the fifth training data set of total size $6 \cdot 10^5$ with a variable recombination rate.

Feedforward neural networks. In this project we consider two artificial neural networks: a linear dense feedforward neural network (no hidden layer) and a dense feedforward neural network with one hidden layer and an adaptive loss function to ensure a robust performance. All neural networks take S as input and are trained for θ chosen uniformly in (0, 100). ReLU was used as activation function and as optimizer the Adam algorithm [28] was chosen. The neural networks have been implemented in Python 3 via the library tensorflow and keras. Details of the training procedure and hyperparameter choice are given in section B in S1 Text. The code is made available on Github: fbaumdicker/ML_in_pop_gen.

Linear neural network. As a simple check whether the network is able to learn the mutation rate at all we considered a neural network with no hidden layer and no bias node included, i.e. a linear neural network. This simple network structure ensures that the estimator is linear in S, but not necessarily unbiased.

Neural network with one hidden layer. To investigate how more complexity in the architecture of the neural network improves the performance, we also implemented a neural network with one hidden layer and one bias node. See Fig L in S1 Text for visualization. Naturally, the question arises on how many hidden nodes to include in this additional layer. We observed that the optimal number of nodes depends on the sample size n. Performace for larger n improved when using more hidden nodes. In our experience, it is advisable to use at least $2n$ nodes. Using too few hidden nodes introduces an increased risk of losing robustness and using less than n nodes very often results in non-termination of the training process. Using significantly more hidden nodes produced similar results, but the runtime can increase significantly. For $n = 40$, using 200 nodes in the hidden layer has proven to be a good choice.

Adaptive reweighting of the loss function by model-based estimators. All neural networks have been trained on simulated data where θ is uniformly chosen in (0, 100). From the linear model-based estimators in absence of recombination we know that the coefficients of the optimal estimator depend on θ. Hence, within the neural networks, we included an adaptive reweighting of the training data with respect to the parameter θ. This ensures that the inferred estimator is not worse than the iterative estimators of Fu and Futschik nor Watterson's estimator for all possible values of θ. A visualization of the training procedure is shown in Algorithm 1. The training of the "adaptive" neural network is done in several iterations. Each iteration consists of training a neural network as before and subsequently increasing the weight of those parts of the training data where the normalized MSE (nMSE), i.e. the MSE divided by θ, is not close enough to the minimal nMSE of the iterative versions of Fu's and Futschik's, Watterson's, and the linear neural network estimator. Note, this evaluation in between the training steps requires a second validation data set, in addition to the one used to train the neural networks themselves.

For this comparison we divided the validation and training data into six subsets with respect to θ. The borders of the subsets are defined by $(t_i)_{i=0,\dots,6}$, where $t_0 = 0$, $t_1 = 1$ and $t_0 < t_1 < \cdots < t_6 = 100$. In the k-th subset we let $t_{k-1} < \theta \leq t_k$. The t_k are chosen such that the range of coefficients $a_j(\theta)$ in Fu's estimator is the same in the subsets, i.e.

$$\sum_{j=1}^{n-1} a_j(t_k) - a_j(t_{k-1}) \tag{1}$$

yields the same value for all $1 < k \leq 6$. Fig K in S1 Text illustrates the chosen interval borders. The total loss function is then given by

$$\frac{1}{m}\sum_{i=1}^{m}\left(\frac{\hat{\theta}_i - \theta_i}{\theta_i}\right)^2 \cdot \omega(\theta_i) \tag{2}$$

with $\omega(\theta_i) = \sum_{k=1}^{6} \omega_k \cdot 1_{\{t_{k-1} < \theta_i \leq t_k\}}$. In particular, the loss function can penalize errors in some subsets more than in others.

The weights ω_k are initialised by 1 and ω_k is updated in every iteration of comparable poor performance by setting

$$\omega_k = \omega_k + R \cdot \frac{\max\left(\frac{D_k}{b_k}, 0\right)}{\max_k\left(\max\left(\frac{D_k}{b_k}\right), 0\right)} \tag{3}$$

with

$$b_k \coloneqq \min(\mathrm{nMSE}_k(\hat{\theta}_{\mathrm{W}}), \mathrm{nMSE}_k(\hat{\theta}_{\mathrm{ItV}}), \mathrm{nMSE}_k(\hat{\theta}_{\mathrm{ItMSE}}), \mathrm{nMSE}_k(\hat{\theta}_{\mathrm{LinearNN}})),$$
$$D_k \coloneqq \mathrm{nMSE}_k(\hat{\theta}_{\mathrm{ANN}}) - b_k,$$

where $\mathrm{nMSE}_k(\hat{\theta})$ is the empirical normalized MSE of $\hat{\theta}$ on the subset where $t_{k-1} < \theta \leq t_k$ and R is drawn uniformly at random between 0.25 and 0.5.

The training is finished as soon as an iteration does not result in a weight update, i.e. the adaptive neural network performs comparable or better than the model-based estimators and the linear neural network on each of the six subsets or to be precise

$$\mathrm{nMSE}_k(\hat{\theta}_{\mathrm{ANN}}) \overset{!}{\leq} 1.02 \cdot b_k \tag{4}$$

for $k = 1, \ldots, 6$. In principle, if the network architecture does not provide enough capacity, i.e. if the number of hidden nodes is too low, the condition in (4) can lead to longer runtimes or prevent a termination. In our scenario, using 200 hidden nodes, this rarely occurred.

Algorithm 1: Overview on adaptive training procedure. This pseudocode illustrates the basic principle of the adaptive training procedure used for the adaptive neural networks.

input : n, classes, data
output: trained adaptive NN

obtain : six classes for adaptive training via Eq (1)
split : data into training and validation data
obtain : nMSE for $\hat{\theta}_{\mathrm{W}}$, $\hat{\theta}_{\mathrm{ItV}}$, $\hat{\theta}_{\mathrm{ItMSE}}$ & $\hat{\theta}_{\mathrm{LinearNN}}$ for each class on validation data
obtain : b containing smallest nMSE of $\hat{\theta}_{\mathrm{W}}$, $\hat{\theta}_{\mathrm{ItV}}$, $\hat{\theta}_{\mathrm{ItMSE}}$ & $\hat{\theta}_{\mathrm{LinearNN}}$ per class
init : loss weights $\omega(\theta) = 1$ and $\mathrm{nMSE}(\hat{\theta}_{\mathrm{NN}}) \gg b$

while not *nMSE*$(\hat{\theta}_{NN}) \leq 1.02 \cdot b$ *in each class (cond. (4))* **do**
  train : NN via adaptive loss function (2)
  obtain : $\mathrm{nMSE}(\hat{\theta}_{\mathrm{NN}})$
  update: update loss weights $\omega(\theta)$ for each class as shown in Eq (3)

Results

To investigate the capabilities of neural networks, two dense feedforward neural networks, with zero or one hidden layer, as a function of the site frequency spectrum were considered in this work. Those fairly simple network architectures were consciously chosen to facilitate a better understanding of the underlying learning process of the neural networks. Additionally, neural networks as a function of the site frequency spectrum can be easily compared to the model-based estimators, which use the same information. We observed multiple properties in our simulations:

If no recombination is included in the training and test data, as in Fig 1A, the neural nets perform comparably to Futschik and Gach's estimator, which has the lowest MSE among linear estimators in this scenario. If the recombination rate is high in the training and test data, as in Fig 1D the neural networks perform comparably to Watterson's estimator, which is a uniformly MVUE for θ in this situation. For training and test data sets with moderate recombination (Fig 1B and 1C), the neural networks outperform the other estimators. The bias of the adaptive neural network estimator is higher for smaller values of θ than for larger θ, but always smaller than the bias of the estimator of Futschik and Gach, see Fig J in S1 Text.

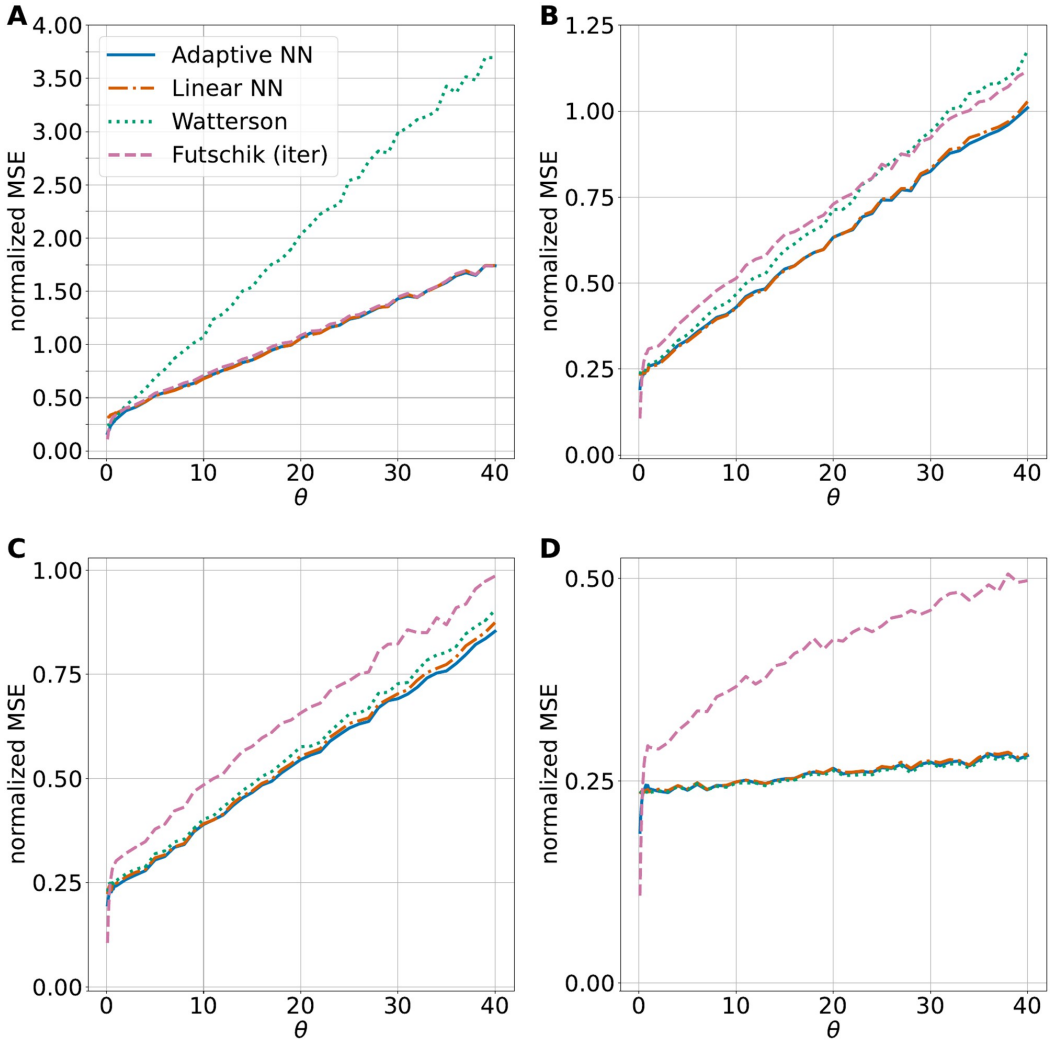


Fig 1. Performance of mutation rate estimators. The normalized MSE for four different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and two neural network estimators. A linear network without a hidden layer and an adaptive network with one hidden layer and 200 hidden nodes. For each subplot, we used msprime to generate a total of $4.9 \cdot 10^5$ independent simulations with sample size $n = 40$, recombination rate ρ, and 49 different mutation rates $\theta \in (0, 40]$. A: recombination rate $\rho = 0$, B: $\rho = 20$, C: $\rho = 35$, D: $\rho = 1000$. All shown neural networks have been trained on data sets with the corresponding recombination rate ρ.

https://doi.org/10.1371/journal.pcbi.1010407.g001

So far the considered neural network estimators depend on the recombination rate as the neural networks are trained on data sets with the same recombination rate as the test data. However, the recombination rate is often unknown or varies along the genome sequence such that a method that has a low variance regardless of the local recombination rate is desirable. To see if the neural networks can achieve this we trained them on data with variable

recombination rates. In this case, due to the restriction to linear estimators, the linear neural network is not able to match the performance of the adaptive neural network, especially for low and high recombination rates, but is surprisingly good for intermediate recombination rates. In contrast, the non-linear neural network with one hidden layer produced an estimator that had the lowest MSE for almost all possible values of the recombination rate (see Fig 2). This is for example beneficial when a sliding window approach is used to estimate the local mutation rate along a genome sequence. We illustrate this case applying the estimator to data based on the human recombination map. There the estimate depends on the local recombination rate, such that the neural network trained for variable recombination has the advantage to produce good estimates without an explicit estimate of the recombination map.

Naturally, the question of what has been learned by the neural network arises. To answer this question, one usually tries to draw conclusions from the learned weights, but this is often not trivial. Here, we used SHAP [29] to compute and analyze the feature importances of the linear and adaptive neural network, see section A in S1 Text. The importance of mutations in low frequency, i.e. S_i for small i, was larger than the importance of high frequency mutations. For the latter, however, the network has learned to take positive values of S_i with larger i differently into account depending on the recombination rates. A more detailed consideration of the SHAP values is given in section A in S1 Text.

Comparison with alternative ML methods

In this section, we compare the adaptive neural network to more sophisticated ML methods. For this purpose, we consider the convolutional neural network by Flagel et al. [21] and a standard Approximate Bayesian Computation (ABC) approach. The CNN introduced by Flagel et al. were applied to a number of evolutionary questions including identifying local introgression, estimating the recombination rate, detecting selective sweeps, and inferring population size changes. As a test case, Flagel et al. also used a CNN to infer θ based on the genotype matrix. We trained this CNN based on the same data sets as the other neural networks considered here. While the genotype matrix carries more information than the site frequency spectrum, e.g. about joint occurrence of mutations, it might be easier to learn some important features directly from summary statistics. ABC is a method based on Bayesian statistics to estimate the posterior distribution of model parameters. The basic idea of ABC is first to approximate the likelihood function using simulations and then to compare the outcome with the observed data. For our comparison we used the rejection method of the R package abc [30] based on the training data sets of the adaptive neural network, i.e. with $2 \cdot 10^5$ simulations to estimate θ.

Fig 3 compares the adaptive neural network to the CNN, the ABC method and Watterson's estimator in the same setting as before. Both, the CNN and ABC were trained for $\theta \in (0, 100)$, as the adaptive neural network, and especially struggle for small θ values. The CNN stabilizes for larger θ, whereas the ABC based estimator does not. Even though the CNN stabilizes for larger θ, we generally did not observe a stable performance, as Fig N in S1 Text shows. In contrast, the adaptive neural network is more robust due to the adaptive training procedure, a comparison is included in Fig N in S1 Text. In all four subplots of Fig 3 the adaptive neural network represents the preferred choice. This highlights the capabilities of the adaptive comparison with model-based estimators despite the architectural simplicity of the neural network.

Application

If a larger genome sequence is considered, the recombination rate varies between different regions of the chromosomes. We applied the trained neural networks and model-based

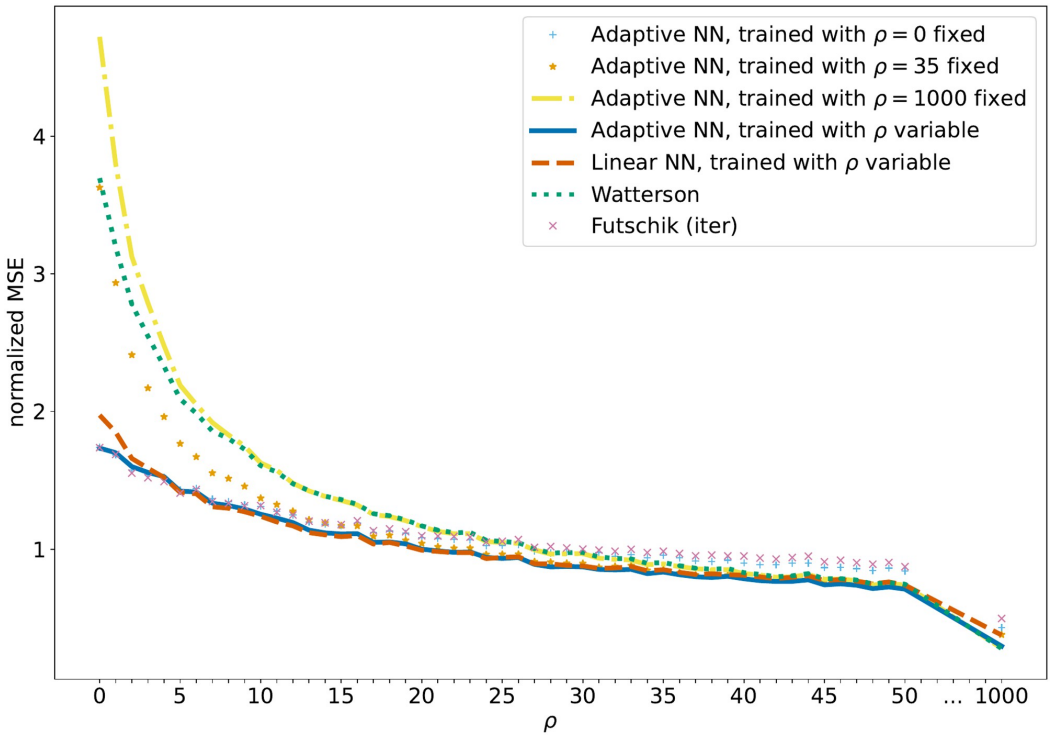


Fig 2. MSE for variable recombination rates. The normalized MSE of feedforward neural network estimators of the mutation rate compared to model-based estimators. In particular, the adaptive networks trained with a fixed recombination rate are compared to the networks trained with variable recombination rates. The recombination rate is given by $\rho = 0, \ldots, 50$ or $\rho = 1000$, while the sample size $n = 40$ and mutation rate $\theta = 40$ are fixed. MSE estimates are based on 10^5 independently simulated SFS for each value of ρ.

https://doi.org/10.1371/journal.pcbi.1010407.g002

estimators in this more realistic setting. For this purpose, data for human chromosome 2 (chr2) were generated using stdpopsim [31] based on the HapMap human recombination map [32], see Fig 4. The estimators were applied in a sliding window approach along the chromosome. Using a constant per generation mutation rate of $1.29 \cdot 10^8$ per base pair [33] and a window size of 70kb this resulted in a scaled mutation rate of $\theta = 36.12$ and a scaled local recombination rate between 0 and 1622 illustrated in Fig 4. The estimates of θ along the chromosome are shown in Fig 5. Close to chromosome position $1.0 \cdot 10^8$ the impact of the dependency on a single tree due to no recombination is observable. Watterson's estimator often resulted in a larger deflection from the true $\theta = 36.12$ than the other estimators. This is in particular true for regions with low or no recombination. If multiple windows fall within one such region the corresponding estimates are strongly correlated. For the region of low recombination near chromosome position $1.0 \cdot 10^8$ Fig P and Q in S1 Text show a zoomed-in version of the recombination map in Fig 4 and the estimates of θ in Fig 5. To better illustrate this effect, the estimators have been applied to data generated by an artificial recombination map with multiple regions without recombination separated by regions with recombination, see section C in S1 Text.

A separate consideration of disjoint windows with low ($0 < \rho \leq 1$), intermediate ($30 < \rho \leq 40$), or high ($\rho > 150$) recombination according the the human recombination map is shown

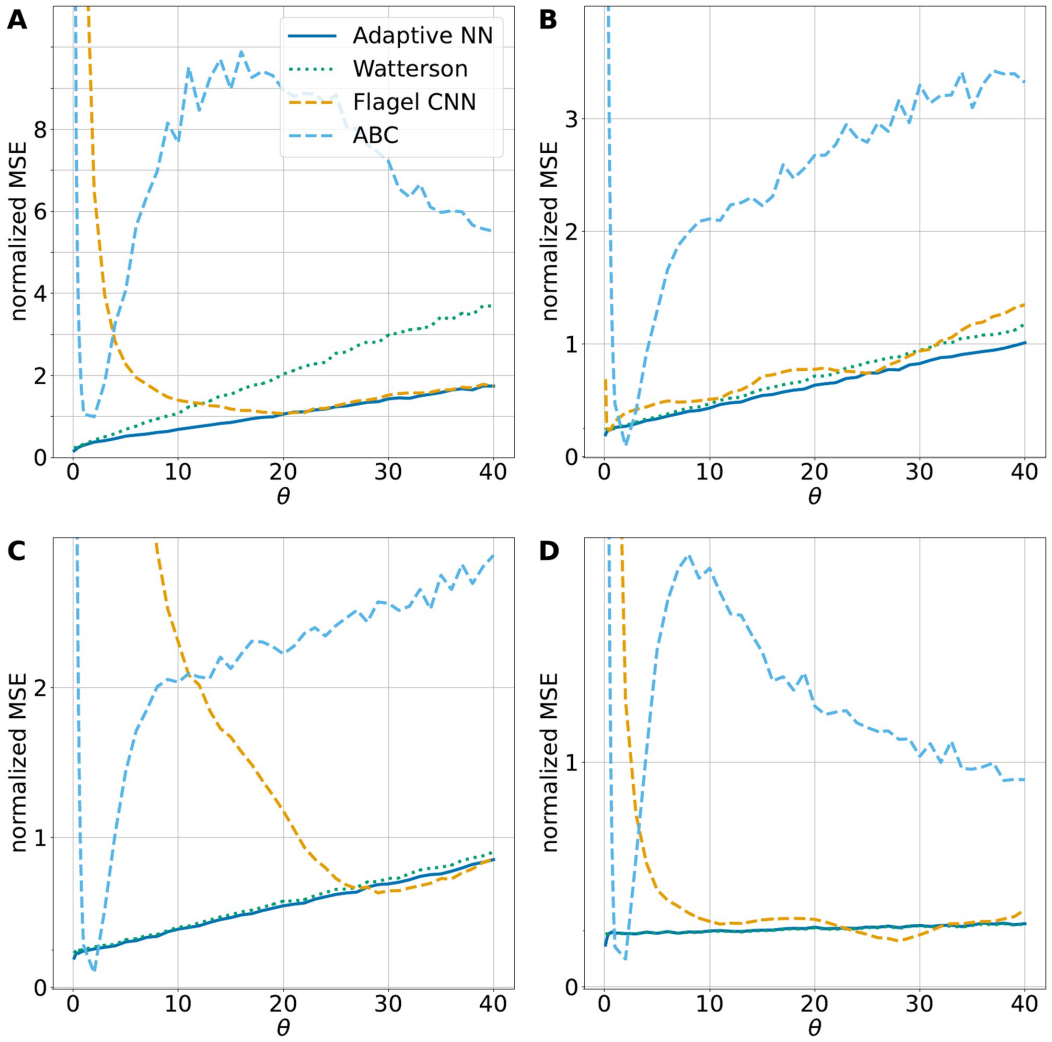


Fig 3. Comparison to other ML methods. The normalized MSE for four different estimators are shown: Watterson's estimator and the adaptive neural network (as in Fig 1), a retrained CNN by Flagel et al. [21] and Approximate Bayesian Computation (ABC). The same simulations as in Fig 1 have been used. A: recombination rate $\rho = 0$, B: $\rho = 20$, C: $\rho = 35$, D: $\rho = 1000$. All shown neural networks and ABC have been trained on data sets with the corresponding recombination rate ρ.

https://doi.org/10.1371/journal.pcbi.1010407.g003

in Fig 6. The observations in the constant recombination setting are also present in this more realistic scenario. Watterson's estimator more often strongly overestimates theta for low and medium recombination rates while Futschik's estimator performs better for low than high recombination rates. The neural networks trained with variable ρ is also in this more realistic scenario based on a recombination map able to adapt to the local recombination rate.

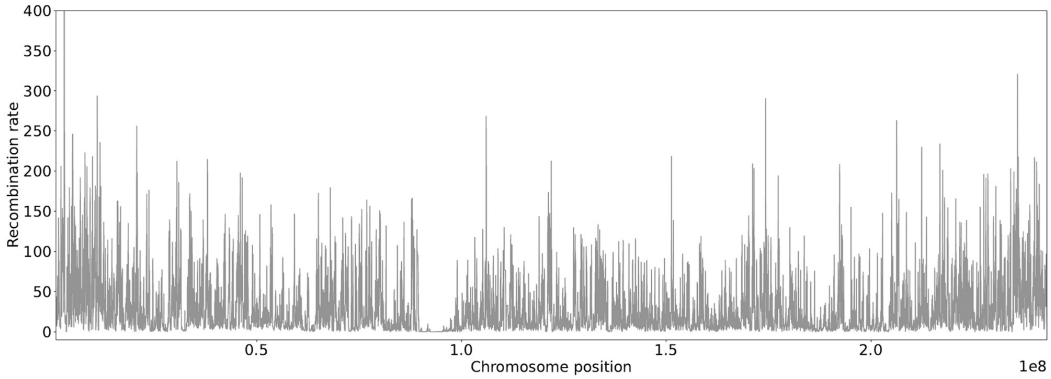


Fig 4. Recombination map for human chr2. Displayed recombination rates along the chromosome are based on a sliding window of 70kb. This also ensures better comparability with Fig 5. The mean recombination rate in the 70kb windows varies along the genome and is on average approximately $\rho = 31$.

https://doi.org/10.1371/journal.pcbi.1010407.g004

Discussion and conclusion

Different optimal estimators for the mutation rate or population size are known in scenarios of low and high recombination. Fu's and Watterson's estimator are linear unbiased estimators with minimal variance in the absence of recombination and in the limit of large recombination rates, respectively. However, simulations show that they do not adapt well to other frequencies of recombination. In the case of unknown recombination rates, it is therefore necessary for model-based estimators to estimate the recombination rate in a separate step and numerically compute a suitable estimator [12].

We were able to circumvent this step and created a simple neural network that has a low MSE regardless of the recombination rate. It is worth noting that the true recombination rate

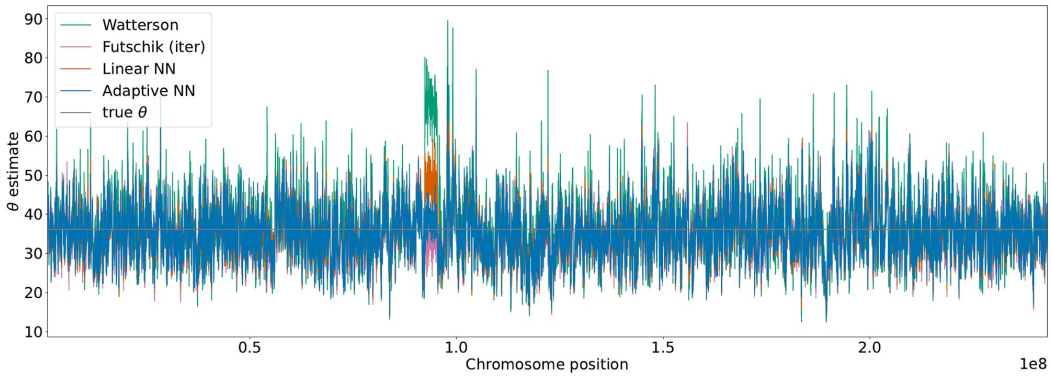


Fig 5. Estimation of θ for human chr2. The SFS for θ estimation was calculated based on a sliding window of size 70kb. Estimates based on Watterson, Futschik, the linear neural network and the adaptive neural network are displayed. The neural networks have been trained with variable recombination rates. The underlying true mutation rate $\theta = 36.12$ is shown as a grey line.

https://doi.org/10.1371/journal.pcbi.1010407.g005

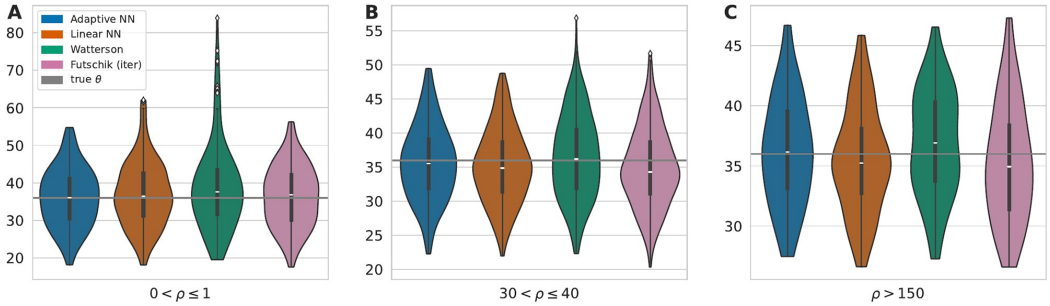


Fig 6. Estimates of θ in regions of chr2 with low, intermediate, and high recombination. The mean recombination rates of disjoint 70kb windows of chr2 were calculated and estimates for regions with low ($0 < \rho \leq 1$) (A), medium ($30 < \rho \leq 40$) (B) and large ($\rho > 150$) (C) recombination rates are shown in a Violin plot. The white marker in the box of the violins visualizes the median. The true mutation rate $\theta = 36.12$ is shown as a grey line.

https://doi.org/10.1371/journal.pcbi.1010407.g006

was not used as an additional input here. Instead, the neural network adapts to the unknown recombination rate solely based on the observed SFS.

