Efficient Pedigree-Based Imputation

University of Michigan-Flint

COMPUTER SCIENCE AND INFORMATION SYSTEMS

Khalid Kunji

Presented to the Computer Science and Information Systems Faculty at the University of Michigan-Flint in partial fulfillment of the requirements for the Master of Computer Science

5 December 2018

First Reader Murali Mani _____

Second Reader Halil Bisgin _____

Efficient Pedigree-Based Imputation

University of Michigan-Flint

COMPUTER SCIENCE AND INFORMATION SYSTEMS

Khalid Kunji

Presented to the Computer Science and Information Systems Faculty
at the University of Michigan-Flint in partial fulfillment of the
requirements for the Master of Computer Science

5 December 2018

First Reader Murali Mani

Second Reader Halil Bisgin

Khalid Kunji

ttbek0@gmail.com
ttbek@umich.edu
ttbek@umflint.edu

ORCID ID: 0000-0003-2946-512X

Abstract

When performing a Genome-Wide Association Study (GWAS), one attempts to associate a phenotype with some genomic information, commonly a gene or set of genes. Often we wish to have more accuracy and attempt to identify a Single Nucleotide Polymorphism (SNP) or Single Nucleotide Polymorphisms (SNPs) that are associated with the phenotype. Sometimes a GWAS is also used to associate other kinds of genetic data, like methylation or Copy Number Variations (CNVs) with the phenotype. The phenotype in such studies is often a disease, e.g. Type II Diabetes Melitus (T2D), Coronary Heart Disease (CHD), cancer, or others, but can be other traits as well, for instance, height, weight, eye color, or intelligence. In order to perform a GWAS it is necessary to sequence the Deoxyribonucleic Acid (DNA) of the individuals in the study. This sequencing is much cheaper than it once was, but is still very expensive for large scale studies. Large scale studies are needed in order to achieve the necessary statistical power to reliably identify associations. By performing imputation we are able to increase the size of studies in two ways. Individual studies are able to sequence more individuals on their budget because they can sequence individuals for only certain sites and impute the rest of the sites to recover part of the power. Also, large scale meta-studies can impute in order to have full sequences for all the individuals in the smaller studies in order to make them comparable, this is the approach taken by Fuchsberger et al [33]. Imputation for genetic data is done in two main ways. The first way is population-based imputation, which depends on Linkage Disequilibrium (LD) and knowing the allele frequencies for a reference population that the study population is believed to be similar to. The second main way to impute is Identity By Descent (IBD)-based imputation, in which we infer genotypes based on the familial relationships in pedigree data. In this thesis, we focus on IBD-based imputation. Imputing on pedigree data can be quite time consuming, for instance, the original implementation of GIGI (Genome Imputation Given Inheritance), Cheung et al [15], took around 17 days to impute chromosome 2 (2,402,346 SNPs) of a pedigree with 189 members, using 28 GB of RAM [53]. Being able to complete family (IBD)-based imputation in a timely manner with high accuracy is of great value to researchers around the world, especially now as this data becomes more available to those without large budgets for sheer computing power. The basis for phasing and imputation along with the details of the calculations involved and exploration of ways to increase the speed for imputing large pedigree data are described in this thesis.

Dedication

I would like to dedicate this thesis to my mom and dad for always pushing me to do better than I thought my best was.

Declaration

I declare that this work is my own except as specifically cited and mentioned here. Several figures and large parts of the text of chapters 5 and 6 appear in other manuscripts to which I am a contributor. In particular in Kunji et al. 2017 [53], "Comparison and Assessment of Family- and Population-based Genotype Imputation Methods in Large Pedigrees" [108] (publication pending in Genome Research) and in the yet unpublished manuscript "GIGI2: A Fast Approach for Parallel Genotype Imputation in Large Pedigrees" [107], submitted (but not yet accepted) to Bioinformatics to which I am a contributor. For GIGI-Quick, I wrote the paper, did all of the coding except for the file split and merge utilities, and ran the experiments, others contributed to testing the code on real data and editing of the manuscript, the initial idea is suggested in the original GIGI paper [15] and brought to my attention by Mohamad Saad. For the comparison of large pedigree imputation, I contributed ideas, attempted to run Primal (which we found was not in a working state, see the section in this thesis), contributed to the writing and editing of the paper, and produced almost all of the figures jointly with Mostafa Hamza. For GIGI2, Ehsan Ullah contributed to the bulk of the coding. I contributed to the writing and editing of the manuscript, testing the code, running the experiments, and packaging it in various formats as well as setting up the webserver for it. The base code for the webserver is open source, as provided to the public by the coders of the Michigan Imputation Server [25], available here: `https://github.com/genepi`. Further breakdown of the contributions to these manuscripts is available upon request. Portions of chapters 5 and 6 also appear in the readme files for GIGI-Quick and GIGI2, I am the primary author of these readme files. Our contributions of code and readme information are available in the Git histories for GIGI-Quick and GIGI2 in the following locations:
GIGI-Quick: `https://cse-git.qcri.org/Imputation/GIGI-Quick/commits/master`
Split Utility: `https://cse-git.qcri.org/eullah/GIGISplit/commits/master`
Merge Utility: https://cse-git.qcri.org/eullah/GIGIMerge/commits/master
GIGI2: `https://cse-git.qcri.org/eullah/GIGI2/commits/master`

Acknowledgements

I extend my thanks to my colleagues for their patience as I completed this work with the delays that it caused their projects. Including Mohamad Saad, Raghvendra Mall, Michele Ceccarelli, and Halima Bensmail, that actually had projects put on hold or delayed as well as Ehsan Ullah, Reda Rawi, and others for their support and encouragement. If your name isn't here it isn't because you're not appreciated, but merely that I don't wish to go through the extensive version and accidentally leave off a name there, so I've only mentioned those I worked most closely with and saw most often.

I would also like to thank our collaborators at the University of Washington for their help in testing GIGI-Quick and GIGI2 on real pedigrees. In particlular to Ellen Wijsman, Hiep Nguyen, and Alejandro (Andrew) Q. Nato, Jr.

I would like to thank my advisors, Murali Mani and Halil Bisgin. I would especially like to thank Murali Mani for his extreme patience with me as I failed to put things together the first time around and for still believing in me when I wanted to attempt it again.

A great part of the computation used in this study was performed on the Raad2 cluster at Texas A&M University in Qatar.

Partially supported by grants from the US National Institutes of Health, R01 HD088431, P01 AG005136, R37 GM046255, R24 OD021324, and P50 AG005136.

Contents

Acronyms

CAAPA Consortium on Asthma among African-ancestry Populations in the Americas

CHD Coronary Heart Disease

cM centimorgan

CNV Copy Number Variation

DNA Deoxyribonucleic Acid

GWAS Genome-Wide Association Study

HapMap International Haplotype Map Project

IBD Identity By Descent

IQ Intelligence Quotient

LD Linkage Disequilibrium

MAF Minor Allele Frequency

MCMC Markov Chain Monte Carlo

POPRES POPulation REference Sample

RAM Random Access Memory

SNP Single Nucleotide Polymorphism

T2D Type II Diabetes Melitus

UCLA University of California, Los Angeles

UCSF University of California, San Francisco

Glossary

1000 Genomes Project A project with the goal of sequencing 1000 individuals with high accuracy from different populations for use as a reference population. This initial goal has been surpassed.

allele A variant of a gene or SNP, at each location in the genome humans have two variants (so we say humans are diploid organisms), one from each parent.

autozygosity Alleles or other segments of DNA that are IBD.

CAAPA CAAPA stuff

cancer A wide set of diseases involving abnormal cell growth or cell cycle that can spread through the body to the detriment of healthy/normal cells.

centimorgan A centimorgan (cM), aka map unit, is the distance between two loci for which the expected number of crossovers is 0.01 between generations. On average one centimorgan in humans corresponds to one million base pairs.

centromere The center of X shaped chromosome pair where the sister chromatids are joined.

chromatid One half of an X shaped chromosome pair joined to the other chromatid (its sister chromatid) at the centromere.

chromosome A DNA molecule containing some or all of the genetic material in an organism. In humans, during the metaphase stage of the cell cycle the chromosomes condense into a chromatid that is copied to a joined chromatid making an X shaped structure visible under a microscope.

Copy Number Variation Variations in the number of repeats of highly repetitive sequences in the telomeres.

crossover The process by which sister chromatids exchange portions of their sequence.

diploid Having two distinct sets of DNA.

E. coli Escherichia coli, a bacterium that serves as a common model organism, i.e. an organism that has been extensively characterized and studied.

founder Members of a pedigree for which no earlier ancestor is known.

fully sequenced An individual that has had their entire genome sequenced.

gamete A fully divided germline cell that merges with a corresponding gamete during reproduction (i.e. sperm and egg cells).

gene The basic unit of heredity that occupies a specific location on a chromosome. Each consists of nucleotides arranged in a linear manner. Most genes code for a specific protein or segment of protein leading to a particular characteristic or function.

genome The full set of genetic material encoding an organism.

germline Cells of the reproductive line that either are gametes can give rise to gametes.

GWAS A GWAS (genome-wide association study) is a way for scientists to identify inherited genetic variants associated with risk of disease or a particular trait. This method surveys the entire genome for genetic polymorphisms, typically single nucleotide polymorphisms (SNPs) (pronounced "snips"), that occur more frequently in cases (people with the disease or trait being assessed) than in controls (people without the disease or trait). Also called genome-wide association study.

haplotype A set of genotype markers in high LD with each other such that they are correlated and are often inherited together as a block.

Haplotype Reference Consortium A project to map a large number of haplotypes for a number of reference populations. They have started by mapping the haplotypes for the 1000 Genomes Project references and intend to incorporate other cohorts as they become available and as they have the resources to do the identify reference haplotypes.

haplotyping The process of assigning a haplotype to a segment of DNA.

HapMap Another project to map a large number of haplotypes for a number of reference populations.

Hardy-Weinberg Equilibrium Allele and genotype frequencies should remain the same from generation to generation given seven assumptions, (1) organisms are diploid, (2) sexual (as opposed to asexual) reproduction, (3) generations do not overlap, (4) random mating, (5) infinite population size, (6) allele frequencies are not affected by sex, and (7) there is no mutation, selection, or outside influence on the gene pool like that which can result from migration, admixture, or gene flow.

heterozygous Having two different alleles.

homozygous Having two of the same allele.

IBD graphs A representation of alleles at a given marker location that indicates the restrictions on possible alleles due to the relationships found between individual's genomes when considering their inheritance vectors. To solve the graph is to find the set of possible alleles after resolving these constraints.

Identity By Descent Two segments of DNA are Identical By Descent if they are identical sequences without recombination from a common ancestor.

imputation Assigning (a value) to something by inference from the value of the products or processes to which it contributes. - Google Dictionary 2nd definition, originating in finance.

inheritance vector A vector containing a number identifying the original genome that position can be traced back to for each marker position.

Intelligent Quotient A score from testing designed to measure human intelligence. They are limited in their utility and have been subject to many legitimate criticisms, especially as some have misused them to further racist and/or sexist agendas.

Kappa Statistic A κ statistic is a statistic for measuring inter-rater agreement and is differentiated by attempting to account for the possibility that they agree by chance.

Linkage Disequilibrium The deviance from Hardy-Weinberg Equilibrium between SNPs, aka a lack of independence between them, i.e. the correlation between SNPs.

Markov Chain Monte Carlo A class of methods that sample the distribution of a Markov Chain, for instance via Metropolis-Hastings or Gibbs sampling.

Mb Megabase pairs, i.e. millions of base pairs (nucleotides, aka SNPs).

meiosis The process by which germ (reproductive) cells divide into gametes.

Mendelian That which follows Mendel's laws of inheritance based on his very early studies of inheritance in pea plants. The Law of Segregation (gametes carry just one allele), the Law of Independent Assortment (the chance of inheriting alleles at each location is independent), and the Law of Dominance (there is a dominant and a recessive allele and having even one dominant allele results in a dominant phenotype while having two recessive alleles is required to show the recessive phenotype).

methylation The process by which methyl (CH3) groups are added to DNA.

methylation profile A data set indicating the methylation status over a range, or even the entirety, of the DNA.

microsatellite DNA A section of repetitive DNA, also often referred to as short tandem repeats.

Minor Allele Frequency The Minor Allele Frequency is the frequency of the minor allele at a given locus in the population.

missing heritability problem The enigma presented by the fact that studies of heritability indicate that a larger portion of phenotypic traits is inherited than has been able to have been explained based on the genotype.

mitosis The process by which somatic (non-reproductive) cells divide.

pedigree A formal description of a family structure, often with additional annotations of phenotypes and genotypes.

phasing The process of sorting out which alleles come from which of the two sets of DNA that humans possess, one from each parent.

phenotype The observable trait or traits that an organism possesses, including instances of disease.

POPulation REference Sample The Population Reference Sample, POPRES: a resource for population, disease, and pharmacological genetics research.

quantitative A measurable trait with a continuous distribution, as opposed to a categorical trait.

Random Access Memory A class of computer memory for which access times to any part of the memory for reading or writing are about the same.

recombination The process by which genetic material is exchanged between chromatids during meiosis (though it may also occur during mitosis) via crossover between the chromatids.

sequence To read the genetic material of an organism via any of a number of methods, some of which read small fragments and other that read longer strands at a time. Aside from construction of a de novo reference genome these are then aligned to an existing reference genome to place the location of each gene read.

sparsely sequenced An individual that has been sequenced at and/or around specific marker locations in the genome.

statistical power The statistical power is defined as follows:

$$P(reject\ H_0 | H_1\ is\ true)$$

For binary hypotheses, in which H_0 and H_1 are usually specific hypotheses other than H_0 being $!H_1$, then it is usually not possible to get the probabilities needed to calculate H_0. Rather than calculating the power, the power is often more useful when considered as a parameter to be met. For this, one can carry out a power analysis, in which one chooses the desired power and calculates the needed sample size to achieve that power.

telomere The ends of the chromatids where Copy Number Variation occurs.

Type II Error A type II Error is failing to reject a false null hypothesis.

Type I Error A type I Error is the rejection of a true null hypothesis.

variant Each distinguishable version of an allele.

Chapter 1

Introduction

1.1 Imputation

While performing a GWAS, on a budget, whether large or small, it is usually best to optimize the statistical power of the study for that budget. One of the main approaches for this is utilize imputation while gathering data for the study. In population-based studies where the subjects are of a known reference population it is possible to sparsely sequence everyone in the study at carefully chosen marker locations and impute all the rest based on the reference. This form of imputation works because many of the Single Nucleotide Polymorphisms (SNPs) are in LD, so they are not independent. This form of imputation has been the more popular one, in part because the tools to do it are faster and easier to run. Some existing popular tools for doing this include Beagle [9], Impute [43], and Minimac [32]. There are servers available for some of these tools, including the very well known and popular Michigan Imputation Server [25].

In order to use imputation in family/pedigree-based studies, some of the subjects will be fully sequenced while the rest will be sparsely sequenced at a chosen set of markers. Then, the family relationships will be used to impute the remaining data for the sparsely sequenced individuals, this is IBD-based imputation. This is the kind of imputation that GIGI does, and is that which will be examined in detail. Some of the main tools for doing this kind of imputation are GIGI [15] and Merlin [1].

Aside from the imputation itself, there are some other important aspects to consider when doing imputation. These each have their own tools available as well. The two largest considerations are choosing which individuals to sequence, and doing the phasing for the pedigree relationships. Some programs for selecting subjects for full sequencing are GIGI-Pick, ExomePicks, and PRIMUS. Some programs for doing the phasing are MaCH [57] (can also do imputation, but is implemented with Minimac as a two step procedure), SHAPEIT 2 [26] (w/ or without duoHMM [78]), and Eagle [61]. These tools are compared in Ullah et al [108].

1.2 Back to Basics

In order to understand the basis of these algorithms, it is useful to look back to the underlying biology. Every living organism has genes that encode for proteins, which in turn are responsible for their biological life processes. These genes are encoded in DNA, but there is also DNA that does not code for genes, but may have other functions. For the moment it is sufficient for our model to say that each protein is encoded by a segment of DNA, though there are other steps in between DNA and protein, but we do not need to go into those details here. DNA is arranged in highly compact structures that we call chromosomes. In E. coli there is only a single circular chromosome consisting of 4.6 Mb pairs. In humans, there are 23 pairs of linear chromosomes. That is, humans are diploid, they keep two different sets of their genome, totalling 46 chromosomes. Each of these chromosomes has two identical chromatids, so in total a human cell has 4 copies of the genome, 2 identical copies derived from their mother and 2 identical copies derived from their father. For our purposes we can say they have two unique genomes, referred to as A and B.

