Efficient Computational Techniques for Tag SNP Selection, Epistasis Analysis, and Genome-Wide Association Study

WANG YUE
(B.Eng.(Hons.), NWPU)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2012

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Wang Yue
28th November 2012

I would like to dedicate this thesis to my loving mother Zhang Meiying and father Wang Yisong.

Acknowledgements

I would like to extend my deep gratitude to every person in my life who has helped me during the past four years of my PhD studies.

Foremost, I thank my mentor, Professor Wong Limsoon. He has given me the academic freedom to explore a variety of topics in bioinformatics, which brings me to the field of genome-wide association studies. He guided me in developing ideas rigorously and logically through our regular meetings over the past four years. I especially appreciate his encouragement and patience towards me so that I can finish this thesis while supporting my family.

I thank also my other two Thesis Advisory Committee members: Professor Tan Kian-Lee and Professor Wynne Hsu. Professor Tan Kian-Lee introduced and explained Hadoop technology to me, which, later, is used in my research. I am grateful to both of them for providing invaluable comments at our regular TAC meetings.

I am extremely grateful to my two seniors: Dr Liu Guimei and Dr Feng Mengling. Dr Liu Guimei has been very supportive and would always inspire me to find solutions when I faced difficulties at the early stages of my PhD. Dr Feng Mengling introduced me to many data mining techniques and has been like an older brother, who cares about my leisure life and taught me street dance.

I would also like to express special thanks to Dr Giovanni Montana and Professor Philip Keith Moore, who gave me an opportunity to do research at Imperial College London.

I thank the NUS Graduate School for Integrative Sciences and Engineering (NGS) for providing a generous scholarship and abundant opportunities to attend conferences, as well as the School of Computing for providing software and hardware facilities to me.

Also, I would like to extend my appreciation to my dear Computational Biology Lab mates like Sucheendra Kumar Palaniappan, Benjamin Mate Gyori, Fan

Mengyuan, Yong Chern Han, Chandana Tennakoon, Zhang Haojun, Hugo Willy and other members. We had a wonderful time discussing and exchanging ideas with each other over the past four years.

Last but not the least, I deeply thank my beloved parents for raising me. I would also like to thank my father's greatness, who supported me in achieving my goals despite his own struggles.

Summary

This thesis explores data analysis involved in genome-wide association studies (GWAS) using Hadoop technologies and data mining techniques. GWAS is amongst the most popular study designs to identify potential genetic variants that are linked to the etiologies of diseases. In future, GWAS is also expected to play an important role in personalized medicine. The complex data analysis in GWAS calls for new technologies and techniques.

We first give an independent, empirical comparison of epistasis detection methods in GWAS. The experimental results show that methods that examine all possible candidate pairs are more powerful. Also, the results encourage users to choose suitable test statistics to detect corresponding epistasis. These two observations lead us to use a powerful, fault-tolerant and parallel technology—Hadoop. We are probably the first practitioners to effectively "marry" the epistasis detection in GWAS with Hadoop, resulting in two new computing tools for detecting epistasis called CEO and efficient CEO (eCEO). Our experiments show that CEO and eCEO are computationally efficient, flexible, and scalable. However, CEO and eCEO are limited to binary datasets.

Another major category of GWAS concerns quantitative traits, especially high-dimensional traits. Seeing the advantage of using Hadoop in GWAS, we adapt a powerful machine learning technique—Random Forest (RF)—to develop a Parallel Random Forest Regression (PaRFR) algorithm on Hadoop for high-dimensional traits. The algorithm is significantly faster than a standard implementation of RF. The motivating application of this algorithm on Alzheimer's Disease Neuroimaging Initiative (ADNI) data illustrates its power in detecting known Alzheimer-linked genes like APOE. We further extract insights from the ADNI data by hypothesizing that (i) there is a large set of biomarkers (mutation patterns) that are relevant to the development of Alzheimer's Disease (AD) and (ii) the more members of this set are observed in a patient, the more likely he/she

has a more severe level of AD. To validate this, we define the mutation patterns and the severity of AD in a novel way. Through investigating the relationship between the count of certain mutation patterns and the severity of AD, we have established a positive correlation between these two, and the hypotheses are thus supported.

The final part of this thesis investigates another two research problems in GWAS: tag SNP selection and SNP imputation. We realize that the computationally expensive and memory-intensive tag SNP selection methods in the literature cannot work on genome-wide data. So we propose a fast and efficient genome-wide tag SNP selection algorithm (called FastTagger) using multi-marker linkage disequilibrium. The algorithm can work on data with more than 100k SNPs that previous methods cannot handle. We further utilize the rules produced by FastTagger and develop a new tag-based imputation method called RuleImpute, which suggests rules with minimum span to achieve the best imputation accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Genome-wide association studies (GWAS)

A genome-wide association study (GWAS) searches for inherent expression (genetic patterns) from a genome that is potentially associated with outward expression (phenotypes) in a carefully designed study. A typical study usually consists of 500k∼1,000k [Psychiatric GWAS Consortium Coordinating Committee et al., 2009] genetic markers. These markers capture at least 80% of common genetic variations of the human genome using a cost-effective genotyping platform. The phenotypes of the study normally record the observed characteristics of hundreds or thousands of samples selected from a certain population. Such a study design provides an unbiased, full-genome search for genetic patterns in samples with different phenotypes. The identified genetic patterns act as risk factors of developing certain outward expression. Depending on the research methods, there are different associations between genetic patterns and outward expressions. Broadly speaking, three widely studied genetic patterns are single-marker, multi-marker and pair-marker. The early research focuses on identifying susceptible single-marker patterns that are associated with outward expression. These studies are relatively less powerful and account for less amount of explained heritability. By considering several genetic markers simultaneously in a statistical model [Hoggart et al., 2008; Wang et al., 2012], the multi-marker patterns reported are more

powerful since complex diseases are possibly caused by multiple causal variants. To further explain the "missing heritability" [Eichler et al., 2010; Manolio et al., 2009], pair marker patterns (gene-gene interaction), termed as epistasis [Bateson, 1909; Phillips, 2008], attract more attention. The discovery of epistasis is motivated by biological observations and statistical findings. On the other hand, the outward expression, also termed trait/phenotype/disease status interchangeably, has a variety of forms. They can be a sample's body mass index when studying quantitative trait; they can be the healthy and disease status for a case-control study; they can be the record of a large number of different voxels in brain images for a high-dimensional imaging genetic study. Different forms of inherent genetic patterns and outward expression make GWAS a rather general concept. GWAS is expected to be superior to conventional linkage and candidate gene studies [Psychiatric GWAS Consortium Coordinating Committee et al., 2009] in terms of power and fine-mapping due to its unbiased, large-cohort and full-genome study design.

The first exciting finding of GWAS was on age-related macular degeneration (AMD) [Klein, 2005], which uncovers a disease allele (tyrosinehistidine polymorphism) with an effect size of 4.6 in 100,000 single nucleotide polymorphisms (SNPs). In 2007, Wellcome Trust Case Control Consortium (WTCCC) [Burton et al., 2007] released its well-designed GWAS data of seven complex diseases to researchers, which was the landmark of GWAS discovery in the past decade. Research on WTCCC GWAS data has uncovered many previously unknown susceptible genes in type 1 diabetes, type 2 diabetes, breast cancer, multiple sclerosis, Crohn's disease, colorectal cancer, and prostate cancer. Since then, reported GWAS discoveries have accumulated significantly and have therefore largely expanded our understanding of the etiology of complex diseases. As of June 2012, there are 1,287 publications and 6,499 reported SNPs associated with over 300 traits or diseases. All these discoveries are done in 7 years; thus the success of GWAS is undeniable [Visscher et al., 2012].

1.1.2 Computational challenges in GWAS

Despite the successful application of GWAS, it poses some computational challenges to the community. The early analysis of GWAS is centered around single-marker analysis using different test statistics. The main reason is due to the heavy computational burden in estimating model parameters if hundreds of thousands of SNPs are analyzed together. For example, the two-locus χ^2 test requires constructing a contingency table with 2 rows and 9 columns, and the three-locus χ^2 test requires 2 rows and 81 columns. The number of columns to construct grows exponentially when more SNPs have to be considered together. Currently, it is impossible go beyond three-locus association test due to limited sample size and computational complexity. Therefore, as described in Chapter 2, the assumption that a small number of SNPs are jointly associated with the phenotype is imposed for retrospective (like χ^2 test) and prospective (like logistic regression) statistical modeling.

Studies on gene ontologies, protein-protein interaction networks, protein complexes, protein triplets, and pathways have accumulated a wealth of biological knowledge. Although they are not complete and still evolving, researchers agree that these biological and other domain knowledge can be used to benefit GWAS. Some researchers [Wilke et al., 2008] suggest we should not begin GWAS before we have extensive of knowledge on candidate genes and pathways. However, there is still no consensus on what the best way to integrate the abundant "high level" knowledge into GWAS is. Moreover, there is no all-in-one database that stores different types of biological knowledge in one place and supports cross query in different formats. The computational challenge not only comes from storing, extracting, and loading these data, but also from the proper use of the accumulated knowledge in an efficient and meaningful way.

Computational challenge also arises when the aim of GWAS is to detect gene-gene/environment interactions (epistasis) that are associated with a phenotype. Biologically, epistasis [Bateson, 1909] is defined as the change of segregation ratio and the interaction of genes. However, detecting epistasis in GWAS is computationally challenging because it involves analyzing a large number of SNP pairs. Given that current SNP chips can genotype at least 1 million SNPs, the number

of possible SNP pairs can be as large as $5*10^{11}$. Ma et al. [2008] estimate that 4.8 years are needed to finish epistasis testing of 1 million SNPs using a sequential program on a 2.66 GHz single processor. Different heuristics have been proposed to prune the huge number of pairs so that the remaining pairs are within a more manageable size, ranging from several hundreds to thousands [Long et al., 2009; Yang et al., 2009].

Computational challenges not only occur in statistical analysis, but also in machine learning techniques. Most machines learning techniques are non-parametric, and are able to handle high dimensionality. Although they are widely used in the analysis of GWAS data, the computational obstacle is the headache of many researchers. For example, Random Forest [Breiman, 2001] is a popular method for detecting epistasis [Cook et al., 2004; Jiang *et al.*, 2009; Lunetta et al., 2004] by modeling epistasis as the two connected nodes of an edge in a tree of a random forest. In applying Random Forest to a typical case-control data set with 1,000,000 SNPs and 2,000 samples, on average 1,000 SNPs are used to construct a tree. A rough estimate for building a tree with 1,000 nodes for 2,000 data points is $\sim$ 1 hour on a typical PC. How many trees are "ideal" for detecting epistasis? There are 1,000 SNPs in each tree on average and in total there are 1,000,000 SNPs. So the probability of a given SNP being in a specific tree is 10^{-3}. The probability of the two specific SNPs occurring in the same tree is then 10^{-6}. This means that, after building 1,000,000 trees, we can only expect to see the two SNPs occurring in the same tree once. But building 1,000,000 trees takes 1,000,000 hours in a single PC, or, 114.15 years. This makes the analysis of typical GWAS data a computationally prohibitive task.

1.1.3 Big data, Hadoop and associated technologies

We live with digital data every day. Searching keywords, reading news, sending emails, listening to music, browsing websites, sharing social media feeds, shopping online, watching videos and so on are part of daily routines of 2.5 billion netizens in the world. All these digital activities are backed by a variety of data and

related technologies. As long as one accesses the Internet of things[1] , he/she is in the process of generating, communicating and consuming data. Data is no longer a meaningless bit that people can neglect. It is now considered as a digital asset for a person, an organization and an industry [Manyika et al., 2011].