In contrast to the recombination rate, variability in the mutation rate is no issue in practice, although some of the considered estimators depend on the true but unknown mutation rate θ. Iterative procedures for Fu's and Futschik's estimator are usually converging after 3–5 iterations and show almost the same performance as the optimal estimators (Fig H in S1 Text). In general, when training a neural network, the parameter range must be chosen carefully as they can learn the simulation range and consequently overestimate or underestimate outside this range. In our setting, this is mainly a problem if θ is smaller in the application than in the training data set since small θ values demand special attention to be learned. The linear and adaptive neural networks do not show dramatic overestimation or underestimation outside the training range, as Fig I in S1 Text demonstrates. In an application, if there is no idea about the realistic range of θ, we would recommend first using the Watterson's estimator to get an approximate idea about the range of θ and retrain the neural network if necessary.

We observe that the neural networks approximate model-based estimators in situations where the model-based estimators are close to the optimal estimator (no or high recombination) and outperform them in scenarios that are hard to treat theoretically (moderate recombination). This remains true when applied in a more realistic setting with variable recombination along the human chr2. Even compared to more sophisticated ML methods, i.e. Flagel's CNN and ABC, the adaptive neural network remains the preferred choice for all recombination scenarios considered. In particular, the advantages of an adaptive method or training procedure become apparent, since otherwise, for example, small θ values are not given enough attention. This again highlights the capabilities of the adaptive neural network despite its architectural simplicity and the potential of using model-based results to adjust the training process. There is a multitude of other model-based estimators for the mutation rate, population size and related quantities, e.g. for the expected heterozygosity [34], that could enable machine learning methods in population genetics to adjust their performance.

Clearly, neural networks are computationally more demanding than model-based approaches. For example, it is necessary to train the network separately for each sample size. However, training is fast, even for the adaptive neural network. For $n = 40$ and training data with $2 \cdot 10^5$ simulated SFS, training is completed after approximately 20–50 iterations, which takes about 10 minutes on a laptop with an AMD Ryzen 7 octacore and 32 GB RAM. If these pitfalls are considered, even minimal artificial neural networks are attractive alternatives to

model-based estimators of the scaled mutation rate since they can perform uniformly well for different recombination scenarios.

It is worth noting that the observed outperformance of the neural networks was obtained solely based on the site frequency spectrum, but genetic data usually contains more information about the distribution of mutations among samples and their location along the genome. Machine learning methods based on the raw genetic data will require more complex network architectures and could help to automatically extract this additional information. Recent advances in the development of machine learning methods for population genetics [22, 23, 35–37], indicate that expanding the toolbox of neural network-based inference tools to more complex estimation tasks and larger data sets is a promising approach.

Supporting information

S1 Text. Supporting Information regarding the feature importance of neural networks, neural networks training procedure, application with a block recombination map, and further supporting figures.
(PDF)

Acknowledgments

We thank Philipp Harms for helpful discussions, Axel Fehrenbach for thoughtful input, and Libera Lo Presti for careful proof-reading of the manuscript.

Author Contributions

Conceptualization: Klara Elisabeth Burger, Peter Pfaffelhuber, Franz Baumdicker.

Formal analysis: Klara Elisabeth Burger.

Funding acquisition: Franz Baumdicker.

Methodology: Klara Elisabeth Burger, Peter Pfaffelhuber, Franz Baumdicker.

Software: Klara Elisabeth Burger.

Supervision: Franz Baumdicker.

Writing – original draft: Klara Elisabeth Burger.

Writing – review & editing: Klara Elisabeth Burger, Peter Pfaffelhuber, Franz Baumdicker.

References

1. Schrider DR, Kern AD. Supervised Machine Learning for Population Genetics: A New Paradigm. Trends in Genetics. 2018; 34(4):301–312. https://doi.org/10.1016/j.tig.2017.12.005 PMID: 29331490
2. Charlesworth B. Fundamental concepts in genetics: Effective population size and patterns of molecular evolution and variation. Nature Reviews Genetics. 2009; 10(3):195–205. https://doi.org/10.1038/nrg2526 PMID: 19204717
3. Frankham R. Effective population size/adult population size ratios in wildlife: A review. Genetics Research. 2008; 89(5-6):491–503. https://doi.org/10.1017/S0016672308009695
4. Dutheil JY, Hobolth A. Ancestral population genomics. vol. 1910; 2019.
5. Gossmann TI, Woolfit M, Eyre-Walker A. Quantifying the variation in the effective population size within a genome. Genetics. 2011; 189(4):1389–1402. https://doi.org/10.1534/genetics.111.132654 PMID: 21954163
6. Hodgkinson A, Eyre-Walker A. Variation in the mutation rate across mammalian genomes. Nature Reviews Genetics. 2011; 12(11):756–766. https://doi.org/10.1038/nrg3098 PMID: 21969038

7. Mathieson I, Reich D. Differences in the rare variant spectrum among human populations. PLoS Genetics. 2017; 13(2):1–17. https://doi.org/10.1371/journal.pgen.1006581 PMID: 28146552
8. Harris K, Pritchard JK. Rapid evolution of the human mutation spectrum. eLife. 2017; 6:1–17. https://doi.org/10.7554/eLife.24284 PMID: 28440220
9. Wang J, Santiago E, Caballero A. Prediction and estimation of effective population size. Heredity. 2016; 117(4):193–206. https://doi.org/10.1038/hdy.2016.43 PMID: 27353047
10. Kingman JFC. On the genealogy of large populations. Journal of Applied Probability. 1982; 19A:27–43. https://doi.org/10.1017/S0021900200034446
11. Hudson RR. Properties of a neutral allele model with intragenic recombination. Theoretical Population Biology. 1983; 23:183–201. https://doi.org/10.1016/0040-5809(83)90013-8 PMID: 6612631
12. Fu YX. Estimating Effective Population Size or Mutation Rate Using the Frequencies of Mutations of Various Classes in a Sample of DNA Sequences. Genetics. 1994; 138:1375–1386. https://doi.org/10.1093/genetics/138.4.1375 PMID: 7896116
13. Hey J, Wakeley J. A coalescent estimator of the population recombination rate. Genetics. 1997; 145 (3):833–846. https://doi.org/10.1093/genetics/145.3.833 PMID: 9055092
14. Watterson GA. On the number of segregating sites in genetical models without recombination. Theoretical Population Biology. 1975; 7(2):256–276. https://doi.org/10.1016/0040-5809(75)90020-9 PMID: 1145509
15. Fu YX. A Phylogenetic Estimator of Effective Population Size or Mutation Rate. Genetics. 1994; 136:685–692. https://doi.org/10.1093/genetics/136.2.685 PMID: 8150291
16. Futschik A, Gach F. On the inadmissibility of Watterson's estimator. Theoretical Population Biology. 2008; 73(2):212–221. https://doi.org/10.1016/j.tpb.2007.11.009 PMID: 18215409
17. Paliwal M, Kumar UA. Neural networks and statistical techniques: A review of applications; 2009.
18. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, Van Den Driessche G, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016; 529(7587):484–489. https://doi.org/10.1038/nature16961 PMID: 26819042
19. Lu L, Jin P, Karniadakis GE. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators;.
20. Sheehan S, Song YS, Buzbas E, Petrov D, Boyko A, Auton A. Deep Learning for Population Genetic Inference. PLOS Computational Biology. 2016; 12(3):e1004845. https://doi.org/10.1371/journal.pcbi.1004845 PMID: 27018908
21. Flagel L, Brandvain Y, Schrider DR. The unreasonable effectiveness of convolutional neural networks in population genetic inference. Molecular Biology and Evolution. 2019; 36(2):220–238. https://doi.org/10.1093/molbev/msy224 PMID: 30517664
22. Torada L, Lorenzon L, Beddis A, Isildak U, Pattini L, Mathieson S, et al. ImaGene: A convolutional neural network to quantify natural selection from genomic data. BMC Bioinformatics. 2019; 20(Suppl 9):1–12. https://doi.org/10.1186/s12859-019-2927-x PMID: 31757205
23. Sanchez T, Cury J, Charpiat G, Jay F. Deep learning for population size history inference: Design, comparison and combination with approximate Bayesian computation. Molecular Ecology Resources. 2020; p. 1–16. PMID: 32644216
24. Chan J, Spence JP, Mathieson S, Perrone V, Jenkins PA, Song YS. A likelihood-free inference framework for population genetic data using exchangeable neural networks. Advances in Neural Information Processing Systems. 2018;(NeurIPS 2018):8594–8605. PMID: 33244210
25. Hejase HA, Dukler N, Siepel A. From Summary Statistics to Gene Trees: Methods for Inferring Positive Selection. Trends in Genetics. 2019; p. 1–16.
26. Shao J. Mathematical Statistics. 2nd ed. Springer-Verlag New York Inc; 2003.
27. Kelleher J, Etheridge AM, McVean G. Efficient Coalescent Simulation and Genealogical Analysis for Large Sample Sizes. PLoS Comput Biol. 2016; 12(5):1–22. https://doi.org/10.1371/journal.pcbi.1004842 PMID: 27145223
28. Kingma D, Ba J. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations. 2014;.
29. Lundberg SM, Lee SI. A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems 30; 2017. p. 4765–4774.
30. Csillery K, Francois O, Blum MGB. abc: an R package for approximate Bayesian computation (ABC). Methods in Ecology and Evolution. 2012;.
31. Adrion JR, Cole CB, Dukler N, Galloway JG, Gladstein AL, Gower G, et al. A community-maintained standard library of population genetic models. bioRxiv. 2019.

32. Frazer KA, Ballinger DG, Cox DR, Hinds DA, Stuve LL, Gibbs RA, et al. A second generation human haplotype map of over 3.1 million SNPs. Nature. 2007; 449(7164):851–861. https://doi.org/10.1038/nature06258 PMID: 17943122

33. Tian X, Browning BL, Browning SR. Estimating the Genome-wide Mutation Rate with Three-Way Identity by Descent. American Journal of Human Genetics. 2019; 105(5):883–893. https://doi.org/10.1016/j.ajhg.2019.09.012 PMID: 31587867

34. DeGiorgio M, Rosenberg NA. An unbiased estimator of gene diversity in samples containing related individuals. Molecular Biology and Evolution. 2009; 26(3):501–512. https://doi.org/10.1093/molbev/msn254 PMID: 18988687

35. Adrion JR, Galloway JG, Kern AD. Predicting the landscape of recombination using deep learning. Molecular Biology and Evolution. 2020; 37(6):1790–1808. https://doi.org/10.1093/molbev/msaa038 PMID: 32077950

36. Suvorov A, Hochuli J, Schrider DR. Accurate Inference of Tree Topologies from Multiple Sequence Alignments Using Deep Learning. Systematic Biology. 2020; 69(2):221–233. https://doi.org/10.1093/sysbio/syz060 PMID: 31504938

37. Hejase HA, Mo Z, Campagna L, Siepel A. SIA: Selection Inference Using the Ancestral Recombination Graph. bioRxiv. 2021; p. 2021.06.22.449427.

Supporting Information for

Neural Networks for self-adjusting Mutation Rate Estimation when the Recombination Rate is unknown

A Feature Importance of Neural Networks

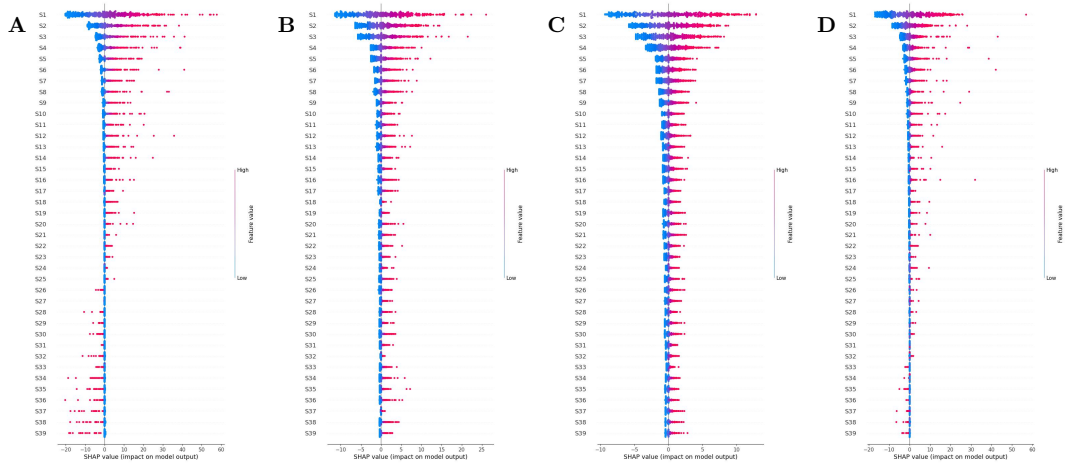


Fig A. Feature importance linear neural network via SHAP. Feature importance of the linear neural network for $\rho = 0$ (A), $\rho = 35$ (B), $\rho = 1000$ (C) and ρ variable (D). Here, the calculated shap values for $\rho = 0$ refer to the linear neural network trained for $\rho = 0$ and so on. The shap values were calculated for 500 SFS. Each dot represents the shap value for a particular prediction of one feature, i.e. one SFS entry of one of the 40 SFS. The color indicates whether the particular SFS entry had a high or low value, and the position on the x-axis shows its significance, i.e., the extent to which it increased or decreased the θ estimate.

Artificial neural networks tend to have complex architectures, which can impair the comprehension of the underlying learning process. However, it is often of interest to understand what the neural network has learned. Depending on the ML method, there are different ways to obtain knowledge about it. In our case, we determined the feature importance, i.e. a score for each feature representing its effect on the model. One tool to do so is SHAP (SHapley Additive exPlanations), a game theoretic approach to explain the output of machine learning models [1]. Hereby, the shap values describe the responsibility of an SFS entry for a change in the θ estimate, the higher the absolute shap value, the greater the impact.

In Fig A the feature importance of the linear neural network is displayed. In general the first entry of the SFS, S_1, is the one with the highest impact on the estimation as its shap value is the most pronounced. For $\rho = 0$ the importance of $S_2, S_3, \ldots$ more and more decreases and increases again for the last SFS entries. For $\rho = 35$ and $\rho = 1000$, the feature importance also decreases initially, but is quasi-constant for middle and posterior S_i. For variable

recombination rates (D), we observe a mix of the feature importance for $\rho = 0$ and $\rho = 35, 1000$. The pattern from A can still be seen, i.e. that the impact of the last SFS entries increases again and high feature values lead to a reduction of the θ estimation, but to a lesser extent. Subplots A-D in Fig A have in common that they show a higher importance for the first SFS entries than to the others. This is not surprising as it means that the estimator puts more weight on the mutation events happening in the more recent past, just as Fu's estimator does. Considering the weights of the linear neural network, which can be compared to the coefficients of the model-based estimators by nature, we can also extract information about the sign of the coefficient by Fig A. For the linear neural network the weights corresponding to $S_{26}, \ldots, S_{39}$ are negative for $\rho = 0$. This can also be observed in Fig A A as higher values for SFS entries lead to an decreased θ estimation. Those negative weights are not surprising as half of the coefficients of the optimal model-based estimator, Fu's estimator, are negative for large enough θ, see Fig K. For $\rho = 1000$ the linear neural network has no negative weights, which can also be observed in Fig A. This is also reasonable since Watterson's estimator has constant positive coefficients and is the optimal estimator for high recombination rates.

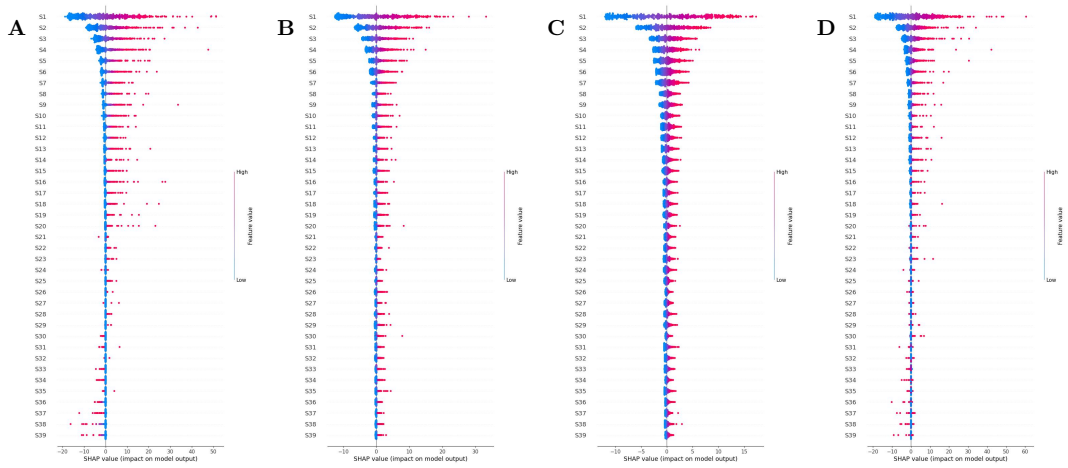


Fig B. Feature importance adaptive neural network via SHAP. Feature importance of the adaptive neural network for $\rho = 0$ (A), $\rho = 35$ (B), $\rho = 1000$ (C) and ρ variable (D). Here, the calculated shap values for $\rho = 0$ refer to the adaptive neural network trained for $\rho = 0$ and so on. The shap values were calculated for 500 SFS. Each dot represents the shap value for a particular prediction of one feature, i.e. one SFS entry of one of the 40 SFS. The color indicates whether the particular SFS entry had a high or low value, and the position on the x-axis shows its significance, i.e., the extent to which it increased or decreased the θ estimate.

Fig B displays the feature importance for the adaptive neural network. Compared to Fig A, no too big differences are noticeable. This shows that the adaptive neural network distributes the importance between the SFS entries more or less the same as the linear neural network, at least when trained for fixed recombination rates.

Fig C shows the feature importance for the adaptive neural network trained for variable recombination rates. Shap values were calculated as before for $\rho = 0, \rho = 35, \rho = 1000$, and ρ variable. By doing so, we can observe, how the adaptive network adapts to the different recombination scenarios. Again, the first entries of the SFS have the most importance, especially S_1, since their shap values are the most pronounced. In addition, the adaptive neural network adapts to different recombination scenarios, as we can observe, for example, from a different pattern for S_{31} to S_{39} between subplot A and subplot B/C in Fig C. For these features, a high value for $\rho = 0$ results in a decreased θ estimate, but for $\rho = 35$ or $\rho = 1000$ in an increased estimate.

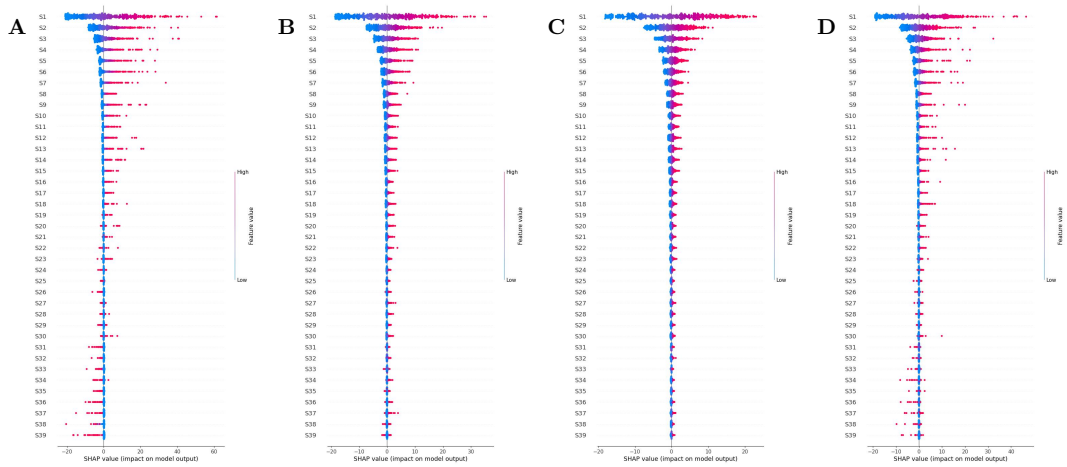


Fig C. Feature importance adaptive neural network, trained with ρ variable, via SHAP. Feature importance of the adaptive neural network for $\rho = 0$ (A), $\rho = 35$ (B), $\rho = 1000$ (C) and ρ variable (D). Here, the calculated shap values for A, B, C, and D are based on the adaptive neural network trained for variable recombination rates. The shap values were calculated for 500 SFS. Each dot represents the shap value for a particular prediction of one feature, i.e. one SFS entry of one of the 500 SFS. The color indicates whether the particular SFS entry had a high or low value, and the position on the x-axis shows its significance, i.e., the extent to which it increased or decreased the θ estimate.

B Neural Network Training Procedure

We considered two artificial neural networks: a linear dense feedforward neural network (no hidden layer) and a dense feedforward neural network with one hidden layer and an adaptive loss function. All neural networks take the site frequency spectrum S as input and are trained for θ chosen uniformly in $(0, 100)$. The neural networks for fixed recombination rates have been trained based on $2 \cdot 10^5$ simulations, for variable recombination rates with $6 \cdot 10^5$ simulations. For the linear neural network the division of simulations between training and validation was chosen to be 80% and 20%. Since the adaptive training procedure is similar to an ensemble training, in the sense that the process involves training multiple neural networks, a second validation set is required to evaluate in between the training steps. Therefore, an additional 20% is subtracted from the training set as a second validation set and the larger validation set is used to decide if another update of the adaptive loss function is needed. The test set is simulated separately and usually of size $4.9 \cdot 10^5$.

Hyperparameters are selected via a grid search. ReLU is used as activation function, the adam optimizer [2] with learning rate 0.001, the number of hidden nodes is chosen as 200 and the batch size as 64.

Preventing overfitting is always a challenging problem in ML. The risk of overfitting can be reduced not only in the training process by using methods such as regularization methods, but also via architectural design choices. First of all, the adaptive neural network has a lower model capacity compared to more sophisticated ML methods, making an overfit less likely. In addition, the adaptive training process is similar to ensemble learning with a termination condition (Eq 4Adaptive Reweighing of the Loss Function by Model-Based Estimatorsequation.0.4) preventing the model from fitting the training data too well. Furthermore, each of the neural networks in the adaptive training procedure was trained using the regularization method *Early Stopping*. Training, validation and test errors were monitored. We observed no tendency of the model to overfit. For the generalization outside the training range, see also Figure I.

C Application with a Block Recombination Map

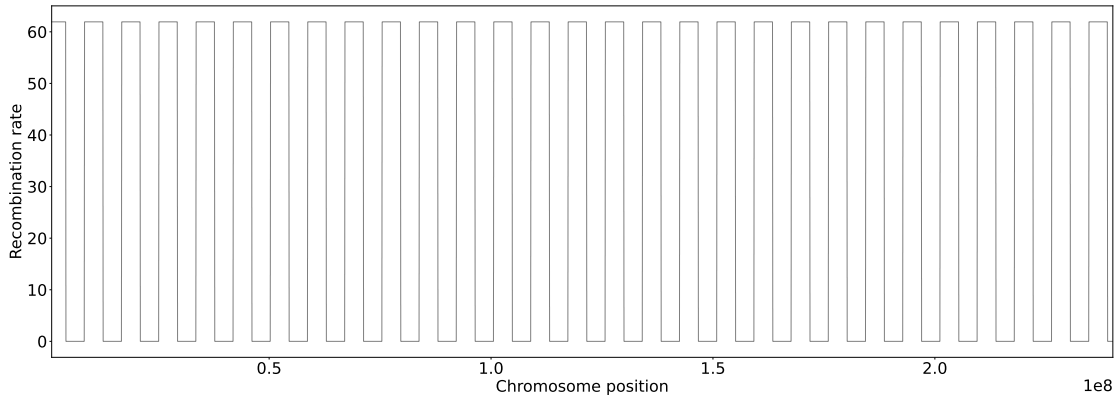


Fig D. Block recombination map. Recombination map with only two different recombination rates. The mean recombination rate matches that of chr2. Displayed recombination rates along the chromosome are based on a sliding window of 70kb. This also ensures comparability with Fig E.

To better illustrate the variability of the considered estimators along a chromosome we created an artificial recombination map with separated regions with low recombination. In this map the recombination rate jumps between 0 and 61.9 such that the mean recombination rate matches that of human chr2. This results in a *block recombination map* as displayed in Fig D.

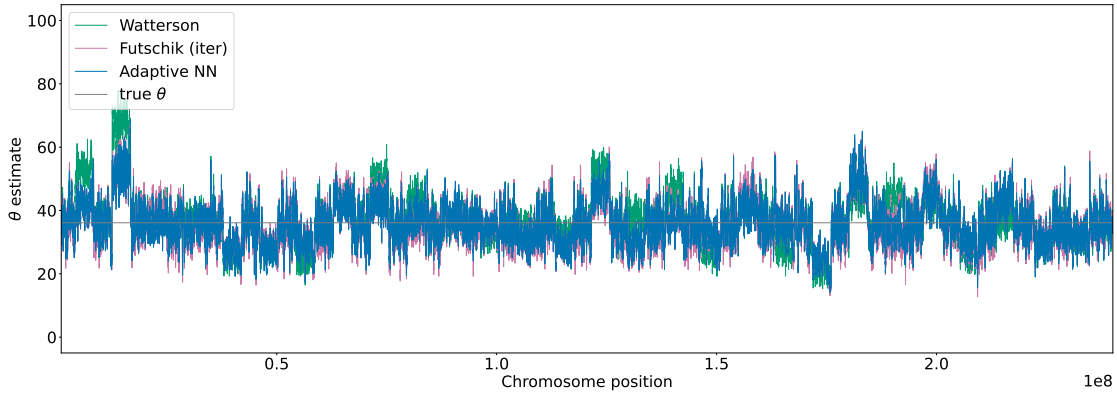


Fig E. Estimation of θ for block recombination map. SFS for θ estimation was calculated based on a sliding window of size 70kb. Estimates for Watterson, iterative Futschik and the adaptive neural network are displayed. The neural network is trained with variable recombination rates. The underlying $\theta = 36.12$ is shown as the grey line.

Fig E displays the estimates of θ for Watterson's estimator, the iterated minimal MSE estimator (Futschik (iter)) and the adaptive neural network for the block recombination map (Fig D). An extract not displaying the whole chromosome is shown in Fig F and for completeness the corresponding recombination map section in Fig G. Figures E and F illustrate how the estimators adapt to the two recombination levels. Obviously, Watterson's estimator struggles for the parts without recombination. The iterative Futschik estimator is not particularly noticeable, in part because for

high recombination rates the errors are generally much lower than for $\rho = 0$ and thus less detectable on the scale (see e.g. Fig 1**Performance of mutation rate estimators.** The normalized MSE for four different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and two neural network estimators. A linear network without a hidden layer and an adaptive network with one hidden layer and 200 hidden nodes. For each subplot, we used msprime to generate a total of $4.9 \cdot 10^5$ independent simulations with sample size $n = 40$, recombination rate ρ, and 49 different mutation rates $\theta \in (0, 40]$. A: recombination rate $\rho = 0$, B: $\rho = 20$, C: $\rho = 35$, D: $\rho = 1000$. All shown neural networks have been trained on data sets with the corresponding recombination rate ρ. figure.caption.19). Of the estimators considered, the adaptive neural network uniform performs best.

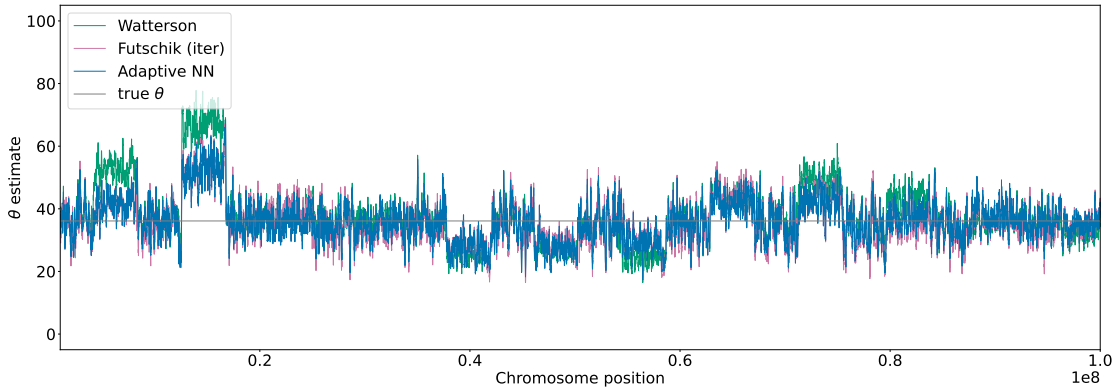


Fig F. Extract of θ estimation for block recombination map. SFS for θ estimation was calculated based on a sliding window of size 70kb. Estimates for Watterson, iterative Futschik and the adaptive neural network are displayed. The neural network is trained with variable recombination rates. The underlying $\theta = 36.12$ is shown as the grey line.

