Title
Scalable Algorithms for Genetic Association Studies, Genotype Imputation, and Ancestry Inference

Author
Chu, Benjamin

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
Los Angeles

Scalable Algorithms for Genetic Association Studies, Genotype Imputation, and Ancestry Inference

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Biomathematics

by

Benjamin Beichen Chu

2021

ABSTRACT OF THE DISSERTATION

Scalable Algorithms for Genetic Association Studies, Genotype Imputation, and Ancestry Inference

by

Benjamin Beichen Chu
Doctor of Philosophy in Biomathematics
University of California, Los Angeles, 2021
Professor Kenneth Lange, Co-Chair
Professor Janet S. Sinsheimer, Co-Chair

This dissertation develops statistical and computational methods for human genetics. We consider problems in genome-wide association studies, imputation, phasing, and ancestry inference. The methods we develop are statistically robust, grounded in biological reality, and run extremely fast. Furthermore, we test these methods on the largest data available to us, such as the UK Biobank and Haplotype Reference Consortium. We implement our methods in individual, open-sourced Julia packages. They are freely available to the scientific community through the `OpenMendel` platform.

The dissertation of Benjamin Beichen Chu is approved.

Eric M. Sobel

Hua Zhou

Janet S. Sinsheimer, Committee Co-Chair

Kenneth Lange, Committee Co-Chair

University of California, Los Angeles

2021

To my mom and dad, who loved me so much

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Let me begin by expressing my immeasurable gratitude to my advisors Kenneth Lange and Janet Sinsheimer. Ken is the best mentor I could have hoped for, brilliant, generous, witty, elegant in writing, and has impeccable taste for mathematics. Janet has been equally supportive to me, charming, creative, persistent, forever enthusiastic, and leads by example. Their emotional support and encyclopedic knowledge are infinitely valuable. I hope one day I can pay it forward.

I also met wonderful mentors who profoundly guided my growth. I thank Hua Zhou for installing scientific computing in my education; Eric Sobel for always advising my work meticulously; Tom Chou and Joseph DiStefano III for their mentorship while I was a masters student at UCLA; Kevin Keys for being my 2018 GSoC mentor. Finally, I want to thank Carmay Lim and members in her lab Yu-Ming (Leon) Lee, Cédric Grauffel, Karine Mazmanian, and Jon Wright, for teaching me the basics of research every summer from high school to college, and most importantly, for igniting my passion in science that ultimately encouraged me to pursue this degree.

Most of the PhD related stress were unwillingly bore by my friends. I thoroughly enjoy the research banter with Calvin Chi and Shi-Zhuo Looi whom I met at Berkeley. My cohort Alfonso Landeros, Alexander Fisher, and Tianyun (Jason) Lin have been great companions and constant source of inspirations throughout graduate school. Also a big shout-out to the biomath folks for the long hours we spent together at the our 5th floor office followed by dorm dinner swipes. Finally, the OpenMendel students were extremely helpful and supportive, both emotionally and technically. All of your company made this journey very fun and pleasant.

In the end, let me thank my parents who enabled and nurtured me. You can always provide me solid advice no matter how old I become. I thank my sister for sharing similar hobbies as me and for being a good sister. I also thank my grandma, for always believing in me and spoiling me to the best of her ability. Lastly, I want to thank Jianxiao Yang. I can always count on your patience and wisdom to steer me in the right direction.

VITA

1993	Born. Taipei, Taiwan
2012–2016	B.A. Applied Mathematics, University of California, Berkeley
2017–2021	Graduate student researcher
2018–2020	Genomic Analysis Training Program (T32 HG002536), UCLA

PUBLICATIONS

Chu BB, Ko S, Zhou, JJ, Zhou H, Sinsheimer JS, Lange K. "Multivariate Genomewide Association Analysis with IHT" *bioRxiv*. 2021 Aug.

Chu BB, Sobel E, Wasiolek R, Sinsheimer JS, Zhou H, Lange K. "A Fast Data-Driven Method for Genotype Imputation, Phasing, and Local Ancestry Inference: MendelImpute.jl". *Bioinformatics*. 2021; btab489

Chu BB, Keys KL, German CA, Zhou H, Zhou JJ, Sobel EM, Sinsheimer JS, Lange K. "Iterative hard thresholding in genome-wide association studies: Generalized linear models, prior weights, and double sparsity" *GigaScience*. 2020 Jun;9(6):giaa044.

Zhou H, Sinsheimer JS, Bates DM, **Chu BB**, German CA, Ji SS, Keys K, Mosher G, Papp J, Sobel EM, Zhai J, Zhou J, Lange K (2020). "OpenMendel: a cooperative programming project for statistical genetics" *Human genetics* 139.1 (2020): 61-71.

Grauffel C, **Chu BB**, and Lim C. “An efficient protocol for computing the pK_a of Zn-bound water" *Physical Chemistry Chemical Physics* 20.47 (2018): 29637-29647.

CHAPTER 1

Introduction

In this chapter, we will remind readers of three concepts in statistical genetics, which materials from subsequent chapters are based upon. We will especially try to develop a mental model for how different genetics data are represented. Based on this understanding, we hope those who do not fully grasp all the biological or mathematical intricacies can appreciate what we are trying to do. We conclude with a section on high performance computing, a crucial theme penetrating every chapter of this dissertation.

1.1 Genome Wide Association Studies (GWAS)

GWAS is the most important concept in this dissertation, and possibly in all of human genetics as of 2021. It examines the relationship between genetic markers and a given phenotype. In a typical GWAS, a number of samples are randomly recruited, and their genetic information is collected via dense single nucleotide polymorphism (SNP) microarrays that survey a person's genome at particular locations. The resulting data can be assembled into a matrix $\mathbf{X}^{n\times p}$ where n is the number of samples and p is the number of SNPs. Each entry x_{ij} of $\mathbf{X}$ is the count of minor alleles for sample i at SNP j. Thus, $x_{ij} \in \{0,1,2\}$ since each person inherits two copies of each chromosome. In 2021, the number of SNPs p is usually on the order of 10^6, and n is between 10^4 to 10^6. Thus GWAS is fundamentally a very high dimensional problem.

Elucidating which SNP influences* the observed phenotypic values is the central problem GWASes

*Here we ignore the fact that statistically significant SNP markers are typically only in linkage disequilibrium with

Purpose	High coverage sequencing	Imputation
Estimate allele frequencies (MAF > 2e-4)	YES	YES
Estimate genotypes (MAF > 5%)	YES	YES
Estimate genotypes (MAF < 5%)	YES	NO
Detecting de novo mutations	YES	NO

Table 1.1: Pros and Cons of Sequencing vs imputation, adapted from [19]. Essentially, imputation is a good substitute for sequencing if a scientific study does not rely on detecting very rare variants.

attempt to answer. Besides the high diemensionality, we often need to make certain distributional assumptions on the phenotypes $\mathbf{y} \in \mathbb{R}^n$ (e.g. the number of seeds a plant produces must be a non-negative integer). This complexity, combined with the massive size of modern SNP data sets, limits statistical analysis and interpretation. Thus, how to analyze GWAS data remains a hot research area in genetics. In this dissertation, sections 2 and 4 are devoted to developing novel methods to analyze GWAS data.

1.2 Phasing and genotype imputation

Genotype imputation and phasing have become standard procedures in human genetics. Michigan's imputation server [31] alone imputes over 10 million genomes annually, which is a quarter of California's total population. What is genotype imputation, and why is it so popular?

As discussed earlier, GWASes survey $\sim 10^6$ SNPs from the human genome, which is roughly 0.1% of the 3 billion DNA basepairs in humans. Sequencing will recover 100% of the DNA information but presently it is ~ 20 times[†] more expensive than genotyping by a GWAS chip. A cost-effective alternative to sequencing is to genotype via a GWAS chip, and impute the remaining 99.9% using computational methods. The pros and cons of sequencing vs genotype via a GWAS chip and then using imputation are summarized in Table 1.1.

---

the true causal SNP

[†]As of 2021, sequencing a person costs around 1000 USD, while a GWAS chip is about 50 USD.

1.2.1 What is Genotype imputation?

Input/output. There are two inputs to genotype imputation. The first is a dense SNP data set $\mathbf{X}$, which has been introduced already. The second is a haplotype reference matrix $\mathbf{H}$, which usually contains 10–100 times more genotypes than $\mathbf{X}$. Given these two inputs, we will impute all SNPs that are in $\mathbf{H}$ to those in $\mathbf{X}$, and also determine the *phase* of every sample in $\mathbf{X}$. These interacting pieces are described below.

Phased vs unphased data. We previously mentioned that an entry of a GWAS data set satisfies $x_{ij} \in \{0,1,2\}$. This kind of data is *unphased* because if $x_{ij} = 1$, we cannot determine whether the minor allele (or alternate allele) resides on the paternal or maternal chromosome. In contrast, *phased* genotypes are such that we can distinguish the parental origin of heterozygous genotypes, that is, there is a difference between $0|1$ and $1|0$. Put another way, a sample's overall genotype $\mathbf{x}$ is phased when we can decompose it into the sum of two haplotypes $\mathbf{x} = \mathbf{h} + \mathbf{h}'$, where $x_i \in \{0,1,2\}$ and $h_j, h'_j \in \{0,1\}$. The latter interpretation is used heavily in section 3. For genotype imputation, the target GWAS data set $\mathbf{X}$ is usually unphased, but the reference haplotype panel is phased. The output $\mathbf{X}_{\text{imputed}}$ has the same number of SNPs as $\mathbf{H}$ and each sample is phased.

Reference haplotype panel. The second input to genotype imputation is a haplotype reference panel. This is a data set assembled by large consortia, and individual investigators will usually not analyze this panel. Rather, these panels provide 10^4 to 10^6 samples' phased genotypes (for which subjects are often selected from diverse ethnic backgrounds) and include 10^7 to 10^8 genomic markers that will be used as an input to imputation. As noted above, a phased genotype can be decomposed into two haplotypes. Thus, the haplotype panel can be essentially thought of as a huge matrix containing individual genotypes, where each genotype is stored in the form of $\mathbf{h}_1 + \mathbf{h}_2$ and the entries of $\mathbf{h}_1, \mathbf{h}_2$ are in $\{0,1\}$.

Since the haplotype reference panel is the only source for the missing genotype information, the quality of the panel exclusively determines imputation and phasing accuracy. As such, the sizes of these reference panels have been increasing by roughly 10 times every year since 2012 (see

introduction of Section 3 for specific references). The sizes of these reference panels in addition to the ever increasing size of GWAS data sets make genotype imputation an extremely computationally intensive task.

Why insist on outputting phased genotypes? Most genotype imputation algorithms output phased genotypes by default. For standard association testing, only the count of minor alleles (the so-called "dosage") is needed. Although haplotype blocks can be used as covariates for rare variant testing [111], that application is used less frequently compared to imputation. So why insist on outputting phased genotypes? One explanation is that most genotype imputation methods rely on hidden Markov models (HMM). HMMs require pre-phasing the genotypes prior to imputation, otherwise the algorithm scales as $O(n^2)$ instead of $O(n)$ [30]. HMMs also output phased genotypes as part of the Markov process. Thus, one recovers phased genotypes as a consequence of using HMM. As discussed in Section 3, alternatives to HMMs for imputation are possible. Phasing is also crucial for some types of genetic analysis, for example when imprinting is suspected.

1.3 Ancestry estimation

In GWAS, one of the most important confounders to control for is population structure. Estimation of population structure is, thus, a central problem in statistical genetics.

At the highest level, there is a distinction between local and global ancestry estimation. Local ancestry estimation divides an individual's genome into small segments and assigns a certain ancestral origin (e.g., an ethnic, country, or region label) to each of the segments. The goal is to find appropriate chromosomal segment boundaries and correctly infer each segment's origin. In global ancestry estimation, we estimate for each individual an overall fraction representing each ancestral population (e.g., individual I is $x\%$ European, $y\%$ Asian, etc).

1.3.1 Unsupervised approach for ancestry estimation

There are generally two unsupervised approaches[‡] to estimate population structure (i.e., global ancestry proportions), via principal component analysis (PCA) [91] and via model-based estimation of ancestry [6, 92]. PCA results are generally less interpretable because PCA is essentially a low-dimensional projection of the original genotype data. Model-based estimation of ancestry constructs a probabilistic model for gamete differentiation and estimates ancestry proportions as parameters of the model. Common model-based approaches include Bayesian MCMC [92] and maximum likelihood (which is the method used in the software package `ADMIXTURE`) [6]. Both regimes are *unsupervised* in the sense that ancestry information is inferred from genetic data $\mathbf{X}$ alone, without any ancestral labeling.

1.3.2 Supervised approach for ancestry estimation

There is considerably less literature on supervised approaches for ancestry estimation. By supervised we mean either certain alleles exhibit known different frequencies in different populations, or that some samples are labeled with their known ancestral origins. (Although it generally uses unsupervised approaches, a method which can exploit the latter type of information is also implemented in `ADMIXTURE` [4].) The lack of supervised approaches for ancestry determination is partly because unsupervised approaches are more flexible (since no labeling is required) and partly due to the scarcity of genome annotations.

Although less popular, supervised approaches may perform better in certain scenarios than unsupervised approaches. For instance, model-based methods such as `ADMIXTURE` perform poorly when a distinct sub-population has a small number of individuals (e.g., < 10 samples). A supervised approach would, in principal, not be affected by the sample size of distinct sub-populations.

[‡]Actually there is a 3rd way: by adding the genetic relationship matrix to a linear mixed model (LMM). This approach has been shown to adjust for small-scale family relatedness in a GWAS which PCAs cannot adjust for [117], but this is different than PCA and ADMIXTURE in the sense that we are not estimating ancestry to be included in the GWAS model as covariates. Hence, the discuss on LMM is omitted.

Haplotype reference panels can be exploited for supervised ancestry determination. These panels have grown exponentially in size over the past decade due to advances in genotype imputation. In section 3, we present a new approach for supervised (local and global) ancestry estimation that exploits haplotype reference panels. In particular, we show that a supervised approach delivers similar, and sometimes superior, results compared to the best unsupervised approaches such as `ADMIXTURE`.

1.4 Parallel and High Performance Computing (HPC)

Clock-speeds for computer CPUs hit their physical limit more than a decade ago [58]. Current high performance computing has thus focused on utilizing multiple cores simultaneously. For the largest problems, multiple machines each equipped with multi-core CPUs are often needed. Due to the sheer quantity of genetics data, mastery of various parallel computing regimes are necessary to use modern laptops, clusters, and cloud computing services to their full potential.

Here we review the basics of single-machine-multiple-core and multi-machine-multi-core parallel computing models, paying particular attention to their advantages and pitfalls. A few examples in the Julia language [12] are provided.

1.4.1 Shared memory (single-machine) parallelism

Shared-memory parallelism (also called multi-threading) assumes that all data for computation are accessible on a single computer. Thus, every CPU core can "see" the same data and any data modifications will be visible to every process. This type of parallel computing model is very useful for some problems encountered in genetics in which we need to repeatedly apply some routine f to a set of independent variables $x_1, \ldots, x_n$ loaded in memory. For example, in section 3, we impute small genotype blocks in parallel (treating each block as independent) and phase individuals in parallel (treating each sample as independent). Here the function f is imputation (or phasing), and each core applies f to genotype block (or sample) x_i independent of others.

The simplest way to incorporate shared memory parallelism is to exploit established libraries such as BLAS [13] and LAPACK [7]. The linear algebra routines in these libraries have been fine-tuned for performance over decades by experts and should be used whenever possible. Many Julia routines automatically call the appropriate BLAS or LAPACK functions under the hood. In other cases, one needs to define parallel routines on a case-by-case basis. In general, when writing multi-threaded code, one should beware of (1) race conditions, (2) false sharing, (3) over-subscriptions, (4) any non-uniform task used with a static scheduler, and (5) memory allocation (because garbage collection in Julia is single-threaded). A detailed discussion of these topics derails from the main theme of this dissertation, but mastery of all of these concepts are necessary to write performant multi-threaded code.

1.4.2 Distributed-memory (multi-machine) parallelism

Distributed memory parallelism can utilize the computing power of multiple machines simultaneously, such as multiple compute nodes on a HPC cluster. In contrast to the shared-memory model, in the distributed model a variable x that is visible to computer a may not be visible to computer b, unless contents of x are explicitly transferred from a to b. This highlights one of the major constraints of writing distributed parallel code: latency of data transfer. Since message passing among compute nodes is an expensive operation, sending a scalar, vector, or matrix makes a huge difference on the overall compute time. Algorithm design is therefore crucial. However, once data are sent, each compute node can fully utilize the multi-core CPU that it has. This can relieve part of the over-subscription (e.g., every node can run multi-threaded BLAS) and memory allocation burdens (e.g., every node can separately run garbage collection) often seen in the shared-memory model, as mentioned in the previous section.

The simplest way to access distributed computing in Julia is via the `Distributed.jl` package[§],

[§]It is also possible to use the more standard Message Passing Interface (MPI) [47] routines wrapped in the package `MPI.jl`, but one needs to be explicit about sending and receiving data, in contrast to using `Distributed.jl` where users only explicitly manages one side of the two-sided operation

which provides useful functions such as `@everywhere`, `pmap`, and `@distributed for`. In sections 2 and 4, we tried to incorporate distributed computing in q-fold cross-validation. In that case we needed to run penalized regression on the genotype matrix $\mathbf{X}$ with different sparsity parameters $k \in \{1, ..., n\}$. We usually needed to test many different values of k, so ideally we would have liked to utilize multiple computers for this routine. Thus, we settled on a design where $\mathbf{X}$ is memory-mapped, so only certain meta-information and allele summary statistics (e.g., mean and variance) need to be sent to distributed processes, while only a scalar representing the mean-squared error has to be sent back to the master process. Unfortunately, it is very difficult to request multiple compute nodes simultaneously within UCLA's computing cluster Hoffman2. Thus, while it is possible to distribute cross-validation folds to multiple computers in principal, we have never been able to exhaustively test and benchmark our implementation. Fortunately, as seen in section 4, we can still run the UK Biobank with $\sim 200,000$ samples and $\sim 500,000$ predictors on a single machine within 20 hours.

CHAPTER 2

Iterative hard thresholding in genome-wide association studies: Generalized linear models, prior weights, and double sparsity

2.1 Introduction

In genome-wide association studies (GWAS), modern genotyping technology coupled with imputation algorithms can produce an $n \times p$ genotype matrix $\mathbf{X}$ with $n \approx 10^6$ subjects and $p \approx 10^7$ genetic predictors [23, 108]. Data sets of this size require hundreds of gigabytes of disk space to store in compressed form. Decompressing data to floating point numbers for statistical analyses leads to matrices too large to fit into standard computer memory. The computational burden of dealing with massive GWAS datasets limits statistical analysis and interpretation. This paper discusses and extends a class of algorithms capable of meeting the challenge of multiple regression models with modern GWAS data scales.

Traditionally, GWAS analysis has focused on SNP-by-SNP (single nucleotide polymorphism) association testing [23, 22], with a p-value computed for each SNP via linear regression. This approach enjoys the advantages of simplicity, interpretability, and a low computational complexity of $\mathcal{O}(np)$. Furthermore, marginal linear regressions make efficient use of computer memory, since computations are carried out on genotype *vectors* one at a time, as opposed to running on the full genotype *matrix* in multiple regression. Some authors further increase association power by reframing GWAS as a linear mixed model problem and proceeding with variance component selection [49, 72]. These advances remain within the scope of marginal analysis.

Despite their numerous successes [108], marginal regression is less than ideal for GWAS. It

implicitly assumes that all SNPs have independent effects. In contrast, multiple regression can in principle model the effect of all SNPs simultaneously. This approach captures the biology behind GWAS more realistically because traits are usually determined by multiple SNPs acting in unison. Marginal regression selects associated SNPs one by one based on a pre-set threshold. Given the stringency of the p-value threshold, marginal regression can miss many causal SNPs with low effect sizes. As a result, heritability is underestimated. When $p \gg n$, one usually assumes that the number of variants k associated with a complex trait is much less than n. If this is true, we can expect multiple regression models to perform better because it a) offers better outlier detection [95] and better prediction, b) accounts for the correlations among SNPs, and c) allows investigators to model interactions. Of course, these advantages are predicated on finding the truly associated SNPs.

Adding penalties to the loss function is one way of achieving parsimony in multiple regression. The lasso [105, 107] is the most popular model selection device in current use. The lasso model selects non-zero parameters by minimizing the criterion

$$f(\boldsymbol{\beta}) \;=\; \ell(\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1,$$

where $\ell(\boldsymbol{\beta})$ is a convex loss, λ is a sparsity tuning constant, and $\|\boldsymbol{\beta}\|_1 = \sum_j |\beta_j|$ is the ℓ_1 norm of the parameters. The lasso has the virtues of preserving convexity and driving most parameter estimates to 0. Minimization can be conducted efficiently via cyclic coordinate descent [40, 113]. The magnitude of the nonzero tuning constant λ determines the number of predictors selected.

Despite its widespread use, the lasso penalty has some drawbacks. First, the ℓ_1 penalty tends to shrink parameters toward 0, sometimes severely so. Second, λ must be tuned to achieve a given model size. Third, λ is chosen by cross-validation, a costly procedure. Fourth and most importantly, the shrinkage caused by the penalty leaves a lot of unexplained trait variance, which tends to encourage too many false positives to enter the model ultimately identified by cross-validation.

Inflated false positive rates can be mitigated by substituting nonconvex penalties for the ℓ_1

penalty. For example, the minimax concave penalty (MCP) [123]

$$\lambda p(\beta_j) \;=\; \lambda \int_0^{|\beta_j|} \left(1 - \frac{s}{\lambda\gamma}\right)_+ ds$$

starts out at $\beta_j = 0$ with slope λ and gradually transitions to a slope of 0 at $\beta_j = \lambda\gamma$. With minor adjustments, the coordinate descent algorithm for the lasso carries over to MCP penalized regression [16, 80]. Model selection is achieved without severe shrinkage, and inference in GWAS improves [50]. However, in our experience its false negative rate is considerably higher than IHT's rate [56]. A second remedy for the lasso, stability selection, weeds out false positives by looking for consistent predictor selection across random halves of the data [84]. However, it is known to be under-powered for GWAS compared to standard univariate selection [5].

In contrast, iterative hard thresholding (IHT) minimizes a loss $\ell(\boldsymbol{\beta})$ subject to the nonconvex sparsity constraint $\|\boldsymbol{\beta}\|_0 \leq k$, where $\|\boldsymbol{\beta}\|_0$ counts the number of non-zero components of $\boldsymbol{\beta}$ [9, 10, 15]. Figure 2.1 explains graphically how the ℓ_0 penalty reduces the bias of the selected parameters. In the figure λ, γ, and k are chosen so that the same range of β values are sent to zero. To its detriment, the lasso penalty shrinks all β's, no matter how large their absolute values. The nonconvex MCP penalty avoids shrinkage for large β's but exerts shrinkage for intermediate β's. IHT, which is both nonconvex and discontinuous, avoids shrinkage altogether. For GWAS, the sparsity model-size constant k also has a simpler and more intuitive interpretation than the lasso tuning constant λ. Finally, both false positive and false negative rates are well controlled. Balanced against these advantages is the loss of convexity in optimization and concomitant loss of computational efficiency. In practice, the computational barriers are surmountable and are compensated by the excellent results delivered by IHT in high-dimensional regression problems such as multiple GWAS regression.

This article has four interrelated goals. First, we extend IHT to generalized linear models. These models encompass most of applied statistics. Previous IHT algorithms focused on normal or logistic sparse regression scenarios. Our software can also perform sparse regression under Poisson and negative binomial response distributions and can be easily extended to other GLM distributions as

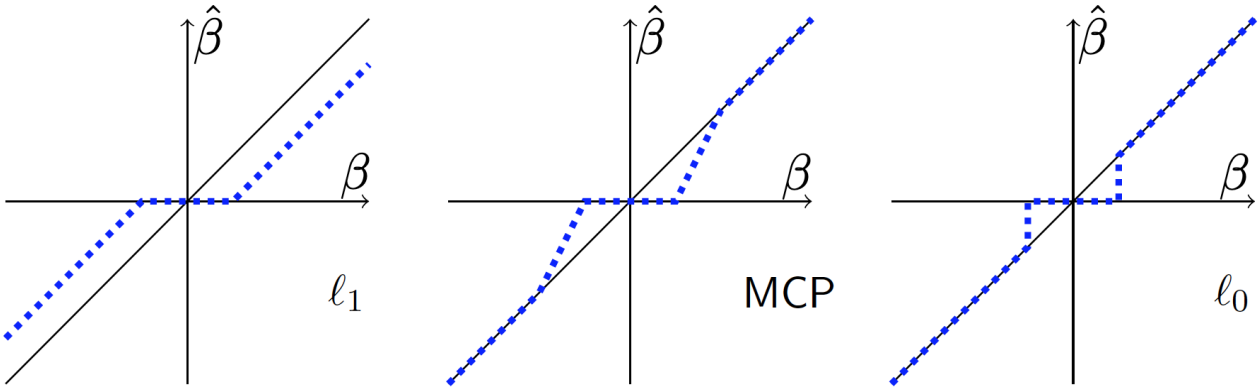


Figure 2.1: The ℓ_0 quasinorm of IHT enforces sparsity without shrinkage. The estimated effect size (dashed line) is plotted against its true value (diagonal line) for ℓ_1, MPC, and ℓ_0 penalties.

needed. The key to our extension is the derivation of a nearly optimal step size s for improving the loglikelihood at each iteration. Second, we introduce doubly-sparse regression to IHT. Previous authors have considered group sparsity [115]. The latter tactic limits the number of groups selected. It is also useful to limit the number of predictors selected per group. Double sparsity strikes a compromise that encourages selection of correlated causative variants in linkage disequilibrium (LD). Notably, this technique generalizes group-IHT. Third, we demonstrate how to incorporate predetermined SNP weights in IHT. Our simple and interpretable weighting option allows users to introduce prior knowledge into sparse projection. Thus, one can favor predictors whose association to the response is supported by external evidence. Fourth, we present `MendelIHT.jl`: a scalable, open source, and user friendly software for IHT in the high performance programming language Julia [12].

2.2 Model Development

This section sketches our extensions of iterative hard thresholding (IHT).

2.2.1 IHT Background

IHT was originally formulated for sparse signal reconstruction, which is framed as sparse linear least squares regression. In classical linear regression, we are given an $n \times p$ design matrix $\mathbf{X}$ and a corresponding n-component response vector $\mathbf{y}$. We then postulate that $\mathbf{y}$ has mean $E(\mathbf{y}) = \mathbf{X}\boldsymbol{\beta}$ and that the residual vector $\mathbf{y} - \mathbf{X}\boldsymbol{\beta}$ has independent Gaussian components with a common variance. The parameter (regression coefficient) vector $\boldsymbol{\beta}$ is estimated by minimizing the sum of squares $f(\boldsymbol{\beta}) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$. The solution to this problem is known as the ordinary least squares estimator and can be written explicitly as $\hat{\boldsymbol{\beta}} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}$, provided the problem is overdetermined (n > p). This paradigm breaks down in the high-dimensional regime $n \ll p$, where the parameter vector $\boldsymbol{\beta}$ is underdetermined. In the spirit of parsimony, IHT seeks a sparse version of $\boldsymbol{\beta}$ that gives a good fit to the data. This is accomplished by minimizing $f(\boldsymbol{\beta})$ subject to $\|\boldsymbol{\beta}\|_0 \leq k$ for a small value of k, where $\|\cdot\|_0$ counts the number of nonzero entries of a vector. The optimization problem is formally:

$$\min \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 \quad \text{subject to } ||\boldsymbol{\beta}||_0 \leq k. \tag{2.1}$$

IHT abandons the explicit formula for $\hat{\boldsymbol{\beta}}$ because it fails to respect sparsity and involves the numerically intractable matrix inverse $(\mathbf{X}^t\mathbf{X})^{-1}$.

IHT combines three core ideas. The first is steepest descent. Elementary calculus tells us that the negative gradient $-\nabla f(\mathbf{x})$ is the direction of steepest descent of $f(\boldsymbol{\beta})$ at $\mathbf{x}$. First-order optimization methods like IHT define the next iterate in minimization by the formula $\boldsymbol{\beta}_{n+1} = \boldsymbol{\beta}_n + s_n\mathbf{v}_n$, where $\mathbf{v}_n = -\nabla f(\boldsymbol{\beta}_n)$ and $s_n > 0$ is some optimally chosen step size. In the case of linear regression $-\nabla f(\boldsymbol{\beta}) = \mathbf{X}^t(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$. To reduce the error at each iteration, the optimal step size s_n can be selected

by minimizing the second-order Taylor expansion

$$\begin{aligned} & f(\boldsymbol{\beta}_n + s_n \mathbf{v}_n) \\ = \; & f(\boldsymbol{\beta}_n) + s_n \nabla f(\boldsymbol{\beta}_n)^t \mathbf{v}_n + \frac{s_n^2}{2} \mathbf{v}_n^t d^2 f(\boldsymbol{\beta}_n) \mathbf{v}_n \\ = \; & f(\boldsymbol{\beta}_n) - s_n \|\nabla f(\boldsymbol{\beta}_n)\|_2^2 + \frac{s_n^2}{2} \nabla f(\boldsymbol{\beta}_n)^t d^2 f(\boldsymbol{\beta}_n) \nabla f(\boldsymbol{\beta}_n) \end{aligned}$$

with respect to s_n. Here $d^2 f(\boldsymbol{\beta}) = \mathbf{X}^t \mathbf{X}$ is the Hessian matrix of second partial derivatives. Because $f(\boldsymbol{\beta})$ is quadratic, the expansion is exact. Its minimum occurs at the step size

$$s_n = \frac{\|\nabla f(\boldsymbol{\beta}_n)\|_2^2}{\nabla f(\boldsymbol{\beta}_n)^t d^2 f(\boldsymbol{\beta}_n) \nabla f(\boldsymbol{\beta}_n)}. \tag{2.2}$$

This formula summarizes the second core idea.

The third component of IHT involves projecting the steepest descent update $\boldsymbol{\beta}_n + s_n \mathbf{v}_n$ onto the sparsity set $S_k = \{\boldsymbol{\beta} : \|\boldsymbol{\beta}\|_0 \leq k\}$. The relevant projection operator $P_{S_k}(\boldsymbol{\beta})$ sets all but the k largest entries of $\boldsymbol{\beta}$ in magnitude to 0. In summary, IHT solves problem (2.1) by updating the parameter vector $\boldsymbol{\beta}$ according to the recipe:

$$\boldsymbol{\beta}_{n+1} = P_{S_k}(\boldsymbol{\beta}_n - s_n \nabla f(\boldsymbol{\beta}_n))$$

with the step size given by formula (2.2).