Big data, describing the current digital era situation which we are in, is distinguished from traditional data in the four "V"s[2]: Volume, Velocity, Variety and Variability. Volume indicates that the size of data is too big to process using traditional IT infrastructures. Velocity defines the speed at which data are processed. Depends on the task, the requirement of velocity can be real-time or within several hours. Variety describes the analysis complexity of big data which is a mix of structured and un-structured data. Variability refers to the flexible ways of interpreting the insights extracted from big data, and different questions lead to different story tellings. These four "V"s characteristics of big data attract academic institutes and industrial companies to mine value out of them.

To support aggregation, manipulation, management and analysis of big data, many innovative technologies that use distributed storage and computation are emerging rapidly. In particular, Hadoop, an open-source framework originally developed based on Google's MapReduce [Dean and Ghemawat, 2004] and Google File System [Ghemawat et al., 2003], has now become the kernel of the Hadoop ecosystem which is a project under Apache Software Foundation. The core parts of the Hadoop ecosystem are HDFS and MapReduce. Hadoop Distributed File System (HDFS) is the distributed storage file system that creates user-defined replicas of data blocks and distributes them on data nodes throughout a cluster to enable fault-tolerant and fast computations. MapReduce is a programming model that divides data processing into map and reduce phases which have been known and used in functional programming. To better utilize the power of distributed storage and computation, the Hadoop ecosystem is adding other useful

[1] http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/the_internet_of_things

[2] http://blogs.forrester.com/brian_hopkins/11-08-29-big_data_brewer_and_a_couple_of_webinars

Table 1.1: Summary of different big data technologies

Technology	Summary
Cassandra	A scalable and high-availability distributed database management system for large-scale data.
BI softwares	They are used to read, analyze, and generate standard report to the user, possibly, on a periodic basis. Example softwares are IBM Cognos Series 10, Tableau, SAP NetWeaver BI and so on.
ETL tools	They are used for tasks of Extract, Transform and Load data. Example tools are SAP BusinessObjects Data Integrator, SQL Server Integration Services, and Informatica Powercenter.
R	An open-source, powerful programming language and software mainly for statistical computing. The R framework has been extended to analyze big data recently.
Visualization	This use pictures, diagrams, shapes and animations to better present the insights extracted from data. Popular tools include IBM Cognos Insight, Palantir financial, and SAP Visual Intelligence.

components. Some examples are Hive[1], HBase[2], Pig[3] and Mahout[4]. Hive is developed as SQL-like data warehouse for data summarization, query and analysis. Pig is a high-level data flow language used to ease the burden of map and reduce functional programming. HBase is built on top of HDFS to store unstructured data, thus it is fault-tolerant and can cooperate with MapReduce jobs seamlessly. Mahout is an open-source machine learning library specifically for large-scale data analysis on Hadoop. The Hadoop ecosystem is evolving and becoming the "standard" technology for big data analysis. As Manyika et al. [2011] suggested, other big data technologies include Cassandra, business intelligence (BI) software, Extract-transform-load (ETL) tools, R, visualization and so on. Their descriptions are given in Table 1.1.

[1] http://hive.apache.org/
[2] http://hbase.apache.org/
[3] http://pig.apache.org/
[4] http://mahout.apache.org/

1.1.4 Hadoop in genome analysis

Considering the consistently dropping cost of sequencing technologies, it is anticipated that by mid 2013, we will enter an era of sequencing one genome at the cost of $1,000 or below[1]. At that time, we will need to analyze and interpret whole-genome data for personalized medicine. Currently, many preparations for genome analysis using big data technologies are on the way. Hadoop-BAM [Niemenmaa et al., 2012], specifically designed for sequence alignment of NGS data, provides a library for directly manipulating the aligned NGS data, which is stored in BAM file (Binary Alignment Map). Eoulsan [Jourdren et al., 2012] provides a cloud computation framework including analysis of high-throughput sequence data from upstream quality control to downstream differential expression detection. Schatz *et al.* [2010] provide a Hadoop software to accelerate the SNP calling and sequence alignment. Langmead et al. [2010] develop an ultrafast and memory efficient program called Bowtie for aligning short DNA sequence reads to large genomes. The same group [Langmead et al., 2009] also develop a cloud-computing pipeline—Myran—for analyzing transcriptome sequencing (RNA-Seq) data. CEO [Wang et al., 2010b] and eCEO [Wang et al., 2011] focus mainly on dividing the exponential combination of tests into the distributed computing tasks in the cloud. Wang et al. [2012] further extend this work by providing a general framework for combinatorial data analysis.

1.2 Outline of the thesis

This thesis investigates the use of big data technologies for GWAS data analysis. Computational complexity is always a factor to consider when analyzing the ever-growing volume of genomic data. Effective application of big data technologies can free researchers to uncover more biological insights. The outline of this thesis is as follows:

Chapter 1 presents an overview of GWAS, big data technologies, Hadoop and the motivation for combining them. The research contributions are listed.

Chapter 2 provides background on GWAS data analysis and two components

[1] http://en.wikipedia.org/wiki/$1,000_genome

of big data technologies: Hadoop HDFS and MapReduce.

Chapter 3 is an empirical study of current epistatic interaction detection methods in GWAS. The study motivates us to use big data technologies for GWAS in Chapter 4, 5, and 6.

Chapter 4 investigates the marriage between big data technologies and GWAS epistatic interaction detection. The computational difficulties are largely alleviated.

Chapter 5 proposes an even more efficient approach for detecting epistasis in the cloud than that described in Chapter 4.

Chapter 6 continues the discussion of using big data technologies in GWAS but in a more challenging setting: analyzing high-dimensional phenotypes instead of binary data. A novel hypothesis on the connection between the number of mutations and severity of the Alzheimer's disease is proposed and preliminary results are obtained. This may inspire further application of such analysis in GWAS.

Chapter 7 discusses another two research problems in GWAS: tag SNP selection and SNP imputation. A novel algorithm called FastTagger is developed to reduce the number of tag SNPs and to improve efficiency. FastTagger is further extended for the SNP imputation problem.

Chapter 8 concludes the thesis with some discussion on the achievements reached.

1.3 Research contributions

In the following, the detailed contribution of each chapter is outlined:

Chapter 3:

Many new methods have recently been proposed for detecting epistatic interactions in GWAS data. There is however no in-depth independent comparison of these methods yet. This chapter provides an independent, empirical comparison of the epistatic interaction detection methods in the literature on different

simulation settings. Unexpectedly, the comparison results show that all the non-exhaustive methods are computationally efficient but at the cost of losing power. This is not a desirable property when designing algorithms for detecting epistatic interactions. That being said, the work guides researchers to design algorithms that can examine all possible pairs or not miss any pairs to achieve enough power, given the increasing computational power. This work also distinguishes the concept of "pure epistasis" and "epistasis allowing for association", which is not clearly mentioned in the literature. This chapter is based on the following paper:

- **Yue Wang**, Guimei Liu, Mengling Feng, Limsoon Wong. An empirical comparison of several recent epistatic interaction detection methods. *Bioinformatics*, 27(21):2936–2943, November 2011. Corrigendum in *Bioinformatics*, 28(1):147–148, January 2012.

Chapter 4 and Chapter 5:

The results of Chapter 3 reveal the necessity of exhaustively examining all possible pairs of genes for epistatic interactions. Such an exhaustive examination is computationally costly and calls for effective parallelization. Chapter 4 describes the first-ever cloud-based epistasis model using Hadoop HDFS and MapReduce technologies. Chapter 5 expands the work of Chapter 4 by describing several ideas for optimizing the distributed computations and significantly speeding up the calculations of test statistics and mining of epistatic interactions. For example, to construct a contingency table, we adopt a Boolean representation of the data and use a bit operation to get the intersection of two Boolean arrays, which is memory efficient and computationally fast compared with using a link list representation and hash operations. The new square chopping model refines the distributed model further by "square chopping" candidate SNP pairs, which can reduce computation further when there is a lot more computation resources. The open-source software eCEO, is specifically designed for users to conduct exhaustive epistatic interaction analysis in private clusters and commercial cloud platforms in several days, which is impossible for a single PC. Additionally, the software has the option of choosing different test statistics for epistasis, depending on the definition of epistasis. For example, the χ^2 test is designed for "epistasis

allowing for association" and likelihood ratio test with 4 df is designed for "pure epistasis". Since the software is open sourced, user can adapt the codes to include more ad-hoc definitions of epistasis. The experimental results and our design of the software demonstrate that it is computationally efficient, flexible, scalable and practical. These two works are published in a conference and a journal separately.

- Zhengkui Wang, **Yue Wang**, Kian-Lee Tan, Limsoon Wong, Divyakant Agrawal. CEO: A Cloud Epistasis cOmputing model in GWAS. In *Proceedings of 4th IEEE International Conference on Bioinformatics & Biomedicine*, pages 85–90, Hong Kong, December 2010.

- Zhengkui Wang, **Yue Wang**, Kian-Lee Tan, Limsoon Wong, Divyakant Agrawal. eCEO: An efficient Cloud Epistasis cOmputing model in genome-wide association study. *Bioinformatics*, 27(8):1045–1051, April 2011

In the two papers above, the greedy and square chopping load-balancing model design should be attributed to Wang Zhengkui. My contribution is the statistical test design, Boolean data operation optimization, and problem abstraction to the MapReduce framework.

Chapter 6:

Chapters 4 and 5 deal with binary traits in GWAS, but another major category of GWAS is those concerning quantitative traits, especially high-dimensional quantitative traits. High-dimensional traits arise naturally in recent neuroimaging genetics studies, in which the phenotypic variability in the human brain is measured by means of 3D neuroimaging data. Random Forest (RF) is amongst the best performing machine learning algorithms for classification tasks and has been successfully applied to the identification of genome-wide associations in case-control studies. RF can also be applied to population association studies with multivariate quantitative traits, whereby the classification task is replaced by a regression task. When applied to whole-genome mapping involving hundreds of thousands of SNPs and multivariate quantitative traits, a very large ensemble of regression trees must be inferred from the data in order to obtain a stable

SNP ranking. The effective application of Hadoop technologies in previous chapters shows a promising direction of analyzing GWAS data. Therefore, Chapter 6 continues the discussion of using Hadoop technologies for analyzing the more challenging high-dimensional quantitative phenotype data on Alzheimer's disease. We have developed a parallel version of RF for regression tasks with both univariate and multivariate responses, called *PaRFR* (Parallel Random Forest Regression), to support multivariate quantitative trait loci mapping in unrelated subjects. *PaRFR* takes advantage of the MapReduce programming model and is deployed on Hadoop. Notable speed-ups have been obtained by introducing a distance-based criterion for node splitting. We also present experimental results from a genome-wide association study on Alzheimer's disease in which the quantitative trait is a high-dimensional neuroimaging phenotype that describes the longitudinal changes in the human brain structure. *PaRFR* provides a ranking of SNPs that reflects their predictive power, and produces pair-wise measures of genetic proximity that can be directly compared to pair-wise measures of phenotypic proximity. Several known AD-related variants have been identified, including APOE4 and TOMM40. Based on the top-ranked SNPs from *PaRFR*, we further propose a hypothesis on the relation between the number of top-ranked SNP patterns (frequent mutation patterns) and the severity of the Alzheimer's disease. Specifically, the more frequent mutation patterns an individual carries, the more severe the disease an individual has, which is supported by Alzheimers Disease Neuroimaging Initiative (ADNI) data. This work is summarized in the following manuscript to be submitted to a journal.