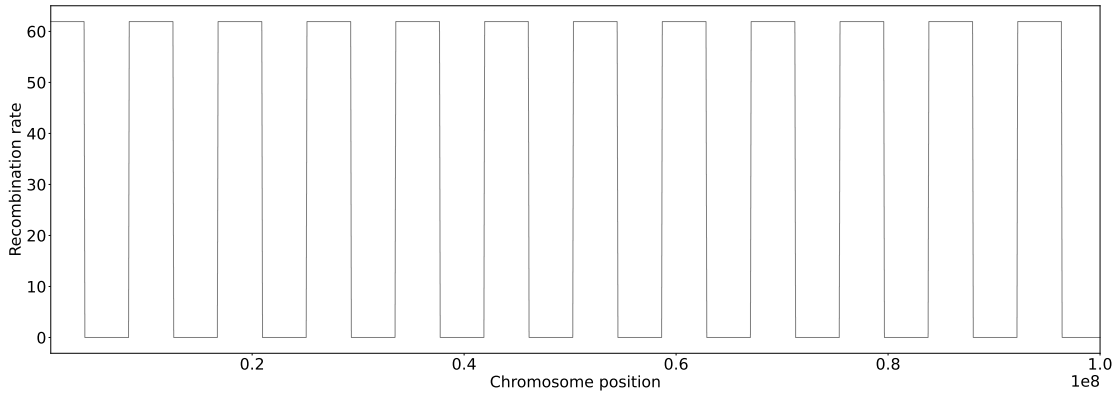


Fig G. Extract of the block recombination map. Extract of recombination map with only two different recombination rates. The mean recombination rate matches that of chr2. Displayed recombination rates along the chromosome are based on a sliding window of 70kb. This also ensures comparability with Fig F.

D Supplemental Figures

Supplemental Figure H: Comparison of iterative and non-iterative Estimators for $\theta \in (0, 40]$

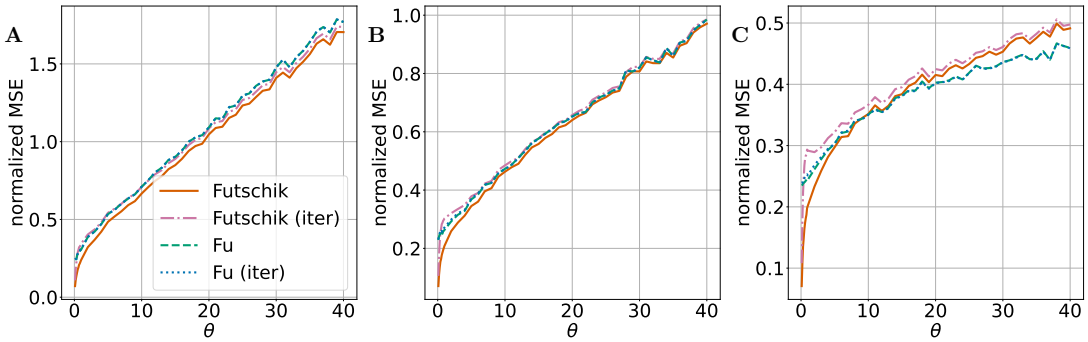


Fig H. Iterative estimators vs. non-iterative estimators. The normalized MSE for four different estimators are shown: The minimal MSE estimator (Futschik), the iterated minimal MSE estimator (Futschik (iter)), the minimal variance estimator (Fu) and the iterated minimal variance estimator (Fu (iter)). The same simulations as in Figure 1**Performance of mutation rate estimators.** The normalized MSE for four different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and two neural network estimators. A linear network without a hidden layer and an adaptive network with one hidden layer and 200 hidden nodes. For each subplot, we used msprime to generate a total of $4.9 \cdot 10^5$ independent simulations with sample size $n = 40$, recombination rate ρ, and 49 different mutation rates $\theta \in (0, 40]$. A: recombination rate $\rho = 0$, B: $\rho = 20$, C: $\rho = 35$, D: $\rho = 1000$. All shown neural networks have been trained on data sets with the corresponding recombination rate ρ. figure.caption.19 have been used. A: recombination rate $\rho = 0$ B: $\rho = 35$ C: $\rho = 1000$

Supplemental Figure I : Neural Network Performance outside the Training Range: θ in $(0, 500]$

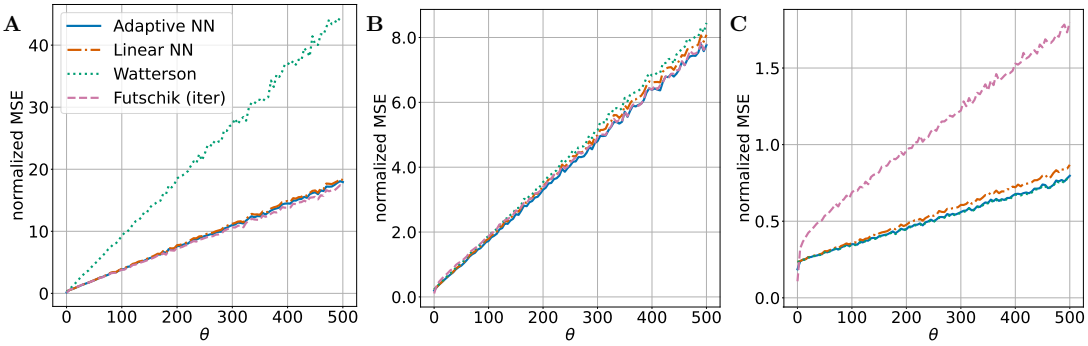


Fig I. MSE for estimators up to $\theta = 500$. The normalized MSE for four different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and two neural network estimators. A linear network without a hidden layer and an adaptive network with one hidden layer and 200 hidden nodes. All shown neural networks have been trained on data sets with the corresponding recombination rate ρ. For each subplot, we used msprime to generate a total of $18.9 \cdot 10^5$ independent simulations with sample size $n = 40$, recombination rate ρ, and 189 different mutation rates $\theta \in (0, 500]$. A: recombination rate $\rho = 0$ B: $\rho = 35$ C: $\rho = 1000$

Supplemental Figure J: Normalized Bias of Estimators

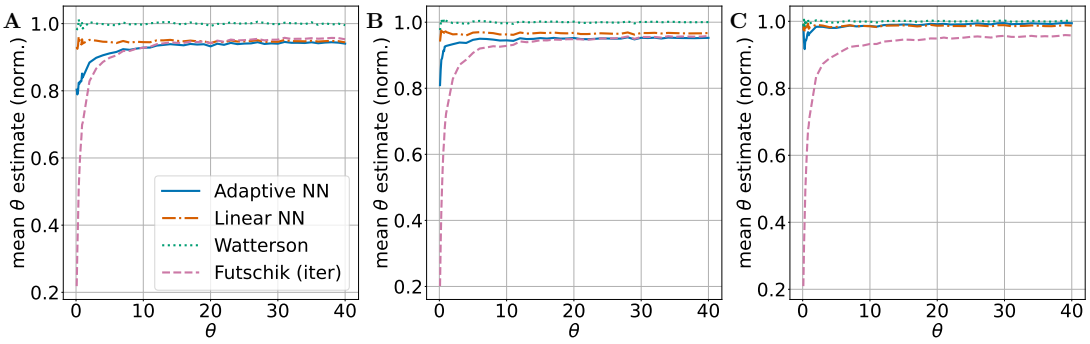


Fig J. Normalized bias of estimators. The normalized bias for four different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and two neural network estimators. A linear network without a hidden layer and an adaptive network with one hidden layer and 200 hidden nodes. All shown neural networks have been trained on data sets with the corresponding recombination rate ρ. The same simulations as in Figure 1**Performance of mutation rate estimators.** The normalized MSE for four different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and two neural network estimators. A linear network without a hidden layer and an adaptive network with one hidden layer and 200 hidden nodes. For each subplot, we used msprime to generate a total of $4.9 \cdot 10^5$ independent simulations with sample size $n = 40$, recombination rate ρ, and 49 different mutation rates $\theta \in (0, 40]$. A: recombination rate $\rho = 0$, B: $\rho = 20$, C: $\rho = 35$, D: $\rho = 1000$. All shown neural networks have been trained on data sets with the corresponding recombination rate ρ. figure.caption.19 have been used. A: recombination rate $\rho = 0$ B: $\rho = 35$ C: $\rho = 1000$

Supplemental Figure K: Coefficients of Fu's Estimator with Choice of Subsets for Adaptive Training

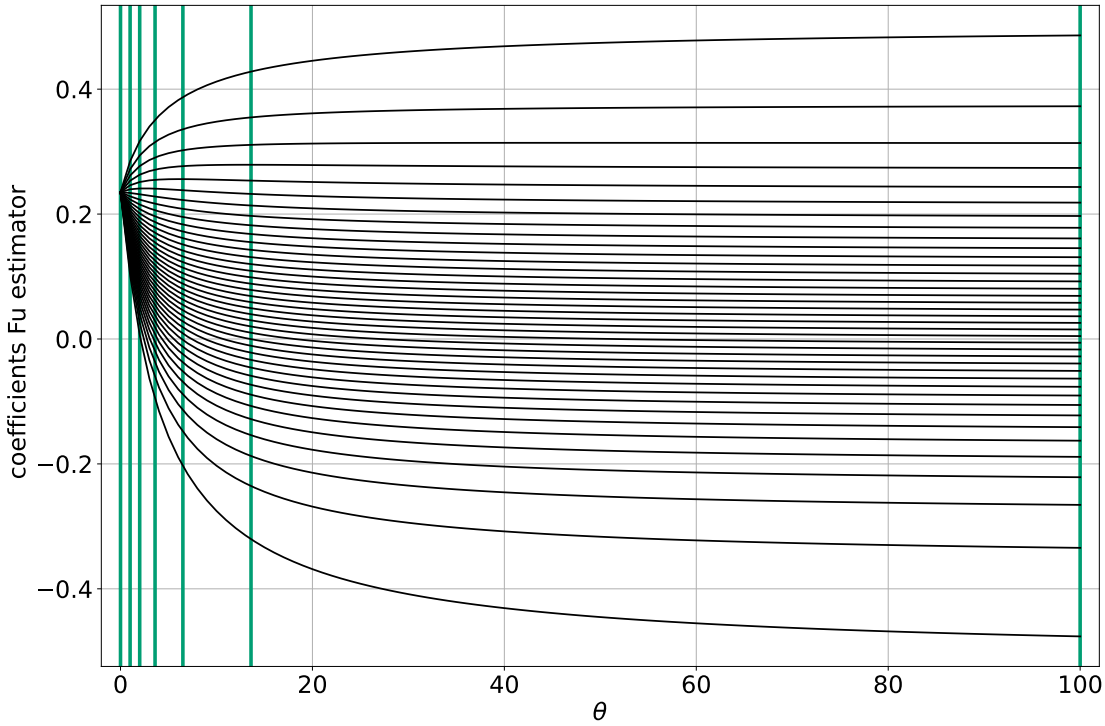


Fig K. Coefficients of Fu's estimator with subsets for adaptive training procedure. This figure shows the coefficients of non-interative Fu estimator for $n = 40$ (black) and the vertical lines represent the boundaries of the classes used in the adaptive training procedure for $n = 40$ (green).

Supplemental Figure L: Visualization of the Adaptive Neural Network Architecture

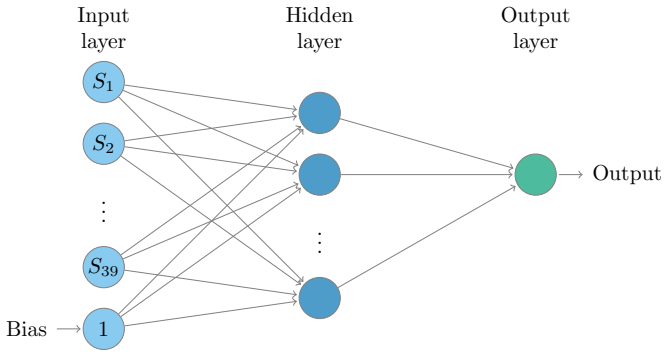


Fig L. Adaptive neural network architecture. The adaptive neural network considered in this publication is a dense feedforward neural network with one hidden layer, the site frequency spectrum as input and one bias node.

Supplemental Figure M: Example for Mutations along a Coalescent Tree

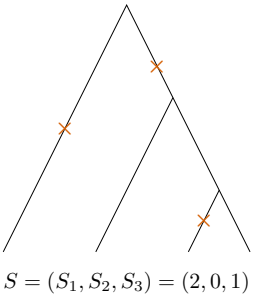


Fig M. Coalescent tree with mutations. Mutations along a coalescent tree for $n = 4$ and the corresponding site frequency spectrum S.

Supplemental Figure N: Robustness of the Adaptive Neural Network and the CNN by Flagel et al.

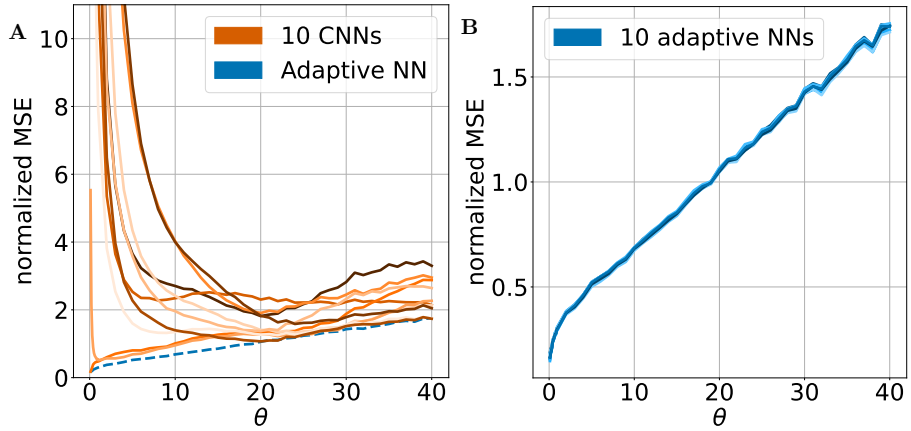


Fig N. Robustness of neural networks. Robustness for the CNN by Flagel et al. (A) and the adaptive neural network (B). In each subplot, ten trained CNNs or adaptive neural networks, respectively, are displayed in the scenario where the recombination rate $\rho = 0$. One adaptive neural network (dashed line) has been added to (A) to facilitate the comparison.

Supplemental Figure O: Modification of Watterson's Estimator by Futschik and Gach.

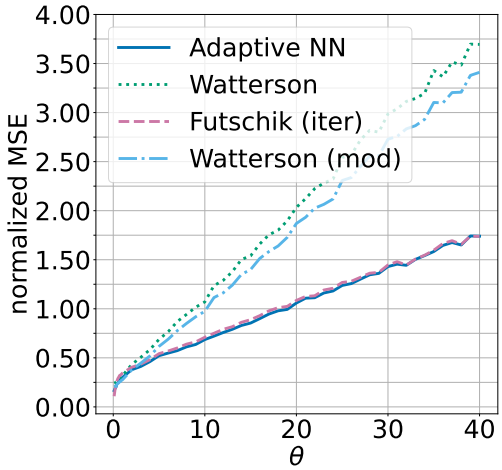


Fig O. Performance of Modification of Watterson's estimator. The normalized MSE for four different estimators are shown: The adaptive neural network, the iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and a modification of Watterson's estimator (Watterson (mod)) by Futschik and Gach [3]. For this plot, we used msprime to generate a total of $4.9 \cdot 10^5$ independent simulations with sample size $n = 40$, recombination rate $\rho = 0$, and 49 different mutation rates $\theta \in (0, 40]$. The adaptive neural network has been trained on a data set with $\rho = 0$. The modified version of Watterson's estimator by Futschik and Gach [3] is of the form

$$f(S) := \frac{2\sum_{i=1}^{n-1} S_i}{\mathbb{E}[L_n] + \frac{\mathbb{V}[L_n]}{\mathbb{E}[L_n]}},$$

with L_n being the length of the coalescent. For $\rho = 0$ the modified Watterson estimator is uniformly better than Watterson's estimator, see Lemma 2 in [3].

P Supplemental Figure: Extract of Recombination Map for Human Chr2.

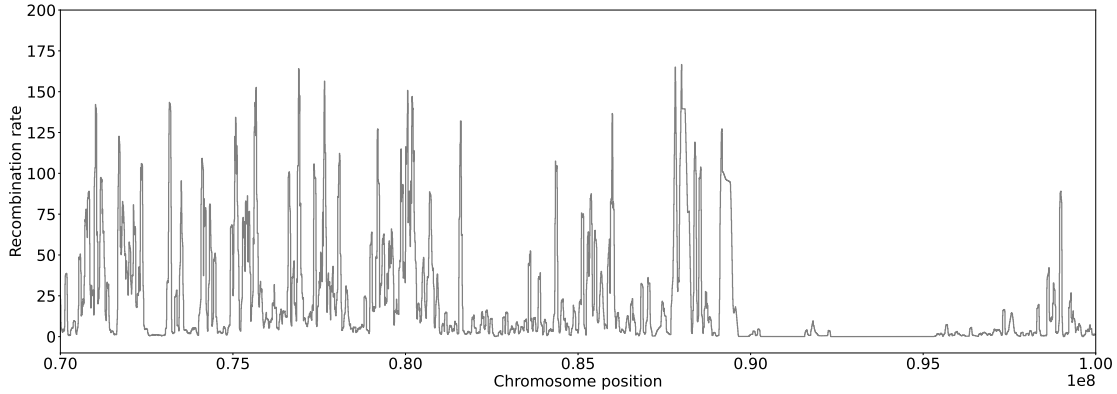


Fig P. Extract of recombination map for human chr2. Displayed recombination rates along the chromosome are based on a sliding window of 70kb. This equals the window size for the estimation of θ in Fig Q.

Supplemental Figure Q: Extract of θ Estimation for Human Chr2.

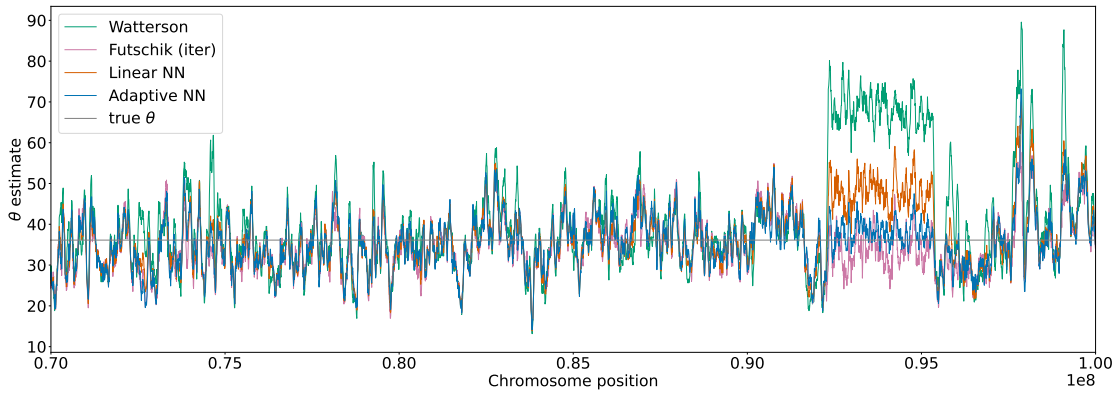


Fig Q. Extract of θ estimation for human chr2 The SFS for θ estimation was calculated based on a sliding window of size 70kb. Estimates based on Watterson, Futschik, the linear neural network and the adaptive neural network are displayed. The neural networks have been trained with variable recombination rates. The underlying true mutation rate $\theta = 36.12$ is shown as a grey line.

References

1. Lundberg SM, Lee SI. A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems 30; 2017. p. 4765–4774.

2. Kingma D, Ba J. Adam: A Method for Stochastic Optimization. International Conference on Learning Representations. 2014;.

3. Futschik A, Gach F. On the inadmissibility of Watterson's estimator. Theoretical Population Biology. 2008;73(2):212–221.

4. Flagel L, Brandvain Y, Schrider DR. The unreasonable effectiveness of convolutional neural networks in population genetic inference. Molecular Biology and Evolution. 2019;36(2):220–238.

2.2 Inferring Ancestral Relationships with the Hierarchical Soft Clustering Approach tangleGen

Klara Elisabeth Burger, Solveig Klepper, Ulrike von Luxburg, and Franz Baumdicker.

Accepted in Genome Research, 2024 [Burger et al., 2024].

Inferring ancestry with the hierarchical soft clustering approach tangleGen

Klara Elisabeth Burger[1], Solveig Klepper[1,2], Ulrike von Luxburg[1,2], and Franz Baumdicker[3,4,✉]

[1]Department of Computer Science, University of Tübingen, Germany
[2]Tübingen AI Center
[3]Cluster of Excellence "Controlling Microbes to Fight Infections", Mathematical and Computational Population Genetics, University of Tübingen, Germany
[4]Institute for Bioinformatics and Medical Informatics (IBMI), University of Tübingen, Germany

Abstract. Understanding the genetic ancestry of populations is central to numerous scientific and societal fields. It contributes to a better understanding of human evolutionary history, advances personalized medicine, aids in forensic identification, and allows individuals to connect to their genealogical roots. Existing methods, such as ADMIXTURE, have significantly improved our ability to infer ancestries. However, these methods typically work with a fixed number of independent ancestral populations. As a result, they provide insight into genetic admixture, but do not include a hierarchical interpretation. In particular, the intricate ancestral population structures remain difficult to unravel. Alternative methods with a consistent inheritance structure, such as hierarchical clustering, may offer benefits in terms of interpreting the inferred ancestries. Here, we present tangleGen, a soft clustering tool that transfers the hierarchical machine learning framework Tangles, which leverages graph theoretical concepts, to the field of population genetics. The hierarchical perspective of tangleGen on the composition and structure of populations improves the interpretability of the inferred ancestral relationships. Moreover, tangleGen adds a new layer of explainability, as it allows identifying the SNPs that are responsible for the clustering structure. We demonstrate the capabilities and benefits of tangleGen for the inference of ancestral relationships, using both simulated data and data from the 1000 Genomes Project.

Tangles | ADMIXTURE | ancestry inference | hierarchical clustering | population genetics
Correspondence: ***franz.baumdicker@uni-tuebingen.de***

Introduction

Inference of population structure is central to many studies (Mathieson and Scally 2020; Padhukasahasram 2014). Insights into the genetic ancestry of populations contribute to a better understanding of human evolutionary history (Reich et al. 2009) and advances personalized medicine (Rotimi and Jorde 2010; Reich et al. 2009), forensic identification (Pfaffelhuber et al. 2022), but also the conservation of endangered species (Pearse and Crandall 2004; Randi 2008). The analysis relies on genomic data, specifically the sequencing of single nucleotide polymorphisms (SNPs) for each individual. Distinct populations exhibit varying allele frequencies for specific SNPs, which allows conclusions to be drawn about the population structure. This information is often used as a basis to infer ancestries.

The development of methods to infer population structure has been part of research for decades, with both model-based (Pritchard et al. 2000; Tang et al. 2005; Raj et al. 2014; Behr et al. 2016; Wang 2022; Dominguez Mantes et al. 2023; Chiu et al. 2022) and model-free approaches (Patterson et al. 2006; Jombart et al. 2010) being pursued. For a compact summary, we recommend (Wang 2022). A popular modeling framework was introduced by Pritchard et al. (2000) in a Bayesian setting and also serves as the basis of ADMIXTURE by Alexander et al. (2009), a maximum likelihood method for the simultaneous estimation of ancestry proportions of individuals and allele frequencies in populations. The probabilistic model is based on biallelic SNP genotypes, where SNPs are assumed to be independent and originate from a fixed number of independent ancestral populations. The inferred ancestry proportions are often difficult to interpret due to the model assumptions, inference methods, and the complexity of actual demographic histories (Lawson et al. 2018). For example, in ADMIXTURE the user has to pre-specify the number of ancestral populations, which influences the inferred ancestry proportions. Since this number is typically unknown, it must be estimated (Wang 2022; Lawson et al. 2018). In practice, results are usually considered for a range of numbers of ancestral populations. However, results can provide inconsistent predictions for different numbers of ancestor populations. Furthermore, the results depend on the initialization, also known as *multimodality* (Jakobsson and Rosenberg 2007; Behr et al. 2016). Challenges such as these increase the risk of over-interpreting the results (Lawson et al. 2018). Several efforts aimed to address specific constraints of ADMIXTURE. For example, Behr et al. (2016) addresses multimodality, while Dominguez Mantes et al. (2023) reimplements the ADMIXTURE framework with neural networks to improve computational efficiency. Moreover, independent of ADMIXTURE, a more recent approach considers a hierarchical structure for four ancestral populations based on the length of ancestral tracts along genomes (Zhang et al. 2024). Methods with a consistent inheritance structure, such as hierarchical clustering methods, may offer benefits in terms of interpreting the inferred ancestries.

Here we introduce tangleGen, a hierarchical clustering method, to infer population structures in population genetics and exploit its capabilities. tangleGen is a model-free approach and does not infer ancestry proportions in the sense of ADMIXTURE and related methods that are based on a model that assumes K independent ancestral populations. Instead, it infers a hierarchical clustering of the individuals which sheds light on their ancestral relationships. tangleGen is based on Tangles, a theoretical

concept that originated from mathematical graph theory, where they were initially conceptualized to represent highly cohesive structures within graphs (Diestel 2018, 2019; Diestel and Whittle 2023).
Recently, the theoretical concept has been translated into an algorithmic framework (Klepper et al. 2023) that offers a highly flexible approach to soft and hard hierarchical clustering. The flexibility of the framework allows for a versatile application for genetic data, where the key components of the algorithm can be customized to account for different data types as well as to focus on different aspects of the underlying population structure. Tangles aggregates information about the data set's structure from many bipartitions that give local insight. Thereby, the information from many weaker, imperfect bipartitions is combined into an expressive clustering.
tangleGen provides a powerful new tool for researchers in population genetics. It not only improves interpretability by providing a hierarchical perspective on the composition and structure of populations, but also adds a new level of explainability to the clustering by disentangling the individual contributions of relevant SNPs. In addition, tangleGen is deterministic and identifies the number of related populations as a result of its hierarchical clustering, rather than requiring the number of independent populations as a pre-specified input. Thus evolutionary research on species that have a recent and entangled population structure, as well as practical applications where the impact of particular SNPs on the inference is of importance, can specifically benefit from tangleGen.

Results

The fundamental concept of Tangles revolves around the aggregation of information from bipartitions of the data set. Therefore, tangleGen uses bipartitions of genetic data to achieve a hierarchical soft clustering. We first introduce the concepts of tangleGen using a minimal example before applying the method to both simulated and real data in the next section.

tangleGen: a graph-based concept for population genetics. To infer ancestral relationships, tangleGen proceeds in four steps:

1. *Constructing basic cuts on the set of individuals*: The foundation of tangleGen are many bipartitions on the set of individuals, dividing them into two groups. Thus we will refer to bipartitions as "cuts" here. tangleGen uses cuts based on SNPs, which enhances explainability in the context of population genetics and genetic data.
2. *Assigning costs to cuts to sort them for the hierarchical clustering*: A well chosen cost function favors cuts with higher discriminative power by assigning them lower costs, while penalizing cuts that separate closely related groups of individuals. The correspondingly sorted cuts enable the next iterative hierarchical clustering step.
3. *Iteratively composing the tangles tree by orienting cuts*: Beginning with the lowest-cost cut, the algorithm iteratively orients the cuts to delimit clusters of individuals. Such a meaningful orientation of a subset of cuts that identifies a cluster is called a "tangle". These tangles form the basis for constructing a tangles tree, which represents the resulting hierarchical cluster structure.
4. *Computing a soft clustering based on characteristic cuts to infer ancestral relationships*: Characteristic cuts are cuts that define the tangles tree and, consequently, the identified cluster structure. Based on them, the soft clustering is computed, a value between 0 and 1 for each individual and cluster, indicating how likely an individual belongs to a particular cluster. Finally, the method infers the ancestry of individuals based on the soft clustering results.

