Ancestry Deconvolution via Differential Privacy

by

Raiyan Hasan Chowdhury

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2026

© Raiyan Hasan Chowdhury 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis presents the first study of ancestry determination under differential privacy (DP). Direct-to-consumer genomics companies, such as 23andMe, offer ancestry testing to millions of individuals, yet remain vulnerable to severe data breaches. Such incidents are especially concerning because genomic data is uniquely identifying, highly correlated, and permanent once exposed. At the time of writing, 23andMe disclosed a catastrophic breach in October 2023 that compromised the genetic profiles of an estimated 6.9 million users, underscoring the urgent need for stronger privacy guarantees in genomic analysis.

In this work, we investigate the application of DP to ancestry deconvolution. Using the 1000 Genomes dataset and Gnomix, a state-of-the-art ancestry inference model, we evaluate how privatizing single nucleotide polymorphism (SNP) data affects ancestry classification accuracy. We implement both naïve and correlation-aware local differential privacy (LDP) mechanisms across varying privacy budgets (ϵ), enabling a systematic study of the privacy–utility trade-off in ancestry inference.

Our results demonstrate that while naïve DP perturbations significantly degrade accuracy, correlation-aware LDP mechanisms preserve substantially more predictive power by accounting for linkage disequilibrium (LD). This thesis establishes a foundation for private ancestry deconvolution, providing an empirical benchmark of state-of-the-art DP methods in genomics and highlighting both the challenges and potential of integrating DP into ancestry testing.

Acknowledgements

All praise and gratitude is foremost due to God, the Lord and Sustainer of the Worlds.

I'd like to thank my past supervisor, Dan Brown, for getting me so far, and my current supervisor, Florian Kerschbaum, for getting me to the end.
May God grant you a speedy recovery, Dan.

Thank you to Gautam Kamath for introducing me to the wonderful world of differential privacy in CS 860 (Algorithms for Private Data Analysis) of Fall 2022.
It's unequivocally the best course I've ever taken.
I'd like to acknowledge my coursemates who were formative in building my understanding and curiosity: Chris, Vecna, Sabrina, Alice, Sara K. and Sara S.

I'd like to thank Urs Hengartner for making numerous suggestions to improve my thesis clarity and rigour. It reads a lot better thanks to his helpful comments.

I'd like to thank the friends who were there at a formative point of my life:
Sibthain, Hrithik, Ahmed, Ziyyad, Junaid and Bilal.
They were my support through the highs and lows of grad school.

I'd like to thank Fettah, my boss and great friend, for his guidance, wisdom, and support over the years.

I'd like to thank my cousin, Ayman, who always instilled in me a drive and curiosity to learn more.

Lastly, I would like to thank my parents and sister for their continual support.
I could not have done it without their encouragement and perseverance to see my thesis to the end.

Dedication

To the people of Palestine.

Your courage is noticed, and your voices are clearly heard.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

Dedication v

List of Figures ix

List of Tables x

1 Introduction 1

2 Background 4

2.1 Genomics 4

2.1.1 Linkage Disequilibrium 4

2.1.2 Ancestry Inference 5

2.2 Background on Differential Privacy 8

2.2.1 Formal Definition 8

2.2.2 Mechanisms 9

2.2.3 Central vs. Local DP 9

2.2.4 Challenges in Genomic Data 9

3 Problem Description **11**

3.1 Motivation . . . 11

3.2 Problem Statement . . . 11

3.3 Chapter Summary . . . 12

4 Genomics DP **13**

4.1 Early Applications of DP in Genomics . . . 13

4.2 Genomic Data-Sharing Beacons . . . 14

4.3 Global vs. Local DP in Genomic Settings . . . 15

4.4 Addressing Dependencies in Genomic Data . . . 15

4.4.1 The Yilmaz Algorithm for Dependent Local Differential Privacy . . . 16

4.5 DP for Synthetic Genomic Data . . . 18

4.6 DP in Ancestry Inference . . . 18

4.7 Challenges and Future Directions . . . 18

4.8 Summary . . . 19

5 Evaluation **20**

5.1 Experimental Setup . . . 20

5.1.1 Dataset . . . 20

5.1.2 Preprocessing . . . 21

5.1.3 Privacy Mechanism . . . 21

5.1.4 Accuracy Definition . . . 21

5.2 Results . . . 22

5.3 Discussion . . . 24

5.3.1 Privacy–Utility Trade-off . . . 24

5.3.2 Limitations . . . 24

5.3.3 Broader Implications . . . 25

5.4 Preliminary Reidentification Attack . . . 25

5.4.1 Attack Model 25
5.4.2 Results 26
5.4.3 Discussion 28

6 Conclusion **30**

References **33**

APPENDICES **38**

A Vectorized LDP SNP Perturbation **39**

B Scalable dLDP SNP Perturbation **42**

C Gnomix Ancestry Deconvolution **46**

D Reidentification Attack **52**

List of Figures

5.1 Accuracy of ancestry deconvolution under naïve and correlation-aware local differential privacy at varying ϵ values, compared against the non-private baseline (accuracy = 1.0). 23

5.2 Reidentification attack success rate as a function of the privacy budget ε for naïve randomized response and correlation-aware local differential privacy. 28

List of Tables

5.1 Ancestry inference accuracy under local differential privacy for varying privacy budgets ϵ. Baseline accuracy without privacy is 1.0. 23

5.2 Reidentification attack success rates under naïve randomized response. . . 27

5.3 Reidentification attack success rates under correlation-aware LDP. 27

Chapter 1

Introduction

Genetic ancestry testing services have emerged as a cultural phenomenon in recent years. Companies such as 23andMe and AncestryDNA use a simple saliva sample to deliver insights into heritage, making genomics accessible to millions. Yet this convenience comes with deep privacy risks. In late 2023, a major cybersecurity breach exposed the raw genetic data and personal details of approximately 6.9 million users of 23andMe [3]. Sensitive information such as ancestry, health predispositions, and family connections was compromised. The UK Information Commissioner's Office later fined 23andMe over £2.3 million for inadequate protections [11], and multiple U.S. states have taken legal action to prevent the sale of customers' genetic data during the company's bankruptcy proceedings [4]. These events highlight how, unlike passwords, genetic data cannot be changed, leaving individuals permanently vulnerable when breaches occur.

Surveys show that consumers are acutely aware of these risks: over 50% of users express serious concern about for-profit DNA-testing platforms sharing genetic information without explicit consent, even with trusted entities like law enforcement or researchers [5]. The gravity of these concerns is compounded by real-world harm. Cases like that of Michael Usry, a man falsely accused of murder after a partial DNA match via a genealogy site, illustrate how irreversible and far-reaching the consequences of misused genetic data can be [35].

In this context, *differential privacy (DP)* has emerged as the gold standard for protecting individual data. First formalized by Dwork *et al.* [18], DP offers strong theoretical guarantees by statistically limiting what adversaries can infer about any individual, regardless of background knowledge. This allows personal data to metaphorically "get lost in the crowd." However, genomic data presents unique challenges. Unlike many datasets,

DNA is high-dimensional, highly correlated, and biologically structured, which violates the independence assumptions most DP mechanisms rely on.

This thesis explores how differential privacy (DP) can be adapted to protect genomic data while preserving its utility for *ancestry deconvolution*, the task of inferring the ancestral origin of local chromosomal segments within an individual's genome. Unlike global ancestry estimation, which assigns a single ancestry proportion to an entire genome, ancestry deconvolution relies on fine-scale correlations among nearby single nucleotide polymorphisms (SNPs) and is therefore highly sensitive to perturbations that disrupt linkage disequilibrium. We evaluate local (individual-level) DP approaches, with a focus on reproducing and analyzing the method of Yilmaz *et al.* [45], the first to incorporate genomic correlations into a DP framework that enables privatized, per-individual genotype release for downstream analysis.

Research Hypothesis

Local Differential Privacy (LDP) provides a tunable privacy–accuracy trade-off in ancestry deconvolution. Prior work by Yilmaz et al. [45] suggests that correlation-aware perturbation improves this trade-off for genetic analyses. We hypothesize that these benefits extend to ancestry deconvolution: specifically, that correlation-aware LDP mechanisms will maintain higher accuracy across a wider range of privacy budgets compared to naïve, independent perturbation.

Scope of the Thesis

This thesis focuses specifically on evaluating differential privacy for ancestry inference. While genomic privacy is a broad field, we narrow our scope along three key dimensions:

- **Dataset.** We use the 1000 Genomes Project dataset [15], a widely adopted benchmark for population genetics research. To ensure dense and realistic SNP coverage, genotypes are imputed using Beagle [13].
- **Downstream Task.** We focus on ancestry classification as the primary utility measure, using Gnomix for high-resolution local ancestry deconvolution. This choice reflects the central role of ancestry inference in both research and consumer genomics.

- **Comparative Evaluation.** We evaluate a naïve randomized response mechanism against the dependent local differential privacy (dLDP) framework proposed by Yilmaz et al. [45], the only existing approach that explicitly models SNP correlations under local differential privacy.

By constraining the problem in this way, we establish a rigorous but tractable framework for evaluating whether dependency-aware differential privacy can meaningfully improve utility in genomic applications.

Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 presents foundational background on genomics and differential privacy. Chapter 3 formalizes the problem and scope. Chapter 4 reviews prior work in DP for genomics and explains the rationale for selecting Yilmaz *et al.* [45] as the evaluation focus. Chapter 5 details experimental methods and results, assessing ancestry inference accuracy under DP. Finally, Chapter 6 concludes with insights, limitations, and future research directions.

Chapter 2

Background

2.1 Genomics

Genomics is the study of the complete set of an organism's DNA, which is shorthand for deoxyribonucleic acid. DNA forms the building blocks of living organisms.

DNA consists of 4 nucleotides, which are: adenosine (A), guanine (G), cytosine (C), and thymine (T). DNA nucleotides are commonly referred to by their one-letter shorthand notation, as specified above. DNA nucleotides are nearly always present in pairs in DNA molecules, which have particular binding specificity. In particular, adenine binds to thymine, and guanine binds to cytosine. This is known as Chargaff's rules. DNA forms a double-helical polynucleotide structure.

The human genome consists of approximately 3 billion nucleotides. Over 99.9% of these are identical in humans. The remaining 0.1%, called single nucleotide polymorphisms (SNPs) form the variability across all humans.

Generally, one nucleotide occurs much more frequently in the population, known as the *major allele*, while at least one occurs much less frequently, the *minor allele*.

2.1.1 Linkage Disequilibrium

A central concept in human genetics is *linkage disequilibrium (LD)*, which refers to the non-random association of alleles at different loci. In other words, if two single nucleotide

polymorphisms (SNPs) are in linkage disequilibrium, then the allele observed at one SNP provides statistical information about the allele likely to be observed at the other.