- **Yue Wang**, Limsoon Wong, Giovanni Montana. Parallel random forests regression on Hadoop for multivariate quantitative trait mapping. In preparation

Part of this work was done when I visited Imperial College London between Jan 2012 to Jun 2012.

Chapter 7:

This chapter discusses two other research problems in GWAS: tag SNP selection and SNP imputation. Tag SNP selection aims at selecting a small number of SNPs (called tag SNPs) from a large number of SNPs using the non-random association (linkage disequilibrium, LD) between SNPs. SNP imputation is used to impute the missing SNPs which may be caused by quality control or not being included in a genotyping chip. The imputed SNPs can be further used to study the association with the traits. The two problems are interlinked with each other. Tag SNP selection is usually used to design genotyping chips. Depending on the algorithms used, chips from different companies genotype a different set of "tag SNPs". SNP imputation can be applied to impute the values of different missing SNPs in different chips, thereby producing a unified set of genotyping data where all SNPs are present uniformly. The small number of genotyped tag SNPs also reduces genotyping cost. However, those genotyped tag SNPs may not be the "causal" SNPs in an association study. SNP imputation is applied to improve the chance of detecting "causal" SNPs.

Algorithms based on the r^2 LD statistic (defined in Equation 7.1) have gained popularity because r^2 is directly related to statistical power in detecting disease associations. Most existing r^2 based algorithms use pairwise LD. Recent studies show that multi-marker LD can help further reduce the number of tag SNPs. However, existing tag SNP selection algorithms based on multi-marker LD are both time and memory consuming. They cannot work on chromosomes containing more than 100k SNPs using length-3 tagging rules.

We propose an efficient algorithm called FastTagger to calculate multi-marker tagging rules and select tag SNPs based on multi-marker LD. FastTagger uses several techniques to reduce running time and memory consumption. Our experimental results show that FastTagger is several times faster than existing multi-marker-based tag SNP selection algorithms, and it consumes much less memory at the same time. As a result, FastTagger can work on chromosomes containing more than 100k SNPs using length-3 tagging rules. FastTagger also produces smaller sets of tag SNPs than existing multi-marker-based algorithms.

The generated tagging rules can be used for genotype imputation. We thus develop a rule-based imputation method called RuleImpute. To study the prediction accuracy of individual rules, we have proposed 5 different rule selection

strategies, and experimental results show that rules with minimum span give the highest prediction accuracy. This Chapter is based on the following papers:

- Guimei Liu, **Yue Wang**, Limsoon Wong. FastTagger: An efficient algorithm for genome-wide tag SNP selection using multi-marker linkage disequilibrium. *BMC Bioinformatics*, 11:66, February 2010.
- **Yue Wang**, Guimei Liu, Limsoon Wong. A study of different rule selection strategies for rule-based SNP imputation. Poster in *The 20th International Conference on Genome Informatics*,Yokohama, Japan, 14-16 December 2009.

Chapter 2

Background

2.1 Inherent expression: Genotype

The human genome consists of 23 chromosome pairs and some mitochondrial deoxyribonucleic acid (DNA). For every chromosome pair, one is from the mother and the other from the father. The first 22 chromosome pairs are called autosomes, and the remaining pair depends on the gender: for male it is a X and a Y chromosome and for female it is two X chromosomes. Each chromosome pair is made of DNAs and has a double helix structure formed by base-wise pairing of the two long strands of DNAs. In total, approximately 3.4 billion base pairs are aligned in 46 chromosomes. Some stretches of DNA nucleotides, called genes, are meaningful segments since they tell cells how to make proteins. Currently, there are ∼25,000 genes identified in the human genome [Stein, 2004]. The nucleotides on the two DNA strands in a chromosome, are paired in accordance to the Crick-Watson rule: adenine (A) is paired with thymine (T) and cytosine (C) is paired with guanine (G). The nucleotide base may be differ from individual to individual at the same location of a chromosome strand. Such a difference is called a Single Nucleotide Polymorphism (SNP). A rough estimate is that an SNP exists every 100∼300 nucleotide bases, leading to a total of 10-30 million potential SNPs in the human genome. According to the latest NCBI SNP statistics, there are 38,077,993 SNPs with validated information either supported by non-computational methods or by frequency information associated with them.

However, this number may go up or down since different researchers use different scrutiny criteria. Compared to other genetic variations like copy number variation, segment insertion and deletion, the amount of SNP genetic markers is considered more abundant and informative. The relationship between human genome, chromosome, gene and SNP is illustrated in Figure 2.1. The possible bases that can be observed at the locus of an SNP are called the alleles of that SNP. The alleles of a SNP is usually given as a pair due to its chromosome pair, and is called a genotype. In this thesis, we focus on biallelic SNPs, which are SNPs having only two alleles. The allele that appears in the majority of a population is termed the major allele, the other is called the minor allele. In our illustration, the genotypes of the 3 SNPs for the 1st, 2nd and nth sample are: (GC,CT,AG), (CG,CT,AG) and (GG,TT,GG). If a study population only consists of these 3 samples, then the major allele for first SNP is G since it occur 5 times in 3 samples. To ensure the allele is not too rare in the population, a minor allele frequency threshold like 1% is imposed for all SNPs. In this illustration, the minor allele frequency of all three SNPs passes this threshold.

2.2 Outward expression: Phenotype

A phenotype represents the outward expression of inherent genetic code for an organism. An outward expression is either an observable or visible characteristic, trait or behavior. Different Body Mass Indexes (BMI) for a study population constitute an example of visible traits. While other phenotypes like brain volume size change rate, which is not directly visible, can be observed by Magnetic Resonance Imaging (MRI) technology. Samples with the same genetic patterns may not lead to the same phenotypes and vice versa, since a phenotype is determined both by genotype and by natural environment. For example, a “mimicry” phenotype is mainly determined by interactions with the environment and the genotype plays a lesser role, while a Mendelian disease phenotype is mostly determined by genetic patterns and the environment plays a lesser role. These two types of phenotypes are illustrated in Figure 2.2. The relationship of genotypes

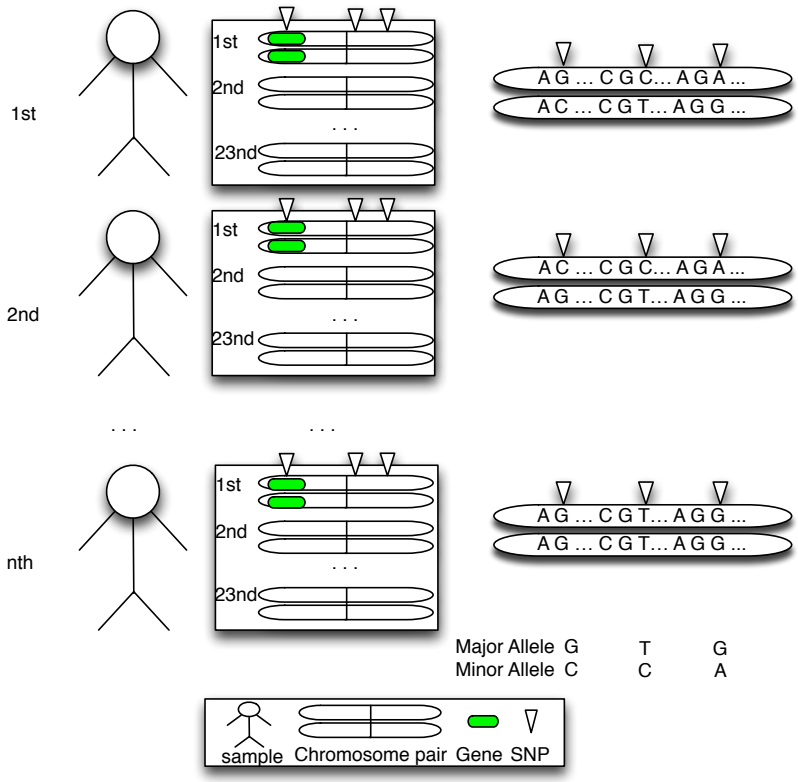

Figure 2.1: Illustration of genome, chromosome, gene and SNP. Here we omit the genetic information from mitochondrial DNA. Each sample in a population has 23 chromosome pairs, one is from his father and the other from his mother. A gene is labeled in one stretch of the first chromosome. Three SNPs are indicated by down triangles in the right part of the figure. Note that it is not necessary that SNPs reside in a gene region.

and phenotypes is described as follows [Herskowitz, 1977]:

$$Genotype + Environment + Genotype \times EnvironmentInteraction \rightarrow Phenotype$$

In this thesis, we assume that the environment covariates are properly adjusted. Therefore, we do not study how the environment and genotype×environment interaction affect the phenotypes, and the focus is to study the association relationship between genotypes and phenotypes as follows:

$$Genotype \rightarrow Phenotype$$

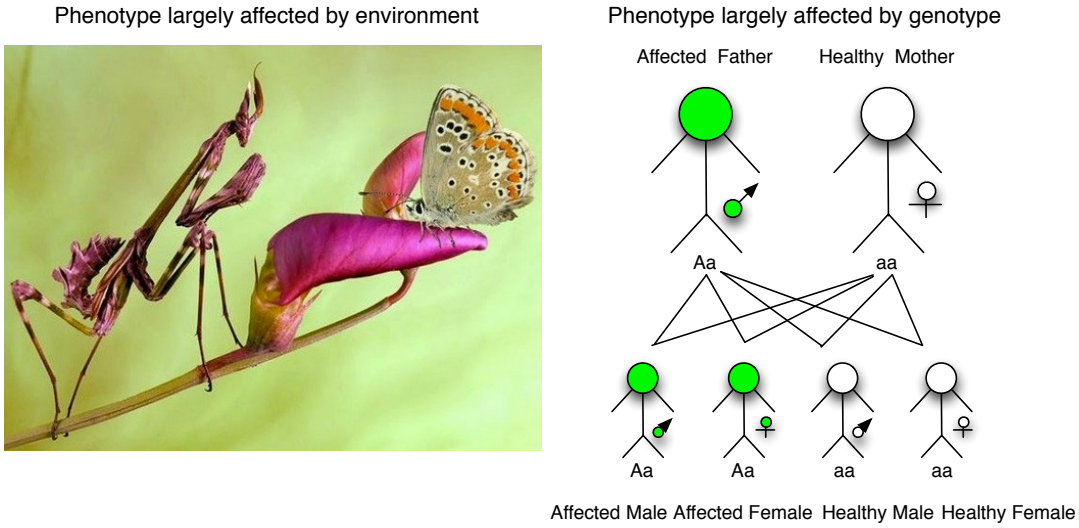

Figure 2.2: Illustration of two types of observable phenotype. Mimicry is largely determined by interactions with the environment and Mendelian's disease determined by genetic patterns.

Two types of phenotypes are studied in this thesis: case-control disease status and high-dimensional quantitative traits. The case-control disease status is used in a retrospective case-control study, where the healthy and disease samples are carefully selected so that their age, gender, race and other covariates are matched. The cases are those affected by the disease under the study and the controls are the healthy samples. The case-control phenotypes contain coarse information since only two states of disease information are recorded. In contrast, quantitative measurements of the phenotype can provide more information and get closer to representing the phenotype. For samples diagnosed with the same disease, the severity of disease varies person by person. We study the quantitative change of brain atrophy over time for a study group of Alzheimer's disease samples. 147,721 brain signatures located in various parts of the brain, called voxels, are selected to represent brain shapes. This phenotype information is recorded in a high-dimensional voxel value vector, each element of the vector summarizes the brain volume change rate over time. The two studied phenotypes are illustrated in Figure 2.3.