When genomes are inherited, there are many genes that are highly correlated. Some of these correlations are easily explained, for instance, by spatial proximity in the chromosome, but others have no known physical explanation or mechanism. Which genes are highly correlated in this way can be calculated for a given population, the reference population, from a known set of many samples. Some example reference populations are the 1000 Genomes Project [20], HapMap [22], CAAPA [66], and Haplotype Reference Consortium [67]. This can be extended to the SNPs within each gene. If our subjects in a GWAS study are from the same population as a reference population, then if they have a particular gene, say g1, then we can say that they are very likely to also have some other genes that are in LD with g1, say g2, g3, and g4. This is the basis upon which most population-based imputation works. This tends to work better when doing imputation with European populations than African ones because European populations have more genes in LD [34]. This also works much better for common SNPs than rare ones because it is difficult to find strong LD associations between markers when one rarely encounters a given marker. Despite the name, rare SNP markers are actually quite common [19] [31], even outnumbering common allele frequency markers. Their low frequency of occurence makes them uninformative, for both kinship and LD-based analysis. For this reason they are often underepresented in the sparse genotyping arrays used for gathering genetic information in population studies. In pedigree data it becomes possible to analyze rare alleles because they will frequently be shared by many members of the family so that they are disproportionately represented, i.e. an allele that occurs in 0.001 percent of the population may be found in 7 of 11 members in a small pedigree.

During cell division for germline cells, meiosis, there is a stage in which crossover occurs, during which segments of Deoxyribonucleic Acid (DNA) are exchanged between genome A and genome B. After meiosis one cell has divided into four gametes, each of which has only a single copy of the genome, but a portion of that genome could be from genome A or genome B due to the crossover. The program gl_auto in the MORGAN software [103] (MORGAN was not actually dubbed with that name until later in May of 1997, previously called Gibbs) is designed to use a Markov

Chain Monte Carlo (MCMC) algorithm to identify where these crossovers occur, given a pedigree of known relationships and sparsely sequenced subjects, with at least some subjects fully sequenced. The program will output inheritance vectors, which trace each location in the genome of sparsely sequenced subjects to that of a founder in the pedigree. This process is also called haplotyping. These inheritance vectors give IBD relations between the genomes of those in the pedigree. These relations can be considered as graphs, in particular as IBD graphs [104] and solved for the most likely set of SNPs that complete the graph. The program GIGI imputes by solving these IBD graphs. This kind of imputation is much better at imputing rare SNPs variants than the population-based methods are. Merlin does something similar, but by solving for the maximum likelihood more exactly via the Lander-Green algorithm [54] without sampling.

1.3 Missing Heritability

Historically, a number of rare diseases were detected in small family, pedigree-based, studies. Most of these were relatively easy to trace in the lineage because they were Mendelian diseases. In recent years, the advances in sequencing have allowed for the exploration of non-Mendelian, or quantitative traits. These traits are believed to be influenced by many genes and have an essentially continuous distribution. Some examples include height, weight, blood pressure, and intelligence (as measured by some metric, e.g. Intelligence Quotient (IQ)), as well as diseases like Type II Diabetes Melitus (T2D). These traits have have largely been explored through GWAS, with the aim of identifying strongly associated mutations. For this approach, many turned to larger sample sizes available from population-based studes, instead of family-based ones, in order get better statistical power. Thousands of risk factors for hundreds of complex traits have been discovered this way. Estimates of the amount of heritability explained by genetics suggests that the amount should be much higher than what variants discovered in GWAS type studies have been able to explain. This is known as the missing heritability problem [63]. It has since been shown that most, or even all, prior GWAS lack the statistical power to detect many of these very rare, small effect, variants, so they may be where we find this missing heritability. A number of large meta-studies have been published more recently that identify more statistically significant associated variants, and these depend heavily on imputation to get matching data sets across all the included data from individual studies. Unfortunately, even these GWAS results from the large meta studies have not yet been able to provide all that some hoped for or that they believed that they would. In order to achieve better power in detecting rare variants of small effect, it is essential to do a better job of imputing these rare variants. There has been a renewed interest in family-based studies because pedigree-based imputation methods are much better at correctly imputing rare variants than population-based imputation methods (which are better for common variants).

Chapter 2

Metrics

2.1 Power Analysis

The Power of a statistical study is the conditional probability of rejecting H_0 given that H_1 is true, for binary analyses at least. Somewhat more generally, it is 1 minus the probability of making a type II error. There are two main ways to decrease the type II error, by reducing the measurement error (obtaining more reliable measurements) or by reducing the sampling error (for instance, by increasing the sample size). Imputation allows for much larger sample sizes at a given cost, by allowing more sampled individuals with the same number of SNPs per individual as if they had all been fully sequenced in the final analysis. Unfortunately, some of this gained ground is lost back to imputation errors, that is, there is higher measurement error for imputed SNPs than genotyped SNPs. Not all variants of a SNP in a population are equally likely, and as we lean more towards examining the effects of less common SNPs, we find that they are vastly underrepresented relative to common SNPs and former studies do not have the necessary power to confidently identify the effects of rare SNPs. In order to address this, there has been a trend towards re-examining the existing data by combining them in a single large data set for meta-analyses. Some software has been developed specifically towards this goal, e.g. METAL [116]. Here, imputation remains a crtical step because many of these studies being combined did not sequence the same sets of SNPs as were sequenced in other studies.

2.2 Metrics

There are several metrics used for determining phasing and imputation accuracy. For phasing, one can use the haplotype accuracy, the imputation accuracy, or the switch error. They tend to produce the same ranking of methods [11]. The haplotype accuracy and the switch error both require the existence of a gold standard, which can be constructed experimentally. A recent review of experimental haplotyping methods is provided by Huang et al. [45]. The imputation accuracy is still a good metric for phasing accuracy because good phasing in prerequisite for successful imputation. The imputation accuracy can also be determined without a true gold standard by masking in simulated data because it doesn't require knowledge of the true haplotype phase.

2.2.1 Haplotype Accuracy

The haplotype accuracy is the number of haplotypes that have been phased fully correctly in the area being examined. When used for simulated data the true haplotype phase must still be known. Experimentally, the use of closely related individuals can be used to find the true phase at most locations via the application of Mendelian constraints. This measure isn't that good for large numbers of markers because the need for 100% correctness of the haplotype to count it becomes very unlikely as the number of markers increases.

2.2.2 Switch Error Rate

The switch error rate is the number of switches needed to change the predicted haplotype to the true haplotype phase divided by the number of locations that such an error could occur at, i.e. the number of heterzygote markers minus one (initial phase can be chosen arbitrarily).

2.2.3 Imputation Accuracy: R^2

Imputation accuracy can be measured in different ways, almost all the remaining sections of this chapter deal with imputation accuracy, which is a direct measure of how well imputed the genotypes are, and indirectly a measure of how well they are phased. These sections are name appropriately. The coefficient of determination, R^2, is the proportion of variance in the response variable that is predicted by the independent variables. This is a common metric used to evaluate imputation methods. In this case, the R^2 is calculated for each SNP between the imputed SNPs and ground truth SNPs from sequencing or simulation. There are various definitions of R^2 and some generalizations, here it is sufficient to treat it as the square of the Pearson correlation coefficient, R, given as

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where cov is the covariance and σ_X and σ_Y are the standard deviation of X and Y respectively.

2.2.4 Imputation Accuracy: Concordance Rate

The concordance rate, for purposes of measuring genetic imputation accuracy is the proportion of genotypes that are in agreement between the imputed values and the ground truth. It is the sum of the diagonal of the confusion matrix divided by the sum of all entries. The confusion matrix is shown in the IQS section below, $P_{i,j}$ indicates the value contributing to confusion matrix at the ith row and jth column. For instance, $P_{1,1}$ is 1 if the imputed value and the ground truth value are both AA, otherwise it is zero. Likewise, $P_{1,2}$ is 1 if the imputed value is AA and the ground truth is AB, and is zero otherwise. To get the total number of instances where both are AA, we sum over the total number of SNPs, so the 1st entry of the confusion matrix is $\sum_{n=1}^{N} P_{1,1_n}$.

$$P_0 = \frac{\sum_{n=1}^{N} P_{1,1_n} + \sum_{n=1}^{N} P_{2,2_n} + \sum_{n=1}^{N} P_{3,3_n}}{N}$$

The condordance rate tends to be vastly overinflated as a measure of imputation quality.

2.2.5 Imputation Accuracy: IQS

IQS is the Imputation Quality Score, it is a fairly new metric, introduced in Lin 2010 [58].

$$IQS = \frac{P_0 - P_c}{1 - P_c}$$

where P_0 is the concordance rate as described above and

$$P_c = \{\sum_{i=1}^{3}\sum_{n=1}^{N} P_{i,1_n} * \sum_{j=1}^{3}\sum_{n=1}^{N} P_{1,j_n} + \sum_{i=1}^{3}\sum_{n=1}^{N} P_{i,2_n} * \sum_{j=1}^{3}\sum_{n=1}^{N} P_{2,j_n} + \sum_{i=1}^{3}\sum_{n=1}^{N} P_{i,3_n} * \sum_{j=1}^{3}\sum_{n=1}^{N} P_{3,j_n}\} \div N^2$$

Where these sums are derived from the following confusion matrix:

		Ground Truth			
		AA	AB	BB	Row Sum
Imputed	AA	$\sum_{n=1}^{N} P_{11_n}$	$\sum_{n=1}^{N} P_{12_n}$	$\sum_{n=1}^{N} P_{13_n}$	$\sum_{j=1}^{3}\sum_{n=1}^{N} P_{1j_n}$
	AB	$\sum_{n=1}^{N} P_{21_n}$	$\sum_{n=1}^{N} P_{22_n}$	$\sum_{n=1}^{N} P_{23_n}$	$\sum_{j=1}^{3}\sum_{n=1}^{N} P_{2j_n}$
	BB	$\sum_{n=1}^{N} P_{31_n}$	$\sum_{n=1}^{N} P_{32_n}$	$\sum_{n=1}^{N} P_{33_n}$	$\sum_{j=1}^{3}\sum_{n=1}^{N} P_{3j_n}$
	Col Sum	$\sum_{i=1}^{3}\sum_{n=1}^{N} P_{i1_n}$	$\sum_{i=1}^{3}\sum_{n=1}^{N} P_{i2_n}$	$\sum_{i=1}^{3}\sum_{n=1}^{N} P_{i3_n}$	N = total number of SNPs

The authors of the IQS paper claimed that Beagle's R^2 and the concordance rate overestimated, and hence IQS should be used. They found that IQS was similar with squared correlation, i.e. non-Beagle/regular R^2 This much is not in dispute. They go on to show that R^2 and IQS deviate for low MAF SNPs, with the R^2 being the higher of the two. The authors suggest that R^2 is overestimating, showing inflation. Ultimately, inflation of R^2 is not observed in our results when compared empirically against the power. We simulated a data set based on 1000G allele frequencies dropped down through generations in 20 different pedigrees containing 1200 individuals. We masked off part of this data to impute and compared the IQS and the R^2 to the final power and we found that the IQS is in less agreement with the power than the R^2 is. Publication pending in Genome Research [108].

Power of association tests performed in European and African data for the different combination of phasing+imputation approaches using the random selection strategy for a=0.05.

Method	MAF Bin													
	[0,0.01)		[0.01,0.05)		[0.05,0.1)		[0.1,0.2)		[0.2,0.3)		[0.3,0.4)		[0.4,0.5]	
	Population													
	EUR	AFR	EUR	AFR	EUR	AFR	EUR	AFR	EUR	AFR	EUR	AFR	EUR	AFR
Observed Genotypes	0.546	0.616	0.807	0.813	0.806	0.812	0.805	0.812	0.800	0.811	0.806	0.810	0.808	0.812
Ped Pop	0.296	0.314	0.403	0.403	0.513	0.427	0.529	0.433	0.591	0.448	0.606	0.471	0.541	0.470
GIGI	0.301	0.315	0.403	0.400	0.393	0.380	0.383	0.359	0.372	0.345	0.370	0.356	0.369	0.368
Merlin	0.216	0.225	0.291	0.297	0.343	0.347	0.355	0.354	0.358	0.354	0.354	0.362	0.349	0.340
MaCH+Minimac	0.052	0.045	0.122	0.086	0.408	0.301	0.441	0.334	0.517	0.363	0.525	0.379	0.476	0.360
SHAPEIT+Minimac	0.118	0.079	0.240	0.159	0.490	0.367	0.515	0.396	0.573	0.422	0.576	0.430	0.516	0.406
duoHMM+Minimac	0.127	0.086	0.253	0.173	0.498	0.377	0.521	0.405	0.578	0.430	0.581	0.437	0.519	0.413
IMPUTE+IMPUTE	0.112	0.119	0.219	0.204	0.476	0.403	0.504	0.430	0.569	0.460	0.573	0.466	0.537	0.447
SHAPEIT+IMPUTE	0.087	0.093	0.179	0.166	0.435	0.360	0.460	0.385	0.528	0.416	0.532	0.425	0.491	0.402
duoHMM+IMPUTE	0.094	0.103	0.192	0.182	0.443	0.373	0.467	0.396	0.533	0.427	0.538	0.434	0.497	0.412
BEAGLE+BEAGLE	0.055	0.052	0.119	0.093	0.389	0.298	0.422	0.332	0.499	0.366	0.504	0.381	0.443	0.354

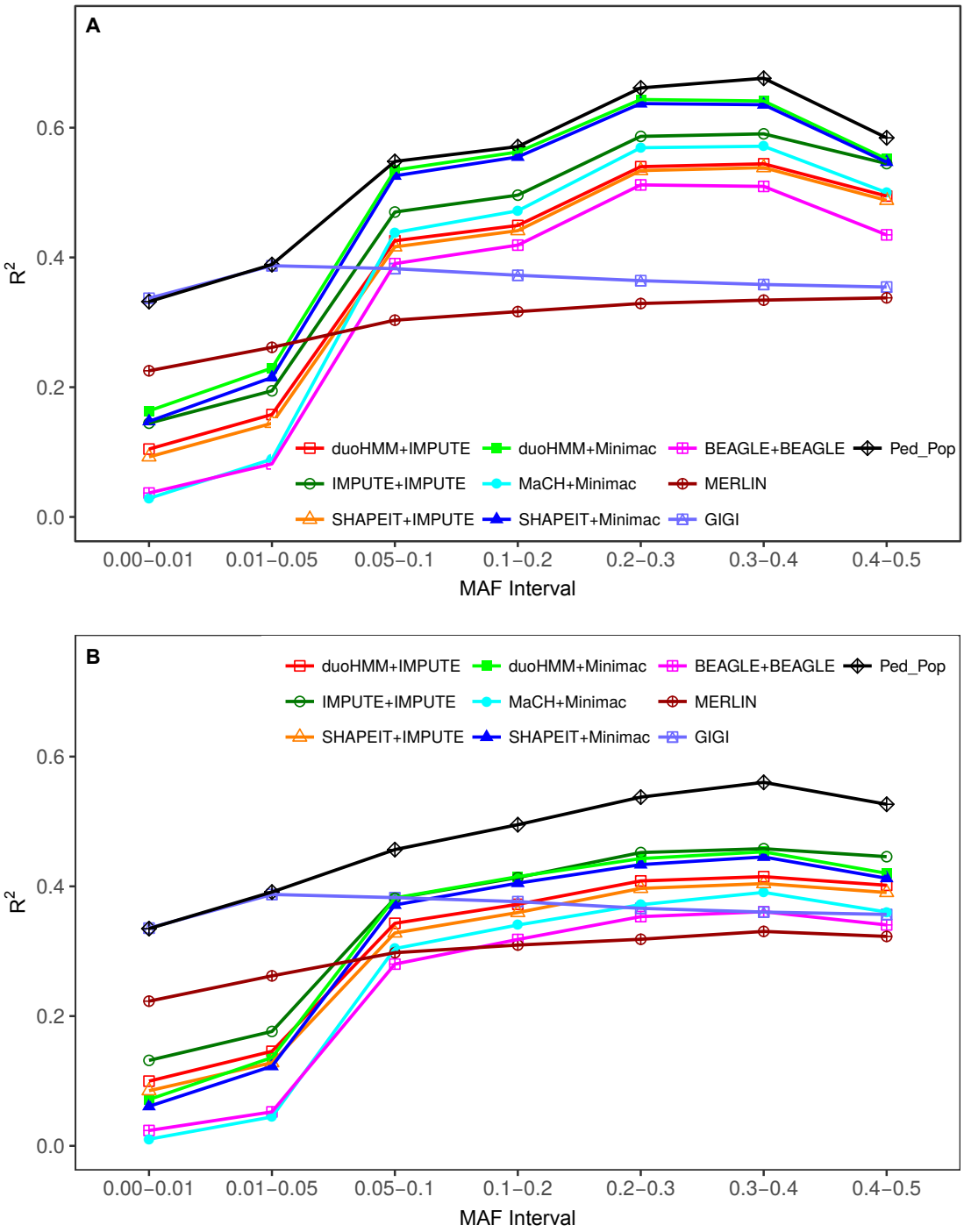

Figure 2.1: Mean correlation R^2 between true and imputed genotypes for all approaches for (A) European and (B) African populations using a random selection strategy for selecting which individuals will be fully sequenced. The first/second of a pair of programs in the key indicates phasing/imputation functions.