An optional debiasing step can be added to improve parameter estimates. This involves replacing $\boldsymbol{\beta}_{n+1}$ by the exact minimum point of $f(\boldsymbol{\beta})$ in the subspace defined by the support $\{j : \boldsymbol{\beta}_{n+1,j} \neq 0\}$ of $\boldsymbol{\beta}_{n+1}$. Debiasing is efficient because it solves a low-dimensional problem. Several versions of hard-thresholding algorithms have been proposed in the signal processing literature. The first of these, NIHT [15], omits debaising. The rest, HTP[38], GraHTP [119], and CoSaMp [86] offer debiasing.

Family	Mean Domain	$Var(y)$	$g(s)$
Normal	$\mathbb{R}$	ϕ^2	1
Poisson	$[0,\infty)$	μ	e^s
Bernoulli	$[0,1]$	$\mu(1-\mu)$	$\frac{e^s}{1+e^s}$
Gamma	$[0,\infty)$	$\mu^2\phi$	s^{-1}
Inverse Gaussian	$[0,\infty)$	$\mu^3\phi$	$s^{-1/2}$
Negative Binomial	$[0,\infty)$	$\mu(\mu\phi+1)$	e^s

Table 2.1: Summary of mean domains and variances for common exponential distributions. In GLM, $\boldsymbol{\mu} = g(\mathbf{x}^t\boldsymbol{\beta})$ denotes the mean, $s = \mathbf{x}^t\boldsymbol{\beta}$ the linear responses, g is the inverse link function, and ϕ the dispersion. Except for the negative binomial, all inverse links are canonical.

2.2.2 IHT for Generalized Linear Models

A generalized linear model (GLM) involves responses y following a natural exponential distribution with density in the canonical form

$$f(y \mid \theta, \phi) \quad = \quad \exp\left[\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right],$$

where y is the data, θ is the natural parameter, $\phi > 0$ is the scale (dispersion), and $a(\phi)$, $b(\theta)$, and $c(y,\phi)$ are known functions which vary depending on the distribution [35, 82]. Simple calculations show that y has mean $\mu = b'(\theta)$ and variance $\sigma^2 = b''(\theta)a(\phi)$; accordingly, σ^2 is a function of μ. Table 2.1 summarizes the mean domains and variances of a few common exponential families. Covariates enter GLM modeling through an inverse link representation $\mu = g(\mathbf{x}^t\boldsymbol{\beta})$, where $\mathbf{x}$ is a vector of covariates (predictors) and $\boldsymbol{\beta}$ is vector of regression coefficients (parameters). In statistical practice, data arrive as a sample of independent responses $y_1, \ldots, y_m$ with different covariate vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$. To put each predictor on an equal footing, each should be standardized to have mean 0 and variance 1. Including an additional intercept term is standard practice.

If we assemble a design matrix $\mathbf{X}$ by stacking the row vectors $\mathbf{x}_i^t$, then we can calculate the

loglikelihood, score, and expected information [35, 60, 82, 114]

$$\begin{aligned} L(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \left[\frac{y_i \theta_i - b_i(\theta_i)}{a_i(\phi_i)} + c(y_i, \phi_i) \right] \\ \nabla L(\boldsymbol{\beta}) &= \sum_{i=1}^{n} (y_i - \mu_i) \frac{g'(\mathbf{x}_i^t \boldsymbol{\beta})}{\sigma_i^2} \mathbf{x}_i \;=\; \mathbf{X}^t \mathbf{W}_1 (\mathbf{y} - \boldsymbol{\mu}) \\ J(\boldsymbol{\beta}) &= \sum_{i=1}^{n} \frac{1}{\sigma_i^2} g'(\mathbf{x}_i^t \boldsymbol{\beta})^2 \mathbf{x}_i \mathbf{x}_i^t \;=\; \mathbf{X}^t \mathbf{W}_2 \mathbf{X}, \end{aligned} \tag{2.3}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are two diagonal matrices. The second has positive diagonal entries; they coincide under the identity inverse link $g(s) = s$.

In the generalized linear model version of IHT, we maximize $L(\boldsymbol{\beta})$ (equivalent to minimizing $f(\boldsymbol{\beta}) = -L(\boldsymbol{\beta})$) and substitute the expected information $J(\boldsymbol{\beta}_n) = E[-d^2 L(\boldsymbol{\beta}_n)]$ for $d^2 f(\boldsymbol{\beta}_n)$ in formula (2.2). This translates into the following step size in GLM estimation:

$$s_n = \frac{\|\nabla L(\boldsymbol{\beta}_n)\|_2^2}{\nabla L(\boldsymbol{\beta}_n)^t J(\boldsymbol{\beta}_n) \nabla L(\boldsymbol{\beta}_n)}. \tag{2.4}$$

This substitution is a key ingredient of our extended IHT. It simplifies computations and guarantees that the step size is nonnegative.

2.2.3 Doubly Sparse Projections

The effectiveness of group sparsity in penalized regression has been demonstrated in general [83, 39] and for GWAS [127] in particular. Group IHT [115] enforces group sparsity but does not enforce within-group sparsity. In GWAS, model selection is desired within groups as well to pinpoint causal SNPs. Furthermore, one concern in GWAS is that two causative SNPs can be highly correlated with each other due to linkage disequilibrium (LD). When sensible group information is available, doubly sparse IHT encourages the detection of causative yet correlated SNPs while enforcing sparsity within groups. Here we discuss how to carry out a doubly-sparse projection that enforces both within- and between-group sparsity.

Suppose we divide the SNPs of a study into a collection G of nonoverlapping groups. Given a parameter vector $\boldsymbol{\beta}$ and a group $g \in G$, let $\boldsymbol{\beta}_g$ denote the components of $\boldsymbol{\beta}$ corresponding to the SNPs in g. Now suppose we want to select at most j groups and at most $\lambda_g \in \mathbb{Z}^+$ SNPs for each group g. In projecting $\boldsymbol{\beta}$, the component β_i is untouched for a selected SNP i. For an unselected SNP, β_i is reset to 0. By analogy with our earlier discussion, we can define a sparsity projection operator $P_g(\boldsymbol{\beta}_g)$ for each group g; $P_g(\boldsymbol{\beta}_g)$ selects the λ_g most prominent SNPs in group g. The potential reduction in the squared distance offered by group g is $r_g = \|\boldsymbol{\beta}_g\|_2^2 - \|P_g(\boldsymbol{\beta}_g)\|_2^2$. The j selected groups are determined by selecting the j largest values of r_g. If desired, we can set the sparsity level λ_g for each group high enough so that all SNPs in group g come into play. Thus, doubly-sparse IHT generalizes group-IHT. In Algorithm 1, we write $P(\boldsymbol{\beta})$ for the overall projection with the component projections $P_g(\boldsymbol{\beta}_g)$ on the j selected groups and projection to zero on the remaining groups.

2.2.4 Prior weights in IHT

Zhou et al. [127] treat prior weights in penalized GWAS. Before calculating the lasso penalty, they multiply each component of the parameter vector $\boldsymbol{\beta}$ by a positive weight w_i. We can do the same in IHT before projection. Thus, instead of projecting the steepest descent step $\boldsymbol{\beta} = \boldsymbol{\beta}_n + s_n \mathbf{v}_n$, we project the Hadamard (pointwise) product $\mathbf{w} \circ \boldsymbol{\beta}$ of $\boldsymbol{\beta}$ with a weight vector $\mathbf{w}$. This produces a vector with a sensible support S. The next iterate $\boldsymbol{\beta}_{n+1}$ is defined to have support S and to be equal to $\boldsymbol{\beta}_n + s_n \mathbf{v}_n$ on S.

In GWAS, weights can and should be informed by prior biological knowledge. A simple scheme for choosing nonconstant weights relies on minor allele frequencies. For instance, Zhou et al. [124] assign SNP i with minor allele frequency p_i the weight $w_i = 1/\sqrt{2p_i(1-p_i)}$. Giving rare SNPs greater weight in this fashion is most appropriate for traits under strong negative selection [121, 98]. Alternatively, our software permits users to assign weights geared to specific pathway and gene information.

de Lamare et al. [33] incorporate prior weights into IHT by adding an element-wise logarithm

of a weight vector $\mathbf{q}$ before projection. The weight vector $\mathbf{q}$ is updated iteratively and requires two additional tuning constants that in practice are only obtained through cross validation. Our weighting scheme is simpler, more computationally efficient, and more interpretable.

2.2.5 Algorithm Summary

The final algorithm combining doubly sparse projections, prior weight scaling, and debiasing is summarized in Algorithm 1.

Algorithm 1: Iterative hard-thresholding

Input : Design matrix $\mathbf{X}$, response vector $\mathbf{y}$, membership vector $\mathbf{g}$, weight vector $\mathbf{w}$, max number of groups j, and overall sparsity projection $P(\boldsymbol{\beta})$.

1 **Initialize:** $\boldsymbol{\beta} \equiv \mathbf{0}$.
2 **while not converged do**
3 **Calculate:** score $= \mathbf{v}$, Fisher information matrix $= \mathbf{J}$, and step size $= s = \frac{\mathbf{v}^t \mathbf{v}}{\mathbf{v}^t \mathbf{J} \mathbf{v}}$
4 **Ascent direction with scaling:** $\tilde{\boldsymbol{\beta}} = \mathbf{w} \circ (\boldsymbol{\beta}_n + s\mathbf{v})$
5 **Project to sparsity:** $\tilde{\boldsymbol{\beta}} = P\left(\tilde{\boldsymbol{\beta}}\right) ./ \mathbf{w}$ (where $./$ is elementwise division)
6 **while** $L(\tilde{\boldsymbol{\beta}}) \leq L(\boldsymbol{\beta}_n)$**, backtrack** ≤ 5 **do**
7 $s = s/2$
8 Redo lines 4 to 5
9 **end**
10 **(Optional) Debias:** Let $F = supp(\tilde{\boldsymbol{\beta}})$, compute $\hat{\boldsymbol{\beta}} = argmax_{\{\boldsymbol{\beta}:\boldsymbol{\beta} \text{ restricted to } F\}} L(\boldsymbol{\beta})$
11 **Accept proposal:** $\boldsymbol{\beta}_{n+1} = \hat{\boldsymbol{\beta}}$
12 **end**

Output: $\boldsymbol{\beta}$ with j active groups and λ_g active predictors for group g

2.3 Results

Readers can reproduce our results by accessing the software, documentation, and Jupyter notebooks at:

`https://github.com/OpenMendel/MendelIHT.jl`

2.3.1 Scalability of IHT

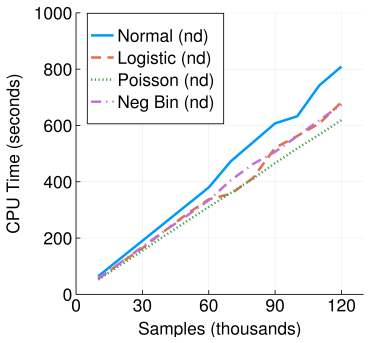
To test the scalability of our implementation, we ran IHT on $p = 10^6$ SNPs for sample sizes $n = 10,000, 20,000, ..., 120,000$ with five independent replicates per n. All simulations rely on a true sparsity level of $k = 10$. Based on an Intel-E5-2670 machine with 63GB of RAM and a single 3.3GHz processor, Figure 2.2 plots the IHT median CPU time per iteration, median iterations to convergence, and median memory usage under Gaussian, logistic, Poisson, and negative binomial models. The largest matrix simulated here is 30GB in size and can still fit into our personal computer's memory. Of course, it is possible to test even larger sample sizes using cloud or cluster resources, which are often needed in practice.

The formation of the vector $\boldsymbol{\mu}$ of predicted values requires only a limited number of nonzero regression coefficients. Consequently, the computational complexity of this phase of IHT is relatively light. In contrast, calculation of the Fisher score (gradient) and information (expected negative Hessian) depend on the entire genotype matrix $\mathbf{X}$. Fortunately, each of the np entries of $\mathbf{X}$ can be compressed to 2 bits. Figure 2.2b and d show that IHT memory demands beyond storing $\mathbf{X}$ never exceeded a few gigabytes. Figure 2.2a and c show that IHT run time per iteration increases linearly in problem size n. Similarly, we expect increasing p will increase run time linearly, since the bottleneck of IHT is the matrix-vector multiplication step in computing the gradient, which scales as $O(np)$. Debiasing increases run time per iteration only slightly. Except for negative binomial responses, debiasing is effective in reducing the number of iterations required for convergence and hence overall run time.

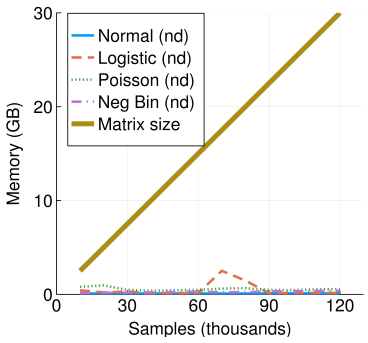
2.3.2 Cross Validation in Model Selection

In actual studies, the true number of genetic predictors k_{true} is unknown. This section investigates how q-fold cross-validation can determine the best model size on simulated data. Under normal, logistic, Poisson, and negative binomial models, we considered 50 different combinations of $\mathbf{X}$, $\mathbf{y}$, and $\boldsymbol{\beta}_{true}$ with $k_{true} = 10$, $n = 5000$ samples, and $p = 50,000$ SNPs fixed in all replicates. Here, k_{true}

(a) Speed per iteration without debiasing



(b) Memory usage without debiasing

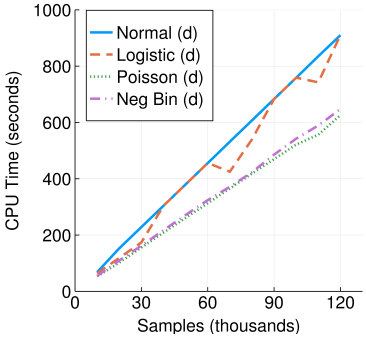


(c) Median iterations until convergence.

	Without debiasing	With Debiasing
Normal	4.0 (1.0)	2.0 (0.0)
Logistic	10.0 (5.5)	2.5 (2.25)
Poisson	47.5 (9.75)	33.0 (5.75)
Neg Bin	9.0 (0.25)	15.0 (1.5)

( ) = interquartile range

(d) Speed per iteration with debiasing



(e) Memory usage with debiasing

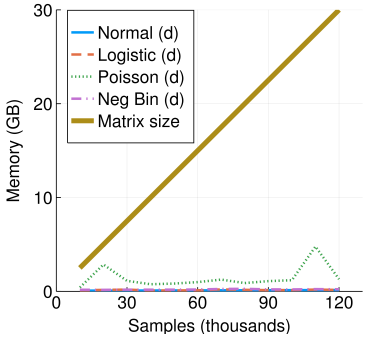


Figure 2.2: (a, d) Time per iteration scales linearly with data size. Speed is measured for compressed genotype files. On uncompressed data, all responses are roughly 10 times faster. (b, e) Memory usage scales as $\sim 2np$ bits. Note memory for each response are usages in addition to loading the genotype matrix. Uncompressed data requires 32 times more memory. (c) Debiasing reduces median iterations until convergence for all but negative binomial regression. Benchmarks were carried out on 10^6 SNPs and sample sizes ranging from 10,000 to 120,000. Hence, the largest matrix here requires 30GB and can still fit into personal computer memories.

is chosen so that it is closer to our NFBC and UK Biobank results. On these data sets we conducted 5-fold cross validation across 20 model sizes k ranging from 1 to 20. Figure 2.3 plots deviance residuals on the holdout dataset for each of the four GLM responses (mean squared error in the case of normal responses) and the best estimate $\hat{k}$ of k_{true}.

Figure 2.3 shows that k_{true} can be effectively recovered by cross validation. In general, prediction error starts off high where the proposed sparsity level k severely underestimates k_{true} and plateaus when k_{true} is reached (Figure 2.3a-d). Furthermore, the estimated sparsity $\hat{k}$ for each run is narrowly centered around $k_{true} = 10$ (Figure 2.3e-f). In fact, $|\hat{k} - k_{true}| \leq 4$ always holds. When $\hat{k}$ exceeds

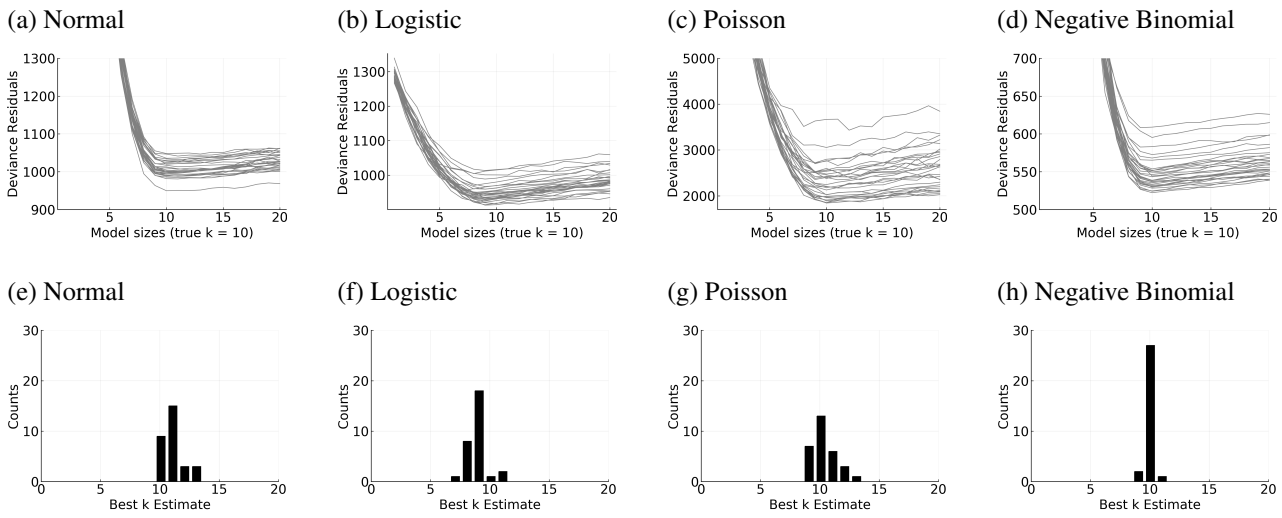


Figure 2.3: Five-fold cross validation results is capable of identifying the true model size k_{true}. (a-d) Deviance residuals of the testing set are minimized when the estimated model size $\hat{k} \approx k_{true}$. Each line represents 1 simulation. (e-h) $\hat{k}$ is narrowly spread around $k_{true} = 10$.

k_{true}, the estimated regression coefficients for the false predictors tend to be very small. In other words, IHT is robust to overfitting, in contrast to lasso penalized regression. We see qualitatively similar results when k_{true} is large. This proved to be the case in our previous paper [56] for Gaussian models with $k_{true} \in \{100, 200, 300\}$.

2.3.3 Comparing IHT to Lasso and Marginal Tests in Model Selection

Comparison of the true positive and false positive rates of IHT and its main competitors is revealing. For lasso regression we use the `glmnet` implementation of cyclic coordinate descent [40, 112, 113] (v2.0-16 implemented in `R 3.5.2`); for marginal testing we use the beta version of `MendelGWAS` [128]. As explained later, Poisson regression is supplemented by zero-inflated Poisson regression implemented under the `pscl` [120] (v1.5.2) package of `R`. Unfortunately, `glmnet` does not accommodate negative binomial regression. Because both `glmnet` and `pscl` operate on floating point numbers, we limit our comparisons to small problems with 1000 subjects, 10,000 SNPs, 50 replicates, and $k = 10$ causal SNPs. IHT performs model selection by 3-fold cross validation across model sizes ranging from 1 to 50. This range is generous enough to cover the models selected by

	Normal	Logistic	Poisson	Neg Bin
IHT TP	8.84	6.28	7.2	9.0
IHT FP	0.02	0.1	1.28	0.98
Lasso TP	9.52	8.16	9.28	NA
Lasso FP	31.26	45.76	102.24	NA
Marginal TP	7.18	5.76	9.04 (5.94)	5.98
Marginal FP	0.06	0.02	1527.9 (0.0)	0.0

Table 2.2: IHT achieves the best balance of false positives and true positives compared to lasso and marginal (single-snp) regression. TP = true positives, FP = false positives. There are $k = 10$ causal SNPs. Best model size for IHT and lasso were chosen by cross validation. $()$ = zero-inflated Poisson regression.

lasso regression. We adjust for multiple testing in the marginal case test by applying a p-value cutoff of 5×10^{-6}.

Table 2.2 demonstrates that IHT achieves the best balance between maximizing true positives and minimizing false positives. IHT finds more true positives than marginal testing and almost as many as lasso regression. IHT also finds far fewer false positives than lasso regression. Poisson regression is exceptional in yielding an excessive number of false positives in marginal testing. A similar but less extreme trend is observed for lasso regression. The marginal false positive rate is reduced by switching to zero-inflated Poisson regression. This alternative model is capable of handling overdispersion due an excess of 0 values. Interestingly, IHT rescues the Poisson model by accurately capturing the simultaneous impact of multiple predictors.

2.3.4 Reconstruction Quality for GWAS Data

Table 2.3 demonstrates that IHT estimates show little bias compared to estimates from lasso and marginal regressions. These trends hold with or without debiasing as described earlier. The proportion of variance explained is approximately the same in both scenarios. The displayed values are the averaged estimated β's, computed among the SNPs actually found. As expected, lasso estimates show severe shrinkage compared to IHT. Estimates from marginal tests are severely overestimated, since each SNP are asked to explain more trait variance than it could. As the magnitude of β_{true}

falls, IHT estimates show an upward absolute bias, consistent with the winner's curse phenomenon. When sample sizes are small, small effect sizes make most predictors imperceptible amid the random noise. The winner's curse operates in this regime and cannot be eliminated by IHT. Lasso's strong shrinkage overwhelms the bias of the winner's curse and yields estimates smaller than true values.

The results displayed in Table 2.3 reflect $n = 5,000$ subjects, $p = 10,000$ SNPs, 100 replicates, and a sparsity level k fixed at its true value $k_{true} = 10$. The λ value for lasso is chosen by cross validation. To avoid data sets with monomorphic SNPs, the minimum minor allele frequency (maf) is set at 0.05. For linear, logistic and Poisson regressions in marginal tests, we first screen for potential SNPs via a score test. Only top SNPs are used in the more rigorous and more computationally intensive likelihood ratio tests, which gives the beta estimates. This procedure is described in [128]. We ran likelihood ratio tests for all SNPs in the negative binomial model because the screening procedure is not yet implemented. However, the inflation in parameter estiamtes are present throughout all marginal tests.

2.3.5 Correlated Covariates and Doubly Sparse Projections

Next we study how well IHT works on correlated data and whether doubly-sparse projection can enhance model selection. Table 2.4 shows that, in the presence of extensive LD, IHT performs reasonably well even without grouping information. When grouping information is available, group IHT enhances model selection. The results displayed in Table 2.4 reflect $n = 1,000$ samples, $p = 10,000$ SNPs, and 100 replicates. Each SNP belongs to 1 of 500 disjoint groups containing 20 SNPs each; $j = 5$ distinct groups are each assigned $1, 2, ..., 5$ causal SNPs with effect sizes randomly chosen from $\{-0.2, 0.2\}$. In all there 15 causal SNPs. For grouped-IHT, we assume perfect group information. That is, groups containing $1 \sim 5$ causative SNPs are assigned $\lambda_g \in \{1, 2, ..., 5\}$. The remaining groups are assigned $\lambda_g = 1$. As described in the Methods Section, the simulated data show LD within each group, with the degree of LD between two SNPs decreasing as their separation increases. Although the conditions of this simulation are somewhat idealized, they mimic what

β_{true}	β_{IHT}^{Normal}	$\beta_{IHT}^{Logistic}$	$\beta_{IHT}^{Poisson}$	β_{IHT}^{NegBin}
.5	$.501 \pm .015$	$.508 \pm .039$	$.492 \pm .039$	$.567 \pm .670$
.25	$.249 \pm .013$	$.256 \pm .038$	$.247 \pm .012$	$.249 \pm .012$
.10	$.097 \pm .014$	$.125 \pm .016$	$.100 \pm .014$	$.010 \pm .012$
.05	$.063 \pm .007$	$.108 \pm .006$	$.057 \pm .008$	$.060 \pm .008$
β_{true}	β_{lasso}^{Normal}	$\beta_{lasso}^{Logistic}$	$\beta_{lasso}^{Poisson}$	β_{lasso}^{NegBin}
.5	$.451 \pm .015$	$.366 \pm .058$	$.458 \pm .037$	NA
.25	$.199 \pm .013$	$.137 \pm .032$	$.208 \pm .015$	NA
.10	$.046 \pm .014$	$.022 \pm .016$	$.058 \pm .016$	NA
.05	$.012 \pm .008$	$.008 \pm .003$	$.012 \pm .009$	NA
β_{true}	$\beta_{marginal}^{Normal}$	$\beta_{marginal}^{Logistic}$	$\beta_{marginal}^{Poisson}$	$\beta_{marginal}^{NegBin}$
.5	$.990 \pm .500$	$.983 \pm .475$	$.942 \pm .331$	$.930 \pm .315$
.25	$.493 \pm .189$	$.480 \pm .216$	$.452 \pm .184$	$.486 \pm .178$
.10	$.203 \pm .078$	*	$.198 \pm .097$	$.190 \pm .090$
.05	*	*	$.165 \pm .049$	$.097 \pm .060$

Table 2.3: Comparison of coefficient estimates among IHT, lasso, and marginal regression methods. Displayed coefficients are average fitted valued $\pm$ one standard error for the discovered predictors. * = zero true positives observed on average. NA = `glmnet` does not support negative binomial lasso regression. There are $k = 10$ true SNPs.

	Ungrouped-IHT		Grouped-IHT	
	TP	FP	TP	FP
Normal	11.1 ± 1.9	3.9 ± 1.9	12.2 ± 2.0	2.8 ± 2.0
Logistic	3.8 ± 1.6	11.2 ± 1.6	7.7 ± 2.2	7.3 ± 2.2
Poisson	11.5 ± 2.2	3.5 ± 2.2	12.4 ± 1.7	2.6 ± 1.7
Neg Bin	11.0 ± 2.1	4.0 ± 2.1	12.4 ± 1.6	2.6 ± 1.6

Table 2.4: Doubly-sparse IHT enhances model selection on simulated data. TP = true positives, FP = false positives, $\pm$ 1 standard error. There are 15 causal SNPs in 5 groups, each containing $k \in= \{1,2,...5\}$ SNPs.

might be observed if small genetic regions of whole exome data were used with IHT.

We repeated this examination of doubly sparse projection for the first $30,000$ SNPs of the NFBC1966 [97] data for all samples passing the quality control measures outlined in our Methods Section. We arbitrarily assembled 2 large groups with 2000 SNPs, 5 medium groups with 500 SNPs, and 10 small groups with 100 SNPs, representing genes of different length. The remaining SNPs are lumped into a final group representing non-coding regions. In all there are 18 groups.

	Ungrouped-IHT		Grouped-IHT	
	TP	FP	TP	FP
Normal	17.0 ± 1.2	2.0 ± 1.2	17.0 ± 1.4	2.1 ± 1.4
Logistic	15.7 ± 1.5	3.3 ± 1.5	15.8 ± 1.6	3.2 ± 1.6
Poisson	17.1 ± 1.3	1.9 ± 1.3	17.0 ± 1.4	2.0 ± 1.4
Neg Bin	17.2 ± 1.5	1.8 ± 1.5	17.0 ± 1.5	2.1 ± 1.5

Table 2.5: Doubly sparse IHT is comparable to regular IHT on NFBC dataset using arbitrary groups. TP = true positives, FP = false positives, $\pm$ 1 standard error. There are 19 causal SNPs in 18 groups of various size. Simulation was carried out on the first 30,000 SNPs of the NFBC1966 [97] dataset.

Since group assignments are essentially random beyond choosing neighboring SNPs, this example represents the worse case scenario of a relatively sparse marker map with undifferentiated SNP groups. We randomly selected 1 large group, 2 medium groups, and 3 small groups to contain 5, 3, and 2 causal SNPs, respectively. The non-coding region harbors 2 causal SNPs. In all there are 19 causal SNPs. Effect sizes were randomly chosen to be -0.2 or 0.2. We ran 100 independent simulation studies under this setup, where the large, medium, small, and non-coding groups are each allowed 5, 3, 2, and 2 active SNPs. The results are displayed in Table 2.5. We find that even in this worse case scenario where group information is completely lacking that grouped IHT does no worse than ungrouped IHT.

2.3.6 Introduction of Prior Weights

This section considers how scaling by prior weights helps in model selection. Table 2.6 compares weighted IHT reconstructions with unweighted reconstructions where all weights $w_i = 1$. The weighted version of IHT consistently finds approximately 10% more true predictors than the unweighted version. Here we simulated 50 replicates involving 1000 subjects, 10,000 uncorrelated variants, and $k = 10$ true predictors for each GLM. For the sake of simplicity, we defined a prior weight $w_i = 2$ for about one-tenth of all variants, including the 10 true predictors. For the remaining SNPs the prior weight is $w_i = 1$. These choices reflect a scenario where one tenth of all genotyped variants fall in a protein coding region, including the 10 true predictors, and where such variants are twice as likely to influence a trait as those falling in non-coding regions.

	Unweighted-IHT		Weighted-IHT	
	TP	FP	TP	FP
Normal	9.2 ± 0.4	0.8 ± 0.4	9.4 ± 0.5	0.6 ± 0.5
Logistic	7.3 ± 0.6	2.7 ± 0.6	8.0 ± 0.6	2.0 ± 0.6
Poisson	8.0 ± 0.6	2.0 ± 0.6	8.3 ± 0.6	1.7 ± 0.6
Neg Bin	9.2 ± 0.5	0.8 ± 0.5	9.4 ± 0.5	0.6 ± 0.5

Table 2.6: Weighted IHT enhances model selection. TP = true positives, FP = false positives, $\pm$ 1 standard error. The true number of SNPs is $k = 10$.

2.3.7 Hypertension GWAS in the UK Biobank

Now we test IHT on the second release of UK Biobank [102] data. This dataset contains $\sim 500,000$ samples and $\sim 800,000$ SNPs without imputation. Phenotypes are systolic blood pressure (SBP) and diastolic blood pressure (DBP), averaged over 4 or fewer readings. To adjust for ancestry and relatedness, we included the following nongenetic covariates: sex, hospital center, age, age^2, BMI, and the top 10 principal components computed with `FlashPCA2` [3]. After various quality control procedures as outlined in the Methods section, the final dataset used in our analysis contains $185,565$ samples and $470,228$ SNPs. For UK biobank analysis, we omitted debiasing, prior weighting, and doubly sparse projections.