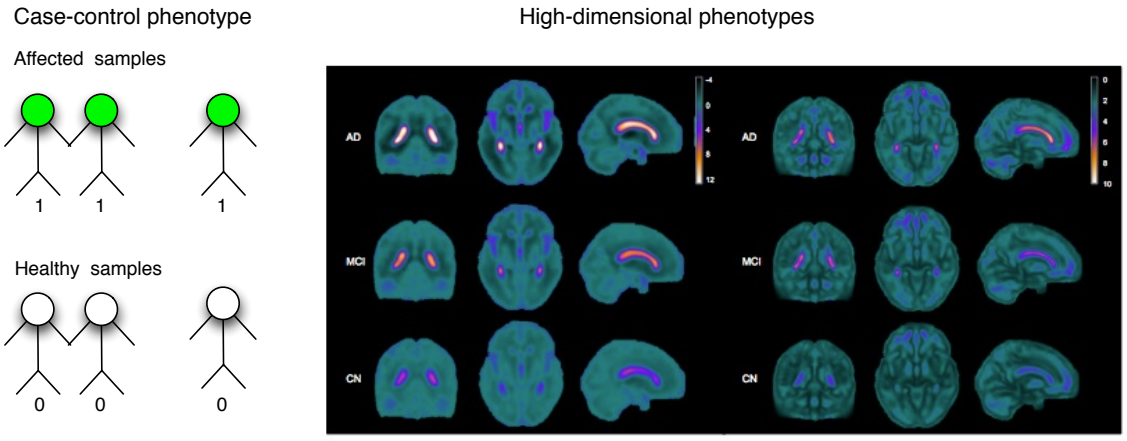

Figure 2.3: Illustration of two studied phenotypes, the case-control phenotype labels the disease status of a sample while high-dimensional brain image phenotypes record the change rate of brain volume size and thus are more close to the disease.

2.3 Overview of analysis flow of GWAS

Linkage studies [Bush and Haines, 2001; Pericak-Vance, 2001] have great success in identifying single genes of large effect which cause Mendelian diseases like neurofibro-matosis. However, there has been little progress in linkage studies of complex diseases. The design of such studies is usually limited to family members and the diseases studied are related only to family heritance (i.e, Mendelian diseases). The findings are hard to generalize to a population. Candidate gene studies [Zhu and Zhao, 2007] carefully select a few to hundreds of genetic variants based on the plausible and incomplete biology knowledge, and aims to test a researcher's proposed hypothesis. Unlike linkage studies and candidate gene studies, GWAS searches for susceptible genetic patterns that are associated with the study phenotypes from the whole genome in an unbiased way. The bedrock of GWAS relies on the "common disease, common variant" (CDCV) hypothesis [Chakravarti, 1998; Lander, 1996, 2001; Risch and Merikangas, 1996], which assumes that a common disease like diabetes and hypertension is caused by a set of common variations in some population. To define the common variations, the frequency of occurrence is at least 1% in a studied population [Buchanan*et*

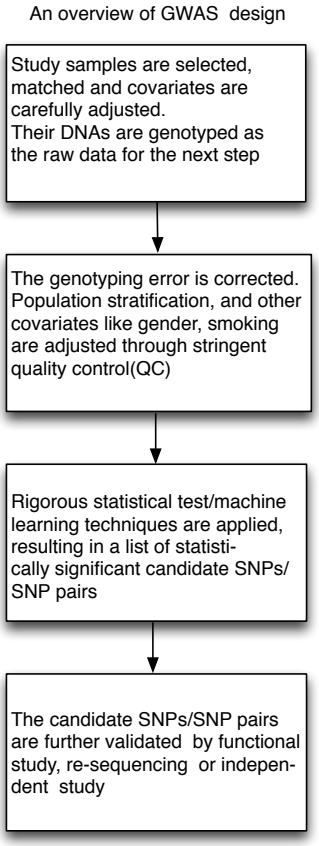

Figure 2.4: A typical workflow of case-control GWAS.

al., 2011]. The CDCV hypothesis has been criticized by the fact that a common/complex disease can also be caused by rare variants. Currently, there is no finalized conclusion on the exact distribution of disease-causing variant frequency. However, we are sure that disease etiology is far more complicated than we previously expected as we study more common/complex diseases. As a first step to elucidate the pathology of a complex disease, GWAS provides various clues to understand gene and pathway functions. As shown in Figure 2.4, the general workflow of GWAS can be divided into four steps: (1) study design, (2) quality control, (3) statistical test analysis, (4) results validation. They are described in the following sections respectively.

2.3.1 Study design

The study design of GWAS can be categorized into two types: retrospective study and prospective study. The most common retrospective study is the case-control study, in which the samples are selected as cases and corresponding counterparts as controls. Case-control study is efficient and cheap compared to prospective study. However, it assumes that the cases have the same severity of disease and controls are totally disease free. This assumption may cause spurious false negatives. In reality, the cases may have different severity and some controls may also be at a high risk to develop the disease. Thus quantitative phenotypes like brain image change rate [Stein *et al.*, 2010b], height [Estrada et al., 2009], blood pressure [Levy et al., 2009] are believed to better characterize disease status in some situations. To directly measure the risk of developing a disease and make less biases, a more expensive and time-consuming prospective study called a cohort study may be needed. It includes a representative group of samples with similar phenotypes of interest and genetic variants at the beginning of the study. Until certain time of the study progress, some samples developed the diseases. Their genetic patterns are compared with the other samples to detect the presence of any disease mutation pattern. Such studies have received more attention in GWAS recently [Cupples et al., 2007].

2.3.2 Quality control

Study samples are carefully selected and genotyped mainly using Illumina or Affimetrix chips. The genotype data obtained from the chips are raw data, which may contain genotyping errors. Genotyping errors may lead to spurious findings. Thus a set of quality control procedures [Teo, 2008] are used.

(1) A SNP calling threshold like 95% is applied to each SNP. The genotyping technology is not perfect, not every SNP is genotyped in every sample, and some genotypes are missing in some samples. The SNP calling value of a SNP is calculated as the proportion of samples for which the genotypes of this SNP are successfully determined. A calling threshold like 95% means that the genotypes of the SNP are successfully determined in at least 95% of samples. SNPs less than the calling threshold are removed. Even though a SNP satisfies the calling

threshold, there may still be some samples for which the genotypes of this SNP are not successfully determined. In such a situation, the missing genotypes can be imputed using methods such as IMPUTE [Howie *et al.*, 2009] and Plink [Purcell et al., 2007].

(2) An allele frequency threshold like 1% or 5% is applied to each SNP. An allele whose frequency is lower than the threshold is considered a rare allele. All the rare alleles of SNPs are removed because they are less likely to be responsible for the associated traits. For the rare disease, rare variants are important. However, the assumption of GWAS is common disease, common variants (CDCV) as mentioned in section 2.3.

(3) The significance of the Hardy-Weinberg equilibrium (HWE) test should be as stringent as 10^{-6}. HWE states that the frequency of allele and genotype in a population is constant from ancient generations to current generations. This is an ideal setting. In reality, many disturbing factors like mutation and non-random mating could occur.

Suppose the frequencies of a biallelic SNP are denoted as p (the major allele) and q (the minor allele). According to HWE, the equation $p + q = 1$ describes the frequency of a gene with two alleles, and the equation $p^2 + 2pq + q^2 = 1$ describes the frequency of the three possible genotypes of a gene with two alleles. The two homozygous genotypes have allele frequency p^2 and q^2 and the heterozygous genotype has frequency $2pq$. A χ^2 test is applied to compare the expected genotype distribution from HWE with the observed count of the three genotypes from the population. A χ^2 value lower than 10^{-6} is usually considered a strong evidence of deviation from HWE. SNPs that significantly deviate from HWE need to be removed because their expected proportions of genotypes are not consistent with observed allele frequency.

(4) Other than SNP quality control, sample quality controls are also used:

i) For a sample, the proportion of SNPs that are not successfully genotyped or removed—using criteria (1), (2) and (3)—should be below 5%. Otherwise, the sample is removed.

ii) When the proportion of heterozygotes is higher than the user-specified threshold, the samples may be contaminated or may contain related or duplicated samples.

iii) Race information should be consistent with the reported race.

iv) Gender information should be consistent with the reported gender.

(5) Population substructures is removed from the data. In GWAS, a majority of SNPs are not associated with disease, a strong association signal can be caused either by some true associated SNPs or by population structure. Before performing statistical association analysis of the genotype data, a tool like "Structure" [Pritchard et al., 2000] should be used to detect population substructures.

2.3.3 Statistical analysis

After the genetic data are properly cleaned, the next step is to conduct rigorous statistical analysis. Potential confounding factors like gender, smoking and drinking should be properly incorporated into the statistical model or adjusted before formal analysis. Otherwise, false positive associations may be detected. There are (1) single-SNP association analysis, (2) multi-SNP association test, and (3) SNP-SNP interaction test (epistasis). Different statistical tests are derived for these three tasks. Before we proceed to the description of the statistical tests, here are some assumptions :

(1) We assume each base pair has biallelic polymorphism.

(2) For a SNP, we write "A" as the major allele and "a" as the minor allele; therefore three genotype combinations AA, Aa, aa are used for a SNP.

(3) We use case-control disease status as phenotype when describing a statistical test.

2.3.3.1 Single-SNP association test

Single-SNP χ^2 test, also known as the homogeneity test or the genotypic test, is used to test the association with case-control status without assuming any relationship between genotype and case-control status. The null hypothesis and alternative hypothesis are respectively:

H_0: The proportion of case vs control is independent of the frequency distribution of the three genotypes.

Table 2.1: Single-SNP χ^2 test contingency table for the additive model

	AA	Aa	aa	Total
Case	$n_{1,0}$	$n_{1,1}$	$n_{1,2}$	$n_{1,}$
Control	$n_{0,0}$	$n_{0,1}$	$n_{0,2}$	$n_{0,}$
Total	$n_{,0}$	$n_{,1}$	$n_{,2}$	n

Table 2.2: Single-SNP χ^2 test contingency table for the recessive model

	AA	Aa or aa	Total
Case	$n_{1,0}$	$n_{1,1} + n_{1,2}$	$n_{1,}$
Control	$n_{0,0}$	$n_{0,1} + n_{0,2}$	$n_{0,}$
Total	$n_{,0}$	$n_{,1} + n_{,2}$	n

H_A: The proportion of case vs control is dependent on the frequency distribution of the three genotypes.

The two alleles in a genotype interact according to one of three possible effects: dominant, recessive, and additive. Given a SNP with two possible alleles (A and a). If AA and aa individuals show different phenotypes, and Aa individuals show the same phenotype as AA individual, then the allele A is said to be dominant and the allele a is said to be recessive. If instead Aa has the same phenotype as aa, the allele a is said to be dominant and A is recessive. On the other hand, the additive effect is the form of allelic interaction in which dominance is absent, resulting in a heterozygote that is intermediate in phenotype between homozygotes for the alternative alleles.