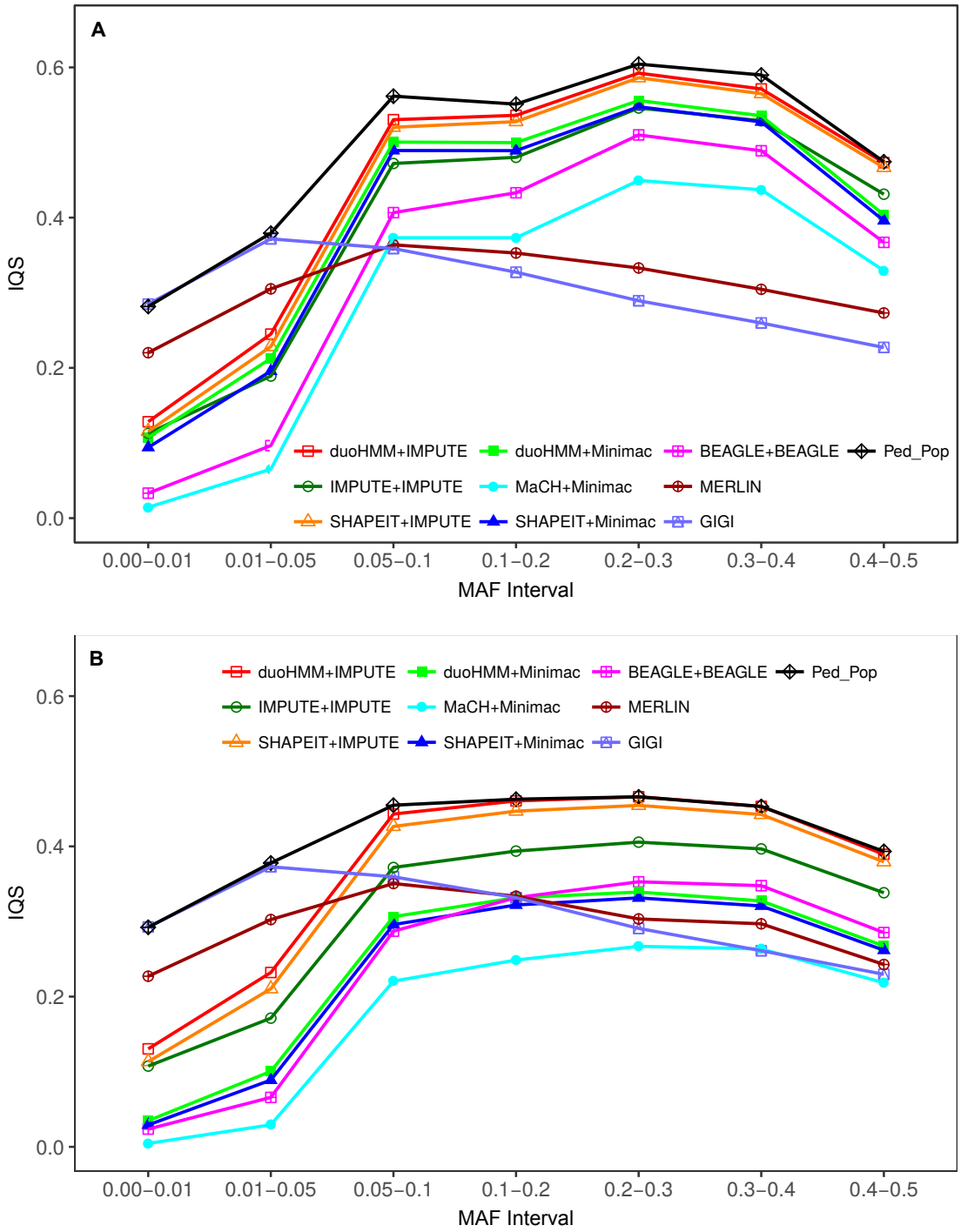

Figure 2.2: Mean IQS between true and imputed genotypes for all approaches for (A) European and (B) African populations using a random selection strategy for selecting which individuals will be fully sequenced. The first/second of a pair of programs in the key indicates phasing/imputation functions.

Consider the lines for GIGI and Merlin in Figure 2.1, the R^2 would indicate that GIGI performs better than Merlin throughout the range of MAF intervals. Looking at figure 2.2, the IQS shows Merlin to be better starting from interval [0.05,0.1) with the European population and starting from interval [0.1,0.2) for the African population. However, the R^2 is more consistent with the power as shown in table 1, in which GIGI is consistently better than Merlin for the European population and is ahead or very close to Merlin for the African population. It seems that the suggestion of the authors of the IQS paper that R^2 is inflated for variants may not be correct, rather, the IQS may be deflated. When it comes down to it, the IQS is an instance of a κ statistic, which are well known to be deflated for rare events with high agreement [110][29][18]. Yet another criticism is that the chance aspect of κ statistics, which supposedly correct for chance, assumes that all possibilities are treated as equally likely and does not take into account other possible chance models [106].

2.2.6 Imputation Accuracy: Hellinger and SEN

Though we have the R^2 and IQS scores, each software tends to report back a slightly different quality metric [82]. For instance, Beagle's R^2 is calculated between the most likely genotype and the true allele dossage [7] while MaCH reports three metrics (Number of iterations where final imputed genotype by majority vote matches, accuracy at each marker averaged across individuals, and an R^2 estimate between true and imputed genotypes by comparing the distribution of sampled genotypes in each iteration with the allele counts estimated by averaging over all iterations. They report that the last of those three measures has a correlation of 0.84 with the actual R^2 [57]. Impute2's INFO score does something quite different, incorporating the allele frequency along with the dosage [43].

$$1 - \frac{\sum_{n=1}^{N}(f_n - e_n^2)}{2N\theta(1-\theta)}$$

In order to compare between these different scores, two more scores were proposed. First, the Hellinger Score, which is a modification of the Bhattacharyya coefficient between two trinomial distributions (the posterior genotype probabilities in this case).

$$B(g) = \sum_{i=1}^{3} \sqrt{f_1^i(g) f_2^i(g)}$$

Here, B(g) is the Bhattacharyya coefficient and the following H(g) is the Hellinger score. $f_1(g)$ and $f_2(g)$ are the probability distributions, with i representing the respective vector component.

$$1 - \sqrt{1 - B(g)}$$

And secondly, the SEN, i.e. Scaled Euclidean Norm, score. The SEN score is the expected Euclidean distance between genotypes.

$$E(f_1(g)) = M^{obs} = 0 * p_{1,1} + 1 * p_{1,2} + 2 * (1 - (p_{1,1} + p_{1,2})) = 2 - (p_{1,2} + 2p_{1,1}))$$

$$E(f_2(g)) = M^{imp} = 0 * p_{1,1} + 1 * p_{1,2} + 2 * (1 - (p_{1,1} + p_{1,2})) = 2 - (p_{1,2} + 2p_{1,1}))$$

So then the SEN score is:

$$S = 1 - \frac{(M^{obs} - M^{imp})^2}{4}$$

2.2.7 G_{ST} and F_{ST}

Nei's G-statistics, or G_{ST} and F-statistics, F_{ST}, are not meant to measure imputation quality, but are used to quantify genetic similarity between groups. In the case of imputation they can be used to find the distance between sample populations and reference poplations in order to select the best reference population to use when performing imputation on one's sample. In particular Nei's 1986 G_{ST} is used here [74]

$$G_{ST} = \frac{D'_{ST}}{H_T}$$

D'_{ST} is the mean gene diversity between subpopulations and H_T is the heterozygosity that would be expected under Hardy Weinberg-Equilibrium [38][112]. The following is an estimate for G_{ST} at marker k as proposed by Bhatia in 2013 [5].

$$H_T^k = 2p_{avg}^k(1 - p_{avg}^k)$$

$$p_{avg}^k = \frac{p_1^k + p_2^k}{2}$$

$$G_{ST} = \frac{\sum_{k=1}^{N} D^k}{\sum_{k=1}^{N} H_T^k}$$

Where p_1^k and p_2^k are the allele frequencies of marker k in the 1st and 2nd population respectively.

In 2015 [83] it was demonstrated that the G_{ST} has a linear relationship with the Hellinger score for several imputation programs. This confirms the intuitive hypothesis that using a genetically closer population as the reference gives better population results. They note the African American population (AfAm from the POPRES dataset [76]) turns out to be an outlier. The authors went on to confirm that this difference was not due to the sample size alone. While one might initially think this is because AfAm is an admixed population, something similar is not observed for the Mexican population in POPRES, which one would also expect to be admixed [47][69]. Going back to the POPRES paper, the UCSF African American group was recruited from all across the US, while other groups were recruited largely from a single geographical location. Also, there may be effects due to the genetic distance between the groups admixed in the UCSF African American group vs. those admixed in the Mexican population.

Nei's G_{ST}, though called a G-Statistic, is essentially an F-Statistic and is similar in formulation to some other F-test measures and measures of genetic distance. Other such measures, some more similar than others, include: the Hudson estimator,

the WC estimator, the WH moment-based estimator, the WH-ML estimator, Nei's standard genetic distance 1972, The Cavalli-Sforza chord distance, the Reynolds, Weir, and Cockerham genetic distance, Nei's D_A distance 1983, the Goldstein distance 1995, Nei's minimum genetic distance 1973, Roger's distance 1972, and the Fixation index. The measures are numerous and some are better for specific application than others, e.g. Nei's D_A distance is known to be quite successful for microsatellite DNA data [75] [102].

Chapter 3

Phasing, Crossovers, and Haplotypes

3.1 Phasing

The data for imputation methods should usually be phased. That is, it is usually not sufficient to know the genotype alone, it is also necessary to know whether that genotype is from the paternal or maternal chromosome. In other words, at a given locus with alleles a1 and a2, we cannot treat [a1,a2] the same as [a2,a1]. In population data this is often determined based on the haplotype frequency. That is, if we have genotypes {A,C}, {G,T}, and {A,T}, they could be arranged on the chromosomes four different ways. The first chromosome (arbitrarily chosen) could read {A,G,A}, {A,G,T}, {A,T,A}, or {A,T,T} with the second chromosome having the corresponding complement. Each of these four can be considered as a haplotype and the most frequent haplotype in the population is also indicative of the most likely phase of the data. In GIGI, phasing is pre-computed by gl_auto from the MORGAN package. Many of the population-based methods use the following methods.

3.2 Population-Based

3.2.1 MaCH

MaCH utilizes an HMM to model the set of genes at the markers in an individual as an imperfect mozaic of the known haplotypes. The forward-backward algorithm for HMMs [79] is used to do this, and it has a complexity of $O(MK^2)$, where M is the number of markers and K is the number of known haplotypes. MaCH attempts to mitigate the running time by sampling a subset from K randomly.

3.2.2 Impute2

In Impute2, it uses a model similar to that of MaCH. Like MaCH, it uses a subset of the K known haplotypes to reduce the complexity. These are chosen to be similar to the most recent haplotype estimates for the individual in Impute2.

3.2.3 Beagle

Beagle also takes a similar approach to MaCH, but their set of K haplotypes, H, is represented as a graph where the edges of the graph are states of the HMM. The graph complexity increases more slowly than $O(MK^2)$ as K increases.

3.2.4 Fastphase

In fastphase a small number of parameterized states are used are used instead of the set of all known haplotypes. These need to be estimated as the model is fitted, this takes some overhead time, but the method is constant with the number of SNPs.

3.2.5 SHAPEIT

SHAPEIT [26] uses two main improvements to the basic HMM of MaCH. First, it utilizes a graph structure for H, H_g, in which H is partitioned into disjoint segments with J haplotypes per segment and second, a linear coplexity sampler for the graph with linear complexity $O(MJ)$

3.2.6 EAGLE

Eagle [61] attempts to do long range phasing, with IBD-based regions > 4 centimorgan in outbred population data. In short, the idea is to use the outbred population as a big pedigree to identify large haplotype blocks spreading from up to $\sim$12 generations back, use these to call approximate phases, then do two iterations with an HMM to phasing in overlapping 1 glscmg windows by detecting complementary haplotype pairs from the previous step. The paper makes some strong performance claims that didn't seem to work out here. Maybe it is because of the kind of data (simulated, but with a biologically based process dropping down each generation), or perhaps some parameter could have been tweeked. We saw SHAPEIT 2 and duoHMM perform significantly better, 3.1.

3.3 Pedigree-Based

3.3.1 Elston-Stewart

The Elston-Steward method does phasing and imputation at the same time, but separate from the likelihood calculation. It will be discussed further in the imputation section.

3.3.2 Merlin

Merlin [1] is a tool often used for pedigree-based imputation developed by Abecasis et. al. It utilizes the Lander-Green algorithm, which does phasing and imputation at the same time. Further discussion of Merlin will be deferred to the Imputation section.

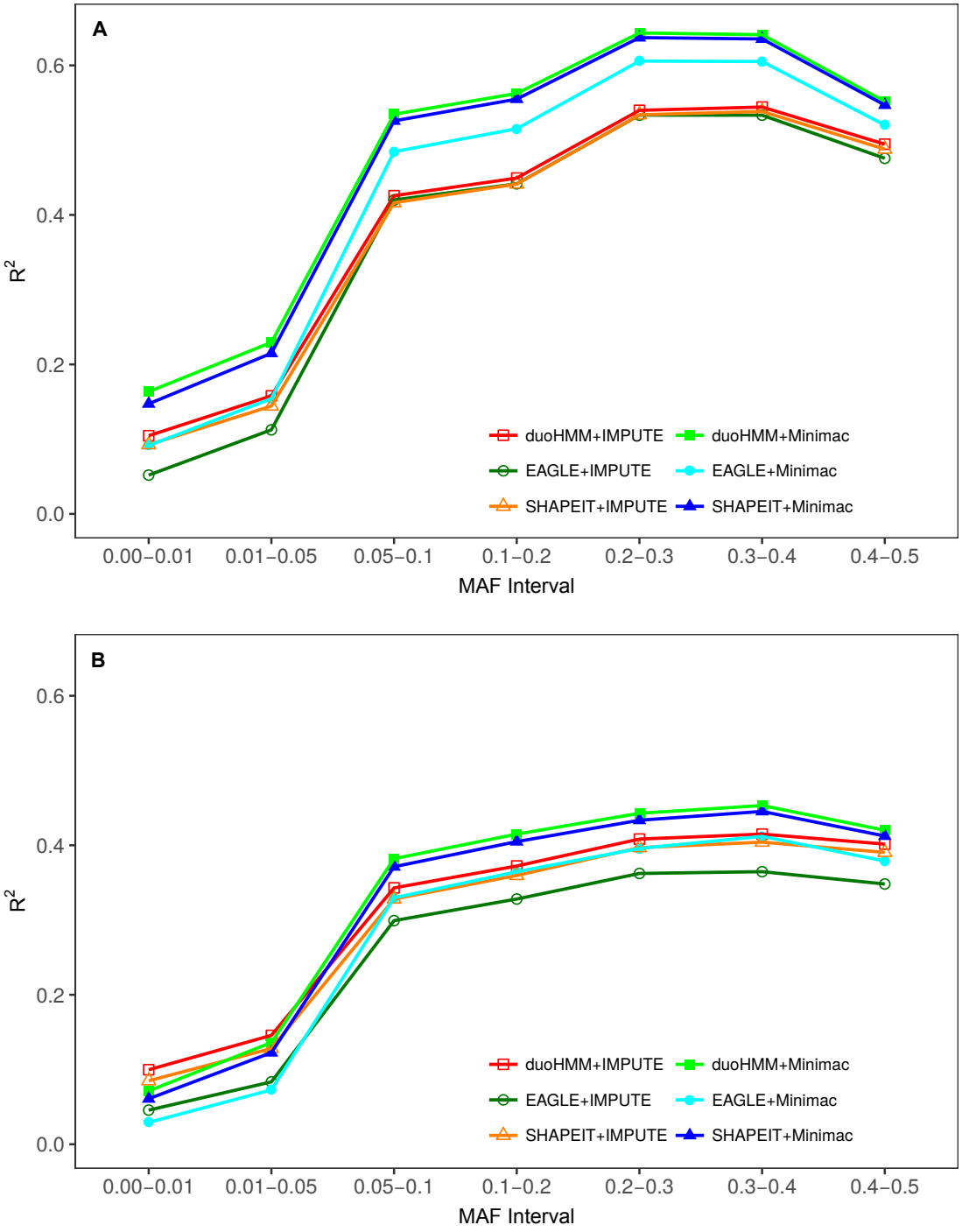

Figure 3.1: Mean correlation R^2 between true and imputed genotypes for different phasing approaches with IMPUTE and Minimac for (A) European and (B) African populations using a random selection strategy for selecting which individuals will be fully sequenced. The first/second of a pair of programs in the key indicates phasing/imputation functions.