2.3.7.1 Stage 2 Hypertension under a Logistic Model

Consistent with the clinical definition for stage 2 hypertension (S2 Hyp) [110], we designated patients as hypertensive if their SBP $\geq$ 140mmHG or DBP $\geq$ 90 mmHG. We ran 5-fold cross validated logistic model across model sizes $k = \{1, 2, ..., 50\}$. The work load was distributed to 50 computers, each with 5 CPU cores. Each computer was assigned one model size, and all completed its task within 24 hours. The model size that minimizes the deviance residuals is $\hat{k} = 39$. The selected predictors include the 33 SNPs listed in Table 2.7 and 6 non-genetic covariates: intercept, sex, age, age^2, BMI, and the fifth principal component.

Figure 2.4, generated by `MendelPlots.jl` [45], compares univariate logistic GWAS with logistic IHT. SNPs recovered by IHT are circled in black. Our Github page records the full list of

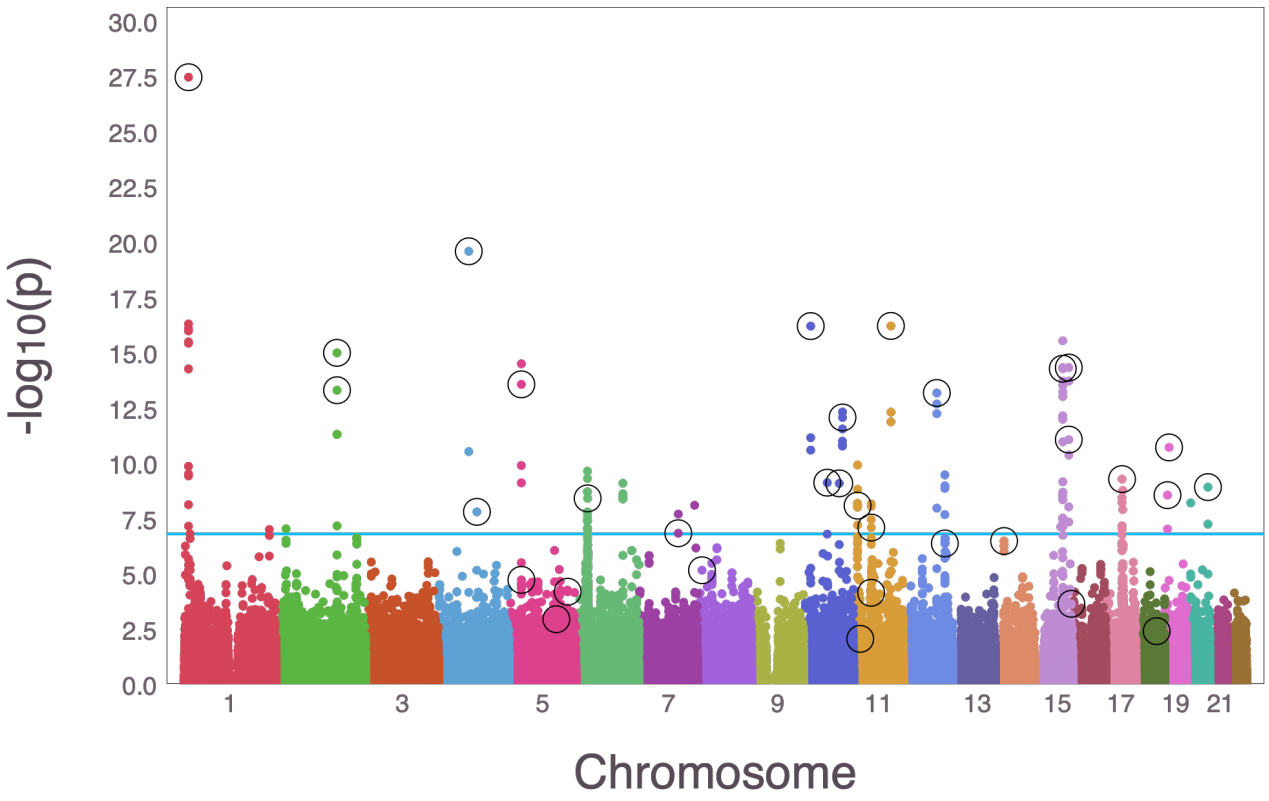


Figure 2.4: Manhattan plot comparing a logistic (univariate) GWAS vs logistic IHT on UK Biobank data. Colored dots are $\log_{10}$ p-values from a logistic GWAS, and the circled dots are SNPs recovered by IHT.

significant SNPs detected by univariate GWAS. There are 10 SNPs selected by IHT that have a p-value less than 5×10^{-8}; 83 SNPs pass the threshold in the univariate analysis but remain unselected by IHT. IHT tends to pick the most significant SNP among a group of SNPs in LD. Table 2.7 shows 25 SNPs selected by IHT that were previously reported to be associated with elevated SBP/DBP [73] or that exhibit genome-wide significance when the same data are analyzed as an ordinal trait [44]. Ordinal univariate GWAS treats the different stages of hypertension as ordered categories. Ordinal GWAS has higher power than logistic or multinomial GWAS [44]. The known SNPs displayed in Table 2.7 tend to have larger absolute effect sizes (avg 0.033) than the unknown SNPs (avg = 0.027). Finally, IHT is able to recover two pairs of highly correlated SNPs: (rs1374264,rs1898841) and (rs7497304,rs2677738) with pairwise correlations of $r_{1,2} = 0.59$ and $r_{3,4} = 0.49$.

SNP	Chrom	Position	$\hat{\beta}$	Known?
rs17367504	1	11862778	0.046	[73, 44]
rs757110	2	17418477	-0.025	
rs1898841	2	165070207	0.022	[44]
rs1374264	2	164999883	0.020	[44]
rs16998073	4	81184341	-0.048	[73, 44]
rs1173771	4	32815028	0.046	[73, 44]
rs13107325	4	103188709	0.030	[73, 44]
rs72742749	5	32834974	0.029	
rs11241955	5	127626884	0.028	
rs2072495	5	158296996	-0.027	
rs805293	6	31688518	-0.029	[44]
rs2392929	7	106414069	-0.039	[73, 44]
rs73203495	8	11580334	-0.031	
rs12258967	10	18727959	0.039	[73, 44]
rs11191580	10	104906211	0.039	[73, 44]
rs2274224	10	96039597	0.036	[44]
rs1530440	10	63524591	0.028	[73, 44]
rs10895001	11	100533021	0.043	[44]
rs2293579	11	47440758	-0.035	[44]
rs2923089	11	10357572	-0.029	[44]
rs762551	11	75041917	-0.027	[44]
rs4548577	11	46998512	0.026	
rs2681492	12	90013089	0.030	[73, 44]
rs10849937	12	111792427	0.030	[44]
rs35085068	14	23409909	-0.027	[44]
rs12901664	15	98338524	-0.027	
rs7497304	15	91429176	-0.021	[73, 44]
rs2677738	15	91441673	0.021	[44]
rs3744760	17	43195981	-0.043	[44]
rs292445	18	55897720	-0.026	
rs167479	19	11526765	0.036	[73, 44]
rs34328549	19	7253184	0.035	[44]
rs16982520	20	57758720	-0.030	[73, 44]

Table 2.7: UK Biobank GWAS results generated by running IHT on Stage 2 Hypertension (S2 Hyp) under a logistic model. The SNP ID, chromosome number, position (in basepair), and estimated effect sizes are listed.

SNP	Chrom	Position	$\hat{\beta}$	Known?
rs6917603	6	30125050	0.17	[56, 73]
rs9261256	6	30129920	-0.07	[56]
rs9261224	6	30121866	-0.03	
rs7120118	11	47242866	-0.03	[56, 97, 73]
rs1532085	15	56470658	-0.04	[56, 97, 73]
rs3764261	16	55550825	-0.05	[56, 97, 73]
rs3852700	16	65829359	-0.03	
rs1800961	20	42475778	0.03	[73]

Table 2.8: NFBC GWAS results generated by running IHT on high density lipoprotein (HDL) phenotype as a normal response. The SNP ID, chromosome number, position (in basepair), and estimated effect sizes are listed.

2.3.8 Cardiovascular GWAS in NFBC1966

We also tested IHT on data from the 1966 Northern Finland Birth Cohort (NFBC1966) [97]. Although this dataset is relatively modest with 5402 participants and 364,590 SNPs, it has two virtues. First, it has been analyzed multiple times [56, 97, 42], so comparison with earlier analysis is easy. Second, due to a population bottleneck [79], the participants' chromosomes exhibit more extensive linkage disequilibrium than is typically found in less isolated populations. Multiple regression methods, including the lasso, have been criticized for their inability to deal with the dependence among predictors induced by LD. Therefore this dataset provides an interesting test case.

2.3.8.1 High Density Lipoprotein (HDL) Phenotype as a Normal model

Using IHT we find previously associated SNPs as well as a few new potential associations. We model the HDL phenotype as normally-distributed and find a best model size $\hat{k} = 9$ based on 5-fold cross validation across model sizes $k = \{1, 2, ..., 20\}$. Without debiasing, the analysis was completed in 2 hours and 4 minutes with 30 CPU cores on a single machine. Table 2.8 displays the recovered predictors. SNP rs1800961 was replaced by rs7499892 with similar effect size if we add the debiasing step in obtaining the final model.

Importantly, IHT is able to simultaneously recover effects for SNPs (1) rs9261224, (2) rs6917603,

and (3) rs6917603 with pairwise correlations of $r_{1,2} = 0.618$, $r_{1,3} = 0.984$, and $r_{2,3} = 0.62$. This result is achieved without grouping of SNPs, which can further increase association power. Compared with earlier analyses of these data, we find 3 SNPs that were not listed in our previous IHT paper [56], presumably due to slight algorithmic modifications. The authors of NFBC [97] found 5 SNPs associated with HDL under SNP-by-SNP testing. We did not find SNPs rs2167079 and rs255049. To date, rs255049 was replicated [42]. SNP rs2167079 has been reported to be associated with an unrelated phenotype [85]. If we repeat the analysis with HDL dichotomized into low and high HDL using a cutpoint of 60ml/DL, then we identify the 5 SNPs rs9261224, rs6917603, rs9261256, rs3764261, and rs9898058; all but one of these, SNP rs9898058, is also found under the continuous model. This SNP is not replicated in previous studies. As in the continuous model, rs6917603 has the largest effect of all the selected SNPs. Readers interested in the full result can visit our Github site.

2.3.8.2 Low Density Lipoprotein (LDL) as a Binary Response

Unfortunately we did not have access to any qualitative phenotypes for this cohort, so for purposes of illustration, we fit a logistic regression model to a derived dichotomous phenotype, high versus low levels of Low Density Lipoprotein (LDL). The original data are continuous, so we choose 145 mg/dL, the midpoint [110] between the borderline-high and high LDL cholesterol categories, to separate the two categories. This dichotomization resulted in 932 cases (high LDL) and 3961 controls (low LDL). Under 5-fold cross validation without debiasing across model sizes $k = \{1, 2, ..., 20\}$, we find $\hat{k} = 3$. Using 30 CPU cores, our algorithm finishes in 1 hours and 7 minutes.

Despite the loss of information inherent in dichotomization, our results are comparable to the prior results under a normal model for the original quantitative LDL phenotype. Our final model still recovers two SNP predictors with and without debiasing (Table 2.8). We miss all but one of the SNPs that the NFBC analysis found to be associated with LDL treated as a quantitative trait. Notably we again find an association with SNP rs6917603 that they did not report.

2.4 Discussion

Multiple regression methods like iterative hard thresholding provide a principled way of model fitting and variable selection. With increasing computing power and better software, multiple regression methods are likely to prevail over univariate methods. This paper introduces a scalable implementation of iterative hard thresholding for generalized linear models. Because lasso regression can handle group and prior weights, we have also extended IHT to incorporate such prior knowledge. When it is available, enhanced IHT outperforms standard IHT. Given its sharper parameter estimates and more robust model selection, IHT is clearly superior to lasso selection or marginal association testing in GWAS.

Our real data analyses and simulation studies suggest that IHT can (a) recover highly correlated SNPs, (b) avoid over-fitting, (c) deliver better true positive and false positive rates than either marginal testing or lasso regression, (d) recover unbiased regression coefficients, and (e) exploit prior information and group-sparsity. Our Julia implementation of IHT can also exploit parallel computing strategies that scale to biobank-level data. In our opinion, the time is ripe for the genomics community to embrace multiple regression models as a supplement to and possibly a replacement of marginal analysis.

Although we focused our attention on GWAS, the potential applications of iterative hard thresholding reach far beyond gene mapping. Our IHT implementation accepts arbitrary numeric data and is suitable for a variety of applied statistics problems. Genetics and the broader field of bioinformatics are blessed with rich, ultra-high dimensional data. IHT is designed to solve such problems. By extending IHT to the realm of generalized linear models, it becomes possible to fit regression models with more exotic distributions than the Gaussian distributions implicit in ordinary linear regression. In our view IHT will eventually join and probably supplant lasso regression as the method of choice in GWAS and other high-dimensional regression settings.

2.5 Supplementary materials

2.5.1 Data Simulation

Our simulations mimic scenarios for a range of rare and common SNPs with or without LD. Unless otherwise stated, we designate 10 SNPs to be causal with effect sizes of $0.1, 0.2, ..., 1.0$.

To generate independent SNP genotypes, we first sample a minor allele frequency $\rho_j \sim Uniform(0, 0.5)$ for each SNP j. To construct the genotype of person i at SNP j, we then sample from a binomial distribution with success probability ρ_j and two trials. The vector of genotypes (minor allele counts) for person i form row $\mathbf{x}_i^t$ of the design matrix $\mathbf{X}$. To generate SNP genotypes with linkage disequilibrium, we divide all SNPs into blocks of length 20. Within each block, we first sample $x_1 \sim Bernoulli(0.5)$. Then we form a single haplotype block of length 20 by the following Markov chain procedure:

$$x_{i+1} = \begin{cases} x_i & \text{with probability } \ p \\ 1 - x_i & \text{with probability } \ 1 - p \end{cases}$$

with default $p = 0.75$. For each block we form a pool of 20 haplotypes using this procedure, ensuring every one of the 40 alleles (2 at each SNP) are represented at least once. For each person, the genotype vector in a block is formed by sampling 2 haplotypes with replacement from the pool and summing the number of minor alleles at each SNP.

Depending on the simulation, the number of subjects range from $1,000$ to $120,000$, and the number of independent SNPs range from $10,000$ to $1,000,000$. We simulate data under four GLM distributions: normal (Gaussian), Bernoulli, Poisson, and negative binomial. We generate component y_i of the response vector $\mathbf{y}$ by sampling from the corresponding distribution with mean $\mu_i = g(\mathbf{x}_i^t \boldsymbol{\beta})$, where g is the inverse link function. For normal models we assume unit variance, and for negative binomial models we assume 10 required failures. To avoid overflows, we clamp the mean $g(\mathbf{x}_i^t \boldsymbol{\beta})$ to stay within $[-20, 20]$. (See Ad Hoc Tactics for a detailed explanation). We apply the canonical link for each distribution, except for the negative binomial, where we apply the log link.

2.5.2 Real Data's Quality Control Procedures

UK Biobank. Following the UK biobank's own quality control procedures, we first filtered all samples for sex discordance and high heterozygosity/missingness. Second, we included only people of European ancestry and excluded first and second-degree relatives based on empiric kinship coefficients. Third, we also excluded people who had taken hypertension related medications at baseline. Finally, we only included people with $\geq 98\%$ genotyping success rate over all chromosomes and SNPs with $\geq 99\%$ genotyping success rate. Calculation of kinship coefficients and filtering were carried out via the OpenMendel modules `SnpArrays` [126]. Remaining missing genotypes were imputed using modal genotypes at each SNP. After these quality control procedures, our UK biobank data is the same data that was used in [44].

Northern Finland Birth Cohort. We imputed missing genotypes with Mendel [62]. Following [56], we excluded subjects with missing phenotypes, fasting subjects, and subjects on diabetes medication. We conducted quality control measures using the OpenMendel module `SnpArrays` [126]. Based on these measures, we excluded SNPs with minor allele frequency ≤ 0.01 and Hardy Weinberg equilibrium p-values $\leq 10^{-5}$. As for non-genetic predictors, we included sex (the `sexOCPG` factor defined in [97]) as well as the first 2 principal components of the genotype matrix computed via `PLINK 2.0 alpha` [24]. To put predictors, genetic and non-genetic, on an equal footing, we standardized all predictors to have mean zero and unit variance.

2.5.3 Linear Algebra with Compressed Genotype Files

The genotype count matrix stores minor allele counts. The PLINK genotype compression protocol [24] compactly stores the corresponding 0's, 1's, and 2's in 2 bits per SNP, achieving a compression ratio of 32:1 compared to storage as floating point numbers. For a sparsity level k model, we use OpenBLAS (a highly optimized linear algebra library) to compute predicted values. This requires transforming the k pertinent columns of $\mathbf{X}$ into a floating point matrix $\mathbf{X}_k$ and multiplying it times the corresponding entries $\boldsymbol{\beta}_k$ of $\boldsymbol{\beta}$. The inverse link is then applied to $\mathbf{X}_k\boldsymbol{\beta}_k$ to give the mean vector

$\boldsymbol{\mu} = g(\mathbf{X}_k\boldsymbol{\beta}_k)$. In computing the GLM gradient (equation 2.3), formation of the vector $\mathbf{W}_1(\mathbf{y}-\boldsymbol{\mu})$ involves no matrix multiplications. Computation of the gradient $\mathbf{X}^t\mathbf{W}_1(\mathbf{y}-\boldsymbol{\mu})$ is more complicated because the full matrix $\mathbf{X}$ can no longer be avoided. Fortunately, the OpenMendel module `SnpArrays` [126] can be invoked to perform compressed matrix times vector multiplication. Calculation of the steplength of IHT requires computation of the quadratic form $\nabla L(\boldsymbol{\beta}_n)^t\mathbf{X}^t\mathbf{W}_2\mathbf{X}\nabla L(\boldsymbol{\beta}_n)$. Given the gradient, this computation requires a single compressed matrix times vector multiplication. Finally, good statistical practice calls for standardizing covariates. To standardize the genotype counts for SNP j, we estimate its minor allele frequency p_j and then substitute the ratio $\frac{x_{ij}-2p_j}{\sqrt{2p_j(1-p_j)}}$ for the genotype count x_{ij} for person i at SNP j. This procedure is predicated on a binomial distribution for the count x_{ij}. Our previous paper [56] shows how to accommodate standardization in the matrix operations of IHT without actually forming or storing the standardized matrix.

Although multiplication via the OpenMendel module `SnpArrays` [126] is slower than OpenBLAS multiplication on small data sets, it can be as much as 10 times faster on large data sets. OpenBLAS has advantages in parallelization, but it requires floating point arrays. Once the genotype matrix $\mathbf{X}$ exceeds the memory available in RAM, expensive data swapping between RAM and hard disk memory sets in. This dramatically slows matrix multiplication. `SnpArrays` is less vulnerable to this hazard owing to compression. Once compressed data exceeds RAM, `SnpArrays` also succumbs to the swapping problem. Current laptop and desktop computers seldom have more than 32 GB of RAM, so we must resort to cluster or cloud computing when input files exceed 32 GB.

2.5.4 Computations Involving Non-genetic Covariates

Non-genetic covariates are stored as double or single precision floating point entries in an $n \times r$ design matrix $\mathbf{Z}$. To accommodate an intercept, the first column should be a vector of 1's. Let $\boldsymbol{\gamma}$ denote the r vector of regression coefficients corresponding to $\mathbf{Z}$. The full design matrix is the block

matrix $(\mathbf{XZ})$. Matrix multiplications involving $(\mathbf{XZ})$ should be carried out via

$$(\mathbf{XZ}) \begin{pmatrix} \beta \\ \gamma \end{pmatrix} = \mathbf{X}\beta + \mathbf{Z}\gamma \quad \text{and} \quad (\mathbf{XZ})^t \mathbf{v} = \begin{pmatrix} \mathbf{X}^t\mathbf{v} \\ \mathbf{Z}^t\mathbf{v} \end{pmatrix}.$$

Adherence to these rules ensures a low memory footprint. Multiplication involving $\mathbf{X}$ can be conducted as previously explained. Multiplication involving $\mathbf{Z}$ can revert to BLAS.

2.5.5 Parallel Computation

The OpenBLAS library accessed by Julia is inherently parallel. Beyond that we incorporate parallel processing in cross validation. Recall that in q-fold cross validation we separate subjects into q disjoint subsets. We then fit a training model using $q-1$ of those subsets on all desired sparsity levels and record the mean-squared prediction error on the omitted subset. Each of the q subsets serve as the testing set exactly once. Testing error is averaged across the different folds for each sparsity levels k. The lowest average testing error determines the recommended sparsity.

`MendelIHT.jl` offers 2 parallelism strategies in cross validation. Either the q training sets are each loaded to q different CPUs where each compute and test differ sparsity levels sequentially, or each of the q training sets are cycled through sequentially and each sparsity parameter is fitted and tested in parallel. The former tactic requires enough disk space and RAM to store q different training data (where each typically require $(q-1)/q$ GB of the full data), but offers immense parallel power because one can assign different computers to handle different sparsity levels. This tactic allows one to fit biobank scale data in less than a day assuming enough storage space and computers are available. The latter tactic requires cycling through the training sets sequentially. Since intermediate data can be deleted, the tactic only requires enough disk space and RAM to store 1 copy of the training set. `MendelIHT.jl` uses one of Julia's [12] standard library `Distributed.jl` to achieve the aforementioned parallel strategies.

2.5.6 Ad Hoc Tactics to Prevent Overflows

In Poisson and negative binomial regressions, the inverse link argument $\exp(\mathbf{x}_i^t\boldsymbol{\beta})$ experiences numerical overflows when the inner product $\mathbf{x}_i^t\boldsymbol{\beta}$ is too large. In general, we avoid running Poisson regression when response means are large. In this regime a normal approximation is preferred. As a safety feature, `MendelIHT.jl` clamps values of $\mathbf{x}_i^t\boldsymbol{\beta}$ to the interval $[-20, 20]$. Note that penalized regression suffers from the same overflow catastrophes.

2.5.7 Convergence and Backtracking

For each proposed IHT step we check whether the objective $L(\boldsymbol{\beta})$ increases. When it does not, we step-halve at most 5 times to restore the ascent property. Convergence is declared when

$$\frac{||\boldsymbol{\beta}_{n+1} - \boldsymbol{\beta}_n||_\infty}{||\boldsymbol{\beta}_n||_\infty + 1} < \text{Tolerance},$$

with the default tolerance being 0.0001. The addition of 1 in the denominator of the convergence criterion guards against division by 0.

2.6 Availability of source code

Project name: MendelIHT

Project home page:

https://github.com/OpenMendel/MendelIHT.jl

Operating systems: Mac OS, Linux, Windows

Programming language: Julia 1.0, 1.2

License: MIT

The code to generate simulated data, as well as their subsequent analysis, are available in our github repository under *figures* folder. Project.toml and Manifest.toml files can be used together to in-

stantiate the same computing environment in our paper. Notably, `MendelIHT.jl` interfaces with the OpenMendel [128] package `SnpArrays.jl` [126] and JuliaStats's packages `Distribution.jl` [11] and `GLM.jl` [67].

CHAPTER 3

A Fast Data-Driven Method for Genotype Imputation, Phasing, and Local Ancestry Inference: MendelImpute.jl

3.1 Introduction

Haplotyping (phasing) is the process of inferring unobserved haplotypes from observed genotypes. It is possible to deduce phase from the observed genotypes of surrounding pedigree members [100], but pedigree data are no longer considered competitive with linkage disequilibrium data. Current methods for phasing and genotype imputation exploit public reference panels such as those curated by the Haplotype Reference Consortium (HRC) [81] and the NHLBI TOPMed Program [103]. The sizes of these reference panels keep expanding: from 1000 samples in 2012 [2], to about 30,000 in 2016 [81], and to over 90,000 in 2019 [103]. Genome-wide association studies (GWAS), the primary consumers of imputation, exhibit similar trends in increasing sample sizes and denser SNP typing [102]. Despite these technological improvements, phasing and imputation methods are still largely based on hidden Markov models (HMM). Through decades of successive improvements, HMM software is now more than 10,000 times faster than the original software [30], but the core HMM principles remain relatively unchanged. This paper explores an attractive data-driven alternative for imputation and phasing that is faster and simpler than HMM methods.

Hidden Markov models (HMMs) capture the linkage disequilibrium in haplotype reference panels based on the probabilistic model of Li and Stephens [66]. The latest HMM software programs include `Minimac 4` [31], `Beagle 5` [20], and `Impute 5` [96]. These HMMs programs all have essentially the same imputation accuracy [20], are computationally intensive, and generally require

pre-phased genotypes. The biggest computational bottleneck facing these programs is the size of the HMM state space. An initial pre-phasing (imputation) step fills in missing phases and genotypes at the typed markers in a study. The easier second step constructs haplotypes on the entire set of SNPs in the reference panel from the pre-phased data [51]. This separation of tasks forces users to chain together different computer programs, reduces imputation accuracy [30], and tends to inflate overall run-times even when the individual components are well optimized.

Purely data-driven techniques are potential competitors to HMMs in genotype imputation and haplotyping. Big data techniques substitute massive amounts of training data for detailed models in prediction. This substitution can reduce computation times and, if the data are incompatible with the assumptions underlying the HMM, improve accuracy. Haplotyping HMMs, despite their appeal and empirically satisfying error rates, make simplifying assumptions about recombination hot spots and linkage patterns. We have previously demonstrated the virtues of big data methods in genotype imputation with haplotyping [25] and without haplotyping [26]. `SparRec` [53] refines the later method by adding additional information on matrix co-clustering. These two matrix completion methods efficiently impute missing entries via low-rank approximations. Unfortunately, they also rely on computationally intensive cross validation to find the optimal rank of the approximating matrices. On the upside, matrix completion circumvents pre-phasing, exploits reference panels, and readily imputes dosage data, where genotype entries span the entire interval $[0,2]$.

Despite these advantages, data-driven methods have not been widely accepted as alternatives to HMM methods. Although it is possible in principle, our previous program [26] did not build a pipeline to handle large reference panels. Here we propose a novel data-driven method to fill this gap. Our software `MendelImpute` (a) avoids the pre-phasing step, (b) exploits known haplotype reference panels, (c) supports dosage data, (d) runs extremely fast, (e) makes a relatively small demand on memory, and (f) naturally extends to local and global ancestry inference. Its imputation error rate is slightly higher than the best HMM software but still within a desirable range. `MendelImpute` is open source and forms a part of the `OpenMendel` platform [128], which is in the modern Julia programming language [12]. We demonstrate that `MendelImpute` is capable of dealing with HRC

data even on a standard laptop. In coordination with our packages `VCFTools.jl` for VCF files, `SnpArrays.jl` for PLINK files, and `BGEN.jl` for BGEN files, `OpenMendel` powers a streamlined pipeline for end-to-end data analysis.

For each chromosome of a study subject, `MendelImpute` reconstructs two extended haplotypes $\mathbf{E}_1$ and $\mathbf{E}_2$ that cover the entire chromosome. Both $\mathbf{E}_1$ and $\mathbf{E}_2$ are mosaics of reference haplotypes with a few break points where a switch occurs from one reference haplotype to another. The break points presumably represent contemporary or ancient recombination events. `MendelImpute` finds these reference segments and their break points. From $\mathbf{E}_1$ and $\mathbf{E}_2$ it is trivial to impute missing genotypes, both typed and untyped. The extended haplotypes can be painted with colors indicating the region on the globe from which each reference segment was drawn. The number of SNPs assigned to each color immediately determine ethnic proportions and plays into admixture mapping. The extended segments also serve as a convenient device for data compression. One simply stores the break points and the index of the reference haplotype assigned to each segment. Finally, $\mathbf{E}_1$ and $\mathbf{E}_2$ can be nominated as maternal or paternal whenever either parent of a sample subject is also genotyped.

3.2 Materials and Methods

Our overall imputation strategy operates on an input matrix $\mathbf{X}$ whose columns are sample genotypes at the typed markers. The entries of $\mathbf{X}$ represent alternative allele counts $x_{ij} \in [0,2] \cup \{\text{missing}\}$. The reference haplotypes are stored in a matrix $\mathbf{H}$ whose columns are haplotype vectors with entries $h_{ij} \in \{0,1\}$, representing reference and alternative alleles, respectively. Given these data, the idea is to partition each sample's genotype vector into small adjacent genomic windows. In each window, many reference haplotypes collapse to the same unique haplotype at the typed SNPs. We find the two unique haplotypes whose vector sum best matches the sample genotype vector. Then we expand the unique haplotypes into two sets of matching full haplotypes and intersect these sets across adjacent windows. Linkage disequilibrium favors long stretches of single reference haplotypes punctuated by break points. Our strategy is summarized in Figure 3.1. A detailed commentary on the interacting

tactics appears in subsequent sections.

3.2.1 Missing Data in Typed and Untyped SNPs

There are two kinds of missing data requiring imputation. A GWAS data set may sample on the order of 10^6 SNPs across the genome. We call SNPs that are sampled at this stage typed SNPs. Raw data from a GWAS study may contain entries missing at random due to experimental errors, but the missing rate is usually low, at most a few percent, and existing programs [70] usually impute these in the pre-phasing step. When modern geneticists speak of imputation, they refer to imputing phased genotypes at the unsampled SNPs present in the reference panel. We call the unsampled markers untyped SNPs. The latest reference panels contain from 10^7 to 10^8 SNPs, so an imputation problem can have more than 90% missing data. We assume that the typed SNPs sufficiently cover the entire genome. From the mosaic of typed and untyped SNPs, one can exploit local linkage disequilibrium to infer for each person his/her phased genotypes at all SNPs, typed and untyped. As a first step one must situate the typed SNPs among the ordered SNPs in the reference panel (Figure 3.1A). The Julia command `indexin()` quickly finds the proper alignment.

3.2.2 Finding Optimal Haplotype Pairs via Least Squares

Suppose there are d unique haplotypes $\mathbf{h}_1, \ldots, \mathbf{h}_d$ (entries 0 or 1) in a genomic window (Figure 3.1B), where the supplement discusses how to efficiently compute them. Consider a genotype vector $\mathbf{x}$ with entries $x_i \in [0,2] \cup \{\text{missing}\}$. The goal is to find the two unique haplotypes $\mathbf{h}_i$ and $\mathbf{h}_j$ such that $\mathbf{x} \approx \mathbf{h}_i + \mathbf{h}_j$. The best haplotype pair is selected by minimizing the least squares criterion

$$||\mathbf{x} - \mathbf{h}_i - \mathbf{h}_j||_2^2 = ||\mathbf{x}||_2^2 + ||\mathbf{h}_i||_2^2 + ||\mathbf{h}_j||_2^2 + 2\mathbf{h}_i^T \mathbf{h}_j - 2\mathbf{x}^T \mathbf{h}_i - 2\mathbf{x}^T \mathbf{h}_j \tag{3.1}$$

over all $\binom{d}{2} + d$ haplotype combinations. To fill in a missing value x_i, we naively initialize it with the mean at each typed SNP. This action may lead to imputation errors when the typed SNPs exhibit a large proportion of missing values. We discuss a strategy to remedy this bias in the supplement.