LD arises because SNPs that are *physically close together on a chromosome* tend to be inherited together during meiosis. Recombination events, which reshuffle chromosomes across generations, are less likely to separate nearby loci, leading to correlated allele patterns. Over evolutionary time, recombination and mutation gradually erode these correlations, but substantial LD persists across the genome, particularly in regions with low recombination rates. Demographic events such as population bottlenecks and admixture can also shape the extent and structure of LD.

Formally, LD is often quantified using statistics such as D, D', or the squared correlation coefficient r^2. These measures compare the *observed frequency of allele combinations (haplotypes)* against the expected frequency under the assumption of independence. A high r^2 value between two SNPs indicates strong LD, whereas values near zero indicate linkage equilibrium (independence).

From a practical perspective, LD has several important implications. In *genome-wide association studies (GWAS)*, LD allows researchers to identify disease-associated loci even when the causal variant itself is not directly genotyped. In *ancestry inference*, models exploit LD patterns to detect chromosomal segments inherited from different ancestral populations. For the purposes of this thesis, LD is especially significant in the context of *differential privacy*: most privacy mechanisms assume independence across data attributes, but in genomics, SNP correlations due to LD violate this assumption. Thus, any attempt to privatize genomic data must contend with the information redundancy and dependency induced by LD.

2.1.2 Ancestry Inference

Ancestry inference is the broad task of determining the genetic origins of an individual by analyzing patterns of variation in their genome. Human genetic diversity is structured across populations, and differences in allele frequencies, haplotype structure, and linkage disequilibrium (LD) patterns provide signals that can be used to identify ancestral backgrounds. Within this broader task, methods may assign either global ancestry, defined as the proportion of an individual's genome originating from broad population groups, or local ancestry, which identifies the ancestral origin of specific chromosomal segments.

In this thesis, we focus specifically on local ancestry deconvolution, a form of ancestry inference in which the genome is segmented and each segment is assigned an ancestry label.

Early approaches to ancestry inference primarily relied on principal component analysis (PCA) of genome-wide SNP data, which captures broad-scale population structure. However, PCA lacks the resolution required for local ancestry deconvolution. Hidden Markov Models (HMMs) and related probabilistic frameworks were later introduced to explicitly incorporate LD and recombination patterns, enabling ancestry assignment at the haplotype level. Methods such as LAMP-LD and RFMix [33] significantly improved accuracy by modeling local ancestry along the genome.

Recent work has pushed the resolution further. State-of-the-art tools such as Gnomix [25] frame local ancestry deconvolution as a sequence classification problem and exploit correlations among SNPs to assign ancestry at fine genomic scales. These models are trained on reference panels such as the 1000 Genomes Project and learn to map genomic windows to ancestry labels with high accuracy. By leveraging modern machine learning techniques, they can capture complex dependency structures among SNPs that are difficult to model with classical HMM-based approaches.

For the purposes of this thesis, ancestry inference refers specifically to local ancestry deconvolution, and we use these terms interchangeably unless otherwise stated. Because local ancestry deconvolution relies heavily on LD structure, it represents a challenging and informative downstream task for evaluating the utility of differentially private genomic data releases. Privacy mechanisms that fail to preserve SNP correlations are therefore expected to substantially degrade ancestry deconvolution accuracy, making this task a natural benchmark for the work presented here.

Gnomix: High-Resolution Local Ancestry Deconvolution

Gnomix [25] is a state-of-the-art method for local ancestry inference (LAI) that leverages modern machine learning techniques to model fine-scale haplotype structure. Unlike earlier approaches such as RFMix, which rely on random forests combined with conditional random fields, Gnomix is built around a hierarchical deep learning architecture trained directly on phased SNP sequences from reference populations.

At a high level, Gnomix treats local ancestry assignment as a sequence-classification problem: the genome is partitioned into overlapping windows of SNPs, and the model predicts the ancestry label for each window based on its haplotypic pattern. This enables Gnomix to capture subtle correlations among nearby SNPs—arising from linkage disequilibrium (LD)—that define population-specific haplotypes. The model is trained using

supervised learning on large reference panels such as the 1000 Genomes Project [15], where the ancestry of each haplotype is known.

Model Architecture The Gnomix pipeline consists of two main components:

- **Base classifier.** A convolutional neural network (CNN) processes windows of phased SNPs and produces, for each window, a probability distribution over ancestral populations. The convolutional layers enable the model to learn short-range haplotypic motifs that are characteristic of different ancestries.
- **Smoothing module.** Because local ancestry along a chromosome is piecewise constant, with transitions induced by recombination, neighboring windows are highly correlated. Gnomix applies a smoothing step (implemented as a conditional-random-field–like module or XGBoost) to enforce spatial consistency and reduce noisy fluctuations in the per-window predictions.

Together, these components enable Gnomix to model both *local* haplotype features and *long-range* LD patterns shaped by demographic history.

Comparison to Prior Methods Gnomix improves upon earlier LAI methods in several ways:

- **Representation learning:** Whereas RFMix uses handcrafted features and decision trees, Gnomix learns haplotype representations directly from raw SNP sequences.
- **High-resolution predictions:** The combination of CNNs and smoothing enables ancestry calls at fine granularity, with transitions that more closely align with true recombination breakpoints.
- **Robustness to reference panel imbalance:** The neural architecture is less sensitive to uneven sample sizes across populations, a common challenge in large-scale datasets.

Empirical evaluations in Hilmarsson et al. [25] show that Gnomix outperforms RFMix and HMM-based tools on both simulated and real admixed genomes.

Relevance and Usage in This Thesis This thesis uses Gnomix as the downstream task for evaluating the utility of locally differentially private genotype data. In particular, we rely on the *pretrained chromosome 22 models* released by the authors [25]. We do *not* retrain Gnomix: training requires large-scale reference panels and substantial computational resources, and is beyond the scope of this work. Instead, we use the authors' provided reference dataset and model weights, ensuring consistency with prior literature and avoiding retraining variability.

For the perturbed-data evaluation, we apply local differential privacy to the sample genotype dataset distributed with the Gnomix software package. The privatized genotypes are then passed through the pretrained Chr22 model to measure how LDP noise affects Gnomix's local ancestry predictions.

Because Gnomix relies heavily on detailed LD patterns and haplotype structure, it serves as a sensitive benchmark for differentially private genomics. Any mechanism that disturbs SNP correlations is likely to degrade performance, making Gnomix an ideal system for evaluating the privacy–utility trade-off explored in this thesis.

2.2 Background on Differential Privacy

Differential privacy (DP) is a mathematical framework for quantifying privacy guarantees when releasing information derived from sensitive datasets. It ensures that the inclusion or exclusion of any individual record has a limited impact on the output of a computation, thereby protecting participants' data while still allowing meaningful statistical analysis.

2.2.1 Formal Definition

The central definition of differential privacy is as follows. A randomized algorithm $\mathcal{M}$ provides (ε, δ)-differential privacy if, for all pairs of neighboring datasets D and D' differing in at most one individual, and for all measurable subsets of outputs $S \subseteq \text{Range}(\mathcal{M})$,

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta.$$

Here, ε controls the privacy-accuracy trade-off (smaller values correspond to stronger privacy guarantees), while δ allows for a small probability of failure of the guarantee. In the special case where $\delta = 0$, the guarantee is called *pure* DP.

2.2.2 Mechanisms

Several standard mechanisms achieve differential privacy by introducing carefully calibrated noise:

- **Laplace mechanism**: Adds noise drawn from the Laplace distribution, scaled to the sensitivity of the query and the privacy parameter ε.
- **Gaussian mechanism**: Adds Gaussian noise, typically used for (ε, δ)-DP guarantees.
- **Randomized response**: A local DP technique where individuals perturb their own data before sharing, ensuring plausible deniability.

These mechanisms ensure that the outputs of $\mathcal{M}$ remain statistically similar whether or not a particular individual is included in the dataset.

2.2.3 Central vs. Local DP

Two deployment models of DP are particularly relevant:

- **Central DP**: A trusted curator collects raw data, applies a DP mechanism, and releases privatized outputs. This is widely used in applications such as the U.S. Census.
- **Local DP (LDP)**: Individuals apply the privacy mechanism before sharing their data, so no trusted curator is required. LDP provides stronger guarantees for participants but often results in lower data utility.

2.2.4 Challenges in Genomic Data

While DP has been successfully applied in many domains, applying it to genomic data introduces unique challenges. Genomic datasets are extremely high-dimensional, with millions of correlated SNPs per individual. Standard DP mechanisms often assume independence across attributes, but as discussed in Section 2.1.2, linkage disequilibrium creates strong correlations. Perturbing SNPs independently under DP can therefore destroy essential structure, reducing the utility of the data for downstream tasks such as ancestry inference.

This tension between privacy and utility in the presence of correlated genomic data motivates the exploration of specialized approaches to DP genomics, which will be the focus of Chapter 4.

Chapter 3

Problem Description

3.1 Motivation

The dual challenge of genomic data is that it is both highly informative and highly sensitive. On the one hand, single nucleotide polymorphisms (SNPs) and their correlation structures are crucial for tasks such as disease-gene mapping, ancestry inference, and population genetics. On the other hand, these same data can enable privacy attacks ranging from membership inference in genomic beacons [40] to re-identification of supposedly anonymized individuals [43]. Traditional anonymization and access-control mechanisms have proven insufficient, motivating the adoption of differential privacy (DP) as a formal framework for genomic data protection. Yet, the application of DP to genomics is far from straightforward. Standard DP mechanisms, such as randomized response in the local model, assume independence across attributes. This assumption is fundamentally violated in genomics, where linkage disequilibrium (LD) induces strong correlations among SNPs. Ignoring these correlations can severely degrade the utility of privatized data, undermining downstream applications such as ancestry inference. This raises the core research problem: How can differential privacy be applied to genomic data in a way that balances privacy guarantees with utility preservation, especially in the presence of SNP correlations?

3.2 Problem Statement

Formally, let each individual be represented by a genotype vector $g \in \{0, 1, 2\}^n$, where n is the number of single nucleotide polymorphisms (SNPs). Each entry g_i encodes the allele

count at SNP i:

- 0 denotes a homozygous reference genotype,
- 1 denotes a heterozygous genotype, and
- 2 denotes a homozygous alternate genotype.

Thus, the vector has length n, with one entry per SNP. In practice, n is on the order of hundreds of thousands to millions, reflecting dense coverage across the genome. Missing values (e.g., due to sequencing errors) can be encoded as -1, but are excluded from analysis in this thesis.

We consider a local differential privacy (LDP) setting in which each individual perturbs their genotype vector before sharing it. The objective is to design and evaluate a mechanism M such that:

- **Privacy:** M satisfies ε-local differential privacy.
- **Utility:** The perturbed data preserves sufficient signal to enable ancestry inference, particularly when evaluated using state-of-the-art local ancestry deconvolution methods such as Gnomix [25].
- **Correlation Awareness:** The mechanism accounts for linkage disequilibrium (correlations between SNPs), avoiding the naïve independence assumption that severely compromises downstream utility.

