Tractable and Expressive Generative Models of Genetic Variation Data

Meihua Dang[1], Anji Liu[1], Xinzhu Wei[1], Sriram Sankararaman[⋆1,2,3], and Guy Van den Broeck[⋆1]

[1] Department of Computer Science, UCLA, Los Angeles, USA
{mhdang,liuanji,aprilwei,sriram,guyvdb}@cs.ucla.edu
[2] Department of Human Genetics, School of Medicine, UCLA, Los Angeles, USA
[3] Department of Computational Medicine, School of Medicine, UCLA, Los Angeles, USA

Abstract. Population genetic studies often rely on artificial genomes (AGs) simulated by generative models of genetic data. In recent years, unsupervised learning models, based on hidden Markov models, deep generative adversarial networks, restricted Boltzmann machines, and variational autoencoders, have gained popularity due to their ability to generate AGs closely resembling empirical data. These models, however, present a tradeoff between expressivity and tractability. Here, we propose to use hidden Chow-Liu trees (HCLTs) and their representation as probabilistic circuits (PCs) as a solution to this tradeoff. We first learn an HCLT structure that captures the long-range dependencies among SNPs in the training data set. We then convert the HCLT to its equivalent PC as a means of supporting tractable and efficient probabilistic inference. The parameters in these PCs are inferred with an expectation-maximization algorithm using the training data. Compared to other models for generating AGs, HCLT obtains the largest log-likelihood on test genomes across SNPs chosen across the genome and from a contiguous genomic region. Moreover, the AGs generated by HCLT more accurately resemble the source data set in their patterns of allele frequencies, linkage disequilibrium, pairwise haplotype distances, and population structure. This work not only presents a new and robust AG simulator but also manifests the potential of PCs in population genetics.

1 Introduction

Generative models of genetic sequence data play a central role in population genomics [1, 2]. By modeling dependencies across individuals and sites, these models have empowered genomic analyses such as genotype imputation [3], haplotype phasing [4], and ancestry inference [5]. Such models also form the basis for programs that simulate artificial genomes (AGs) [6, 7, 8, 9, 10] that, in turn, have played a critical role in testing evolutionary hypothesis, inferring population genetic models, validating empirical results, and benchmarking methods. The ability to accurately and efficiently simulate AGs has been important to foster reproducibility and equity in research: trained models (or data simulated under the models) can be made available without restriction thereby side-stepping privacy restrictions associated with sharing primary genetic data.

There are generally two classes of models that can simulate AGs: the traditional and widely used population genetic approach such as the coalescent model [11] which simulates AG given a demographic history, and the latter unsupervised learning approach which produces "new data from old". The first approach has the benefit of producing AGs independent of the existing genetic data, but the simulated AGs do not necessarily mimic real genomes and history. The second approach has the advantage of closely mimicking the history and genomic properties of existing data at finer scale [12].

The coalescent with recombination [13] describes the probability of genetic variation in chromosomes sampled across individuals to population genetic parameters (population size, rates of mutation, and recombination) through latent gene genealogies along the genome (with the genealogies varying along the genome due to recombination) [14]. While expressive, inference under this model is computationally challenging due to the non-Markovian dependence induced among the latent genealogies. As a result, exact computation of the likelihood function under the coalescent with recombination is intractable. This difficulty has motivated investigation into tractable approximations to the coalescent. One class of approaches improve tractability by approximating the coalescent as a Markovian process along the genome [15, 12]. These approximations are the cornerstones of population coalescent-based simulators [9, 10]. An alternate class of approximations,

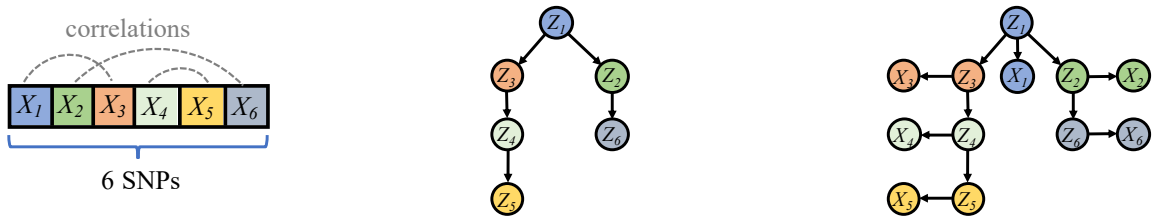
⋆ Equal contribution.

exemplified by the product of approximate conditionals (PAC) model and its extensions [16], aim to directly model the distribution of genetic variation without invoking a well-defined genealogical process. These models improve tractability by imposing a Markovian assumption which leads naturally to a hidden Markov model (HMM) [17] (while tractable, these models can still be computationally intensive leading to additional approximations [18]). The move away from an explicit connection to a genealogical process can limit the evolutionary inferences from such a model. Nevertheless, the PAC models and their variants are used in settings where the goal is to obtain an accurate probability model for the data distribution and have been widely used in applications such as haplotype phasing [18, 19], genotype imputation [20, 3], and ancestry inference [21, 22].

Recent advances in deep learning have led to the application of deep generative models to genetic variation [23, 24]. While these models are more expressive than a HMM, current deep models proposed for this task (based on GANs [25], variational autoencoders VAEs [26], and RBMs [27, 28]) are limited in important ways. First, these models do not permit exact probabilistic queries. As a result, it is not possible to compute likelihoods on held-out data for GANs or RBMs while VAEs only allow computation of a lower bound). More importantly, this difficulty precludes the application of these models to the missing data setting and to the tasks such as genotype imputation (all of which involve marginalizing over the joint probability distribution to be able to compute the probability of the observed variables or the conditional probability of the missing variables given the observed variables). Finally, these models are challenging to learn due to the complicated cost functions that need to be optimized as a means of learning parameters or hyperparameters. For example, training GANs involve a minimax objective whose optimization can lead to degenerate distributions (termed mode collapse [29]). Learning VAEs requires approximately maximizing a lower bound on the marginal likelihood while learning RBMs involves approximating the gradient of the log-likelihood which is typically achieved by running a Markov Chain Monte Carlo sampling algorithm [28].

1.1 Contributions

We propose a class of probabilistic models that can still give us those tractability advantages while fitting the data extremely well. To model the distribution over a sequence of variants, we propose a class of latent variable models where each hidden variable is associated with a SNP and the hidden variables are related by a tree-structured graphical model. This model, termed the *hidden Chow-Liu tree* [30], generalizes previously proposed HMMs. HMMs also associate a hidden random variable with each SNP. The hidden variables are related by a chain (a special type of tree) with the restriction that the only edges are present between variables associated with consecutive SNPs along the genome. By allowing for more general tree structures, the HCLT model is more general and can potentially capture long-range correlations or linkage-disequilibrium (LD) among SNPs [31]. While the HCLT model is more expressive than HMMs, it is unclear if such a model can be efficiently learned from data. A second contribution of our work is an affirmative answer to this question by representing HCLTs as Probabilistic Circuits (PCs) [32, 33], a large class of probabilistic models encoded using circuit representations. PCs have been shown to permit tractable inference tasks (*e.g.*, marginal likelihood computation) which are beyond the reach of most deep generative models. The representation of HCLTs as PCs enables us to leverage recent advances in deep learning such as stochastic learning algorithms and the use of GPUs to enable efficient parameter estimation and inference. We also leverage the framework of PCs to explore more restrictive models including fully factorized models (that assume all SNPs are independent) and Markov models. Finally, we perform extensive experiments to show that HCLT generates more accurate AGs relative to more restrictive models (fully factorized models and Markov models) suggesting that the structure encoded by the HCLT captures dependencies in genetic variation data. More interestingly, we find that HCLT as well as deep generative models (generative adversarial networks GAN and restricted Boltzmann machines RBM) preserve LD structure among SNPs. When trained on a subset of individuals from the 1000 Genomes Project (1KGP) across 805 SNPs that are distributed across the genome (and chosen to capture global population structure) as well as a second dataset of 10K SNPs from a contiguous region on chromosome 15, PCs learn distributions with improved log-likelihoods on a distinct set of individuals not used in training. We also evaluate the AGs generated by different models by comparing the PCA plots, allele frequencies, and LD patterns and observe that the AGs generated by the PCs are substantially closer to the patterns observed in real data. Our results suggest that the increased



(a) Genetic data with 6 SNPs

(b) Learned CLT structure captures strong pairwise correlation between SNPs.