3.3.3 gl_auto

The gl_auto program is a part of the MORGAN [103] software package and is one of three programs included in it that measure autozygosity (i.e. IBD). The other two programs lm_auto and lm_pval need to include the affectation status as input in addition to the genotypes and have some different outputs. Here, we are concerned with gl_auto because the output of gl_auto is one of the inputs to GIGI. gl_auto uses an MCMC algorithm to sample possible inheritance vector (IV) realizations. It is not an exact method and by sampling it is able to give results with complexity that is linear with both pedigree size and the number of markers. As we will see in the next section, this is different from other pedigree-based approaches like the Elston-Stewart and Lander-Green algorithms. Simwalk2 [98] is a comparable program. Loki [39] also did something similar, but has been superseded by Morgan, and if one sees references to Pangaea, Pedigree Analysis for Genetics (and Epidemiological Attributes), it is a blanket term that includes Loki, gl_auto (inside MORGAN), and some more specific IBD software. Instead of outputting the probabilites of each IV gl_auto outputs each realization. They are expressed in a compact form indicating chiefly the locations of crossover events, i.e. where an IV, V_i differs from V_{i+1}. The MCMC methods used for these vectors utilize the LM-Sampler in MORGAN, which has been shown to be the only (of the two) program to give accurate results in a computationally practical time for large numbers of dense diallelic markers when compared to Simwalk2 [115].

Chapter 4

Imputation

4.1 Population-Based Imputation

Population-based imputation is based almost entirely on comparing phased haplotypes. Methods like Beagle, Minimac, and Impute, and their subsequent versions, in essence are all variations on using the phasing results of the test populations to search for the best match in the reference population and use that haplotype. This works well for common alleles, but has trouble with imputing rare markers correctly because they are unlikely to appear in population haplotypes, which in turn are expressions of LD. This improves as the size of the reference population increases, but so far pedigree-based imputation methods have been the best for imputing rare alleles correctly and tracing inheritance of rare mutations. It is not clear how far results can be improved through increased size of the reference population. For now, having more rare haplotypes in the reference population increases the chance that a rare haplotype (which also contains more rare alleles) in the subject population can be well matched. At some point, having more rare haplotypes should start to fill the possiblity space, in which case even incorrect haplotypes would be matched because we have a matching reference for any haplotype.

4.2 Elston-Stewart

Elston and Stewart put forward a method for estimating the likelihood of genotypes in pedigree data in 1971 [27]. They consider multiple pedigrees ($1, 2, ..i_0$), and use x_{i_j}, where i_j is a sequence starting with the pedigree i_0 and then followed by the number of each member in each generation from the top of the pedigree to the current individual. Those that are marying in are denoted with y instead and are numbered separately. Therefore, in a simple example of a three generation family, the grandparents are x_1 and y_1, their two children are x_{11} and x_{12}, whom are married to y_{11} and y_{12} respectively. The third generation are x_{111} and x_{112}, children of the couple [x_{11},y_{11}] and x_{121}, child of couple [x_{12},y_{12}].

The genotype for every phenotype, x, is then modeled as $g_u(x)$ for u possible genotypes. $g_u(x)$ is assumed to be a continuous multi-nomial distribution, even normal, though these assumptions are not strictly necessary. $g_u(x)$ is then the conditional probability of x given u. The likelihood of a child having genotype u, given that the parents have the genotypes s and t is p_{stu} and the likelihood that two sets of genotypes come from siblings is:

$$\prod_{i=1}^{n}\sum_{u=1}^{k} p_{stu} g_u(x_i)$$

For n traits or markers and k potential genotypes. Elston and Stewart go on to derive that:

$$\prod_{i0}\sum_{s_0=1}^{k} \psi_{s_0} g_{s_0}(x_{i_0}) \sum_{t_0=1}^{k} \psi_{t_0} g_{t_0}(y_{i_0})$$

Is the likelihood of observing all the original x traits on the parents in all the pedigress of individuals with k genotypes per trait, where ψ_v is the likelihood that an individual has genotype v, i.e. the fraction of the population with v.

They further derive that for an individual in a pedigree with a finite number of genotypes, the probability that the individual w has genotype u conditional on all the other observed data on their relatives is:

$$q(w,u) = L(w,u) / \sum_{u=1}^{k} L(w,u)$$

where $L(w,u)$ is the following:

$$\sum_{s_0=1}^{k} \psi_{s_0} g_{s_0}(x_{i_0}) \sum_{t_0=1}^{k} \psi_{t_0} g_{t_0}(y_{i_0})$$

I.e., the probability of observing the data for a single individual instead of for the entire pedigree or pedigrees. Unfortunately, the number of haplotypes to be considered here is k^W, where W is the number of individuals and k is still the number of possible genotypes at a given location.

4.3 Lander-Green

Take $M_1, ..., M_m$ as m ordered loci and θ_i as the recombination fraction from one loci to the next, and assuming no crossover interference (add to glossary). What is the set of $\theta = (\theta_1, ..., \theta_{m-1})$ that maximizes the likelihood of the data. In the Elston-Stewart algorithm given above, they went as far as calculating probabilities of having a set of genotypes given the data, in the Lander-Green paper, this is then used to find the "genetic map" (phasing + imputation) by then starting from the top of the pedigress and computing the probability distributions for each triple (child and its parents) using Bayes theorem. Then, for each triplet of genotypes corresponding to them, find the expected number of triples having that set of genotypes. There are 2^{2m-2} such sets of triplet genotypes, sum the probabilites for each triplet. Finally, add the expected occurences of crossover patterns with a recombination in each interval. These four steps have an expected complexity of $O(a^{6m})$, where m is still the number of loci and a is the number of alleles. This is massively problematic because, for full genome sequencing, we're potentially considering over 3 billion markers. Even for just the exome m could potentially be over thirty million. Often genetic statistical techniques use the (mostly true) assumption that chromosomes are independent and run on each chromosome separately, even then m will be well into

the millions. Lander and Green proposed to overcome this by using hidden markov chains. Take k to be the number of non-founders in the pedigree. For each locus M_i define an inheritance vector v_i of length 2k. Each position on the inheritance vector is either 0 or 1, 0 when the marker comes from the paternal side and 1 if it comes from the maternal side. θ_i is the recombination fraction, or the chance that there was a crossover, between v_i and v_{i+1}. The probability distributions p_i can then be computed for each v_i given M_i, and the probability values stored as q_i. In the worst case, there are 2^{2k} possible p_i and q_i, and hence it would have that length, but in practice many can be excluded, i.e. uninformative SNPs can be excluded. At the time, they posited that $k <= 20$ may be feasible and potentially practical. The steps for this markov chain are to (1) compute the left and right conditioned probabilities for p_{i+1}^L given p_i^L and p_{i-1}^R given p_i^R respectively.

$$p_{i+1}^L = \frac{[p_i^L T(\theta_i)] * [q_{i+1}]}{[p_i^L T(\theta_i)] \bullet [q_{i+1}]}$$

Where * is for element by element multiplication and $\bullet$ is the dot product. $T(\theta_i)$ is the transition matrix between M_i and M_{i+1}, computed as the Kronecker product of the 2x2 transition matrices representing the 2k transitions. The right conditioned probability is analogous. (2) $T(\theta_i)$ has entries t_{vw} corresponding to the transition from inheritance vector v to vector w. Let $t*_{vw} = d(v, w)t_{vw}$ where d(v,w) is the number of positions at which v and w are different. Let these be entries of a transition matrix $T^*(\theta_i)$, then the number of recombinations expected is:

$$\frac{[p_i^L T^*(\theta_i)] * [p_{i+1}^R]}{[p_i^L T(\theta_i)] \bullet [p_{i+1}^R]}$$

This approach gives a complexity of $O(6mk2^{2k})$, Lander and Green note that the linear complexity with the number of markers will allow it to scale well in that sense, but that it will probably only be practical for pedigrees with 10-25 non-founders. This turns out to be a better tradeoff than in the Elston-Stewart algorithm because the number of markers that must be analyzed has exploded in number while most pedigrees even today are not very big. Note that the likelihoods are also calculated for free in the denominator of p_{i+1}^L, so imputation (genetic reconstruction) and likelihood calculation happen at the same time with the markov chain approach.

4.4 Merlin

Merlin is a tool often used for pedigree-based imputation developed by Abecasis et. al. It utilizes the Lander-Green algorithm and attempts to mitigate the poor (exponential) memory scaling of the Lander-Green algorithm with the size of the pedigree by introducing some novel data structures for this kind of program. In particular, they applied sparse binary trees with bit-wise indexing. While this greatly reduces the absolute memory requirements, the scaling is still exponential. To estimate the amount of memory needed, one can calculate the number of meioses needed as a measure of the complexity of the pedigree. This is two times the number of non-founders in the pedigree minus the number of founders. Merlin is able to work on a 24-bit (24 meioses) pedigree with 2 Gb of memory, but a 30-bit pedigree already requires more than 128 Gb of memory and is not feasible for most individuals, but

still is for many companies and institutions. 35-bit pedigrees are likely out of reach for all but the most extreme, possibly experimental, system configurations. In the Merlin paper there are some ways presented to approximate a solution with reduced resources, but the dominant narrative in the literature is that Merlin fails to work for large pedigrees.

4.5 GIGI

GIGI was originally described in Cheung et al [15], and uses the inheritance vectors from gl_auto (MORGAN) to build IBD graphss. The use of these IBD graphss was expanded in in 2014 by Blue and Cheung et al [6]. For the exact case, gl_auto is subject to some of the same limitations on the number of meioses that they can do exact computation for. They default to exact computation when the number of meioses is less than or equal to 8, and use approximations when the number of meioses is above that. What the gl_auto program provides in particular is a set of meioses indicators that determine the Founder Genome Label (FGL) matrix. The file consists of a number of realizations computed by gl_auto that indicates the locations where crossovers occur. GIGI does SNP by SNP by solving IBD graphss between related individuals. When the graph is unsolvable GIGI will fall back on using the population frequency of the SNP (given as one of the inputs to GIGI). GIGI is the only pedigree-based imputation available for large pedigrees. Overall, there are three approaches for pedigree data, Elston-Stewart, which cannot handle large numbers of markers, Lander-Green, which cannot handle large pedigrees but is what many modern pedigree imputation approaches are based on, and MCMC sampled approximate methods, for which the only implementations of repute are Simwalk2 and the LM-sampler in MORGAN. Among them, only the MORGAN LM-Sampler used by gl_auto has been shown to have accurate results for large numbers of diallelic markes in a feasible time. As mentioned earlier, Simwalk2 is not sufficient in this case [115], and GIGI is the only method utilizing the IBD information from gl_auto for imputation. In short, GIGI+gl_auto is the only well known method for doing pedigree-based imputation on large pedigrees, i.e. with a bit-depth of more than 30 in most cases.

4.6 Other Potential Approaches

4.6.1 Primal

Primal [60] appeared on PLOS One in 2015. They take a different approach to IBD, rather than trying to identify crossover locations, they attempt to de novo match IBD segments as if they were haplotypes via clique graphs. They targetted founder populations in particular and also incorporate LD in their imputation. The software apparently generated a lot of interest, we were asked on more than one occasion to compare with Primal. Unfortunately, the software is not currently available in a working state to compare against. We look forward to doing so if it receives an update.

4.6.2 cnF2freq

This imputation software, cnF2freq [77], is a bit of an unevaluated case. The claim is that they implement the same idea as Merlin, but with automatic overlapping areas of the pedigree. The software is more difficult to install and use than many others. It requires compiling with the Boost libraries and OpenMP. It seems to have undergone a number of modifications to deal with plant genomes in particular. The author was responsive to helping with some compilation issues but development on the code base has been sporadic and I'm not confident in the current code base. I also do not have a great understanding of the expected input and output formats from this particular software. It should be evaluated in the future for correctness against masked data and against some of the stalwarts to see if it merits more widespread use.

4.6.3 Matrix Completion

A rather different approach to imputation, on family or population data, is Matrix Completion. The technique gained widespread recognition when it won the Netflix challenge [51]. It was applied by Kenneth Lange in (cite) and while he showed low error for certain scenarios, the R^2 in the most important cases were not particularly impressive. This implementation [16] is available in the Mendel software package made availabe on Kenneth Lange's UCLA site: `http://software.genetics.ucla.edu/`. As Lange notes, the results from matrix completion are heavily biased towards zero. We performed some exploratory analysis for this approach. Lange indicates that the output from matrix completion be further fit with a gaussian mixture model to fit the three alleles. He also recommends correcting for Mendellian errors and stated in a lecture that these corrections were more effective than trying to incorporate additional constraints [55].

4.7 FamPipe

FamPipe [17] is pipeline for family GWAS studies that includes an imputation step. For a part of their pipeline they perform imputation. They include two options for this imputation, Merlin, and GIGI. For using Merlin on large pedigrees they make use of PedCut [59] in order to split up the pedigree into sub-pedigrees of less than a specified number of bits. They recommend using Merlin for small pedigrees, but GIGI for large ones. This is because splitting up the pedigrees with PedCut leads to worse performance for Merlin. In our pending imputation comparison paper we saw that by choosing sub-pedigrees by hand, of a maximum size that we can compute, and maximizing overlap between sub-pedigrees, we can get better results than by using PedCut. Even then, the results of using Merlin on split pedigrees is not as good as GIGI directly, as shown in figure 4.1. These conclusions still hold for several other automated splitting strategies, including Pedstr [50], and most of the other methods do worse than those two do.

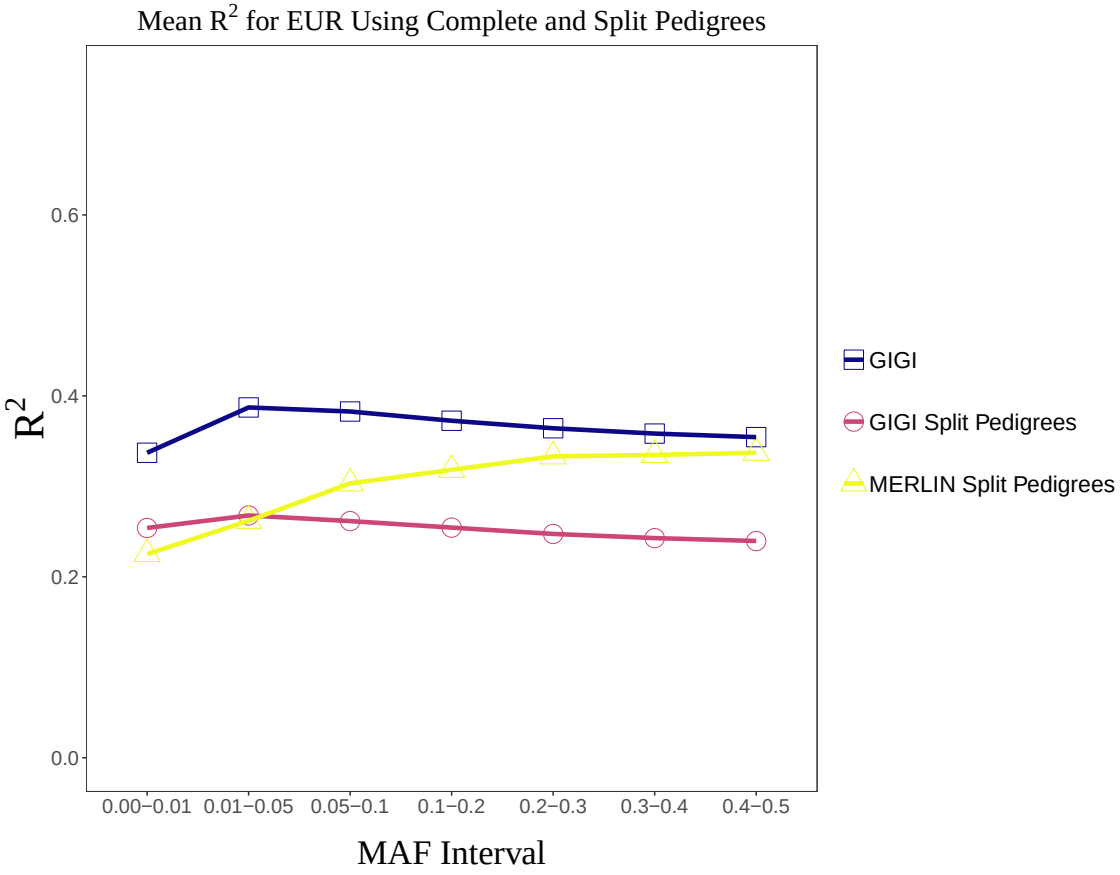

Figure 4.1: GIGI and Merlin. Merlin is evaluated on only the split pedigree because the amount of memory to run it on the full pedigree is not feasible (2018). GIGI is shown running on the split pedigree and also on the full pedigree. While Merlin does better on the sub-pedigrees, GIGI is able to outperform it by utilizing the complete pediree at once.