The trade-off between these objectives defines the problem space explored in this thesis.

3.3 Chapter Summary

In summary, the problem addressed in this thesis is the application of differential privacy to genomic data under correlation constraints, with a focus on ancestry inference as the downstream task. The next chapter (Chapter 4) surveys existing literature on DP in genomics, with special attention to dependency-aware approaches, and explains the rationale for selecting Yilmaz et al.'s work as the primary baseline for evaluation.

Chapter 4

Genomics DP

To our knowledge, this thesis pioneers private ancestry deconvolution, and as such, there is no published work exploring ancestry determination with differential privacy. However, it is notable that DP has been explored for domains such as genomics, which we delve into in this chapter.

The sensitive nature of genomic information raises significant privacy challenges in data sharing and analysis. Unlike ordinary health records, DNA sequences uniquely identify individuals and encode information about their relatives, which makes them inherently re-identifiable even after the removal of explicit identifiers. Several works have demonstrated successful re-identification attacks on genomic datasets and genomic data-sharing services, highlighting the limitations of conventional anonymization techniques [27, 21, 40]. In response, researchers have increasingly turned to Differential Privacy (DP) [18] as a rigorous mathematical framework for quantifying and controlling privacy risks. This chapter reviews the development of DP methods tailored for genomic data, highlighting early efforts, methodological advances, and remaining challenges.

4.1 Early Applications of DP in Genomics

The first applications of DP in genomics primarily targeted summary statistics from genome-wide association studies (GWAS). Homer et al. showed that the presence of an individual in a GWAS could be inferred from aggregated allele-frequency data [27], revealing that even coarse-grained genomic summaries can leak membership information. This observation motivated a line of work applying differential privacy as a formal protection

mechanism. Johnson and Shmatikov [31] proposed DP mechanisms for releasing GWAS statistics, such as χ^2 values, while limiting the risk of membership inference. Their results highlight an important structural property of genomic data: GWAS statistics are derived from thousands to millions of SNPs that are correlated through linkage disequilibrium (LD). Because DP mechanisms must add noise proportional to the global sensitivity of the released function, the high dimensionality and correlation structure of genomic data lead to very large sensitivity values. Consequently, a substantial amount of noise is required to satisfy DP, often overwhelming the biological signal. It is in this sense that these early works exposed a curse of dimensionality in genomic DP. Genomes contain millions of features (SNPs), many of which exhibit complex dependencies shaped by population history. When noise is added independently or uniformly across such a large space, utility degrades sharply. Johnson and Shmatikov's contributions [31] and related works make this challenge explicit: while DP can theoretically protect genomic releases, straightforward applications of DP to high-dimensional, correlated genomic statistics are insufficient without additional structure, modeling assumptions, or dimensionality reduction. These limitations set the stage for subsequent methods—including those discussed in later sections—that seek to balance privacy and utility by exploiting LD structure, using more tailored DP mechanisms, or by privatizing specific downstream inferences rather than the full data.

4.2 Genomic Data-Sharing Beacons

Genomic beacons—databases that allow binary queries of the form "Does any individual in the dataset have allele X at position Y?"—were also shown to be vulnerable to re-identification. Shringarpure and Bustamante demonstrated that repeated queries against beacons could allow an attacker to infer whether an individual's genome was present in the dataset [40]. von Thenen et al. extended these attacks by showing that allele frequency inference could further increase re-identification power [43]. As a countermeasure, Ayday proposed differential privacy–based mechanisms for genomic beacons, in which randomized responses are used to introduce uncertainty into query answers and limit information leakage [7]. While effective at mitigating re-identification risk, these approaches highlighted the fundamental trade-off between privacy protection and query utility.

4.3 Global vs. Local DP in Genomic Settings

In most applications, DP is implemented in the global model, where a trusted curator holds the raw data and applies noise before releasing statistics. This paradigm suits GWAS settings where centralized biobanks manage genomic data [31]. However, genomic privacy risks also extend to direct-to-consumer services (e.g., ancestry testing), where individuals may prefer to privatize their own data before submission. This motivates the local DP (LDP) paradigm, in which each individual perturbs their data independently before sharing.

Yilmaz et al. introduced dependent local differential privacy (dLDP) for genomic data, an approach that leverages correlations between single nucleotide polymorphisms (SNPs) to improve utility under LDP [45]. Unlike naïve LDP, which perturbs each SNP independently, dLDP accounts for linkage disequilibrium (LD)—the non-random association of alleles across loci—thereby preserving biologically meaningful correlations. This represents a significant step toward practical LDP mechanisms for genomic data.

4.4 Addressing Dependencies in Genomic Data

A central challenge in applying DP to genomic datasets is the presence of correlations between variants. SNPs are not independent; their correlation structure (linkage disequilibrium) reflects recombination history and population structure. Ignoring these dependencies can cause DP mechanisms to either underestimate privacy risk or inject excessive noise.

Liu et al. showed that when data points are dependent, the effective privacy guarantees degrade compared to the independent case [32]. Almadhoun et al. further formalized differential privacy under dependent tuples, demonstrating how correlations alter the effective sensitivity of queries [6]. Building on this line of work, Yilmaz et al. designed mechanisms that explicitly incorporate SNP dependencies, allowing for improved utility in genomic analyses while maintaining formal DP guarantees [45].

Related ideas have also appeared in other domains, where prior information is used to improve the utility of randomized response mechanisms under DP. For example, Ghazi et al. proposed Randomized Response with Prior (RRWithPrior) for label differential privacy, showing that priors can significantly improve learning performance without weakening privacy guarantees [20]. However, such approaches do not address dependencies among high-dimensional input features. In contrast, genomic data exhibit strong, structured

correlations across millions of SNPs, necessitating DP mechanisms that explicitly model inter-attribute dependencies rather than relying on priors over outputs alone.

4.4.1 The Yilmaz Algorithm for Dependent Local Differential Privacy

Yilmaz et al. proposed a dependent local differential privacy (dLDP) mechanism specifically designed for genomic data [45]. The key insight is that nearby single nucleotide polymorphisms (SNPs) exhibit statistical dependencies due to linkage disequilibrium (LD), and that accounting for these dependencies allows more efficient use of the privacy budget ε. Note that we modify the algorithm to improve scalabilty and efficiency. The full software is presented in Appendix B.

Algorithm 1: Dependent Local Differential Privacy (dLDP) for Genomic Data [45]

Input : SNP sequence $\mathbf{x} = [x_1, x_2, \ldots, x_n]$,
privacy budget ε,
dependency window size w (number of SNPs to condition on)

Output: Perturbed SNP sequence $\mathbf{y} = [y_1, y_2, \ldots, y_n]$

$p \leftarrow \frac{e^{\varepsilon}}{e^{\varepsilon}+2}, \quad q \leftarrow \frac{1-p}{2}$; `// Base probabilities (same as randomized response)`

for $i \leftarrow 1$ **to** n **do**

  $\mathbf{c}_i \leftarrow$ context window $(x_{i-w}, \ldots, x_{i+w})$; `// Capture local LD context`

  Compute conditional probabilities $P(X_i \mid \mathbf{c}_i)$ from reference data;

  `// Adjust response distribution based on context`

  foreach $a \in \{0, 1, 2\}$ **do**

    $P'(a) \leftarrow p \cdot \mathbb{I}[a = x_i] + q \cdot (1 - \mathbb{I}[a = x_i])$;

    $P''(a) \leftarrow \text{Normalize}\big(P'(a) \cdot P(X_i = a \mid \mathbf{c}_i)\big)$;

  Sample $y_i \sim P''(a)$; `// Context-aware perturbation`

return y

Core Idea. Under standard local differential privacy (LDP), each SNP is privatized independently using a randomized response mechanism, which replaces the true genotype x_i with a noisy version $\tilde{x}_i$. This independent perturbation destroys the correlation structure between SNPs, leading to substantial accuracy loss. The Yilmaz algorithm mitigates this

by introducing a *context-aware* perturbation process, where the privatization of each SNP depends on the surrounding genomic context within a sliding window of neighboring SNPs.

Algorithm Overview. Let $X = (x_1, x_2, \ldots, x_n)$ denote an individual's sequence of SNPs, where $x_i \in \{0, 1, 2\}$ encodes the genotype dosage. The algorithm proceeds as follows:

1. **Context modeling:** For each SNP x_i, a context window $W_i = (x_{i-w}, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{i+w})$ of size $2w$ is defined. This window captures local LD relationships.
2. **Probability estimation:** The joint or conditional probabilities $P(x_i \mid W_i)$ are estimated from reference data or inferred from population-level haplotype frequencies.
3. **Context-aware perturbation:** Rather than perturbing x_i directly, the mechanism samples $\tilde{x}_i$ according to a privacy-preserving distribution:

$$P(\tilde{x}_i \mid x_i, W_i) \propto \begin{cases} e^{\varepsilon} P(x_i \mid W_i), & \text{if } \tilde{x}_i = x_i, \\ P(\tilde{x}_i \mid W_i), & \text{otherwise.} \end{cases}$$

   This formulation ensures ε-LDP while weighting perturbations by their contextual likelihoods.
4. **Reconstruction:** The aggregator (or analyst) combines the privatized SNPs $\tilde{X}$ across individuals to estimate true population-level frequencies or perform downstream inference (e.g., ancestry deconvolution).

Interpretation. The Yilmaz mechanism can be viewed as a generalization of randomized response that incorporates a probabilistic prior over neighboring SNPs. By adapting the perturbation probabilities to the local genomic structure, it preserves key LD patterns while maintaining a formal ε-LDP guarantee. This approach dramatically improves utility in downstream tasks such as haplotype frequency estimation and genotype imputation, compared to naïve independent perturbation.

Window Size and Trade-offs. The choice of window size w governs the balance between accuracy and computational cost. Small windows ($w = 1$–5) capture short-range LD and are computationally efficient, while larger windows yield better correlation modeling but incur higher runtime. The optimal window size depends on the linkage structure of the genomic region, ideally capturing the majority of LD without overfitting to noise.

Significance. The Yilmaz algorithm represents one of the first practical frameworks for context-aware LDP in genomics. It demonstrates that privacy mechanisms can be adapted to the inherent structure of biological data, providing both strong privacy guarantees and usable outputs. This dependency-aware view of DP has influenced subsequent works on privacy-preserving haplotype reconstruction, genotype imputation, and ancestry inference.

4.5 DP for Synthetic Genomic Data

An alternative approach is to generate synthetic datasets under DP guarantees. Beaulieu-Jones et al. explored DP-enabled generative neural networks for clinical and genomic-like data [9]. Such approaches allow researchers to release simulated datasets that capture population-level trends without exposing real individuals' genomes. While promising, synthetic data generation under DP remains an active research area, particularly given the complexity of genomic distributions and the need for fidelity in downstream tasks such as ancestry inference and GWAS.