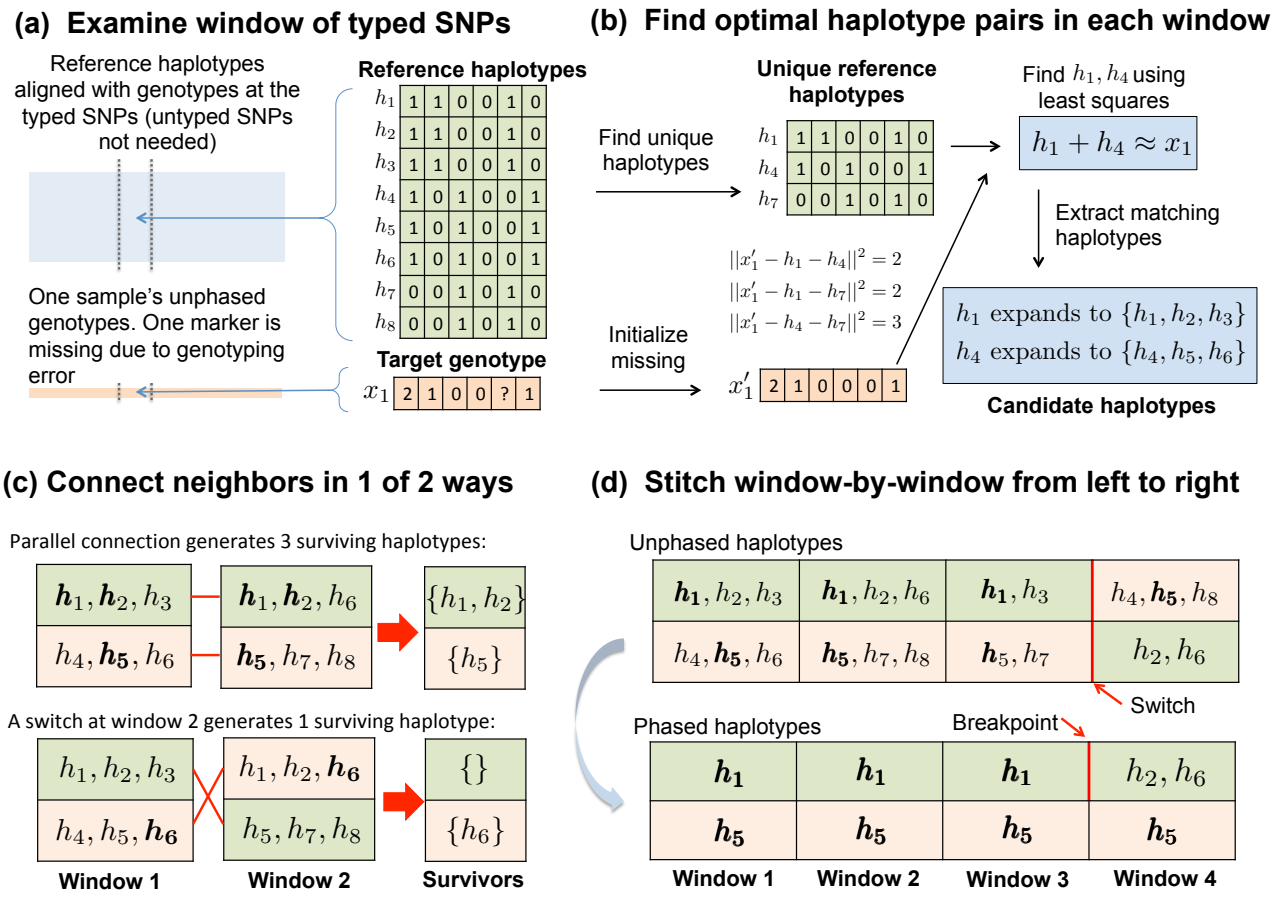


Figure 3.1: Overview of `MendelImpute`'s algorithm. (a) After alignment, imputation and phasing are carried out on short, non-overlapping windows of the typed SNPs. (b) Based on a least squares criterion, we find two unique haplotypes whose vector sum approximates the genotype vector on the current window. Once this is done, all reference haplotypes corresponding to these two unique haplotypes are assembled into two sets of candidate haplotypes. (c) We intersect candidate haplotype sets window by window, carrying along the surviving set and switching orientations if the result generates more surviving haplotypes. (d) After three windows the top extended chromosome possess no surviving haplotypes, but a switch to the second orientation in the current window allows $\mathbf{h}_5$ to survive on the top chromosome. Eventually we must search for a break point separating $\mathbf{h}_1$ from $\mathbf{h}_2$ or $\mathbf{h}_6$ between windows 3 and 4 (bottom panel).

To minimize the criterion (3.1) efficiently, suppose the genotype vectors $\mathbf{x}_i$ constitute the columns of a genotype matrix $\mathbf{X}$, and suppose the haplotype vectors $\mathbf{h}_i$ constitute the columns of a haplotype matrix $\mathbf{H}$. Given these conventions we recover all inner products $\mathbf{x}_i^T\mathbf{h}_j$ and $\mathbf{h}_i^T\mathbf{h}_j$ in equation (3.1) as entries of two matrix products; the two corresponding BLAS (Basic Linear Algebra Subroutines)[64] level-3 calls produce

$$\mathbf{X}^T\mathbf{H} = \begin{pmatrix} \mathbf{x}_1^T\mathbf{h}_1 & \cdots & \mathbf{x}_1^T\mathbf{h}_d \\ \vdots & & \vdots \\ \mathbf{x}_n^T\mathbf{h}_1 & \cdots & \mathbf{x}_n^T\mathbf{h}_d \end{pmatrix}_{n\times d} \quad \text{and} \quad \mathbf{H}^T\mathbf{H} = \begin{pmatrix} \|\mathbf{h}_1\|_2^2 & \cdots & \mathbf{h}_1^T\mathbf{h}_d \\ \vdots & & \vdots \\ \mathbf{h}_d^T\mathbf{h}_1 & \cdots & \|\mathbf{h}_d\|_2^2 \end{pmatrix}_{d\times d}. \tag{3.2}$$

These allow one to quickly assemble a matrix $\mathbf{M}$ with entries $m_{ij} = \|\mathbf{h}_i\|_2^2 + \|\mathbf{h}_j\|_2^2 + 2\mathbf{h}_i^T\mathbf{h}_j$ and for each sample $\mathbf{x}_k$ a matrix $\mathbf{N}$ with entries $n_{ij} = -2\mathbf{x}_k^T\mathbf{h}_i - 2\mathbf{x}_k^T\mathbf{h}_j$. Therefore, to find the best haplotype pair $(\mathbf{h}_i, \mathbf{h}_j)$ for the sample $\mathbf{x}_k$, we search for the minimum entry $m_{ij} + n_{ij}$ of the $d \times d$ matrix $\mathbf{M} + \mathbf{N}$ across all indices $i \geq j$. Extremely unlikely ties are arbitrarily broken. Note that the constant term $||\mathbf{x}_k||_2^2$ can be safely ignored in optimization. Data import and this minimum entry search are the computational bottlenecks of our software. Once such a haplotype pair is identified, all reference haplotype pairs identical to $(\mathbf{h}_i, \mathbf{h}_j)$ in the current window give the same optimal ℓ_2 error.

3.2.3 Phasing by Intersecting Haplotype Sets

As just described, each window w along a sample chromosome generates an optimal pair of unique haplotypes. These expand into two sets S_{w1} and S_{w2} of reference haplotypes (Figure 3.1B). In the first window we arbitrarily assign S_{11} to extended haplotype 1 and S_{12} to extended haplotype 2. From here on the goal is to reconstruct two extended composite haplotypes $\mathbf{E}_1$ and $\mathbf{E}_2$ that cover the entire chromosome. Let w index the current window. The two sets $S_{w-1,1}$ and $S_{w-1,2}$ are already phased. The new sets S_{w1} and S_{w2} are not, and their phases must be resolved and their entries pruned by intersection to achieve extended haplotype parsimony. The better orientation is one which generates more surviving haplotypes after intersection (Figure 3.1C). Here we count surviving haplotypes

across both sets of an orientation. The better orientation and the corresponding survivor sets are propagated to subsequent windows. If either intersection is empty at window w, then a break is declared, the empty set is replaced by the entire haplotype set of window w, and a new reference segment commences (Figure 3.1D). Ties and double empties virtually never occur. Repeated intersection may fail to produce singleton haplotype sets, in which case we randomly designate a winner to use for breakpoint search.

For example, suppose $S_{11} = \{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$ and $S_{12} = \{\mathbf{h}_4, \mathbf{h}_5, \mathbf{h}_6\}$ are the (arbitrarily) phased sets in window 1. Since window 2 is not yet phased, the two sets $S_{21} = \{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_6\}$ and $S_{22} = \{\mathbf{h}_5, \mathbf{h}_7, \mathbf{h}_8\}$ can be assigned to extended haplotypes 1 and 2, respectively, or vice versa as depicted in Figure 3.1C. The first orientation is preferred since it generates two surviving haplotypes $\mathbf{h}_1$ and $\mathbf{h}_5$ bridging windows 1 and 2. Thus, $\{\mathbf{h}_1\}$ and $\{\mathbf{h}_5\}$ are assigned at window 2 with this orientation and propagated to window 3. In window 3 the contending pairs are $\{\mathbf{h}_1\} \cap \{\mathbf{h}_1, \mathbf{h}_3\}$ and $\{\mathbf{h}_5\} \cap \{\mathbf{h}_2, \mathbf{h}_5\}$ versus $\{\mathbf{h}_1\} \cap \{\mathbf{h}_2, \mathbf{h}_5\}$ and $\{\mathbf{h}_5\} \cap \{\mathbf{h}_1, \mathbf{h}_3\}$. The former prevails, and $\{\mathbf{h}_1\}$ and $\{\mathbf{h}_5\}$ are assigned to window 3 and propagated to window 4. In window 4 the opposite orientation is preferred (Figure 3.1D). In this empty intersection case we set $S_{41} = \{\mathbf{h}_2, \mathbf{h}_6\}$ and $S_{42} = \{\mathbf{h}_5\}$ and continue the process. Later we return and resolve the breakpoint in extended haplotype 1 between windows 3 and 4.

3.2.4 Resolving Breakpoints

The unique haplotype pairs found for adjacent windows are sometimes inconsistent and yield empty intersections. In such situations, we search for a good break point. Figure 3.1D illustrates a single-breakpoint search. In this example, we slide the putative break point b across windows 3 and 4 in the top extended haplotype to minimize the least squares value determined by the observed genotype, $\mathbf{h}_5$ spanning both windows, and the breakpoint b between $\mathbf{h}_1$ and $\mathbf{h}_2 \cup \mathbf{h}_6$. When there is a double mismatch, we must search for a pair (b_1, b_2) of breakpoints, one for each extended haplotype. The optimal pair can be determined by minimizing the least squares distances generated by all possible breakpoint pairs (b_1, b_2). Thus, double breakpoint searches scale as a quadratic. Fortunately, under the adaptive window width strategy described in Section 3.5.2.1, the number of typed SNPs in each

Data Set	size (MB) for format:				
	vcf.gz	jlso	bref3	m3vcf.gz	imp5
sim 10K	49	6	18	7	57
sim 100K	472	56	81	29	565
sim 1M	4660	776	415	NA	5650
1000G chr10	459	188	342	116	621
1000G chr20	200	89	154	52	279
HRC chr 10	3346	1328	1156	529	3636
HRC chr 20	1510	706	554	253	1610

Table 3.1: Storage size required for various compressed reference haplotype formats. Here `vcf.gz` is the standard compressed VCF format, `jlso` is used by `MendelImpute`, `bref3` is used by `Beagle` 5.1, `m3vcf.gz` is used by `Minimac` 4, and `imp5` is used by `Impute` 5. For all `jlso` files we chose the maximum number of unique haplotypes per window to be $d_{max} = 1000$. Note we could not generate the `m3vcf.gz` file for the sim 1M panel because it required too much memory (RAM).

window typically is on the order of 10^2. In this range, quadratic search remains fairly efficient.

3.2.5 Imputation and Phasing of Untyped SNPs

Once haplotyping is complete, it is trivial to impute missing SNPs. Each missing SNP is located on the reference map, and its genotype is imputed as the sum of the alleles on the extended haplotypes $\mathbf{E}_1$ and $\mathbf{E}_2$. Observed genotypes are untouched unless the user prefers phased genotypes. In this case `MendelImpute` will override observed genotypes with phased haplotypes similar to `Minimac` 4. Unfortunately, `MendelImpute` cannot compute estimated dosages. As shown in Section 3.3.4 on alternative compression schemes, the extended haplotypes $\mathbf{E}_1$ and $\mathbf{E}_2$ can be output rather than imputed genotypes at the user's discretion.

3.2.6 Compressed Reference Panels

Large reference files are typically stored as compressed VCF files. Since VCF files are plain text files, they are large and slow to read. Read times can be improved by computing and storing an additional tabix index file [65], but file size remains a problem. Consequently, every modern imputation program has developed its own specialized reference file format (for instance, the `m3vcf`, `imp5`, and

Data Set	Total SNPs	Typed SNPs	Samples	Ref Haplotypes	Missing %
Sim 10K	62704	22879	1000	10000	0.5
Sim 100K	80029	23594	1000	100000	0.5
Sim 1M	97750	23092	1000	1000000	0.5
1000G Chr10	3968020	192683	52	4904	0.1
1000G Chr20	1802261	96083	52	4904	0.1
HRC Chr10	1809068	191210	1000	52330	0.1
HRC Chr20	829265	95414	1000	52330	0.1

Table 3.2: Summary of real and simulated data sets used in our experiments. For the 1000G and HRC datasets, the SNPs also present on the Infinium Omni5-4 Kit constitute the typed SNPs. Missing % is the percentage of typed SNPs randomly masked to mimic random genotyping error. (* We used the top 50,000 most ancestry informative SNPs.)

`bref3` formats of `Minimac`, `Impute`, and `Beagle`, respectively) for improving read times and storage efficiency. We propose yet another compressed format for this purpose: the `jlso` format, and we compare it against other formats in Table 3.1. Details for generating the `jlso` format are discussed in the supplement.

3.2.7 Real and Simulated Data Experiments

For each data set, we use only bi-allelic SNPs. Table 3.2 summarizes the real and simulated data used in our comparisons.

3.2.7.1 Simulated Data

To test scalability across a broad range of panel sizes, we simulated three 10 Mb sequence data sets with 12,000, 102,000, and 1,002,000 haplotypes using the software `msprime` [55]. We randomly selected 1000 samples (2000 haplotypes) from each pool to form the target genotypes and used the remaining to form the reference panels. The SNPs with minor allele frequency greater than 5% were designated the typed SNPs. Approximately 0.5% of the typed genotypes were randomly masked to introduce missing values.

3.2.7.2 1000 Genomes Data

We downloaded the publicly available 1000 Genomes (1000G) phase 3 data set [1, 2] from Beagle's website [20]. The original 1000 genomes data set contains 2504 samples with 49,143,605 phased genotypes across 26 different populations, as summarized in Table 3.5 in the supplement. In our study, structural variants and markers with < 5 minor alleles or non-unique identifiers were excluded. We focused on chromosomes 10 and 20 data in our speed and accuracy experiments. Following [96, 20], we randomly selected 2 samples from each of the 26 populations to serve as imputation targets. All other samples became the reference panel. We chose SNPs present on the Infinium Omni5-4 Kit to be typed SNPs and randomly masked 0.1% of the typed genotypes to mimic data missing at random.

For ancestry inference experiments, we chose the top 50,000 most ancestry informative markers (AIMs) on each chromosome with minor allele frequency ≥ 0.01 as the typed SNPs [18]. The AIM markers were computed using `VCFTools.jl`. Samples from ACB, ASW, CLM, MXL, PEL, and PUR (African Caribbeans, Americans of African Ancestry, Colombians, Mexican Ancestry, Peruvians, and Puerto Ricans, respectively; $n = 504$) are assumed admixed and employed in admixture experiments. Samples from the 20 remaining populations $(n = 2000)$ are assumed to exhibit less continental-scale admixture and serve as the reference panel. Unfortunately, the indigenous Amerindian populations are not surveyed in the 1000 Genomes data, so we use East Asian (EAS) and South Asian (SAS) populations as the best available proxy.

3.2.7.3 Haplotype Reference Consortium Data

We also downloaded the Haplotype Reference Consortium (HRC) v1.1 data from the European Genotype-Phenome archive [63] (data accession = EGAD00001002729). This data set consists of 39,741,659 SNPs in 27,165 individuals of predominantly European ancestry. We randomly selected 1000 samples in chromosomes 10 and 20 to serve as imputation targets and the remaining to serve as the reference panel. SNPs also present on the Infinium Omni5-4 Kit are chosen to be typed SNPs.

Finally, we randomly masked 0.1% of the typed genotypes to mimic data missing at random.

3.3 Results

Due to Julia's flexibility, `MendelImpute` runs on Windows, Mac, and Linux operating systems, equipped with either Intel or ARM hardware.

3.3.1 Comparison setup

All programs were run on Linux CentOS 7 equipped with 10 cores of an Intel i9 9920X CPU and 64 GB of RAM. All reference files were previously converted to the corresponding compressed formats, `bref3`, `m3vcf`, `imp5`, or `jlso`. All target genotypes were unphased, and 0.1-0.5% of typed genotypes were deleted at random. Since `Minimac` 4 and `Impute` 5 required pre-phased data, we used `Beagle` 5.1's built-in pre-phasing algorithm and report its run-time and RAM usage beside their run times. All output genotypes are phased and complete.

`MendelImpute` was run with Julia v1.5.0. The maximum number of unique haplotypes per window (see supplement on `jlso` compression) was set to $d_{max} = 1000$, and the number of BLAS threads was set to 1 to avoid over-subscription. `Beagle` and `Minimac` were run under their default settings. `Impute` 5 was run on 20Mb chunks corresponding to different chromosome regions, except for chromosome 10 of HRC. Chunks were initialized in parallel and imputed separately. Each chunk potentially employs multithreading. We used as many threads as possible without exceeding 10 total threads over all chunks. For chromosome 10 of HRC, we imputed 10Mb chunks, with a maximum of 8 processes active at any given time. Using the maximum 10 processes or employing longer chunks resulted in out of memory error.

sim 10K	Error Rate	Time (sec)	Memory (GB)
MendelImpute	3.00E-04	**10**	**1.6**
Impute 5	2.82E-05	43	8.9 [6.6]
Beagle 5.1	2.81E-05	189	8.8
Minimac 4	**2.38E-05**	271 [177]	1.0 [6.6]
sim 100K	Error Rate	Time (sec)	Memory (GB)
MendelImpute	2.19E-05	**14**	**1.6**
Impute 5	9.36E-06	74 [253]	9.9 [14.5]
Beagle 5.1	8.22E-06	279	20.1
Minimac 4	**7.91E-06**	3032 [253]	2.6 [14.5]
sim 1M	Error Rate	Time (sec)	Memory (GB)
MendelImpute	2.21E-05	**27**	**4.4**
Impute 5	1.33E-05	153 [752]	12.6 [25.6]
Beagle 5.1	**7.01E-06**	769	25.6
Minimac 4	NA	NA	NA
1000G chr10	Error Rate	Time (sec)	Memory (GB)
MendelImpute	7.52E-03	**34**	4.1
Impute 5	**4.79E-04**	35 [151]	5.6 [7.3]
Beagle 5.1	4.82E-04	178	26.2
Minimac 4	4.89E-04	298 [151]	**3.8** [7.3]
1000G chr20	Error Rate	Time (sec)	Memory (GB)
MendelImpute	7.49E-03	**14**	**2.2**
Impute 5	**4.48E-03**	21 [74]	5.5 [5.3]
Beagle 5.1	4.53E-03	88	6.4
Minimac 4	**4.48E-03**	243 [74]	2.8 [5.3]
HRC chr10	Error Rate	Time (sec)	Memory (GB)
MendelImpute	1.70E-03	**183**	**8.4**
Impute 5	**8.00E-04**	754 [2619]	47.6 [13.0]
Beagle 5.1	8.34E-04	3191	33.2
Minimac 4	8.57E-04	17 100 [2619]	11.5 [13.0]
HRC chr20	Error Rate	Time (sec)	Memory (GB)
MendelImpute	1.89E-03	**84**	**5.1**
Impute 5	**8.57E-04**	645 [1319]	39.0 [13.3]
Beagle 5.1	9.10E-04	1549	22.1
Minimac 4	9.18E-04	8640 [1319]	15.1 [13.3]

Table 3.3: Error, time, and memory comparisons on real and simulated data. The best number in each cell is bold faced. The displayed error rate is the proportion of incorrectly imputed genotypes. Minimac 4 and Impute 5's benchmarks include the pre-phasing step done by Beagle 5.1, whose memory and time are reported in brackets. For the sim 1M data, the m3vcf reference panel required for Minimac could not be computed due to excessive memory requirements.

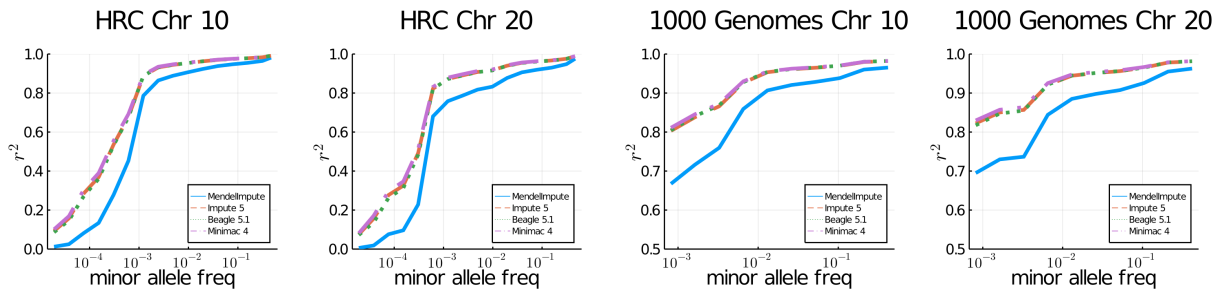


Figure 3.2: Imputation accuracy for imputed genotypes of the 1000 Genomes Project and the Haplotype Reference Consortium. Imputed alleles are binned according to minor allele frequency in the reference panel.

3.3.2 Speed, Accuracy, and Peak Memory Demand

Table 3.3 compares the speed, raw imputation accuracy, and peak memory (RAM) usage of `MendelImpute`, `Beagle` 5.1, `Impute` 5 version 1.1.4, and `Minimac` 4. Following the earlier studies [20, 31, 96], Figure 3.2 additionally plots the squared Pearson correlation r^2 between the vector of genotypes and the vector of imputed posterior genotype dosages. Note that we binned imputed minor alleles according to the minor allele count in the reference panel. For `MendelImpute` we plot squared r^2 between true genotypes and the imputed genotypes. Memory was measured via the `usr/bin/time` command, except for `Impute` 5 where memory is monitored manually via the `htop` command.

On HRC, `MendelImpute` runs 18-23 times faster than `Impute` 5 (including pre-phasing time), 17-18 times faster than `Beagle` 5.1, and 108-119 times faster than `Minimac` 4. On the smaller 1000 Genomes data set, `MendelImpute` runs 5-7 times faster than `Impute` 5, 5-6 times faster than `Beagle` 5.1, and 10-23 times faster than `Minimac` 4. Increasing the reference panel size by a factor of 100 on simulated data only increases `MendelImpute`'s computation time by a factor of at most three. `MendelImpute` also scales better than HMM methods as the number of typed SNPs increase. This improvement is evident from the timing results for simulated and real data. Thus, denser SNP arrays may benefit disproportionately from using `MendelImpute`. The 1000 Genomes data set is exceptional in that it has fewer than 5000 reference haplotypes. Therefore, traversing that HMM state space is not much slower than performing the corresponding linear algebra calculations in

MendelImpute. Notably, except for the HRC panels, MendelImpute spends at least 50% of its total compute time importing data.

In terms of error rate, MendelImpute is 1.5-1.7 times worse 1000 Genomes data and 2.1-2.2 worse on the HRC compared to the best HMM method. This translates to slightly lower r^2 values, particularly for rare alleles. On simulated data, our error rate is 2.7-10 times worse. Note the 1000 genomes have higher aggregate r^2 because the panel have undergone quality control procedures. The error rates of Impute 5, Beagle 5.1, and Minimac 4 are similar, consistent with previous findings [96]. As discussed in the supplement, it is possible to improve MendelImpute's error rate by computationally intensive strategies such as phasing by dynamic programming.

Finally, MendelImpute requires much less memory for most data sets. As explained in the methods section and the supplement, the genotype matrix and compressed reference panel are compactly represented in memory. Since most analysis is conducted in individual windows, only small sections of these matrices need to be decompressed into single-precision arrays at any one time. Consequently, MendelImpute uses at most 8.4 GB of RAM in each of these experiments. In general, MendelImpute permits standard laptops to conduct imputation even with the sizeable HRC panels.

3.3.3 Ancestry Inference for Admixed Populations

MendelImpute lends itself to chromosome painting, the process of coloring each haplotype segment by the country, region, or ethnicity of the reference individual assigned to the segment. For chromosome painting to be of the most value, reference samples should be representative of distinct populations. Within a reference population there should be little admixture. Also the colors assigned to different regions should be coordinated by physical, historical, and ethnic proximity. The overall proportions of the colors assigned to a sample individual genome immediately translate into global admixture coefficients. Here we illustrate chromosome painting using chromosome 18 data from the 1000 Genomes Project. The much larger Haplotype Reference Consortium data would

be better suited for chromosome painting, but unfortunately its repository does not list country of origin. Our examples should therefore be considered simply as a proof of principle. As already mentioned, the populations present in the 1000 Genomes Project data are summarized in Table 3.5 of the supplement.

Figure 3.3A displays the painted chromosomes 18 of a native Puerto Rican (PUR, sample 1), a Peruvian from Lima, Peru (PEL, sample 2), and a person of African ancestry from the Southwest USA (ASW, sample 3). Here a total of 20 reference populations potentially contribute genetic segments. They are colored with red, brown, blue, or green to capture South Asian, East Asian, European, or African backgrounds, respectively. Note that the samples from South/East Asian populations serve as a proxy for Amerindian ancestral populations. After coloring, the two PUR extended haplotypes are predominantly blue, the two PEL haplotypes are predominantly red/brown and blue, while the two ASW haplotypes are predominantly green. Interestingly, one of the PUR and one of the PEL haplotypes contain a block of African origin as well as blocks of Asian and European origin, while the ASW haplotypes contain two blocks of European origin. The relatively long blocks are suggestive of recent admixture. The resulting chromosome barcodes vividly display population origins and suggest the locations of ancient or contemporary recombination events.

We compared `MendelImpute` to our ADMIXTURE [6] software using $K = 4$ populations. Figure 3.3B displays every admixed sample's ($n = 504$) global admixture proportions. Both programs qualitatively agree for samples from the CLM, MXL, and PUR populations. However, `ADMIXTURE` assigned $\sim 25\%$ South Asian ancestry to the two populations with African ancestry (ACB, ASW) whereas `MendelImpute` predicts $\sim 20\%$ Europeans for them. Previous studies [21, 68] suggest Europeans constitute $\sim 20\%$ ancestry in African Americans, so `MendelImpute` delivered the more reliable estimate. On the other hand, `ADMIXTURE` finds $> 70\%$ East Asian ancestry (proxy for Amerindians) for Peruvians (PEL), whereas `MendelImpute` predicts 50-50 split between European and South/East Asians. Based on previous admixture studies [88, 87], Peruvians tend to have 50% to 90% Amerindian ancestries, so `ADMIXTURE` appears more reliable for them. It is noteworthy that the Peruvian studies both used the `ADMIXTURE` software to infer ancestry, while the African Amer-

Data Set	vcf.gz size (MB)	ultra-compressed size (MB)	compression ratio
Sim 10K	10.07	0.05	201
Sim 100K	10.71	0.04	267
Sim 1M	11.05	0.04	276
1000G Chr10	24.04	0.43	56
1000G Chr20	10.69	0.25	43
HRC Chr10	157.76	6.01	26
HRC Chr20	70.93	3.47	20

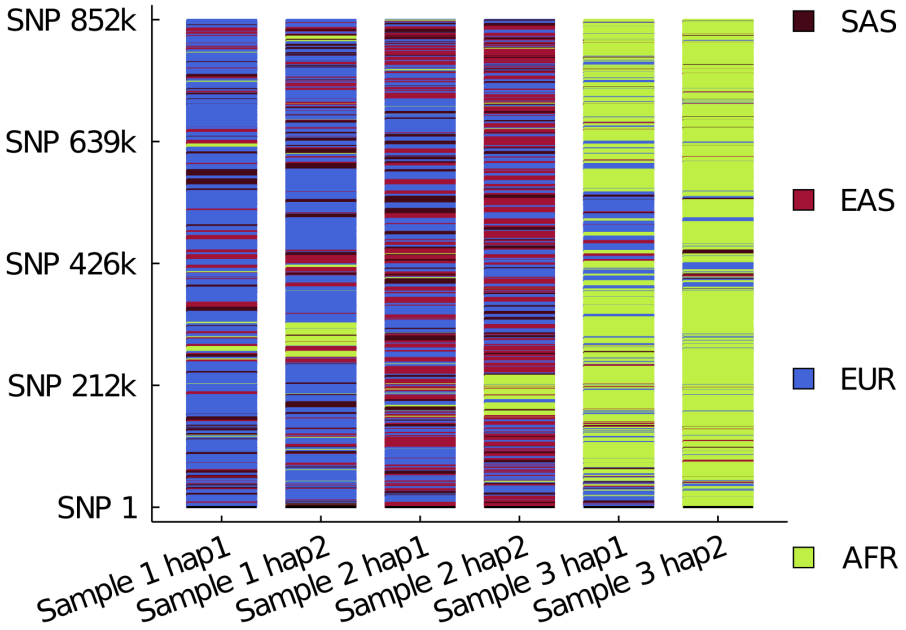
Table 3.4: Output file size comparison of compressed VCF and ultra-compressed formats.

ican studies did not. Nevertheless, the fact that `MendelImpute` sometimes performs better than `ADMIXTURE` suggests caution. The addition of proper Amerindian samples to the reference panels would likely clarify results and bring the two programs into closer agreement.