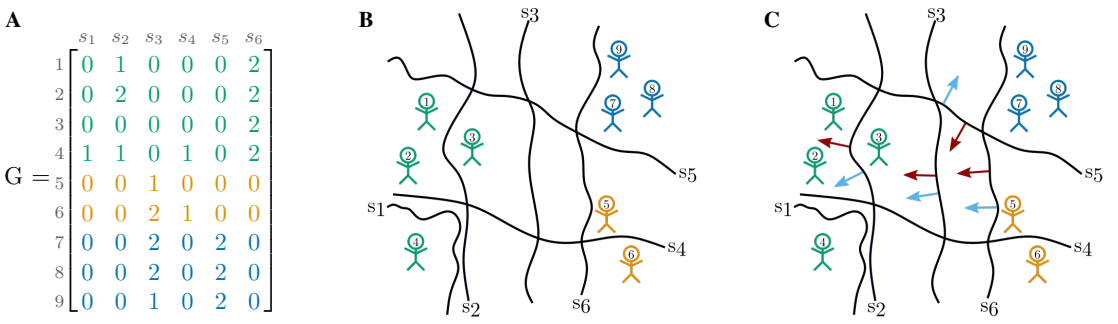


Fig. 1. tangleGen concepts: cuts and orientations in a minimal example. A: Representative genotype matrix, which indicates the number of derived alleles per individual per site. Individuals in the rows, SNPs in the columns. **B: SNPs induce cuts** (black lines) on the set of individuals, separating individuals without the derived allele (genotype matrix entry 0) from those carrying at least one derived allele (genotype matrix entry > 0). Individuals are shown as stick figures with the corresponding individual IDs. The colors correspond to the underlying population structure. **C: Orientations of cuts**: Red arrows indicate a meaningful orientation, that is a consistent orientation of the cuts for agreement parameter $\alpha = 2$, blue arrows a non-meaningful/inconsistent orientation.

Let us explain the steps in more detail, based on the following minimal example. We consider nine individuals from three populations. Population "A" comprises four individuals, while populations "B" and "C" comprise two and three individuals, respectively, and are more closely related. We consider six SNPs, and the corresponding genotype matrix is shown in Fig. 1 A.

Constructing basic cuts on the set of individuals. The basic ingredients, necessary to construct tangles, are cuts that divide the set of individuals. Naturally, most cuts will not separate distinct populations perfectly from one another. However, the information from many weaker, imperfect cuts is aggregated into an expressive clustering. Therefore, this approach can be seen as a form of boosting (Klepper et al. 2023). For population genetic data, each SNP naturally divides the individuals into two groups: Those individuals who do not carry the derived allele at all (genotype matrix entry is 0) and those who carry it at least once (genotype matrix entry is 1 or 2). A visualization is shown in Fig. 1 B. The rationale behind this procedure is that individuals in the same population are more likely to exhibit similar presence/absence patterns for mutations with discriminative power. Conceptualized in terms of cuts, individuals from the same population tend to lie on the same side of many cuts. The information provided by the cuts will later be aggregated by *tangles*.

Assigning costs to cuts to sort them for the hierarchical clustering. Clearly for some SNPs the corresponding cuts provide more information about the underlying population structure than others. Even if a SNPs occurs in different frequencies in different populations, the corresponding cut does not necessarily have to distinguish the populations well. But some mutations do distinguish populations, for example mutations in the LCT gene associated with lactose tolerance distinguish very well between Northern Europeans and East Asians (Sahi 1994; Ingram et al. 2009). To allow tangleGen to exploit this knowledge, we need to provide a cost function to evaluate and sort the cuts. Here, we define a custom cost function to infer demographic structures from genomic data based on the Fixation Index (F_{ST}) F, a commonly used measure of genetic differentiation. F compares the genetic variability within and between populations and is therefore an intuitive choice.
The F_{ST} value is usually calculated based on a single SNP and known populations S and T. However, in our case we do not yet know the underlying populations. Instead each cut based on a SNP s provides a suggestion on how to separate the individuals into two groups. If the resulting groups A and B are the underlying populations for the F_{ST} calculation, the F_{ST} value for the particular SNP s is obviously not informative, as it induced A and B. Therefore, we instead evaluate a cut by taking the mean F_{ST} value over all SNPs with respect to the division induced by the cut s:

$$c(s) = c_1 \cdot c_2 \cdot \left(\frac{1}{M} \sum_{m=1}^{M} F_A^m + F_B^m \right)^{-1},$$

where M is the number of SNPs, F_A^m (F_B^m) the F_{ST} value of SNP s_m regarding the separation of group A (B) and c_1, c_2 are penalizing factors as described in the Methods section, Eq. 3 and Eq. 4. The cost function favors cuts that divide individuals into groups that are more differentiated. The lower the cost, the more expressive the cut for the underlying population structure. However, such a F_{ST}-based cost function does not necessarily punish cuts that separate smaller sub-populations if the global division into A and B still has a high mean fixation index. To disfavor such cuts, which can potentially create inconsistencies early on, we add the penalizing factor c_1 based on k-nearest-neighbors (kNN). Hereby, the number of neighbors to be considered k is introduced as a hyperparameter. Supplemental Note 11 includes an analysis of the effect of k. The more a cut separates closely related individuals, the more it is penalized by c_1. Second, when inferring population structures using a hierarchical method, it is desirable to identify larger populations first. Therefore, the second penalizing factor c_2 slightly prefers cuts that divide individuals into groups of equal size. Details can be found in the Methods section.
In the minimum example, with $k = 2$ for the kNN penalty c_1, we obtain the following ordering starting with the cut with the lowest cost: s_3, s_6, s_5, s_2, s_1, s_4. The algorithm then processes all cuts sequentially, starting with the most useful cut, s_3. Low-cost SNPs will be positioned higher in the hierarchical tangles tree and therefore contribute to the formation of the main clusters. Conversely, more costly SNPs are considered later and contribute to more fine-grained population structures or are disregarded in the process.
Of course, the choice of the cost function is not unique. The flexibility of the Tangles framework allows tangleGen to prioritize different aspects by tailoring the cost function. Depending on the specific use case, an alternative cost function to the one presented in this context may be more appropriate. For an alternative cost function based on the mean deviation from a Hardy-Weinberg equilibrium, see Supplemental Note 6.

Iteratively composing the tangles tree by orienting cuts. Tangles, and therefore tangleGen, aggregates the information provided by the individual cuts by assigning orientations to each set of cuts. Multiple orientations that point towards the same "dense structure", in our case a group of more closely related individuals, are considered meaningful. The intuition is that orientations can characterize clusters, but not every orientation of cuts characterizes a cluster. For a set of oriented cuts to be meaningful, the intersection of individuals pointed to must contain at least a individuals, for each triplet of oriented cuts. Here, a represents the agreement parameter, a hyperparameter of tangleGen that roughly defines the minimum size of a cluster to be considered and should be set appropriately for each application scenario. A meaningful orientation of cuts is then called a $tangle$.

Fig. 1 C illustrates this concept for the cuts s_2, s_3, s_5 and s_6 and two possible sets of orientations indicated by red and blue arrows. Let us first consider the set of orientations indicated by the red arrows. This set of orientations is considered meaningful for agreement parameter $a = 2$, and therefore forms a tangle, because for each triplet of orientations (red arrows), all three orientations point to at least $3 > a$ individuals. Conversely, the orientation indicated by the blue arrows is deemed not meaningful: For each triplet of blue arrows containing the oriented cut s_5, the intersection of individuals pointed to by the corresponding blue arrows is empty and thus smaller than a. This concept of meaningful orientations outlined here is formalized in the definition of consistency in the Methods section.

Since the number of possible sets of cuts is large and checking the orientations for consistency can become computationally infeasible, tangleGen uses a tree-based search algorithm to handle the complexity. This approach iteratively generates the tangles tree: the cuts sorted by cost are evaluated one after the other, starting with the lowest-cost cut, and each cut is added to the tangle tree if it can be consistently oriented with respect to the previous cuts and their orientations. For an existing tangle, adding an oriented cut can either lead to a larger meaningful set of oriented cuts or to a loss of consistency. For an inconsistent orientation of cuts, consistency can not be restored by adding any cut. Consequently, larger tangles can only be built from smaller tangles, and thus naturally allow for a hierarchical representation in terms of a binary tree (Fig. 2 A).

Here we illustrate the construction of such a tangles tree along the steps in the minimal example from Fig. 1 as depicted in Fig. 2 A. Starting with the cut of the lowest cost, s_3, the algorithm attempts to meaningfully orient this cut in both directions. For agreement parameter $a = 2$, s_3 can be oriented in both directions, leading to a split in the tree at the root. The next cut with the second lowest cost is s_6. This cut can be oriented with both branches, but only in one direction each, since the orientation must sufficiently overlap with the one of s_3, otherwise their intersection would point to $0 < a$ individuals. Therefore, the existing branches are extended without adding a split in the tangles tree. The same applies to s_2. The cut s_5 can again be meaningfully oriented in both directions, but only if the previous cuts point towards individuals 5, 6, 7, 8 and 9 (as illustrated in Fig.1 C). Therefore, a split occurs only in the right subtree. As soon as the next cut can no longer be oriented consistently, the algorithm automatically terminates, and the tangles tree is complete. Each node in the tangles tree corresponds to the tangle resulting from the oriented cuts above the node. Consequently, the algorithm alone determines the number of considered cuts and the number of populations. Unlike ADMIXTURE, we do not need to specify the number of populations; we define the size a of the smallest group considered a meaningful population. The resulting tree has several levels, where each level ℓ represents a step in the hierarchical clustering, and the corresponding nodes give the number of populations identified at that stage. For visualization, see Fig. 3. Additionally, tangleGen offers a pruning option that allows to exclude the external branches in the tangles tree, which are not supported by a sufficient number of cuts/SNPs.

In the next step, the constructed tangels tree will be used to cluster the individuals.

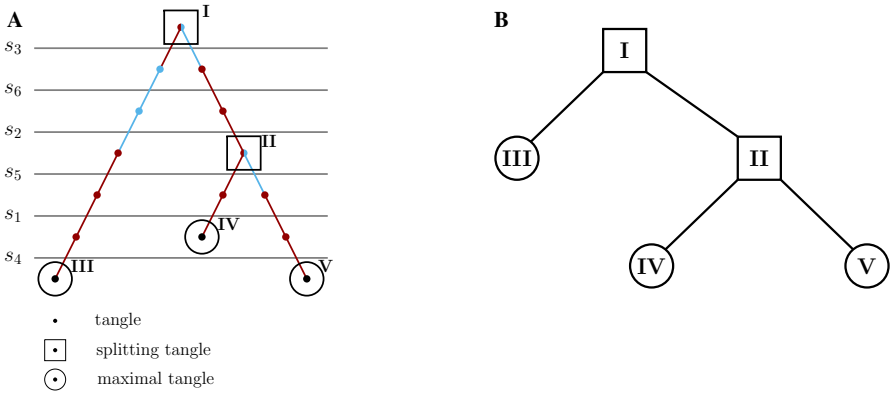


Fig. 2. Tangles trees for the minimal example. A: Tangles search tree. The constructed tangles search tree for the minimal example for agreement parameter $a = 2$. Cuts, sorted by cost, are evaluated sequentially, starting with the lowest-cost cut (s_3). A cut is added to the tangles tree if it can be consistently oriented with respect to the previous cuts and their orientations, forming a tangle. The color indicates whether the orientation of the last added cut points to all individuals with at least one derived allele (light blue) or in the opposite direction (red). When a cut can be oriented in both directions (s_3 and s_5), it creates a split at that node, resulting in a splitting tangle (squares). The cut s_4 cannot be added to every branch of the tangle tree. If no cut can be consistently added at any node or all cuts have been added to the tangles tree, the tangles tree is considered complete. The leaves represent the maximal tangles (circles). **B: Condensed tangles tree.** Condensed tangles tree for agreement parameter $a = 2$, that is the tangles search tree from A reduced to splitting and maximal tangles.

Computing a soft clustering based on characteristic cuts to infer ancestral relationships. Not all SNPs differentiate between different populations — but tangleGen can identify the SNPs that directly contributed to the splits in the tangles tree, which we refer to as characteristic SNPs here. This enhances the explainability of the method. With the F_{ST}-based cost function, the characteristic SNPs have a high, but not necessarily the highest, F_{ST} value, as the cost function is based on the induced mean F_{ST} of all *other* SNPs and takes further aspects into account. For a more detailed analysis, see Supplemental Table S2. In the

minimal example, s_2, s_3 and s_6 are characteristic for the first split and s_5 for the second. These characterizing SNPs/cuts are then processed via the soft clustering (Fig. 3 A for individual 3) into the desired inferred ancestry (Fig. 3 B).
The soft clustering step of Tangles can and should be adapted to the application, as it combines all the components of tangles considered so far into the final output, in our case, the inference of ancestral relationships. We thus developed an ancestry-inference-specific soft clustering step. All details can be found in the Methods section. In the following, we explain the concept and intuition behind tangleGen's soft clustering. This soft clustering is then illustrated in a stacked bar plot. Given the similarity of STRUCTURE and ADMIXTURE plots to this soft cluster bar plot (Fig. 3 B and all further soft clustering plots), it is important to clarify that tangleGen illustrates a hierarchical soft clustering instead of ancestry proportions derived from a probabilistic model with independent populations. In particular, the soft clustering is solely based on characteristic SNPs weighted according to their reliability factor.

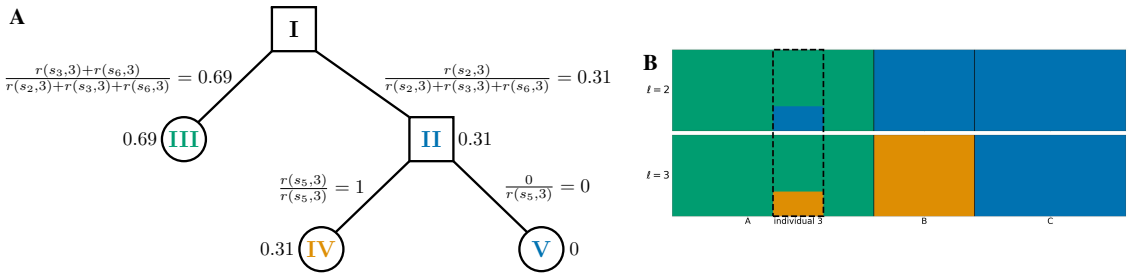


Fig. 3. Hierarchical soft clustering for the minimal example. A: Soft clustering for individual 3 and agreement parameter $\mathbf{a = 2}$, $r(s, 3)$ refers to the reliability factor for SNP s and individual 3 as defined in Eq. 5 in the Methods section. **B: Hierarchical clustering plot** of the individuals in the minimal example based on the soft clustering, with agreement parameter $a = 2$. Each subplot corresponds to a level ℓ in the tangles tree (Fig. 2 B), where the different levels result from splits in the tangles tree. Individuals are shown on the x-axis, the y-axis illustrates the soft clustering per individual. First bar plot shows clustering into two population, second into three. Individual 3 is highlighted because the soft clustering in A is calculated for this individual. tangleGen clusters the individuals hierarchically into the three populations A, B and C. The first split, which corresponds to the upper bar plot, is supported by three SNPs (s_2, s_3 and s_6) and the second by one SNP (s_5).

To infer the ancestry of the individuals, our objective is to determine, for each individual and split, a value between 0 and 1, indicating how likely, based on the SNP-based cuts, a specific individual belongs to the left or right subtree/cluster. To achieve this, we evaluate each split in the tangles tree and its characterizing SNPs separately for each individual. Given an individual and a split, we could simply compute the fraction of characterizing SNPs that assign the individual to, say, the left subtree/cluster, relative to the total number of characterizing SNPs. However, not every characterizing SNP is equally reliable for identifying population structure. Well-mixed populations are assumed to be in Hardy-Weinberg equilibrium, but SNP-based cuts cannot account for this. For example, the cut based on SNP s_2 in Fig. 1 B separates individuals 1, 2 and 4 from 3, 5, 6, 7, 8 and 9. But, under the Hardy-Weinberg assumption, it is clear that individual 3 is misclassified and should be grouped with individuals 1, 2 and 4 instead. Due to the nature of SNP-based cuts, such misclassifications are inevitable at SNPs that are not homozygous for all individuals, that is the SNPs s_1, s_2, s_3 and s_4 in the minimal example. To alleviate this, we assign a reliability factor to each cut, which calculates the probability that the individuals are correctly assigned to the two sides in respect to an assumed Hardy-Weinberg equilibrium. For explicit calculations, see the section about the soft clustering in Methods. Here we only give an intuition based on cut s_2 in the minimal example.
Considering the side of s_2 with individuals 1, 2, and 4, it is clear that under the assumption of a Hardy-Weinberg equilibrium one homoyzgous sample, here likely individual 3, has been misclassified. Thus, 1 out of 6 individuals on the other side of the cut is expected to be misclassified, resulting in a reliability of $r = \frac{5}{6}$. For the other side of the cut, we expect at least one of the individuals 1 and 4 to be misclassified, with individual 2 most likely being correctly assigned. This leads to a reliability of about $r = 0.39$. With this, we can now calculate the soft clustering: we take the fraction of characterizing SNPs that assign an individual to one side, and then weight each characteristic cut according to its reliability r. Fig. 3 A shows the complete soft clustering of individual 3, which matches our intuition. For example, for the first split, individual 3 has a soft clustering probability of 0.69 for belonging to the left subtree (cluster A) and 0.31 for the right subtree (clusters B and C), which is reasonable as only SNP s_2 orients individual 3 towards the right cluster. Computing the soft clustering for every individual and split results in a hierarchical plot showing the inferred ancestries for each individual, Fig. 3 B. In particular, individual 3 is partially assigned to populations A and B/C, which matches Fig. 1 B. In comparison, due to the very low number of SNPs the minimal example represents a greater challenge for ADMIXTURE, as can be seen in the Supplemental Fig. S1.

tangleGen infers population structures.

Inferring simulated population structures. First, we demonstrate tangleGen on simulated data, which allows the underlying demographics to be specified while controlling for factors such as migration, mutation rate and recombination rate. We used msprime (Baumdicker et al. 2022) to simulate 800 diploid individuals, with a total of 5041 biallelic SNPs within the predefined

population structure shown in Fig. 4 A. Details on the simulation can be found in the Methods section *Data Simulation*. Fig. 4 D shows the hierarchical soft clustering of tangleGen with this data set. Clearly, tangleGen clusters individuals into the correct populations. The comparison of Fig. 4 A with Fig. 4 B demonstrates that tangleGen reveals a hierarchical clustering consistent with the underlying population structure in the simulation. However, at the lowest level, involving migration between populations A and E, tangleGen splits A from B before E from F.

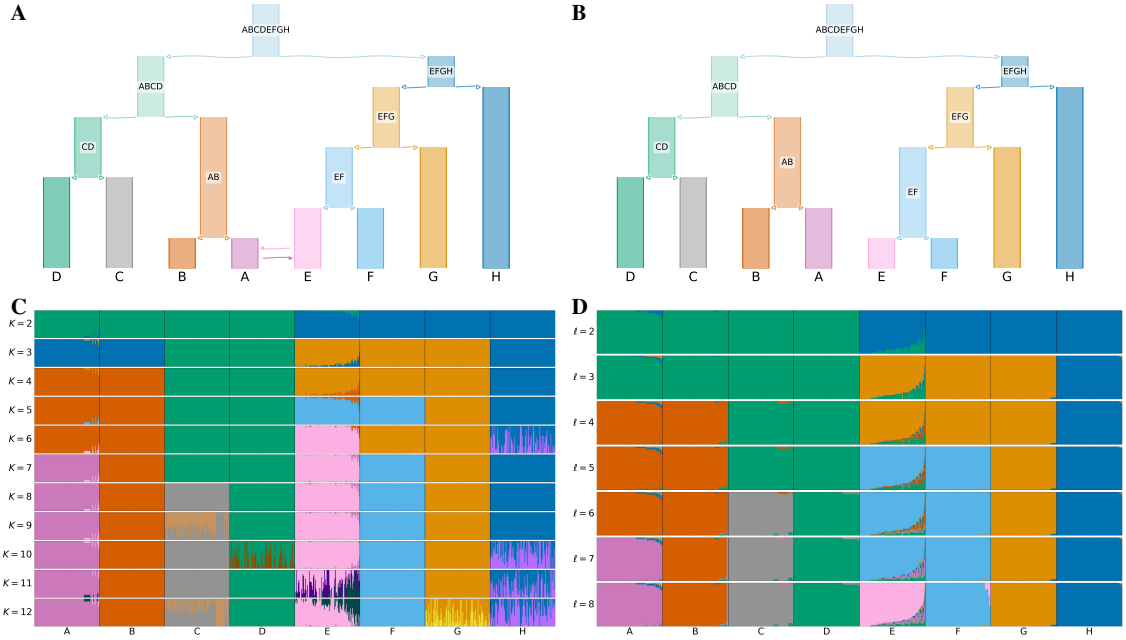


Fig. 4. Comparison of ADMIXTURE and tangleGen's inferred hierarchical population structure on simulated data. A: Underyling demographic structure in simulation with migration added between populations A and E. **B: Inferred population structure by tangleGen.** The hierarchy extracted from the tangles tree in tangleGen coincides with the simulated population structure except for the reversed order of splitting AB and EF. Note that tangleGen only estimates the population structure and not the height of the hierarchy; the branch lengths shown here are purely schematic. Figures A and B are visualized with *DemesDraw*. **C: ADMIXTURE ancestry proportions.** Inferred ancestries by ADMIXTURE from simulated genetic data for different numbers of populations, denoted by K. Individuals are sorted within the populations in the same order as in the tangleGen soft custer plot D. The best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. ADMIXTURE estimates the ancestry proportions well for the true number of populations $K = 8$, but gives also plausible results for other choices of K, although for $K > 8$ the simulated data does not contain any further population structure. Furthermore, the results for different K show inconsistencies, for example, for $K = 3$ and also $K = 6$. **D: Soft clustering of tangleGen.** Inferred ancestral relationships by tangleGen on simulated data with underlying demography as shown in A. The individuals are sorted within the populations based on the soft cluster proportions to achieve a block structure in the plots. Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. tangleGen clusters individuals hierarchically into the correct populations, whereby the hierarchy matches the underlying population structure up to the lowest level. In contrast to ADMIXTURE, tangleGen independently determines the depth of the population structure and completes the inference when no further SNP-based cuts can be consistently added. Starting with the top split between the ancestral populations ABCD and EFGH, each population split is based on 51, 177, 96, 76, 139, 1 and 18 characteristic SNPs. tangleGen parameters: Agreement parameter $a = 50$ and cost function as in Eq. 2 with $k = 40$.

For comparison, we run ADMIXTURE with the same simulated data set for K between 2 and 12, the input parameter indicating the number of populations to be identified. Results for this simulated scenario, along with those for the 1000 Genomes dataset, using fastStructure and SCOPE, are provided in Supplemental Fig. S8 and S9. In contrast to tangleGen, ADMIXTURE, Fig. 4 C, also correctly assigns individuals for $K = 8$ to their respective populations but, as expected, does not produce a consistent hierarchical clustering. Inconsistencies are, for example, already observed between $K = 2$ and $K = 3$, where populations ABCD and EFGH are initially determined as ancestral populations, and in the next level, ABH, CD, and EFG, leading to an arrangement inconsistent with any underlying demography. Furthermore, interpretability with ADMIXTURE is hindered by the uncertainty surrounding the choice of the value of K, an input parameter. This makes it unclear which K value best captures the most detailed population structure. From $K = 9$ ADMIXTURE does not determine any further populations, but also already subdivides population H at $K = 6$. In contrast, tangleGen demonstrates consistency, stopping when it cannot identify additional populations of sufficient size, and is unaffected by random factors, unlike ADMIXTURE, which exhibits multimodality issues as the inferred ancestry proportions vary significantly depending on the initialization.

tangleGen is particularly effective at inferring ancestral relationships from clearly differentiated populations, as demonstrated by these simulations. However, tangleGen also handles more complex demographies well, such as those with increased migration leading to less differentiated populations. Fig. 5 B illustrates tangleGen's performance on simulations with a quadrupled migration rate and migration between all populations from A to H. Due to tangleGen's hierarchical structure, the migration

patterns are clearly recognizable. In addition, except for an early split of population E, which shares ancestry with both subtrees in the underlying population structure (Fig. 5 A), the order of splits is consistent with the underlying population structure. Population E is also already split off using a low K with ADMIXTURE (see Supplemental Fig. S10 D). Furthermore, increased migration reduces the number of SNPs that separate populations well. This is also reflected in tangleGen, as clustering is now based on fewer SNPs. If the migration rate is increased even further, this effect becomes more pronounced, and the barplot of the soft clustering becomes less smooth (see Supplemental Fig. S10). A detailed analysis with additional migration scenarios, including a comparison to ADMIXTURE, is provided in Supplemental Note 10. Applications that involve a much higher number of SNPs pose a greater challenge to tangleGen, as expected for any hierarchical SNP-based method. For an example involving significantly more SNPs and significantly shorter time intervals between population splits, see Supplemental Note 6.

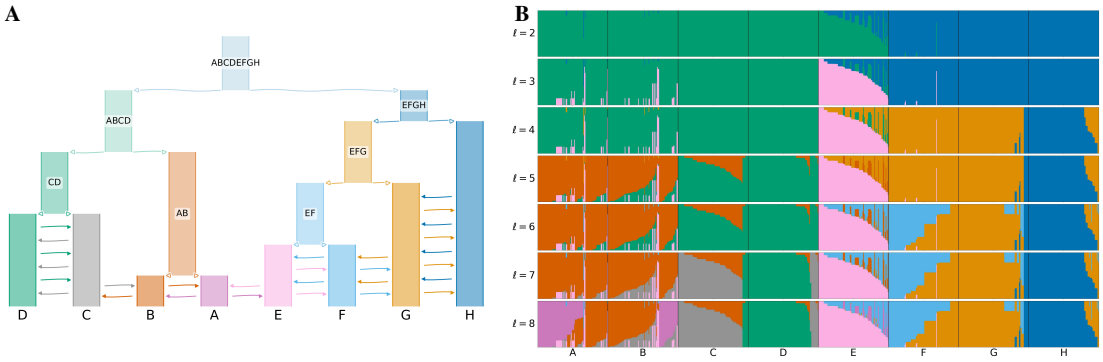


Fig. 5. tangleGen on simulated data with many admixture events. A: Underlying demographic structure in the simulation with migration added between populations A to H as indicated by the arrows. Migration rate is quadrupled in comparison to Fig. 4 A. See the Data Simulation section in Methods for more details. **B: Soft clustering of tangleGen.** Inferred ancestral relationships by tangleGen on simulated data with underlying demography as shown in A. The plot shows the soft clustering for different levels ℓ just as in Fig. 4 D. Note, that the maximal value of ℓ is a result of its hierarchical clustering in combination with the agreement parameter. tangleGen infers meaningful population structures even with increased migration rate. Higher migration rates result in more admixed populations and the hierarchical nature of tangleGen shows the migration pattern clearly. Starting with the top split between the ancestral populations ABCDE and FGH, each split is based on 29, 11, 10, 36, 10, 4 and 3 characteristic SNPs. In contrast to Fig. 4 D, the agreement parameter is set to $a = 30$ and all external branches supported by only one SNP are pruned (pruning parameter equals 1), to account for the less differentiated populations.

Inferring population structure within the 1000 Genomes Project. We used 2157 diploid individuals sampled in the data from 1000 Genomes Project Consortium (2015) Phase 3 and extracted the SNPs of the AIMs panel by Kidd et al. (2014). This panel is widespread, includes 55 SNPs strategically distributed across almost all chromosomes, and is known to be suitable for distinguishing superpopulations within the 1000 Genome data set: Africa (AFR), Europe (EUR), South Asia (SAS) and East Asia (EAS) (Pfaffelhuber et al. 2020).

Fig. 6 B shows tangleGen's soft clustering for Kidd's AIMs panel. It is evident that tangleGen correctly clusters the individuals into the four superpopulations: African (AFR), European (EUR), South Asian (SAS), and East Asian (EAS) populations. The inferred hierarchy reflects meaningful patterns and aligns with the out of africa dispersal (Mellars 2006). Fig. 6 A illustrates that individuals that are not assigned to a single population by tangleGen, are the ones positioned between multiple populations within a PCA analysis based on the AIMs panel. Furthermore, once again, the hierarchical nature of soft clustering is evident, and the leaves of the tree, when tangleGen has inferred the maximal level $\ell = 4$, can be assigned to the AFR, EUR, SAS, and EAS superpopulations from left to right. This soft clustering is then processed into the plot in Fig. 6 B.