(c) Learned HCLT structure given the SNPs in (a).

Fig. 1: Generating HCLT structures given genetic data. The hidden variables (Z_i) corresponding to SNPs with high pairwise correlations (X_i) are connected with each other in the HCLT graphical model.

expressivity of HCLTs leads to more accurate models of genetic variation. Furthermore, recent advances in learning and inference enabled by PCs allows us to fully exploit this increased capacity.

2 Methods

We first describe the class of probabilistic models that forms the basis for our work (Section 2.1). We then provide background on probabilistic circuits (PCs), a class of probabilistic models that include HCLTs and form the basis for efficient inference and learning (Section 2.2). Finally, we describe the details of the PC algorithms that we use to enable efficient learning and inference (Section 2.3).

2.1 Hidden Chow-Liu Trees (HCLT)

Hidden Chow-Liu Trees (HCLTs) [30] represent a distribution over a collection of random variables (RVs) ($\mathbf{Z} = (Z_1, \ldots, Z_N), \mathbf{X} = (X_1, \ldots, X_n)$). $\mathbf{Z}$ denote hidden or latent RVs while $\mathbf{X}$ denote observed RVs. The joint distribution is described by a graphical model ($\mathcal{G}$) where nodes in the graph represents the RVs and lack of edges among the nodes represent conditional independence assumptions. In HCLTs, we have edges from each hidden variable to its corresponding observed random variable ($Z_n \rightarrow X_n$) while the edges among the hidden variables form a tree. When the graph over the hidden variables is a chain ($Z_1 \rightarrow Z_2 \rightarrow \ldots Z_N$), we obtain a hidden Markov model (HMM). By permitting tree-structured graphs, HCLTs generalize HMMs and can provide a better representation of the data to capture long-range dependence, *e.g.*, RV X_1 and X_5 are highly correlated without being correlated with X_2, X_3, X_4. To accurately model dependencies among input variables with a tree, we use the Chow-Liu algorithm [34], which can capture major pairwise correlations between variables.

For genetic variation data over N single nucleotide polymorphisms (SNPs), each X_n denotes the genotype value at SNP $n \in \{1, \ldots, N\}$ ($X_n \in \{0, 1\}$ when we model haploid genomes while $X_n \in \{0, 1, 2\}$ when we model diploid genomes. Each Z_n is a discrete RV that can take one of L values ($Z_n \in \{0, \ldots, L-1\}$). Figure 1 demonstrates how to construct an HCLT using genetic data. Given a dataset $\mathcal{D}$ that contains 6 SNPs (Figure 1(a)), we first invoke the Chow-Liu algorithm to generate a tree over the SNPs' latent variables (Figure 1(b)). The tree encodes strong variable dependencies by placing highly correlated SNPs (e.g., X_1 and X_3) closer in the generated tree. Finally, HCLT is constructed by adding an edge from every latent variable Z_i to its corresponding observed variable X_i (Figure 1c).

The parameters of the HCLT are those associated with the discrete conditional probability distributions $P(X_n|Z_n)$ and $P(Z_n|Z_{Pa(n)})$, where $Pa(n)$ denotes the parent of node n in the tree. Further, the hyper-parameters include the range L of hidden RVs. Learning HCLTs that accurately represent the distribution of genetic variation requires learning its parameters efficiently. To enable this, we show that HCLTs can be represented as a class of tractable probabilistic models termed probabilistic circuits (PCs). This representation allows us to leverage recent advances in efficient learning and inference in PCs [35] to the problem of learning HCLTs. HCLTs are a special class of smooth and decomposable (see Section 2.2 for their definition) PCs that support efficient (*i.e.*, linear in the size of the PC which, in turn, is linear in the number of SNPs) computation of marginal probabilities. Details of the HCLT learning and inference algorithms will be introduced in Section 2.3.

2.2 Probabilistic Circuits

Probabilistic Circuits (PCs) [32, 33] are a class of probabilistic models that support tractable probabilistic inference. These capabilities have allowed PCs to perform various probabilistic reasoning tasks that are out of reach for most deep generative models [25, 26]. For example, the tractability of PCs helps solve problems in explainable AI [36, 37, 38], algorithmic fairness [39, 40], and missing data robustness [41, 38, 42].

PCs are furthermore appealing for their expressive power and suitability for density estimation. Recent advances in structure learning [43] and parameter estimations [40, 30] allow PCs to accurately capture useful correlations in the data. Here we show that the learning and reasoning capabilities of PCs can achieve state-of-the-art results in artificial genomes generation. In the following, we first define the syntax and semantics of PCs and describe key assumptions that unlock their capabilities.

Representation PCs are an umbrella term for a wide family of tractable probabilistic models [44], including arithmetic circuits [45], sum-product networks [46], and cutset networks [47]. A PC $(\mathcal{G}, \boldsymbol{\theta})$ represents a joint probability distribution $\Pr(\mathbf{X})$ over random variables $\mathbf{X}$ through a directed acyclic graph (DAG) $\mathcal{G}$ parametrized by $\boldsymbol{\theta}$. The DAG $\mathcal{G}$ consists of three types of nodes — *input*, *sum*, and *product*. Each leaf node is an input node; each inner node n (i.e., sum or product) receives inputs from its children $\mathsf{ch}(n)$. Each node $n \in \mathcal{G}$ encodes a probability distribution $\Pr_n$, which is defined recursively as follows:

$$\Pr_n(\mathbf{x}) = \begin{cases} f_n(\mathbf{x}) & \text{if } n \text{ is an input node,} \\ \prod_{c \in \mathsf{ch}(n)} \Pr_c(\mathbf{x}) & \text{if } n \text{ is a product node,} \\ \sum_{c \in \mathsf{ch}(n)} \theta_{n,c} \Pr_c(\mathbf{x}) & \text{if } n \text{ is a sum node,} \end{cases} \tag{1}$$

where $f_n(\mathbf{x})$ is a univariate input distribution (e.g., Binomial, Gaussian), and $\theta_{n,c}$ denotes the parameter that corresponds to edge (n, c). Intuitively, a product node defines a factorized distribution over its inputs, and a sum node represents a mixture over its input distributions weighted by θ. Finally, the probability distribution of a PC is defined as the distribution represented by its root node. The size of a PC $(\mathcal{G}, \boldsymbol{\theta})$ is defined as the number of parameterized edges in its DAG $\mathcal{G}$.