4.8 PedPop

As we have seen in previous sections, in figures 2.1 and 2.2, population-based imputation (Impute, Minimac, Beagle) does better on common alleles, and pedigree-based imputation (Merlin, GIGI) does better on rare alleles. In Saad et al [86], it was proposed to combine pedigree and population-based methods as PedPop. In figures 2.1 and 2.2 we can see that overall PepPop performs much better than either other imputation approach individually. It suffers very slightly for rare allele imputation relative to GIGI, but does even better than the population-based imputation method on common alleles.

Chapter 5

Data

5.1 Real Data

GIGI-Quick and GIGI were both tested on various real pedigrees of sizes, 16, 36, 92, 154 and 189 and others. These pedigrees reside with our collaborators at the University of Washington and we are unfortunately unable to release them to the public due to privacy concerns.

5.2 Simulated Data

We simulated 20 pedigrees with a total of 1200 subjects. The pedigrees had a median of 47 subjects, minimum of 10, and maximum of 174 with structures extracted from an Alzheimer's disease cohort. This set is simulated 100 times for African and 100 times for European ancestry similar to the LD structure and MAF distribution of the 1000 Genomes Project data with the same procedure as described in Saad and Wijsman 2014 [86].

20,000 haplotypes were simulated for each ancestry for a region on chromosome 22 of GRCh37, 26443384 - 32049917 using HAPGEN (Su et al. 2011) [101]. For the founders of the outbred populations haplotypes were drawn randomly without replacement. Haplotypes were dropped through the generations with a recombination rate of 1%/cM/meiosis assuming 1 cM $\sim$ 1000 kbp. The number of SNPs from the 1000 Genomes data was 8,954 for European and 11,891 for African. 500 from each list were selected at random to be the SNPs selected for sequencing in the GWAS study. Quantitative traits were also simulated, 10 for the null hypothesis and 10 for the hypothesis of association for each SNP and each dataset. These are used to compute the type 1 error rate and statistical power respectively. Further details are available in Ullah et al [108].

The simulated data is available on the website of the Qatar Computing Research Institute's Bioinformatics group, a subset of their Data Analytics group. `https://bioinformatics.qcri.org/IRD` As well as via Zenodo at DOI: `10.5281/zenodo.1485557`

Chapter 6

Improvements to GIGI

6.1 GIGI-Quick

6.1.1 Approach

GIGI-Quick is a set of Bash scripts and C++ utilities developed for running GIGI in parallel to obtain the results more quickly. GIGI-Quick requires a recent version of Bash for full compatility (Bash 4.3 or later) because 'wait -n' was added in 4.3. Older versions may be used if one does not intend to utilize the -q option. Compiling the C++ utilites requires a reasonably recent C++ compiler and CMake if you intend to use the included compilation script. GIGI-Quick takes your input parameters and writes a GIGI input file for the number of chunks one specified for GIGI-Quick to split into. It then splits the appropriate input files into that number of chunks and runs GIGI on each chunk while tracking the process. When all the chunks have completed running GIGI-Quick will combine them into a single set of outputs as if one had just run GIGI on the input directly.

GIGI-Quick supports a flexible set of command line options while retaining the exact same parameter file for GIGI-Quick as for GIGI. Other flags define other parameters, such as the output file names, number of CPUs to utilize, and requested memory. Internally, GIGI-Quick splits three input files, I4, I5, and I6, into C non-overlapping chunks. By default, this number is the number of cores - 1 times the number of threads per core. The user is able to override the number of threads to be used. After splitting, GIGI-Quick prepares the appropriate parameter files to run GIGI on the C chunks. Finally, it merges all output chunks, column-wise for O1, O2, and O3, and row-wise for O4. GIGI-Quick is a combination of two utilities written in C++ and several Bash scripts. The first C++ program is used to split the inputs and the second program is used to merge the imputation outputs. The Bash scripts (1) parse the user's parameters (such as the number of splits, GIGI original parameter file, and the long format flag), (2) call the splitting program, (3) run GIGI in parallel, and (4) call the merging program that will give one set of outputs, similar to what GIGI gives without splitting. Independent logs, which show memory usage, time, and other information, are kept for each run with GIGI as well as for the split and merge programs. GIGI-Quick implements two different approaches to memory constrained scenarios, one queue-based and the other utilizing cgroups, a feature of the Linux kernel for controlling resource usage. GIGI-Quick also allows the user to easily perform imputation on a specified region of interest rather than

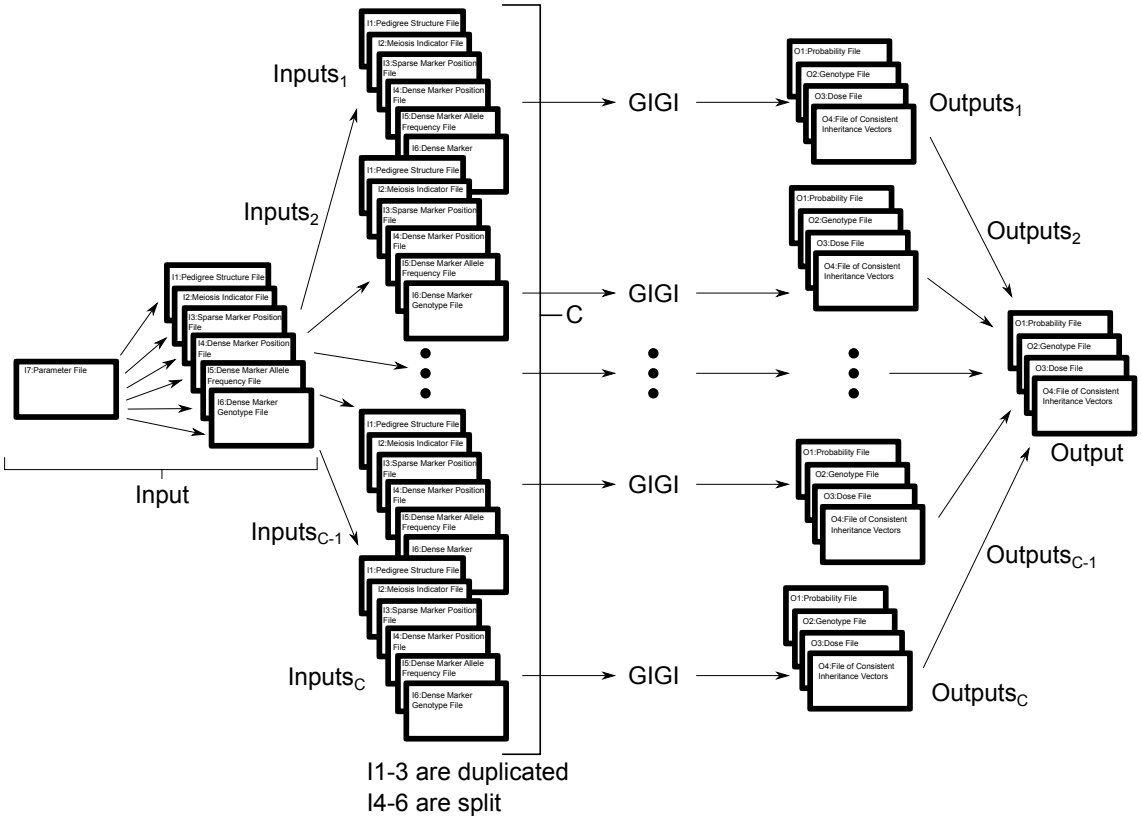

Figure 6.1: GIGI-Quick flow of inputs and outputs.

an entire chromosome by giving the start and end positions of the region.

6.1.2 Availability

GIGI-Quick can be cloned via git from `https://cse-git.qcri.org/Imputation/GIGI-Quick.git` and can also be downloaded from a browser by going to `https://cse-git.qcri.org/Imputation/GIGI-Quick/tree/master` and downloading it in a compressed format.

6.1.3 Installing and Compiling

Once you have the files, most users won't need to do anything else to use GIGI. There are executables compiled on Ubuntu 64 bit Linux for 64 bit and 32 bit (via multilib) x86 systems. GIGI-Quick will automatically choose which of these to run. We recommend using these unless your system has a different architecture (e.g. PowerPC, ARM). When GIGI-Quick runs, if there are locally compiled versions of the binaries then GIGI-Quick will use those, it will check for them in the following locations: ./GIGI/GIGI, ./MERGE/gigimerge, ./SPLIT/gigisplit.

We use cmake to create make files for the architecture being compiled on, to use that method one will need a reasonably recent cmake installed. This approach should be compiler and architecture agnostic. To do this, one need only run the included make.sh script:

./make.sh

This should create the make file then compile all three binaries. It will write a log file in ./make.log. If the cmake method is not working on your system, you can compile directly with your compiler, we give an example with g++ from the gnu gcc:

```
cd ./SPLIT/
g++ -O2 GIGISplit.cpp -o gigisplit
cd ../MERGE/
g++ -O2 GIGIMerge.cpp -o gigimerge
cd ../GIGI/src/GIGI_v1.06.1
g++ -O2 GIGI.cpp -o ../../GIGI
```

Extra Integration

The folder structure of GIGI-Quick should not be separated, GIGI-Quick depends on relative paths to locate the scripts and executables included other than run_GIGI.

As an Unprivileged User

If you like you can now add GIGI-Quick to your path, the examples assume that you have, you can do this by adding the following to your .bashrc (located in your home folder)

```
export PATH=$PATH:/path/to/folder/where/you/put/run_GIGI
```

Then source your .bashrc to apply the changes right away

```
source ~/.bashrc
```

As a Root/Sudo User

To add run_GIGI to the path system-wide for all users you can create a symlink in /usr/bin pointing to the run_GIGI script:

```
ln -s /path/to/run_GIGI/script /usr/bin/run_GIGI
```

6.1.4 Running GIGI-Quick

Note: The parameter file is the same as you would use for GIGI normally, but if you are using the long format, then pass the "-l" option The examples in shown below use the file "param-v1_06.txt" because it is included in the repository and can be run by simply cutting and pasting the example line.

run_GIGI parameter_file -o [OUTPUT FOLDER] -n [RUN NAME] -t [THREADS] -m [MEMORY IN MB] [-l] [-v] -q [THREADS] -r [START] [END] [-V] [-h]

-o [OUTPUT FOLDER] : This is the path to use for the outputs from the run_GIGI scripts, including temporary files.
-n [RUN NAME] : This is a path relative to the [OUTPUT FOLDER] to use to keep the outputs from more than one run of run_GIGI separated.
-t [THREADS] : The number of threads to use for run_GIGI, and also the number of chunks to split the input into.
-m [MEMORY IN MB] : The amount of RAM that run_GIGI will restrict its use to, not yet implemented
-l : Specifies that the input is in the long format.
-V : Verbose mode, output from run_GIGI is much quieter now, you can see much more of what it is doing and what variables are set to at various stages with -V.
-v : Display the version of GIGI-Quick and exit.
-h : Display this help text.
-r [START] [END] : Run on only a selected region, starting at start and ending at end, this region will be selected before any further splitting.
-q [THREADS] : Run in queued mode, this mode will run up to THREADS instances of GIGI at a time and will attempt to keep the total amount of memory being used less than [MEMORY IN MB] using an estimate of the amount of memory GIGI may need. If -m [MEMORY IN MB] wasn't given, then it will use the amount of memory available as shown by 'free.' For older kernels this isn't shown and we use an estimate that is no longer accurate for modern systems (amount free + amount of buff/cache). `https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=34e431b0ae398fc54ea69ff85ec700722c9da773`
Also, -t is ignored when -q is given.
-e [MEMORY IN MB] : Manual estimate of how much memory GIGI will need for queued mode in case the calculated estimate is too inaccurate

Examples:

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt #Output in the current folder with no run name identifying subfolder, threads and memory determined automatically

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -o ./OUTPUTS -n test_run #Output in ./OUTPUTS/test_run

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -o ./OUTPUTS -n test_run -V #Output in ./OUTPUTS/test_run, verbose mode (print more detailed information)

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -o ./OUTPUTS -n test_run -l #Output in ./OUTPUTS/test_run for a parameter file in the long format, do not cut and paste this one because the included param-v1_06.txt is not in the long format

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -t 2 #Limit to only 2 threads (and hence two chunks)

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -m 1000 #Limit memory use to 1 GB, please read the section on memory and cgroups

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -lmt 1000 2 #Limit memory use to 1 GB, please read the section on memory and cgroups, and threads to 2 with input in the long format, do not cut and paste this one because the included param-v1_06.txt is NOT in the long format

./run_GIGI INPUTS/Sample_Input/param-v1_06.txt -o RUN_FOLDER/ -n test_run -m 20 -q 3 -V -r 3 70 #Output in ./RUN_FOLDER/test_run, limit memory to 20 MB, use the queued mode with up to 3 threads at a time, and run on only the region from 3 to 70, note: the memory estimated as needed in queued mode does not account for the restricted region

If there is a problem that makes GIGI stop before completion, then the output files are left as they are in order to allow users to rerun only failed portions as needed. If you are unsure where the failure occurred, then the safest approach will be to remove the intermediate files before rerunning (e.g. rm -R [OUTPUT FOLDER]/[RUN NAME]), use rm with caution as always e.g. if the 2nd example failed, I would "rm -R ./OUTPUTS/test_run" before rerunning.

The -n option is largely redundant, as it is equivalent to using the -o option with a longer path giving the subfolder, e.g.

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -o ./OUTPUTS -n test_run

is equivalent to:

./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -o ./OUTPUTS/test_run

The inclusion of -n is mostly a semantic convenience.

6.1.5 Logging

With the addition of -v and cleanup of output, you may notice that even with -v you don't see the output of split, gigi, and merge any longer. These are now written to their own individual log files in the output directory/run subdirectory.

e.g. ./run_GIGI ./INPUTS/Sample_Input/param-v1_06.txt -o ./OUTPUTS -n test_run will have logs in ./OUTPUTS/test_run/LOGS

6.1.6 Miscellaneous

Memory and cgroups:

We handle memory restrictions using cgroups. After looking at a number of different memory limiting mechanisms we saw this as the best solution, unfortunately it has some caveats. One is that root/sudo access is required to create the

initial cgroup. If you are on a shared machine then we encourage you to discuss this with your system administrator if you intend to use the cgroups. For most shared clusters, we encourage you to use the built in memory limiting mechanims of your submission system (e.g. qsub, SLURM, Torque) instead of limiting it through run_GIGI, most of these also themselves make use of cgroups (e.g. `https://slurm.schedmd.com/cgroups.html` and HTCondor `http://help.uis.cam.ac.uk/supporting-research/research-support/camgrid/camgrid/technical3/cgroups`). If you are using this on your own system where you have root/sudo access, then you will need to make sure that your cgroups are set up and that you have your equivalent of the libcgroup library installed for the cgcreate and cgexec commands for your distribution. If you have a very old (e.g. maybe 7+ years old) kernel, then you may need to install a newer kernel that has cgroups (they are part of the Linux kernel technically).

Here is a list of common distributions and links to help/documentation on cgroups

Redhat: `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Managemen_Guide/ch-Using_Control_Groups.html`

Arch: `https://wiki.archlinux.org/index.php/cgroups` you may note that libcgroup is an AUR package, to install such packages: `https://wiki.archlinux.org/index.php/Arch_User_Repository`

Debian/Ubuntu: `https://www.devinhoward.ca/technology/2015/feb/implementing-cgroups-ubuntu-or-debian`

Fedora: `https://docs.fedoraproject.org/en-US/Fedora/17/html/Resource_Management_Guide/ch-Using_Control_Groups.html`

Fedora: `https://docs.fedoraproject.org/en-US/Fedora/15/html/Resource_Management_Guide/sec-Creating_Cgroups.html`

OpenSuSE: `https://www.suse.com/documentation/opensuse114/book_tuning/data/sec_tuning_cgroups_usage.html`

Once you have a functional cgcreate command to create cgroups, you can make them permanent (unfortunately in different syntax) by editing /etc/cgconfig.conf on Linux distributions using systemd (most of them).