4.6 DP in Ancestry Inference

Accurate ancestry inference relies on subtle correlations between SNPs, which are easily disrupted by DP noise. Traditional ancestry inference tools such as RFMix [33] and the more recent Gnomix [25] achieve high accuracy on unperturbed genomes, but their performance degrades sharply when input data are privatized with naïve DP mechanisms. Jiang et al. benchmarked a range of DP mechanisms in genomic settings, highlighting both the opportunities and limitations of applying DP directly to high-dimensional genomic data [30]. While dependency-aware DP mechanisms have been proposed to better preserve genomic correlations, their impact on downstream ancestry deconvolution has not been systematically evaluated. This thesis addresses this gap by providing the first empirical assessment of local ancestry inference accuracy under correlation-aware local DP.

4.7 Challenges and Future Directions

Despite promising progress, several challenges remain in applying DP to genomics:

- **Utility loss:** Adding DP noise to millions of SNPs can render data nearly useless without correlation-aware methods.
- **Correlation modeling:** Accurately capturing LD structure is essential yet computationally demanding.
- **Interpretation of ε:** In genomic contexts, understanding what values of ε correspond to acceptable privacy risk is still unresolved [16].
- **Scalability:** Current methods struggle to handle biobank-scale data with hundreds of thousands of genomes.
- **Integration with other privacy methods:** Hybrid approaches combining DP with secure multiparty computation or homomorphic encryption may provide stronger guarantees.

4.8 Summary

Differential Privacy (DP) has emerged as a promising solution for protecting genomic data against re-identification risks. While early applications focused on genome-wide association study (GWAS) statistics and beacon queries, recent work has proposed dependency-aware DP mechanisms that explicitly account for genomic correlations, significantly improving utility. Synthetic data generation and ancestry inference under DP remain active research areas. In this thesis, we focus specifically on challenges related to preserving linkage disequilibrium structure under local DP and evaluating the resulting privacy–utility trade-offs in ancestry deconvolution. Issues such as large-scale deployment, federated data sharing, and user-facing interpretation of privacy budgets are acknowledged but lie beyond the scope of this work.

Chapter 5

Evaluation

In this chapter, we evaluate the utility of applying local differential privacy (LDP) to genotype data prior to ancestry inference. The downstream task we study is local ancestry deconvolution using Gnomix [25], a state-of-the-art machine learning system described in Section 2.1.2. Because Gnomix relies directly on phased SNP sequences and the fine-scale linkage disequilibrium (LD) structure they encode, it provides a stringent test of whether privatized genomic data retain sufficient utility for population-genetic inference.

Our experiments use the *pretrained chromosome 22* Gnomix models provided by the authors [25]. We do *not* retrain Gnomix: training requires large reference panels and substantial computational resources, and retraining on privatized data is neither feasible nor meaningful for this study. Instead, we compare Gnomix's predictions on (i) unmodified genotype data and (ii) LDP-perturbed genotype data created by applying our mechanism to the sample dataset distributed with the Gnomix software.

5.1 Experimental Setup

5.1.1 Dataset

Our evaluation uses the Phase 3 release of the 1000 Genomes dataset [15], which provides phased haplotypes for 2,504 individuals across 22 autosomes. Gnomix itself was trained by the authors on haplotypes derived from the same reference panel, and their publicly released pretrained model for chromosome 22 is used for all predictions in this chapter.

While commercial systems such as 23andMe's Ancestry Composition algorithm [17] train on millions of samples, using 1000 Genomes offers full transparency and reproducibility. All analyses are conducted on GRCh37 (b37), matching the reference used by the authors of Gnomix.

5.1.2 Preprocessing

Each VCF file was converted into diploid genotype vectors encoded as $\{0, 1, 2\}$, corresponding to homozygous reference, heterozygous, and homozygous alternate states. Missing values were resolved through imputation and phasing with Beagle [13] to match the input format expected by Gnomix's pretrained models, which operate on phased SNP sequences.

For compatibility with our LDP pipeline, each individual-level genotype array was serialized to `.pkl` format. Perturbed datasets were produced by applying our LDP mechanism directly to these arrays. The same per-individual files were used for both unperturbed and perturbed evaluation for controlled comparison.

5.1.3 Privacy Mechanism

We apply a randomized-response LDP mechanism to each SNP genotype. Formally, each value in $\{0, 1, 2\}$ is perturbed according to a probability distribution calibrated by the privacy parameter ε. Smaller values of ε produce distributions closer to uniform (stronger privacy, higher noise), while larger values concentrate probability mass on the true value (weaker privacy, higher utility).

We evaluate the following range:

$$\varepsilon \in \{0.01, 0.1, 0.5, 1, 2, 5, 10, 25\}.$$

For each value of ε, Gnomix is run *unchanged* on the resulting perturbed dataset. Because the model expects phased haplotypes with realistic LD structure, this setup allows us to quantify directly how LDP noise disrupts the information Gnomix relies on.

5.1.4 Accuracy Definition

Gnomix outputs, for each genomic window along chromosome 22, a probability distribution over a fixed set of ancestry populations. Let K denote the number of ancestry labels, and

let

$$\mathbf{p}_{i,w} \in [0,1]^K$$

denote the predicted ancestry probability vector for individual i at genomic window w, with $\sum_{k=1}^{K} p_{i,w}^{(k)} = 1$.

For each window, Gnomix also produces a hard ancestry assignment given by the maximum-probability label:

$$\hat{y}_{i,w} = \arg \max_{k \in \{1,\ldots,K\}} p_{i,w}^{(k)}.$$

Because ground-truth local ancestry labels are not available for the evaluation dataset, we define accuracy relative to the predictions on unperturbed genotype data. Specifically, let $\hat{y}_{i,w}^{\text{orig}}$ denote the predicted ancestry label for individual i at window w when using unmodified genotypes, and let $\hat{y}_{i,w}^{\text{LDP}}$ denote the corresponding label when using LDP-perturbed genotypes.

We define accuracy as the fraction of genomic windows for which the predicted ancestry label is preserved under perturbation:

$$\text{Accuracy} = \frac{1}{NW} \sum_{i=1}^{N} \sum_{w=1}^{W} \mathbb{I}\left[\hat{y}_{i,w}^{\text{LDP}} = \hat{y}_{i,w}^{\text{orig}}\right],$$

where N is the number of individuals, W is the number of genomic windows along chromosome 22, and $\mathbb{I}[\cdot]$ denotes the indicator function.

Under this definition, an accuracy of 1.0 indicates that ancestry assignments are identical to the non-private baseline across all windows, while lower values quantify the extent to which LDP noise alters downstream ancestry inference.

5.2 Results

Figure 5.1 shows ancestry deconvolution accuracy as a function of ϵ, alongside the non-private baseline.

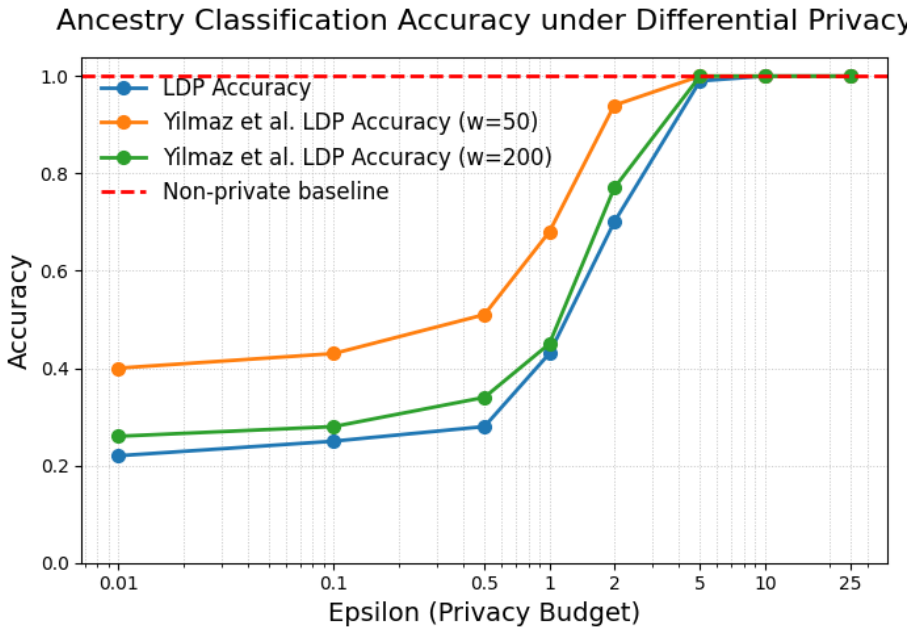

Figure 5.1: Accuracy of ancestry deconvolution under naïve and correlation-aware local differential privacy at varying ϵ values, compared against the non-private baseline (accuracy = 1.0).

Table 5.1 reports the exact accuracy values.

Table 5.1: Ancestry inference accuracy under local differential privacy for varying privacy budgets ϵ. Baseline accuracy without privacy is 1.0.

ϵ	0.01	0.1	0.5	1	2	5	10	25
Naïve LDP	0.22	0.25	0.28	0.43	0.70	0.99	1.00	1.00
dLDP ($w = 50$)	0.40	0.43	0.51	0.68	0.94	1.00	1.00	1.00
dLDP ($w = 200$)	0.26	0.28	0.34	0.45	0.77	1.00	1.00	1.00

The results demonstrate a strong dependence of ancestry inference accuracy on the privacy parameter ϵ, as well as on the choice of privacy mechanism. Under naïve LDP, accuracy collapses to near-random levels for very small privacy budgets ($\epsilon = 0.01, 0.1$), achieving only 22–25% accuracy. At intermediate values of ϵ ($0.5 \leq \epsilon \leq 2$), naïve perturbation yields gradual improvements but remains substantially below the non-private

baseline, reflecting the sensitivity of ancestry inference to the disruption of SNP correlations.

In contrast, the correlation-aware dLDP mechanism of Yilmaz *et al.* consistently outperforms naïve LDP across all privacy regimes. At low ϵ, dLDP with a window size of $w = 50$ nearly doubles classification accuracy relative to naïve perturbation, demonstrating that incorporating local linkage disequilibrium structure substantially mitigates the utility loss induced by strong privacy constraints. At intermediate ϵ, dLDP rapidly approaches baseline performance, achieving over 90% accuracy by $\epsilon = 2$, while naïve LDP remains far below this level.

The choice of dependency window size further affects performance. A moderate window size ($w = 50$) provides the strongest gains at low to intermediate privacy budgets, while a larger window ($w = 200$) yields more modest improvements. This suggests a trade-off between capturing sufficient local correlation structure and introducing additional estimation noise from higher-dimensional dependency models. For sufficiently large ϵ ($\epsilon \geq 5$), all mechanisms converge to baseline accuracy, indicating that privacy noise becomes negligible in this regime and differences between methods vanish.