Inference In contrast to many other generative models, PCs support efficient reasoning over its encoded distribution. One can compute likelihoods by evaluating the PCs feed-forward as in Equation 1. Many common reasoning tasks such as marginal probabilities and maximum a posterior probability (MAP) are also supported by PCs. To guarantee the efficiency for computing these queries, the DAG of the PC should satisfy certain structural constraints. In the following, we introduce the constraints that are necessary for computing marginals and MAP, respectively. Please refer to [48] for a more detailed summary of various inference scenarios for PCs.

To support linear-time computation (with respect to the size of the PC) of arbitrary marginal queries, PCs need to satisfy two structural properties — smoothness and decomposability. Both are properties of the scope $\phi(n)$ of PC units n, that is, the collection of variables defined by all its input nodes.

Definition 1 (Smoothness) *A PC $(\mathcal{G}, \boldsymbol{\theta})$ is smooth if for any sum node $n \in \mathcal{G}$, its children have identical scope: $\forall c_1, c_2 \in \mathsf{ch}(n) : \phi(c_1) = \phi(c_2)$.*

Definition 2 (Decomposability) *A PC $(\mathcal{G}, \boldsymbol{\theta})$ is decomposable if for any produce node $n \in \mathcal{G}$, its children have disjoint scopes: $\forall c_1, c_2 \in \mathsf{ch}(n), c_1 \neq c_2 : \phi(c_1) \cap \phi(c_2) = \emptyset$.*

Given a smooth and decomposable PC, querying an arbitrary marginal probability boils down to a feedforward evaluation of its DAG, thus the computation time is linear with respect to the size of the PC.

To compute MAP queries in linear time, a PC should additionally satisfy a structural constraint termed determinism [49], which is a property of the PC nodes' support: for any PC node n, its support $\mathtt{supp}(n)$ is the set of complete assignments for which the output of n is non-zero: $\mathtt{supp}(n) := \{\mathbf{x} \in \mathtt{val}(\mathbf{X}) | \Pr_n(\mathbf{x}) \neq 0\}$. Intuitively, the support of a node n is the set of assignments $\mathbf{x}$ that activate it. Note that although the support sizes could be exponential with respect to the number of input variables, we never explicitly materialize them during training and inference, so it will not harm the efficiency of these algorithms.

Definition 3 (Determinism) *A PC $(\mathcal{G}, \boldsymbol{\theta})$ is deterministic if for any sum node $n \in \mathcal{G}$, its children have disjoint support:* $\forall c_1, c_2 \in \mathsf{ch}(n)(c_1 \neq c_2), \mathtt{supp}(c_1) \cap \mathtt{supp}(c_2) = \emptyset$.

HCLTs can be represented as smooth and decomposable PCs, meaning that they support efficient computation of marginal queries. This also implies that the likelihoods of HCLTs can be computed efficiently. When considering both the observed and hidden variables, HCLTs are also deterministic, meaning that the MAP instance $\arg\max_{\mathbf{x} \in \mathbf{X}, \mathbf{z} \in \mathbf{Z}} \Pr_n(\mathbf{x}, \mathbf{z})$ can be evaluated in linear time. However, note that HCLTs are not deterministic with respect to the observed variables $\mathbf{X}$, hence we need approximation algorithms to compute $\arg\max_{\mathbf{x} \in \mathbf{X}} \Pr_n(\mathbf{x})$.

Parameter Estimation Depending on the structural constraints possessed by a PC, different parameter learning techniques can be applied. First, if a PC is smooth, decomposable, and deterministic, its maximum-likelihood estimation (MLE) parameters can be efficiently learned in closed form [50]. To formalize the MLE parameters, we define the context γ_n of any node n as follows. The context of the root node n_r is its support $\mathtt{supp}(n_r)$. The context of any other node is the intersection of its support and the union of its parents' contexts:

$$\gamma_n := \bigcup_{c \in \mathsf{pa}(n)} \gamma_c \cap \mathtt{supp}(n).$$

For any sum node n and its child c, the associated MLE parameter $\theta^*_{n,c}$ on a dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ is

$$\theta_{n,c} = F_{\mathcal{D}}(n, c) / \sum_{c \in \mathsf{ch}(n)} F_{\mathcal{D}}(n, c), \qquad \text{where} \quad F_{\mathcal{D}}(n, c) := \sum_{i=1}^{N} \mathbb{1}[\mathbf{x}_i \in \gamma_n \cap \gamma_c]. \tag{2}$$

The quantity $F_{\mathcal{D}}(n, c)$ is called the *circuit flow* of edge (n, c). Intuitively, circuit flows count the number of samples in $\mathcal{D}$ that "activate" an edge.

If determinism is not satisfied, which is the case for HCLTs, the MLE solution will not have a closed-form expression. Instead, we resort to iterative algorithms such as Expectation-Maximization (EM). Since every non-deterministic PC can be augmented as a deterministic PC with additional hidden variables [51], the parameter learning problem of non-deterministic PCs can be equivalently viewed as learning the parameters of deterministic PCs given incomplete data. Specifically, an HCLT can be viewed as a deterministic PC considering both the observed variables $\mathbf{X}$ and the hidden ones $\mathbf{Z}$; and the dataset over $\mathbf{X}$ can be viewed as a dataset over $\mathbf{X}, \mathbf{Z}$ but values for $\mathbf{Z}$ are all missing. This leads to the EM algorithm where we compute the *expected circuit flow* given incomplete data in the E step and estimate the closed-form MLE parameters given the expected circuit flows in the M step [52, 40].

Concretely, given a deterministic PC $(\mathcal{G}, \boldsymbol{\theta})$ with root node r and an incomplete dataset $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$, the parameters for the next EM iteration are given by [40]:

$$\theta^{(\text{new})}_{n,c} = \mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c) / \sum_{c \in \mathsf{ch}(n)} \mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c), \qquad \text{where} \quad \mathrm{EF}_{\mathcal{D},\boldsymbol{\theta}}(n, c) := \sum_{i=1}^{N} \mathbb{E}_{\mathbf{z} \sim \Pr_r(\cdot|\mathbf{x}_i)} \big[\mathbb{1}[\mathbf{z}\mathbf{x}_i \in \gamma_n \cap \gamma_c]\big] \tag{3}$$

defines an expected version of circuit flow for edge (n, c) given samples with missing values in $\mathcal{D}$.

2.3 Fitting PCs to data

Any PGM can be transformed into a PC that encodes the same probability distribution. We demonstrate the high-level idea of this transformation, and refer interested readers to [33] for more details. To transform a HCLT into an equivalent PC, we iteratively encode every conditional probabilities $\Pr(\mathbf{X}_n | \mathbf{X}_{Pa(n)})$ by representing each possible value of $\mathbf{X}_n$ as a sum node. Thus, the probability of $\Pr(\mathbf{X}_n = \mathbf{x}_n | \mathbf{X}_{Pa(n)} = \mathbf{x}_{Pa(n)})$ can be represented by the weight of an edge connecting $\mathbf{x}_n$ and $\mathbf{x}_{Pa(n)}$. Take the HCLT in Figure 1 as an example. The conditional probabilities are encoded into a single PC in a bottom-up manner: we first encode $\Pr(X_5|Z_5)$ and then followed by $\Pr(X_4|Z_4)$ and $\Pr(Z_5|Z_4)$, and so on.