For comparison, we run ADMIXTURE with a fixed number of ancestral populations between 2 to 4, as shown in Fig. 6 C. Like tangleGen, ADMIXTURE also identifies the four superpopulations, with tangleGen clustering in a similar way to the well-established method. However, as in the simulation, ADMIXTURE also shows inconsistencies here, even if these do not relate to miss-classifying entire populations. For example, when differentiating into three populations, we observe that the South Asian populations share a significant part of their ancestry with the East Asian populations. However, when differentiating into four populations, this component disappears almost completely. Again, ADMIXTURE's performance depends on the choice of K. Fig. 6 C considers K only up to 4, which is reasonable in this application as ADMIXTURE starts to identify non-existing population structures from there (see Supplemental Fig. S3 for K up to 7). In contrast, tangleGen determines the number of populations as a result of its hierarchical clustering in combination with the agreement parameter.

Identification of characteristic SNPs in the tangles tree. Apart from the inherent advantages of a hierarchical method, tangleGen offers the possibility to identify specific cuts and therefore SNPs crucial for the cluster formation in the tangles tree. This enhances the method's explainability, as identified SNPs can be cross-referenced with expert knowledge. tangleGen determined 15 out of 55 SNPs of the Kidd's AIMs set as characterizing SNPs to cluster the 2157 individuals in the 1000 Genomes Project (AMR excluded), Fig. 6 A. In particular, nine SNPs were found to be characteristic for the first split in the tangles tree separating the AFR populations, while three each were used for the second and third splits that separate the

EUR, SAS, and EAS populations. The characterizing cut with the lowest cost, SNP with ID rs2814778 (following dbSNP ID notation by Sherry et al. (2001)), influences all subsequent orientations and therefore determines that AFR populations are identified first in the hierarchical tangles tree. Notably, this SNP is a recognized component of the molecular basis of the Duffy blood group system, linked to malaria resistance and exhibits near-fixed differentiation between Africans and the other superpopulations (Pfaffelhuber et al. 2020). Furthermore, rs3827760 is known to separate the East Asian populations from the other superpopulations (Pfaffelhuber et al. 2022) and has been identified by tangleGen as the main SNP responsible for the corresponding split in the tangles tree. Finally, rs1426654, a SNP associated with iris color and skin pigmentation in South Asians (Riddell et al. 2020), is the lowest-cost characteristic SNP for the third split in the tangles tree separating between European and South Asian populations. Although tangleGen aims to extract the genetic background structure and does not explicitly identify or require SNPs under local adaptation, such SNPs will accelerate the clustering process and thus frequently appear among the most reliable characterizing cuts. In total, all 15 SNPs identified by tangleGen are meaningfully linked to their respective population splits in the tangles tree. The complete list of characterizing SNPs is provided in Supplemental Note 4. In addition, tangleGen can be employed to construct a cluster-typical set of SNPs, comprising relevant cuts for each cluster alongside their orientations. This can be useful as it, for example, facilitates a rapid, albeit approximate, clustering of individuals without the need to rerun the method. Note, the cluster-typical SNPs do not represent the genome of any specific individual as consistencies during the construction of the tangles tree are always only ensured for triplets of cuts. Instead they represent the abstract concept of a cluster center. Supplemental Table S1 includes the cluster-typical SNPs for the application on the 1000 Genomes Project data based on Kidd's AIMs panel.

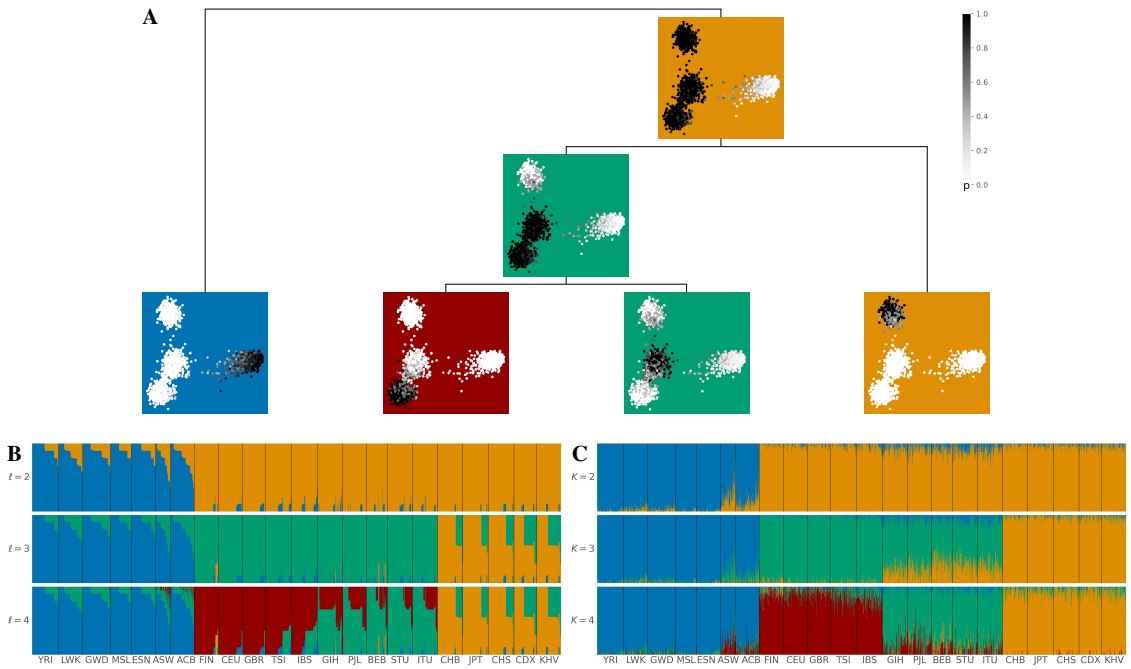


Fig. 6. tangleGen's hierarchical soft clustering for the 1000 Genomes Project. Inferred ancestries by tangleGen (B) and ADMIXTURE (C) based on SNPs from Kidd's AIMs set for individuals sampled in the 1000 Genomes Project Phase 3, excluding AMR. **A: Soft clustering of tangleGen visualized with PCA** for each individual and each split in the tangles tree. Panels at each split in the tangles tree show a PCA plot based on the AIMs set genotype matrix. Each individual is represented as a dot, with darker colors indicating higher confidence in the assignment of the individual to the corresponding cluster. The PCA visualization highlights that individuals positioned between two populations are softly clustered into both populations. The same soft clustering is processed in B into an ADMIXTURE like bar plot. The resulting soft clustering is hierarchical and the leaves of the tree can be assigned to the AFR (blue), EUR (red), SAS (green), and EAS (orange) superpopulations. **B: Soft clustering by tangleGen visualized in a bar plot.** Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. tangleGen identifies the four superpopulations AFR, EUR, SAS and EAS with a meaningful inferred hierarchical population structure. The first split is supported by nine characteristic SNPs, and the second and third by three characteristic SNPs each. Cost function as specified in Eq. 2 with $k = 40$, and agreement parameter $a = 225$. **C: Ancestry proportions by ADMIXTURE.** ADMIXTURE identifies the four superpopulations AFR, EUR, SAS and EAS, when the number of populations is set to $K = 4$, but shows inconsistencies, compared to smaller values of K in the differentiation between the four populations. The best of ten runs is shown in terms of the clarity of the inferred populations and minimization of inconsistencies compared to the underlying population structure. In B and C, the individuals are sorted within the populations in the same way such that a block structure for the tangleGen plot B is achieved. The four superpopulations are made up as follows: AFR: YRI, LWK, GWD, MSL, ESN, ASW, ACB; EUR: FIN, CEU, GBR, TSI, IBS; SAS: GIH, PJL, BEB, STU, ITU; EAS: CHB, JPT, CHS, CDX, KHV. Results that include Ad-Mixed American (AMR) populations can be found in the Supplemental Fig. S2.

Discussion

tangleGen leverages the hierarchical Tangles framework for population genetics. Here we present tangleGen, an ancestry inference method that exploits the flexibility and robust hierarchical functionality of the Tangles framework. Robustness comes from the fact that Tangles provides results that are deterministic and consistent across different numbers of populations and do not exhibit multimodalities. The flexibility of Tangles lies mainly in the customization of the cuts and cost function to the specific research questions and data structures. This allows tangleGen to focus on ancestry inference based on sequencing data. tangleGen investigates the ancestry of diploid individuals based on biallelic SNPs using cuts that separate individuals according to the presence or absence of at least one derived allele. This one-to-one relationship between SNPs and cuts provides interpretability, as it allows identifying individual SNPs responsible for the inferred population structure. However, as the derived allele is usually not homozygous for all individuals in any subpopulation, the corresponding cut will inevitably misassign some individuals. This error is mitigated by the weights of the cuts in the soft clustering. To obtain a meaningful hierarchical structure within the tangles tree we need to prioritize cuts according to their discriminative power between populations. Therefore, it is crucial to assign a suitable cost to each cut. Here, we opt for a cost based on the fixation index, a classical approach to distinguish populations. To derive meaningful inferred ancestral relationships from the tangles tree, tangleGen accounts for the reliability of the cuts. To do so, the soft clustering incorporates an estimate of the number of misclassified individuals according to an assumed Hardy-Weinberg equilibrium.
The Tangles framework has the flexibility to emphasize different research questions as incorporating alternative definitions of cuts, cost functions, and soft clustering allows to focus on different aspects of the data. Future investigations could, for example, explore cuts that consider multiallelic sites or separate haploids. Furthermore, cuts based on multiple SNPs rather than single SNPs could improve the performance but might lower the interpretability. Given the fundamental principles underlying F_{ST} and Hardy-Weinberg equilibrium (HWE), we expect that the proposed F_{ST}-based cost function and HWE-based reliability factor will be suitable for most diploid species. Nonetheless, the costs of the cuts and the soft clustering can be adjusted to specific use cases. As an example we propose an alternative cost function based on the divergence from an Hardy-Weinberg equilibrium in Supplemental Note 6 and evaluate it's performance in Supplemental Fig. S4, S5 and S6. Together, these features of the Tangles framework make tangleGen a particularly flexible approach in population genetics.

tangleGen parameters in practice. Through the choice of hyperparameters tangleGen provides an additional option to focus on different aspects of the clustering (Klepper et al. 2023). The agreement parameter a roughly corresponds to the minimum size for a cluster/population to be identified. Adjusting the agreement parameter provides the flexibility to obtain either major clusters (with a large a value) or minor clusters (with a small a value), allowing users to experiment with different settings. Typically, the agreement parameter should be set between 33% and 66% of the smallest desired population (Klepper et al. 2023). For example, in the simulated scenario, the agreement parameter is set to 50, which corresponds to half of each population size. For the analysis of the 1000 Genomes Project data set in Results, the agreement parameter is set to 225, which corresponds to about 34% of the largest superpopulation (AFR) and 46% of the smallest superpopulation (SAS). Further details are added in the Methods section and Supplemental Fig. S11, S12 and S13.
As Tangles does not require many SNPs for clustering and aggregates the available information efficently, we concentrated on data sets that contain a comparably low number of SNPs, which are the most promising application of the current method. For datasets with a higher number of SNPs, tangleGen currently exhibits reduced speed and accuracy. However, future improvements, such as the implementation of a more efficient cost function, can enhance its performance. Clearly, every hierarchical SNP based method will face unique challenges when dealing with large numbers of SNPs. A too early consideration of a SNP positioned high in the tangles tree may introduce cuts that are not representative of the actual population structures. When dealing with such extensive datasets containing a considerably higher number of SNPs compared to the two scenarios considered here, using the pruning option that defines the minimum number of cuts required to support a split in the tangles tree can help to avoid noise clusters (Klepper et al. 2023). In our experience, a pruning parameter of 1 or 2 is often sufficient. For population structures with less differentiated populations, it is advisable to use a smaller agreement parameter and positive pruning to ensure the detection of the populations while mitigating the formation of noise clusters. Further details can be found in the Supplemental Note 6 and 10.

tangleGen infers interpretable population structures. To evaluate tangleGen for inferring ancestral relationships, we applied the method to both simulated and the 1000 Genomes Project data. tangleGen performs comparably well to the established ADMIXTURE method in terms of clustering and ancestry inference. The advantage of using tangleGen lies in its interpretability. The hierarchical nature provides insights into ancestral relationships and offers a new perspective on the composition and evolution of populations. Unlike ADMIXTURE, tangleGen operates independently of any random factor and the specified number of ancestral populations K. Instead, tangleGen determines the number of populations as a result of its hierarchical clustering in combination with the agreement parameter as the algorithm terminates automatically if the consistency condition is no longer met. At this point, no further populations are split, and the number of maximal tangles in the tangles tree determines the final number of populations. Furthermore, as discussed in the Results section, tangleGen yields the characteristic SNPs for clustering. This increases the explainability of the method and allows to identify SNPs that differentiate between populations in

the tangles tree. Concentrating on a smaller set of automatically inferred cluster-distinguishing SNPs can be beneficial for the analysis of large-scale data sets, such as biobanks. In addition, a cluster-typical set of oriented characterizing SNPs can be used to cluster new individuals without running the method again. Identifying an explicit genetic state for each of the populations can provide a more intuitive view of the inferred ancestry. However, it is important to note that this set of oriented SNPs simply represents the center of the identified clusters and does not correspond to any existing individual in the population.

The hierarchical approach in tangleGen not only infers ancestries and identifies populations, but also constructs consistent relationships between populations over time. In addition to the inherent hierarchy, the ability to define a cluster-typical set of characteristic SNPs responsible for the clustering decisions distinguishes it from other methods. Moreover, tangleGen provides deterministic results and automatically determines the number of populations without requiring the user to specify. Furthermore, tangleGen inherits the flexibility of the underlying method, Tangles. With the ability to customize cuts, cost function, soft clustering and hyperparameters, tangleGen can be tailored to a wide range of research questions and is a versatile tool to investigate the genetic diversity and ancestral relationships between populations.

Methods

tangleGen implements the Tangles concept for population genetics. Tangles originate in mathematical graph theory, where they capture the concept of a highly cohesive structure in graphs. In their original form, they are based on all possible bipartitions of the nodes in a graph, and the existence of a tangle defines a densely connected structure. The algorithm presented in Klepper et al. (2023) distills this concept to its essence to cluster arbitrary data hierarchically. In the following, we outline the key concepts of Tangles and describe the specific adjustments to utilize the framework in the context of ancestry inference, specifically constructing SNP-based cuts, the cost function, and soft clustering. The implementation of tangleGen is available on GitHub: k-burger/tangles_in_pop_gen.
We use the notation and definitions introduced in Klepper et al. (2023). Assume we are given a set $V = \{v_1 \dots, v_N\}$ of individuals. A subset $A \subset V$ induces a **bipartition** or **cut** of the data into the set and its complement $S = \{A, A^\complement\}$. In order to construct tangles, we will consider a set of initial cuts $\mathcal{S} = \{\{A_1, A_1^\complement\}, \dots, \{A_M, A_M^\complement\}\}$.

Constructing SNP-based cuts to infer ancestries. Since we want to build upon the Tangles framework to analyze ancestries and population admixtures, we follow the most common methods and use the genotype matrix $G = (g_{nm}) \in \{0,1,2\}^{N \times M}$ as input, with g_{nm} the observed number of copies of the derived allele at site m of individual n. In particular, we consider diploid individuals and biallelic SNPs. To infer ancestries from the genotype matrix, we choose an initial set of cuts induced by SNPs. Thereby, each cut separates all individuals into two sets of individuals based on a single SNP, so that individuals carrying the derived allele at position m ($g_{nm} \in \{1,2\}$) are separated from those not carrying the derived allele at position m ($g_{nm} = 0$).

Consistent orientations of cuts and tangles. For a single cut $S = \{A, A^\complement\}$, an orientation "points" towards one of the sides. In our minimal example, SNP s_6 points towards $A = \{1,2,3,4\}$, that is, the individuals carrying the derived allele at SNP 6. For a set $\mathcal{S}$ of cuts, we define an orientation $O_\mathcal{S}$ by choosing one side for each cut, giving an orientation to every $S \in \mathcal{S}$. We write $A \in O_\mathcal{S}$ if $\{A, A^\complement\} \in \mathcal{S}$ and $O_\mathcal{S}$ orients it towards A. The intuition is that orientations can characterize clusters, but not every orientation characterizes a cluster: the orientations need to be "consistent" in some way. For a meaningful orientation, we have to ensure that the chosen sides of all the cuts point to one meaningful subgroup. This is precisely the purpose of tangles.
Definition 1 (**Consistency and tangles**) Let $\mathcal{S}$ be a set of cuts on a set V. For a fixed parameter $a \in \mathbb{N}$, an orientation $O_\mathcal{S}$ of $\mathcal{S}$ is *consistent* if all sets of three oriented cuts have at least a individuals in common:

$$\forall A, B, C \in O_\mathcal{S} : \quad |A \cap B \cap C| \geq a\,. \tag{1}$$

We call Eq. (1) the *consistency condition* and a the *agreement parameter*. A consistent orientation of $\mathcal{S}$ is called a tangle.
In the minimal example, Fig. 1 C, we illustrate the consistency condition for the following set of bipartitions:

$$\mathcal{S} = \{\{A_3 = \{5,6,7,8,9\},\ A_3^\complement\},\ \{A_6 = \{1,2,3,4\},\ A_6^\complement\},\ \{A_5 = \{7,8,9\},\ A_5^\complement\},\ \{A_2 = \{1,2,3\},\ A_2^\complement\}\}.$$

The **agreement parameter** a controls the minimum degree to which the orientations have to agree. When a is chosen too small, it weakens the induced consistency condition, causing the algorithm to identify non-cohesive subgroups. On the other hand, we want to avoid a too-large a; in practice, it should be slightly smaller than the expected size of the smallest cluster to account for noise. If the cuts in $\mathcal{S}$ respect the cluster structure, and if we have a diverse and rich set of $\mathcal{S}$, we can reduce a without mistakenly marking disconnected structures as clusters (compare with the appendix of Klepper et al. (2023)).

Finding all the tangles. To identify all *tangles* within a set of cuts, the algorithm in Klepper et al. (2023) progressively constructs a tangle search tree by considering cuts in an iterative manner. Useful cuts are prioritized, that is, those that lead to a meaningful split of the individuals (for example, do not cut through a group of similar individuals). This intuition is formalized through a

cost function $c : \mathcal{S} \to \mathbb{R}$. This function, application-dependent, quantifies the "quality" of a cut. Our choice of cost function is discussed in detail in the following section. The resulting labeled binary tree (for example Fig. 2 for the minimal example) contains all consistent orientations (tangles) within the considered subset of cuts that do not exceed some cost. Each node corresponds to a specific orientation of a particular cut, ensuring a one-to-one relationship with tangles. Specifically, for a node e with i nodes on the root-to-e path, the corresponding node labels form a consistent orientation of $S_1, \ldots, S_i$, that is a tangle.

New cost function for inferring ancestries. Tangles offers the flexibility to tailor the method to various applications and its requirements by using a suitable cost function. In order to infer ancestral relationships with tangleGen, we have chosen a cost function based on the fixation index F_{ST}, which has already been introduced in the minimal example in Results and is described in detail below. Let us consider a cut S_m that separates the individuals into groups A and $B = A^{\complement}$, $T = A \cup B$, with $N_A,\ N_B$ and N the group sizes. Since each SNP introduces a cut, we average the F_{ST} value F over all SNPs rather than considering only the SNP responsible for the cut. Let f_A^m, f_B^m and f_T^m be the frequency of the derived allele at SNP S_m in the groups $A,\ B$ and total population T respectively:

$$f_A^m = \frac{1}{2N_A} \sum_{n \in A} g_{nm}, \quad f_B^m = \frac{1}{2N_B} \sum_{n \in B} g_{nm}, \quad f_T^m = \frac{N_A}{N}\, f_A^m + \frac{N_B}{N}\, f_B^m.$$

Then, the mean F_{ST} value F for subpopulation A is given by

$$\bar{F}_A = \frac{1}{M} \sum_{m=1}^{M} \left| 1 - \frac{f_A^m (1 - f_A^m)}{f_T^m (1 - f_T^m)} \right|$$

and the resulting cost function $c : \mathcal{S} \to \mathbb{R}$ is defined by

$$c(S_m) := c_1 \cdot c_2 \cdot \left(\bar{F}_A + \bar{F}_B \right)^{-1} \tag{2}$$

with penalization factors c_1 (Eq. 3) and c_2 (Eq. 4). If kNN_A_in_B is the number of nearest neighbours from A that lie in B and kNN_B_in_A respectively, the penalization factor c_1 is given by

$$c_1 = 1 + \left(\frac{\text{kNN\_A\_in\_B}}{N_A \cdot k} + \frac{\text{kNN\_B\_in\_A}}{N_B \cdot k} \right) \tag{3}$$

with k the number of neighbors considered and

$$c_2 = \left(\frac{N}{N_A} + \frac{N}{N_B} \right)^b. \tag{4}$$

Hereby, b weights the penalization factor c_2. The higher b, the more severe cuts are penalized that divide individuals into groups of unequal size. In practice, $b = 0.05$ has proven to be a good choice, as cuts that divide individuals into groups of unequal size are sufficiently penalized, but changes in $\bar{F}_A,\ \bar{F}_B$ and c_1 are still reflected in the cost. The lower the cost $c(S_m)$, the more informative the cut is for the underlying population structure. To get a better impression of the influence of the cost function, we analyzed the impact of using a random instead of a cost-based order for the cuts (Supplemental Fig. S14 and S15), the effect of the penalizing factor c_2 (Supplemental Fig. S16), and the hyperparameters k and b (Supplemental Fig. S11 - S13).

Computing the clustering. The tangle search tree is constructed hierarchically on cuts, acting as a proxy for our ultimate interest: a hierarchical cluster structure of our individuals. To simplify, we can convert this tree into a condensed form, denoted as T^*, resembling a dendrogram (shown in Fig. 2 B for the minimal example). Internal nodes, which we call splitting tangles, represent divisions between regions in the data. However, for a single individual, a splitting tangle does not enforce a binary decision; rather, the condensed tree allows assigning probabilities for each individual and tangle, yielding a soft clustering. In the following, we explain how to summarize the information into a condensed tree and how to derive the soft clustering from it.
We first condense the tree to the splitting tangles and only keep the information of cuts that do give insight into the cluster structure; for every split in the tree, we identify all cuts that contribute to the split and thus 'characterize' the separation of the two denser structures. In our minimal example, Fig. 2, we consider the cuts s_2, s_3 and s_6 as characterizing cuts for the first splitting tangle τ at the node I since they provide information about the separation between the left and the right group. For the second splitting tangle τ at node II we identify s_5, which separates the upper from the lower cluster structure on the right side. All information is taken from the tangles search tree. For a cut S to be characterizing for a split, every tangle corresponding to a leaf in the one subtree needs to orient S the one way, and every tangle corresponding to a leaf in the other subtree needs to orient S the other way. In this sense, the characterizing cuts are the ones that help in distinguishing between the two subtrees of τ. In our minimal example, we therefore have a set of characterizing cuts or rather characterizing SNPs $\mathcal{S}(\tau|_I) = \{s_2, s_3, s_6\}$ for the

first split I and for the second $\mathcal{S}(\tau|_{II}) = \{s_5\}$.
More formally, let $S(\tau)$ be the orientation of S in a tangle τ and let $T_\tau^{(\text{left})}$ be the left subtree and $T_\tau^{(\text{right})}$ be the right subtree of the node at a tangle τ. Then we define the set of characterizing cuts as

$$\mathcal{S}(\tau) \coloneqq \{S \in \mathcal{S} \mid \forall \text{ leaves } \tau^l \in T_\tau^{(\text{left})},\ \text{leaves } \tau^r \in T_\tau^{(\text{right})} \ :\ S(\tau^l) \neq S(\tau^r)\}.$$

Based on this information, the tree is condensed, and the set of characterizing cuts is tracked for each of the splitting tangles.

Inferring ancestries from the soft clustering. Let us now compute the soft clustering. Using the characterizing cuts, we derive a heuristic to assess how likely an individual $v \in V$ belongs to the left or right subtree (or in our case rather a subgroup) of τ. For each individual v and splitting tangle τ, we could simply calculate the fraction of characterizing cuts oriented toward v relative to the total number of characterizing cuts. However, not every characterizing cut is equally reliable in separating the identified populations. While characterizing cuts are well suited to distinguish structures based on the mean F_{ST}-value, each cut inevitably misclassifies some individuals. Once an SNP is not homozygous for each subpopulation individual, the corresponding cut will make classification errors. To solve this problem, we assign a reliability factor to each cut, which calculates how likely individuals are correctly assigned to the two sides based on the estimated allele frequencies assuming a well-mixed population in Hardy-Weinberg equilibrium on both sides of the cut. For intuition, see the soft clustering section for the minimal example within Results.
For a given cut S_m, let N_0^m, N_1^m and N_2^m be the number of individuals that carry the derived allele not at all, once or twice. As before, let A be the group of individuals that carry the derived allele at least once and $B = A^{\complement}$. Then, a first rough estimate of the frequency of the derived allele in A is given by

$$f_A^m = \frac{N_1^m + 2 \cdot N_2^m}{2 \cdot (N_1^m + N_2^m)}$$

and in B, based on the heterozygotes

$$f_B^m = \frac{N_1^m}{2 \cdot (N_0^m + N_1^m)}.$$

Hereby, f_A^m tends to overestimate and f_B^m tends to underestimate the true allele frequencies. To address this issue, we propose an iterative procedure to approximate the true allele frequencies. Initially, we start with the allele frequencies f_A^m, f_B^m. Remember that group A consists solely of individuals that have at least one derived allele and all individuals in group B are homozygous non-derived. Based on f_A^m and f_B^m we calculate the expected number of individuals that do not carry derived allele and should belong to group A and the expected number of individuals that are homozygous or heterozygous for the derived allele and should belong to group B:

$$\begin{aligned}
\mathbb{E}_{0,A}^m &= (1 - f_A^m)^2 \cdot \frac{N_1^m + N_2^m}{1 - (1 - f_A^m)^2}, \\
\mathbb{E}_{1,B}^m &= 2 \cdot f_B^m \cdot (1 - f_B^m) \cdot \frac{N_0^m}{(1 - f_B^m)^2}, \\
\mathbb{E}_{2,B}^m &= (f_B^m)^2 \cdot \frac{N_0^m}{(1 - f_B^m)^2}.
\end{aligned}$$

For the next iteration, we then update the estimated allele frequencies

$$\begin{aligned}
f_A^m &= \frac{N_1^m + 2 \cdot N_2^m}{2 \cdot (\mathbb{E}_{0,A}^m + N_1^m + N_2^m)}, \\
f_B^m &= \frac{\mathbb{E}_{1,B}^m + 2 \cdot \mathbb{E}_{2,B}^m}{2 \cdot (N_0^m + \mathbb{E}_{1,B}^m + \mathbb{E}_{2,B}^m)}.
\end{aligned}$$

This procedure is repeated until the estimated allele frequencies converge, which is usually achieved after $20 - 50$ iterations. The final reliability factor of cut S and individual v is then based on the expected number of misclassified individuals according to the estimated allele frequencies f_A^m and f_B^m:

$$r(S, v) := \begin{cases} \max\left\{1 - \frac{\min\{N_1, \mathbb{E}_{1,B}\} + \min\{N_2, \mathbb{E}_{2,B}\}}{N_1 + N_2}, 0\right\} & \text{if } v \text{ in } A, \\ \max\left\{1 - \frac{\mathbb{E}_{0,A}}{N_0}, 0\right\} & \text{if } v \text{ in } B. \end{cases} \tag{5}$$

With this, we can now calculate the soft clustering. Let the set $\{S \in \mathcal{S}(\tau) \mid v \in S^{(\text{right})}\}$ represent characterizing cuts oriented toward individual v on the right side of the tree. Then, we assign a probability $p_\tau^{(\text{right})}$ for belonging to the right subtree at a tangle τ to every individual v:

$$p_\tau^{(\text{right})}(v) = \frac{\sum_{S \in \{S \in \mathcal{S}(\tau) \mid v \in S^{(\text{right})}\}} r(S, v)}{\sum_{S \in \mathcal{S}(\tau)} r(S, v)}. \quad \textbf{(6)}$$

Based on these probabilities, we define the probability $p_k(v)$ that individual v arrives at node e as the product of the edge probabilities along the unique path from the root to τ. The soft clustering becomes clear by the minimal example and is explained in detail for this example in Results, see also Fig. 3 A. Note that the calculations do not account for a possible linkage disequilibrium (LD) between the SNPs. Pre-filtering SNPs based on LD, as in the Kidds AIMs panel, may therefore be beneficial. However, as SNPs in LD are expected to result in similar cuts, taking them into account could have a small effect on the soft clustering proportions, but does not affect the tangles tree and the resulting inferred hierarchy.