5.3 Discussion

5.3.1 Privacy–Utility Trade-off

The evaluation confirms the central trade-off of differential privacy: strong privacy comes at the expense of utility. At small ϵ, ancestry inference becomes unreliable, suggesting that practical deployment with such stringent privacy guarantees would be infeasible. At moderate values (e.g., $\epsilon = 1$ or $\epsilon = 2$), the model retains partial ancestry signal, offering a compromise between privacy and accuracy. At larger ϵ, the protection is effectively diminished and accuracy is indistinguishable from non-private inference.

5.3.2 Limitations

Several limitations qualify the findings:

- **Dataset size and diversity.** The 1000 Genomes dataset, while widely used, is small compared to commercial panels and does not fully capture global population diversity.

- **Model assumptions.** Gnomix relies on phased haplotypes, while our encoding and perturbation operate at the diploid genotype level. This may reduce effective ancestry resolution.
- **Imputation bias.** Beagle-based imputation may introduce biases or artifacts that interact with privacy noise in unpredictable ways.
- **Reference genome build.** Our reliance on GRCh37 ensures compatibility with Gnomix but limits generalization, as many contemporary datasets and tools use GRCh38.

5.3.3 Broader Implications

These results highlight the tension between privacy and utility in genomic analysis. While differential privacy offers formal guarantees, strong protection can render ancestry inference unusable. This raises practical questions for consumer genomics and biomedical applications: moderate privacy budgets may provide acceptable compromises, but stringent privacy regimes could prevent meaningful inference. Future work should investigate whether alternative mechanisms (e.g., federated learning with secure aggregation, or model-level DP) can offer stronger protection without such severe utility loss.

5.4 Preliminary Reidentification Attack

While the primary focus of this chapter is the utility of LDP-perturbed genotypes for ancestry deconvolution, we additionally conduct an exploratory empirical evaluation of reidentification risk. The purpose of this analysis is not to challenge the formal guarantees of local differential privacy (LDP), but to assess whether simple, naïve attacks can succeed in small-cohort settings relevant to ancestry inference.

5.4.1 Attack Model

We consider a basic reidentification scenario in which an adversary is given:

- A candidate pool of $M = 9$ individuals, each with access to their original (unperturbed) genotype vectors.

- A single privatized genotype vector produced using either naïve randomized response or the correlation-aware LDP mechanism inspired by Yilmaz [45].

The candidate individuals are drawn uniformly at random from the same evaluation subset of the 1000 Genomes Phase 3 dataset used throughout this chapter. Exactly one candidate corresponds to the individual whose genotype has been privatized; the remaining $M - 1$ individuals act as decoys.

Given a privatized genotype vector $\tilde{\mathbf{x}}$ and candidate original genotype vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$, the adversary computes a similarity score between $\tilde{\mathbf{x}}$ and each candidate using the ℓ_1 distance:

$$d(\tilde{\mathbf{x}}, \mathbf{x}_j) = \|\tilde{\mathbf{x}} - \mathbf{x}_j\|_1.$$

The adversary predicts the identity of the target individual as:

$$\hat{j} = \arg \min_{j \in \{1,\ldots,M\}} d(\tilde{\mathbf{x}}, \mathbf{x}_j).$$

A reidentification attempt is considered successful if $\hat{j}$ corresponds to the true source individual.

This threat model is intentionally simplistic. It assumes full access to the original genotypes of all candidates but does not exploit auxiliary metadata, population-level statistics, linkage disequilibrium models, or more sophisticated inference techniques. As such, it should be interpreted as a lower bound on empirical reidentification risk in constrained settings rather than a realistic adversarial model.

5.4.2 Results

Tables 5.2 and 5.3 report reidentification success rates for naïve randomized response and correlation-aware LDP, respectively, across a range of privacy budgets ε. Figure 5.2 visualizes the success rate as a function of ε.

ε	Total	Correct	Success Rate
0.01	9	4	0.44
0.1	9	9	1.00
0.5	9	9	1.00
1	9	9	1.00
2	9	9	1.00
5	9	9	1.00
10	9	9	1.00
25	9	9	1.00

Table 5.2: Reidentification attack success rates under naïve randomized response.

ε	Total	Correct	Success Rate
0.01	9	9	1.00
0.1	9	9	1.00
0.5	9	9	1.00
1	9	9	1.00
2	9	9	1.00
5	9	9	1.00
10	9	9	1.00
25	9	9	1.00

Table 5.3: Reidentification attack success rates under correlation-aware LDP.

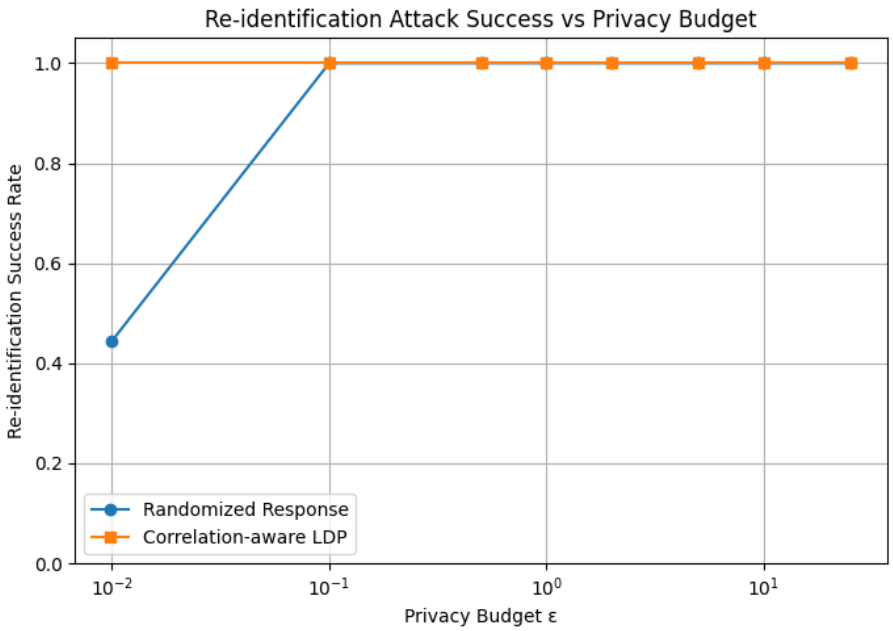

Figure 5.2: Reidentification attack success rate as a function of the privacy budget ε for naïve randomized response and correlation-aware local differential privacy.

Across both mechanisms, reidentification success rapidly approaches 100% as ε increases. Notably, the correlation-aware LDP mechanism achieves perfect reidentification across all evaluated privacy budgets, including extremely small values of ε.

5.4.3 Discussion

These results indicate that, in small candidate pools, simple distance-based attacks can be highly effective against both perturbation mechanisms. While naïve randomized response provides partial empirical protection at very small privacy budgets, this protection degrades rapidly as ε increases. The correlation-aware mechanism, while preserving substantial utility for ancestry deconvolution, does not empirically prevent reidentification in this constrained setting.

These findings should be interpreted with care. Differential privacy provides worst-case, distribution-independent guarantees that bound information leakage but do not preclude successful inference in specific finite-sample scenarios. Observing high reidentification accuracy in this experiment therefore does not invalidate the formal guarantees of LDP.

Nevertheless, the results highlight a practical tension between utility and empirical privacy. Mechanisms that preserve sufficient genomic structure for accurate ancestry inference may simultaneously expose individuals to reidentification risk in small cohorts. In the context of ancestry deconvolution, this suggests that local differential privacy alone may be insufficient to provide meaningful protection against identity disclosure, motivating the exploration of complementary or hybrid privacy approaches.

Chapter 6

Conclusion

This thesis has examined the interaction between *genomic privacy* and *differential privacy (DP)* in the context of ancestry deconvolution. Genomic data poses unique challenges for privacy preservation: it is permanent, uniquely identifying, and characterized by strong correlations arising from linkage disequilibrium. These properties complicate the application of conventional privacy mechanisms and motivate careful empirical evaluation when differential privacy is applied to real genomic inference tasks.

The central contribution of this thesis is an empirical study of ancestry deconvolution under local differential privacy (LDP). Using phased haplotypes from the 1000 Genomes Project, imputation and phasing via Beagle, and ancestry inference with the state-of-the-art Gnomix model, we constructed an end-to-end pipeline that enables direct comparison between non-private inference and inference on locally privatized genotype data. Within this framework, we implemented and evaluated both naïve randomized response and a correlation-aware LDP mechanism inspired by the work of Yilmaz *et al.*, which explicitly incorporates dependencies between single nucleotide polymorphisms (SNPs).

Our research hypothesis proposed that LDP enables a tunable privacy–utility trade-off in ancestry deconvolution, and that mechanisms accounting for genomic correlations would preserve inference accuracy across a wider range of privacy budgets than naïve, independent perturbation. The experimental results support this hypothesis. Naïve randomized response, which treats SNPs as independent, caused ancestry deconvolution accuracy to degrade sharply at moderate and small privacy budgets. In contrast, the correlation-aware mechanism preserved substantially higher accuracy, recovering near-baseline performance at larger values of ε. These findings demonstrate that in high-dimensional genomic settings, utility loss under LDP is governed not only by the privacy budget but also by how

effectively a mechanism leverages biological structure such as linkage disequilibrium.

However, this improvement in utility comes with important caveats. The exploratory reidentification analysis presented shows that even correlation-aware LDP mechanisms can permit successful reidentification in small candidate pools across a wide range of privacy budgets. While this does not contradict the formal guarantees of differential privacy—which are worst-case and attack-agnostic it highlights a gap between theoretical privacy assurances and empirical resistance to simple matching attacks in constrained settings. Taken together, these results suggest that mechanisms capable of preserving high ancestry inference accuracy under LDP may still offer limited practical protection against reidentification, particularly in small or structured populations.

Contributions

The contributions of this thesis are as follows:

1. A clear formulation of ancestry deconvolution as a downstream genomic task under local differential privacy, emphasizing the role of linkage disequilibrium and SNP correlations.
2. An end-to-end experimental pipeline integrating public genomic datasets, modern imputation and ancestry inference tools, and LDP mechanisms for systematic evaluation.
3. An empirical comparison of naïve and correlation-aware LDP mechanisms, quantifying their impact on ancestry deconvolution accuracy and illustrating the resulting privacy–utility trade-offs.

Future Work

This work suggests several directions for future research:

- **Stronger privacy mechanisms:** Developing DP mechanisms that exploit genomic correlations while offering improved empirical protection against reidentification, particularly in small cohorts.

- **Alternative privacy models:** Investigating hybrid approaches that combine local and central DP, secure aggregation, or federated learning to balance deployability with stronger privacy guarantees.
- **Expanded evaluation:** Extending analyses to larger and more diverse datasets, additional chromosomes, and higher-resolution ancestry models to assess robustness at scale.
- **Broader genomic tasks:** Evaluating the impact of DP on other downstream analyses such as polygenic risk scoring or genome-wide association studies, where privacy–utility trade-offs may differ.