3.3.4 Ultra-Compressed Phased Genotype Files

As discussed earlier, VCF files are enormous and slow to read. If genotypes are phased with respect to a particular reference panel, then an alternative is to store each haplotype segment's starting position and a pointer to its corresponding reference haplotype. This offers massive compression because long haplotype segments are reduced to two integers. Instead of outputting compressed VCF files, which is the default, `MendelImpute` can optionally output such ultra-compressed phased data. Table 3.4 shows that the ultra-compressed format gives 20-270 fold compression compared to standard compressed VCF outputs. In principle, all phased genotypes can be stored in such files. The drawback is that compressed data can only be decompressed with the help of the original reference panel. Thus, this tactic relies on universal storage and curation of reference haplotype panels. These panels should be stored on the cloud for easy access and constructed so that they can be consistently augmented by new reference haplotypes.

(a) Chromosome 18: local admixture with `MendelImpute` for 3 samples



(b) Genome-wide global admixture for 504 samples

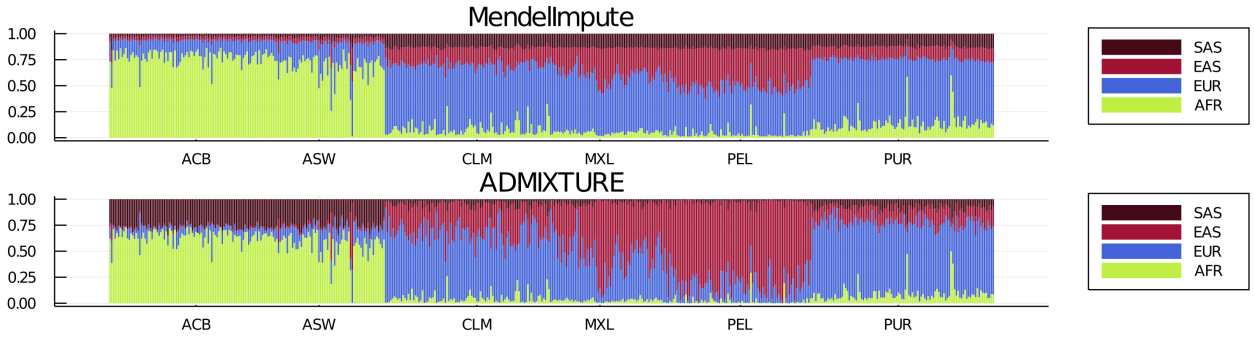


Figure 3.3: (a) Local ancestry inference on chromosome 18 using MendelImpute. Sample 1 is Puerto Rican (PUR), sample 2 is Peruvian (PEL), and sample 3 is African American (ASW). Other abbreviations are explained in the text. (b) Every sample's ($n = 504$) global ancestry proportions estimated using every chromosome. Each column is a sample's admixture proportion. Samples in (a) are located at index 2, 339, and 500.

3.4 Discussion

We present MendelImpute, the first scalable, data-driven method for phasing and genotype imputation. MendelImpute and supporting OpenMendel software [128] provide an end-to-end analysis pipeline in the Julia programming language that is typically 10–100 times faster than methods based on hidden Markov models, including Impute 5, Beagle 5.1 (Java) and Minimac 4 (C++). The speed difference increases dramatically as we increase the number of typed SNPs. Thus, denser SNP chips potentially benefit more from MendelImpute's design. Furthermore, MendelImpute occupies a smaller memory footprint. This makes it possible for users to run MendelImpute on standard desktop or laptop computers on very large data sets such as the HRC. Unfortunately we cannot yet have the best of both worlds, as MendelImpute exhibits a 1.5-2.2 fold worse error rate on real data. However, as seen in Table 3.3, MendelImpute's error rate is still acceptably low. One can improve its error rate by implementing strategies that detect recombinations within windows [69] or that phase by dynamic programming as discussed in the appendix. Regardless, it is clear that big data methods can compete with HMM based methods on the largest data sets currently available and that there is still room for improvement and innovation in genotype imputation.

Beyond imputation and phasing, our methods extend naturally to ancestry estimation and data compression. If each reference haplotype is labeled with its country or region of origin, then MendelImpute can decompose a sample's genotypes into segments of different reference haplotypes colored by these origins. The cumulative lengths of these colored segments immediately yield an estimate of admixture proportions. These results are comparable and possibly superior to those provided by ADMIXTURE [6]. Countries can be aggregated into regions if too few reference haplotypes originate from a given country. The colored segments also present a chromosome barcode that helps one visualize subject variation, recombination hotspots, and global patterns of linkage disequilibrium. Data compression is achieved by storing the starting positions of each segment and its underlying reference haplotype. This leads to output files that are 20-270 fold smaller than standard compressed VCF files. Decompression obviously requires ready access to stable reference panels

stored on accessible sites such as the cloud. Although such an ideal resource is currently part dream and part reality, it could be achieved by a concerted international effort.

For potential users and developers, the primary disadvantage of `MendelImpute` is its reliance on the importation and storage of a haplotype reference panel. Acquiring these panels requires an application process which can take time to complete. Understanding, storing, and wrangling a panel add to the burden. The imputation server for `Minimac` 4 thrives because it relieves users of these burdens [31]. `Beagle` 5.1 and `Impute5` are capable of fast parallel data import on raw VCF files [20] that neither `Minimac` 4 nor `MendelImpute` can currently match. This makes target data import, and especially pre-processing the reference panel, painfully slow for both programs. Fortunately, pre-processing only has to happen once.

Finally, let us reiterate the goals and achievements of this paper. First, we show that data-driven methods are competitive with HMM methods on genotype phasing and imputation, even on the largest data sets available today. Second, we challenge the notion that pre-phasing and imputation should be kept separate; `MendelImpute` performs both simultaneously. Third, we argue that data-driven methods are ultimately more flexible; for instance, `MendelImpute` readily handles imputation and phasing on dosage data. Fourth, we demonstrate that data-driven methods yield dividends in ancestry identification and data compression. Fifth, `MendelImpute` is completely open source, freely downloadable, and implemented in Julia, an operating system agnostic, high-level programming language for scientific research. Julia is extremely fast and enables clear modular coding. Our experience suggests that data-driven methods will offer a better way forward as we face increasingly larger reference panels, denser SNP array chips, and greater data variability.

3.5 Supplemental Material

3.5.1 Imputation Quality Scores

Consider the observed genotype $x_{ij} \in [0,2] \cup \{missing\}$ at SNP i of sample j and the corresponding imputed genotype g_{ij} derived from the two extended haplotypes of j. If S_i denotes the set of individuals with observed genotypes at the SNP, then `MendelImpute`'s quality score q_i for the SNP is defined as

$$q_i \;=\; 1 - \frac{1}{|S_i|} \sum_{j \in S_i} \left(\frac{x_{ij} - g_{ij}}{2} \right)^2.$$

Note that $0 \leq q_i \leq 1$ and that the larger the quality score, the more confidence in the imputed values. Because q_i can only be computed for the typed SNPs, an untyped SNP is assigned the average of the quality scores for its two closest flanking typed SNPs. Figure 3.4A plots each SNP's quality score in the 1000G Chr20 experiment summarized in Table 3.3. For each sample, one can also compute the mean least squares error over all p SNPs to obtain a per-sample quality score. This is shown in Figure 3.4B. By default `MendelImpute` outputs both quality scores. Thus, investigators can perform post-imputation quality control by SNPs and by samples separately.

Empirically, it is rather common for a sample subject to harbor a few poorly imputed windows. Thus, we observe a long left tail in the histogram for per-sample error in Figure 3.4. Unfortunately, the bad windows generally do not exhibit any discernible regional patterns across subjects. We suspect that poorly imputed windows involve breakpoints that occur near the middle of a window. It is possible to detect such breakpoints by IBS matching and ancestry weighting [69]. We plan a detailed analysis of this issue in future work.

3.5.2 JLSO Compressed Reference Haplotype Panels

The `jlso` format is constructed in three steps: (a) specify window intervals, (b) compute unique haplotypes in addition to hash maps to reference haplotypes in each window, and (c) save the result

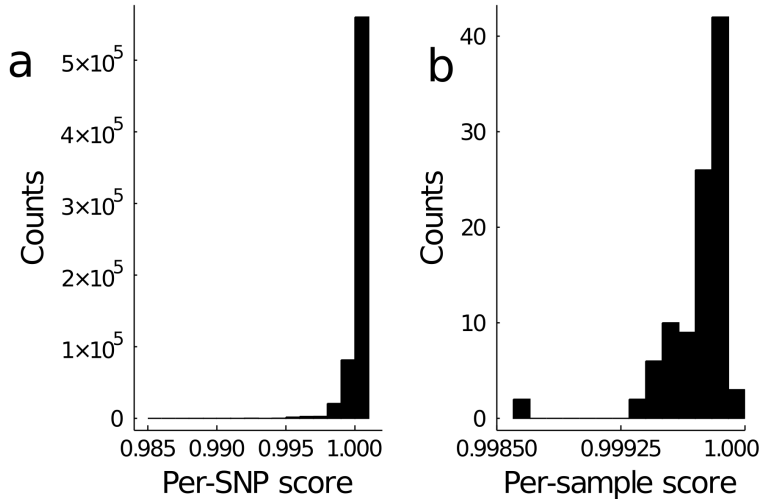


Figure 3.4: Histograms of per-SNP and per-sample quality scores for chromosome 20 in our 1000G analysis. By default `MendelImpute` computes (a) per-SNP quality scores and (b) per-sample quality scores. SNPs and samples with noticeably lower quality scores should be removed from downstream analysis.

in a binary compressed format via the `JLSO.jl` package [37]. The resulting `jlso` files are 30-50x faster to read and 3-5x smaller in file size (varies depending on window width) than compressed VCF files in the `vcf.gz` format. Note the `jlso` format is simply a container object that facilitates reading and transferring large VCF files stored as Julia variables. In principle, all files that are slow to read can be pre-processed and stored in this alternative format for quicker access. As such, we similarly store ultra-compressed sample haplotypes, as discussed in Section 3.3.4, using the `JLSO.jl` package.

3.5.2.1 Adaptive Window Widths via Recursive Bisection

The first step in generating JLSO haplotype panel is to specify genomic window ranges. The width of genomic windows is an important parameter determining both imputation efficiency and accuracy. Empirically, larger window widths give better error rates but also increase the computational burden of the matrix multiplications and minimum entry search described in Section 3.2.2. The magnitudes of these burdens depend on local haplotype diversity. Thus, we choose window widths dynamically.

This goal is achieved by a bisection strategy. After aligning all typed SNPs with the reference panel, initially we view all typed SNPs on a large section of a chromosome as belonging to a single window. We then divide the window into equal halves if it possesses too many unique haplotypes. Each half is further bisected and so forth recursively until every window contains fewer than a predetermined number of unique haplotypes. Empirically, choosing the maximum number d_{max} of unique haplotypes per window to be 1000 works well for both real and simulated data. When a larger number is preferred, we resort to a stepwise search heuristic for minimizing criterion (3.1) that scales linearly in the number of unique haplotypes d. This heuristic is described above.

3.5.2.2 Elimination of Redundant Haplotypes by Hashing

Within a small genomic window of the reference panel, multiple haplotype pairs may be identical at the typed SNPs. Only the unique haplotypes play a role in matching reference haplotypes to sample genotypes. `MendelImpute` identifies redundant haplotypes by hashing. For each reference haplotype limited to the window, hashing stores an integer representation of the haplotype via a hash function. This integer serves as an index (key) to locate the reference haplotype (value). Put another way, hashing stores the inverse images of the map from reference haplotypes to unique haplotypes. In our software, the `GroupSlices.jl` package [46] identifies a unique key for each haplotype.

3.5.2.3 Save in binary compressed format

Since haplotypes are long binary vectors, the entire haplotype reference panel can be compactly represented using a single bit per entry. We have already divided the full reference panel into non-overlapping windows of various widths after proper alignment of all typed and reference SNPs in the window. For each window we save two compressed mini-panels. The first houses the unique haplotypes determined by just the typed SNPs. The second houses the unique haplotypes determined by all SNPs in the window, typed or untyped. The former is much smaller than the later, but each entry of both can be compactly represented by a single bit per entry in memory. Thus, for

each window we save two compressed windows in addition to meta information and pointers that coordinate reference haplotypes with the two mini-panels per each window. This whole ensemble is stored in the jlso file. Because there are only a limited number of SNP array chips on the market, one can in principle store just a few jlso files on a universal source such as the cloud. The same JLSO compressed panel can also be re-used as long as the set of typed SNPs does not change drastically between different GWAS chips.

3.5.3 Parallel Computing and Memory Requirements

`MendelImpute` employs a shared-memory parallel computing model where each available core handles an independent component of the entire problem. Work is assigned via Julia's multi-threading functionality. When computing the optimal haplotype pairs in equation (3.1), we parallelize over windows. This requires allocating c copies of $\mathbf{X}^T\mathbf{H}$ and $\mathbf{H}^T\mathbf{H}$, where c is the number of CPU cores available. Note the dimensions of these matrices vary across windows. To avoid accruing memory allocations, we pre-allocate c copies of $n \times d_{max}$ and $d_{max} \times d_{max}$ matrices and re-use their top-left corners in windows with $d < d_{max}$. For intersecting adjacent reference haplotype sets (phasing), we parallelize over samples. This step requires no additional memory. Writing to output is also trivially parallelizable by assigning each thread to write a different portion of the imputed matrix to a different file, then concatenating these files into a single output file. Data import is not parallelized. Beyond allocating $\mathbf{X}^T\mathbf{H}$ and $\mathbf{H}^T\mathbf{H}$, our software requires enough memory (RAM) to load the target genotype matrix and the compressed haplotype reference panel.

3.5.4 Bias Correction for Initializing Missing Data

Since BLAS requires complete data, we must first initialize the missing data in each genotype vector $\mathbf{x}$ before computing $\mathbf{M}$ and $\mathbf{N}$ in equation (3.2). This may introduce bias in our minimization of criterion (3.1) if there is a high fraction of missing genotypes in the typed SNPs, for example above 10%. One way to alleviate bias is to initialize missing data with the mean and save all unique

haplotype pairs minimizing criterion (3.1) under this convention. Once this set of optimal haplotype pairs are identified, we re-minimize criterion (3.1) but now skipping the missing entries of $\mathbf{x}$. That is equivalent to setting $x_k - h_{ik} - h_{jk} = 0$ when x_k is missing.

3.5.5 Avoidance of Global Searches for Optimal Haplotype Pairs

Recall that minimizing the criterion (3.1) requires searching through all lower-triangular entries of the $d \times d$ matrix $\mathbf{M} + \mathbf{N}$, where d denotes the number of unique haplotypes in the window. When $d < 1000$, searching through all $\binom{d}{2} + d$ lower-triangular entries of $\mathbf{M} + \mathbf{N}$ via `MendelImpute`'s standard procedure is fast, but this global search quickly degrades as $d \to \infty$. Below we outline two heuristic procedures for large d. These heuristics typically produce sub-optimal solutions compared to global searches, so they should be used with caution.

3.5.5.1 Stepwise Search Heuristics

Consider minimizing the loss $f_i(\boldsymbol{\beta}) = \frac{1}{2}||\mathbf{x}_i - \mathbf{H}\boldsymbol{\beta}||_2^2$, where the d columns of $\mathbf{H} \in \{0,1\}^{p \times d}$ store unique haplotypes, p is the window width, and $\mathbf{x}_i$ is a sample genotype vector. The original problem (3.1) minimizes $f_i(\boldsymbol{\beta})$ under the constraint that exactly two $\beta_j = 1$ and the remaining $\beta_k = 0$ or the constraint that exactly one $\beta_j = 2$ and the remaining $\beta_k = 0$. As an approximate alternative, one first finds the r unique haplotypes with the largest influence on $f_i(\boldsymbol{\beta})$. This is accomplished by identifying the r most negative components of the gradient

$$\nabla f_i(\boldsymbol{\beta}) = -\mathbf{H}^T(\mathbf{x}_i - \mathbf{H}\boldsymbol{\beta}) = -\mathbf{H}^T\mathbf{x}_i + \mathbf{H}^T\mathbf{H}\boldsymbol{\beta}$$

at $\boldsymbol{\beta} = \mathbf{0}$. These are the r directions of steepest descent. Note that $\nabla f_i(\mathbf{0}0) = -\mathbf{H}^T\mathbf{x}_i$ and that $\mathbf{H}^T\mathbf{x}_i$ is pre-computed and cached in $\mathbf{N}$. The residual function $g_{ij}(\mathbf{h}_k) = \frac{1}{2}||\mathbf{x} - \mathbf{h}_j - \mathbf{h}_k||_2^2$ is then minimized over $\mathbf{h}_k$ to find the candidate pair $(\mathbf{h}_j, \mathbf{h}_k)$ generated by each of the vectors $\mathbf{h}_j$ determined by the gradient $\nabla f_i(\mathbf{0})$. The ingredients to perform these minimizations are already in hand. This heuristic scales as $\mathcal{O}(rd)$, much better than $\mathcal{O}(d^2)$ in equation (3.2). `MendelImpute` sets the default $r = 100$.

In the same spirit as the first step, one can alternatively find for each j the most negative component k of the gradient $\nabla f_i(\mathbf{e}_j)$, where $\mathbf{e}_j$ is the standard unit vector with 1 in position j. This again determines a nearly optimal pair $(\mathbf{h}_j, \mathbf{h}_k)$. Under this tactic the Gram matrix $\mathbf{H}^T\mathbf{H}$ comes into play. Note that $\mathbf{H}^T\mathbf{H}\mathbf{e}_j$ reduces to its jth column $\mathbf{v}_j$. Hence, no new matrix-by-vector multiplications are necessary in calculating $\nabla f_i(\mathbf{e}_j) = -\mathbf{H}^T\mathbf{x}_i + \mathbf{v}_j = \nabla f_i(\mathbf{0}0) + \mathbf{v}_j$.

Alternatively, one can find the best r unique haplotypes for a given sample $\mathbf{x}_i$ *en masse* by arranging all pairwise column distances of $\mathbf{X}$ and $\mathbf{H}$ in the matrix

$$\mathbf{R} = \begin{bmatrix} ||\mathbf{x}_1 - \mathbf{h}_1||_2^2 & \cdots & ||\mathbf{x}_n - \mathbf{h}_1||_2^2 \\ \vdots & & \vdots \\ ||\mathbf{x}_1 - \mathbf{h}_d||_2^2 & \cdots & ||\mathbf{x}_n - \mathbf{h}_d||_2^2 \end{bmatrix}_{d \times n}.$$

Then we partially sort each column of $\mathbf{R}$ to identify the top r haplotypes matching each sample $\mathbf{x}_i$. Here $\mathbf{R}$ is computed via the `Distance.jl` package of Julia, which internally performs BLAS level-3 calls analogous to computing $\mathbf{H}^T\mathbf{H}$ and $\mathbf{X}^T\mathbf{H}$. Instead of searching through all haplotypes to minimize $g_{ij}(\mathbf{h}_k)$ for a given sample $\mathbf{x}_i$, one can instead search only over the $\binom{r}{2} + r$ combinations of the top haplotypes. This allows one to entertain much larger values of r. Empirically, choosing $r = 800$ works well for most data sets.

3.5.6 Phasing by Dynamic Programming

We also investigated a dynamic programming strategy that gives the global solution for minimizing the number of haplotype breaks across the extended haplotypes $\mathbf{E}_1$ and $\mathbf{E}_2$. For each given haplotype pair $\mathbf{p}_1 = (\mathbf{h}_i, \mathbf{h}_j)$ in window w, we can compute the squared Hamming distance between it and the

pair $\mathbf{p}_2 = (\mathbf{h}_k, \mathbf{h}_l)$ in window $w+1$; in symbols

$$d(\mathbf{p}_1, \mathbf{p}_2) = \begin{cases} 0 & \mathbf{h}_i = \mathbf{h}_k, \mathbf{h}_j = \mathbf{h}_l \quad (0 \text{ breaks}) \\ 1 & \mathbf{h}_i = \mathbf{h}_k, \mathbf{h}_j \neq \mathbf{h}_l \quad (1 \text{ break}) \\ 1 & \mathbf{h}_i \neq \mathbf{h}_k, \mathbf{h}_j = \mathbf{h}_l \quad (1 \text{ break}) \\ 4 & \mathbf{h}_i \neq \mathbf{h}_k, \mathbf{h}_j \neq \mathbf{h}_l \quad (2 \text{ breaks}). \end{cases}$$

Observe that a double break is assigned an error of 4 to favor 2 single breaks across 3 windows as opposed to a double break plus a perfect match.

Now we describe a dynamic programming strategy for finding the two paths with the minimal number of unique haplotype breaks. We start with all candidate pairs $\mathbf{p}_i$ in the leftmost window and initialize sums $s_i = 0$ and traceback path vectors $\mathbf{t}_i$ to be empty. One then recursively visits all windows in turn from left to right. If w is the current window, then every candidate haplotype pair $\mathbf{p}_i$ in window w is connected to every candidate pair $\mathbf{p}_j$ in window $w+1$. The traceback path $\mathbf{t}_j$ is determined by the pair $\mathbf{p}_k$ minimizing $d(\mathbf{p}_i, \mathbf{p}_j)$. The traceback path $\mathbf{t}_j$ is constructed by appending $\mathbf{p}_k$ to $\mathbf{t}_k$ and setting $s_j = s_k + d(\mathbf{p}_k, \mathbf{p}_j)$. This process is continued until the rightmost window v is reached. At this point the pair $\mathbf{p}_j$ with lowest running sum s_j is declared the winner. The traceback path $\mathbf{t}_j$ allows one to construct the extended haplotypes $\mathbf{E}_1$ and $\mathbf{E}_2$ in their entirety. Unfortunately, too many haplotype pairs per window can overwhelm dynamic programming with large reference panels because one must enumerate and store all possible haplotype pairs in every genomic window for every individual. For large reference panels such as HRC, the number of possible haplotype pairs often exceeds 100,000 per window. Thus, it is impossible to store all of these pairs in memory. One partial recourse is to discard partial paths and associated partial termini $\mathbf{p}_i$ that are unpromising in the sense that their running sums s_i are excessively large. This does not completely alleviate the memory burden, and the approximate algorithm is burdened by extra bookkeeping. The bookkeeping of the exact algorithm is already demanding.

3.5.7 Msprime simulation script

```
# Usage: python3 msprime_script.py n ne seq recomb mut seed > full.vcf
# We used ne=10000; seq=10000000; seed=2020; recomb=2e-8; mut=2e-8
# n = 12000 or 102000 or 1002000

import msprime, sys

# parameters
sample_size = int(sys.argv[1])
effective_population_size = int(sys.argv[2])
sequence_length = int(sys.argv[3])
recombination_rate=float(sys.argv[4])
mutation_rate=float(sys.argv[5])
seed = int(sys.argv[6])

# run the simulation
ts = msprime.simulate(sample_size=sample_size, Ne=effective_population_size,
length=sequence_length, recombination_rate=recombination_rate, random_seed=seed)
model = msprime.InfiniteSites(msprime.NUCLEOTIDES)
ts = msprime.mutate(ts, rate=mutation_rate, model=model, random_seed=seed)

# print results (2 is for diploid, "legacy" = no matching positions)
with sys.stdout as vcffile:
    ts.write_vcf(vcffile, 2, position_transform="legacy")
```

Population Code	Description	Super Population
CHB	Han Chinese in Beijing, China	East Asian (EAS)
JPT	Japanese in Tokyo, Japan	East Asian (EAS)
CHS	Southern Han Chinese	East Asian (EAS)
CDX	Chinese Dai in Xishuangbanna, China	East Asian (EAS)
KHV	Kinh in Ho Chi Minh City, Vietnam	East Asian (EAS)
CEU	Utah Residents with NW European Ancestry	European (EUR)
TSI	Toscani in Italia	European (EUR)
FIN	Finnish in Finland	European (EUR)
GBR	British in England and Scotland	European (EUR)
IBS	Iberian Population in Spain	European (EUR)
YRI	Yoruba in Ibadan, Nigeria	Africans (AFR)
LWK	Luhya in Webuye, Kenya	Africans (AFR)
GWD	Gambian in Western Divisions in the Gambia	Africans (AFR)
MSL	Mende in Sierra Leone	Africans (AFR)
ESN	Esan in Nigeria	Africans (AFR)
ASW	Americans of African Ancestry in SW USA	Africans (AFR)
ACB	African Caribbeans in Barbados	Africans (AFR)
MXL	Mexican Ancestry from Los Angeles USA	Ad Mixed American (AMR)
PUR	Puerto Ricans from Puerto Rico	Ad Mixed American (AMR)
CLM	Colombians from Medellin, Colombia	Ad Mixed American (AMR)
PEL	Peruvians from Lima, Peru	Ad Mixed American (AMR)
GIH	Gujarati Indian from Houston, Texas	South Asian (SAS)
PJL	Punjabi from Lahore, Pakistan	South Asian (SAS)
BEB	Bengali from Bangladesh	South Asian (SAS)
STU	Sri Lankan Tamil from the UK	South Asian (SAS)
ITU	Indian Telugu from the UK	South Asian (SAS)

Table 3.5: The 26 population codes present in the 1000 genomes project.

3.5.8 Summary of 1000 Genomes Reference Panel

A total of 26 different populations contribute to the 1000 Genomes Project data set. These populations are further organized into five super population. While this information is freely available online, we summarize it in Table 3.5 for completeness.

3.6 Availability of source code

Project name: MendelImpute.jl

Project home page: https://github.com/OpenMendel/MendelImpute.jl

Supported operating systems: Mac OS, Linux, Windows

Programming language: Julia 1.5, 1.6

License: MIT

All commands needed to reproduce the following results are available at the `MendelImpute` site in the `manuscript` sub-folder. `SnpArrays.jl` is available at https://github.com/OpenMendel/SnpArrays.jl. `VCFTools.jl` is available at https://github.com/OpenMendel/VCFTools.jl. The Haplotype Reference Consortium data is available at https://www.ebi.ac.uk/ega/datasets/EGAD00001002729. Raw 1000 genomes data is available at `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/re` and Beagle's webpage http://bochet.gcc.biostat.washington.edu/beagle/1000_Genomes_phase3_v5a/ provides a quality controlled 1000 genomes data which we used in our experiments.

CHAPTER 4

Multivariate Genomewide Association Analysis with IHT

4.1 Introduction

Current statistical methods for genome-wide association studies (GWAS) can be broadly categorized as single variant or multi-variant in their genomic predictors. Multi-variant sparse models ignore polygenic background and assume that only a small number of single-nucleotide polymorphisms (SNPs) are truly causal for a given phenotype. Model fitting is typically accomplished via regression with penalties such as the least absolute shrinkage and selection operator (LASSO) [5, 94, 112, 127, 124], minimax concave penalty (MCP) [17, 122], iterative hard thresholding (IHT) [28, 56], or Bayesian analogues [48]. Linear mixed models (LMM) dominate the single variant space. LMMs control for polygenic background while focusing on the effect of a single SNP. LMMs are implemented in the contemporary programs GEMMA [130], BOLT [71], GCTA [54, 116], and SAIGE [129]. The virtues of the various methods vary depending on the genetic architecture of a trait, and no method is judged uniformly superior [43].

Although there is no consensus on the best modeling framework for single-trait GWAS, there is considerable support for analyzing multiple correlated phenotypes jointly rather than separately [43, 90, 106]. When practical, joint analysis (a) incorporates extra information on cross-trait covariances, (b) distinguishes between pleiotropic and independent SNPs, (c) reduces the burden of multiple testing, and (d) ultimately increases statistical power. Surprisingly, simulation studies suggest these advantages hold even if only one of multiple traits is associated with a SNP or if the correlation among traits is weak [43]. These advantages motivate the current paper and our search for an efficient

method for analyzing multivariate traits.

Existing methods for multivariate-trait GWAS build on the polygenic model or treat SNPs one by one. For instance, GEMMA [131] implements multivariate linear mixed models (mvLMM), mvPLINK [36] implements canonical correlation analysis, and MultiPhen [89] and Scopa [75] invert regression so that the genotypes at a single SNP become the trait and observed phenotypes become predictors.

To our knowledge, there are no sparse regression methods for multivariate-trait GWAS. In this paper, we extend IHT [14] to the multivariate setting and implement it in the Julia [12] package `MendelIHT.jl`, part of the larger `OpenMendel` statistical genetics ecosystem [128]. We have previously demonstrated the virtues of IHT compared to LASSO regression and single-SNP analysis for univariate GWAS [28, 56]. Since IHT assumes sparsity and focuses on mean effects, it is ill suited to capture polygenic background as represented in classic variance components models. In the sequel we first describe our generalization of IHT. Then we study the performance of IHT on simulated traits given real genotypes. These simulations explore the impact of varying the sparsity level k and the number of traits r. To demonstrate the potential of IHT on real large-scale genomic data, we also apply it to three hypertension related traits from the UK Biobank. These studies showcase IHT's speed, low false positive rate, and scalability to large numbers of traits. Our concluding discussion summarizes our main findings, limitations of IHT, and question worthy of future research.

4.2 Materials and Methods

4.2.1 Model Development

Consider multivariate linear regression with r traits under a Gaussian model. Up to a constant, the loglikelihood $\mathcal{L}(\mathbf{B}, \mathbf{\Gamma})$ for n independent samples is

$$\mathcal{L}(\mathbf{B}, \mathbf{\Gamma}) = \frac{n}{2} \log(\det \mathbf{\Gamma}) - \frac{1}{2} \operatorname{tr} \left[\mathbf{\Gamma} (\mathbf{Y} - \mathbf{B}\mathbf{X})(\mathbf{Y} - \mathbf{B}\mathbf{X})^T \right]. \tag{4.1}$$

The loglikelihood $\mathcal{L}(\mathbf{B},\mathbf{\Gamma})$ is a function of the $r \times p$ regression coefficients matrix $\mathbf{B}$ and the $r \times r$ unstructured precision (inverse covariance) matrix $\mathbf{\Gamma}$. Furthermore, $\mathbf{Y}$ is the $r \times n$ matrix of traits (responses), and $\mathbf{X}$ is the $p \times n$ design matrix (genotypes plus non-genetic predictors). All predictors are treated as fixed effects.