Computing the hard clustering. If desired, we can now assign individuals to a hard clustering: We assign each individual to the tangle with the highest probability based on the soft clustering.

Run time of tangleGen. In terms of run time, tangleGen and ADMIXTURE behave similarly on the Kidd's AIMs panel, both needing about 20 seconds on a laptop with an AMD Ryzen 7 octa-core processor and 32 GB RAM. As the number of SNPs increases, both tangleGen and ADMIXTURE become significantly slower. However, tangleGen has the advantage that pre-computed costs can be saved and re-used to shorten runtime. A more detailed run time analysis is added in Supplemental Fig. S7.

Data simulation. We employ the software msprime (Baumdicker et al. 2022) to simulate the data with an underlying demographic structure for Fig. 4 A and Fig. 5 A. In the simulation for Fig. 4 A, migration has been added between populations A and E, population sizes are constant over time. We generated a total of 800 recombining diploid individuals characterized by 5041 biallelic SNPs. In the simulation for Fig. 5 A, the migration rate is quadrupled and added between all populations A to H. Again, we generated a total of 800 recombining diploid individuals and used the same mutation and recombination rate as in the simulation for Fig. 4 A, resulting in 5078 biallelic SNPs. The simulation scripts are available on GitHub at k-burger/tangles_in_pop_gen.

Data access

The data used in this study is available at GitHub (https://github.com/k-burger/tangles_in_pop_gen) for download or simulation and is also provided as Supplemental Data. The code is also freely available on GitHub and is provided as Supplemental Code.

Competing interest statement

The authors declare no competing interests.

Acknowledgements

This work has been supported by the Cluster of Excellence "Controlling Microbes to Fight Infections" (DFG, EXC 2124, Project number 390838134) and the Cluster of Excellence "Machine Lerning: New Perspectives for Science" (DFG, EXC 2064/1, Project number 390727645) and the International Max Planck Research School for Intelligent Systems (IMPRS-IS). This work was supported with funding from the Reinhard Frank-Stiftung at the University of Tübingen.
Author contributions: Conceptualization by K.B., S.K., U.L., and F.B.; methodology by K.B., S.K., U.L., and F.B.; software by K.B. and S.K.; writing (original draft) by K.B. and S.K.; writing (review & editing) by K.B., S.K., U.L., and F.B.; supervision by U.L. and F.B.

References

1000 Genomes Project Consortium . 2015. A global reference for human genetic variation. *Nature* **526**:68.
Alexander DH, Novembre J, Lange K. 2009. Fast model-based estimation of ancestry in unrelated individuals. *Genome Research* **19**:1655–1664.
Baumdicker F, Bisschop G, Goldstein D, Gower G, Ragsdale AP, Tsambos G, Zhu S, Eldon B, Ellerman EC, Galloway JG, et al. 2022. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics* **220**. doi: 10.1093/genetics/iyab229.
Behr AA, Liu KZ, Liu-Fang G, Nakka P, Ramachandran S. 2016. Pong: fast analysis and visualization of latent clusters in population genetic data. *Bioinformatics* **32**:2817–2823.
Chiu AM, Molloy EK, Tan Z, Talwalkar A, Sankararaman S. 2022. Inferring population structure in biobank-scale genomic data. *The American Journal of Human Genetics* **109**:727–737.
Diestel R. 2018. Abstract separation systems. *Order* **35**:157–170.
Diestel R. 2019. Tangles in the social sciences. *arXiv preprint arXiv:1907.07341* .
Diestel R, Whittle G. 2023. Cluster analysis based on tangles in abstract separations systems. *US Patent No. 11,651,049* .
Dominguez Mantes A, Mas Montserrat D, Bustamante CD, Giró-i Nieto X, Ioannidis AG. 2023. Neural admixture for rapid genomic clustering. *Nature Computational Science* **3**:621–629.
Ingram CJ, Mulcare CA, Itan Y, Thomas MG, Swallow DM. 2009. Lactose digestion and the evolutionary genetics of lactase persistence. *Human Genetics* **124**:579–591.
Jakobsson M, Rosenberg NA. 2007. Clumpp: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. *Bioinformatics* **23**:1801–1806.

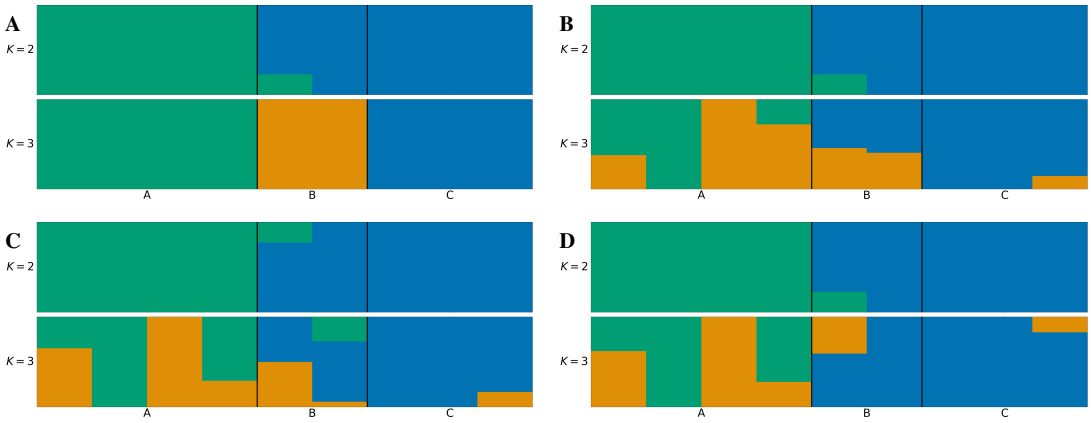
Jombart T, Devillard S, Balloux F. 2010. Discriminant analysis of principal components: a new method for the analysis of genetically structured populations. *BMC Genetics* **11**:1–15.
Kidd KK, Speed WC, Pakstis AJ, Furtado MR, Fang R, Madbouly A, Maiers M, Middha M, Friedlaender FR, Kidd JR, et al. 2014. Progress toward an efficient panel of snps for ancestry inference. *Forensic Science International: Genetics* **10**:23–32.
Klepper S, Elbracht C, Fioravanti D, Kneip J, Rendsburg L, Teegen M, von Luxburg U. 2023. Clustering with tangles: Algorithmic framework and theoretical guarantees. *Journal of Machine Learning Research* **24**:1–56.
Lawson DJ, Van Dorp L, Falush D. 2018. A tutorial on how not to over-interpret structure and admixture bar plots. *Nature Communications* **9**. doi: 10.1038/s41467-018-05257-7.
Mathieson I, Scally A. 2020. What is ancestry? *PLOS Genetics* **16**:1–6.
Mellars P. 2006. Going east: new genetic and archaeological perspectives on the modern human colonization of eurasia. *Science* **313**:796–800.
Padhukasahasram B. 2014. Inferring ancestry from population genomic data and its applications. *Frontiers in Genetics* **5**. doi: 10.3389/fgene.2014.00204.
Patterson N, Price AL, Reich D. 2006. Population structure and eigenanalysis. *PLOS Genetics* **2**:1–20.
Pearse DE, Crandall KA. 2004. Beyond f st: analysis of population genetic data for conservation. *Conservation Genetics* **5**:585–602.
Pfaffelhuber P, Grundner-Culemann F, Lipphardt V, Baumdicker F. 2020. How to choose sets of ancestry informative markers: A supervised feature selection approach. *Forensic Science International: Genetics* **46**. doi: 10.1016/j.fsigen.2020.102259.
Pfaffelhuber P, Sester-Huss E, Baumdicker F, Naue J, Lutz-Bonengel S, Staubach F. 2022. Inference of recent admixture using genotype data. *Forensic Science International: Genetics* **56**. doi: 10.1016/j.fsigen.2021.102593.
Pritchard JK, Stephens M, Donnelly P. 2000. Inference of population structure using multilocus genotype data. *Genetics* **155**:945–959.
Raj A, Stephens M, Pritchard JK. 2014. faststructure: variational inference of population structure in large snp data sets. *Genetics* **197**:573–589.
Randi E. 2008. Detecting hybridization between wild species and their domesticated relatives. *Molecular Ecology* **17**:285–293.
Reich D, Thangaraj K, Patterson N, Price AL, Singh L. 2009. Reconstructing indian population history. *Nature* **461**:489–494.
Riddell J, Basu Mallick C, Jacobs GS, Schoenebeck JJ, Headon DJ. 2020. Characterisation of a second gain of function edar variant, encoding edar380r, in east asia. *European Journal of Human Genetics* **28**:1694–1702.
Rotimi CN, Jorde LB. 2010. Ancestry and disease in the age of genomic medicine. *New England Journal of Medicine* **363**:1551–1558.
Sahi T. 1994. Genetics and epidemiology of adult-type hypolactasia. *Scandinavian Journal of Gastroenterology* **29**:7–20.
Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K. 2001. dbsnp: the ncbi database of genetic variation. *Nucleic Acids Research* **29**:308–311.
Tang H, Peng J, Wang P, Risch NJ. 2005. Estimation of individual admixture: analytical and study design considerations. *Genetic Epidemiology: The Official Publication of the International Genetic Epidemiology Society* **28**:289–301.
Wang J. 2022. Fast and accurate population admixture inference from genotype data from a few microsatellites to millions of snps. *Heredity* **129**:79–92.
Zhang S, Zhang R, Yuan K, Yang L, Liu C, Liu Y, Ni X, Xu S. 2024. Reconstructing complex admixture history using a hierarchical model. *Briefings in Bioinformatics* **25**. doi: 10.1093/bib/bbad540.

Supplemental material: Inferring ancestry with the hierarchical soft clustering approach tangleGen

Klara Elisabeth Burger[1], **Solveig Klepper**[1,2], **Ulrike von Luxburg**[1,2], **and Franz Baumdicker**[3,4,✉]

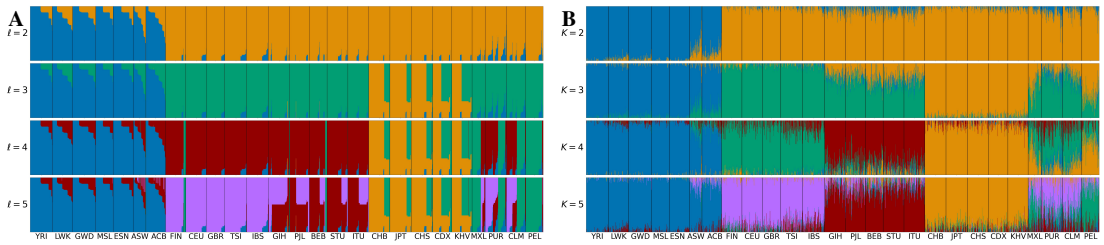
[1]Department of Computer Science, University of Tübingen, Germany
[2]Tübingen AI Center
[3]Cluster of Excellence "Controlling Microbes to Fight Infections", Mathematical and Computational Population Genetics, University of Tübingen, Germany
[4]Institute for Bioinformatics and Medical Informatics (IBMI), University of Tübingen, Germany

Supplemental Note 1: ADMIXTURE ancestry proportions for the minimal example



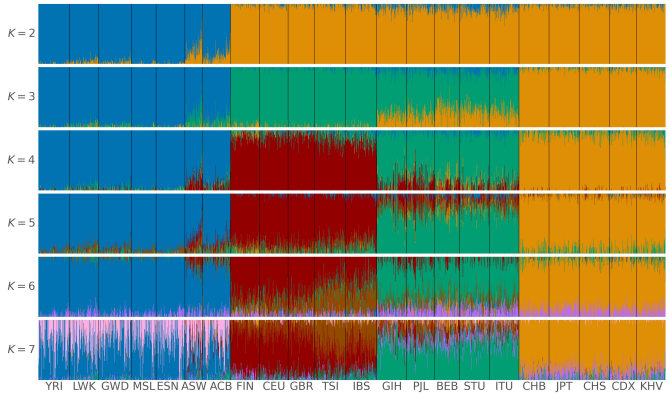
Supplemental Fig. S1. A: ADMIXTURE ancestry proportions on the minimal example from Results section. The best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. The input parameter K, which denotes the number of populations to be identified, is specified for each subplot. ADMIXTURE identifies the three populations for $K = 3$. For this minimal exmaple, ADMIXTURE states three modes, the result in A is obtained in five out of ten runs. tangleGen's soft clustering on the same data set is shown in Fig. 3 B. **B, C & D: ADMIXTURE with different initial initializations.** For such a small data set, ADMIXTURE is particularly dependent on the initial initialization leading to significantly different estimates as shown in plots B, C and D. The ancestry proportions in B are achieved in three out of ten runs and in C and D in one out of ten runs each.

Supplemental Note 2: Results on data from 1000 Genomes Project including Ad-Mixed American populations



Supplemental Fig. S2. tangleGen and ADMIXTURE on the AIMs panel by Kidd et al. (2014) for individuals sampled in the data from 1000 Genomes Project Consortium (2015) Phase 3, including Ad-Mixed American (AMR) populations. The individuals are sorted within the populations to achieve a block structure for tangleGen. The five superpopulations consist of the following populations: AFR: YRI, LWK, GWD, MSL, ESN, ASW, ACB; EUR: FIN, CEU, GBR, TSI, IBS; SAS: GIH, PJL, BEB, STU, ITU; EAS: CHB, JPT, CHS, CDX, KHV; AMR: MXL, PUR, CLM, PEL. **A: tangleGen soft clustering.** Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. tangleGen identifies the four superpopulations AFR, EUR, SAS and EAS, while AMR is composed of the other superpopulations. In Fig. 6, the EAS populations shared parts of their ancestry with the SAS populations and with the EUR populations. Here, no soft clustering between SAS and EAS is inferred, instead EAS now clusters partly with the Ad-mixed American populations. The first split in the tangles tree is supported by 9 characteristic SNPs, the second by 2, third by 1 and forth by 2. Parameters: Cost function as specified in Eq. 2, agreement parameter $a = 225$, $k = 40$. **B: ADMIXTURE ancestry proportions.** The best of ten ADMIXTURE runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. The input parameter K, which denotes the number of populations to be identified, is specified for each subplot. ADMIXTURE identifies the four superpopulations AFR, EUR, SAS and EAS and even partially assigns AMR as a separate population. ADMIXTURE continues to show inconsistencies: for $K = 3$, for example, we observe that the South Asian populations share a considerable part of their ancestry with the East Asian populations (plotted in orange). However, for $K = 4$, this component disappears almost completely.

Supplemental Note 3: ADMIXTURE on 1000 Genomes data set up to $\mathrm{K} = 7$



Supplemental Fig. S3. ADMIXTURE on Kidd's AIMs panel for individuals sampled in the 1000 Genomes Project Phase 3, excluding AMR populations. The four superpopulations are made up as follows: AFR: YRI, LWK, GWD, MSL, ESN, ASW, ACB; EUR: FIN, CEU, GBR, TSI, IBS; SAS: GIH, PJL, BEB, STU, ITU; EAS: CHB, JPT, CHS, CDX, KHV. The best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. The input parameter K, which denotes the number of populations to be identified, is specified for each subplot. To compare ADMIXTURE with tangleGen, only K values within the range considered by tangleGen are shown in Fig. 6 B in the main manuscript. This plot illustrates how ancestry proportions are estimated for larger K values, that is $K > 4$. From $K = 5$, no further individual populations or superpopulations are identified; instead, superpopulations are shown as admixed.

Supplemental Note 4: Characteristic SNPs for clustering of 1000 Genomes data set based on Kidd's AIMs panel

A further advantage of the hierarchical clustering method tangleGen is that the SNPs responsible for the splits in the tangles tree and thus for the inferred ancestry can be identified. These SNPs are referred to as characteristic SNPs. Listed below are the characteristic SNPs for clustering individuals in the 1000 Genomes Project Phase 3 based on Kidd's AIMs panel as described in the Results section, Fig. 6, ordered from lowest to highest cost. We follow dbSNP ID notation by Sherry et al. (2001).

- *top split:* rs2814778, rs1871534, rs10497191, rs3916235, rs4891825, rs7326934, rs7554936, rs11652805, rs1462906.
- *second split:* rs3827760, rs1800414, rs3811801.
- *last split:* rs1426654, rs16891982, rs12913832.

Supplemental Note 5: Cluster-typical set of SNPs for 1000 Genomes data set based on Kidd's AIMs panel

tangleGen can extract a cluster-typical set of SNPs, comprising relevant cuts for each cluster found with the tangles tree alongside their orientations, enhancing the explainability of the method. A cluster-typical set of SNPs consists of all SNPs that were used for clustering, including those that were not characteristic of a particular split in the tangles tree. A cluster-typical set of SNPs currently indicates the presence of the derived allele without distinguishing between homozygous and heterozygous states as it is based on the underlying cuts, which cannot make a finer distinction due to the current design of cuts. Note that the cluster-typical set of SNPs does not represent the genome of any specific individual, as consistencies during the construction of the tangles tree are only ensured for triplets of cuts. In addition, the cluster-typical set of SNPs does not consist of all SNPs provided as input and usually varies in length for each cluster.

SNP	AFR	EUR	SAS	EAS
rs2814778	derived	ancestral	ancestral	ancestral
rs1871534	derived	ancestral	ancestral	ancestral
rs3827760	ancestral	ancestral	ancestral	derived
rs10497191	ancestral	derived	derived	derived
rs3916235	ancestral	derived	derived	derived
rs1800414	ancestral	ancestral	ancestral	derived
rs4891825	ancestral	derived	derived	derived
rs3811801	ancestral	ancestral	ancestral	derived
rs1426654	derived	ancestral	derived	derived
rs7326934	derived	ancestral	ancestral	ancestral
rs7554936	ancestral	derived	derived	derived
rs11652805	ancestral	derived	derived	derived
rs16891982	ancestral	derived	ancestral	ancestral
rs1462906	ancestral	derived	derived	derived
rs1229984	derived	derived	derived	
rs12913832	ancestral	derived	ancestral	
rs1876482	ancestral	ancestral		
rs9522149	ancestral	derived		
rs310644	derived	ancestral		
rs3823159	derived	ancestral		
rs1834619	ancestral	ancestral		
rs798443	ancestral	derived		
rs7226659	ancestral	ancestral		
rs4918664	ancestral	ancestral		
rs2196051	derived			

Supplemental Table S1. Cluster-typical set of SNPs as a result of the clustering process by tangleGen for 1000 Genomes data set based on Kidd's AIMs panel. The resulting clusters can be associated with the superpopulations AFR, EUR, SAS, and EAS. In this context, "ancestral" denotes that the corresponding SNP is homozygous for the ancestral allele, while "derived" indicates that the corresponding SNP is homozygous or heterozygous for the derived allele. The characteristic SNPs within the tangles tree (Supplemental Note 4) are colored according to their corresponding split in the tangles tree, that is top split, second split and last split.

Supplemental Note 6: tangleGen on data with many SNPs and less differentiated populations

This section discusses what happens when tangleGen is faced with two challenges simultaneously: first, that the underlying populations are not well differentiated and second, that the data contains many SNPs. It is clear that such a scenario poses a challenge for any hierarchical method, as considering too early a SNP positioned high in the ancestral tree may introduce population structures that are not representative of the actual data. By considering many SNPs, this risk increases considerably. Both these challenges are met, for example, when applying tangleGen to chromosome 22 in the 1000 Genomes Project data set. To provide a better understanding, we first examine simulated data in a similar setting. Moreover, for this scenario, a cost function based on the divergence of the Hardy-Weinberg equilibrium distinguishes the populations more clearly and is therefore introduced below.

Cost function based on the divergence from Hardy-Weinberg equilibrium. Let us use the same notation as in the Methods section and consider a cut S_m that separates the individuals into subpopulations A and $B = A^\complement$, $T = A \cup B$, with N_A, N_B and N the group sizes. In the following, we present a cost function that is also specific to population genetics, but targets different aspects compared to the F_{ST}-based cost function presented in the Methods section. While the F_{ST}-based cost function favors cuts that divide individuals into as different groups A and B as possible, we now want to use a cost function that favors cuts that lead to two well-mixed groups A and B. To achieve this, we need to compare the expected heterozygosity with the observed heterozygosity in the groups. This evaluation metric is appropriate because these measures closely align in well-mixed subpopulations, that is a population in Hardy-Weinberg equilibrium. Depending on the application, a different cost function may be more suitable.

If the empirical frequency of the derived allele in group A at SNP S_m is given by

$$p_1^A = \frac{1}{2N_A} \sum_{n \in A} g_{nm}$$

then the expected heterozygosity in A is

$$2p_1^A\left(1-p_1^A\right)$$

if A is a well-mixed subpopulation. The number of observed heterozygotes in group A at SNP S_m is

$$x_{01}^A = \frac{1}{N_A}\sum_{n\in A}\mathbb{1}_{g_{nm}=1}$$

and therefore, the mean divergence from the Hardy-Weinberg equilibrium over all SNPs is given by

$$\bar{D}^A = \frac{1}{M}\sum_{m=1}^{M}|x_{01}^A - 2p_1^A(1-p_1^A)|.$$

Finally, the cost function is given by

$$c_{\text{HWE}}(S_m) := \frac{c_1^2\cdot c_2}{c_3}\cdot\left(1+\min\left(\bar{D}^A,\bar{D}^B\right)\right) \tag{1}$$

with

$$c_1 = 1+\left(\frac{\text{kNN\_A\_in\_B}}{N_A\cdot 40}+\frac{\text{kNN\_B\_in\_A}}{N_B\cdot 40}\right)$$

and

$$c_2 = \left(\frac{N}{N_A}+\frac{N}{N_B}\right)^{0.1}$$

as introduced in Eq. 3 and Eq. 4 in the main manuscript and

$$c_3 = \frac{1}{M}\sum_{m=1}^{M}|p_1^A - p_1^B|^{\frac{1}{2}}$$

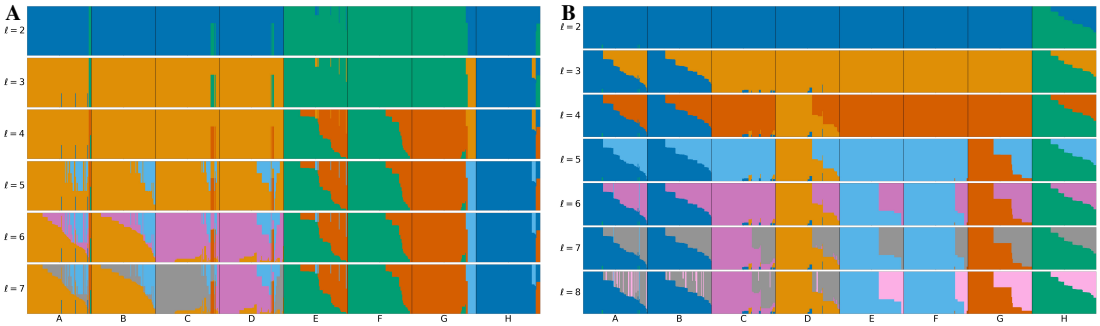
a penalization factor that accounts for the differences between A and B, not just the differences within the groups. This also ensures that random cuts through an overall population that is in Hardy-Weinberg equilibrium does not receive a very low cost leading to a prioritization by tangleGen.

Comparison with the Hardy-Weinberg-Equilibrium based reliability factor in the soft clustering step. When using this cost function, both the cost function and soft clustering are based on the Hardy-Weinberg equilibrium, which might suggest that they have the same effect. However, this is not the case. The soft clustering works with individual SNPs and indicates their reliability in population differentiation with respect to Hardy-Weinberg equilibrium. Conversely, the cost function evaluates each SNP/cut for its division into two groups based on all other SNPs and assigns a low cost if the two groups are significantly different and well-mixed. This is also evident in the minimal example, where the Hardy-Weinberg-based cost function assigns the same and lowest cost to both SNPs s_3 and s_6, while SNP s_5 has a higher cost. In contrast, only s_5 and s_6 are given a reliability factor of 1 by the soft clustering, as they are both homozygous for every individual.

Simulation. In the following simulation, we used the same simulation script with the same underlying demographic structure as previously described in Fig. 4 A. However, we reduced the time between population splits by a factor of 10 to obtain less differentiated populations. In addition, we increased the mutation rate by a factor of 20 to generate significantly more SNPs. After deleting SNPs with an allele frequency of less than 5% or higher than 95%, a total of 9420 SNPs are considered for the clustering of 800 individuals. The simulation script is available on GitHub at k-burger/tangles_in_pop_gen.

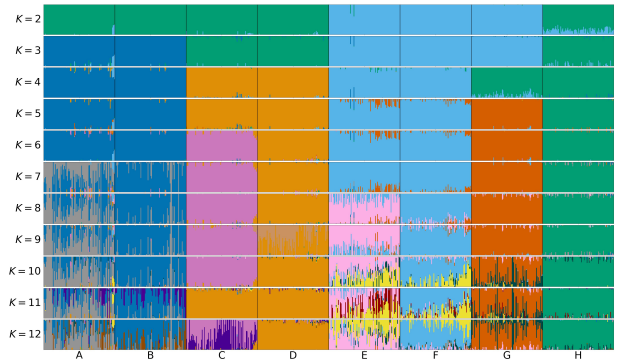
Comparison of two cost functions on simulated data with many SNPs and less differentiated populations. To compare the F_{ST}-based cost function introduced in the main manuscript with the cost function stated in in this Supplemental Note, we apply tangleGen twice to the simulated data with many SNPs and less differentiated populations from supplemental simulation section. Fig. S4 A shows that the F_{ST}-based cost function has difficulties in this scenario compared to Fig. 4 D, especially since it is not able to distinguish between the populations A and B as well as E and F, that is the populations that have split last in the underlying population structure (Fig. 4 A) and are therefore not well-differentiated. In contrast, tangleGen

with the Hardy-Weinberg-based cost function, Fig. S4 B, shows a clearer clustering compared to Fig. S4 A. However, this cost function does not reveal the underlying population structure when applied to the simulated data described in Fig. 4 A, as it tends to split the most differentiated individual populations first (like population H).



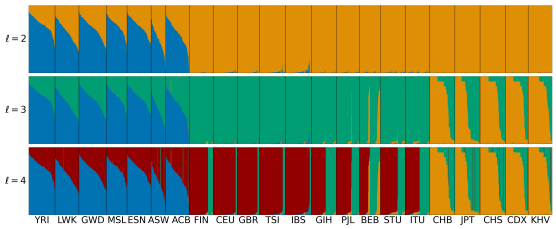
Supplemental Fig. S4. tangleGen on simulated data with many SNPs and less differentiated populations as described in the supplemental simulation section and two different cost functions. The individuals are sorted within the populations to achieve a block structure. Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. **A: tangleGen with F_{ST}-based cost function.** Used cost function as in Eq. 2, agreement parameter $\alpha = 40$, pruning is set to 2 and $k = 40$. tangleGen clusters the individuals hierarchically into seven populations. However, tangleGen does not distinguish between the most closely related populations A and B and E and F, the populations that split last in the underlying population structure (Fig. 4 A) and are therefore not well differentiated. Instead tangleGen identifies clusters that correspond rather to ancestral populations instead of extant populations. For example parts of population E and F cluster together with population G. Starting with the top split between the ancestral populations ABCDH and EFG, each population split is based on 15, 12, 26, 16, 25 and 3 characteristic SNPs. **B: tangleGen with Hardy-Weinberg-based cost function.** Used cost function as in Eq. 1, agreement parameter $\alpha = 40$, pruning is set to 2 and $k = 40$. tangleGen with this cost function is similar to A but results at level $\ell = 8$ in a clearer distinction and explicit separation of left and right subtree within the underlying population structure from Fig. 4, i.e. between the ancestral populations ABCD (pink cluster) and EFGH (gray cluster). Furthermore, the order of splits varied due to the different cost function. Starting with the top split between the ancestral populations ABCDEFG and H, each population split is based on 47, 11, 8, 7, 3, 5 and 4 characteristic SNPs.

In contrast to tangleGen, the scenario with less differentiated populations and more SNPs is much less of a challenge for ADMIXTURE, as Fig. S5 shows for $K = 8$. This outcome is not surprising as ADMIXTURE benefits from having many SNPs available and does not face the challenges of an hierarchical method. However, ADMIXTURE still performs significantly worse than in Fig. 4 and also displays inconsistencies with the underlying population structure.