Closing Remarks

Genomic data holds enormous promise for advancing science, medicine, and personal understanding of ancestry, yet its misuse poses significant ethical and societal risks. This thesis demonstrates that while local differential privacy can be tuned to preserve substantial utility for ancestry deconvolution—especially when genomic correlations are explicitly modeled—current mechanisms face fundamental limitations in providing strong practical protection against reidentification. Bridging this gap between theoretical privacy guarantees and empirical privacy risk remains a critical challenge. Addressing it will be essential if privacy-preserving genomic analysis is to be deployed responsibly in real-world settings.

References

[1] Differential Privacy Protection Against Membership Inference Attack on Machine Learning for Genomic Data.

[2] Electronics | Free Full-Text | Blockchain-Based Privacy-Preserving System for Genomic Data Management Using Local Differential Privacy.

[3] 23andMe: Profiles of 6.9 million people hacked. December 2023.

[4] Dozens of states sue to block the sale of 23andMe personal genetic data without customer consent, June 2025. Section: U.S. News.

[5] Portal Admin. Genetic Testing: Ancestry Interest, But Privacy Concerns - AP-NORC, July 2018.

[6] Nour Almadhoun, Erman Ayday, and Özgür Ulusoy. Differential privacy under dependent tuples—the case of genomic privacy. *Bioinformatics*, 36(6):1696–1703, March 2020.

[7] Erman Ayday. Privacy Challanges of Genomic Data-Sharing Beacons and Solutions | Case School of Engineering.

[8] Franz Baumdicker, Gertjan Bisschop, Daniel Goldstein, Graham Gower, Aaron P Ragsdale, Georgia Tsambos, Sha Zhu, Bjarki Eldon, E Castedo Ellerman, Jared G Galloway, Ariella L Gladstein, Gregor Gorjanc, Bing Guo, Ben Jeffery, Warren W Kretzschumar, Konrad Lohse, Michael Matschiner, Dominic Nelson, Nathaniel S Pope, Consuelo D Quinto-Cortés, Murillo F Rodrigues, Kumar Saunack, Thibaut Sellinger, Kevin Thornton, Hugo van Kemenade, Anthony W Wohns, Yan Wong, Simon Gravel, Andrew D Kern, Jere Koskela, Peter L Ralph, and Jerome Kelleher. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*, 220(3):iyab229, March 2022.

[9] Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P. Bhavnani, James Brian Byrd, and Casey S. Greene. Privacy-Preserving Generative Deep Neural Networks Support Clinical Data Sharing. *Circulation. Cardiovascular Quality and Outcomes*, 12(7):e005122, July 2019.

[10] Saurabh Belsare, Michal Levy-Sakin, Yulia Mostovoy, Steffen Durinck, Subhra Chaudhuri, Ming Xiao, Andrew S. Peterson, Pui-Yan Kwok, Somasekar Seshagiri, and Jeffrey D. Wall. Evaluating the quality of the 1000 genomes project data. *BMC Genomics*, 20(1):620, August 2019.

[11] Robert Booth and Robert Booth UK technology editor. DNA testing firm 23andMe fined £2.3m by UK regulator for 2023 data hack. *The Guardian*, June 2025.

[12] Brian L. Browning, Xiaowen Tian, Ying Zhou, and Sharon R. Browning. Fast two-stage phasing of large-scale sequence data. *The American Journal of Human Genetics*, 108(10):1880–1890, October 2021.

[13] Brian L. Browning, Ying Zhou, and Sharon R. Browning. A One-Penny Imputed Genome from Next-Generation Reference Panels. *The American Journal of Human Genetics*, 103(3):338–348, September 2018.

[14] Erika Check Hayden. The rise and fall and rise again of 23andMe. *Nature*, 550(7675):174–177, October 2017.

[15] Laura Clarke, Xiangqun Zheng-Bradley, Richard Smith, Eugene Kulesha, Chunlin Xiao, Iliana Toneva, Brendan Vaughan, Don Preuss, Rasko Leinonen, Martin Shumway, Stephen Sherry, and Paul Flicek. The 1000 Genomes Project: data management and community access. *Nature Methods*, 9(5):459–462, May 2012.

[16] Rachel Cummings, Damien Desfontaines, David Evans, Roxana Geambasu, Yangsibo Huang, Matthew Jagielski, Peter Kairouz, Gautam Kamath, Sewoong Oh, Olga Ohrimenko, Nicolas Papernot, Ryan Rogers, Milan Shen, Shuang Song, Weijie Su, Andreas Terzis, Abhradeep Thakurta, Sergei Vassilvitskii, Yu-Xiang Wang, Li Xiong, Sergey Yekhanin, Da Yu, Huanyu Zhang, and Wanrong Zhang. Advancing Differential Privacy: Where We Are Now and Future Directions for Real-World Deployment, March 2024. arXiv:2304.06929 [cs].

[17] Eric Y. Durand, Chuong B. Do, Peter R. Wilton, Joanna L. Mountain, Adam Auton, G. David Poznik, and J. Michael Macpherson. A scalable pipeline for local ancestry inference using tens of thousands of reference haplotypes, January 2021.

[18] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating Noise to Sensitivity in Private Data Analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer.

[19] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 1054–1067, Scottsdale Arizona USA, November 2014. ACM.

[20] Badih Ghazi, Noah Golowich, Ravi Kumar, Pasin Manurangsi, and Chiyuan Zhang. Deep Learning with Label Differential Privacy, October 2021. arXiv:2102.06062 [cs].

[21] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science (New York, N.Y.)*, 339(6117):321–324, January 2013.

[22] Benjamin C Haller and Philipp W Messer. SLiM 3: Forward Genetic Simulations Beyond the Wright–Fisher Model. *Molecular Biology and Evolution*, 36(3):632–637, March 2019.

[23] Benjamin C. Haller and Philipp W. Messer. SLiM 4: Multispecies Eco-Evolutionary Modeling. *The American Naturalist*, 201(5):E127–E139, May 2023.

[24] Zaobo He, Yingshu Li, Ji Li, Kaiyang Li, Qing Cai, and Yi Liang. Achieving differential privacy of genomic data releasing via belief propagation. *Tsinghua Science and Technology*, 23(4):389–395, August 2018.

[25] Helgi Hilmarsson, Arvind S. Kumar, Richa Rastogi, Carlos D. Bustamante, Daniel Mas Montserrat, and Alexander G. Ioannidis. High Resolution Ancestry Deconvolution for Next Generation Genomic Data, September 2021. Pages: 2021.09.19.460980 Section: New Results.

[26] Robin J. Hofmeister, Diogo M. Ribeiro, Simone Rubinacci, and Olivier Delaneau. Accurate rare variant phasing of whole-genome and whole-exome sequencing data in the UK Biobank. *Nature Genetics*, 55(7):1243–1249, July 2023.

[27] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V. Pearson, Dietrich A. Stephan, Stanley F. Nelson, and David W. Craig. Resolving Individuals Contributing Trace Amounts of DNA to Highly Complex Mixtures Using High-Density SNP Genotyping Microarrays. *PLoS Genetics*, 4(8):e1000167, August 2008.

[28] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. Reconciling Utility with Privacy in Genomics. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 11–20, Scottsdale Arizona USA, November 2014. ACM.

[29] Tianxi Ji, Erman Ayday, Emre Yilmaz, and Pan Li. Robust fingerprinting of genomic databases. *Bioinformatics*, 38(Supplement_1):i143–i152, June 2022.

[30] Bo Jiang, Mohamed Seif, Ravi Tandon, and Ming Li. Answering Count Queries for Genomic Data With Perfect Privacy. *IEEE Transactions on Information Forensics and Security*, 18:3862–3875, 2023.

[31] Aaron Johnson and Vitaly Shmatikov. Privacy-Preserving Data Exploration in Genome-Wide Association Studies. *KDD : proceedings / International Conference on Knowledge Discovery & Data Mining. International Conference on Knowledge Discovery & Data Mining*, 2013:1079–1087, August 2013.

[32] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence Makes You Vulnerable: Differential Privacy Under Dependent Tuples. In *Proceedings 2016 Network and Distributed System Security Symposium*, San Diego, CA, 2016. Internet Society.

[33] Brian K. Maples, Simon Gravel, Eimear E. Kenny, and Carlos D. Bustamante. RFMix: A Discriminative Modeling Approach for Rapid and Robust Local-Ancestry Inference. *American Journal of Human Genetics*, 93(2):278–288, August 2013.

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

[35] Rhana Natour. A father took an at-home DNA test. His son was then falsely accused of murder, November 2019. Section: Science.

[36] Young-Hoon Park, Yejin Kim, and Junho Shim. Blockchain-Based Privacy-Preserving System for Genomic Data Management Using Local Differential Privacy. *Electronics*, 10(23):3019, January 2021. Number: 23.

[37] The Associated Press. Dozens of states sue to block the sale of 23andMe personal genetic data. *NPR*, June 2025.

[38] Caryn Kseniya Rubanovich, Riley Taitingfong, Cynthia Triplett, Ondrej Libiger, Nicholas J. Schork, Jennifer K. Wagner, and Cinnamon S. Bloss. Impacts of personal DNA ancestry testing. *Journal of Community Genetics*, 12(1):37–52, January 2021.

[39] Wei Shi, Xiaolei Zhang, Hao Chen, and Xing Zhang. High Dimensional Data Differential Privacy Protection Publishing Method Based on Association Analysis. *Electronics*, 12(13):2779, January 2023.

[40] Suyash S. Shringarpure and Carlos D. Bustamante. Privacy Risks from Genomic Data-Sharing Beacons. *American Journal of Human Genetics*, 97(5):631–646, November 2015.

[41] Florian Tramèr, Zhicong Huang, Jean-Pierre Hubaux, and Erman Ayday. Differential Privacy with Bounded Priors: Reconciling Utility and Privacy in Genome-Wide Association Studies. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1286–1297, New York, NY, USA, October 2015. Association for Computing Machinery.

[42] Elisabeth M. van Leeuwen, Alexandros Kanterakis, Patrick Deelen, Mathijs V. Kattenberg, P. Eline Slagboom, Paul I. W. de Bakker, Cisca Wijmenga, Morris A. Swertz, Dorret I. Boomsma, Cornelia M. van Duijn, Lennart C. Karssen, and Jouke Jan Hottenga. Population-specific genotype imputations using minimac or IMPUTE2. *Nature Protocols*, 10(9):1285–1296, September 2015.

[43] Nora von Thenen, Erman Ayday, and A. Ercument Cicek. Re-identification of individuals in genomic data-sharing beacons via allele inference. *Bioinformatics (Oxford, England)*, 35(3):365–371, February 2019.