IHT maximizes $\mathcal{L}(\mathbf{B},\mathbf{\Gamma})$ subject to the constraints that k or fewer entries of $\mathbf{B}$ are non-zero and that $\mathbf{\Gamma}$ is symmetric and positive definite. Optimizing $\mathcal{L}(\mathbf{B},\mathbf{\Gamma})$ with respect to $\mathbf{B}$ for $\mathbf{\Gamma}$ fixed relies on three core ideas. The first is gradient ascent. Elementary calculus tells us that the gradient $\nabla_{\mathbf{B}}\mathcal{L}(\mathbf{B},\mathbf{\Gamma})$ is the direction of steepest ascent of $\mathcal{L}(\mathbf{B},\mathbf{\Gamma})$ at $\mathbf{B}$ for $\mathbf{\Gamma}$ fixed. IHT updates $\mathbf{B}$ in the steepest ascent direction by the formula $\mathbf{B}_{m+1} = \mathbf{B}_m + t_m \nabla_{\mathbf{B}}\mathcal{L}(\mathbf{B}_m,\mathbf{\Gamma}_m)$, where $t_m > 0$ is an optimally chosen step length and $(\mathbf{B}_m,\mathbf{\Gamma}_m)$ is the current value of the pair $(\mathbf{B},\mathbf{\Gamma})$. The gradient is derived in the appendix as the matrix

$$\nabla_{\mathbf{B}}\mathcal{L}(\mathbf{B},\mathbf{\Gamma}) \;=\; \mathbf{\Gamma}(\mathbf{Y}-\mathbf{B}\mathbf{X})\mathbf{X}^T. \tag{4.2}$$

The second core idea dictates how to choose the step length t_m. This is accomplished by expanding the function $t \mapsto \mathcal{L}[\mathbf{B}_m + t_m \nabla_{\mathbf{B}}\mathcal{L}(\mathbf{B}_m,\mathbf{\Gamma}_m)]$ in a second-order Taylor series around $(\mathbf{B}_m,\mathbf{\Gamma}_m)$. Our appendix shows that the optimal t_m for this quadratic approximant is

$$t_m = \frac{\|\mathbf{C}_m\|_F^2}{\mathrm{tr}(\mathbf{X}^T\mathbf{C}_m^T\mathbf{\Gamma}_m\mathbf{C}_m\mathbf{X})} \tag{4.3}$$

given the abbreviation $\mathbf{C}_m = \nabla_{\mathbf{B}}\mathcal{L}(\mathbf{B}_m,\mathbf{\Gamma}_m)$. The third core idea of IHT involves projecting the steepest ascent update $\mathbf{B}_{m+1} = \mathbf{B}_m + t_m \nabla_{\mathbf{B}}\mathcal{L}(\mathbf{B}_m,\mathbf{\Gamma}_m)$ to the sparsity set $S_k = \{\mathbf{B} : \|\mathbf{B}\|_0 \leq k\}$. The projection operator $P_{S_k}(\mathbf{B})$ sets to zero all but the largest k entries in magnitude of $\mathbf{B}$. This goal can be achieved efficiently by a partial sort on the vectorized version $vec(\mathbf{B}_{m+1})$ of $\mathbf{B}_{m+1}$. For all predictors to be treated symmetrically in projection, they should be standardized to have mean 0 and variance 1. Likewise, in cross-validation of k with mean square error prediction, it is a good idea to standardize all traits.

To update the precision matrix $\boldsymbol{\Gamma}$ for $\mathbf{B}$ fixed, we take advantage of the gradient

$$\nabla_{\boldsymbol{\Gamma}} \mathcal{L}(\mathbf{B}, \boldsymbol{\Gamma}) \;=\; \frac{n}{2}\boldsymbol{\Gamma}^{-1} - \frac{1}{2}(\mathbf{Y} - \mathbf{B}\mathbf{X})(\mathbf{Y} - \mathbf{B}\mathbf{X})^T \tag{4.4}$$

spelt out in the appendix. At a stationary point where $\nabla_{\boldsymbol{\Gamma}} \mathcal{L}(\mathbf{B}, \boldsymbol{\Gamma}) = \mathbf{0}_{r \times r}$, the optimal $\boldsymbol{\Gamma}$ is

$$\boldsymbol{\Gamma}_{m+1} = \left[\frac{1}{n}(\mathbf{Y} - \mathbf{B}_m \mathbf{X})(\mathbf{Y} - \mathbf{B}_m \mathbf{X})^T \right]^{-1}. \tag{4.5}$$

Equation (4.5) preserves the symmetry and positive semidefiniteness of $\boldsymbol{\Gamma}_m$. The required matrix inversion is straightforward unless the number of traits r is exceptionally large. Our experiments suggest solving for $\boldsymbol{\Gamma}_{m+1}$ exactly is superior to running full IHT jointly on both $\mathbf{B}$ and $\boldsymbol{\Gamma}$. Figure 2 displays our block ascent algorithm.

Algorithm 2: Block Ascent Multivariate Iterative Hard-Thresholding

Input : Genotypes $\mathbf{X}_{p \times n}$, traits $\mathbf{Y}_{r \times n}$, sparsity parameter k.
Initialize: $\mathbf{B}$ to univariate regression values, $\boldsymbol{\Gamma}_{r \times r}$ to identity matrix.
while not converged do
  Calculate gradient: $\mathbf{C} = \boldsymbol{\Gamma}(\mathbf{Y} - \mathbf{B}\mathbf{X})\mathbf{X}^T$
  Calculate step size: $t = \frac{\|\mathbf{C}\|_F^2}{\mathrm{tr}(\mathbf{X}^T\mathbf{C}^T\boldsymbol{\Gamma}\mathbf{C}\mathbf{X})}$
  Project to Sparsity: $\mathbf{B}_{new} = P_{S_k}(\mathbf{B} + t\mathbf{C})$
  Update: $\boldsymbol{\Gamma}_{new} = \left[\frac{1}{n}(\mathbf{Y} - \mathbf{B}_{new}\mathbf{X})(\mathbf{Y} - \mathbf{B}_{new}\mathbf{X})^T\right]^{-1}$
  while $L(\mathbf{B}, \boldsymbol{\Gamma}) > L(\mathbf{B}_{new}, \boldsymbol{\Gamma}_{new})$ **and backtrack steps** ≤ 3 **do**
    $t = t/2$
    $\mathbf{B}_{new} = P_{S_k}(\mathbf{B} + t\mathbf{C})$
    $\boldsymbol{\Gamma}_{new} = \left[\frac{1}{n}(\mathbf{Y} - \mathbf{B}_{new}\mathbf{X})(\mathbf{Y} - \mathbf{B}_{new}\mathbf{X})^T\right]^{-1}$
  end
end
Output: $\mathbf{B}$ with k non-zero entries and a symmetric and positive definite $\boldsymbol{\Gamma}$

4.2.2 Linear Algebra with Compressed Genotype Matrices

We previously described how to manipulate PLINK files using the `OpenMendel` module `SnpArrays.jl` [128], which supports linear algebra on compressed genotype matrices [28]. We now outline several

enhancements to our compressed linear algebra routines.

Compact genotype storage and fast reading. A binary PLINK genotype [93] stores each SNP genotype in two bits. Thus, an $n \times p$ genotype matrix requires $2np$ bits of memory. For bit-level storage `Julia` [12] supports the 8-bit unsigned integer type (`UInt8`) that can represent four sample genotypes simultaneously in a single 8-bit integer. Extracting sample genotypes can be achieved via bitshift and bitwise `and` operations. Genotypes are stored in little endian fashion, with 0, 1, 2, and missing genotypes mapped to the bit patterns 00, 01, 11, and 10, respectively. For instance, if a locus has four sample genotypes 1, 0, 2, and missing, then the corresponding `UInt8` integer is 10110001 in binary representation. Finally, because the genotype matrix is memory-mapped, opening a genotype file and accessing data is fast even for very large files.

SIMD-vectorized and tiled linear algebra. In IHT the most computationally intensive operations are the matrix-vector and matrix-matrix multiplications required in computing gradients. To speed up these operations, we employ SIMD (single instruction, multiple data) vectorization and tiling. On machines with SIMD support such as AVX (Advanced Vector Extensions), our linear algebra routine on compressed genotypes is usually twice as fast as BLAS 2 (Basic Linear Algebra Subroutines) [64] calls with an uncompressed numeric matrix and comparable in speed to BLAS 3 calls if $\mathbf{B}$ is tall and thin.

Computation of the matrix product $\mathbf{C} = \mathbf{AB}$ requires special care when $\mathbf{A}$ is the binary PLINK-formatted genotype matrix and $\mathbf{B}$ and $\mathbf{C}$ are numeric matrices. The idea is to partition these three matrices into small blocks and exploit the representation $\mathbf{C}_{ij} = \sum_k \mathbf{A}_{ik}\mathbf{B}_{kj}$ by computing each tiled product $\mathbf{A}_{ik}\mathbf{B}_{kj}$ in parallel. Because entries of a small matrix block are closer together in memory, this strategy improves cache efficiency. The triple `for` loops needed for computing products $\mathbf{A}_{ik}\mathbf{B}_{kj}$ are accelerated by invoking Julia's `LoopVectorization.jl` package, which performs automatic vectorization on machines with SIMD support. Furthermore, this routine can be parallelized because individual blocks can be multiplied and added independently. Because multi-threading in Julia is composable, these parallel operations can be safely nested inside other multi-threading Julia functions such as IHT's cross-validation routine.

4.2.3 Simulated Data Experiments

Our simulation studies are based on the chromosome 1 genotype data of the Northern Finland Birth Cohort (NFBC) [97]. The original NFBC1966 data contain 5402 samples and 364,590 SNPs; 26,906 of the SNPs reside on chromosome 1. After filtering for samples with at least 98% genotype success rate and SNPs with missing data less than 2%, we ended with 5340 samples and 24,523 SNPs on chromosome 1. For r traits, phenotypes are simulated according to the matrix normal distribution [32, 41, 118] as

$$\mathbf{Y}_{r\times n} \sim \text{MatrixNormal}(\mathbf{B}_{r\times p}\mathbf{X}_{p\times n},\ \boldsymbol{\Sigma}_{r\times r},\ \sigma_g^2\Phi_{n\times n}+\sigma_e^2\mathbf{I}_{n\times n})$$

using the `OpenMendel` module `TraitSimulation.jl` [52]. Here $\mathbf{X}$ is the chromosome 1 NFBC $p\times n$ genotype matrix with n samples aligned along its columns. The matrix $\mathbf{B}$ contains the true regression coefficients b_{ij} uniformly drawn from $\{0.05, 0.1, ..., 0.5\}$ and randomly set to 0 so that k_{true} entries b_{ij} survive. In standard mathematical notation, $\|\mathbf{B}\|_0 = k_{true}$. Note the effect-size set $\{0.05, 0.1, ..., 0.5\}$ is comparable to previous studies [28]. To capture pleiotropic effects, k_{plei} SNPs are randomly chosen to impact 2 traits. The remaining k_{indep} causal SNPs affect only one phenotype. Thus, $k_{true} = 2k_{plei} + k_{indep}$. Note it is possible for 2 traits to share 0 pleiotropic SNPs. The row (trait) covariance matrix $\boldsymbol{\Sigma}$ is simulated so that its maximum condition number does not exceed 10. The column (sample) covariance matrix equals $\sigma_g^2\Phi + \sigma_e^2\mathbf{I}$, where Φ is the centered genetic relationship matrix (GRM) estimated by GEMMA [131]. We let $\sigma_g^2 = 0.1$ and $\sigma_e^2 = 0.9$, so polygenic heritability is 10%. Different combinations of r, k_{true}, k_{indep}, and k_{plei} are summarized in Table 4.1. Each combination is replicated 100 times. It is worth emphasizing that this generative model should favor LMM analysis.

4.2.4 Method Comparisons

In our simulation experiments, we compared `MendelIHT.jl` to `mv-PLINK` [36] and `GEMMA` [131]. The linear mixed model software `GEMMA` enjoys wide usage in genetic epidemiology. The software

`mv-PLINK` is chosen for its speed. A recent review [43] rates `mv-PLINK` as the second fastest of the competing programs. The fastest method, MV-BIMBAM [101], is an older method published by the authors of `GEMMA`, so it is not featured in this study.

In simulated data experiments, all programs were run within 16 cores of an Intel Xeon Gold 6140 2.30GHz CPU with access to 32 GB of RAM. All experiments relied on version 1.4.2 of `MendelIHT` and Julia v1.5.4. IHT's sparsity level k is tuned by cross-validation. The number of cross-validation paths is an important determinant of both computation time and accuracy. Thus, for simulated data, we employed an initial grid search involving 5-fold cross validation over the sparsity levels $k \in \{5, 10, ..., 50\}$. This was followed by 5-fold cross-validation for $k \in \{k_{best} - 4, ..., k_{best} + 4\}$. This strategy first searches the space of potential values broadly, then zooms in on the most promising candidate sparsity level. `GEMMA` and `mv-PLINK` were run under their default settings. For both programs, we declared SNPs significant whose p-values were lower than $0.05/24523$. For `GEMMA`, we used the Wald test statistic.

4.2.5 Quality Control for UK Biobank

We ran `MendelIHT.jl` on a subset of the data from the second release of the UK Biobank [102], which contains $\sim 500,000$ samples and $\sim 800,000$ SNPs. Specifically, we jointly analyzed three potentially correlated traits: average systolic blood pressure (SBP), average diastolic blood pressure (DBP), and body mass index (BMI). These phenotypes were first log-transformed to minimize the impact of outliers. Then each phenotype was standardized to mean 0 and variance 1, so that the three traits were treated similarly in mean-squared error (MSE) cross-validation. Following [28], we first filtered samples exhibiting sex discordance, high heterozygosity, or high SNP missingness. We then excluded samples of non-European ancestry and first and second-degree relatives based on empirical kinship coefficients. We also excluded samples who were on hypertension medicine at baseline. Finally, we excluded samples with $< 98\%$ genotyping success rate and SNPs with $< 99\%$ genotyping success rate and imputed the remaining missing genotypes by the corresponding sample-mean genotypes. Note that imputation occurs in IHT on-the-fly. The final dataset contains $185,656$

samples and 470,228 SNPs. Given these reduced data and ignoring the Biobank's precomputed prinicipal components, we computed afresh the top 10 principal components via `FlashPCA2` [3] and included these as predictors to adjust for hidden ancestry. We also designated sex, age, and age^2 as non-genetic predictors.

4.3 Results

4.3.1 Simulation Experiments

Table 4.1 summarizes the various experiments conducted on the simulated data. For IHT cross-validation times are included. Multivariate IHT is the fastest method across the board, and is the only method that can analyze more than 50 traits. Multivariate IHT's runtime increases roughly linearly with the number of traits. All methods perform similarly in recovering the pleiotropic and independent SNPs. Univariate IHT exhibit slightly worse true positive rate compared to multivariate methods. Given the identically distributed effect sizes in our simulations, all methods are better at finding pleiotropic SNPs than independent SNPs.

Notably, the false positive rates for both univariate and multivariate IHT are much lower than competing methods, often by more than an order of magnitude. Presumably, many of the false positives from mvLMM and CCA represent SNPs in significant LD with the causal SNP. IHT is better at distilling the true signal within these LD blocks because it considers the effect of all SNPs jointly. `GEMMA`'s mvLMM is better at controlling false positives than `mv-PLINK`'s CCA, but model fitting for mvLMM is slower, especially for large numbers of traits. In summary, IHT offers better model selection than these competitors with better computational speed.

4.3.2 UK Biobank Analysis

The UK Biobank analysis completed in 20 hours and 8 minutes on 36 cores of an Intel Xeon Gold 6140 2.30GHz CPU with access to 180 GB of RAM. As described in methods section, the fea-

	Time (sec)	Plei TP	Indep TP	FP
Set 1: (2 traits, $k_{true} = 10$, $k_{indep} = 4$, $k_{plei} = 3$)				
mIHT	164.6 ± 69.3	0.92 ± 0.16	0.76 ± 0.2	3.7 ± 6.4
uIHT	114.9 ± 48.6	0.93 ± 0.16	0.72 ± 0.2	1.4 ± 3.7
CCA	152.6 ± 57.3	0.96 ± 0.14	0.78 ± 0.2	77.8 ± 40.6
mvLMM	307.7 ± 121.4	0.95 ± 0.15	0.76 ± 0.2	42.8 ± 18.5
Set 2: (3 traits, $k_{true} = 20$, $k_{indep} = 10$, $k_{plei} = 5$)				
mIHT	214.4 ± 100.1	0.91 ± 0.12	0.75 ± 0.14	5.7 ± 6.0
uIHT	169.6 ± 81.9	0.86 ± 0.16	0.72 ± 0.16	2.4 ± 2.5
CCA	226.8 ± 101.9	0.95 ± 0.09	0.79 ± 0.15	125.3 ± 55.3
mvLMM	449.9 ± 221.7	0.93 ± 0.1	0.75 ± 0.16	66.1 ± 22.8
Set 3: (5 traits, $k_{true} = 30$, $k_{indep} = 16$, $k_{plei} = 7$)				
mIHT	227.9 ± 41.1	0.93 ± 0.09	0.73 ± 0.12	5.9 ± 4.6
uIHT	213.8 ± 45.7	0.9 ± 0.11	0.69 ± 0.12	3.2 ± 3.8
CCA	371.5 ± 34.0	0.96 ± 0.07	0.75 ± 0.11	173.2 ± 54.9
mvLMM	1135.3 ± 125.5	0.94 ± 0.09	0.71 ± 0.11	93.6 ± 22.7
Set 4: (10 traits, $k_{true} = 10$, $k_{indep} = 4$, $k_{plei} = 3$)				
mIHT	278.8 ± 53.0	0.97 ± 0.09	0.74 ± 0.2	2.2 ± 2.0
uIHT	245.8 ± 34.8	0.96 ± 0.11	0.7 ± 0.23	3.1 ± 6.4
CCA	985.1 ± 97.5	0.99 ± 0.06	0.78 ± 0.2	64.6 ± 30.5
mvLMM	8067.4 ± 3900.8	0.99 ± 0.06	0.74 ± 0.18	41.8 ± 16.4
Set 5: (50 traits, $k_{true} = 20$, $k_{indep} = 10$, $k_{plei} = 5$)				
mIHT	1892.2 ± 419.0	0.93 ± 0.12	0.75 ± 0.14	2.9 ± 2.5
uIHT	1336.9 ± 310.2	0.92 ± 0.11	0.72 ± 0.12	7.6 ± 5.9
CCA	26589.1 ± 907.7(*)	NA	NA	NA
mvLMM	NA	NA	NA	NA
Set 6: (100 traits, $k_{true} = 30$, $k_{indep} = 16$, $k_{plei} = 7$)				
mIHT	3699.3 ± 410.4	0.91 ± 0.11	0.71 ± 0.11	2.8 ± 2.1
uIHT	2353.8 ± 212.3	0.92 ± 0.11	0.7 ± 0.10	10.7 ± 4.3
CCA	NA	NA	NA	NA
mvLMM	NA	NA	NA	NA

Table 4.1: Comparison of multivariate IHT (mIHT) and multiple univariate IHT (uIHT) implemented in MendelIHT, canonical correlation analysis (CAA) implemented in mv-PLINK, and multivariate linear mixed models (mvLMM) implemented in GEMMA on chromosome 1 of the NFBC1966 data with simulated traits. Plei TP is the proportion of true positives for pleiotopic SNPs, Indep TP is proportion of true positives for independent SNPs, and FP is the total number of false positive. $\pm$ indicates standard deviations. k_{true} is the total number of non-zero entries in $\mathbf{B}$, k_{indep} is the number of independent SNPs affecting only one trait, and k_{plei} is the number of pleiotropic SNPs affecting two traits. These numbers satisfy $k_{true} = 2 \times k_{plei} + k_{indep}$. Each simulation relied on 100 replicates. NA: not available due to excessive run time. (*) Only two replicates contribute to timing.

tured phenotypes are body mass index (BMI), average systolic blood pressure (SBP), and average diastolic blood pressure (DBP). A first pass with 3-fold cross-validation across model sizes $k \in \{100, 200, ..., 1000\}$ showed that $k = 200$ minimizes the MSE (mean squared error). A second pass with 3 fold cross-validation across model sizes $k \in \{110, 120, ..., 290\}$ showed that $k = 190$ minimizes the MSE. A third 3-fold cross-validation pass across $k \in \{191, 192, ..., 209\}$ identified $k = 197$ as the best sparsity level. Given $k = 197$, we ran multivariate IHT on the full data to estimate effect sizes, correlation among traits, and proportion of phenotypic variance explained by the genotypes.

IHT selected 13 pleiotropic SNPs and 171 independent SNPs. Selected SNPs and non-genetic predictors appear in the supplement as Tables 4.2-4.6. To compare against previous studies, we used the `R` package `gwasrapidd` [76] to search the NHGRI-EBI GWAS catalog [74] for previously associated SNPS within 1 Mb of each IHT discovered SNP. After matching, all 13 pleiotropic SNPs and 158 independent SNPs are either previously associated or are within 1Mb of a previously associated SNP. We discovered 3 new associations with SBP and 10 new associations associated with DBP. Seven SNPs, rs2307111, rs6902725, rs11977526, rs2071518, rs11222084, rs365990, and rs77870048, are associated with two traits in opposite directions.

One can estimate the genotypic variance explained by the sparse model as $Var(\hat{\beta}_i \mathbf{X})/Var(\mathbf{y}_i)$ for each trait $\mathbf{y}_i$ where $\hat{\beta}_i \in \mathbb{R}^{1\times p}$ is the ith row of $\mathbf{B}$. `MendelIHT.jl` output the values $\sigma^2_{BMI} = 0.033$, $\sigma^2_{SBP} = 0.143$, and $\sigma^2_{DBP} = 0.048$. Note these estimates do not include contributions from the intercept or non-genetic predictors. The estimated correlations among traits are $r_{BMI,SBP} = 0.197$, $r_{BMI,DBP} = 0.286$, and $r_{SBP,DBP} = 0.738$. As expected, all traits are positively correlated, with a strong correlation between SBP and DBP and a weak correlation between BMI and both SBP and DBP.

4.4 Discussion

This paper presents multivariate IHT for analyzing multiple correlated phenotypes. In simulation studies, multivariate IHT exhibits similar true positive rates, significantly lower false positive rates, and better overall speed than linear mixed models and canonical correlation analysis. Computational time for multivariate IHT increases roughly linearly with the number of traits. Since IHT is a penalized regression method, the estimated effect size for each SNP is explicitly conditioned on other SNPs and non-genetic predictors. Analyzing three correlated UK Biobank traits with $\sim 200,000$ samples and $\sim 500,000$ SNPs took 20 hours on a single machine. IHT can output the correlation matrix and proportion of variance explained for component traits. `MendelIHT.jl` also automatically handles various input formats (binary PLINK, BGEN, and VCF files) by calling the relevant `OpenMendel` packages. If binary PLINK files are used, `MendelIHT.jl` avoids decompressing genotypes to full numeric matrices.

IHT's statistical and computational advantages come with limitations. For instance, it ignores hidden and explicit relatedness. IHT can exploit principal components to adjust for ancestry, but PCA alone is insufficient to account for small-scale family structure [91]. To overcome this limitation, close relatives can be excluded from a study. Although cross-validation may implicitly adjust for family relatedness, a more comprehensive analysis is required before we can recommend including related samples. Although our simulation studies suggest the contrary, there is also the possibility that strong linkage disequilibrium may confuse IHT. Finally, it is unclear how IHT will respond to wrongly imputed markers and the rare variants generated by sequencing. In spite of these qualms, the evidence presented here is very persuasive about IHT's potential for multivariate GWAS.

We will continue to explore improvements to IHT. Extension to non-Gaussian traits is hindered by the lack of flexible multivariate distributions with non-Gaussian margins. Cross-validation remains computationally intensive in tuning the sparsity level k. Although our vectorized linear algebra routine partially overcomes many of the computational barriers, we feel that further gains are possible through GPU computing [58, 57, 59, 125] and trading cross-validation for knockoff strate-

gies in model selection [8, 99]. Given IHT's advantages, we recommend it for general use with the understanding that epidemiologists respect its limitations and complement its application with standard univariate statistical analysis.

SNP	Chrom	BMI	SBP	DBP	# prior report
rs1801131	1	0.0	0.009	0.008	41
rs17367504	1	0.0	0.02	0.019	41
rs16998073	4	0.0	-0.023	-0.023	24
rs1173727	5	0.0	0.016	0.017	19
rs2307111	5	0.013	0.0	-0.01	34
rs6902725	6	0.0	-0.009	0.01	4
rs11977526	7	0.0	0.013	-0.011	6
rs2071518	8	0.0	-0.008	0.009	6
rs11222084	11	0.0	-0.011	0.009	9
rs3184504	12	0.0	-0.011	-0.018	41
rs365990	14	0.0	0.008	-0.011	6
rs7497304	15	0.0	-0.019	-0.018	14
rs77870048	16	0.0	-0.011	0.01	20

Table 4.2: 13 pleiotropic SNPs selected by IHT listed with their effect sizes. An effect size of 0 means the particular predictor was not selected. The field *prior reports* records the number of SNPs previously associated with BMI, SBP, or DBP (p value $< 10^{-8}$) that are within 1Mb of the given SNP. BMI = body mass index; SBP = systolic blood pressure; DBP = diastolic blood pressure.

4.5 Appendix

4.5.1 Significant SNPs from the UKB analysis

Tables 4.2-4.6 list the SNPs discovered by our UK Biobank analysis. The reported effect sizes correspond to the predictors of the log-transformed and standardized phenotypes. To compare against previous studies, we searched the NHGRI-EBI GWAS catalog [74] using the `R` package `gwasrapidd` [76]. For each SNP discovered by IHT, we queried a 1Mb radius for other SNPs that have been previously associated with the given trait with p value $< 10^{-8}$. Each known association is defined as the most significant SNP in a locus identified to be associated with the trait. All 13 pleiotropic SNPs were previously known and 158 out of 171 independent SNPs were previously known. Among the 13 newly discovered associations, 3 were with SBP and 10 were with DBP.

covariate	BMI	SBP	DBP
intercept	0.0002	-0.0	0.0003
sex	0.1165	0.1639	0.1808
age	0.1073	-0.1395	0.6484
age^2	-0.0505	0.4581	-0.6131
PC1	-0.0287	-0.0066	-0.0074
PC2	0.0052	-0.0077	-0.0011
PC3	0.0174	0.0043	-0.0091
PC4	-0.0019	-0.0055	0.0015
PC5	0.0022	0.0137	0.0167
PC6	-0.0165	-0.0067	-0.0056
PC7	-0.003	-0.0038	-0.0078
PC8	-0.0014	-0.0015	-0.0014
PC9	0.0039	-0.0004	0.0017
PC10	0.0068	0.0005	0.0001

Table 4.3: Non-genetic covariates estimated by IHT listed with their effect sizes. PC is short for principal component. BMI = body mass index, SBP = systolic blood pressure, and DBP = diastolic blood pressure.

4.5.2 Loglikelihood

Consider multivariate linear regression with r traits under a Gaussian model. Up to a constant, the loglikelihood for the response vector $\mathbf{y}_i$ of subject i can be written

$$\begin{aligned} \mathcal{L}_i \begin{pmatrix} \mathbf{B} \\ \mathbf{\Gamma} \end{pmatrix} &= \frac{1}{2}\log(\det \mathbf{\Gamma}) - \frac{1}{2}(\mathbf{y}_i - \mathbf{B}\mathbf{x}_i)^T \mathbf{\Gamma}(\mathbf{y}_i - \mathbf{B}\mathbf{x}_i) \\ &= \frac{1}{2}\log(\det \mathbf{\Gamma}) - \frac{1}{2}\operatorname{tr}[\mathbf{\Gamma}(\mathbf{y}_i - \mathbf{B}\mathbf{x}_i)(\mathbf{y}_i - \mathbf{B}\mathbf{x}_i)^T], \end{aligned}$$

where $\mathbf{B}$ is the $r \times p$ matrix of regression coefficients, $\mathbf{x}_i$ is the $p \times 1$ vector of predictors, and $\mathbf{\Gamma}$ is the $r \times r$ unstructured precision (inverse covariance) matrix. For n independent samples, let $\mathbf{Y}$ be the $r \times n$ matrix with ith column $\mathbf{y}_i$ and let $\mathbf{X}$ be the $p \times n$ design matrix with ith column $\mathbf{x}_i$. Then the

SNP	Chrom	β	# Prior reports	SNP	Chrom	β	# Prior reports
rs2815757	1	0.019	14	rs17207196	7	0.016	9
rs543874	1	-0.028	22	rs925946	11	-0.007	24
rs2820312	1	-0.014	9	rs6265	11	0.012	24
rs62106258	2	0.027	51	rs2049045	11	0.007	24
rs11127485	2	0.017	51	rs10835211	11	-0.011	24
rs13393304	2	0.008	51	rs7138803	12	-0.007	9
rs713586	2	-0.026	26	rs7132908	12	-0.014	9
rs9821675	3	0.007	20	rs4776970	15	0.01	17
rs1062633	3	0.012	20	rs16951304	15	0.008	17
rs957919	3	-0.015	17	rs2531995	16	0.016	17
rs61587156	3	0.017	9	rs72793809	16	-0.015	17
rs34811474	4	0.015	3	rs4788190	16	0.019	16
rs10938397	4	-0.022	17	rs4889490	16	0.014	20
rs13107325	4	-0.026	7	rs1421085	16	-0.05	56
rs2112347	5	0.009	22	rs17782313	18	-0.011	47
rs1422192	5	-0.015	20	rs10871777	18	-0.015	47
rs2744962	6	-0.017	33	rs17773430	18	-0.014	45
rs987237	6	-0.012	37	rs1800437	19	0.017	20
rs9473932	6	-0.009	37	rs3810291	19	0.022	12

Table 4.4: 38 SNPs associated with BMI independently of SBP and DBP listed with their effect sizes. The field *prior reports* records the number of SNPs previously associated with BMI (p value $< 10^{-8}$) that are within 1Mb of the given SNP.