Using their equivalent PC representations, HCLTs can be trained efficiently using the PC package Juice.jl [35]. Compared to classic parameter learning implementations for graphical models, We make parameter estimation more accurate and efficient by the following three main improvements. First, by representing

Table 1: **Density estimation results in 805 and 10K SNPs data.** Averaged training and test log-likelihoods and models sizes (number of parameters, edges and nodes in the PCs) for INDEP, MARKOV (order is 10), HMM, CLT, STRUDEL, and HCLT. The bold values highlight the best averaged test set log-likelihoods.

Dataset	Category	INDEP	MARKOV	HMM	CLT	STRUDEL	HCLT
805	**train LL**	-490.73	-433.57	-402.45	-414.68	-402.02	-387.97
	test LL	-491.10	-438.64	-402.50	-415.83	-407.26	**-389.20**
	#params	1.61k	51.26k	231.10k	4.55k	77.80k	61.12k
10K	**train LL**	-2389.69	-626.18	-1192.78	-444.06	-444.03	-282.07
	test LL	-2390.09	-633.14	-1194.72	-456.39	-459.61	**-310.93**
	#params	20.0k	20461.57k	2879.26k	49.0k	2194.32k	5661.95k

graphical models (e.g., HCLTs) as PCs, we exploit the structure of the model to extensively parallelize the computation required by EM updates (Equation 3). We develop specialized GPU kernels to significantly speedup the EM algorithm. As a result, despite the model having more than a million parameters, a single EM epoch on the 805 SNP dataset (Section 3.1) can be done in 5 seconds.

Next, we benefit from the stochastic optimization algorithms that were popularized for deep learning by using a stochastic version of EM. Specifically, we use mini-batches of data to compute $\boldsymbol{\theta}^{(\text{new})}$ (Equation 3) and update the parameters toward this target with a step size η: $\boldsymbol{\theta}^{(k+1)} \leftarrow (1-\eta)\boldsymbol{\theta}^{(k)} + \eta\boldsymbol{\theta}^{(\text{new})}$. In practice, we run stochastic EM for 400 epochs with batch size 1024. The step size η is annealed linearly from 0.05 to 0.01.

Finally, Juice.jl provides effective parameter regularization algorithms to combat overfitting, which is a major problem, as in some datasets we used, the number of variables is much larger than the number of training samples. Therefore, we utilize classic methods such as adding pseudocounts, as well as recently proposed data softening and entropy regularization methods to mitigate overfitting [30]. As we will show in the experiments, despite the expressiveness of HCLTs, they are not prone to overfitting by virtue of these regularization methods.

3 Results

In this section, we empirically demonstrate the effectiveness of PCs in terms of modeling genome sequencing data. In order to show the generality of PC structures, in addition to HCLT (Section 2.1), we also evaluate another PC learner named STRUDEL [43]. STRUDEL learns a deterministic PC by first transforming a CLT into a PC and then performing a heuristic search guided by log-likelihoods to edit PC structures. We show the benefits of both structures in the following sections.

3.1 Data

We use 2504 genomes from the 1000 Genomes Project [53] to evaluate our models and generate artificial genomes (AGs). When analyzing global structure (Section 3.3), we use a set of 805 highly differentiated SNPs from across the genome that are a subet of the SNP set identified from Colonna et al [54]. When analyzing local structure (Section 3.4), we use a set of 10K SNPs drawn from a single genomic locus on chromosome 15. For all the experiments, we apply a 0.8/0.2 train/test split to phased data. Models are trained on the training set and evaluated on the test set. For every model, we simulate 5000 AGs and compare them to the test set genomes.

3.2 Evaluation

Baselines To benchmark our model performance in estimating density and simulating artificial genomes, we first compare it to three popular probabilistic graphical models (PGMs) that support tractable likelihood

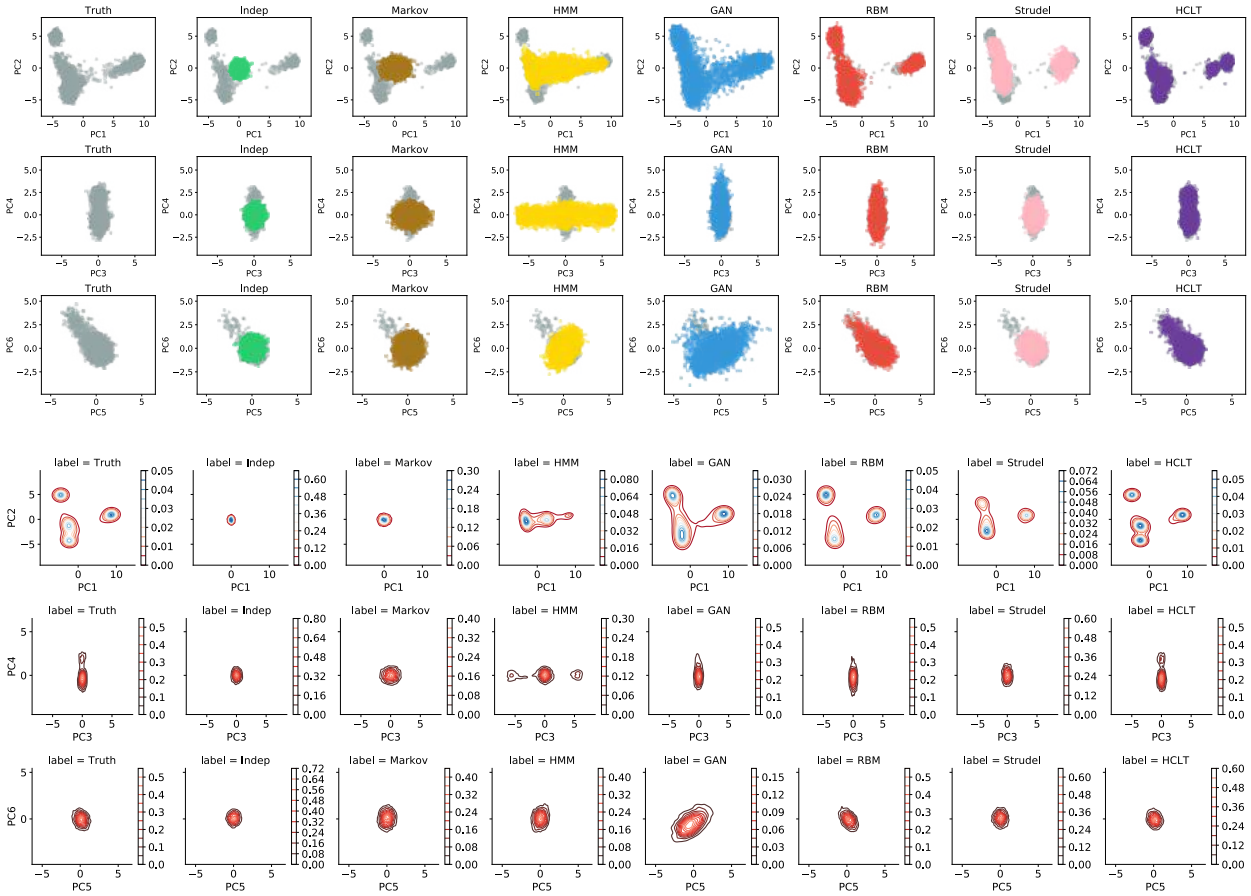


Fig. 2: **Principal components analysis for models trained on the 805 dataset**. The first six axes of a single PCA applied to the test set of the 805 dataset (gray) and AGs generated via Indep (green), Markov (brown), HMM (orange), GAN (blue), RBM (red), Strudel (pink), and HCLT (purple). The test set contains 961 haplotypes, and each model generate 5000 haplotypes as AGs. The top three panels plot the samples while the bottom three panels show the density plot of these samples.