The observed genotype counts in the case group for AA, Aa, aa are $n_{1,0}$, $n_{1,1}$, $n_{1,2}$ and in the control group are $n_{0,0}$, $n_{0,1}$, $n_{0,2}$, respectively. The first number 0 and 1 indicates the control or case status, and the second number 0, 1, 2 indicates the number of minor allele copy in the genotype. The three contingency tables corresponding to the three kinds of allelic interaction effects are constructed as follows:

For a genotype with additive effect, Table 2.1 is used to calculate the χ^2 test statistics. The formula is :

Table 2.3: Single-SNP χ^2 test contingency table for the dominant model

	AA or Aa	aa	Total
Case	$n_{1,0} + n_{1,1}$	$n_{1,2}$	$n_{1,}$
Control	$n_{0,0} + n_{0,1}$	$n_{0,2}$	$n_{0,}$
Total	$n_{,0} + n_{,1}$	$n_{,2}$	n

$$\chi^2 = \sum_{i=0}^{1} \sum_{j=0}^{2} \frac{(n_{i,j} - n_{i,}n_{,j}/n)^2}{n_{i,}n_{,j}/n},$$

where $n = \sum_{i=0}^{1} \sum_{j=0}^{2} n_{i,j}$, $n_{i,} = \sum_{j=0}^{2} n_{i,j}$, and $n_{,j} = \sum_{i=0}^{1} n_{i,j}$.

The χ^2 test statistic follows asymptotically the χ^2 distribution with 2 degrees of freedom (df). If the null hypothesis is rejected at a nominal significance level like 0.05, then there exists an association between the SNP and the case-control disease status. However, we do not know which genotype increases or decreases the disease rate. A more detailed insight can be obtained by calculating the Pearson correlation coefficient. When the "aa" cell (rare variants) of the contingency table has count less than 5, the Fisher's exact test should be used because the approximation by the χ^2 test statistic is not accurate in this scenario.

For a genotype with the recessive or the dominant effect, the association between the SNP and the case-control disease status can be tested by aggregating the two columns of Table 2.1. For example, the null hypothesis and the alternative hypothesis for the dominant model can be restated as follow:

H_0: The proportion of case vs control is independent of the frequency distribution of the two genotypic groups (AA or Aa and aa).

H_A:The proportion of case vs control is dependent of the frequency distribution of the two genotypic groups (AA or Aa and aa).

The dominant and recessive test statistics follow the χ^2 distribution but with 1 df. In general, the dominant test has optimal power if the inherent genetic model is the dominant model.

Another commonly used test is the Cochran-Armitage trend test, which is

often preferred over allele frequency test. Both test the same null hypothesis but the latter assumes the HWE. The null hypothesis to be tested is:

H_0: the frequency for allele "a" in the case group is the same as in the control group.

The general formula for calculating the Cochran-Armitage trend test is :

$$Z_x = \frac{\sqrt{n}\sum_{i=0}^{2} g_i(\frac{n_{1,i}}{n_{1,}} - \frac{n_{0,i}}{n_{0,}})}{\sqrt{\frac{n(n_{1,}+n_{0,})}{n_{1,}n_{0,}}\left[\sum_{i=0}^{2} g_i^2\theta_i - (\sum_{i=0}^{2} g_i^2\theta_i)^2\right]}}$$

where θ_i=$\frac{n_{1,i}n_{0,i}}{n}$, i=0,1,2. The coding of <g_0,g_1,g_2> represents three genotype combinations of a SNP, like AA, Aa, and aa. To encode AA, Aa and aa, there are three main models: additive, recessive, and dominant model. Normally, if users choose additive model, then <g_0,g_1,g_2> is coded as <0,1,2>. If users choose recessive model, then <g_0,g_1,g_2> is coded as <0,1,1>, and for dominant model <g_0,g_1,g_2> is coded as <1,1,0>. Although there are three coding schemes, the additive coding scheme is recommended in the literature since it does not assume the high-risk allele "A" in the dominant model and "a" in the recessive model. However, Li et al. [2009] proposes a MAX statistic which uses the maximum of the three genetic models to determine the p-value since the true underlying genetic model is unknown. Their experiments show that the MAX statistics has high power. The above Cochran-Armitage trend test statistic Z_x follows the normal distribution with mean 0 and standard deviation 1, and the null hypothesis is rejected when $|Z_x| > 1.96$. This test is equivalent to the test statistic derived from the standard logistic regression model with following form:

$$\log\left(\frac{Pr(case|g)}{1 - Pr(case|g)}\right) = \alpha + \beta g + \gamma z$$

where z can be a covariate like sex, drinking quantity and g represents the genotypic coding. The null hypothesis H_0 is $\beta = 0$.

2.3.3.2 Multi-SNP association test

Single-SNP test assumes that different SNPs are independent and therefore assert no joint effect on the phenotype. It simplifies the genetic cause of disease, while a

multi-SNP association test is considered more informative and powerful because it takes into account the joint effect of SNPs [Kim et al., 2010; Slavin et al., 2011]. One natural extension from single-SNP association to multi-SNP association is the multivariate logistic regression model:

$$\log\left(\frac{Pr(case|g)}{1-Pr(case|g)}\right) = \alpha + \sum_{i=1}^{l} \beta_i g_i + \gamma z$$

The subscript in the above model indicates the ith SNP. The null hypothesis H_{0i} for the ith SNP is: $\beta_i = 0$. If some hypotheses are rejected, the corresponding SNPs are associated with the disease jointly.

A test like the χ^2 test and the Fisher's exact test can be easily extended from being a single-SNP association test to become a multi-SNP association test. For a three-SNP association test constructed by the contingency table with 81 columns (df 80), most of the cells are sparse if the sample size is not large enough. The power of such a test drops significantly. Currently such a retrospective association test is limited only to two-SNP association. For a prospective model like logistic regression, a similar problem also exists when the number of SNPs is far more than the sample size, as the estimation of coefficients is problematic.

To overcome these issues, the assumption that only a small number of SNPs are associated with the phenotype is imposed, and a penalty term can be incorporated to estimate the coefficients. In the literature, the lasso and ridge penalty regression have been proposed [Liu et al., 2011b; Wu et al., 2009; Zhao and Chen, 2012]. Another method is to select a small number of SNPs to consider before multivariate modeling, thus dimensional reduction techniques like least-squares kernel machines [Kwee et al., 2008], partial least-squares [Xue et al., 2012; Zhang et al., 2011] and principle component analysis have been used [Wang and Abbott, 2008].

A variety of powerful Bayesian modeling methods have also been proposed [Albrechtsen et al., 2006; Kilpikari and Sillanp, 2003; Marttinen and Corander, 2010]. These methods are powerful in the sense that their modeling framework can easily include diverse prior knowledge, handle missing data, avoid multiple testing correction procedure, control false-positive and avoid model overfitting. Su et al. [2009] develop a novel association test by incorporating genealogy infor-

mation from the HapMap data to improve the signal detection. Logsdon et al. [2012] provide a Bayes statistic to control the false-positive and model complexity. Stephens and Balding [2009] present an overview of Bayesian modeling methods and discuss their advantages over frequentist statistics.

2.3.3.3 SNP-SNP interaction test (Epistasis)

Upto June 2012, there are 1,287 publications and 6,499 reported SNPs associated with over 300 traits or diseases. These reported SNPs still explain a small proportion of heritability. In the search of "missing heritability", SNP-SNP interaction detection plays an important role. The concept is originally proposed by Bateson [1909] who defines it as the effect of one gene masking the effect of another. Moore [2005] presents a global view of epistasis and classifies the epistasis into three levels:

- Genetic epistasis is the interaction of DNA sequence variants so that the phenotype is affected through a set of biological processes ranging from transcription to physiological homeostasis.
- Biological epistasis is the interactions between proteins and other biomolecules that affect final expressed phenotype in a sample.
- Statistical epistasis results from genetic epistasis and biological epistasis occurring in a population. It uses statistical language to describe the biological epistasis and detects it using statistical methods.

Here we focus on statistical epistasis detected through a statistical test. Without special notation, we refer to epistasis as statistical epistasis in the rest of the thesis.

Cordell [2002] defines statistical epistasis as a deviation from the additive linear model. If there is no epistasis, the linear model is additive since there is no interaction term. To assess such a deviation, a natural extension from the logistic regression model is to add an interaction term to the model and test the significance of the coefficient:

$$\log\left(\frac{Pr(case|g)}{1-Pr(case|g)}\right) = \alpha + \beta_1 g_1 + \beta_2 g_2 + \beta_{1,2} g_1 * g_2 + \gamma z$$

The subscript in the above equation indicates the 1st and 2nd SNP. The null hypothesis H_0 for this statistical epistasis is: $\beta_{12} = 0$. If the hypothesis is rejected, the corresponding SNP pair (epistasis) is associated with the disease.

In some loose situation [Wan et al., 2010b], the definition of epistasis allows for incorporating the association of a single SNP. Therefore, the null hypothesis is: $\beta_{12} = \beta_1 = \beta_2$. This is in contrast to a pure epistasis test [Wan et al., 2010a] which only considers the interaction term. Therefore, when there is a single association in the genetic model (main effect), the power of the test which considers the association of single SNPs is expected to be higher than that of the pure test.

Other than tests derived from statistical models, recently a new statistic called IndOR [Emily, 2012] has been proposed based on the biological definition of epistasis [Cordell, 2002], where epistasis is defined as a departure from independence between the two loci of interest. The name IndOR stands for independence-based odds ratio, and it uses odds ratio to measure the amount of independence between SNP pairs in cases and controls. The test statistics is shown globally more powerful than a likelihood ratio test derived from logistic regression model [Wan et al., 2010a] in 45 different simulated epistasis models. Emily [2012] extends this further to search for the missing heritability caused by epistasis.

Another group of methods without using statistical model and test are called model-free methods. The Multifactor Dimensionality Reduction (MDR) method [Ritchie et al., 2001] is the most popular one. Using two-SNP epistasis as an example, the basic idea is to group the 9 genotypic combinations into high-risk and low-risk groups based on the proportion of cases in each genotypic combination. This technique reduces a two-SNP pair with 9 combinations to a one dimension variable with two states. The newly transformed variables are used as predictors and the classification performance is evaluated by cross-validation. The one with the most accurate performance is selected as the most "significant" epistasis. The method is good at handling high dimensional interaction especially when the epistasis involves more than two SNPs. Also, the computational burden is relatively lesser than classical statistical methods. However, there are many issues

not addressed in the original MDR paper like handling an imbalanced case-control data set [Velez et al., 2007]. Since then, many variants have been derived from this work [Bush et al., 2006; Chung et al., 2006; Dai et al., 2012; Fang and Chiu, 2012; Lee et al., 2007; Motsinger and Ritchie, 2006; Pattin et al., 2009].

2.3.4 Validation of results

The rigorous statistical analysis—as described in the previous subsection—provides a ranked list of SNPs or epistasis. Only some of them are the biological links to the phenotypes of interest, the rest are false positives. Therefore, a replicated study is often used to validate the discoveries in an independent sample. For example, Sim et al. [2011] conduct a GWAS on Chinese, Malay and Indian type 2 diabetes mellitus populations and the discovered SNPs can serve as an independent validation of the SNPs identified in type 2 diabetes mellitus European population. However, not all results can be replicated easily due to population stratification, quality control, sample selection bias and other factors. To overcome the inconsistency between different studies, a large cohort is often needed [NCI-NHGRI Working Group, 2007]. When there are some susceptible loci validated in different studies, a more rigorous functional study is conducted. A general approach is to study LD where the susceptible loci may not show obvious function or resides in a gene "desert". Studying the SNPs in LD is inspired by the observation that some loci are inherited together and may serve similar function. The SNPs in high LD with susceptible SNPs may yield some new biological discovery [Yeager et al., 2007]. A further re-sequencing of the region where susceptible loci reside can help identify rare variants with more functional implication [Libioulle et al., 2007]. Some other functional studies like knockout or knock-in [Frayling and McCarthy, 2007] are also used to uncover function implicated by plausible SNPs.