Ubuntu: `https://askubuntu.com/questions/836469/install-cgconfig-in-ubuntu-16-04`

Already covered in many of the other links above.

If your distro isn't covered, it is still worth looking at the above guides, most things will be similar in your distro though they may not be exactly the same (e.g. package names could be different, package manager, etc..).

Here is some distribution agnostic information on cgroups: `http://man7.org/linux/man-pages/man7/cgroups.7.html`

`https://www.kernel.org/doc/Documentation/cgroup-v1/`

cgroups will eventually be replaced with cgroups2, but most of their controllers are not yet functional: `https://www.kernel.org/doc/Documentation/cgroup-v2.txt`

Technically you can create the cgroup/s we need with mount and mkdir commands, but we ourselves depend on cgcreate and cgexec in code, of course you could create cgcreate and cgexec scripts and add them to your path instead of using the programs in cgroup-tools. We wouldn't recommend that route though.

Example:

Essentially, the goal here is to get a user writable cgroup setup that run_GIGI (running as your user) can make use of to create its own subcgroup.

On Ubuntu in BASH you can do this as follows:

First we install cgroup-tools to get cgcreate and cgexec, etc...

sudo apt-get install cgroup-tools

Then we create a cgroup that your user has access to:

sudo cgcreate -a $USER -g memory,cpu:user_cgroup

We can see that it was create by checking the contents of /sys/fs/cgroup/memory and/or /sys/fs/cgroup/cpu

They should both now have a folder user_cgroup that your user has write permissions to the contents of

ls -la /sys/fs/cgroup/memory/user_cgroup

ls -la /sys/fs/cgroup/cpu/user_cgroup

When run as your user normally with -m, run_GIGI will make its own subcgroup of this cgroup (do not run run_GIGI with sudo) These are not persistent cgroups (that is, they will disappear on reboot).

To make persistent ones, please see the distribution documentation above, for most this involves editing a configuration file /etc/cgconfig.conf

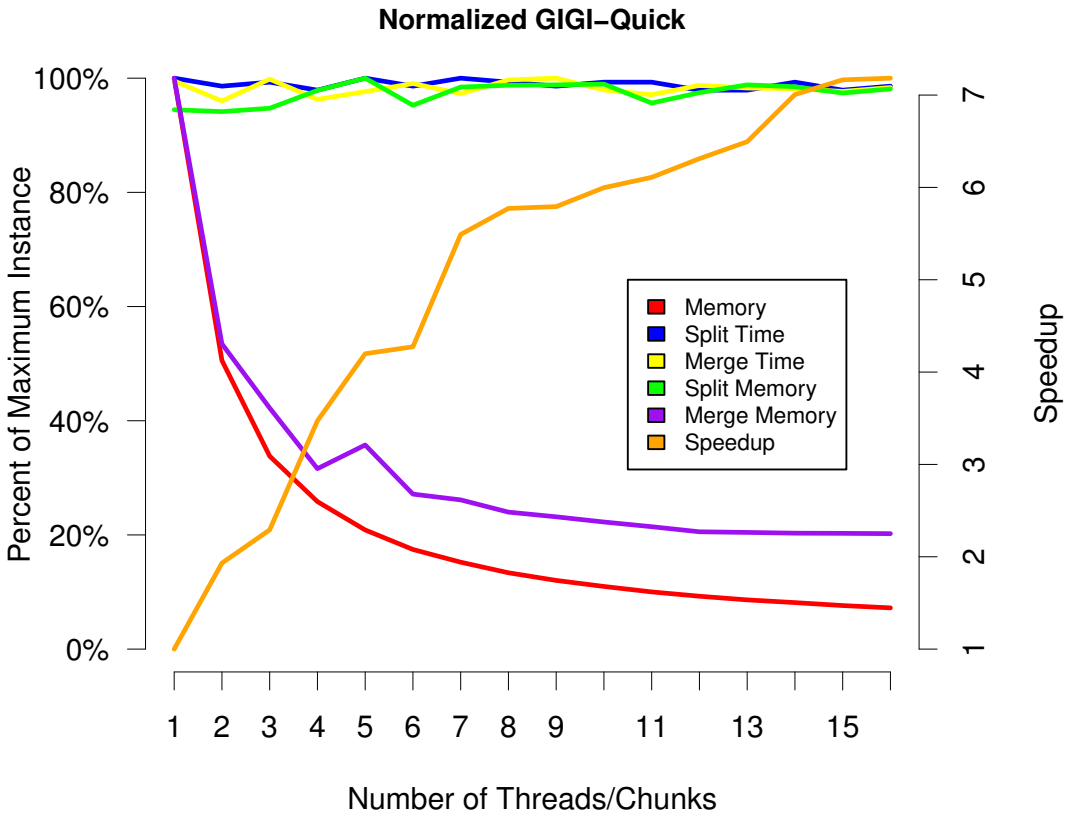

Figure 6.2: GIGI-Quick when run on a pedigree of 189 individuals for chromosome 22 split into different numbers of chunks on a Ryzen 1800X. This is expressed as a percentage of the maximum to show scaling.

6.1.7 GIGI input and output files

Assume a goal of imputation on a pedigree of size N, where we sequence S subjects on a dense marker panel of M markers. Assume that the IBD was computed using a sparse marker panel of P informative markers. GIGI requires seven input files. I1: Pedigree structure file (∼N rows× 5), I2: Meiosis indicator file, containing the IBD information, I3: Sparse marker position file used to compute the IBD ($P \times 1$), I4: Dense marker position file ($M \times 1$), I5: Dense marker allele frequency file for the pair of allele frequencies ($M \times 2$), I6: Dense marker genotype file ($S \times M$), and I7: A parameter file that specifies all the aforementioned files and other flags for GIGI. GIGI outputs four files. O1: A probability file, for each of three possible genotypes ($N \times (3M)$), O2: A genotype file, containing the best guess genotypes ($N \times (2M)$), O3: A dose file, containing the estimation of the number of alleles, ($N \times M$), and O4: A file of consistent inheritance vectors (M rows).

6.1.8 Performance

GIGI-Quick was tested on the new Ryzen platform 1800X CPU, a general purpose amd64 processor. We have tested it with a wide variety of pedigrees on both real and simulated sequence data, from just a few individuals up to 189, which is a very large pedigree. We measured several statistics while testing GIGI-Quick: the

amount of memory and time used for the input splitting, output merging, and by a single instance of GIGI that was run by GIGI-Quick. We report the maximum memory and time of all instances.
Figure 5.2 shows how GIGI-Quick scales with the number of chunks (which is also the number of threads used) on a single consumer CPU. Note that the processor has 8 cores, so using 9-16 threads shows gains from AMD's simultaneous multithreading (SMT), which is similar to Intel's hyperthreading, but is not expected to have large gains in speedup, 1.1x-1.2x going from 8 to 16 cores. The time and memory to run GIGI-Quick is directly inversely proportional to the number of chunks (i.e. 8 chunks uses 1/8th of the memory and requires 1/8th of the computation) as GIGI-Quick is data parallel but, running these threads concurrently introduces a level of contention for L1-L3 cache and bandwidth on the main memory bus that bottlenecks CPU operations. If you have processors available on separate cluster nodes, then the scaling is ideal (i.e. splitting in 8 chunks gives 8x speedup) but you will have some additional overhead to transfer the data to the nodes and it is not the most efficient cluster utilization, but does give the fastest time to completion. More details are given in the supplementary material. The time and memory to split the files remains constant. The time to merge the files remains constant as well but the memory is inversely proportional because the buffer is written to disk after reading in each individual output file. It is noteworthy that the time and memory to run GIGI are much more than for the splitting and merging steps, which is not apparent from this plot. In our test runs of splitting input files into up to 16 chunks on the Ryzen platform, running all instances of GIGI accounted for more than 98.5 % of the clock time and more than 99.5% of the total memory use.

We modeled the memory use for GIGI with a linear regression model (memory= $\alpha + \beta_1 NM + \beta_2 S$) and found that we get good predictions of memory usage. The values of the coefficients are, α: -20490, β_1: 0.05799, and β_2: 6164.00, all in kilobytes. Using this information, we were able to add an additional running mode to GIGI-Quick for users to run in an automatically queued manner for machines with less memory. Given an amount of memory and a desired number of splits (C), GIGI can queue and run these jobs within the memory envelope in most cases by including an additional buffer or five standard deviations over the predicted amount.

In this study, GIGI-Quick was tested on a 189-subject pedigree, which is very large. GIGI-Quick also works on larger pedigrees, such as the Framingham ones. The computational performance of GIGI-Quick conducted on the largest of the Framingham pedigrees (˜210 individuals) is expected to be comparable to what we observed in our data because our pedigree is much deeper and is likely to have more meiosis. Prior to GIGI [15] such pedigrees were a significant problem and novel pedigree splitting was proposed just to run Merlin on Framingham pedigrees [13] GIGI-Quick offers a number of useful features for running GIGI that address both high and low end use cases. In scenarios where there are many threads on a node they can all be utilized to get GIGI results more quickly. On the other end of the spectrum, it is now possible using GIGI-Quick to run GIGI on machines that previously would not have enough memory to do so. The design of GIGI-Quick can also be built upon easily to adjust GIGI-Quick to particular high-performance computing queue environments like SLURM, Torque, or PBS.

6.2 GIGI2

6.2.1 Performance Improvements and New Features

Multithreading

GIGI2 offers a number of performance improvements over GIGI. GIGI2 allows multithreaded computation. By default, GIGI2 uses the number of available hardware threads - 1. An optional flag `--threads` in the parameter file allows the user to set up the number of threads. If the user specifies the number of threads greater than or equal to the number of hardware threads available, the default number of threads is used to achieve optimal computation performance and hardware utilization.

Memory Usage and Management

Compared to GIGI, GIGI2 uses much less memory allowing it to run on computers with limited memory resources. The amount of memory used by GIGI2 is reduced by processing markers in batches. The size of a batch is the number of SNPs loaded in the memory for processing. The batch size can be set in the parameter file (`--mbuffer`). The default value is 10,000. Reducing the batch size to a very small value will have impact on the performance as it will increase the number of time consuming disk IO operations.

Algorithmic Improvements

A major performance improvement in GIGI2 is accomplished by modifying the data layout in memory and by using compressed edge IBD graphs for genotype imputation.

The data layout in the memory has been optimized to provide a better memory cache performance. Moreover, the layout facilitates a data access pattern that improves data pre-fetching from the memory at hardware level.

Thomson used IBD graphs to calculate likelihoods from the observed individuals and assigning genotypes to founder genome labels (FGLs) of the pedigree to find the missing genotypes of unobserved individuals [104]. An IBD graph $G(V, E)$ consists of a set of nodes V representing FGLs to be resolved and a set of edges E representing genotypes to be assigned to the given FGLs. An edge exists between two FGLs if a genotype exists for the FGLs. For details see Thompson et al 2011 [104].

In GIGI2, the each edge is represented by a 32-bit variable with bits corresponding to different alleles instead of a list of alleles in previous implementation. A bit is set (having value 1) if an allele is present and reset (having a value 0) if the allele is absent in a given genotype. For example, there are two alleles A and T for a marker position corresponding to least significant two bits respectively. The genotypes AA, AT and TT will be represented as 00000000 00000000 00000000 00000001, 00000000 00000000 00000000 00000011 and 00000000 00000000 00000000 00000010 respectively. This encoding uses of a single variable for multiple genotype representation enabling compression of genotypes. The major advantage of this encoding is reduced memory usage along with reduced amount of data transfer between process and memory, thus reducing the memory bandwidth usage.

The tests for checking of homozygosity and heterozygosity of a genotype `x` in C++ are defined as:

```
#define isHomozygous(x) ((x != 0) && (x == (x & -x)))
#define isHeterozygous(x) (x != (x & -x))
```

where, `&` is bitwise logical and operation. The above mentioned test only uses bitwise operations, which do not use iterating over any list thus avoiding additional memory accesses. These logical operations require only a single clock cycle, therefore improving the computational throughput.

Imputing a Genomic Region of Interest

GIGI2 supports imputation of a set of SNPs in a genomic region specified by the start and end positions of the region. This feature is important when one wants to focus on a linkage analysis region instead of the whole chromosome. The genomic region of interest can be specified by an optional flag `--drange` in the parameter file to specify the starting and ending location of dense markers to be imputed.

Reproducibility of Results

GIGI and GIGI2 imputation algorithm involves the use of a random number generator. When performing imputation in parallel, reproducibility of results may be challenging due to the order of SNP processing. Unlike GIGI-Quick, GIGI2 always generates the same imputation results for a given seed of the random number generator, which can be specified in the parameter file by an optional flag `--seed`. The results generated by GIGI2 would be different from GIGI and GIGI-Quick due to different sequence of numbers generated by random number generator, which is also expected by changing the random number generator seed in GIGI and GIGI-Quick.

Output Logging

GIGI2 generates a log file for each run containing the details of all operations along with their timestamps. The log file contains all the messages printed on the screen during program execution.

GIGI2 is written in C++. GIGI2 inputs are the same as GIGI except that GIGI handles wide (rows are subjects) and long (rows are SNPs) formats, while GIGI2 only uses the long format. The long format is preferred whenever the number of SNPs is much larger than the number of subjects. Nonetheless, we provide a utility *Wide2Long* to convert the file format for backward compatibility. Another difference is the parameter file. The GIGI2 parameter file is more flexible: the flags can be ordered in any way, the user can specify the output file names, and many important flags (representing new features) are introduced. Note that GIGI2's output files are exactly the same as GIGI.

GIGI2 offers a number of performance improvements over GIGI: (1) *Multi-threading*: GIGI2 allows multi-threaded computation; (2) *Memory Usage and Management*: Compared to GIGI, GIGI2 uses much less memory by processing markers in batches; (3) *Algorithm*: A major performance improvement in GIGI2 is accomplished by modifying the data layout in memory and by using compressed edge IBD graphs for genotype imputation, which provides better memory cache performance and high computation throughput, note that the differences between GIGI and GIGI2 are functionally equivalent but more efficient and the compressed data structure is a losslessly compressed data representation. The outputs should be identical to

that of GIGI excepting differences in which random numbers are used by GIGI vs. GIGI2 due to threading, i.e. we would expect a GIGI run and a GIGI2 run to have outputs as close to each other as running GIGI with two different seeds. We have compared the output between GIGI and GIGI2 for chromosomes 2 and 22 of 189 member pedigree, and find that the mean Pearson correlation is > 0.9999 and ¿ 0.9998 respectively.; (4) *Imputing a Genomic Region of Interest*: GIGI2 supports imputation of a set of SNPs in a genomic region specified by the start and end positions of the region, which is important when one wants to focus on a linkage analysis region instead of the whole chromosome; (5) *Reproducibility of Results*: GIGI and GIGI2 imputation algorithm involves the use of random number generator (RNG). Unlike GIGI-Quick, GIGI2 always generates the same imputation results for a given RNG seed; and (6) *Output Logging*: GIGI2 generates a log file for each run containing timestamped details of all operations. More details are available in the online Supplementary Material file and at `https://cse-git.qcri.org/eullah/GIGI2`.

6.2.2 Results

We extensively tested GIGI2 on several pedigrees and two chromosomes (i.e., 2 and 22, one of the largest and smallest, respectively) on two hardware platforms: a general purpose amd64 processor (Ryzen 1800X) and a Cray XC40-AC supercomputer with Intel Xeon Haswell cores. These were used for comparisons between GIGI2, GIGI-Quick, and GIGI with the same input data. These comparisons were quantified in terms of speedup, runtime, and memory usage. The speedup is the ratio of elapsed time required for GIGI (or GIGI-Quick) to that required by GIGI2.
The performance of GIGI2 is shown in Fig. 6.3a-e with respect to: (a) the pedigree size, (b) the number of IV realizations, (c) the total number of SNPs to be imputed, (d) the number of threads used, and (e) the batch size of SNPs to be processed. We compared the performance of GIGI2 to GIGI-Quick and GIGI for various numbers of threads (Fig. 6.3f). The results can be summarized as follows: GIGI2 scales linearly (directly proportional) with (a) the pedigree size, (b) the number of IV realizations, (c) the number of SNPs to be imputed, and (d) the number of threads. The memory usage increases with batch size while the runtime slightly decreases because of the increase in the number of thread synchronizations. Thread synchronization is required to save the results after processing each batch (Fig. 6.3e). Finally, we compared the performance of GIGI2 and GIGI-Quick, which uses GIGI, for imputation. As shown in Figure 6.3f, on one thread, the speedup of GIGI2 is 28x compared to GIGI. On 8 threads, the speedup of GIGI2 reaches 131x (Note that GIGI cannot be run on multiple threads). The memory usage of GIGI2 is 35x (one thread) and 32x (8 threads) less than GIGI's memory usage. Compared to GIGI-Quick, which can be run on multiple threads, the speedup of GIGI2 was always ¿ 22x and memory usage is at least 34x less.