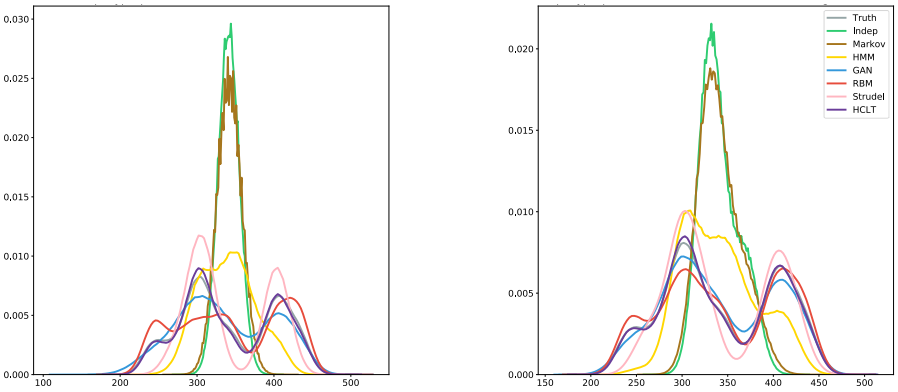
computation: fully-factorized distributions (Indep), higher-order Markov chain models (Markov) and non-homogeneous hidden Markov models (HMM). In order to estimate the parameters of these models, Indep and Markov have closed-form MLE solutions; while for HMM, we perform EM algorithm with random initialization. To facilitate the implementation benefits discussed in Section 2.3, we transform all these PGMs as equivalent PCs for efficient parameter estimation. We also compare against existing neural networks methods: (1) generative adversarial networks (GAN) and (2) Restricted Boltzmann machines (RBM) as implemented in [23]. For both neural network baselines, we use the samples generated by the corresponding authors for comparison.[4]

Evaluation criteria We evaluate these models using the following metrics: (1) log-likelihood on test data to assess the capability of each model as a density estimator; (2) summaries of AGs sampled from each model that include the top principal components that summarize the dominant axes of variation in the samples; (3) allele frequencies at individual SNPs, which calculate marginal probabilities and act as a one-point estimation; (4) linkage disequilibrium at pairs of SNPs, which calculate pairwise probabilities and act as a two-point estimation.

[4] Note that [23] did not do train/test splits so GAN and RBM are actually trained on train+test.

Table 2: **Evaluating the performance in preserving population structure using principal component analysis**: Wasserstein 2D distances between the PCA representations of real versus generated individuals. **within, between**: Wasserstein distance between the pairwise Euclidean distances of haploid genomes within a single dataset or between the real and generated individuals. r^2: Squared Pearson correlations between real and generated LD across all pairs of samples. We denote REAL for the testset. Bolded values indicate the best among all compared models.

Dataset		REAL	INDEP	MARKOV	HMM	GAN	RBM	STRUDEL	HCLT
805	PCA1-2	0.0010	0.2272	0.1666	0.0758	0.0040	0.0089	0.0065	**0.0015**
	PCA3-4	0.0015	0.0082	0.0280	0.0588	0.0175	0.0045	0.0107	**0.0020**
	PCA5-6	0.0013	0.0019	0.0213	0.0270	0.0043	0.0017	0.0017	**0.0013**
	within	0.98	43.92	42.10	24.99	4.96	6.96	9.25	**2.41**
	between	0.49	37.17	35.96	20.27	2.34	3.28	5.90	**1.26**
	r^2	0.99	0.67	0.76	0.73	0.95	0.98	0.96	**0.99**
10K	PCA1-2	0.0012	0.1905	0.0881	0.0946	0.0065	0.0144	0.0056	**0.0029**
	PCA3-4	0.0014	0.1655	0.0148	0.0572	**0.0018**	0.0107	0.0037	0.0022
	PCA5-6	0.0013	0.0889	0.0091	0.0169	**0.0014**	0.0063	0.0036	0.0020
	within	1.41	177.86	1223.19	148.65	107.84	29.88	36.69	**21.28**
	between	0.81	128.95	678.85	115.65	44.77	47.61	36.74	**24.51**
	r^2	0.99	0.38	0.66	0.50	0.95	0.94	0.94	**0.96**



(a) within each dataset (b) between datasets and ground truth

Fig. 3: Distribution of haplotypic pairwise Euclidean distances within (3a) datasets and between (3b) AG datasets and test set from 805 dataset using different models.

3.3 Reconstructing Global Population Structure

We first compared the ability of different models to represent genetic variation across 805 SNPs sparsely sampled from across the genome. We simulate AGs with STRUDEL, HCLT and all five baselines (INDEP, MARKOV, HMM, GAN, and RBM) for comparison. We use 2504 diploid genomes (5008 haploid genomes) from the 1000 Genomes Project [53] as our dataset, and all models are learned on 805 SNPs which are sparsely sampled from across the genome. We tune hyper-parameters on a small split of training data as validation: we use the order of 5 for higher-order Markov chain, and hidden states of 16 for HCLT and HMM.

Table 1 shows that STRUDEL and HCLT learn more accurate probabilistic models than fully-factorized distributions, Markov chains and HMMs as measured by their log likelihood on the test set. Additionally, the PC models are relatively lightweight: they have similar sizes compared to the adopted Markov chains. Note that we do not compare with GAN and RBM since they do not support tractable exact likelihood computation.

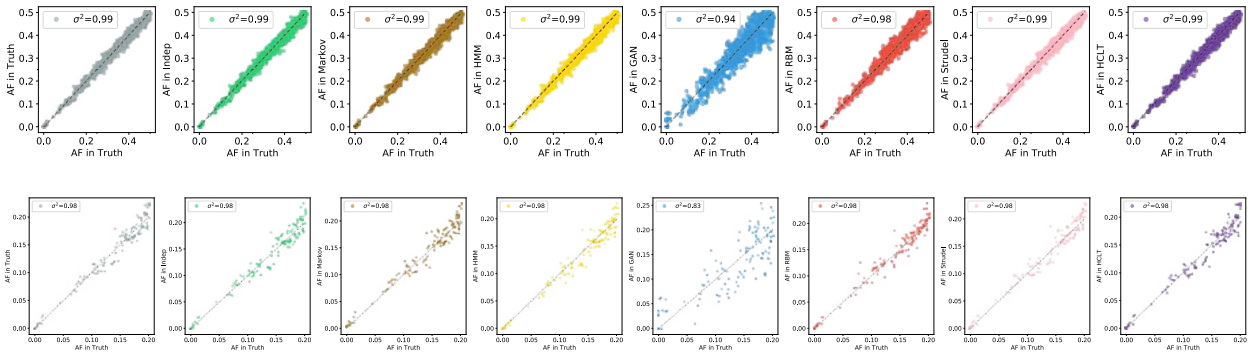


Fig. 4: Comparison of allele frequency between ground truth genomes from the 805 SNP dataset and the AGs counterparts generated using Indep, Markov, GAN, RBM, Strudel and HCLT models (a) for the whole range and (b) with a focus on low frequencies SNPs. The plot legend σ^2 refers to squared Pearson correlations between ground truth genomes and AGs.