[44] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, Anima Anandkumar, Karianne Bergen, Carla P. Gomes, Shirley Ho, Pushmeet Kohli, Joan Lasenby, Jure Leskovec, Tie-Yan Liu, Arjun Manrai, Debora Marks, Bharath Ramsundar, Le Song, Jimeng Sun, Jian Tang, Petar Veličković, Max Welling, Linfeng Zhang, Connor W. Coley, Yoshua Bengio, and Marinka Zitnik. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, August 2023. Number: 7972.

[45] Emre Yilmaz, Tianxi Ji, Erman Ayday, and Pan Li. Genomic Data Sharing under Dependent Local Differential Privacy, February 2021. arXiv:2102.07357 [cs].

APPENDICES

Appendix A

Vectorized LDP SNP Perturbation

```
import numpy as np
import pickle
import os
from concurrent.futures import ProcessPoolExecutor

# ======================================================
# Vectorized LDP SNP perturbation
# ======================================================
def perturb_snp_vector_vectorized(snp_vector, epsilon):
    """
    Apply Local Differential Privacy to a SNP vector using randomized response.
    """
    snp_vector = np.array(snp_vector, dtype=np.int8)
    p = np.exp(epsilon) / (np.exp(epsilon) + 2)
    q = (1 - p) / 2

    choices = np.array([0, 1, 2])
    perturbed = np.empty_like(snp_vector)

    mask0 = snp_vector == 0
    mask1 = snp_vector == 1
    mask2 = snp_vector == 2
    mask_other = ~(mask0 | mask1 | mask2)  # -1 or missing

    if mask0.any():
        perturbed[mask0] = np.random.choice(choices, size=mask0.sum(), p=[p, q, q])
    if mask1.any():
        perturbed[mask1] = np.random.choice(choices, size=mask1.sum(), p=[q, p, q])
    if mask2.any():
```

```
        perturbed[mask2] = np.random.choice(choices, size=mask2.sum(), p=[q, q, p])
    if mask_other.any():
        perturbed[mask_other] = snp_vector[mask_other]

    return perturbed

# =====================================================
# Load .pkl → numeric array
# =====================================================
def load_snp_pkl(file_path):
    """
    Load a pickled SNP vector, converting either string or numeric
    representation to np.int8 array with -1 for missing.
    """
    with open(file_path, "rb") as f:
        snp_data = pickle.load(f)

    snp_array = []
    for c in snp_data:
        if isinstance(c, str):
            snp_array.append(int(c) if c in '012' else -1)
        else:
            snp_array.append(int(c) if c in (0, 1, 2) else -1)
    return np.array(snp_array, dtype=np.int8)

# =====================================================
# Save numeric array → .pkl string
# =====================================================
def save_snp_pkl(snp_vector, out_path):
    """
    Convert numeric SNP vector to string format and pickle it.
    """
    out_string = ''.join(str(x) if x >= 0 else 'N' for x in snp_vector)
    with open(out_path, "wb") as f:
        pickle.dump(out_string, f)

# =====================================================
# Process one epsilon
# =====================================================
def process_epsilon(epsilon, input_dir, output_dir):
    epsilon_dir = os.path.join(output_dir, f"eps{epsilon}")
    os.makedirs(epsilon_dir, exist_ok=True)
    print(f"\nProcessing  = {epsilon:.4f}, output dir: {epsilon_dir}")

    for fname in os.listdir(input_dir):
```

```
        if not fname.endswith(".pkl"):
            continue

        input_path = os.path.join(input_dir, fname)
        snp_vector = load_snp_pkl(input_path)
        perturbed_vector = perturb_snp_vector_vectorized(snp_vector, epsilon)

        output_path = os.path.join(
            epsilon_dir,
            fname.replace(".pkl", "_perturbed.pkl")
        )
        save_snp_pkl(perturbed_vector, output_path)
        print(f"Saved perturbed SNPs: {output_path}")

# ======================================================
# Main function
# ======================================================
def main(input_dir, output_dir, epsilon_values, max_workers=None):
    if not os.path.exists(input_dir):
        raise FileNotFoundError(f"Input directory not found: {input_dir}")

    # Parallel processing over epsilon values
    with ProcessPoolExecutor(max_workers=max_workers) as executor:
        futures = [
            executor.submit(process_epsilon, eps, input_dir, output_dir)
            for eps in epsilon_values
        ]
        for future in futures:
            future.result()  # Wait and raise exceptions if any

# ======================================================
# Run
# ======================================================
if __name__ == "__main__":
    input_dir = r"-\snps"  # parent dir with .pkl files
    output_dir = r"-\snps_ldp"
    epsilon_values = [0.01, 0.1, 0.5, 1, 2, 5, 10, 25]

    main(input_dir, output_dir, epsilon_values, max_workers=None)
```

Appendix B

Scalable dLDP SNP Perturbation

```
import numpy as np
import pickle
import os
from concurrent.futures import ProcessPoolExecutor

# -------------------- share_snp --------------------
def share_snp(snp_true, snp_idx, eliminated_state, p):
    q = (1 - p) / 2
    a_snp = snp_true[snp_idx]

    if len(eliminated_state) == 3 or len(eliminated_state) == 0:
        if a_snp == 0:
            obv = np.random.choice([0,1,2], p=[p,q,q])
        elif a_snp == 1:
            obv = np.random.choice([0,1,2], p=[q,p,q])
        elif a_snp == 2:
            obv = np.random.choice([0,1,2], p=[q,q,p])
        else:
            obv = -1
        return obv

    if len(eliminated_state) == 2:
        if a_snp in eliminated_state:
            if a_snp == 0:
                obv = 1 if 2 in eliminated_state else 2
            elif a_snp == 1:
                obv = 2 if 0 in eliminated_state else 0
            elif a_snp == 2:
                obv = 1 if 0 in eliminated_state else 0
```

```
            else:
                obv = a_snp
            return obv

        if len(eliminated_state) == 1:
            elim = eliminated_state[0]
            if a_snp != elim:
                obv = a_snp
            else:
                obv = np.random.choice([x for x in [0,1,2] if x != a_snp])
            return obv

        raise ValueError("Unexpected eliminated_state length.")

    # -------------------- load/save --------------------
    def load_snp_pkl(file_path):
        """
        Load a pickled SNP vector, converting either string or numeric representation
        to np.int8 array with -1 for missing.
        """
        with open(file_path, "rb") as f:
            snp_data = pickle.load(f)

        # Convert all elements to np.int8, handling both string and numeric
        snp_array = []
        for c in snp_data:
            if isinstance(c, str):
                snp_array.append(int(c) if c in '012' else -1)
            else:
                snp_array.append(int(c) if c in (0, 1, 2) else -1)
        return np.array(snp_array, dtype=np.int8)

    def save_snp_pkl(snp_vector, out_path):
        out_string = ''.join(str(x) if x>=0 else 'N' for x in snp_vector)
        with open(out_path,"wb") as f:
            pickle.dump(out_string, f)

    # -------------------- on-the-fly correlation-aware perturbation --------------------
    def perturb_snp_correlation_window(snp_vector, epsilon, window_size=50):
        """
        Apply correlation-aware LDP using a sliding window approach.
        Only correlations within ±window_size positions are considered.
        """
        p = np.exp(epsilon) / (np.exp(epsilon) + 2)
        num_snps = len(snp_vector)
```

```
    perturbed = np.empty_like(snp_vector)

    for i in range(num_snps):
        start = max(0, i - window_size)
        end = min(num_snps, i + window_size + 1)

        # Only consider non-missing SNPs
        window_snps = snp_vector[start:end]
        mask_valid = window_snps >= 0

        # Simple correlation approximation: eliminate alleles that do not occur in
        ↪ window
        eliminated_state = []
        for allele in [0,1,2]:
            if not np.any(window_snps[mask_valid] == allele):
                eliminated_state.append(allele)

        perturbed[i] = share_snp(snp_vector, i, eliminated_state, p)

    return perturbed

# -------------------- process a single file --------------------
def process_file(file_path, output_dir, epsilon, window_size):
    snp_vector = load_snp_pkl(file_path)
    perturbed_vector = perturb_snp_correlation_window(snp_vector, epsilon, window_size)
    out_path = os.path.join(output_dir, os.path.basename(file_path).replace(".pkl",
    ↪ f"_eps{epsilon}.pkl"))
    save_snp_pkl(perturbed_vector, out_path)
    print(f"Saved {out_path}")

# -------------------- main pipeline --------------------
def main(input_dir, output_dir, epsilon_values, window_size=50):
    for eps in epsilon_values:
        eps_dir = os.path.join(output_dir, f"eps{eps}")
        os.makedirs(eps_dir, exist_ok=True)

        print(f"\n--- Processing with epsilon={eps} ---")
        files = [os.path.join(input_dir, f) for f in os.listdir(input_dir) if
        ↪ f.endswith(".pkl")]

        for f in files:
            try:
                print(f"Processing {f} ...")
                process_file(f, eps_dir, eps, window_size)
            except Exception as e:
```

```
                print(f"Error in file {f} with epsilon={eps}: {e}")
                raise  # stop immediately on error for debugging

# -------------------- run --------------------
if __name__ == "__main__":
    input_dir = r"-\lcl"
    epsilon_values = [0.01, 0.1, 0.5, 1, 2, 5, 10, 25]
    window_size = 50  # correlation neighborhood size
    output_dir = fr"-\lcl_ldp_window_{window_size}"

    main(input_dir, output_dir, epsilon_values, window_size)
```