6.2.3 Availability

Git

GIGI2 is available at `https://cse-git.qcri.org/eullah/GIGI2`. It is free and open source.

Server

GIGI2 available as a webserver at `https://imputation.qcri.org/`, based on the same framework as the Michigan Imputation Server [25].

Docker

GIGI2 is made available as a docker container at `https://hub.docker.com/r/kkunji/gigi2/`, which can make results more reproducible and allow integration into Kubernetes clusters or other cloud computing clusters for anyone that needs to run it at massive scale. Further instructions for using the docker container are available at the given link above.

Snap

GIGI2 is made available as a snap/snapcraft package. Snap is a linux packaging format, analogous to Debian packages (.deb) or Redhat packages (.rpm), etc..., but is more OS agnostic and has support for push updates. This makes it possible on some more recent Linux distributions (e.g. Ubuntu 18.04) to install GIGI2 with a single line, for instance: 'snap install gigi2' Snaps are quite strictly confined by default. When installing the snap as above it is likely that the snap will only be able to write data to your home directory. Snaps use interfaces of plugs and slots to determine the permissions for the app. To view the permissions for an app, e.g. gigi2, you can run 'snap interfaces gigi2' and the output should show two plugs and which slots they are connected to. These two plugs from gigi2 are gigi2:home and gigi2:removable-media. The first allows reading and writing from the user's home directory, the second for reading and writing to drives mounted to /mnt and /media. It is likely that gigi2:home is already connected to the system slot :home. The plug gigi2:removable-media will not be connected by default, to connect this plug one can run 'snap connect gigi2:removable-media :removable-media' and then /mnt and /media drives should be read and writable. Some situations require less confinement as snaps do not yet support fine grained permissions for them. One such situation is when a user's home is located on an NFS drive or generally if they need to write to a location not currently supported with a snap interface. In these cases the GIGI2 snap can be installed without confinement via 'snap install –devmode gigi2' and will be able to read and write anywhere the user normally can as per standard unix file permissions and/or a domain controller. Note that following standard snap naming conventions the executable is called with 'gigi2' instead of GIGI2 and the utility for converting genotype files is called with 'gigi2.wide2long' instead of 'Wide2Long.'

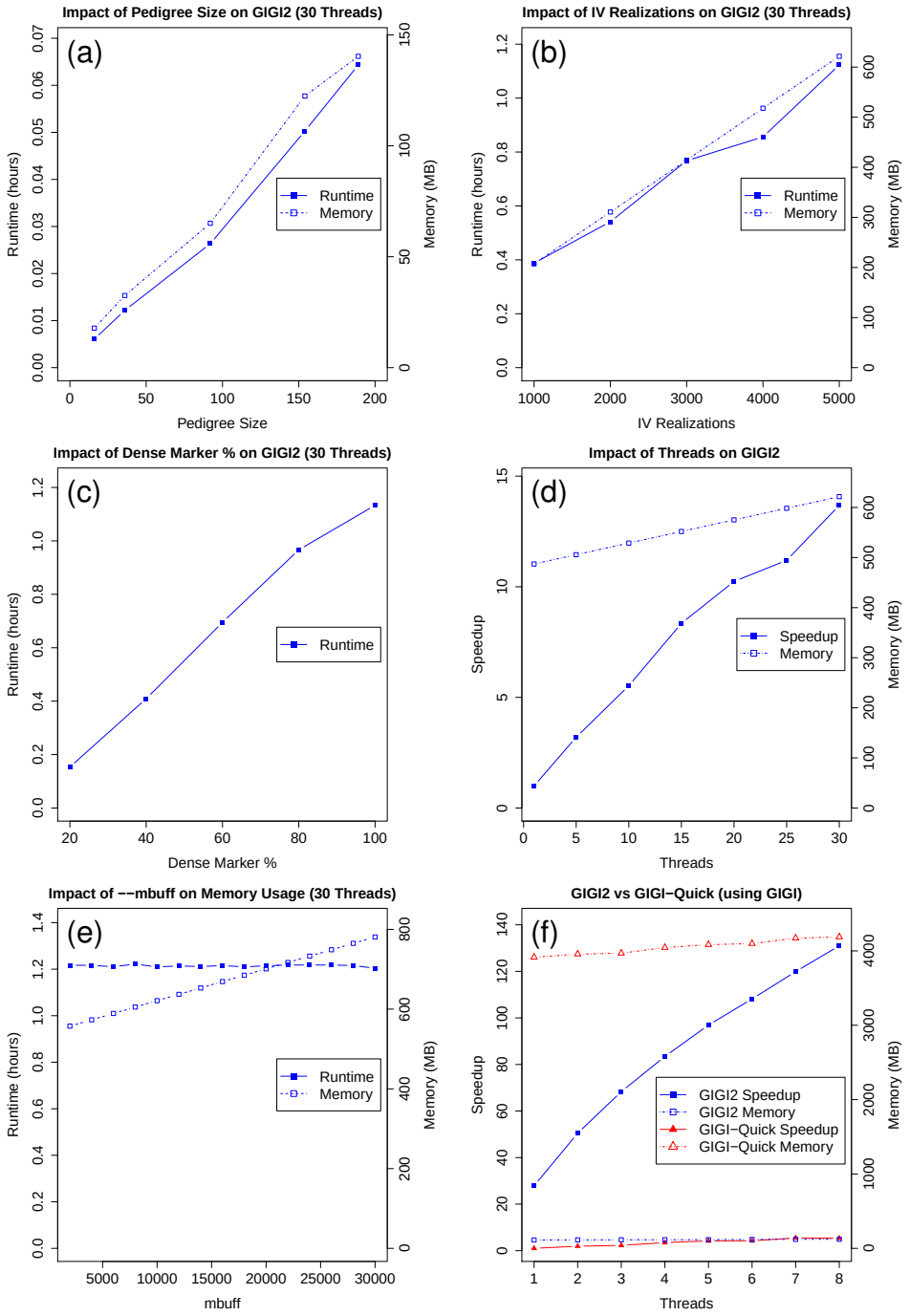

Figure 6.3: Performance of GIGI2 and comparison with GIGI and GIGI-Quick: (a) GIGI2 runtime and memory usage by pedigree size (i.e., 16, 36, 92, 152, 189) on 30 threads, (b) GIGI2 runtime and memory for various IV realizations on 30 threads, (c) GIGI2 runtime vs. percent of dense markers on chromosome 2 (2,402,346 SNPs) imputed on 30 threads, (d) GIGI2 speedup and memory for different numbers of threads, (e) GIGI2 runtime and memory usage for different batch sizes on 30 threads, (f) GIGI2 and GIGI-Quick speedup and memory relative to GIGI for different numbers of threads. The computational characteristics of each figure: chromosome 2 was used for all figures except for figure (a) and (f) where chromosome 22 was used; The 189-member pedigree was used for all figures except figure (a); 5000 IV realizations were used for all figures except figure (a) and (f); a Cray supercomputer was used for figures (a-e) and a Ryzen 1800X processor for figure (f).

Chapter 7

Conclusion

Our new imputation programs, GIGI-Quick and GIGI2, successfully impute genotypes in significantly less time than the original implementation, GIGI, by itself. Our most recent contribution, GIGI2, manages to perform much better than our slightly older GIGI-Quick approach. GIGI-Quick introduced a number of new features utilizing constraints available on most unix-based systems. GIGI2 is able to impute genotypes for very large pedigrees and millions of SNPs at least 25 times faster than GIGI. GIGI2 is now, by far, the fastest and most efficient available family-based imputation tool. GIGI2 was able to impute a pedigree of 189 members on chromosome 2 ($2,402,346$ SNPs) in 10.11 hours on a single thread compared to 17 days needed by GIGI. Moreover, on 8 threads, GIGI2 required 1.5 hours compared to 2.4 days needed by GIGI-Quick. GIGI2 also offers new features and functionality over the original GIGI. For instance, imputing for only a selected range of markers. Finally, GIGI-Quick and GIGI can actually be used together. By replacing calls to the original GIGI in GIGI-Quick with calls to GIGI2, GIGI-Quick can then be used to run GIGI2 in a distributed fashion on large supercomputing clusters. This option requires some setup though and is likely only of interest to those that need to run very large pedigrees many many times over.

Bibliography

[1] Gonçalo R Abecasis, Stacey S Cherny, William O Cookson, and Lon R Cardon. Merlin—rapid analysis of dense genetic maps using sparse gene flow trees. *Nature genetics*, 30(1):97, 2001.

[2] Nada A Al-Tassan, Nicola Whiffin, Fay J Hosking, Claire Palles, Susan M Farrington, Sara E Dobbins, Rebecca Harris, Maggie Gorman, Albert Tenesa, Brian F Meyer, et al. A new gwas and meta-analysis with 1000genomes imputation identifies novel risk variants for colorectal cancer. *Scientific reports*, 5:10442, 2015.

[3] Vikas Bansal, Ondrej Libiger, Ali Torkamani, and Nicholas J Schork. Statistical analysis strategies for association studies involving rare variants. *Nature Reviews Genetics*, 11(11):773, 2010.

[4] Markus Bauer, Gunnar W Klau, and Knut Reinert. Accurate multiple sequence-structure alignment of RNA sequences using combinatorial optimization. *BMC Bioinformatics*, 8:271, 2007.

[5] Gaurav Bhatia, Nick J Patterson, Sriram Sankararaman, and Alkes L Price. Estimating and interpreting fst: the impact of rare variants. *Genome research*, pages gr–154831, 2013.

[6] Elizabeth Marchani Blue. Identity-by-descent graphs offer a flexible framework for imputation and both linkage and association analyses. *BMC Proceedings*, 2014.

[7] Brian L Browning and Sharon R Browning. A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. *The American Journal of Human Genetics*, 84(2):210–223, 2009.

[8] Brian L Browning and Sharon R Browning. Genotype imputation with millions of reference samples. *The American Journal of Human Genetics*, 98(1):116–126, 2016.

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

[10] Sharon R Browning and Brian L Browning. Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering. *The American Journal of Human Genetics*, 81(5):1084–1097, 2007.

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

[12] Joshua T Burdick, Wei-Min Chen, Gonçalo R Abecasis, and Vivian G Cheung. In silico method for inferring genotypes in pedigrees. *Nature genetics*, 38(9):1002, 2006.

[13] Ming-Huei Chen, Jie Huang, Wei-Min Chen, Martin G Larson, Caroline S Fox, Ramachandran S Vasan, Sudha Seshadri, Christopher J O'Donnell, and Qiong Yang. Using family-based imputation in genome-wide association studies with large complex pedigrees: the framingham heart study. *PLoS One*, 7(12):e51589, 2012.

[14] Charles YK Cheung, Elizabeth Marchani Blue, and Ellen M Wijsman. A statistical framework to guide sequencing choices in pedigrees. *The American Journal of Human Genetics*, 94(2):257–267, 2014.

[15] Charles YK Cheung, Elizabeth A Thompson, and Ellen M Wijsman. Gigi: an approach to effective imputation of dense genotypes on large pedigrees. *The American Journal of Human Genetics*, 92(4):504–516, 2013.

[16] Eric C Chi, Hua Zhou, Gary K Chen, Diego Ortega Del Vecchyo, and Kenneth Lange. Genotype imputation via matrix completion. *Genome research*, 23(3):509–518, 2013.

[17] Ren-Hua Chung, Wei-Yun Tsai, Chen-Yu Kang, Po-Ju Yao, Hui-Ju Tsai, and Chia-Hsiang Chen. Fampipe: an automatic analysis pipeline for analyzing sequencing data in families for disease studies. *PLoS computational biology*, 12(6):e1004980, 2016.

[18] Domenic V Cicchetti and Alvan R Feinstein. High agreement but low kappa: Ii. resolving the paradoxes. *Journal of clinical epidemiology*, 43(6):551–558, 1990.

[19] 1000 Genomes Project Consortium et al. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061, 2010.

[20] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56, 2012.

[21] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

[22] International HapMap Consortium et al. The international hapmap project. *Nature*, 426(6968):789, 2003.

[23] International Parkinson's Disease Genomics Consortium, Wellcome Trust Case Control Consortium 2 (WTCCC2, et al. A two-stage meta-analysis identifies several new loci for parkinson's disease. *PLoS genetics*, 7(6):e1002142, 2011.

[24] UK10K consortium et al. The uk10k project identifies rare variants in health and disease. *Nature*, 526(7571):82, 2015.

[25] Sayantan Das, Lukas Forer, Sebastian Schönherr, Carlo Sidore, Adam E Locke, Alan Kwong, Scott I Vrieze, Emily Y Chew, Shawn Levy, Matt McGue, et al. Next-generation genotype imputation service and methods. *Nature genetics*, 48(10):1284, 2016.

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

[27] Robert C Elston and John Stewart. A general model for the genetic analysis of pedigree data. *Human heredity*, 21(6):523–542, 1971.

[28] Evangelos Evangelou, Demetrius M Maraganore, and John PA Ioannidis. Meta-analysis in genome-wide association datasets: strategies and application in parkinson disease. *PLoS One*, 2(2):e196, 2007.

[29] Alvan R Feinstein and Domenic V Cicchetti. High agreement but low kappa: I. the problems of two paradoxes. *Journal of clinical epidemiology*, 43(6):543–549, 1990.

[30] Joachim Friedrich, Thomas Dandekar, Matthias Wolf, and Tobias Müller. ProfDist: a tool for the construction of large phylogenetic trees based on profile distances. *Bionformatics*, 21(9):2108–2109, May 2005.

[31] Yun-Xin Fu. Statistical properties of segregating sites. *Theoretical population biology*, 48(2):172–197, 1995.

[32] Christian Fuchsberger, Gonçalo R Abecasis, and David A Hinds. minimac2: faster genotype imputation. *Bioinformatics*, 31(5):782–784, 2014.

[33] Christian Fuchsberger, Jason Flannick, Tanya M Teslovich, Anubha Mahajan, Vineeta Agarwala, Kyle J Gaulton, Clement Ma, Pierre Fontanillas, Loukas Moutsianas, Davis J McCarthy, et al. The genetic architecture of type 2 diabetes. *Nature*, 536(7614):41, 2016.

[34] Stacey B Gabriel, Stephen F Schaffner, Huy Nguyen, Jamie M Moore, Jessica Roy, Brendan Blumenstiel, John Higgins, Matthew DeFelice, Amy Lochner, Maura Faggart, et al. The structure of haplotype blocks in the human genome. *Science*, 296(5576):2225–2229, 2002.

[35] O. Gascuel. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. *Mol Biol Evol*, 14(7):685–695, Jul 1997.

[36] Daniel Gerlach, Matthias Wolf, Thomas Dandekar, Tobias Müller, Andreas Pokorny, and Sven Rahmann. Deep metazoan phylogeny. *In Silico Biol*, 7(2):151–154, 2007.

[37] Alejandro Grajales, Catalina Aguilar, and Juan A Sánchez. Phylogenetic reconstruction using secondary structures of internal transcribed spacer 2 (its2, rdna): finding the molecular and morphological gap in caribbean gorgonian corals. *BMC Evol Biol*, 7:90, 2007.

[38] Godfrey Harold Hardy et al. Mendelian proportions in a mixed population. *Classic papers in genetics. Prentice-Hall, Inc.: Englewood Cliffs, NJ*, pages 60–62, 1908.

[39] Simon C Heath. Markov chain monte carlo segregation and linkage analysis for oligogenic models. *American journal of human genetics*, 61(3):748, 1997.

[40] Anthony Francis Herzig, Teresa Nutile, Marie-Claude Babron, Marina Ciullo, Céline Bellenguez, and Anne-Louise Leutenegger. Strategies for phasing and imputation in a population isolate. *Genetic epidemiology*, 42(2):201–213, 2018.

[41] M. Hochsmann, M. Hochsmann, B. Voss, and R. Giegerich. Pure multiple RNA secondary structure alignments: a progressive profile approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):53–62, 2004.

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

[43] Bryan N Howie, Peter Donnelly, and Jonathan Marchini. A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. *PLoS genetics*, 5(6):e1000529, 2009.