We then analyze the quality of AGs generated by all models. Figure 2 shows that the AGs generated by Indep and Markov fall within the center of the variation seen in the test samples. The AGs generated by the HMM cover some spaces of PC1 and PC2, whereas GAN and RBM can capture most of the global structure. Regardless, AGs generated by the GAN tend to cover a larger space than the real test data. In contrast, Strudel and HCLT are able to capture more details: the PC1 and PC2 plots of HCLT is almost identical to the ground truth, and the PC3 and PC4 are also well captured. To quantify the accuracy of the PCs computed from the AGs, we computed Wasserstein distances between the 2D PCA representations of test data versus simulated data (Table 2). Wasserstein distances between the 2D PCA representations of test data versus simulated data are lower (closer to 0) for HCLT than for RBM and GAN along every pair of dimensions. We additionally compute the pairwise differences of haploid genomes within a single dataset or between the test dataset and AG datasets. Figure 3 shows the pairwise differences distribution while Table 2 rows 4-5 shows the Wasserstein distances between these distribution. GAN, RBM, Strudel and HCLT all capture the three modes in the distribution while the histogram of GAN and RBM are more uniform.

Next, we examined the allele frequencies in the AGs relative to the allele frequencies in the test data. As shown in Figure 4, the allele frequencieis of Strudel and HCLT are more centered around the diagonal, which indicates that they yield better calibrated probabilities.

3.4 Reconstructing Local Population Structure

To evaluate the ability of PCs to generate genome sequences across a dense set of SNPs from a single genomic region, we applied the Strudel and HCLT learner to a region with 10K SNPs from chromosome15 in the 1000 Genomes data. The log-likelihoods in Table 1 show that Strudel and HCLT can still deliver expressive PC models with moderate sizes. This suggests that these PC learners can handle high-dimensional genetic datasets and accurately capture long-range correlations. Note that although HMM has similar likelihoods as Strudel in 805 SNPs dataset, the performance is much worse in the 10K SNPs dataset, which shows that HMM is hard to capture long-range correlations for it is a linear model. PCA results and comparison of pairwise distances of AGs (Table 2) show that the AGs generated by HCLT are most similar to the test data on the most important principal components and in terms of pairwise distances.

Allele frequency analysis in Figure 5 shows that RBM and GAN perform poorly especially for low frequency alleles while Strudel and HCLT still show well-calibrated correlations. Allele frequency analysis can be seen as a first dimension correlation analysis of AGs. Since SNPs from a given genomic region tend to be correlated, we examined patterns of linkage disequilibrium (LD) to assess how the pairwise short and long-range correlations of SNPs can be captured by AGs. The pairwise LD matrix in Figure 6b shows that HCLTs accurately capture patterns of LD in this region. Plotting LD as a function of SNP distance in Figure 6a demonstrates that HCLT, Strudel and HMM better capture better correlation across shorter

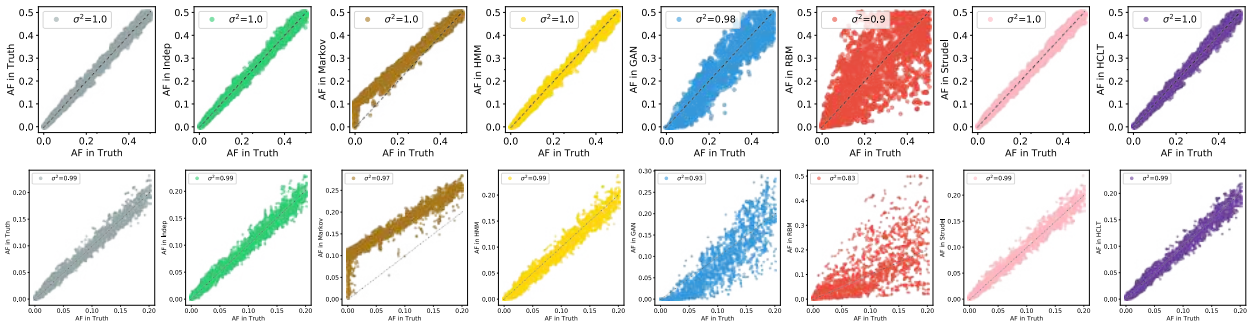
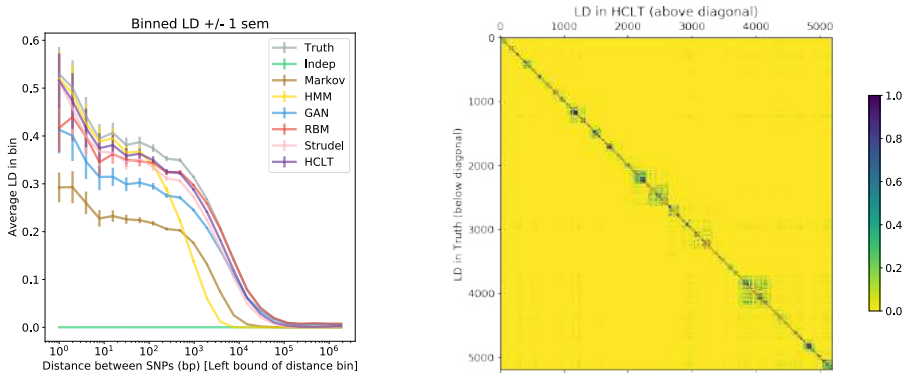


Fig. 5: Allele frequency comparison of corresponding SNPs between ground truth genomes from 10K dataset and AGs counterparts generated using Indep, Markov, GAN, RBM, Strudel and HCLT models (a) for the whole range and (b) zoomed to low frequencies.



(a) (b)

Fig. 6: **Linkage disequilibrium analysis**. (6a) LD as a function of SNP distance on all methods. (6b) Correlation matrices (r^2) of SNPs where the lower triangular parts are in real genomes from 10K dataset and upper triangular parts in AGs generated by HCLT.

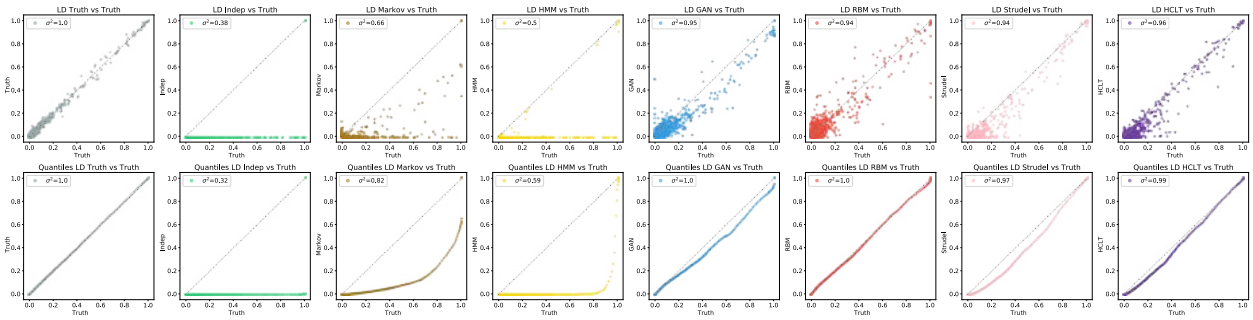


Fig. 7: **Linkage disequilibrium comparison.** The first row plots the pairwise LD between pairs of points from AGs vs. real test set for all models on 10K dataset. The second row shows the respective QQ-plots, which illustrate the corresponding quantiles.

length scales while all models, expect for HMM are accurate at longer length scales. On the other hand, while the HMMaccurate captures LD at shorter length scales, it performs poorly across longer length scales. Correlations between real and AGs shown in the last row of Table 2 and in Figure 7 comes to a similar conclusion that HCLT more accurately captures the distribution of LD across SNPs within the region.