2.4 Big data and Hadoop technologies

Big data means the storage, integration and analysis of data whose volume is beyond the computational capability of a single PC. The processing of big data demands new technologies that are especially designed for them. Big data tech-

Table 2.4: Summary of different extended R packages or technology

Packages	Description
RODBC	An interface to connect SQL-based Database for data manipulation
biglm	A solution to build generalized linear models on big data
ff	Providing fast access functions to data that cannot be loaded into memory
bigmemory	Offering an efficient and parallel data mining techniques for large-scale data
snow	A high-level interface using a workstation cluster for parallel computations in R
SPRINT	A parallel framework designed on high performance computing
Revolution R	An efficient, optimized and scalable R implementation

nologies are used to integrate, model, analyze, share and visualize big data to extract potential insights. Some of these popular technologies are developed originally to analyze data that can be loaded into a single PC. They are gradually extended to handle the emerging big data. For example, R is well known for data analysis and visualization in computer science, finance and other disciplines. Traditionally it works using random access memory, and this causes serious scalability issues. Recently new packages like "RODBC" "biglm","ff","bigmemory", "snow" and parallel frameworks like "sprint" are developed to resolve issues caused by big data. "Revolution R"—a scalable, efficient and optimized R distribution—is designed specifically for heavy computation of big data. The description of these packages are listed in Table 2.4.

Other technologies are developed for big data analysis. The software platform from Palantir, is now adopted in finance companies and government agencies. Its successful applications include detecting financial fraud[1] and informing the outbreak of disease in Haiti[2] by using its powerful integration and visualization technology. Hadoop is a software framework for large-scale distributed data analysis under the Apache Software Foundation. Originally it is derived from two published papers on "MapReduce" and "Google File System" [Dean and Ghemawat,

[1] http://www.palantir.com/tag/anti-fraud/
[2] http://www.palantir.com/2010/04/haiti-effective-recovery-through-analysis/

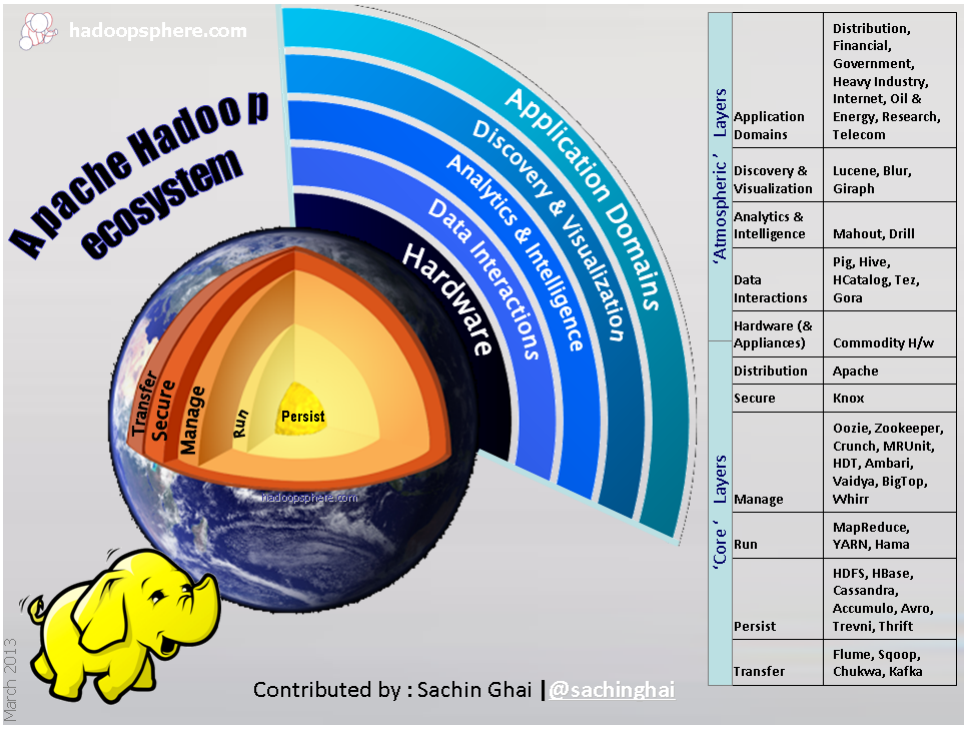

Figure 2.5: The recent Apache Hadoop ecosystem diagram from hadoopsphere.com.

2004; Ghemawat et al., 2003], but now it has evolved as an ecosystem consisting of different sub-systems. Hadoop performs especially well at handling scalable, fault-tolerant and efficient large-scale data analysis. Because of these advantages, it becomes a standard technology used in many industries which require big data analysis. Figure 2.5[1] illustrates the Hadoop ecosystem and Table 2.5 describes the functions of the different components.

In this thesis, we only use the Hadoop Distributed File System (HDFS) and MapReduce for GWAS analysis. In the next two sections. the working principle of these two components is described.

[1] http://www.hadoopsphere.com/2013/03/apache-hadoop-ecosystem-march-2013_12.html

Table 2.5: Summary of different components in Hadoop Ecosystem

Component name	Description
Whirr	Provide a set of API/libraries to run different cloud services
ZooKepper	A high-performance coordination service for distributed applications
Sqoop	A data import mechanism to import/export structured data from RMDBS
Nutch	Used for importing data from Web
Flume	A data import mechanism to import/export stream data
Hive	A data warehouse infrastructure that provides data summarization and ad hoc querying
Hbase	A scalable, distributed database that supports structured data storage for large tables
Pig	A high-level data-flow language and execution framework for parallel computation
Mahout	A scalable machine learning and data mining library for large-scale data
HDFS	A scalable, portable and fault-tolerant filesystem for distributed computing for large-scale data
MapReduce	A functional programming library for large-scale data analysis for large-scale data

2.4.1 HDFS

HDFS is designed for large data storage and streaming data access on commodity hardware. Its design is suitable for an application to read all the data many times and perform analysis on the data instead of random data access or write data many times. It is also well applicable when partial data analysis can be done independently and later be aggregated into final results. Like other filesystems, there is a minimum unit (by default, this is 64MB) of reading and writing called a block in HDFS. A normal file, when loaded into HDFS, is divided into block-sized chunks, which are stored as blocks in HDFS. When a file is too large to be loaded in any node, the blocks of the file can be stored in different nodes of a Hadoop configured cluster through network transfer, which is transparent to the users. The design of block unit storage is also for the sake of fault-tolerance. For large-scale data analysis, it is common that part of the data is not available or

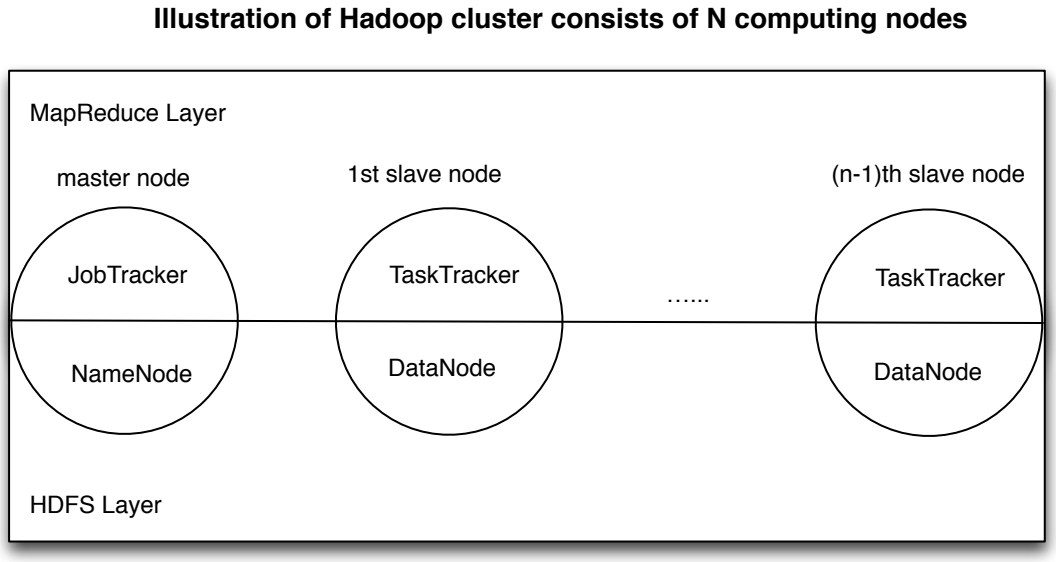

Figure 2.6: Hadoop cluster consists of NameNode and DataNode for HDFS and JobTracker and TaskTracker for MapReduce.

is corrupt due to task failure. Therefore, it is necessary to detect and restore the broken blocks and restart the analysis if there are replicates available in other computing nodes. In fact, HDFS has a block replication mechanism so that each block is replicated to a user-defined number (the default is 3) of different nodes in a Hadoop cluster.

HDFS considers the Hadoop cluster as consisting of two types of storage nodes: namely, a single NameNode and the rest are DataNodes, as illustrated in Figure 2.6. The NameNode works as a master of the file system, which is responsible for splitting data into blocks and distributing the blocks to the DataNodes with replication for fault tolerance. The NameNode stores two types of files: namespace image and the edit log. Namespace image keeps the filesystem tree and metadata like which blocks of a file is located in which DataNode. The edit log keeps the history of file system modifications. Both files are important for file system recovery. DataNodes, physically store file blocks allocated by the NameNode. They communicates with the NameNode through heartbeat to notify the NameNode that they are working. DataNodes can communicate with each other to move replicates from one to another or replicate blocks when requested. Additionally,

a secondary NameNode can be configured in a Hadoop cluster. The secondary NameNode is not to replace the NameNode when it fails to work. Rather, it works as a helper node for the NameNode to periodically build snapshots of the namespace image file and the edit log. When the most serious single point of NameNode fails, these snapshots are used to recover the filesystem structure by applying actions in the edit log to the namespace image file. However, since the snapshots are kept periodically, the recovered file system may not be the exact filesystem structure when the NameNode fails.

The NameNode keeps meta information of blocks. The meta information tell the DataNodes to make the computation done at the block's node instead of transferring the data to other DataNodes. This move-codes-to-data design largely reduces the traffic of unnecesary data transfer over the network and, thus, improves efficiency. This data locality design makes HDFS outperform other file systems when running large-scale computationally intensive analysis.

2.4.2 MapReduce

MapReduce Architecture Under the MapReduce framework, the system architecture of a cluster consists of two kinds of nodes: JobTracker node and TaskTracker node. A JobTracker usually runs on the NameNode and keeps track of job information, job execution and job fault tolerance in the cluster. A job may be split into multiple tasks, each of which is assigned to a TaskTrack node. A TaskTracker node is responsible for task execution and communication with the JobTracker. In a Hadoop cluster, it is common to assign a master node both as NameNode and JobTracker, and slave node both as a DataNode and TaskTracker, as illustrated in Figure 2.6.

MapReduce Computational Paradigm The MapReduce computational paradigm exploits parallelism by dividing a running job into smaller tasks, each of which runs on a task node. The computation of MapReduce follows a fixed model with a `map` phase followed by a `reduce` phase. The MapReduce library is responsible for splitting the data into chunks and distributing each chunk to processing units (called `mappers`) on different nodes. `Mappers` process the data read from

the file system and produce a set of intermediate results which are shuffled to other processing units (called `reducers`) for further processing. Users can set their own computation logic by writing the `map` and `reduce` functions in their applications. The input and output of `map` and `reduce` function are $(key, value)$ pairs. The read and write of a file need to be transformed into the read and write of $(key, value)$ pairs, which is the essential design of such functional programming.

Map Phase The `map` function is used to process the $(key, value)$ pairs $(k1, v1)$ which are read from data blocks. The input set of $(k1, v1)$ pairs are transformed into a new set of intermediate $(k2, v2)$ pairs through the `map` function. The MapReduce library then sorts and partitions all the intermediate pairs and pass them to `reducers`.