SNP	Chrom	β	# Prior reports	SNP	Chrom	β	# Prior reports
rs3936009	1	0.009	7	rs12673516	7	0.01	3
rs1757915	1	-0.009	4	rs2282978	7	0.016	4
rs6684353	1	0.011	2	rs2392929	7	-0.027	5
rs12069946	1	-0.009	2	rs2978456	8	-0.01	1
rs2820441	1	0.009	1	Affx-32837790	8	-0.009	3
rs1522484	2	0.016	3	rs35758124	8	-0.01	2
rs9306894	2	0.009	4	rs10757278	9	-0.01	3
rs55654088	2	-0.01	4	rs10986626	9	-0.009	5
rs13002573	2	0.012	13	rs12258967	10	0.009	10
rs560887	2	0.011	1	rs1908339	10	0.01	4
rs10497529	2	0.011	3	rs11191064	10	0.009	6
rs1052501	3	0.02	6	rs11598702	10	-0.011	17
rs2498323	4	-0.008	3	rs4980389	11	-0.009	14
rs776590	4	0.008	2	rs573455	11	-0.014	6
rs17084051	4	-0.01	3	rs10750441	11	0.01	3
rs1229984	4	0.009	1	rs10770612	12	0.012	9
rs6842241	4	-0.008	2	rs2681492	12	0.011	21
rs4690974	4	-0.01	9	rs12882307	14	0.009	1
rs13116200	4	-0.01	2	rs4903064	14	-0.01	5
rs7715779	5	0.007	12	rs956006	15	0.009	1
rs1173771	5	0.005	12	rs34862454	15	-0.012	14
rs1982192	5	0.009	Novel	rs3803716	16	0.008	3
rs2303720	5	0.011	6	rs4888372	16	0.01	6
rs11954193	5	0.01	13	rs60675007	16	0.008	2
rs12198986	6	-0.008	1	rs9889363	17	-0.011	6
rs9349379	6	0.012	5	rs185478092	17	-0.01	Novel
rs385306	6	0.011	7	rs3744760	17	-0.01	9
rs12191865	6	-0.014	3	rs17608766	17	-0.013	3
rs1012257	6	-0.009	3	rs9909933	17	0.009	7
rs9689048	6	0.009	Novel	rs35688424	17	-0.011	8
rs2221389	6	-0.012	1	rs67882421	18	-0.012	4
rs9505897	6	-0.01	1	rs12459507	19	-0.009	6
rs57301765	7	-0.014	2	rs34328549	19	0.007	8

Table 4.5: 66 SNPs associated with SBP independently of BMI and DBP listed with their effect sizes. The field *prior reports* records the number of GWAS Catalog associations with SBP (p value $< 10^{-8}$) that are within 1Mb of the given SNP. A novel SNP is not within 1Mb of any GWAS Catalog associations.

SNP	Chrom	β	# Prior reports	SNP	Chrom	β	# Prior reports
rs61776719	1	-0.015	Novel	rs4754834	11	-0.009	1
rs12739904	1	-0.009	1	rs59317921	11	0.009	2
rs3766090	1	-0.01	Novel	rs7975252	12	-0.009	1
rs61822997	1	-0.009	Novel	rs7973748	12	0.011	10
rs665834	1	0.011	1	rs12581906	12	0.009	10
rs2275155	1	-0.013	6	rs17287293	12	0.01	Novel
rs11690961	2	-0.012	1	rs74340001	12	-0.01	1
rs10199082	2	-0.01	3	rs653178	12	-0.007	16
rs2692893	2	-0.012	5	rs12875271	13	0.011	3
rs17362588	2	-0.014	3	rs36033161	14	-0.009	Novel
rs12996836	2	-0.011	3	rs686861	15	-0.013	1
rs1863703	2	0.009	3	rs11853359	15	0.011	2
rs2624847	3	-0.009	3	rs1378942	15	-0.012	10
rs3617	3	0.008	5	rs4886615	15	-0.008	10
rs9850919	3	0.009	5	rs2277547	15	0.01	1
rs871606	4	-0.018	1	rs7174546	15	-0.009	2
rs1826909	4	-0.01	Novel	rs67456613	16	-0.01	2
rs1047440	5	-0.014	3	rs72790195	16	-0.008	1
rs11949055	5	0.009	5	rs11078485	17	-0.009	2
rs1233708	6	0.013	2	rs72824497	17	0.009	1
rs11154022	6	0.01	3	rs768168	19	-0.011	Novel
rs12110693	6	-0.012	3	rs997669	19	-0.01	2
rs9376740	6	-0.009	Novel	rs755690	19	-0.01	1
rs58023137	6	-0.011	1	rs2876201	20	-0.009	6
rs194524	7	-0.012	Novel	rs652661	20	0.015	6
rs13226502	7	-0.014	2	rs78309244	20	0.014	6
rs2469997	8	0.01	3	rs6046144	20	0.015	2
rs507666	9	0.012	2	rs76701589	20	0.009	3
rs3812595	9	-0.009	1	rs2831969	21	0.009	Novel
rs183348357	10	-0.009	7	rs9982601	21	-0.009	1
rs3739998	10	0.005	1	rs2298336	21	0.01	1
rs2505083	10	0.007	1	rs112005532	21	0.012	1
rs7125196	11	0.01	5	rs73167017	22	0.011	1
rs2304500	11	-0.012	1				

Table 4.6: 67 SNPs associated with DBP independently of BMI and SBP listed with their effect sizes. The field *prior reports* records the number of GWAS Catalog associations with DBP (p value $< 10^{-8}$) that are within 1Mb of the given SNP. A novel SNP is not within 1Mb of any GWAS Catalog associations.

loglikelihood for all samples is

$$\begin{aligned}
\mathcal{L}\begin{pmatrix}\mathbf{B}\\ \mathbf{\Gamma}\end{pmatrix} &= \sum_{i=1}^{n}\left\{\frac{1}{2}\log(\det\mathbf{\Gamma}) - \frac{1}{2}\operatorname{tr}[\mathbf{\Gamma}(\mathbf{y}_i-\mathbf{B}\mathbf{x}_i)(\mathbf{y}_i-\mathbf{B}\mathbf{x}_i)^T]\right\}\\
&= \frac{n}{2}\log(\det\mathbf{\Gamma}) - \frac{1}{2}\operatorname{tr}\left[\mathbf{\Gamma}\sum_{i=1}^{n}(\mathbf{y}_i-\mathbf{B}\mathbf{x}_i)(\mathbf{y}_i-\mathbf{B}\mathbf{x}_i)^T\right]\\
&= \frac{n}{2}\log(\det\mathbf{\Gamma}) - \frac{1}{2}\operatorname{tr}[(\mathbf{\Gamma}(\mathbf{Y}-\mathbf{B}\mathbf{X})(\mathbf{Y}-\mathbf{B}\mathbf{X})^T].
\end{aligned}$$

In subsequent sections we will present both full and block ascent IHT. The former updates $\mathbf{B}$ and $\mathbf{\Gamma}$ simultaneously. The latter alternates updates of $\mathbf{B}$ and $\mathbf{\Gamma}$, holding the other parameter block fixed.

4.5.3 First Directional Derivative

Recall that the Hadamard's semi-directional derivative [34, 61] of a function $f(\mathbf{x})$ in the direction $\mathbf{v}$ is defined as the limit

$$d_{\mathbf{v}}f(\mathbf{x}) = \lim_{\substack{h\to 0\\ \mathbf{w}\to\mathbf{v}}}\frac{f(\mathbf{x}+h\mathbf{w})-f(\mathbf{x})}{h}.$$

To calculate the directional derivative of the loglikelihood (4.1), we perturb $\mathbf{B}$ in the direction $\mathbf{U}$ and $\mathbf{\Gamma}$ in the symmetric direction $\mathbf{V}$. The sum and product rules then give

$$\begin{aligned}
& d_{\binom{\mathbf{U}}{\mathbf{V}}}\operatorname{tr}[\mathbf{\Gamma}(\mathbf{Y}-\mathbf{B}\mathbf{X})(\mathbf{Y}-\mathbf{B}\mathbf{X})^T]\\
= \;& \operatorname{tr}[\mathbf{V}(\mathbf{Y}-\mathbf{B}\mathbf{X})(\mathbf{Y}-\mathbf{B}\mathbf{X})^T] - \operatorname{tr}[\mathbf{\Gamma}(\mathbf{Y}-\mathbf{B}\mathbf{X})\mathbf{X}^T\mathbf{U}^T - \mathbf{\Gamma}\mathbf{U}\mathbf{X}(\mathbf{Y}-\mathbf{B}\mathbf{X})^T].
\end{aligned}$$

The directional derivative $d_{\mathbf{V}} \ln\det(\boldsymbol{\Gamma}) = \mathrm{tr}(\boldsymbol{\Gamma}^{-1}\mathbf{V})$ is derived in Example 3.2.6 of [61]. The trace properties $\mathrm{tr}(\mathbf{CD}) = \mathrm{tr}(\mathbf{DC})$ and $\mathrm{tr}(\mathbf{C}^T) = \mathrm{tr}(\mathbf{C})$ consequently imply

$$d_{\binom{\mathbf{U}}{\mathbf{V}}} \mathcal{L}(\mathbf{B},\boldsymbol{\Gamma}) = \frac{n}{2}\mathrm{tr}(\boldsymbol{\Gamma}^{-1}\mathbf{V}) - \frac{1}{2}\mathrm{tr}[(\mathbf{Y}-\mathbf{BX})(\mathbf{Y}-\mathbf{BX})^T\mathbf{V}] + \mathrm{tr}[\mathbf{X}(\mathbf{Y}-\mathbf{BX})^T\boldsymbol{\Gamma}\mathbf{U}]. \quad (4.6)$$

Because this last expression is linear in $(\mathbf{U},\mathbf{V})$, the loglikelihood is continuously differentiable.

4.5.4 Second Directional Derivative

Now we take the directional derivative of the directional derivative (4.6) in the new directions $\tilde{\mathbf{U}}$ and $\tilde{\mathbf{V}}$. This action requires the inverse rule $d_{\tilde{\mathbf{V}}}\boldsymbol{\Gamma}^{-1} = -\boldsymbol{\Gamma}^{-1}\tilde{\mathbf{V}}\boldsymbol{\Gamma}^{-1}$ proved in Example 3.2.7 of [61]. Accordingly, we find

$$d_{\binom{\tilde{\mathbf{U}}}{\tilde{\mathbf{V}}}} \frac{n}{2}\mathrm{tr}(\boldsymbol{\Gamma}^{-1}\mathbf{V}) = -\frac{n}{2}\mathrm{tr}\left(\boldsymbol{\Gamma}^{-1}\tilde{\mathbf{V}}\boldsymbol{\Gamma}^{-1}\mathbf{V}\right).$$

We also calculate

$$d_{\binom{\tilde{\mathbf{U}}}{\tilde{\mathbf{V}}}} -\frac{1}{2}\mathrm{tr}[(\mathbf{Y}-\mathbf{BX})(\mathbf{Y}-\mathbf{BX})^T\mathbf{V}] = \frac{1}{2}\mathrm{tr}[(\mathbf{Y}-\mathbf{BX})\mathbf{X}^T\tilde{\mathbf{U}}^T\mathbf{V}] + \frac{1}{2}\mathrm{tr}[\tilde{\mathbf{U}}\mathbf{X}(\mathbf{Y}-\mathbf{BX})^T\mathbf{V}]$$

and

$$d_{\binom{\tilde{\mathbf{U}}}{\tilde{\mathbf{V}}}} \mathrm{tr}[\mathbf{X}(\mathbf{Y}-\mathbf{BX})^T\boldsymbol{\Gamma}\mathbf{U}] = \mathrm{tr}[\mathbf{X}(\mathbf{Y}-\mathbf{BX})^T\tilde{\mathbf{V}}\mathbf{U}] - \mathrm{tr}(\mathbf{X}^T\tilde{\mathbf{U}}^T\boldsymbol{\Gamma}\mathbf{U}\mathbf{X}).$$

Finally, setting the two directions equal so that $\tilde{\mathbf{V}} = \mathbf{V}$ and $\tilde{\mathbf{U}} = \mathbf{U}$ produces the quadratic form

$$\begin{aligned} Q(\mathbf{U},\mathbf{V}) = {} & -\frac{n}{2}\mathrm{tr}[\boldsymbol{\Gamma}^{-1}\mathbf{V}\boldsymbol{\Gamma}^{-1}\mathbf{V}] + \frac{1}{2}\mathrm{tr}[(\mathbf{Y}-\mathbf{BX})\mathbf{X}^T\mathbf{U}^T\mathbf{V}] + \frac{1}{2}\mathrm{tr}[\mathbf{U}\mathbf{X}(\mathbf{Y}-\mathbf{BX})^T\mathbf{V}] \\ & + \mathrm{tr}[\mathbf{X}(\mathbf{Y}-\mathbf{BX})^T\mathbf{V}\mathbf{U}] - \mathrm{tr}(\mathbf{X}^T\mathbf{U}^T\boldsymbol{\Gamma}\mathbf{U}\mathbf{X}). \end{aligned} \quad (4.7)$$

generated by the second differential.

4.5.5 Extraction of the Gradient and Expected Information

To extract the gradient from a directional derivative, we recall the identity $d_{\mathbf{v}} f(\mathbf{x}) = \nabla f(\mathbf{x})^T \mathbf{v}$ for vectors $\mathbf{v}$ and $\mathbf{x}$ and the identity $\mathrm{tr}(\mathbf{A}^T\mathbf{B}) = vec(\mathbf{A})^T vec(\mathbf{B})$ for matrices $\mathbf{A}$ and $\mathbf{B}$ [77]. The first identity shows that the directional derivative is the inner product of the gradient with respect to the direction $\mathbf{v}$. The second displays the trace function as an inner product on dimensionally identical matrices. Thus, the matrix directional derivative is

$$d_{\mathbf{V}} f(\mathbf{X}) \;=\; vec[\nabla f(\mathbf{X})]^T vec(\mathbf{V}) \;=\; \mathrm{tr}[\nabla f(\mathbf{X})^T \mathbf{V}]. \tag{4.8}$$

Inspection of the directional derivative (4.6) now leads to the gradient with blocks

$$\nabla_{\mathbf{B}} \mathcal{L} \begin{pmatrix} \mathbf{B} \\ \boldsymbol{\Gamma} \end{pmatrix} \;=\; [\mathbf{X}(\mathbf{Y} - \mathbf{B}\mathbf{X})^T \boldsymbol{\Gamma}]^T \;=\; \boldsymbol{\Gamma}(\mathbf{Y} - \mathbf{B}\mathbf{X})\mathbf{X}^T \tag{4.9}$$

$$\nabla_{\boldsymbol{\Gamma}} \mathcal{L} \begin{pmatrix} \mathbf{B} \\ \boldsymbol{\Gamma} \end{pmatrix} \;=\; \frac{n}{2}\boldsymbol{\Gamma}^{-1} - \frac{1}{2}(\mathbf{Y} - \mathbf{B}\mathbf{X})(\mathbf{Y} - \mathbf{B}\mathbf{X})^T. \tag{4.10}$$

Analogously, the quadratic form (4.7) implicitly defines the Hessian $\mathbf{H}$ through the identity

$$\begin{aligned} Q(\mathbf{U},\mathbf{V}) \;&=\; \mathrm{tr}\left\{ \begin{bmatrix} vec(\mathbf{U})^T & vec(\mathbf{V})^T \end{bmatrix} \begin{pmatrix} \mathbf{H}_{\mathbf{B}\mathbf{B}} & \mathbf{H}_{\mathbf{B}\boldsymbol{\Gamma}} \\ \mathbf{H}_{\boldsymbol{\Gamma}\mathbf{B}} & \mathbf{H}_{\boldsymbol{\Gamma}\boldsymbol{\Gamma}} \end{pmatrix} \begin{bmatrix} vec(\mathbf{U}) \\ vec(\mathbf{V}) \end{bmatrix} \right\} \\ &\equiv\; -\frac{n}{2}\mathrm{tr}\left(\boldsymbol{\Gamma}^{-1}\mathbf{V}\boldsymbol{\Gamma}^{-1}\mathbf{V}\right) + \frac{1}{2}\mathrm{tr}\left((\mathbf{Y} - \mathbf{B}\mathbf{X})\mathbf{X}^T\mathbf{U}^T\mathbf{V}\right) \\ &\quad + \frac{1}{2}\mathrm{tr}\left(\mathbf{U}\mathbf{X}(\mathbf{Y} - \mathbf{B}\mathbf{X})^T\mathbf{V}\right) + \mathrm{tr}\left(\mathbf{X}(\mathbf{Y} - \mathbf{B}\mathbf{X})^T\mathbf{V}\mathbf{U}\right) - \mathrm{tr}(\mathbf{X}^T\mathbf{U}^T\boldsymbol{\Gamma}\mathbf{U}\mathbf{X}). \end{aligned}$$

Because $E(\mathbf{Y}) = \mathbf{B}\mathbf{X}$, the expected information $\mathbf{J} = E(-\mathbf{H})$ has the off-diagonal blocks $\mathbf{J}_{\mathbf{B},\boldsymbol{\Gamma}} = \mathbf{0}_{pr \times r^2}$

and $\mathbf{J}_{\mathbf{\Gamma},\mathbf{B}} = \mathbf{0}_{r^2 \times pr}$. Now the Kronecker product identity $vec(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})vec(\mathbf{B})$ implies

$$\begin{aligned}
\mathrm{tr}(\mathbf{X}^T\mathbf{U}^T\mathbf{\Gamma}\mathbf{U}\mathbf{X}) &= \mathrm{tr}(\mathbf{X}\mathbf{X}^T\mathbf{U}^T\mathbf{\Gamma}\mathbf{U}) \\
&= \mathrm{tr}[(\mathbf{\Gamma}\mathbf{U}\mathbf{X}\mathbf{X}^T)^T\mathbf{U}] \\
&= vec(\mathbf{\Gamma}\mathbf{U}\mathbf{X}\mathbf{X}^T)^T vec(\mathbf{U}) \\
&= \left[(\mathbf{X}\mathbf{X}^T \otimes \mathbf{\Gamma})vec(\mathbf{U})\right]^T vec(\mathbf{U}) \\
&= vec(\mathbf{U})^T(\mathbf{X}\mathbf{X}^T \otimes \mathbf{\Gamma})vec(\mathbf{U}).
\end{aligned}$$

It follows that $\mathbf{J}_{\mathbf{BB}} = \mathbf{X}\mathbf{X}^T \otimes \mathbf{\Gamma}$. Similarly,

$$\begin{aligned}
\mathrm{tr}(\mathbf{\Gamma}^{-1}\mathbf{V}\mathbf{\Gamma}^{-1}\mathbf{V}) &= \left[(\mathbf{\Gamma}^{-1} \otimes \mathbf{\Gamma}^{-1})vec(\mathbf{V})\right]^T vec(\mathbf{V}) \\
&= vec(\mathbf{V})^T[\mathbf{\Gamma}^{-1} \otimes \mathbf{\Gamma}^{-1}]vec(\mathbf{V}),
\end{aligned}$$

so that $\mathbf{J}_{\mathbf{\Gamma\Gamma}} = \mathbf{\Gamma}^{-1} \otimes \mathbf{\Gamma}^{-1}$. In summary, the expected information matrix takes the block diagonal form

$$\mathbf{J} = \begin{pmatrix} (\mathbf{X}\mathbf{X}^T) \otimes \mathbf{\Gamma} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Gamma}^{-1} \otimes \mathbf{\Gamma}^{-1} \end{pmatrix}. \tag{4.11}$$

In our projected steepest ascent algorithm, the expected information matrix is never explicitly formed. It is implicitly accessed in the step-size calculation through the associated quadratic form $Q(\mathbf{B}, \mathbf{\Gamma})$.

4.5.6 Full IHT Step Size

The next iterate in full IHT is the projection of the point

$$\mathbf{\Delta}_{m+1} = \begin{pmatrix} \mathbf{B}_m \\ \mathbf{\Gamma}_m \end{pmatrix} + t_m \nabla\mathcal{L} \begin{pmatrix} \mathbf{B}_m \\ \mathbf{\Gamma}_m \end{pmatrix} = \begin{pmatrix} \mathbf{B}_m \\ \mathbf{\Gamma}_m \end{pmatrix} + t_m \begin{pmatrix} \mathbf{C}_m \\ \mathbf{W}_m \end{pmatrix}, \tag{4.12}$$

where $\mathbf{C}_m = \nabla_{\mathbf{B}}\mathcal{L}$ and $\mathbf{W}_m = \nabla_{\mathbf{\Gamma}}\mathcal{L}$ evaluated at $(\mathbf{B}_m, \mathbf{\Gamma}_m)$. The loglikelihood along the ascent direction is a function of the scalar t_m and can be approximated by the second-order expansion

$$\mathcal{L}(\mathbf{\Delta}_{m+1}) \approx \mathcal{L}\begin{pmatrix}\mathbf{B}_m \\ \mathbf{\Gamma}_m\end{pmatrix} + t_m \operatorname{tr}\left[\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}^T \begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}\right] - \frac{t_m^2}{2}\operatorname{tr}\left[\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}^T \mathbf{J}\begin{pmatrix}\mathbf{B}_m \\ \mathbf{\Gamma}_m\end{pmatrix}\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}\right].$$

The choice

$$t_m = \frac{\operatorname{tr}\left[\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}^T \begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}\right]}{\operatorname{tr}\left[\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}^T J\begin{pmatrix}\mathbf{B}_m \\ \mathbf{\Gamma}_m\end{pmatrix}\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}\right]} = \frac{\left\|\begin{pmatrix}\mathbf{C}_m \\ \mathbf{W}_m\end{pmatrix}\right\|_F^2}{\operatorname{tr}(\mathbf{X}^T\mathbf{C}_m^T\mathbf{\Gamma}_m\mathbf{C}_m\mathbf{X}) + \frac{m}{2}\operatorname{tr}(\mathbf{\Gamma}_m^{-1}\mathbf{W}_m\mathbf{\Gamma}_m^{-1}\mathbf{W}_m)}$$

maximizes the approximation. If the support of the matrix $(\mathbf{B}, \mathbf{\Gamma})$ does not change under projection, then this IHT update is particularly apt.

4.5.7 IHT Projection

Recall that full IHT iterates according to

$$\begin{pmatrix}\mathbf{B}_{m+1} \\ \mathbf{\Gamma}_{m+1}\end{pmatrix} = P_{S_k}(\mathbf{\Delta}_{m+1}),$$

where $\mathbf{\Delta}_{m+1}$ is derived in equation (4.12). Here k is a positive integer representing the sparsity level, which is assumed known. In practice k is found through cross-validation. The projection $P_{S_k}(\mathbf{\Delta})$ splits into separate projections for $\mathbf{B}$ and $\mathbf{\Gamma}$. One can independently project each row of $\mathbf{B}$ to sparsity. Alternatively, one can require each row of $\mathbf{B}$ to have the same sparsity pattern if the same set of predictors plausibly contribute to all r traits. The $\mathbf{\Gamma}$ projection must preserve symmetry and positive semidefiniteness. Symmetry is automatic because the gradient of $\mathbf{\Gamma}$ is already symmetric.

To project to positive semidefiniteness, one takes the SVD of $\boldsymbol{\Gamma}$ and project its eigenvalues $\boldsymbol{\lambda}$ to nonnegativity. One can even project $\boldsymbol{\Gamma}$ to the closest positive definite matrix with an acceptable condition number [104].

4.5.8 The Block Ascent IHT

In block ascent we alternate updates of $\mathbf{B}$ and $\boldsymbol{\Gamma}$. The exact update (4.4) of $\boldsymbol{\Gamma}$ is particularly convenient, and we take advantage of it. Symmetry and positive semidefiniteness are automatically preserved. Inversion can be carried out via Cholesky factorization of $\boldsymbol{\Gamma}$. This choice of $\boldsymbol{\Gamma}$ simplifies the step length

$$t_m \;=\; \frac{\|\mathbf{C}_m\|_F^2}{\mathrm{tr}(\mathbf{X}^T\mathbf{C}_m^T\boldsymbol{\Gamma}\mathbf{C}_m\mathbf{X})}.$$

Note that the denominator of the step size does not require formation of the $n \times n$ matrix $\mathbf{X}^T\mathbf{C}_m^T\boldsymbol{\Gamma}\mathbf{C}_m\mathbf{X}$. One can write $\mathrm{tr}(\mathbf{X}^T\mathbf{C}_m^T\boldsymbol{\Gamma}\mathbf{C}_m\mathbf{X}) = \mathrm{tr}(\mathbf{X}^T\mathbf{C}_m^T\mathbf{L}\mathbf{L}^T\mathbf{C}_m\mathbf{X}) = \|\mathbf{L}^T\mathbf{C}_m\mathbf{X}\|_F^2$, where $\mathbf{L}$ is the Cholesky factor of $\boldsymbol{\Gamma}$. The matrix $\mathbf{L}^T\mathbf{C}_m\mathbf{X}$ is fortunately only $r \times n$.

4.5.9 UK Biobank Runtime Script

Here is the script used to perform our UK Biobank analysis

```
#
# Parameter explanations
# MvNormal: Distribution of traits is multivariate normal
# q: number of cross-validation folds
# init_beta: get good initial estimates for genetic and nongenetic regression coefficient
# min_iter: iterate at least 10 times before checking for convergence
#
```

```
using MendelIHT, Random
Random.seed!(2021)                          # seed for reproducibility
plinkfile = "ukb.plink.filtered"            # plink files without .bed/bim/fam
phenotypes = "normalized.bmi.sbp.dbp.txt" # comma-seprated phenotype file (no header line
covariates = "iht.covariates.txt"           # comma-separated covariates file (no header li

# cross-validate k = 100, 200, ..., 1000
path = 100:100:1000
@time mses = cross_validate(plinkfile, MvNormal, path=path, q=3,
    covariates=covariates, phenotypes=phenotypes, init_beta=true, min_iter=10,
    cv_summaryfile="cviht.summary.roughpath1.txt")

# cross-validate k = 110, 120, ..., 290 (assuming best k was 200 in previous step)
k_rough_guess = path[argmin(mses)]
path = (k_rough_guess - 9):10:(k_rough_guess + 9)
@time mses = cross_validate(plinkfile, MvNormal, path=path, q=3,
    covariates=covariates, phenotypes=phenotypes, init_beta=true, min_iter=10,
    cv_summaryfile="cviht.summary.roughpath2.txt")

# cross-validate k = 181, 182, ..., 209 (assuming best k was 190 in previous step)
k_rough_guess = path[argmin(mses)]
dense_path = (k_rough_guess - 9):(k_rough_guess + 9)
@time mses_new = cross_validate(plinkfile, MvNormal, path=dense_path,
    covariates=covariates, q=3, phenotypes=phenotypes, min_iter=10, init_beta=true)

# now run IHT on best k
@time iht_result = iht(plinkfile, dense_path[argmin(mses_new)], MvNormal,
```

```
    covariates=covariates, phenotypes=phenotypes, min_iter=10, init_beta=true)
```

4.6 Web Resources

Project name: MendelIHT.jl

Project home page: https://github.com/OpenMendel/MendelIHT.jl

Supported operating systems: Mac OS, Linux, Windows

Programming language: Julia 1.6

License: MIT

All commands needed to reproduce the following results are available at the `MendelIHT` site in the `manuscript` sub-folder. `SnpArrays.jl` is available at https://github.com/OpenMendel/SnpArrays.jl. `VCFTools.jl` is available at https://github.com/OpenMendel/VCFTools.jl. `BGEN.jl` is available at https://github.com/OpenMendel/BGEN.jl

CHAPTER 5

Conclusions and Future research

This dissertation presented three distinct projects with different applications in human genetics. The first project extends IHT to generalized linear models, lifting the Gaussian distribution assumption of standard univariate IHT. We also discuss grouping and weighting strategies to further enhance model selection when the association of certain predictors is supported by external evidences. The second project presents an imputation and phasing algorithm that is much faster than existing methods. Extensions to ancestry inference and new data compression strategies are also presented. The final chapter extends IHT to the multivariate setting. It offers better model selection and faster compute times compared to existing multivariate methods, possibly enabling investigators to analyze dozens or hundreds of correlated traits simultaneously.

In the remaining chapter, we discuss a few potential projects built on these results. Some projects are exploratory in nature, and others I sketch partial progress.

5.1 Tuning hyperparameters without cross validation

Penalized regression algorithms such as LASSO and IHT require tuning hyperparameters λ or k, which is typically accomplished via cross-validation. A 10-fold 10-times repeated cross validation across $k = 1, ..., 100$ requires fitting 10,000 separate IHT models, dramatically increasing computation requirement. Can we dynamically adjust k in each iteration, abandoning cross-validation altogether?

5.1.1 Block update with Knockoffs

The knockoff framework [8, 99] may replace cross-validation in model selection. The knockoff filter controls the false discovery rate for any algorithm in model selection. The idea is to construct a doppelganger $\tilde{\mathbf{X}}$ for the original design matrix $\mathbf{X}$, where $\tilde{\mathbf{X}}$ is uncorrelated with the response but mimic the correlation structure of $\mathbf{X}$ *. We perform model selection on the concatenated matrix $[\mathbf{X}, \tilde{\mathbf{X}}]$, for which $\tilde{\mathbf{X}}$ serves as negative control and consequently informs the false discovery rate.

One can adapt knockoff filters to replace cross validation, by simply trying numerous sparsity levels k and see which generated the best false discovery rate. But that suffers from the same computational burden as cross validation. Instead, we propose to adapt knockoff filters into IHT via block update:

1. Initialize IHT algorithm with some initial sparsity level k
2. Run IHT on $[\mathbf{X}\tilde{\mathbf{X}}]$ with current k for just 1 iteration
3. Determine current false discovery rate and adjust k accordingly
4. Repeat 2-3 until convergence

The crucial idea is to dynamically update the sparsity constraint k and false discovery rate information with each new iteration. Of course, convergence of IHT relies on fixed k, while false discovery control of knockoffs presumably also relies on running an algorithm to convergence. However, even if this block update strategy requires more iterations to complete, it may be a small price to pay compared to using cross-validation with 10,000x increase in compute time.

5.2 Polygenic risk scores on summary statistics

This idea is based on reference [78] and is suggested by Janet Sinsheimer during one of our weekly meetings.

---

*In particular $\mathbf{X}$ remains distributionally unchanged if a column is replaced with the corresponding column from $\tilde{\mathbf{X}}$

5.2.1 Motivation and limitations of polygenic risk scores

Polygenic risk scores (PRS) predict phenotype values using genotype data obtained from microarrays. To keep the discussion simple, let us assume the phenotype has Gaussian distribution. Then PRS relies on the following decomposition

$$\hat{y}_i = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{5.1}$$

where $\hat{y}_i$ is the predicted phenotype value for sample i, X_j is the genotype at the jth locus, and β_j is the estimated effect size for the jth locus. The main criticisms of PRS include

1. It is unclear how many β to include in equation (5.1)
2. If multiple locus are in tight linkage disequilibrium, including them all will exaggerate their contribution to the phenotype
3. Poor penetrance [109]

5.2.2 Improved model selection and estimation with IHT

The paper [78] tackles (1) and (2) above by minimizing the LASSO objective

$$f(\boldsymbol{\beta}) = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^t(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + 2\lambda \|\boldsymbol{\beta}\|_1^1 \tag{5.2}$$

$$= \mathbf{y}^t\mathbf{y} + \boldsymbol{\beta}^t\mathbf{X}^t\mathbf{X}\boldsymbol{\beta} - 2\boldsymbol{\beta}^t\mathbf{X}^t\mathbf{y} + 2\lambda \|\boldsymbol{\beta}\|_1^1 \tag{5.3}$$

where $\mathbf{R} = \mathbf{X}^t\mathbf{X}$ captures correlation among SNPs and $\mathbf{r} = \mathbf{X}^t\mathbf{y}$ captures correlation between SNPs and phenotype. In the setting where we want to exploit summary statistics, the genotypes $\mathbf{X}_r$ used to estimate $\mathbf{R}$ is not the same as $\mathbf{X}$ used to estimate $\mathbf{r}$. Thus, $\mathbf{R} = \mathbf{X}_r^t\mathbf{X}_r$. The original authors manipulates the objective $f(\boldsymbol{\beta})$ by re-expressing it in terms of a convex combination of $\boldsymbol{\beta}^t\mathbf{X}_r^t\mathbf{X}_r\boldsymbol{\beta}$ and $\boldsymbol{\beta}^t\boldsymbol{\beta}$.