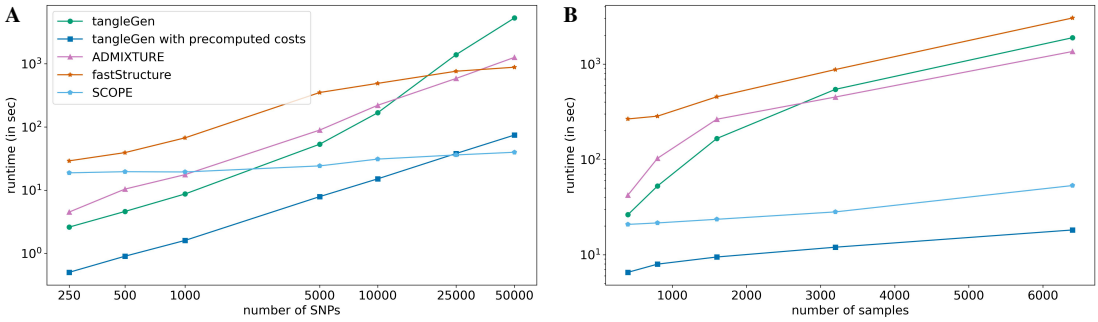
Supplemental Fig. S5. ADMIXTURE on simulated data with many SNPs and less differentiated populations as described in the supplemental simulation section. The best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. The individuals are sorted within the populations in the same way such that a block structure for the tangleGen in Fig. S4 is achieved. The input parameter K, which denotes the number of populations to be identified, is specified for each subplot. For $K = 8$ ADMIXTURE separates the eight populations comparably well to Fig. S4. But in this scenario the separation is less clear compared to the scenario in Fig. 4 and again shows inconsistencies for example between $K = 3$ and $K = 4$. For $K \geq 9$, the further inferred admixtures do and cannot represent meaningful parts of the underlying demography with eight populations.

Results on full 1000 Genomes data set. The application of tangleGen with the Hardy-Weinberg-based cost function on chromosome 22 (from the 1000 Genomes Project, Phase 3 data set) is shown in Fig. S6. At first glance, the clustering appears similar to Fig. 6 B, which is based on Kidd's AIMs panel. However, the differentiation of superpopulations is less pronounced, especially at the lowest level, when European and South Asian populations are distinguished. This observation is further reinforced when considering the number of SNPs responsible for each split in the tangles tree. While the Hardy-Weinberg-based cost function performs better than the F_{ST}-based one in this scenario, further fine-tuning of the cuts and cost functions might be able to address the inherent challenges of hierarchical methods in such scenarios.



Supplemental Fig. S6. tangleGen on chromosome 22 (1000 Genomes Project, Phase 3), AMR populations are excluded and SNPs in frequncy below 5% are ignored. The individuals are sorted within the populations to achieve a block structure for tangleGen. Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. Hardy-Weinberg-based cost function is used (as stated in Eq. 1), agreement parameter $\alpha = 200$, no pruning added and $k = 40$. tangleGen identifies the African and East Asian populations but has difficulties in differentiating European and South Asian populations. The first split is supported by 181 characteristic SNPs, the second by 37 and the third by only a single cut. If the pruning option in tangleGen is used, the last split would thus be omitted.

Supplemental Note 7: Runtime analysis



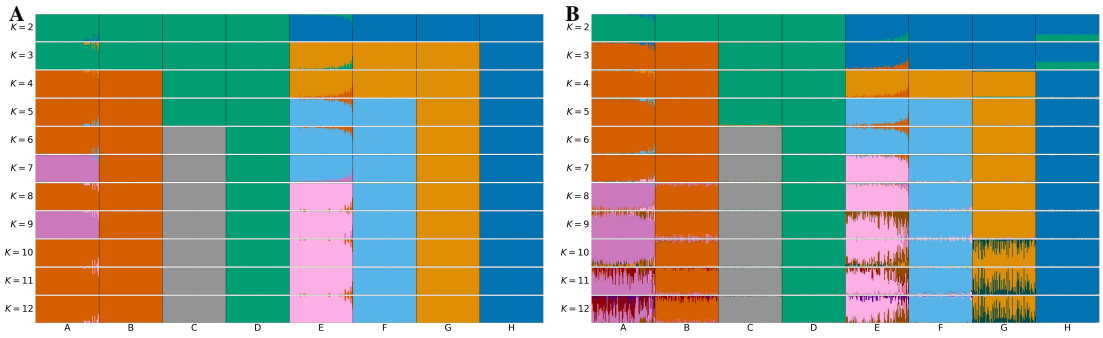
Supplemental Fig. S7. The runtime analysis compares tangleGen with ADMIXTURE, fastStructure (Raj et al. 2014) and SCOPE (Chiu et al. 2022) using simulated data as described in Fig. 4 A and the Data Simulation section in Methods. tangleGen uses the F_{ST}-based cost function with hyperparameters $k = 40$ and $b = 0.05$. The maximum input parameter K (eight in this analysis) for ADMIXTURE, fastStructure and SCOPE is determined by the maximum number of populations identified by tangleGen. It is important to note that determining this K value in practice requires time and careful consideration, which has not been factored in here. The calculations were performed on a laptop with an AMD Ryzen 7 Octa-Core processor and 32 GB RAM. **A: Number of SNPs.** We observe that tangleGen performs faster than ADMIXTURE and fastStructure on smaller data sets. However, as the size of the data set increases, tangleGen becomes significantly slower, which is due to the cost and tangles tree computation. As the cost calculation precedes the actual Tangles algorithm, it is a process that can be saved and reloaded for subsequent runs, for example when adjusting the agreement parameter. The calculation of the tangles tree is computationally challenging on large datasets, as for each SNP to be added, the consistency with all triplets of SNPs already added is checked for each branch. To circumvent this and improve scalability, one could subsample the triplets above a certain SNP count instead of considering all of them. This will improve the runtime without noticeably affecting performance and will be part of future research. SCOPE scales very well to larger data. Number of samples in this analysis: 800. **B: Number of samples.** Increasing the number of samples increases the runtime of all methods considered. As sample sizes grow, the cost calculation with tangleGen becomes increasingly time-consuming. However, with precalculated costs, tangleGen has the lowest runtime for all sample sizes considered. Number of SNPs in this analysis: 5041.

Supplemental Note 8: $\mathrm{F_{ST}}$ value analysis for Kidd's AIMs panel on 1000 Genomes data

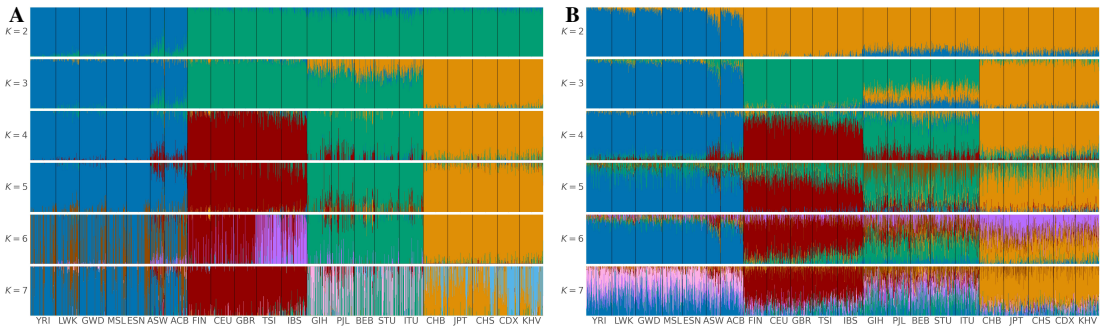
split-off superpopulation	SNP	F_{ST} value	characteristic SNP usage rank	cost
AFR	rs2814778	0.94	1	2.35
	rs1871534	0.87	2	2.38
	rs3916235	0.69	4	2.84
	rs7326934	0.65	6	3.17
	rs4891825	0.64	5	2.87
	rs3823159	0.62	-	4.14
	rs10497191	0.62	3	2.78
	rs1462906	0.58	9	3.53
	rs7657799	0.51	-	4.16
	rs7251928	0.50	-	4.64
	rs310644	0.48	-	4.13
	rs7554936	0.47	7	3.25
	rs2593595	0.45	-	5.69
	rs11652805	0.45	8	3.29
EAS	rs3827760	0.81	1	2.63
	rs1229984	0.58	-	3.85
	rs1800414	0.52	2	2.85
	rs1876482	0.45	-	4.05
	rs3811801	0.44	3	3.00
EUR	rs16891982	0.79	2	3.38
	rs12913832	0.45	3	3.92
	rs9522149	0.44	-	4.12
	rs1426654	0.43	1	3.07

Supplemental Table S2. F_{ST} value analysis for Kidd's AIMs panel on 1000 Genomes Data. This analysis considers three splits: AFR vs. EUR/SAS/EAS, EAS vs. AFR/EUR/SAS and EUR vs. AFR/SAS/EAS. For each split, only the SNPs with the highest F_{ST} values are listed, sorted by F_{ST} value. tangleGen identifies 9 characteristic SNPs for the first split and three each for the second and third split. For each SNP, it is indicated whether it was identified by tangleGen as a characteristic SNP and, if so, the usage rank of this SNP (SNP with rank 1 has the greatest impact, as it is considered first by the hierarchical method). The cost assigned by tangleGen is also given. Since tangleGen assigns each SNP a cost based on the mean F_{ST} value over all SNPs with respect to the division of individuals induced by the corresponding cut, there is a correlation between the characteristic SNPs and the SNPs with the highest F_{ST} values. In addition, the cost function takes k-nearest neighbors and the balance of bipartition sizes into account. Furthermore, the calculation of the true F_{ST} values is based on the classification into superpopulations, information that is not available for tangleGen. Some SNPs have a high F_{ST} value but are not characteristic SNPs. This is because they have a significantly higher cost and tangleGen composes the tangles tree by orienting cuts/SNPs iteratively beginning with the lowest-cost cut. The tangles tree is complete when no further cuts can be consistently added. From this moment any SNP with a higher cost is no longer considered. In a deeper tangles tree (for example with a smaller agreement parameter), some high F_{ST} SNPs might be considered characteristic (for example rs1229984 for the EAS split), but they would have a lower rank than the characteristic SNPs in this analysis.

Supplemental Note 9: fastStructure and SCOPE on simulated and 1000 Genomes data

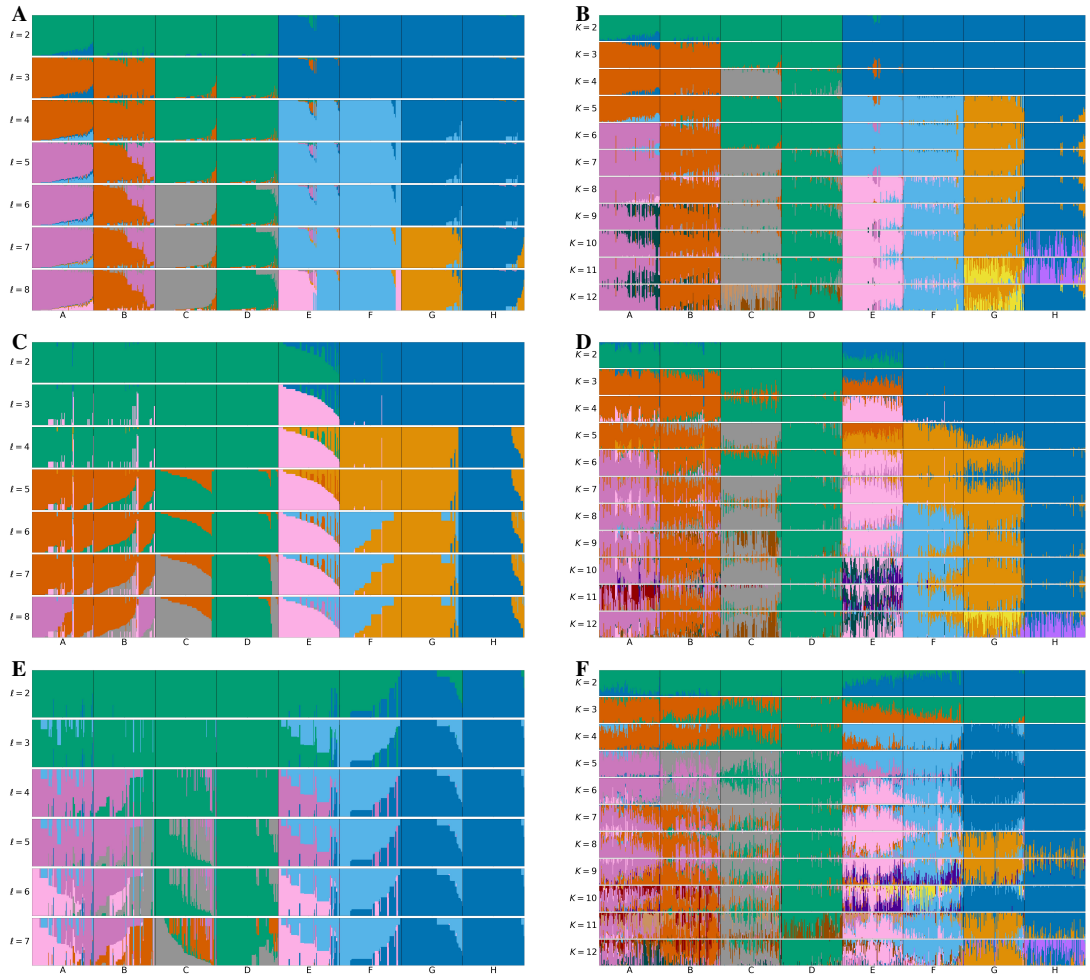


Supplemental Fig. S8. Inferred ancestries by fastStructure and SCOPE from simulated genetic data as in Fig. 4 A for different pre-specified numbers of populations, denoted by K. Individuals are sorted within the populations in the same order as in the tangleGen soft custer plot Fig. 4 D. The best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. **A: fastStructure.** fastStructure estimates the ancestry proportions well for $K \leq 8$ but has difficulties to seperate populations A and B. The number of identified populations does not match K for $K > 7$ since fastStructure can generate empty populations and the simulated data does not contain further population structure for $K > 8$. Furthermore, the results for different K show hierarchical inconsistencies, particularly for $K \geq 8$ in populations A and B. **B: SCOPE.** SCOPE estimates ancestry proportions well for the true number of populations ($K = 8$) and provides plausible results for other values of K. For $K > 8$, the simulated data does not contain additional population structure. The non-hierarchical nature of SCOPE is evident, for example, in population H between $K = 3$ and $K = 4$.



Supplemental Fig. S9. fastStructure and SCOPE on Kidd's AIMs panel for individuals sampled in the 1000 Genomes Project Phase 3, excluding AMR populations. The four superpopulations are made up as follows: AFR: YRI, LWK, GWD, MSL, ESN, ASW, ACB; EUR: FIN, CEU, GBR, TSI, IBS; SAS: GIH, PJL, BEB, STU, ITU; EAS: CHB, JPT, CHS, CDX, KHV. The best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. For each subplot the input parameter K, which denotes the pre-specified number of independent populations to be identified, is shown on the left side. fastStructure **(A)** and SCOPE **(B)** identify the four superpopulations AFR, EUR, SAS and EAS, when the number of populations is set to $K = 4$, but show inconsistencies, compared to smaller values of K in the differentiation between the four populations (for example between $K = 3$ and $K = 4$). Interestingly, fastStructure recognizes TSI and IBS as a separate population for $K = 6$ although Kidd's AIMs panel was not optimized for this. Beyond this, superpopulations are presented as admixed for $K > 4$.

Supplemental Note 10: tangleGen on simulated data with more migration



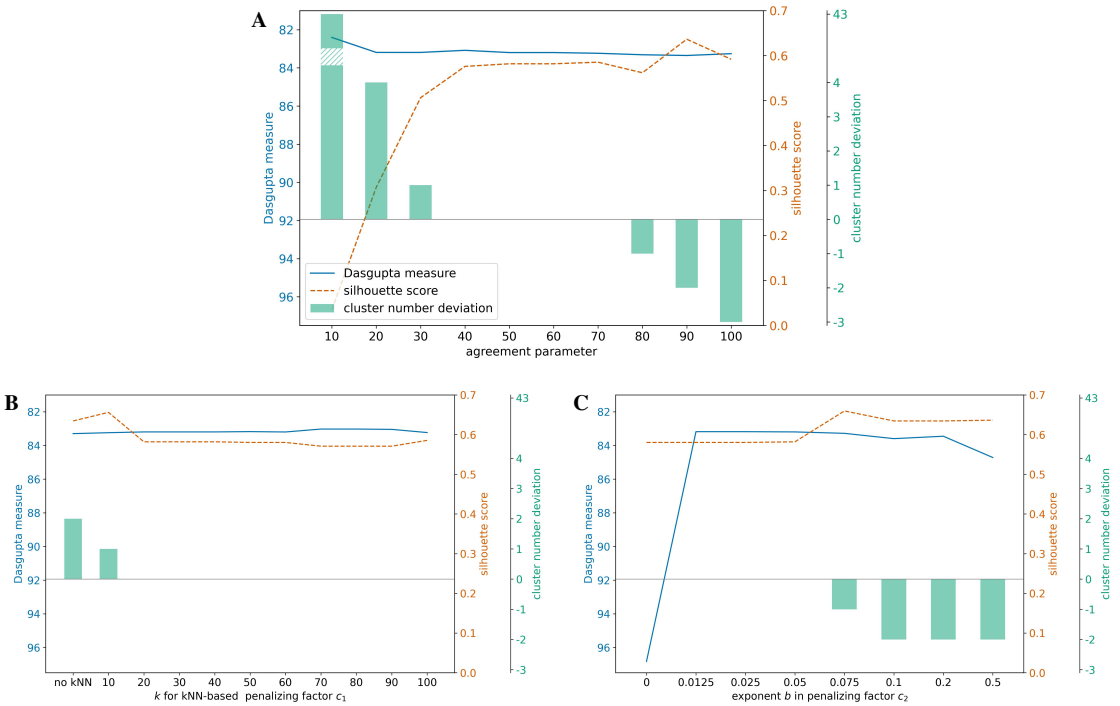
Supplemental Fig. S10. Comparison of tangleGen and ADMIXTURE on simulated data, as described in Fig. 5 A, but with varying migration rates between populations A to H. The first column (A, C, and E) shows tangleGen's soft clustering visualized in bar plots for different level ℓ in the tangles tree. The individuals are sorted within the populations based on the soft cluster proportions. For this analysis, the agreement parameter is set to $a = 30$ and all external branches supported by only one SNP are pruned. The second column (B, D, and F) shows ADMIXTURE's ancestry proportions for different pre-specified numbers of populations (K). Individuals are sorted within populations in the same order as in the corresponding tangleGen plot. In contrast to tangleGen, ADMIXTURE is not deterministic and the best of ten runs is shown in terms of clarity of clusters and minimization of inconsistencies compared to the underlying population structure. Migration rates increase from top to bottom: In **A** and **B**, the migration rate is as in Fig. 4 between populations A and E. In subplots **C** and **D**, the rate is quadrupled (as in Fig. 5); in **E** and **F** doubled again. **A, C** and **E** show that tangleGen infers meaningful population structures even when increased migration leads to more admixed populations. The hierarchical nature of tangleGen helps to clarify the migration pattern. For the highest migration rate (**E**) tangleGen cannot differentiate G and H, which is not surprising as these populations split early in the underlying population structure and thus experience the most migration. Furthermore, increased migration reduces the number of SNPs that separate populations well. Consequently, tangleGen's clustering is based on fewer SNPs. Starting with the top split between the ancestral populations each split in A is based on 94 (C:29, E:8), 100 (C:11, E:6), 14 (C:10, E:9), 14 (C:36, E:6), 16 (C:10, E:8), 17 (C:4, E:8) and 2 (C:3) characteristic SNPs. Subplots **B, D**, and **F** show ADMIXTURE's performance, estimating populations well for the true number of populations ($K = 8$). While the migration patterns are also visible with ADMIXTURE, they are not as clear as with tangleGen due to ADMIXTURE's non-hierarchical approach (A and C). ADMIXTURE results for different K show inconsistencies in each plot (**B, D**, and **F**). For $K > 8$, the simulated data does not contain any further population structure. Unlike ADMIXTURE, which requires choosing K, tangleGen determines the number of populations as a result of its hierarchical clustering in combination with the agreement parameter at $\ell = 8$ or $\ell = 7$.

Supplemental Note 11: Hyperparameters in tangleGen and their effects

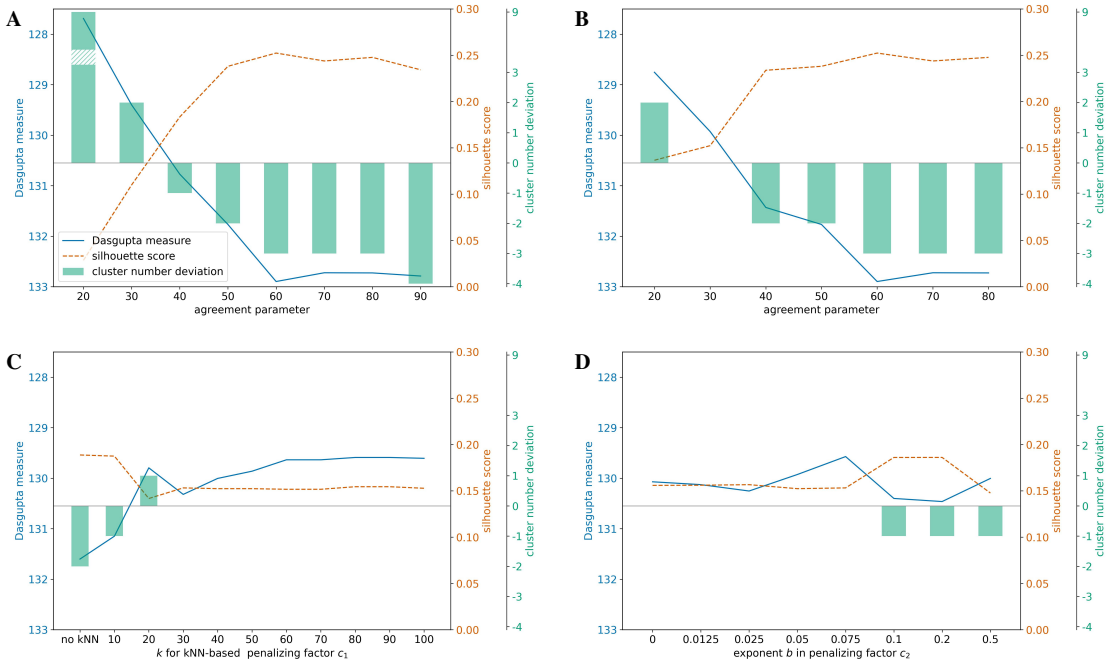
tangleGen uses several hyperparameters: the agreement parameter a, the pruning parameter and two additional parameters k and b in the penalization factors of the F_{ST}-based cost function. Here, k is related to the k-nearest neighbors (kNN) factor c_1, and b controls the preference for balanced cuts in c_2. To assess the impact of these hyperparameters on tangleGen's clustering results, we use a modified version of Dasgupta's hierarchical quality measure (Dasgupta)

$$D = \sum_{i,j} w(i,j) \cdot |C(i,j)|$$

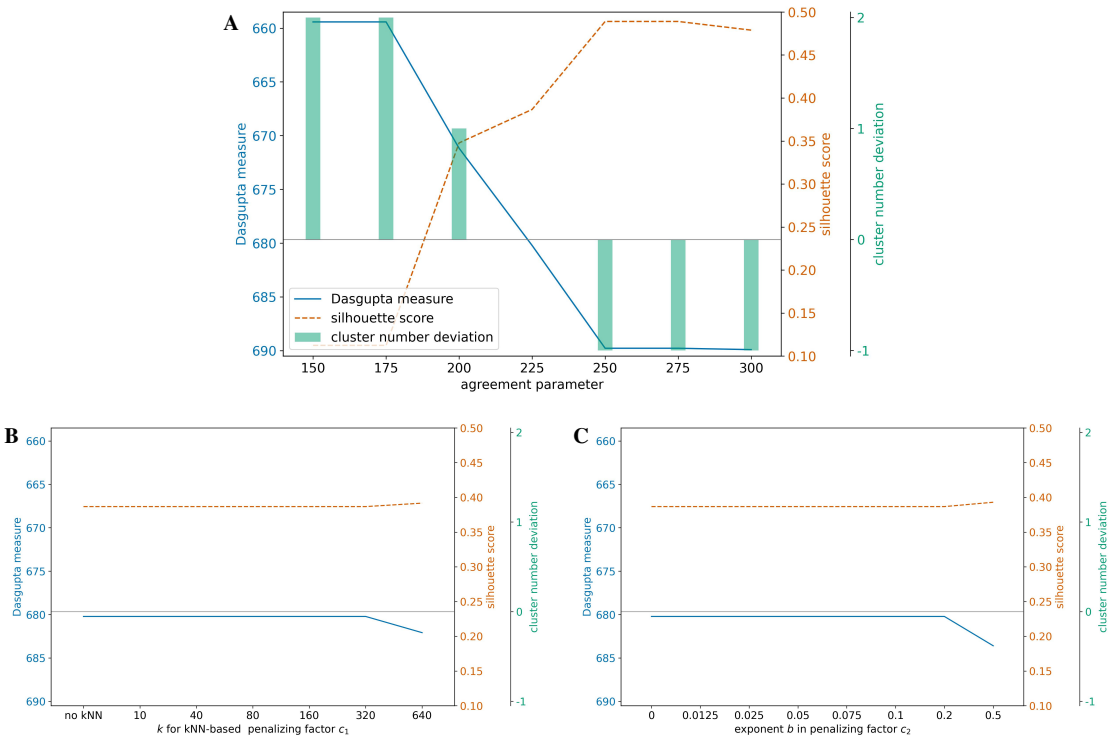
where $w(i,j)$ denotes the pairwise difference between individuals i and j and $|C(i,j)|$ represents the size of the smallest cluster containing both individuals. It is important to note that Dasgupta's measure evaluates hard clustering, where each individual is assigned exactly one cluster, unlike the soft clustering shown in the previous plots. However, the hard clustering can be computed from the soft clustering as discussed in the Methods section. Furthermore, Dasgupta's measure assumes clustering down to the individual level, which differs from tangleGen approach of clustering individuals into populations. Consequently, the measure does not detect overclustering, as smaller cluster sizes decrease D. This limitation is mitigated to some extent by setting $w(i,j) = 0$ for individuals sampled from the same population. To detect overclustering, we consider the silhouette score (Rousseeuw 1987), which effectively identifies small, random clusters but tends to overlook underclustering, a strength of Dasgupta's measure.



Supplemental Fig. S11. Hyperparameter effects on simulation, as in Fig. 4 A, with constant population sizes of 100. The following analysis is based on hard clustering, with pruning omitted since it only affects boundary values. **A: Agreement parameter.** tangleGen performs best with intermediate agreement parameters, yielding good values in both Dasgupta's measure and silhouette score. Low agreement parameters lead to significant overclustering, shown by cluster number deviation, and low silhouette scores. Since Dasgupta's measure is not sensitive to overclustering, it assigns a small error in such cases. Conversely, high agreement parameters cause underclustering, resulting in higher silhouette scores, as this measure tends to overlook underclustering. Dasgupta's measure shows slightly higher errors than for intermediate agreement parameters. These observations align with tangles' recommendations (Klepper et al. 2023). Other hyperparameters: $k = 40,\ b = 0.05$. **B: k for kNN-based penalizing factor c_1.** Adding a kNN-based penalizing factor to the cost function is beneficial, with the choice of k being relatively insensitive as long as it is not too small. Other hyperparameters: $a = 50,\ b = 0.05$. **C: Exponent b in penalizing factor c_2.** Adding a penalizing factor that prefers balanced cuts benefits the hierarchical method tangleGen. However, this parameter is sensitive; a too large b reduces cluster quality by assigning too low a cost to unfavorable clusters, as shown in Fig. S16. Other hyperparameters: $a = 50,\ k = 40$.