Shuffling Phase The partitioning function is used to partition the emitted pairs from the `map` phase into M partitions on local disks, where M is the total number of `reducers`. The partitions are then shuffled to the corresponding `reducers` by the MapReduce library. Users can specify their own partitioning function.

Reduce Phase The intermediate $(k2, v2)$ pairs with the same key that are shuffled from different `mappers` are sorted and merged together to form a values list. The key and the values list are fed to the user-written `reduce` function. The `reduce` function operates on the key and values to produce a new $(k3, v3)$ pairs. The resulting output $(k3, v3)$ pairs are written back to the file system.

MapReduce Workflow Figure 2.7 gives an example of how MapReduce works. The task of the example is to count the frequency of each word in a very large text file from the Web. The input is a text file with each line containing some words, therefore the key is the line number and value is one-line string. The file is so big that it is not possible to store it in single PC. Using HDFS, the file can be divided into different blocks and stored in different disks in a cluster. The figure only shows one split of the data that is fed into a `map` task. The `map` function in this example splits each line into different words and gives each word one count.

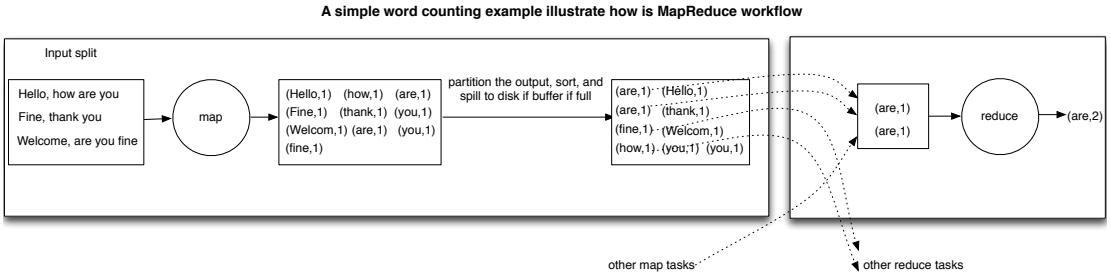

Figure 2.7: A simple word counting example to illustrate the work flow of MapReduce

This is done at the `map` phase. The output of `map` are partitioned according to the `reducers` they are sent to and, within each partition, the output is sorted by the key. When the `map` output size is larger than the memory buffer of the map task, the output is spilled into disks and then merged into a single sorted map output file. This file is fetched by reducers through HTTP. At the `reduce` phase, each `reducer` receives input data from different map tasks. So a `reduce` task needs to copy the output of the corresponding map tasks when the latter finish. In this example the `reduce` function just sums up the value of each element of the value list in <key, value list> and outputs the count of each word.

Chapter 3

An empirical comparison of several recent epistatic interaction detection methods

3.1 Introduction

A genome-wide association study (GWAS) examines the association between phenotypes and genotypes in a study group. The first exciting finding was on age-related macular degeneration (AMD) [Klein, 2005], which uncovers a disease allele (tyrosine-histidine polymorphism) with an effect size of 4.6 in ∼100,000 single-nucleotide polymorphisms (SNPs). Since then, over 600 GWAS have been conducted for 150 diseases and traits; and ∼800 associated SNPs have been reported. The methodologies of these studies are similar: A quality control criteria is first defined to filter the genotype data; then the remaining genotypes are each tested for association with the disease phenotypes. Finally, the significant SNPs are reported after multiple-testing correction. Most of these GWAS could only identify disease alleles with moderate effect size. Thus, single-SNP association studies could explain very limited heritability of these diseases [Emahazion et al., 2001]. Consequently, researchers have started exploring multi-SNP interactions in the hope of discovering more significant associations. Multi-SNP interactions are also called "epistatic interactions". This term originated from Bateson's definition of

epistasis one hundred years ago [Bateson, 1909]. It was defined as the change of segregation ratio and the interaction of genes. However, in the current literature, there is a debate on the exact definition of epistasis [Phillips, 1998, 2008]. Our paper focuses on evaluating epistatic interaction detection methods in their computational aspect and all the experiments are based on simulation data. Thus, we consider epistatic interactions as the statistically significant associations of k-SNP interaction ($k \geq 2$) with phenotypes.

There are mainly two types of epistatic interaction detection methods: model-based methods and model-free methods. In general, model-based methods [Wan et al., 2010a; Wu et al., 2010, 2009; Yang et al., 2009] predefine a statistical model between phenotypes and genotypes; then they fit the data to the model; and finally they output the significant SNPs. They work well for only a small number of important and filtered candidate SNPs; but they often fail when the number of SNPs grows to hundreds of thousands. To make model-based methods more efficient, researchers have proposed a variety of heuristic and filtering techniques. For example, Wan et al. [2010a] develop an upper bound of the likelihood ratio test statistic for two-locus epistatic interaction to prune the search space and a Boolean transformation of data to make collection of contingency table information faster. As another example, Wu et al. [2010] devise a two-stage analysis so that the overall analysis is more efficient. As a third example, Yang et al. [2009] use a stochastic search to identify only 40-50 (set by the user) groups of candidate epistatic interactions for follow-up model-fitting analysis.

In contrast, model-free methods [Ritchie et al., 2001; Wan et al., 2010b; Zhang et al., 2010] have no prior assumption on the data and the model. Given the genotype data, these methods only examine the test statistic of each possible epistatic interaction with phenotypes. Zhang et al. [2010] propose a minimum spanning tree (MST) structure to represent the data; by traversing this MST, exhaustive search of every epistatic interaction is an order faster than that of brute-force search. Wan et al. [2010b] connect the epistatic interactions with predictive rules and use a rule mining strategy to find epistatic interactions.

Our evaluation study of epistatic interaction detection methods is different from earlier studies such as Motsinger-Reif et al. [2008b], Motsinger-Reif et al. [2008a] and Sucheston et al. [2010]. Firstly, Motsinger-Reif et al. [2008a] com-

pare only approaches based on neural networks while our selected methods cover both data mining and statistical methods. Secondly, Motsinger-Reif et al. [2008b] evaluate multifactor dimensionality reduction (MDR) [Ritchie et al., 2001], grammatical evolution neural networks (GENN) [Motsinger-Reif et al., 2006], focused interaction testing framework (FITF) [Millstein et al., 2006], random forests (RF) [Breiman, 2001], and logistic regression (LR) [D. W. Hosmer, 2000] methods. They show that MDR is superior in all settings. After two years of advancement, most methods selected in this chapter have demonstrated that their performance is better than that of MDR; we therefore omit discussing methods mentioned in Motsinger-Reif et al. [2008b]. Thirdly, Sucheston et al. [2010] compare AMBIENCE [Chanda et al., 2008] with MDR, restricted partitioning method (RPM) [Culverhouse, 2007] and logistic regression. They conclude that the performance of AMBIENCE is equivalent to that of logistic regression for two-locus models and better than that of RPM and MDR. However, according to Wan et al. [2010a], the performance of BOOST is better than that of PLINK [Purcell et al., 2007], which uses a pure logistic regression model. Therefore we omit the evaluation of AMBIENCE and RPM in our study. Lastly, Wan et al. [2010b] and Yang et al. [2009] have shown that their overall performance is much better than that of BEAM [Zhang and Liu, 2007]. We thus omit BEAM.

In this chapter, we give an independent empirical comparison of five methods for detecting epistatic interactions—namely, TEAM [Zhang et al., 2010], BOOST [Wan et al., 2010a], SNPRuler [Wan et al., 2010b], SNPHarvester [Yang et al., 2009], and Screen and Clean [Wu et al., 2010]—to help users better understand which method is more suitable for their data, which method is good for

Table 3.1: Summary of the features of the five methods: BOOST (B), TEAM (T), SNPRuler (SR), SNPHarvester (SH), Screen and Clean (SC)

	B	T	SR	SH	SC
Exhaustive Search	×	√	×	×	×
Logit Model Assumed	√	×	×	√	√
Multi-Stage	×	×	×	×	√
Permutation Test	×	√	×	×	×
Bonferroni correction	√	×	√	√	√
Programming language	C	C++	Java	Java	R

detecting epistatic interactions with and without main effect, and which method is scalable to larger datasets. We also analyze why combining several of these methods cannot enhance power. Their basic characteristics are given in Table 3.1.

The organization of this chapter is as follows. We first formulate the problem in Section 3.2. Then we briefly introduce each of the five methods in Section 3.3. We describe how the evaluation data is simulated in Section 3.4 and the detailed setting of each experiment in Section 3.5. After that, we present the results under each setting in Section 3.6. Finally, we discuss the performance of each method and provide advice to users in Section 3.7.

3.2 Problem formulation

In a typical GWAS, researchers collect two types of data: genotype data that encode the genetic information of each individual, and phenotype data that measure the quantitative traits of each individual. Here, we consider only bi-allelic SNPs. The allele that occurs more frequently is called the major allele, denoted as A. The allele that occurs less frequently is called the minor allele, denoted as a. The two alleles form three genotypes—AA, Aa and aa—and they are encoded as 0, 1 and 2 in raw data. For phenotype data, we consider the binary form (0 for control and 1 for case). With minor modification, current methods can handle other types of phenotype data, e.g., by discretizing a continuous phenotype.

The goal of each method is to identify k-SNP ($k \geq 2$) epistatic interactions significantly associated with the phenotype. Thus, each method outputs a list of epistatic interactions, each involving up to k SNPs (usually k is set to 2) and is accompanied by its P-value after correction for multiple testing.

There are two challenges. First, if we constrain k to be 1, then the number of statistical tests is equal to the number of SNPs in a dataset. When k increases by 1, the number of tests grows by n-fold (n is the number of SNPs in a dataset). Thus, the total number of tests grows quickly as k increases, resulting in the inability of current methods to test all the combinations. For example, to study a moderate size of 500,000 SNPs, we can test only two-locus epistatic interactions if we use the EPISNP program [Ma et al., 2008] on a 2.66GHz single processor, as it may take 1.2 years to finish all the tests. Therefore, heavy computation cost

is one of the challenges for current methods [Wang et al., 2011]. Second, since a huge number of possible combinations are tested, a large proportion of significant associations are expected to be false positives. Thus, reducing the number of false positives while retaining power is another challenge.

3.3 Methods

3.3.1 SNPRuler

SNPRuler [Wan et al., 2010b], MDR [Ritchie et al., 2001], and a few other pattern-based methods [Li et al., 2006; Long et al., 2009] adopt data mining approaches for detecting epistatic interactions. These methods do not assume a model-fitting procedure but use some filtering methods to reduce the number of SNP combinations to be tested. SNPRuler [Wan et al., 2010b] is a rule-based approach motivated by the fact that each epistatic interaction induces a set of rules. For example, $\text{SNP}_1 \wedge \text{SNP}_2 \Rightarrow$ Disease contains 9 rules, they are $\text{SNP}_1 = i \wedge \text{SNP}_2 = j \Rightarrow$ Disease, $i, j \in \{0, 1, 2\}$. In the paper, the quality of a rule is given by its χ^2 test value. We define $\text{SNP}_1 \wedge \text{SNP}_2 \Rightarrow$ Disease as a SNP-level epistatic interaction and $\text{SNP}_1 = i \wedge \text{SNP}_2 = j \Rightarrow$ Disease, $i, j \in \{0, 1, 2\}$ as allele-level epistatic interactions. To identify epistatic interactions that are significant, SNPRuler traverses a set enumeration tree where the nodes of the tree are the genotypes of the SNPs, the leaves of the tree are the phenotypes, and the path from the root to a leaf is an allele-level epistatic interaction. Exhaustive tree traversal is theoretically possible but practically impossible due to the explosive number of combinations as the tree grows. Therefore, the authors propose an upper bound on the χ^2 test statistic to prune the search space. After the search procedure, a post-processing step is used to get and rank SNP-level interactions. There are two hidden problems in this work. First, the upper bound they derived from the χ^2 formula is not a true upper bound and does not possess the anti-monotone property [Agrawal and Srikant, 1994]. Although it helps prune a large search space, it also prunes many true-positive epistatic interactions. Second, the upper bound is based on the assumption that the number of cases should be larger than or equal to that of controls in a dataset; otherwise, the upper bound does not

hold. This assumption is inconvenient since the number of controls is larger than that of cases in most GWAS datasets.