[44] Jie Huang, Bryan Howie, Shane McCarthy, Yasin Memari, Klaudia Walter, Josine L Min, Petr Danecek, Giovanni Malerba, Elisabetta Trabetti, Hou-Feng Zheng, et al. Improved imputation of low-frequency and rare variants using the uk10k haplotype reference panel. *Nature communications*, 6:8111, 2015.

[45] Mengting Huang, Jing Tu, and Zuhong Lu. Recent advances in experimental whole genome haplotyping methods. *International journal of molecular sciences*, 18(9):1944, 2017.

[46] Cendrine Hudelot, Vivek Gowri-Shankar, Howsun Jow, Magnus Rattray, and Paul G Higgs. RNA-based phylogenetic methods: application to mammalian mitochondrial RNA sequences. *Mol Phylogenet Evol*, 28(2):241–252, Aug 2003.

[47] Nicholas A Johnson, Marc A Coram, Mark D Shriver, Isabelle Romieu, Gregory S Barsh, Stephanie J London, and Hua Tang. Ancestral components of admixed genomes in a mexican cohort. *PLoS genetics*, 7(12):e1002410, 2011.

[48] H. Jow, C. Hudelot, M. Rattray, and P. G. Higgs. Bayesian phylogenetics using an rna substitution model applied to early mammalian evolution. *Mol Biol Evol*, 19(9):1591–1601, Sep 2002.

[49] T. Jukes and C.R. Cantor. Evolution of protein molecules. In H Munro, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, New York, USA, 1969.

[50] Anatoly V Kirichenko, Nadezhda M Belonogova, Yurii S Aulchenko, and Tatiana I Axenovich. Pedstr software for cutting large pedigrees for haplotyping,

ibd computation and multipoint linkage analysis. *Annals of human genetics*, 73(5):527–531, 2009.

[51] Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81:1–10, 2009.

[52] Leonid Kruglyak, Mark J Daly, Mary Pat Reeve-Daly, and Eric S Lander. Parametric and nonparametric linkage analysis: a unified multipoint approach. *American journal of human genetics*, 58(6):1347, 1996.

[53] Khalid Kunji, Ehsan Ullah, Alejandro Q Nato Jr, Ellen M Wijsman, and Mohamad Saad. Gigi-quick: a fast approach to impute missing genotypes in genome-wide association family data. *Bioinformatics*, 34(9):1591–1593, 2017.

[54] Eric S Lander and Philip Green. Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences*, 84(8):2363–2367, 1987.

[55] Kenneth Lange. Kenneth lange: "next generation statistical genetics: Modeling, [...]", 2016.

[56] Samantha Lent, Xuan Deng, L Adrienne Cupples, Kathryn L Lunetta, CT Liu, and Yanhua Zhou. Imputing rare variants in families using a two-stage approach. In *BMC proceedings*, volume 10, page 48. BioMed Central, 2016.

[57] Yun Li, Cristen J Willer, Jun Ding, Paul Scheet, and Gonçalo R Abecasis. Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genetic epidemiology*, 34(8):816–834, 2010.

[58] Peng Lin, Sarah M Hartz, Zhehao Zhang, Scott F Saccone, Jia Wang, Jay A Tischfield, Howard J Edenberg, John R Kramer, Alison M Goate, Laura J Bierut, et al. A new statistic to evaluate imputation reliability. *PloS one*, 5(3):e9697, 2010.

[59] Fan Liu, Anatoliy Kirichenko, Tatiana I Axenovich, Cornelia M Van Duijn, and Yurii S Aulchenko. An approach for cutting large and complex pedigrees for linkage analysis. *European Journal of Human Genetics*, 16(7):854, 2008.

[60] Oren E Livne, Lide Han, Gorka Alkorta-Aranburu, William Wentworth-Sheilds, Mark Abney, Carole Ober, and Dan L Nicolae. Primal: fast and accurate pedigree-based imputation from sequence data in a founder population. *PLoS computational biology*, 11(3):e1004139, 2015.

[61] Po-Ru Loh, Pier Francesco Palamara, and Alkes L Price. Fast and accurate long-range phasing in a uk biobank cohort. *Nature genetics*, 48(7):811, 2016.

[62] Brendan Maher. Personal genomes: The case of the missing heritability. *Nature News*, 456(7218):18–21, 2008.

[63] Teri A Manolio, Francis S Collins, Nancy J Cox, David B Goldstein, Lucia A Hindorff, David J Hunter, Mark I McCarthy, Erin M Ramos, Lon R Cardon, Aravinda Chakravarti, et al. Finding the missing heritability of complex diseases. *Nature*, 461(7265):747, 2009.

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

[65] Urko M Marigorta, Juan Antonio Rodríguez, Greg Gibson, and Arcadi Navarro. Replicability and prediction: lessons and challenges from gwas. *Trends in Genetics*, 2018.

[66] RA Mathias, MA Taub, CR Gignoux, W Fu, S Musharoff, TD O'Connor, C Vergara, DG Torgerson, M Pino-Yanes, SS Shringarpure, et al. Caapa (2016). a continuum of admixture in the western hemisphere revealed by the african diaspora genome. *Nat. Commun*, 7:12522, 2016.

[67] Shane McCarthy, Sayantan Das, Warren Kretzschmar, Olivier Delaneau, Andrew R Wood, Alexander Teumer, Hyun Min Kang, Christian Fuchsberger, Petr Danecek, Kevin Sharp, et al. A reference panel of 64,976 haplotypes for genotype imputation. *Nature genetics*, 48(10):1279, 2016.

[68] Shane McCarthy, Sayantan Das, Warren Kretzschmar, Olivier Delaneau, Andrew R Wood, Alexander Teumer, Hyun Min Kang, Christian Fuchsberger, Petr Danecek, Kevin Sharp, et al. A reference panel of 64,976 haplotypes for genotype imputation. *Nature genetics*, 48(10):1279, 2016.

[69] Francesco Montinaro, George BJ Busby, Vincenzo L Pascali, Simon Myers, Garrett Hellenthal, and Cristian Capelli. Unravelling the hidden ancestry of american admixed populations. *Nature communications*, 6:6596, 2015.

[70] T. Müller and M. Vingron. Modeling amino acid replacement. *J Comput Biol*, 7(6):761–776, 2000.

[71] Tobias Müller, Sven Rahmann, Thomas Dandekar, and Matthias Wolf. Accurate and robust phylogeny estimation based on profile distances: a study of the Chlorophyceae (Chlorophyta). *BMC Evol Biol*, 4:20, Jun 2004.

[72] Tobias Müller, Rainer Spang, and Martin Vingron. Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol Biol Evol*, 19(1):8–13, Jan 2002.

[73] Mike A Nalls, Nathan Pankratz, Christina M Lill, Chuong B Do, Dena G Hernandez, Mohamad Saad, Anita L DeStefano, Eleanna Kara, Jose Bras, Manu Sharma, et al. Large-scale meta-analysis of genome-wide association data identifies six new risk loci for parkinson's disease. *Nature genetics*, 46(9):989, 2014.

[74] Masatoshi Nei. Definition and estimation of fixation indices. *Evolution*, 40(3):643–645, 1986.

[75] Masatoshi Nei, Fumio Tajima, and Yoshio Tateno. Accuracy of estimated phylogenetic trees from molecular data. *Journal of molecular evolution*, 19(2):153–170, 1983.

[76] Matthew R Nelson, Katarzyna Bryc, Karen S King, Amit Indap, Adam R Boyko, John Novembre, Linda P Briley, Yuka Maruyama, Dawn M Waterworth, Gérard Waeber, et al. The population reference sample, popres: a resource for population, disease, and pharmacological genetics research. *The American Journal of Human Genetics*, 83(3):347–358, 2008.

[77] Carl Nettelblad. Inferring haplotypes and parental genotypes in larger full sib-ships and other pedigrees with missing or erroneous genotype data. *BMC genetics*, 13(1):85, 2012.

[78] Jared O'Connell, Deepti Gurdasani, Olivier Delaneau, Nicola Pirastu, Sheila Ulivi, Massimiliano Cocca, Michela Traglia, Jie Huang, Jennifer E Huffman, Igor Rudan, et al. A general approach for haplotype phasing across the full spectrum of relatedness. *PLoS genetics*, 10(4):e1004234, 2014.

[79] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[80] Sven Rahmann, Tobias Muller, Thomas Dandekar, and Matthias Wolf. Efficient and robust analysis of large phylogenetic datasets. In Hui-Huang Hsu, editor, *Advanced Data Mining Technologies in Bioinformatics*, pages 104–117. Idea Group, Inc., Hershey, PA, USA, 2006.

[81] Shelina Ramnarine, Juan Zhang, Li-Shiun Chen, Robert Culverhouse, Weimin Duan, Dana B Hancock, Sarah M Hartz, Eric O Johnson, Emily Olfson, Tae-Hwi Schwantes-An, et al. When does choice of accuracy measure alter imputation accuracy assessments? *PloS one*, 10(10):e0137601, 2015.

[82] Shelina Raynell Ramnarine. Genetic imputation: Accuracy to application. *Washington University Open Scholarship, Arts and Sciences Electronic Theses and Dissertations*, 2016.

[83] Nab Raj Roshyara and Markus Scholz. Impact of genetic similarity on imputation accuracy. *BMC genetics*, 16(1):90, 2015.

[84] Mohamad Saad, Alejandro Q Nato, Fiona L Grimson, Steven M Lewis, Lisa A Brown, Elizabeth M Blue, Timothy A Thornton, Elizabeth A Thompson, and Ellen M Wijsman. Identity-by-descent estimation with population-and pedigree-based imputation in admixed family data. In *BMC proceedings*, volume 10, page 7. BioMed Central, 2016.

[85] Mohamad Saad and Ellen M Wijsman. Power of family-based association designs to detect rare variants in large pedigrees using imputed genotypes. *Genetic epidemiology*, 38(1):1–9, 2014.

[86] Mohamad Saad and Ellen M Wijsman¡? vsp4pt?¿. Combining family-and population-based imputation data for association analysis of rare and common variants in large pedigrees. *Genetic epidemiology*, 38(7):579–590, 2014.

[87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol*, 4(4):406–425, Jul 1987.

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

[89] Jörg Schultz, Stefanie Maisel, Daniel Gerlach, Tobias Müller, and Matthias Wolf. A common core of secondary structure of the internal transcribed spacer 2 (ITS2) throughout the Eukaryota. *RNA*, 11(4):361–364, Apr 2005.

[90] Jörg Schultz, Tobias Müller, Marco Achtziger, Philipp N Seibel, Thomas Dandekar, and Matthias Wolf. The internal transcribed spacer 2 database–a web server for (not only) low level phylogenetic analyses. *Nucleic Acids Res*, 34(Web Server issue):W704–W707, Jul 2006.

[91] M. Schöniger and A. von Haeseler. A stochastic model for the evolution of autocorrelated DNA sequences. *Mol Phylogenet Evol*, 3(3):240–247, Sep 1994.

[92] Philipp N Seibel, Tobias Müller, Thomas Dandekar, Jörg Schultz, and Matthias Wolf. 4SALE–a tool for synchronous RNA sequence and secondary structure alignment and editing. *BMC Bioinformatics*, 7:498, 2006.

[93] Christian Selig, Matthias Wolf, Tobias Müller, Thomas Dandekar, and Jörg Schultz. The ITS2 Database II: homology modelling RNA structure for molecular systematics. *Nucleic Acids Res*, 36(Database issue):D377–D380, Jan 2008.

[94] Afshan Siddiq, Fergus J Couch, Gary K Chen, Sara Lindström, Diana Eccles, Robert C Millikan, Kyriaki Michailidou, Daniel O Stram, Lars Beckmann, Suhn Kyong Rhie, et al. A meta-analysis of genome-wide association studies of breast cancer identifies two novel susceptibility loci at 6q14 and 20q11. *Human molecular genetics*, 21(24):5373–5384, 2012.

[95] Sven Siebert and Rolf Backofen. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21(16):3352–3359, Aug 2005.

[96] Andrew D Smith, Thomas W H Lui, and Elisabeth R M Tillier. Empirical models for substitution in ribosomal RNA. *Mol Biol Evol*, 21(3):419–427, Mar 2004.

[97] Suzanne Sniekers, Sven Stringer, Kyoko Watanabe, Philip R Jansen, Jonathan RI Coleman, Eva Krapohl, Erdogan Taskesen, Anke R Hammerschlag, Aysu Okbay, Delilah Zabaneh, et al. Genome-wide association meta-analysis of 78,308 individuals identifies new loci and genes influencing human intelligence. *Nature genetics*, 49(7):1107, 2017.

[98] Eric Sobel and Kenneth Lange. Descent graphs in pedigree analysis: applications to haplotyping, location scores, and marker-sharing statistics. *American journal of human genetics*, 58(6):1323, 1996.

[99] Jeffrey Staples, Deborah A Nickerson, and Jennifer E Below. Utilizing graph theory to select the largest set of unrelated individuals for genetic analysis. *Genetic epidemiology*, 37(2):136–141, 2013.

[100] Jeffrey Staples, Dandi Qiao, Michael H Cho, Edwin K Silverman, Deborah A Nickerson, Jennifer E Below, University of Washington Center for Mendelian Genomics, et al. Primus: rapid reconstruction of pedigrees from genome-wide estimates of identity by descent. *The American Journal of Human Genetics*, 95(5):553–564, 2014.

[101] Zhan Su, Jonathan Marchini, and Peter Donnelly. Hapgen2: simulation of multiple disease snps. *Bioinformatics*, 27(16):2304–2305, 2011.

[102] Naoko Takezaki and Masatoshi Nei. Genetic distances and reconstruction of phylogenetic trees from microsatellite dna. *Genetics*, 144(1):389–399, 1996.

[103] EA Thompson, S Lin, AB Olshen, and EM Wijsman. Monte carlo analysis on a large pedigree. *Genetic epidemiology*, 10(6):677–682, 1993.

[104] Elizabeth Thompson. The structure of genetic linkage data: from liped to 1m snps. *Human heredity*, 71(2):86–96, 2011.

[105] Trolltech. http://trolltech.com/products/qt/, 2008.

[106] John S Uebersax. Diversity of decision-making models and the measurement of interrater agreement. *Psychological bulletin*, 101(1):140, 1987.

[107] Ehsan Ullah, Khalid Kunji, Ellen M. Wijsman, and Mohamad Saad. Gigi2: A fast approach for parallel genotype imputation in large pedigrees. *Unpublished*, 2018.

[108] Ehsan Ullah, Raghvendra Mall, Mostafa M. Abbas, Khalid Kunji, Alejandro Q. Nato Jr., Halima Bensmail, Ellen M. Wijsman, and Mohamad Saad. Comparison and assessment of family- and population-based genotype imputation methods in large pedigrees. *Genome Research*, 2018. (in press).

[109] P Vanormelingen, E Hegewald, A Braband, M Kitschke, T Friedl, K Sabbe, and W Vyverman. The systematics of a small spineless Desmodesmus species, D-costato-granulatus (Sphaeropleales, Chlorophyceae), based on ITS2 rDNA sequence analyses and cell wall morphology. *Journal of Phyclology*, 43(2):378–396, APR 2007.

[110] Anthony J Viera, Joanne M Garrett, et al. Understanding interobserver agreement: the kappa statistic. *Fam Med*, 37(5):360–363, 2005.

[111] William YS Wang, Bryan J Barratt, David G Clayton, and John A Todd. Genome-wide association studies: theoretical and practical concerns. *Nature Reviews Genetics*, 6(2):109, 2005.

[112] W Weinberg. On the demonstration of heredity in man. *(1963) Papers on human genetics*, 1908.

[113] D. L. Wheeler, C. Chappey, A. E. Lash, D. D. Leipe, T. L. Madden, G. D. Schuler, T. A. Tatusova, and B. A. Rapp. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Res*, 28(1):10–14, Jan 2000.

[114] Ellen M Wijsman. The role of large pedigrees in an era of high-throughput sequencing. *Human genetics*, 131(10):1555–1563, 2012.

[115] Ellen M Wijsman, Joseph H Rothstein, and Elizabeth A Thompson. Multipoint linkage analysis with many multiallelic or dense diallelic markers: Markov chain–monte carlo provides practical approaches for genome scans on general pedigrees. *The American Journal of Human Genetics*, 79(5):846–858, 2006.

[116] Cristen J Willer, Yun Li, and Gonçalo R Abecasis. Metal: fast and efficient meta-analysis of genomewide association scans. *Bioinformatics*, 26(17):2190–2191, 2010.

[117] Matthias Wolf, Marco Achtziger, Jörg Schultz, Thomas Dandekar, and Tobias Müller. Homology modeling revealed more than 20,000 rRNA internal transcribed spacer 2 (ITS2) secondary structures. *RNA*, 11(11):1616–1623, Nov 2005.