4 Conclusion

We present HCLT, a new simulator for AGs based on PCs, which has comparable or better performance than the current state-of-the-art AG simulators in capturing key population genetic statistics such as allele frequencies, linkage disequilibrium, pairwise haplotype distances, and population structure, while being tractable and expressive. By capturing long-range correlations and offers probabilistic inference, HCLT is particularly suitable for modeling genetic data. This work presents the first population genetic method based on PC and exemplifies its potential for population genetic simulations. We outline limitations of our study and directions for future work. First, while the HCLT shows great performance in simulating 10K SNPs, neither HCLT nor other highly expressive methods can simulate whole artificial genomes (containing millions of SNPs). However, HCLT is significantly faster to train compared to GAN and RBM. Future improvements are needed for making our proposed models scale to these settings. Second, in our current study, we applied all methods to haploid genomes. This process ignores the uncertainty that arises due to phasing diploid genomes. Due to PC's ability to handle missing data easily, we expect the extension of HCLT to unphased diploid genomes to be straightforward and would lead to the simulation of diploid AGs. Third, our current experiments pool individuals from multiple distinct populations genotyped in the 1000 Genomes project as a way to obtain large sample sizes to train our models. It is plausible that the accuracy of simulating AGs will vary across populations (with distinct demographic histories that will, in turn, lead to distinct distributions of allele frequencies and LD). Admixed populations, with long-range LD due to admixture, are expected to have different patterns of LD [55] relative to homogeneous populations. Ongoing attempts to genotype large numbers of individuals across diverse populations will lead to large numbers of samples needed to train population-specific models based on HCLT.

Acknowledgements

This work was funded in part by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005, NIH grants U01HG011715, HG006399, R35GM125055, NSF grants #IIS-1943641, #IIS-1956441, #CCF-1837129, #IIS-2106908, CAREER-1943497, Samsung, CISCO, a Sloan Fellowship, and a UCLA Samueli Fellowship.

Code availability

Code and experiments are available on https://github.com/UCLA-StarAI/genetic-pc.

Bibliography

[1] Magnus Nordborg. Coalescent theory. *Handbook of Statistical Genomics: Two Volume Set*, pages 145–30, 2019.

[2] John Wakeley. Developments in coalescent theory from single loci to chromosomes. *Theoretical population biology*, 133:56–64, 2020.

[3] Jonathan Marchini and Bryan Howie. Genotype imputation for genome-wide association studies. *Nature Reviews Genetics*, 11(7):499–511, 2010.

[4] Sharon R Browning and Brian L Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

[5] Daniel Mas Montserrat, Carlos Bustamante, and Alexander Ioannidis. Class-conditional vae-gan for local-ancestry simulation. *arXiv preprint arXiv:1911.13220*, 2019.

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

[7] Garrett Hellenthal and Matthew Stephens. mshot: modifying hudson's ms simulator to incorporate crossover and gene conversion hotspots. *Bioinformatics*, 23(4), 2007.

[8] Laurent Excoffier and Matthieu Foll. Fastsimcoal: a continuous-time coalescent simulator of genomic diversity under arbitrarily complex evolutionary scenarios. *Bioinformatics*, 27(9):1332–1334, 2011.

[9] Jerome Kelleher, Alison M. Etheridge, and Gilean McVean. Efficient Coalescent Simulation and Genealogical Analysis for Large Sample Sizes. *PLoS Computational Biology*, 12(5):1–22, 2016.

[10] Franz Baumdicker, Gertjan Bisschop, Daniel Goldstein, Graham Gower, Aaron P Ragsdale, Georgia Tsambos, Sha Zhu, Bjarki Eldon, Castedo E Ellerman, Jared G Galloway, et al. Efficient ancestry and mutation simulation with msprime 1.0. *bioRxiv*, 2021.

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

[12] Paul Marjoram and Jeff D. Wall. Fast "coalescent" simulation. *BMC Genetics*, 7(1):16, 2006.

[13] Robert C. Griffiths and Paul Marjoram. An Ancestral Recombination Graph. In P. Donnelly and S. Tavare, editors, *Progress in Population Genetics and Human Evolution, IMA Volumes in Mathematics and its Applications, vol. 87*, pages 257–270. Springer, 1997.

[14] Carsten Wiuf and Jotun Hein. Recombination as a point process along sequences. *Theoretical Population Biology*, 55(3):248–259, 1999.

[15] Gilean A T McVean and Niall J Cardin. Approximating the coalescent with recombination. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 360(1459):1387–93, 2005.

[16] Na Li and Matthew Stephens. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–2233, 2003.

[17] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[18] Paul Scheet and Matthew Stephens. A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *The American Journal of Human Genetics*, 78(4):629–644, 2006.

[19] Olivier Delaneau, Jonathan Marchini, and Jean-François Zagury. A linear complexity phasing method for thousands of genomes. *Nature methods*, 9(2):179–181, 2012.

[20] Bryan Howie, Christian Fuchsberger, Matthew Stephens, Jonathan Marchini, and Gonçalo R Abecasis. Fast and accurate genotype imputation in genome-wide association studies through pre-phasing. *Nature genetics*, 44(8):955–959, 2012.

[21] Yael Baran, Bogdan Pasaniuc, Sriram Sankararaman, Dara G Torgerson, Christopher Gignoux, Celeste Eng, William Rodriguez-Cintron, Rocio Chapela, Jean G Ford, Pedro C Avila, et al. Fast and accurate inference of local ancestry in latino populations. *Bioinformatics*, 28(10):1359–1367, 2012.

[22] Alkes L Price, Arti Tandon, Nick Patterson, Kathleen C Barnes, Nicholas Rafaels, Ingo Ruczinski, Terri H Beaty, Rasika Mathias, David Reich, and Simon Myers. Sensitive detection of chromosomal segments of distinct ancestry in admixed populations. *PLoS genetics*, 5(6):e1000519, 2009.

[23] Burak Yelmen, Aurélien Decelle, Linda Ongaro, Davide Marnetto, Corentin Tallec, Francesco Montinaro, Cyril Furtlehner, Luca Pagani, and Flora Jay. Creating artificial human genomes using generative neural networks. *PLOS Genetics*, 17(2):1–22, 02 2021.

[24] CJ Battey, Gabrielle C Coffing, and Andrew D Kern. Visualizing population structure with variational autoencoders. *G3*, 11(1):1–11, 2021.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[27] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

[28] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[29] Sudarshan Adiga, Mohamed Adel Attia, Wei-Ting Chang, and Ravi Tandon. On the tradeoff between mode collapse and sample quality in generative adversarial networks. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1184–1188. IEEE, 2018.

[30] Anji Liu and Guy Van den Broeck. Tractable regularization of probabilistic circuits. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, dec 2021.