Appendix C

Gnomix Ancestry Deconvolution

```
import gzip
import numpy as np
import os
import pandas as pd
import pickle
import sys
import yaml
import math

from src.utils import run_shell_cmd, join_paths, read_vcf, vcf_to_npy, npy_to_vcf,
↪  update_vcf
from src.utils import read_genetic_map, save_dict, load_dict, read_headers
from src.preprocess import load_np_data, data_process
from src.postprocess import get_meta_data, write_msp, write_fb, msp_to_lai, msp_to_bed
from src.visualization import plot_cm, plot_chm
from src.laidataset import LAIDataset
from src.model import Gnomix

CLAIMER = (
    'When using this software, please cite: \n'
    'Helgi Hilmarsson, Arvind S Kumar, Richa Rastogi, Carlos D Bustamante, \n'
    'Daniel Mas Montserrat, Alexander G Ioannidis: \n'
    '"High Resolution Ancestry Deconvolution for Next Generation Genomic Data" \n'
    "https://www.biorxiv.org/content/10.1101/2021.09.19.460980v1"
)

# ------------------------------
# Helpers for PKL-based inference
# ------------------------------
```

```
def _coerce_geno_array(obj):
    """
    Accepts list/np.array of 0/1/2 or '0'/'1'/'2'/'N' or one long string and
    returns a numpy int8 array with values {0,1,2} and -1 for missing.
    """
    if isinstance(obj, str):
        arr = np.frombuffer(obj.encode('ascii'), dtype='S1').astype(str)
    else:
        arr = np.array(obj)

    # Handle string digits and 'N'
    if arr.dtype.kind in ('U', 'S', 'O'):
        arr = np.char.upper(arr.astype(str))
        miss = (arr == 'N')
        out = np.full(arr.shape, -1, dtype=np.int8)
        mask0 = (arr == '0'); out[mask0] = 0
        mask1 = (arr == '1'); out[mask1] = 1
        mask2 = (arr == '2'); out[mask2] = 2
        # everything else stays -1 (missing)
        return out
    else:
        # numeric
        out = arr.astype(np.int8, copy=False)
        # map any negatives or >2 to missing
        bad = (out < 0) | (out > 2)
        if bad.any():
            out[bad] = -1
        return out

def load_query_from_pkl(query_pkl_path, expected_len, verbose=False):
    """
    Loads a .pkl containing genotype data and returns a (1, C) np.ndarray with
    values in {0,1,2} and -1 for missing. Ensures length == expected_len.
    """
    if verbose:
        print(f"Loading PKL query from: {query_pkl_path}")
    with open(query_pkl_path, "rb") as f:
        obj = pickle.load(f)
    geno = _coerce_geno_array(obj)

    if geno.ndim > 1:
        geno = np.ravel(geno)

    if len(geno) != expected_len:
        raise ValueError(
```

```
            f"Length mismatch: PKL genotypes length={len(geno)} but model expects
            ↪  {expected_len} SNPs. "
            "Ensure your PKL vectors are aligned to the same SNP list used for
            ↪  training."
        )
    # model expects 2D: (n_samples, C)
    X = geno.reshape(1, -1)
    return X

def mask_missing_to_ref(X, missing_value=-1):
    """
    Replace missing (-1) with 0 (homozygous reference) for a conservative default.
    This keeps shapes consistent for the classifier.
    """
    if (X == missing_value).any():
        X = X.copy()
        X[X == missing_value] = 0
    return X

# --------------------------------

def load_model(path_to_model, verbose=True):
    if verbose:
        print("Loading model...")
    if path_to_model.endswith(".gz"):
        with gzip.open(path_to_model, 'rb') as unzipped:
            model = pickle.load(unzipped)
    else:
        model = pickle.load(open(path_to_model, "rb"))
    return model

def run_inference(base_args, model, visualize, snp_level=False, bed_file_output=False,
↪  verbose=False):

    if verbose:
        print("Loading and processing query file...")

    query_file = base_args["query_file"]
    chm = int(base_args["chm"])  # ensure int
    output_path = base_args["output_basename"]
    gen_map_df = model.gen_map_df

    # Ensure output path exists
    if not os.path.exists(output_path):
        os.makedirs(output_path, exist_ok=True)
```

```
# ----------------------------
# Branch: VCF vs PKL query
# ----------------------------
is_vcf = query_file.lower().endswith((".vcf", ".vcf.gz", ".bcf"))
is_pkl = query_file.lower().endswith(".pkl")

if not (is_vcf or is_pkl):
    raise ValueError("Unsupported query file format. Provide a .vcf/.vcf.gz or a
    ↪  .pkl genotype file.")

if is_vcf:
    # Load and process user query VCF file
    query_vcf_data = read_vcf(query_file, chm=chm, fields="*")
    X_query, vcf_idx, fmt_idx = vcf_to_npy(
        query_vcf_data, model.snp_pos, model.snp_ref, return_idx=True,
        ↪  verbose=verbose
    )
else:
    # PKL path: load vector aligned to model SNPs
    X_query = load_query_from_pkl(query_file, expected_len=len(model.snp_pos),
    ↪  verbose=verbose)
    X_query = mask_missing_to_ref(X_query, missing_value=-1)
    # Synthesize minimal VCF-like placeholders for downstream metadata functions
    query_vcf_data = {
        'variants/POS': model.snp_pos,     # assume aligned to model SNP order
        'samples': ['QUERY_SAMPLE']
    }
    vcf_idx = np.arange(X_query.shape[1])  # everything "present"
    fmt_idx = vcf_idx

# after vcf_to_npy(...) or after loading PKL
print("Model SNP count (len(model.snp_pos)):", len(model.snp_pos))
if is_vcf:
    print("vcf_to_npy returned fmt_idx length:", len(fmt_idx))
else:
    print("PKL vector length:", X_query.shape[1])

# predict and find effective prediction
if verbose:
    print("Inferring ancestry on query data...")

B_query = model.base.predict_proba(X_query)
if not base_args["phase"]:
    y_proba_query = model.smooth.predict_proba(B_query)
```

```
    y_pred_query = np.argmax(y_proba_query, axis=-1)
else:
    # For PKL queries, phasing here is artificial; we still allow it to run for
    ↪  consistency.
    if is_vcf:
        X_query_phased, y_pred_query = model.phase(X_query, B=B_query)
        if verbose:
            print("Writing phased SNPs to disk...")
        U = {
            "variants/REF": model.snp_ref[fmt_idx],
            "variants/ALT": model.snp_alt[fmt_idx].reshape(len(fmt_idx), 1)
        }
        query_vcf_data_phase = update_vcf(query_vcf_data, mask=vcf_idx, Updates=U)
        query_phased_prefix = output_path + "/" + "query_file_phased"
        inf_headers = read_headers(query_file)
        npy_to_vcf(query_vcf_data_phase, X_query_phased[:, fmt_idx],
        ↪  query_phased_prefix, headers=inf_headers)
        y_proba_query = model.predict_proba(X_query_phased)
    else:
        # PKL: run phasing-less path but keep interface
        y_proba_query = model.smooth.predict_proba(B_query)
        y_pred_query = np.argmax(y_proba_query, axis=-1)

# writing the result to disk
if verbose:
    print("Saving results...")
meta_data = get_meta_data(
    chm, model.snp_pos, query_vcf_data['variants/POS'], model.W, model.M,
    ↪  gen_map_df
)
out_prefix = output_path + "/" + "query_results"
write_msp(out_prefix, meta_data, y_pred_query, model.population_order,
↪  query_vcf_data['samples'])
write_fb(out_prefix, meta_data, y_proba_query, model.population_order,
↪  query_vcf_data['samples'])

# write the snp level results (BETA)
if snp_level:
    msp_to_lai(msp_file=out_prefix + ".msp",
    ↪  positions=query_vcf_data['variants/POS'], lai_file=out_prefix + ".lai")

if bed_file_output:
    bed_root = output_path + "/" + "query_results_bed"
    if not os.path.exists(bed_root):
        os.makedirs(bed_root)
```

```
    msp_to_bed(msp_file=out_prefix + ".msp", root=bed_root,
    ↪  pop_order=model.population_order)

# visualize results
if visualize:
    vis_path = join_paths(output_path, "visual", verb=False)
    msp_df = pd.read_csv(out_prefix + ".msp", sep="\t", skiprows=[0])
    for sample_id in query_vcf_data['samples']:
        sample_path = join_paths(vis_path, sample_id, verb=False)
        plot_chm(sample_id, msp_df, img_name=sample_path + "/chromosome_painting")

return
```

Appendix D

Reidentification Attack

```python
import numpy as np
import pickle
import os
import csv
from concurrent.futures import ProcessPoolExecutor

# ======================================================
# Load .pkl string → numeric array
# ======================================================
def load_snp_pkl(file_path):
    with open(file_path, "rb") as f:
        snp_data = pickle.load(f)

    # Case 1: data is a string of characters ("012N...")
    if isinstance(snp_data, str):
        return np.array([int(c) if c in "012" else -1 for c in snp_data],
        ↪  dtype=np.int8)

    # Case 2: data is already numeric (list or np.array of ints)
    elif isinstance(snp_data, (list, np.ndarray)):
        return np.array([int(c) if c in (0, 1, 2) else -1 for c in snp_data],
        ↪  dtype=np.int8)

    # Fallback
    else:
        raise ValueError(f"Unexpected data type in {file_path}: {type(snp_data)}")


# ======================================================
```

```python
# Distance function (Hamming distance ignoring -1)
# ====================================================
def snp_distance(vec1, vec2):
    mask = (vec1 >= 0) & (vec2 >= 0)   # ignore missing
    return np.sum(vec1[mask] != vec2[mask])


# ====================================================
# Re-identification attack for one epsilon
# ====================================================
def run_attack_for_epsilon(epsilon_dir, original_dir, output_csv):
    perturbed_files = [
        f for f in os.listdir(epsilon_dir) if f.endswith("_perturbed.pkl")
    ]
    if not perturbed_files:
        print(f"No perturbed files found in {epsilon_dir}")
        return

    n_total = 0
    n_correct = 0

    for fname in perturbed_files:
        perturbed_path = os.path.join(epsilon_dir, fname)
        perturbed_vec = load_snp_pkl(perturbed_path)

        # corresponding true file (before perturbation)
        true_name = fname.replace("_perturbed.pkl", ".pkl")

        # search across ALL originals for best match
        best_match = None
        best_dist = float("inf")

        for orig_fname in os.listdir(original_dir):
            if not orig_fname.endswith(".pkl"):
                continue
            orig_vec = load_snp_pkl(os.path.join(original_dir, orig_fname))
            dist = snp_distance(perturbed_vec, orig_vec)
            if dist < best_dist:
                best_dist = dist
                best_match = orig_fname

        n_total += 1
        print(best_match, true_name)
        if best_match == true_name:
            n_correct += 1
```

```
    success_rate = n_correct / n_total if n_total > 0 else 0.0
    print(f"Epsilon dir {epsilon_dir}: Re-ID success = {success_rate:.3f}")

    # Append to CSV
    with open(output_csv, "a", newline="") as f:
        writer = csv.writer(f)
        writer.writerow([os.path.basename(epsilon_dir), n_total, n_correct,
        ↪  success_rate])


# ====================================================
# Main function
# ====================================================
def main(original_dir, perturbed_root, output_csv, max_workers=None):
    # create/overwrite CSV
    with open(output_csv, "w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["Epsilon", "Total", "Correct", "SuccessRate"])

    epsilon_dirs = [
        os.path.join(perturbed_root, d)
        for d in os.listdir(perturbed_root)
        if d.startswith("eps") and os.path.isdir(os.path.join(perturbed_root, d))
    ]

    with ProcessPoolExecutor(max_workers=max_workers) as executor:
        futures = [
            executor.submit(run_attack_for_epsilon, eps_dir, original_dir, output_csv)
            for eps_dir in epsilon_dirs
        ]
        for future in futures:
            future.result()


# ====================================================
# Run
# ====================================================
if __name__ == "__main__":
    original_dir = r"-\gnomix\demo\snps"          # before perturbation
    window = 50
    perturbed_root = fr"-\gnomix\demo\snps_ldp_window_{window}"    # epsXX subdirs
    ↪  already created
    output_csv = fr"-\gnomix\demo\reid_results_{window}.csv"
```

```
main(original_dir, perturbed_root, output_csv, max_workers=None)
```