From this exposition, it seems we might be able to replace the ℓ_1 penalty with the ℓ_0 IHT penalty

in equation (5.2). This will potentially give rise to numerous advantages. First, as shown in section 2, IHT recovers sharper estimates for the regression coefficients β and finds far fewer false positives under cross validation. Both seem crucial for PRS. Second, IHT may implicitly adjust for problem (2) above. In our experience, if IHT picked multiple SNPs in LD, the total contribution of the LD block will simply be divided among those SNPs. Finally, because `MendelIHT.jl` [27] already supports sparse linear regression with IHT penalty, modifying the existing code to support summary statistics regression may be straightforward.

5.3 Extended discussion on MendelImpute.jl

In chapter 3, we briefly discussed how the `MendelImpute.jl` [29] software can be used for admixture mapping. Unfortunately, due to lack of appropriate data, there are a number of potential improvements and applications that we glanced over. Here we summarize two potential enhancements for admixture mapping and some ideas to improving the existing softwares.

5.3.1 Supervised global ancestry estimation

As discussed in chapter 1, standard methods for obtaining global ancestry estimates relies on unsupervised clustering approaches. Because these methods are unsupervised, often they fail to distinguish sub-population structures when the sub-population contains a small number of samples [6]. `MendelImpute.jl` may potentially overcome this limitation because it matches observed genotypes $\mathbf{X}$ to those present in a haplotype reference panel $\mathbf{H}$. Thus, the rarity of sample population within $\mathbf{X}$ should not matter in principle, as long as said population are present abundantly in the reference panel $\mathbf{H}$. We also illustrate as a proof-of-principle that `MendelImpute.jl` is possibly superior to the best unsupervised approach (see Figure 3.3b) in our real data analysis. Future work may want to investigate this detail and justify our findings via simulation studies.

5.3.2 Phasing by considering ancestral origins

Our local ancestry plot (Figure 3.3a) reveals that the displayed recombination rate is possibly higher than we would normally expect. On closer inspection, we observe that many haplotype blocks can be exchanged with the opposing haplotype block in the other chromosome. This action would absolve one or both of the observed recombination event, decreasing the observed recombination rate and possibly improve phasing accuracy.

To decide when to perform an exchange, one can exploit ancestral origin information. When we intersect haplotype pairs across windows during phasing (see section 3.2.3), the current method only considers exact haplotype matches across windows. However, candidate haplotypes are often labeled with ancestral origins. The intersecting strategy can thus be amended to also consider haplotype origin as an alternative or additional metric for phasing. Whether this extra information should be considered in every window or just be used to resolve ties will require experiments.

5.3.3 Potential improvements to imputation accuracy

Here we sketch a few ideas to potentially improve imputation accuracy.

- **Within-window breakpoints:** Consider recombination within genomic window as noted in section 3. If a breakpoint occurs near the middle of a genomic window, the search routine for the best haplotype pair may be completely thrown-off since it has to compromise for breakpoint.
- Phasing may additionally incorporate ancestral origin information, as discussed above in previous subsection
- **Overlapping windows:** Brian Browning said in his talk [19] that HMM methods typically implement overlapping windows. Currently genomic windows in `MendelImpute` are disjoint.
- **Weighted least squares:** The least squares criteria for imputation treats each SNP equally. We may want to give certain alleles higher weight, depending on e.g. minor allele frequency

or ancestral informativeness.

- Currently, if a breakpoint occurs, the untyped SNPs between 2 genomic window will all be assigned to one of the window. This is an implementation detail that was hard to overcome, but of course, those untyped SNPs should be divided from the middle, half assigned to the first window and the remaining assigned to the next.

5.3.4 Potential improvements to software

Here we sketch a few ideas to improve the performance and user-friendliness of `MendelImpute.jl`.

- **Parallel read of VCF files**. Currently `VCFTools.jl` (the back-end for parsing raw VCF files) does not support parallel read. Parallel read can be enabled by supporting `tabix` index files, as discussed in section 3.
- **Accommodate Multiallelic markers**. Currently multi-allelic markers (markers where a SNP have > 1 alternate alleles) must be filtered out. To impute multi-allelic markers, we can align markers with SNP id instead of SNP positions, since I think multiallelic SNPs have the same allelic position but different SNP ids. This can also allow us to impute target panels against reference panels even if they are on different builds.
- **Ignore typed SNPs not present on reference panel**. Currently, all typed SNPs must be present on the reference panel. This ensures that all output genotypes can be phased. If we do not insist on outputting phased data, we should add an option so that MendelImpute automatically ignores those typed SNPs and just output them unphased.

REFERENCES

[1] 1000 Genomes Project Consortium, A. Auton, G. R. Abecasis, D. M. Altshuler, R. M. Durbin, et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[2] 1000 Genomes Project Consortium, G. A. McVean, D. M. Altshuler, R. M. Durbin, et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

[3] G. Abraham, Y. Qiu, and M. Inouye. FlashPCA: principal component analysis of Biobank-scale genotype datasets. *Bioinformatics*, 2017.

[4] D. H. Alexander and K. Lange. Enhancements to the admixture algorithm for individual ancestry estimation. *BMC bioinformatics*, 12(1):1–6, 2011.

[5] D. H. Alexander and K. Lange. Stability selection for genome-wide association. *Genetic Epidemiology*, 35(7):722–728, 2011.

[6] D. H. Alexander, J. Novembre, and K. Lange. Fast model-based estimation of ancestry in unrelated individuals. *Genome research*, 19(9):1655–1664, 2009.

[7] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.

[8] R. F. Barber and E. J. Candès. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2055–2085, 2015.

[9] A. Beck. *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*, volume 19. Siam, 2014.

[10] A. Beck and M. Teboulle. A linearly convergent algorithm for solving a class of nonconvex/affine feasibility problems. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, pages 33–48. Springer, 2011.

[11] M. Besançon, D. Anthoff, A. Arslan, S. Byrne, D. Lin, T. Papamarkou, and J. Pearson. Distributions.jl: Definition and modeling of probability distributions in the juliastats ecosystem. *arXiv e-prints*, page arXiv:1907.08611, Jul 2019.

[12] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[13] L. S. Blackford, A. Petitet, R. Pozo, K. Remington, R. C. Whaley, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, et al. An updated set of basic linear algebra subprograms (blas). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.

[14] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27:265–274, 2009.

[15] T. Blumensath and M. E. Davies. Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of Selected Topics in Signal Processing*, 4:298–309, 2010.

[16] P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *Annals of Applied Statistics*, 5:232–253, 2011.

[17] P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The Annals of Applied Statistics*, 5(1):232, 2011.

[18] R. Brown and B. Pasaniuc. Enhanced methods for local ancestry assignment in sequenced admixed individuals. *PLoS Comput Biol*, 10(4):e1003555, 2014.

[19] B. Browning. An introduction to genotype imputation.

[20] B. L. Browning, Y. Zhou, and S. R. Browning. A one-penny imputed genome from next-generation reference panels. *The American Journal of Human Genetics*, 103(3):338–348, 2018.

[21] K. Bryc, A. Auton, M. R. Nelson, J. R. Oksenberg, S. L. Hauser, S. Williams, A. Froment, J.-M. Bodo, C. Wambebe, S. A. Tishkoff, et al. Genome-wide patterns of population structure and admixture in West Africans and African Americans. *Proceedings of the National Academy of Sciences*, 107(2):786–791, 2010.

[22] W. S. Bush and J. H. Moore. Genome-wide association studies. *PLoS Computational Biology*, 8:e1002822, 2012.

[23] R. M. Cantor, K. Lange, and J. S. Sinsheimer. Prioritizing GWAS results: a review of statistical methods and recommendations for their application. *The American Journal of Human Genetics*, 86:6–22, 2010.

[24] C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee. Second-generation PLINK: rising to the challenge of larger and richer datasets. *GigaScience*, 4:7, 2015.

[25] G. K. Chen, K. Wang, A. H. Stram, E. M. Sobel, and K. Lange. Mendel-GPU: haplotyping and genotype imputation on graphics processing units. *Bioinformatics*, 28(22):2979–2980, 2012.

[26] E. C. Chi, H. Zhou, G. K. Chen, D. O. Del Vecchyo, and K. Lange. Genotype imputation via matrix completion. *Genome Research*, 23(3):509–518, 2013.

[27] B. B. Chu, K. L. Keys, C. A. German, H. Zhou, J. J. Zhou, J. S. Sinsheimer, and K. Lange. Iterative hard thresholding in gwas: Generalized linear models, prior weights, and double sparsity. *bioRxiv*, 697755, 2019.

[28] B. B. Chu, K. L. Keys, C. A. German, H. Zhou, J. J. Zhou, E. M. Sobel, J. S. Sinsheimer, and K. Lange. Iterative hard thresholding in genome-wide association studies: Generalized linear models, prior weights, and double sparsity. *GigaScience*, 9(6):giaa044, 2020.

[29] B. B. Chu, E. M. Sobel, R. Wasiolek, J. S. Sinsheimer, H. Zhou, and K. Lange. A Fast Data-Driven Method for Genotype Imputation, Phasing, and Local Ancestry Inference: MendelImpute.jl. *arXiv preprint DOI:10.1101/2020.10.24.353755*, 2020.

[30] S. Das, G. R. Abecasis, and B. L. Browning. Genotype imputation from large reference panels. *Annual Review of Genomics and Human Genetics*, 19:73–96, 2018.

[31] S. Das, L. Forer, S. Schönherr, C. Sidore, A. E. Locke, A. Kwong, S. I. Vrieze, E. Y. Chew, S. Levy, M. McGue, et al. Next-generation genotype imputation service and methods. *Nature Genetics*, 48(10):1284, 2016.

[32] A. P. Dawid. Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, 68(1):265–274, 1981.

[33] de Lamare, Rodrigo C. Knowledge-aided normalized iterative hard thresholding algorithms and applications to sparse reconstruction. *arXiv preprint arXiv:1809.09281*, 2018.

[34] M. C. Delfour. *Introduction to Optimization and Semidifferential Calculus*. SIAM, 2012.

[35] A. J. Dobson and A. Barnett. *An introduction to generalized linear models*. Chapman and Hall/CRC, 2008.

[36] M. A. Ferreira and S. M. Purcell. A multivariate test of association. *Bioinformatics*, 25(1):132–133, 2009.

[37] R. Finnegan and L. White. invenia/JLSO.jl: Storage container for serialized Julia objects. https://doi.org/10.5281/zenodo.3992374, 2020.

[38] S. Foucart. Hard thresholding pursuit: an algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49:2543–2563, 2011.

[39] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *arXiv preprint arXiv:1001.0736*, 2010.

[40] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33:1, 2010.

[41] N. A. Furlotte and E. Eskin. Efficient multiple-trait association and estimation of genetic correlation using the matrix-variate linear mixed model. *Genetics*, 200(1):59–68, 2015.

[42] L. Gai and E. Eskin. Finding associated variants in genome-wide association studies on multiple traits. *Bioinformatics*, 34:i467–i474, 2018.

[43] T. E. Galesloot, K. Van Steen, L. A. Kiemeney, L. L. Janss, and S. H. Vermeulen. A comparison of multivariate genome-wide association methods. *PloS one*, 9(4), 2014.

[44] C. A. German, J. S. Sinsheimer, Y. C. Klimentidis, H. Zhou, and J. J. Zhou. Ordered multinomial regression for genetic association analysis of ordinal phenotypes at Biobank scale. *Genetic Epidemiology*, 2019.

[45] C. A. German and H. Zhou. MendelPlots.jl: Julia package for plotting results from GWAS, Feb. 2020.

[46] A. GreenWell and M. Abbott. GroupSlices.jl: A package for the groupslices and associated functions. https://github.com/mcabbott/GroupSlices.jl, 2019.

[47] W. Gropp, W. D. Gropp, E. Lusk, A. Skjellum, and A. D. F. E. E. Lusk. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

[48] Y. Guan and M. Stephens. Bayesian variable selection regression for genome-wide association studies and other large-scale problems. *The Annals of Applied Statistics*, pages 1780–1815, 2011.

[49] B. Han and E. Eskin. Random-effects model aimed at discovering associations in meta-analysis of genome-wide association studies. *The American Journal of Human Genetics*, 88:586–598, 2011.

[50] G. E. Hoffman, B. A. Logsdon, and J. G. Mezey. PUMA: A unified framework for penalized multiple regression analysis of GWAS data. *PLoS Computational Biology*, 9:e1003101, 06 2013.

[51] B. Howie, C. Fuchsberger, M. Stephens, J. Marchini, and G. R. Abecasis. Fast and accurate genotype imputation in genome-wide association studies through pre-phasing. *Nature Genetics*, 44(8):955–959, 2012.

[52] S. S. Ji, C. A. German, K. Lange, J. S. Sinsheimer, H. Zhou, J. Zhou, and E. M. Sobel. Modern simulation utilities for genetic analysis. *BMC bioinformatics*, 22(1):1–13, 2021.

[53] B. Jiang, S. Ma, J. Causey, L. Qiao, M. P. Hardin, I. Bitts, D. Johnson, S. Zhang, and X. Huang. SparRec: An effective matrix completion framework of missing data imputation for GWAS. *Scientific Reports*, 6:35534, 2016.

[54] L. Jiang, Z. Zheng, T. Qi, K. E. Kemper, N. R. Wray, P. M. Visscher, and J. Yang. A resource-efficient tool for mixed model association analysis of large-scale data. *Nature Genetics*, 51(12):1749–1755, 2019.

[55] J. Kelleher, A. M. Etheridge, and G. McVean. Efficient coalescent simulation and genealogical analysis for large sample sizes. *PLoS Comput Biol*, 12(5):1–22, 05 2016.

[56] K. L. Keys, G. K. Chen, and K. Lange. Iterative hard thresholding for model selection in genome-wide association studies. *Genetic Epidemiology*, 41:756–768, 2017.

[57] S. Ko, G. X. Li, H. Choi, and J.-H. Won. Computationally scalable regression modeling for ultrahigh-dimensional omics data with ParProx. *Briefings in Bioinformatics*, in press, 2021.

[58] S. Ko, H. Zhou, J. Zhou, and J.-H. Won. Diststat. jl: Towards unified programming for high-performance statistical computing environments in julia. *arXiv preprint arXiv:2010.16114*, 2020.

[59] S. Ko, H. Zhou, J. J. Zhou, and J.-H. Won. High-performance statistical computing in the computing environments of the 2020s. *Statistical Science*, in press, 2021.

[60] K. Lange. *Numerical analysis for statisticians*. Springer Science & Business Media, 2010.

[61] K. Lange. *MM optimization algorithms*, volume 147. SIAM, 2016.

[62] K. Lange, J. C. Papp, J. S. Sinsheimer, R. Sripracha, H. Zhou, and E. M. Sobel. Mendel: the swiss army knife of genetic analysis programs. *Bioinformatics*, 29:1568–1570, 2013.

[63] I. Lappalainen, J. Almeida-King, V. Kumanduri, A. Senf, J. D. Spalding, G. Saunders, J. Kandasamy, M. Caccamo, R. Leinonen, B. Vaughan, et al. The European genome-phenome archive of human data consented for biomedical research. *Nature Genetics*, 47(7):692–695, 2015.

[64] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic Linear Algebra Subprograms for Fortran Usage. *ACM Transactions on Mathematical Software*, 5(3):308–323, 1979.

[65] H. Li. Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics*, 27(5):718–719, 2011.

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

[67] D. Lin, J. M. White, S. Byrne, D. Bates, A. Noack, J. Pearson, A. Arslan, K. Squire, D. Anthoff, T. Papamarkou, M. Besançon, and et al. JuliaStats/Distributions.jl: a Julia package for probability distributions and associated functions, may 2019.

[68] J. M. Lind, H. B. Hutcheson-Dilks, S. M. Williams, J. H. Moore, M. Essex, E. Ruiz-Pesini, D. C. Wallace, S. A. Tishkoff, S. J. O'Brien, and M. W. Smith. Elevated male European and female African contributions to the genomes of African American individuals. *Human genetics*, 120(5):713–722, 2007.

[69] E. Y. Liu, M. Li, W. Wang, and Y. Li. MaCH-Admix: genotype imputation for admixed populations. *Genetic epidemiology*, 37(1):25–37, 2013.

[70] P.-R. Loh, P. Danecek, P. F. Palamara, C. Fuchsberger, Y. A. Reshef, H. K. Finucane, S. Schoenherr, L. Forer, S. McCarthy, G. R. Abecasis, et al. Reference-based phasing using the haplotype reference consortium panel. *Nature Genetics*, 48(11):1443, 2016.

[71] P.-R. Loh, G. Kichaev, S. Gazal, A. P. Schoech, and A. L. Price. Mixed-model association for biobank-scale datasets. *Nature Genetics*, 50(7):906–908, 2018.

[72] P.-R. Loh, G. Tucker, B. K. Bulik-Sullivan, B. J. Vilhjalmsson, H. K. Finucane, R. M. Salem, D. I. Chasman, P. M. Ridker, B. M. Neale, B. Berger, N. Patterson, and A. L. Price. Efficient bayesian mixed-model analysis increases association power in large cohorts. *Nature Genetics*, 47:284, 2015.

[73] J. MacArthur, E. Bowler, M. Cerezo, L. Gil, P. Hall, E. Hastings, H. Junkins, A. McMahon, A. Milano, J. Morales, and et al. The new NHGRI-EBI Catalog of published genome-wide association studies (GWAS Catalog). *Nucleic acids research*, 45:D896–D901, 2016.

[74] J. MacArthur, E. Bowler, M. Cerezo, L. Gil, P. Hall, E. Hastings, H. Junkins, A. McMahon, A. Milano, J. Morales, et al. The new NHGRI-EBI catalog of published genome-wide association studies (GWAS Catalog). *Nucleic Acids Research*, 45(D1):D896–D901, 2017.

[75] R. Mägi, Y. V. Suleimanov, G. M. Clarke, M. Kaakinen, K. Fischer, I. Prokopenko, and A. P. Morris. Scopa and meta-scopa: software for the analysis and aggregation of genome-wide association studies of multiple correlated phenotypes. *BMC Bioinformatics*, 18(1):1–8, 2017.

[76] R. Magno and A.-T. Maia. gwasrapidd: an R package to query, download and wrangle GWAS Catalog data. *Bioinformatics*, pages 1–2, 2019.

[77] J. R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics , 2nd edition*. John Wiley & Sons, 2019.

[78] T. S. H. Mak, R. M. Porsch, S. W. Choi, X. Zhou, and P. C. Sham. Polygenic scores via penalized regression on summary statistics. *Genetic epidemiology*, 41(6):469–480, 2017.

[79] A. R. Martin, K. J. Karczewski, S. Kerminen, M. I. Kurki, A.-P. Sarin, M. Artomov, J. G. Eriksson, T. Esko, G. Genovese, A. S. Havulinna, and et al. Haplotype sharing provides insights into fine-scale population history and disease in finland. *The American Journal of Human Genetics*, 102:760–775, 2018.

[80] R. Mazumder, J. H. Friedman, and T. Hastie. SparseNet: Coordinate descent with nonconvex penalties. *Journal of the American Statistical Association*, 106:1125–1138, 2011.

[81] S. McCarthy, S. Das, W. Kretzschmar, O. Delaneau, A. R. Wood, A. Teumer, H. M. Kang, C. Fuchsberger, P. Danecek, K. Sharp, et al. A reference panel of 64,976 haplotypes for genotype imputation. *Nature Genetics*, 48(10):1279, 2016.

[82] P. McCullagh. *Generalized Linear Models*. Routledge, 2018.

[83] L. Meier, S. Van De Geer, and P. Bühlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70:53–71, 2008.

[84] N. Meinshausen and P. Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72:417–473, 2010.

[85] S. Melquist, D. W. Craig, M. J. Huentelman, R. Crook, J. V. Pearson, M. Baker, V. L. Zismann, J. Gass, J. Adamson, S. Szelinger, et al. Identification of a novel risk locus for progressive supranuclear palsy by a pooled genomewide scan of 500,288 single-nucleotide polymorphisms. *The American Journal of Human Genetics*, 80:769–778, 2007.

[86] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26:301–321, 2009.

[87] E. T. Norris, L. Rishishwar, A. T. Chande, A. B. Conley, K. Ye, A. Valderrama-Aguirre, and I. K. Jordan. Admixture-enabled selection for rapid adaptive evolution in the Americas. *Genome biology*, 21(1):1–12, 2020.

[88] E. T. Norris, L. Wang, A. B. Conley, L. Rishishwar, L. Mariño-Ramírez, A. Valderrama-Aguirre, and I. K. Jordan. Genetic ancestry, admixture and health determinants in Latin America. *BMC genomics*, 19(8):75–87, 2018.

[89] P. F. O'Reilly, C. J. Hoggart, Y. Pomyen, F. C. Calboli, P. Elliott, M.-R. Jarvelin, and L. J. Coin. MultiPhen: joint model of multiple phenotypes can increase discovery in GWAS. *PloS One*, 7(5):e34861, 2012.

[90] H. F. Porter and P. F. O'Reilly. Multivariate simulation framework reveals performance of multi-trait GWAS methods. *Scientific Reports*, 7(1):1–12, 2017.

[91] A. L. Price, N. A. Zaitlen, D. Reich, and N. Patterson. New approaches to population stratification in genome-wide association studies. *Nature Reviews Genetics*, 11(7):459–463, 2010.

[92] J. K. Pritchard, M. Stephens, and P. Donnelly. Inference of population structure using multilocus genotype data. *Genetics*, 155(2):945–959, 2000.

[93] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. A. Ferreira, D. Bender, J. Maller, P. Sklar, P. I. De Bakker, M. J. Daly, and et al. PLINK: a tool set for whole-genome association and population-based linkage analyses. *The American Journal of Human Genetics*, 81:559–575, 2007.

[94] J. Qian, Y. Tanigawa, W. Du, M. Aguirre, C. Chang, R. Tibshirani, M. A. Rivas, and T. Hastie. A fast and scalable framework for large-scale and ultrahigh-dimensional sparse regression with application to the UK Biobank. *PLoS Genetics*, 16(10):e1009141, 2020.

[95] S. K. Rahman, M. M. Sathik, and K. S. Kannan. Multiple linear regression models in outlier detection. *International Journal of Research in Computer Science*, 2(2):23, 2012.

[96] S. Rubinacci, O. Delaneau, and J. Marchini. Genotype imputation using the Positional Burrows Wheeler transform. *PLoS Genetics*, 16(11):e1009049, 2020.

[97] C. Sabatti, S. K. Service, A.-L. Hartikainen, A. Pouta, S. Ripatti, J. Brodsky, C. G. Jones, N. A. Zaitlen, T. Varilo, M. Kaakinen, and et al. Genome-wide association analysis of metabolic traits in a birth cohort from a founder population. *Nature genetics*, 41:35, 2009.

[98] A. P. Schoech, D. M. Jordan, P.-R. Loh, S. Gazal, L. J. O'Connor, D. J. Balick, P. F. Palamara, H. K. Finucane, S. R. Sunyaev, and A. L. Price. Quantification of frequency-dependent genetic architectures in 25 uk biobank traits reveals action of negative selection. *Nature communications*, 10(1):790, 2019.

[99] M. Sesia, S. Bates, E. Candès, J. Marchini, and C. Sabatti. FDR control in GWAS with population structure. *bioRxiv*, pages 2020–08, 2021.

[100] E. Sobel, K. Lange, J. R. O'Connell, and D. E. Weeks. Haplotyping algorithms. In *Genetic Mapping and DNA Sequencing*, pages 89–110. Springer, 1996.

[101] M. Stephens. A unified framework for association analysis with multiple related phenotypes. *PloS One*, 8(7):e65245, 2013.

[102] C. Sudlow, J. Gallacher, N. Allen, V. Beral, P. Burton, J. Danesh, P. Downey, P. Elliott, J. Green, M. Landray, B. Liu, P. Matthews, G. Ong, J. Pell, A. Silman, A. Young, T. Sprosen, T. Peakman, and R. Collins. UK BioBank: an open access resource for identifying the causes of a wide range of complex diseases of middle and old age. *PLoS Medicine*, 12:e1001779, 2015.

[103] D. Taliun, D. N. Harris, M. D. Kessler, J. Carlson, Z. A. Szpiech, R. Torres, S. A. G. Taliun, A. Corvelo, S. M. Gogarten, H. M. Kang, et al. Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program. *Nature*, 590:290–299, 2021.

[104] M. Tanaka and K. Nakata. Positive definite matrix approximation with condition number constraint. *Optimization Letters*, 8(3):939–947, 2014.

[105] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[106] M. C. Turchin and M. Stephens. Bayesian multivariate reanalysis of large genetic studies identifies many new associations. *PLoS Genetics*, 15(10):e1008431, 2019.

[107] S. Vattikuti, J. J. Lee, C. C. Chang, S. D. Hsu, and C. C. Chow. Applying compressed sensing to genome-wide association studies. *GigaScience*, 3:10, 2014.

[108] P. M. Visscher, N. R. Wray, Q. Zhang, P. Sklar, M. I. McCarthy, M. A. Brown, and J. Yang. 10 years of GWAS discovery: biology, function, and translation. *The American Journal of Human Genetics*, 101:5–22, 2017.

[109] N. J. Wald and R. Old. The illusion of polygenic disease risk prediction. *Genetics in Medicine*, 21(8):1705–1707, 2019.

[110] P. K. Whelton, R. M. Carey, W. S. Aronow, D. E. Casey, K. J. Collins, C. D. Himmelfarb, S. M. DePalma, S. Gidding, K. A. Jamerson, D. W. Jones, et al. 2017 ACC/AHA/AAPA/ABC/ACPM/AGS/APhA/ASH/ASPC/NMA/PCNA guideline for the prevention, detection, evaluation, and management of high blood pressure in adults: a report of the American College of Cardiology/American Heart Association Task Force on Clinical Practice Guidelines. *Journal of the American College of Cardiology*, 71(19):e127–e248, 2018.

[111] M. C. Wu, S. Lee, T. Cai, Y. Li, M. Boehnke, and X. Lin. Rare-variant association testing for sequencing data with the sequence kernel association test. *The American Journal of Human Genetics*, 89(1):82–93, 2011.

[112] T. T. Wu, Y. F. Chen, T. Hastie, E. Sobel, and K. Lange. Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25:714–721, 2009.

[113] T. T. Wu and K. Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, 2:224–244, 2008.

[114] J. Xu, E. Chi, and K. Lange. Generalized Linear Model Regression under Distance-to-set Penalties. In *Advances in Neural Information Processing Systems 30.*, pages 1385–1395. Curran Associates, Inc., 2017.

[115] F. Yang, R. F. Barber, P. Jain, and J. Lafferty. Selective inference for group-sparse linear models. In *Advances in Neural Information Processing Systems*, pages 2469–2477, 2016.

[116] J. Yang, S. H. Lee, M. E. Goddard, and P. M. Visscher. GCTA: a tool for genome-wide complex trait analysis. *The American Journal of Human Genetics*, 88:76–82, 2011.

[117] J. Yang, N. A. Zaitlen, M. E. Goddard, P. M. Visscher, and A. L. Price. Advantages and pitfalls in the application of mixed-model association methods. *Nature genetics*, 46(2):100–106, 2014.

[118] J. Yin and H. Li. Model selection and estimation in the matrix normal graphical model. *Journal of Multivariate Analysis*, 107:119–140, 2012.

[119] X.-T. Yuan, P. Li, and T. Zhang. Gradient hard thresholding pursuit. *Journal of Machine Learning Research*, 18:166–1, 2017.

[120] A. Zeileis, C. Kleiber, and S. Jackman. Regression models for count data in R. *Journal of Statistical Software*, 27:1–25, 2008.

[121] J. Zeng, R. De Vlaming, Y. Wu, M. R. Robinson, L. R. Lloyd-Jones, L. Yengo, C. X. Yap, A. Xue, J. Sidorenko, A. F. McRae, et al. Signatures of negative selection in the genetic architecture of human complex traits. *Nature genetics*, 50(5):746, 2018.

[122] C.-H. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38:894–942, 2010.

[123] T. Zhang. Analysis of multi-stage convex relaxation for sparse regularization. *Journal of Machine Learning Research*, 11:1081–1107, 2010.

[124] H. Zhou, D. Alexander, M. Sehl, J. Sinsheimer, E. Sobel, and K. Lange. Penalized regression for genome-wide association screening of sequence data. *Pacific Symposium on Biocomputing*, 2011:106–117, 2011.

[125] H. Zhou, K. Lange, and M. Suchard. Graphical processing units and high-dimensional optimization. *Statistical Science*, 25:311–324, 2010.

[126] H. Zhou, J. C. Papp, S. Ko, C. A. German, J. Day, M. A. Suchard, A. Landeros, and A. Noack. SnpArrays.jl: Julia package for compressed storage of SNP data, Feb. 2020.

[127] H. Zhou, M. E. Sehl, J. S. Sinsheimer, and K. Lange. Association screening of common and rare genetic variants by penalized regression. *Bioinformatics*, 26:2375, 2010.

[128] H. Zhou, J. S. Sinsheimer, D. M. Bates, B. B. Chu, C. A. German, S. S. Ji, K. L. Keys, J. Kim, S. Ko, G. D. Mosher, et al. OpenMendel: a cooperative programming project for statistical genetics. *Human Genetics*, 139(1):61–71, 2020.

[129] W. Zhou, J. B. Nielsen, L. G. Fritsche, R. Dey, M. E. Gabrielsen, B. N. Wolford, J. LeFaive, P. VandeHaar, S. A. Gagliano, A. Gifford, et al. Efficiently controlling for case-control imbalance and sample relatedness in large-scale genetic association studies. *Nature Genetics*, 50(9):1335–1341, 2018.

[130] X. Zhou and M. Stephens. Genome-wide efficient mixed-model analysis for association studies. *Nature Genetics*, 44(7):821, 2012.

[131] X. Zhou and M. Stephens. Efficient multivariate linear mixed model algorithms for genome-wide association studies. *Nature Methods*, 11(4):407–409, 2014.