[31] Jonathan K Pritchard and Molly Przeworski. Linkage disequilibrium in humans: models and data. *The American Journal of Human Genetics*, 69(1):1–14, 2001.

[32] Antonio Vergari, YooJung Choi, Robert Peharz, and Guy Van den Broeck. Probabilistic circuits: Representations, inference, learning and applications. *AAAI Tutorial*, 2020.

[33] YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. oct 2020.

[34] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[35] Meihua Dang, Pasha Khosravi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. Juice: A julia package for logic and probabilistic circuits. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (Demo Track)*, 2021.

[36] Mahsan Nourani, Chiradeep Roy, Tahrima Rahman, Eric D. Ragan, Nicholas Ruozzi, and Vibhav Gogate. Don't explain without verifying veracity: An evaluation of explainable AI with video activity recognition. *CoRR*, abs/2005.02335, 2020.

[37] Kareem Ahmed, Zhe Zeng, Mathias Niepert, and Guy Van den Broeck. Simple: A gradient estimator for k-subset sampling. In *ICLR*, 2023.

[38] Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. On tractable computation of expected predictions. *Advances in Neural Information Processing Systems*, 32:11169–11180, 2019.

[39] Nikil Roashan Selvam, Guy Van den Broeck, and YooJung Choi. Certifying fairness of probabilistic circuits. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, feb 2023.

[40] YooJung Choi, Meihua Dang, and Guy Van den Broeck. Group fairness by probabilistic modeling with latent fair decisions. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, Feb 2021.

[41] Alvaro Correia, Robert Peharz, and Cassio P de Campos. Joints in random forests. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020.

[42] Wenzhe Li, Zhe Zeng, Antonio Vergari, and Guy Van den Broeck. Tractable computation of expected kernels. In *Proceedings of the 37th Conference on Uncertainty in Aritifical Intelligence (UAI)*, jul 2021.

[43] Meihua Dang, Antonio Vergari, and Guy Van den Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models (PGM)*, sep 2020.

[44] Honghua Zhang, Brendan Juba, and Guy Van den Broeck. Probabilistic generating circuits. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, jul 2021.

[45] Adnan Darwiche. A logical approach to factoring belief networks. In *Proceedings of KR*, pages 409–420, 2002.

[46] Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 689–690. IEEE, 2011.

[47] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 630–645. Springer, 2014.

[48] Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, dec 2021.

[49] Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. In *Proceedings of the Thirty-Fourth International Conference on Machine Learning (ICML)*, 2017.

[50] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2014.

[51] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

[52] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[53] Laura Clarke, Susan Fairley, Xiangqun Zheng-Bradley, Ian Streeter, Emily Perry, Ernesto Lowy, Anne-Marie Tassé, and Paul Flicek. The international Genome sample resource (IGSR): A worldwide collection of genome variation incorporating the 1000 Genomes Project data. *Nucleic Acids Research*, 45(D1):D854–D859, 09 2016.

[54] Vincenza Colonna, Qasim Ayub, Yuan Chen, Luca Pagani, Pierre Luisi, Marc Pybus, Erik Garrison, Yali Xue, Chris Tyler-Smith, 1000 Genomes Project Consortium, Goncalo R Abecasis, Adam Auton, Lisa D Brooks, Mark A DePristo, Richard M Durbin, Robert E Handsaker, Hyun Min Kang, Gabor T Marth, and Gil A McVean. Human genomic regions with exceptionally high levels of population differentiation identified from 911 whole-genome sequences. *Genome Biology*, 15(6):R88, 2014.

[55] Ariel Darvasi and Sagiv Shifman. The beauty of admixture. *Nature Genetics*, 37(2):118–119, 2005.

A Supporting information

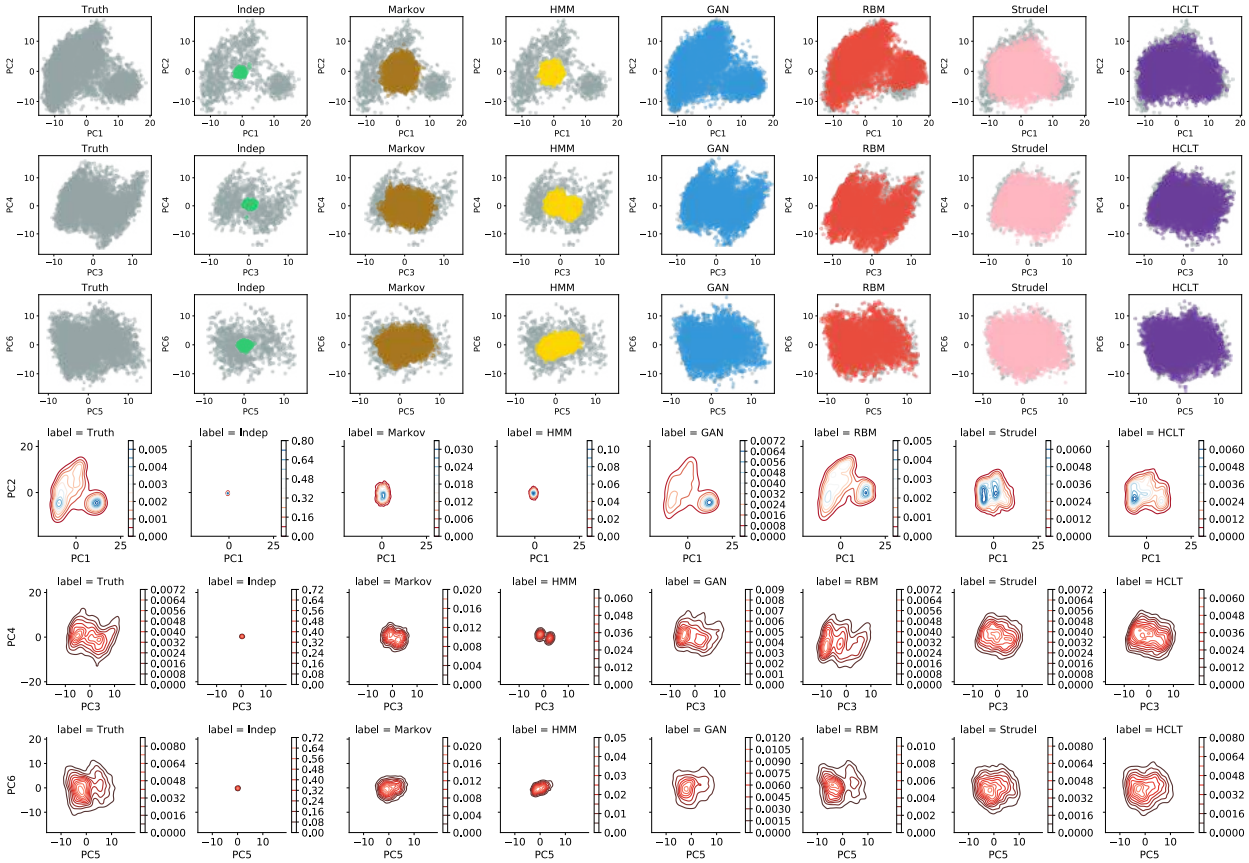


Fig. 8: **Principal component analysis**. The first six axes of a single PCA applied to test set real (gray) and AGs generated via Indep (green), Markov (brown), HMM (orange), GAN (blue), RBM (red), Strudel (pink), and HCLT (purple).There are 5000 haplotypes for each AG dataset and 961 in the test set of 10K dataset.