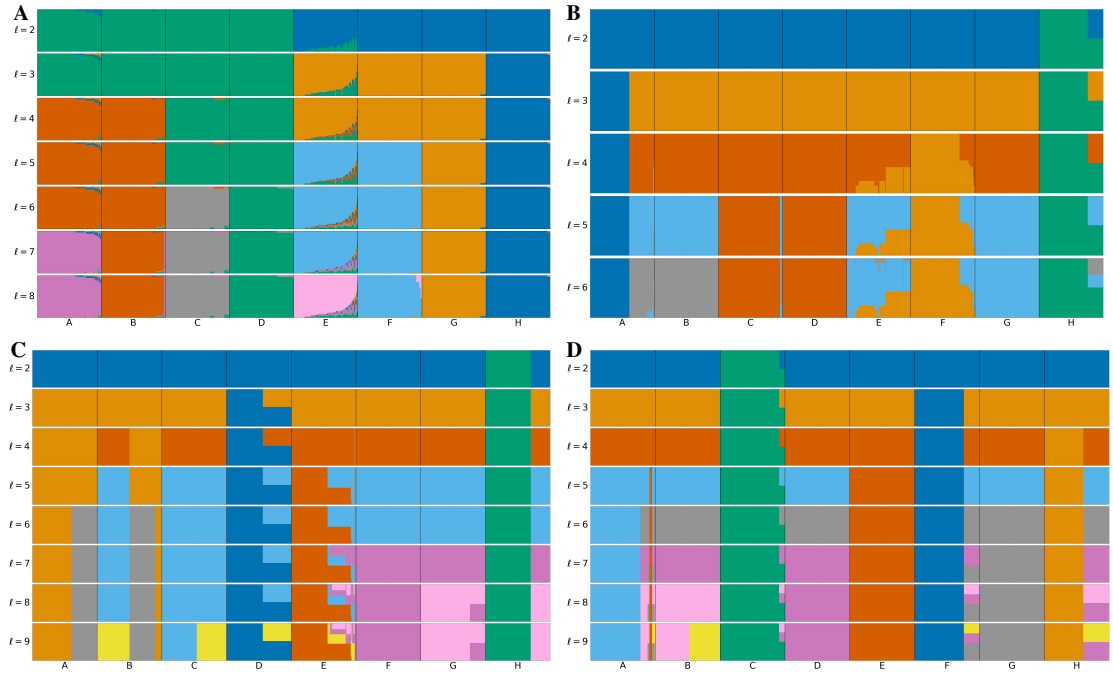


Supplemental Fig. S12. Hyperparameter effects on simulation with migration between populations A to H, as in Fig. 5 A, with constant population sizes of 100. The following analysis is based on hard clustering. **A: Agreement parameter (no pruning).** This complex simulation is more challenging for tangleGen than the scenario in Fig. S11 A. This difficulty is reflected in the generally higher values for the Dasgupta measure and the lower silhouette scores. The agreement parameter is particularly sensitive in this context. Unlike in Fig. S11 A, there is no section where both measures indicate a high quality. Other hyperparameters: $k = 40,\ b = 0.05$. **B: Agreement parameter (external branches of length two pruned).** With pruning clustering becomes more stable in this scenario. Other hyperparameters: $k = 40,\ b = 0.05$. **C: k for kNN-based penalizing factor c_1.** As observed in Fig. S11 B, adding a kNN-based penalty factor to the cost function is beneficial, and the choice of k is relatively insensitive as long as it is not too small. Other hyperparameters: $a = 30,\ b = 0.05$, external branches of length two pruned. **D: Exponent b in penalizing factor c_2.** Using a penalization factor that favors balanced cuts does not have as great an effect in this scenario as in Fig. S11 C. As before, if b is chosen too large, tangleGen's clustering is negatively affected. Other hyperparameters: $a = 30,\ k = 40$, external branches of length two pruned.

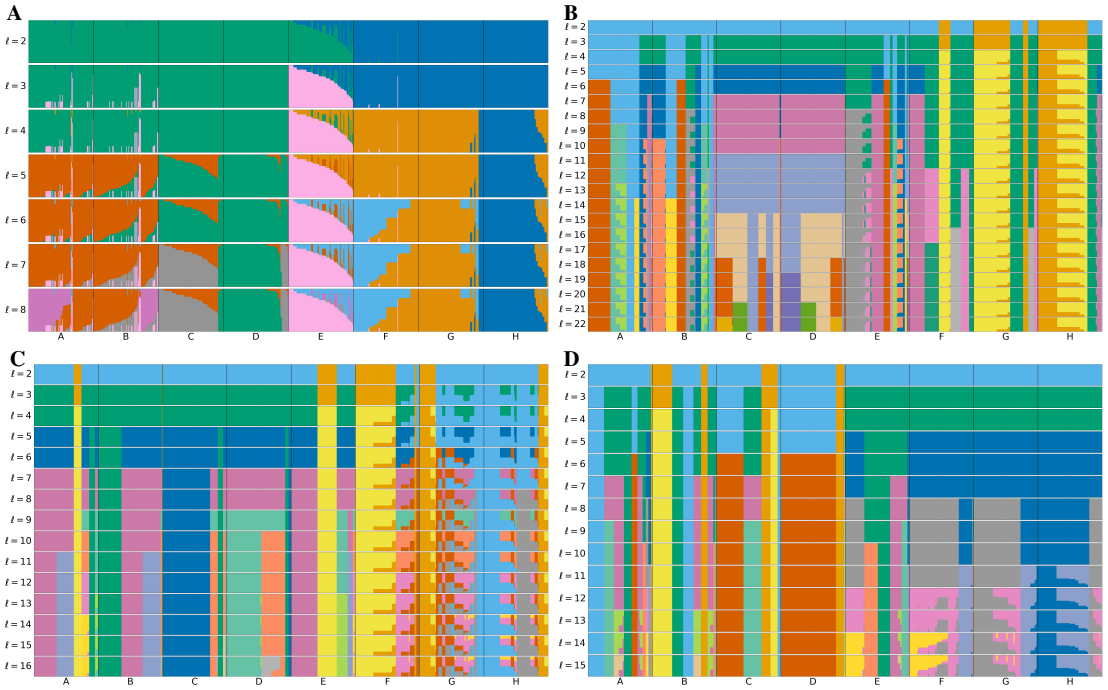


Supplemental Fig. S13. Hyperparameter effects on 1000 Genomes data (Fig. 6). The following analysis is based on hard clustering, with pruning disabled. **A: Agreement parameter.** The agreement parameter is a sensitive hyperparameter in this context. tangleGen's clustering shows overclustering when the parameter is set too low and underclustering when it is set too high, as indicated by the two quality measures and the cluster number deviation. tangleGen performs best with intermediate agreement parameters (225 corresponds to about 34% of the largest superpopulation (AFR) and 46% of the smallest superpopulation (SAS)), aligning with tangles' recommendations (Klepper et al. 2023). Other hyperparameters: $k = 40,\ b = 0.05$. **B & C: k for kNN-based penalizing factor c_1 and exponent b in penalizing factor c_2.** The penalization factors c_1 and c_2 are added to the cost function to enhance stability against cuts separating closely related individuals and to prevent the early onset of inconsistencies caused by unbalanced cuts. In this context, however, we consider Kidd's AIMs panel providing a stable data basis. As a result, c_1 and c_2 do not have a significant effect on this dataset as long as k and b are not chosen too large. Other hyperparameters: $a = 225,\ b = 0.05$ in A, $a = 225,\ k = 40$ in B.

Supplemental Note 12: Random cost function

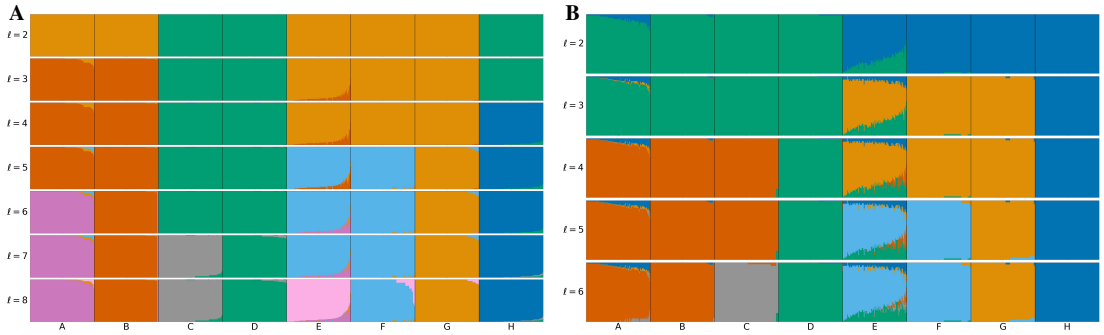


Supplemental Fig. S14. Random cost function on simulated data. The cost function has a significant impact on the performance of tangleGen. Based on the cost function, the cuts are sorted and iteratively oriented to compose the tangles tree. This analysis shows the effects of assigning random costs to each SNP/cut using the simulated data from Fig. 4 A. Panel **A** shows tangleGen with the F_{ST}-based cost function for ease of comparison. Panels **B**, **C**, and **D** show tangleGen with random cost assignments using different seeds. The individuals are sorted within the populations based on the soft cluster proportions to achieve a block structure in the plots. Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. The agreement parameter is set to $\alpha = 50$ in all panels. Without a meaningful cost function, both population splits and the order of splits are no longer consistent with the underlying population structure. In particular, tangleGen almost always splits off individual populations instead of recognizing larger clusters, such as ABCD vs. EFGH for $\ell = 2$. Moreover, the depth of the inferred hierarchy varies depending on the SNPs used first, and the inference is based on significantly fewer SNPs per level than before. Nevertheless, tangleGen succeeds in identifying some individual populations, for example population F in panel C and populations D, E, and G in panel D for $\ell = 9$. This is due to the clear differentiation between the populations in this simulation, leading to a significant amount of SNPs with high discriminatory power to which low costs can be randomly assigned. This is no longer the case for data with less differentiation, as shown in Fig. S15.



Supplemental Fig. S15. Random cost function on simulated data with migration. This analysis shows the effects of assigning random costs to each SNP/cut using the simulated data with migration added between populations A to H (Fig. 5 A). Panel **A** shows tangleGen with the F_{ST}-based cost function to facilitate comparison. Panels **B**, **C**, and **D** show tangleGen with random cost assignments using different seeds. The individuals are sorted within the populations based on the soft cluster proportions to achieve a block structure in the plots. Each subplot corresponds to a level ℓ in the tangles tree, where the different levels result from splits in the tangles tree. The agreement parameter is set to $\alpha = 30$ in all panels and all external branches supported by only one SNP are pruned. In this analysis, the populations are less well differentiated compared to Fig. S14, resulting in significantly fewer SNPs with discriminatory power. Without a reasonable cost function, both population splits and the order of splits are meaningless and do not reflect the underlying population structure or migration patterns. In contrast to Fig. S14, tangleGen now mostly subdivides individual populations. Additionally, the depth of the inferred hierarchy varies depending on the SNPs used first and has increased significantly. Furthermore, the inference is based on even fewer SNPs per level than in Fig. S14.

Supplemental Note 13: Effect of the penalizing factor c_2



Supplemental Fig. S16. The F_{ST}-based cost function contains the penalty factor c_2, which slightly favors cuts that divide individuals into equally sized groups. This preference is meaningful for a hierarchical method but might reduce sensitivity to underrepresented, isolated populations. The simulated scenario in Fig. 4 A is suitable for evaluating this question, as the left subtree is balanced while the right subtree is unbalanced. The data used for panels A and B is identical; only the penalizing factor c_2 differs. **A: The penalty factor c_2 is deactivated.** The top split is now based on only one SNP, which is usually penalized by c_2 to be considered at a later stage. Other splits in the right, unbalanced subtree remain unchanged. **B: Increased penalty factor c_2.** To examine the other extreme, we increased the hyperparameter b in the cost function, Eq. 2 in Methods, from $b = 0.05$ to $b = 0.5$. The root split works well, and H is still separated due to its high differentiation. However, ABC is separated at $\ell = 4$ instead of AB because separating ABC from DEFGH is more favorable than separating AB from CDEFGH. Furthermore, EF and AB can no longer be separated here. Panels A and B show that using the penalty factor c_2 is beneficial. The plots suggest that our choice of hyperparameter b is appropriate, which is further supported by the hyperparameter analysis in Supplemental Note 11. Additionally, tangleGen can effectively detect underrepresented populations (such as H) as long as there are SNPs distinguishing these populations from the rest. If no such SNPs exist, detection would likely be challenging even without the c_2.

References

1000 Genomes Project Consortium . 2015. A global reference for human genetic variation. *Nature* **526**:68.

Chiu AM, Molloy EK, Tan Z, Talwalkar A, Sankararaman S. 2022. Inferring population structure in biobank-scale genomic data. *The American Journal of Human Genetics* **109**:727–737.

Dasgupta S. A cost function for similarity-based hierarchical clustering. *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. doi: 10.1145/2897518.2897527.

Kidd KK, Speed WC, Pakstis AJ, Furtado MR, Fang R, Madbouly A, Maiers M, Middha M, Friedlaender FR, Kidd JR, et al. 2014. Progress toward an efficient panel of snps for ancestry inference. *Forensic Science International: Genetics* **10**:23–32.

Klepper S, Elbracht C, Fioravanti D, Kneip J, Rendsburg L, Teegen M, von Luxburg U. 2023. Clustering with tangles: Algorithmic framework and theoretical guarantees. *Journal of Machine Learning Research* **24**:1–56.

Raj A, Stephens M, Pritchard JK. 2014. faststructure: variational inference of population structure in large snp data sets. *Genetics* **197**:573–589.

Rousseeuw PJ. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* **20**:53–65.

Sherry ST, Ward MH, Kholodov M, Baker J, Phan L, Smigielski EM, Sirotkin K. 2001. dbsnp: the ncbi database of genetic variation. *Nucleic Acids Research* **29**:308–311.

Chapter 3

Conclusion and Outlook

3.1 Conclusion

Machine learning is already widely applied in population genetics and, combined with the ever-growing availability of data, offers enormous potential to significantly advance our understanding of genetic variability and evolution. However, challenges remain, with two of the most crucial being the interpretability and robustness of these methods [Huang et al., 2024; Korfmann et al., 2023]. A promising strategy to address these challenges is to leverage the rich theoretical insights of population genetics within machine learning methods.

In this thesis, we present two novel approaches to machine learning in population genetics that incorporate established population genetic concepts and methods: a supervised method in the form of an adaptive neural network for mutation rate estimation and an unsupervised hierarchical clustering method for inferring population ancestry. With the adaptive neural network, we present a concept of how well-established model-based estimators can be integrated into a loss function. When trained with such an adaptive loss function, the network requires only one hidden layer to obtain a single estimator that performs nearly as well as the model-based estimators for low and high recombination rates, while providing a superior estimation method for intermediate recombination rates. The approach is also validated by comparison with model-based estimators, as the neural network consistently mirrors the behavior of the established model-based estimator in every scenario where such an estimator exists.

The method for inferring population ancestry, tangleGen, is a hierarchical soft clustering method that incorporates fundamental population genetics concepts, such as the fixation index and the Hardy-Weinberg equilibrium, into its cost function. tangleGen is robust, providing deterministic and consistent results across different numbers of populations. Interpretability is greatly enhanced by the hierarchical nature of the method, which provides insight into ancestral relationships, and by the ability to identify the SNPs responsible for each clustering decision. In addition, unlike many other methods, tangleGen automatically determines the number of populations as a result of its clustering process.

In summary, the methods presented in this thesis demonstrate the advantages of incorporating domain-specific knowledge from population genetics into machine learning approaches. The integration of theoretical concepts and models with data-driven learning enables efficient processing of large data sets, effectively capturing and exploiting underlying data patterns, while improving robustness and interpretability.

3.2 Outlook

Machine learning methods specifically tailored to genetic data structures are expected to offer significant advantages. A unique feature of genetic data is that each sample is based on a series of genealogical trees that encode its evolutionary history. Consequently, estimators that infer and exploit information about the unknown underlying tree sequence are particularly powerful. This leads to a key question for future research in the application of machine learning to population genetics: Can we develop machine learning methods that infer and exploit the underlying tree sequence to further enhance performance?

In model-based estimators, knowledge of the underlying tree sequence often enables the computation of optimal estimators, such as the unique uniformly minimum variance unbiased estimator for mutation rate estimation. To explore how machine learning methods can benefit from such additional information, we can reconsider the adaptive neural network introduced in the first publication and provide it with information about the underlying tree sequence. For this minimal example, we assume no recombination, resulting in a single underlying tree rather than a sequence of trees, and obtain basic information about the underlying tree topology from the SFS, as shown in Fig. 1.2. A more balanced tree will force more SFS entries to be zero, and therefore the number of non-zero SFS entries can be used as a measure of the imbalance of the tree.

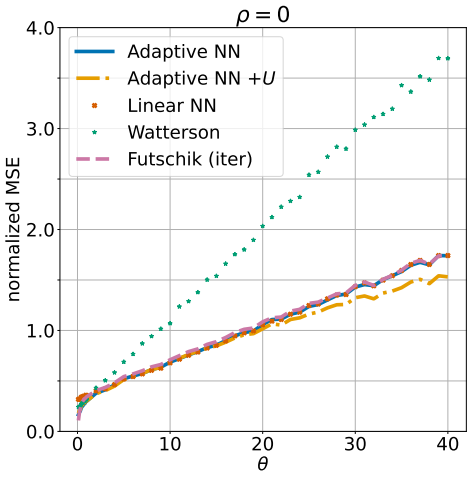


Figure 3.1: The normalized MSE for five different estimators are shown: The iterated minimal MSE estimator (Futschik (iter)), Watterson's estimator, and three neural network estimators: 1. An adaptive network with one hidden layer and 200 hidden nodes, 2. an adaptive network with one hidden layer and 200 hidden nodes and the additional input node U and 3. a linear network without a hidden layer. We used *msprime* to generate a total of $4.9 \cdot 10^5$ independent simulations with sample size $n = 40$, no recombination ($\rho = 0$), and 49 different mutation rates $\theta \in (0, 40]$. All shown neural networks have been trained on a data set without recombination. The adaptive neural network $+U$ outperforms the other estimators by exploiting information about the underlying tree topology through the additional input U.

By incorporating this information into the adaptive neural network through an additional input node U, defined as

$$U = \sum \mathbb{1}_{S_i > 0},$$

the network's performance is significantly improved, see Fig. 3.1. The adaptive network $+U$ now even outperforms the model-based estimator minimizing the mean squared error, Futschik (iter), which lacks any information about the tree topology. This improvement is particularly evident at higher mutation rates, where zero entries in the SFS increasingly indicate a balanced tree topology. The improved performance is reasonable. For example, a perfectly unbalanced tree has a long branch leading from the root to a single individual (see Fig. 1.2), which is likely to carry many mutations. This significantly increases the SFS entry S_1. Without information about the tree topology, an estimator would likely overestimate the mutation rate. However, by assessing the imbalance of the tree, the adaptive neural network $+U$ can adjust the weight of S_1 accordingly to counteract this effect.

While knowledge of the underlying tree sequence is clearly a major advantage, it is typically unknown. Thus, the development of methods that infer and exploit the underlying tree sequence is a crucial and exciting next step in combining data-driven learning with population genetic theory. Such advances are expected to open up new possibilities and deepen our understanding of genetic variability and evolution.

Bibliography

E. Alpaydin. *Introduction to machine learning*. MIT Press, 2020.

F. Baumdicker, G. Bisschop, D. Goldstein, G. Gower, A. P. Ragsdale, G. Tsambos, S. Zhu, B. Eldon, E. C. Ellerman, J. G. Galloway, and et al. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*, 220(3), 2022. doi: 10.1093/genetics/iyab229.

N. Berestycki. Recent progress in coalescent theory. *arXiv preprint arXiv:0909.3985*, 2009.

C. M. Bishop. Pattern recognition and machine learning. *Springer*, 2:1122–1128, 2006.

S. Boitard, C. Schlotterer, and A. Futschik. Detecting selective sweeps: a new approach based on hidden markov models. *Genetics*, 181(4):1567–1578, 2009.

K. E. Burger, P. Pfaffelhuber, and F. Baumdicker. Neural networks for self-adjusting mutation rate estimation when the recombination rate is unknown. *PLOS Computational Biology*, 18(8):1–17, 2022. doi: 10.1371/journal.pcbi.1010407.

K. E. Burger, S. Klepper, U. von Luxburg, and F. Baumdicker. Inferring ancestry with the hierarchical soft clustering approach tangleGen. *Genome Research*, 34 (12):2244–2255, 2024. doi: 10.1101/gr.279399.124.

J. Chan, V. Perrone, J. Spence, P. Jenkins, S. Mathieson, and Y. Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

G. Coop. *Population and quantitative genetics*. 2020.

J.-M. Cornuet, S. Aulagnier, S. Lek, S. Franck, and M. Solignac. Classifying individuals among infra-specific taxa using microsatellite data and neural networks. *Comptes Rendus de L'academie des sciences. Serie III, Sciences de la vie*, 319 (12):1167–1177, 1996.

C. Darwin. *Notebook B*. Cambridge University Library, 1837.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, 2019.

T. Dobzhansky. Genetics and the origin of species. 1951.

M. S. Dodd, D. Papineau, T. Grenne, J. F. Slack, M. Rittner, F. Pirajno, J. O'Neil, and C. T. Little. Evidence for early life in earth's oldest hydrothermal vent precipitates. *Nature*, 543(7643):60–64, 2017.

A. Dominguez Mantes, D. Mas Montserrat, C. D. Bustamante, X. Giró-i Nieto, and A. G. Ioannidis. Neural admixture for rapid genomic clustering. *Nature Computational Science*, 3(7):621–629, 2023.

R. Durrett. *Probability Models for DNA Sequence Evolution*. Probability and Its Applications. Springer New York, 2 edition, 2008.

A. Etheridge. *Some Mathematical Models from Population Genetics*. Springer-Verlag Berlin Heidelberg, 2012.

W. J. Ewens. *Mathematical Population Genetics*. Springer New York, 2 edition, 2004.

L. Flagel, Y. Brandvain, and D. R. Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular Biology and Evolution*, 36(2):220–238, 2019.

G. E. Folk and H. A. Semken. The evolution of sweat glands. *International Journal of Biometeorology*, 35:180–186, 1991.

Y.-X. Fu and W.-H. Li. Coalescing into the 21st century: an overview and prospects of coalescent theory. *Theoretical Population Biology*, 56(1):1–10, 1999.

J. H. Gillespie. *Population Genetics: A Concise Guide*. Johns Hopkins University Press, 1998.

I. Goodfellow. *Deep learning*, volume 196. MIT Press, 2016.

G. H. Hardy. Mendelian proportions in a mixed population. *Science*, 28(706): 49–50, 1908.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

X. Huang, A. Rymbekova, O. Dolgova, O. Lao, and M. Kuhlwilm. Harnessing deep learning for population genetic inference. *Nature Reviews Genetics*, 25(1): 61–78, 2024.

R. R. Hudson. Generating samples under a wright–fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.

R. R. Hudson et al. Gene genealogies and the coalescent process. *Oxford Surveys in Evolutionary Biology*, 7(1):44, 1990.

IPCC. *Climate Change 2022 – Impacts, Adaptation and Vulnerability: Working Group II Contribution to the Sixth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, 2023.

T. Jombart, S. Devillard, and F. Balloux. Discriminant analysis of principal components: a new method for the analysis of genetically structured populations. *BMC Genetics*, 11(1):1–15, 2010.

M. I. Jordan and T. M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

A. D. Kern and D. Haussler. A population genetic hidden markov model for detecting genomic regions under selection. *Molecular Biology and Evolution*, 27 (7):1673–1685, 2010.

M. Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):893, 1969.

J. F. Kingman. On the genealogy of large populations. *Journal of Applied Probability*, 19:27–43, 1982.

S. Klepper, C. Elbracht, D. Fioravanti, J. Kneip, L. Rendsburg, M. Teegen, and U. von Luxburg. Clustering with tangles: Algorithmic framework and theoretical guarantees. *Journal of Machine Learning Research*, 24(190):1–56, 2023.

D. C. Koboldt, K. M. Steinberg, D. E. Larson, R. K. Wilson, and E. R. Mardis. The next-generation sequencing revolution and its impact on genomics. *Cell*, 155(1):27–38, 2013.

K. Korfmann, T. P. P. Sellinger, F. Freund, M. Fumagalli, and A. Tellier. Simultaneous inference of past demography and selection from the ancestral recombination graph under the beta coalescent. *Peer Community Journal*, 4. doi: 10.24072/pcjournal.397.

K. Korfmann, O. E. Gaggiotti, and M. Fumagalli. Deep learning in population genetics. *Genome Biology and Evolution*, 15(2), 2023. doi: 10.1093/gbe/evad008.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

K. Lin, H. Li, C. Schlotterer, and A. Futschik. Distinguishing positive selection from neutral evolution: boosting the performance of summary statistics. *Genetics*, 187(1):229–244, 2011.

J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

T. Mailund, J. Y. Dutheil, A. Hobolth, G. Lunter, and M. H. Schierup. Estimating divergence time and ancestral effective population size of bornean and sumatran orangutan subspecies using a coalescent hidden markov model. *PLOS Genetics*, 7(3):1–15, 2011. doi: 10.1371/journal.pgen.1001319.

Z. Mo and A. Siepel. Domain-adaptive neural networks improve supervised machine learning based on simulated population genetic data. *PLOS Genetics*, 19(11):1–22, 2023. doi: 10.1371/journal.pgen.1011032.

M. P. Muehlenbein. *Human Evolutionary Biology*. Cambridge University Press, 2010.

E. Pardoux. *Models of Sequences and Discrete Traits Evolution*, chapter 2, pages 27–38. John Wiley and Sons, Ltd, 2024. doi: https://doi.org/10.1002/9781394284252.ch2.

N. Patterson, A. L. Price, and D. Reich. Population structure and eigenanalysis. *PLOS Genetics*, 2(12):1–20, 2006. doi: 10.1371/journal.pgen.0020190.

A. L. Price, N. J. Patterson, R. M. Plenge, M. E. Weinblatt, N. A. Shadick, and D. Reich. Principal components analysis corrects for stratification in genome-wide association studies. *Nature Genetics*, 38(8):904–909, 2006.

M. Pybus, P. Luisi, G. M. Dall'Olio, M. Uzkudun, H. Laayouni, J. Bertranpetit, and J. Engelken. Hierarchical boosting: a machine-learning framework to detect and classify hard selective sweeps in human populations. *Bioinformatics*, 31 (24):3946–3952, 2015.

R. Ronen, N. Udpa, E. Halperin, and V. Bafna. Learning natural selection from the site frequency spectrum. *Genetics*, 195(1):181–193, 2013.

T. Sanchez, J. Cury, G. Charpiat, and F. Jay. Deep learning for population size history inference: Design, comparison and combination with approximate bayesian computation. *Molecular Ecology Resources*, 21(8):2645–2660, 2021.

T. Sanchez, E. M. Bray, P. Jobic, J. Guez, A.-C. Letournel, G. Charpiat, J. Cury, and F. Jay. dnadna: a deep learning framework for population genetics inference. *Bioinformatics*, 39(1), 2022. doi: 10.1093/bioinformatics/btac765.

A. Scally and R. Durbin. Revising the human mutation rate: implications for understanding human evolution. *Nature Reviews Genetics*, 13(10):745–753, 2012.

F. Schrenk, T. G. Bromage, C. G. Betzler, U. Ring, and Y. M. Juwayeyi. Oldest Homo and Pliocene biogeography of the Malawi Rift. *Nature*, 365(6449):833–836, 1993.

D. R. Schrider and A. D. Kern. Inferring selective constraint from population genomic data suggests recent regulatory turnover in the human brain. *Genome Biology and Evolution*, 7(12):3511–3528, 2015.

S. Sheehan and Y. S. Song. Deep learning for population genetic inference. *PLOS Computational Biology*, 12(3):1–28, 2016. doi: 10.1371/journal.pcbi.1004845.

G. J. Siegel, B. W. Agranoff, R. W. Albers, S. K. Fisher, and M. D. Uhler. *Basic Neurochemistry: Molecular, Cellular, and Medical Aspects*. Lippincott Williams & Wilkins, 1999.

C. C. Smith and A. D. Kern. dispersenn2: a neural network for estimating dispersal distance from georeferenced polymorphism data. *BMC Bioinformatics*, 24(1), 2023. doi: 10.1186/s12859-023-05522-7.

L. Torada, L. Lorenzon, A. Beddis, U. Isildak, L. Pattini, S. Mathieson, and M. Fumagalli. Imagene: a convolutional neural network to quantify natural selection from genomic data. *BMC Bioinformatics*, 20:337, 2019. doi: 10.1186/s12859-019-2927-x.

J. H. Wakeley. *Coalescent Theory*. Macmillan Learning, 2009.

W. Weinberg. Über den Nachweis der Vererbung beim Menschen. *Jh. Ver. vaterl. Naturk. Wurttemb.*, 64:369–382, 1908.

M. C. Weiss, F. L. Sousa, N. Mrnjavac, S. Neukirchen, M. Roettger, S. Nelson-Sathi, and W. F. Martin. The physiology and habitat of the last universal common ancestor. *Nature Microbiology*, 1(9):1–8, 2016.

S. Wright. Evolution in mendelian populations. *Genetics*, 16:97–159, 03 1931.

S. Wright. Isolation by distance. *Genetics*, 28(2):114–138, 1943.

S. Wright. The genetical structure of populations. *Annals of eugenics*, 15(1): 323–354, 1949.

D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2:165–193, 2015.