3.3.2 SNPHarvester

SNPHarvester [Yang et al., 2009] is a stochastic search algorithm to identify epistatic interactions. It consists of two steps: a filtering and a model-fitting step. The filtering step is to identify m (40–50) significant SNP groups for the subsequent model-fitting step. In the filtering step, it first removes significant single SNPs according to their χ^2 test values, because this method is only interested in epistatic interactions that have weak marginal effect but significant joint effect. Then it randomly picks k SNPs. These form an active set $S = \{SNP_1, SNP_2, ..., SNP_k\}$. The rest of the SNPs form a candidate set S_c. After all these preparations are done, the nested *PathSeeker* algorithm is called to swap $SNP_i \in S$ with $SNP_j \in S_c$ to get the group with the highest χ^2 test value. A total of $k(n-k)$ combinations need to be tested to identify such a group. After this, the identified group is removed from the n SNPs. The next iteration continues to select k SNPs to form an active set and the remaining $n-2k$ SNPs form a candidate set. The same procedure is repeated again. The complexity to identify m groups is $O(knm)$, which is affordable even when there are $> 100,000$ SNPs. In the second step, each of the m significant groups is fitted into the L_2 penalized logistic regression model; see [Park and Hastie, 2008] for details.

3.3.3 Screen and Clean

The Screen and Clean method [Wu et al., 2010] uses a two-stage analysis; datasets from stage 1 for the screening and datasets from stage 2 for the cleaning. In the screening stage, it only considers tag SNPs and marginal significant SNPs. These SNPs are first fitted into the main effect lasso logistic regression model

$$g(E[Y|X]) = \beta_0 + \sum_{j=1}^{N} \beta_j X_j$$

where X_j is the encoded genotype value 0, 1 or 2, Y is the encoded phenotype value 0 or 1. This model first identifies a set of SNPs whose coefficients satisfy $\beta_j \neq 0$, $j \in \{1,2,...,n\}$; then it obtains the least square estimates $\hat{\beta}_k$, $k \in \{1,2,...,n\}$ of these SNPs. To test the significance of each regression coefficient, the t-test statistic value is calculated. Only the significant SNPs and their corresponding two-SNP combinations enter the interaction model

$$g(E[Y|X]) = \beta_0 + \sum_{j=1}^{N} \beta_j X_j + \sum_{i<j;i,j=1,...,N} \beta_{ij} X_i X_j.$$

A similar procedure applies to interaction model fitting. After this stage, the "surviving" SNP pairs go to the second cleaning stage for controlling type-1 error. T-test is used again to remove SNP pairs whose significance level is lower than a user-specified threshold.

3.3.4 BOOST

BOOST [Wan et al., 2010a] contributes to the epistatic detection problem in two aspects. Firstly, it provides a new Boolean representation of the data. By transforming the data representation to the Boolean type, BOOST uses established methods [Wegner, 1960] of logic operations to collect contingency table information, which is very efficient. Secondly, it proposes an upper bound for the likelihood ratio test statistic to prune insignificant epistatic interactions. The likelihood ratio test is originally based on the deviance of difference between the full logistic regression model

$$\log \frac{P(Y=1|X_{l_1}=i, X_{l_2}=j)}{P(Y=2|X_{l_1}=i, X_{l_2}=j)} = \beta_0 + \beta_i^{X_{l_1}} + \beta_j^{X_{l_2}} + \beta_{ij}^{X_{l_1}X_{l_2}},$$

X_{l_1} and X_{l_2} are genotype variables, $i, j \in \{0,1,2\}$, and the main logistic regression model

$$\log \frac{P(Y=1|X_{l_1}=i, X_{l_2}=j)}{P(Y=2|X_{l_1}=i, X_{l_2}=j)} = \beta_0 + \beta_i^{X_{l_1}} + \beta_j^{X_{l_2}}.$$

We denote the log likelihood of the full model under maximum likelihood estimate (MLE) as $\hat{L_F}$, the log likelihood of the main model under MLE as $\hat{L_M}$, the

log likelihood of log-linear saturated model as $\hat{L_S}$, and the homogeneous model as $\hat{L_H}$. The likelihood ratio statistic between the main model and the full model is $-2(\hat{L_M} - \hat{L_F})$. The log-linear homogeneous association model corresponds to the main logistic regression model and the log-linear saturated model corresponds to the full logistic regression model [Agresti, 2002]. This leads to an upper bound for the two log-linear models: $-2(\hat{L_S} - \hat{L_H})$. Matsuda [2000] uses Kirkwood Superposition Approximation to get a lower bound of the homogeneous association model ($\hat{L}_{KSA} \leq \hat{L_H}$). Therefore, the upper bound of the likelihood is established ($\hat{L_S} - \hat{L_H} \leq \hat{L_S} - \hat{L}_{KSA}$). This upper bound is tight and most nonsignificant interactions can be pruned. Its GPU version GBOOST [Yung et al., 2011] provides 40-fold speedup compared with that of BOOST.

3.3.5 TEAM

TEAM [Zhang et al., 2010] is an exhaustive algorithm to detect two-locus epistatic interactions in GWAS. It controls false positives by using permutation test. Permutation test is generally more accurate at finding the cut-off p-value threshold than direct adjustment methods like Bonferroni correction [Benjamini and Hochberg, 1995], but at a much higher cost. TEAM needs to compute the contingency table for every pair of SNPs on all the permutations to calculate p-values, which is very expensive. To reduce the computation cost, the authors observe that if two SNPs have the same genotype values on many individuals, then the computation of their contingency tables can be shared by considering only those individuals with different values. TEAM uses a Minimum Spanning Tree (MST), where nodes are SNPs and the weight of edges is the number of individuals with different values on the two SNPs, to maximize the sharing of contingency table computation. As the construction of MST can be costly, TEAM constructs an approximate MST instead. The performance of TEAM is faster than the brute-force approach by an order of magnitude. As TEAM does not presume any statistical model, it is applicable to any test statistic—e.g., χ^2 test, exact likelihood ratio test, and entropy-based test—based purely on contingency table information.

3.4 Data simulation

We simulate different types of datasets to evaluate the power, type-1 error rate, and scalability of each method.

3.4.1 Power

For each setting in both data with and without main effect below, 100 datasets are generated. In each dataset, we embed one ground truth epistatic interaction. Power is defined as the fraction of the 100 datasets on which the top prediction matches the ground truth.

Data with main effect

The embedded epistatic interaction demonstrates both main effect and interaction effect. There are at least fifty different models that satisfy the constraints for two-locus epistatic interactions [Li and Reich, 2000]. We consider the three commonly used models [Marchini et al., 2005] given in Tables 3.2,3.3 and 3.4, respectively. We simulate the data based on these three models. For each model, we try two different minor allele frequencies (MAF) at 0.2 and 0.5, and three different main effect values at 0.2, 0.3 and 0.5; thus giving a total of six different settings. These values represent the low and high value for each parameter. We use 2,000 samples and 1,000 SNPs for each dataset, as per previous works. These datasets are available from http://compbio.ddns.comp.nus.edu.sg/~wangyue/public_data/.

Table 3.2: Model 1: Two-locus multiplicative disease effect between and within loci

	AA	Aa	aa
BB	a	$a(1+\theta)$	$a(1+\theta)^2$
Bb	$a(1+\theta)$	$a(1+\theta)^2$	$a(1+\theta)^3$
bb	$a(1+\theta)^2$	$a(1+\theta)^3$	$a(1+\theta)^4$

Data without main effect

This type of epistatic interaction demonstrates weak main effect but strong interaction effect. Finding such type of epistatic interactions is a challenging "dark

Table 3.3: Model 2: Two-locus multiplicative disease effect between loci

	AA	Aa	aa
BB	a	a	a
Bb	a	$a(1+\theta)$	$a(1+\theta)^2$
bb	a	$a(1+\theta)^2$	$a(1+\theta)^4$

Table 3.4: Model 3: Two-locus threshold effect

	AA	Aa	aa
BB	a	a	a
Bb	a	$a(1+\theta)$	$a(1+\theta)$
bb	a	$a(1+\theta)$	$a(1+\theta)$

area" which many methods fail to explore. We use data from Dartmouth Medical School. The website, http://discovery.dartmouth.edu/epistatic_data, provides 70 models, composed of combinations of the following parameter values. (1) Two MAF settings of 0.2 and 0.4. (2) Seven heritability settings of 0.4, 0.3, 0.2, 0.1, 0.05, 0.025 and 0.01. (3) Five different penetrance tables. Each model is simulated using four sample sizes of 200, 400, 800 and 1,600. The number of SNPs is 1,000 for each dataset.

3.4.2 Type-1 error rate

We simulate 1,000 datasets without embedding any epistatic interaction, each with 2,000 samples and 1,000 SNPs. The MAF of each SNP is uniformly distributed in [0.05, 0.5]. Type-1 error rate of the methods is defined as the proportion of the 1000 datasets on which the significance level of the top prediction satisfies the user-specified threshold.

3.4.3 Scalability

To test the scalability, we use datasets with 100, 1,000, 10,000 and 100,000 SNPs. Each of the 4 datasets has 2,000 samples.

3.5 Experiment setting

All the experiments are conducted on a 64-bit Ubuntu system, with Intel (R) Xeon(R) CPU 2.66GHz, 16G memory.

SNPRuler provides a Java program. The heap size is set to `-Xmx7000M`, giving the maximum memory for the program to use. The maximum number of rules is set as 50,000. The rule length is set to 2 since we focus on two-locus epistatic interactions. The pruning threshold is set as 0, to test as many combinations as possible.

SNPHarvester also provides a Java program; it has two running modes. One is the "Threshold-Based" mode, where the user indicates the threshold significance level and the program outputs all results whose significance level is lower than the threshold. Another is the "Top-K Based" mode, where the program outputs the top K most significant results regardless of their significance level. The "Top-K Based" mode is used for our analysis.

TEAM provides a C++ program which consists of two sub programs: (1) to test all combinations and record the corresponding test statistic value and (2) to get the SNP pairs according to the user-specified False Discovery Rate (FDR). We use the default setting of other parameters and set the FDR value to 1.

BOOST provides a C program that only runs on Windows system. To let all programs run on the same hardware configuration, we use the `Wine` program (`http://www.wine.org`) which allows us to run a Windows program on a Unix system. There is no setting for BOOST; the output is the list of results whose likelihood ratio test statistic values are higher than 30 with 4 degrees of freedom.

Screen and Clean provides an R program; it has 4 running strategies, among which we choose the "Kitchen Sink". We set the P-value threshold to 0.1 and the number of pairs to be tested to 100.

BOOST filters out epistatic interactions with test statistic values less than 30 with 4 degrees of freedom. This corresponds to 0.1 significance level. For fair comparison, we add a post-processing step to filter output with P-values higher than 0.1 for other methods.

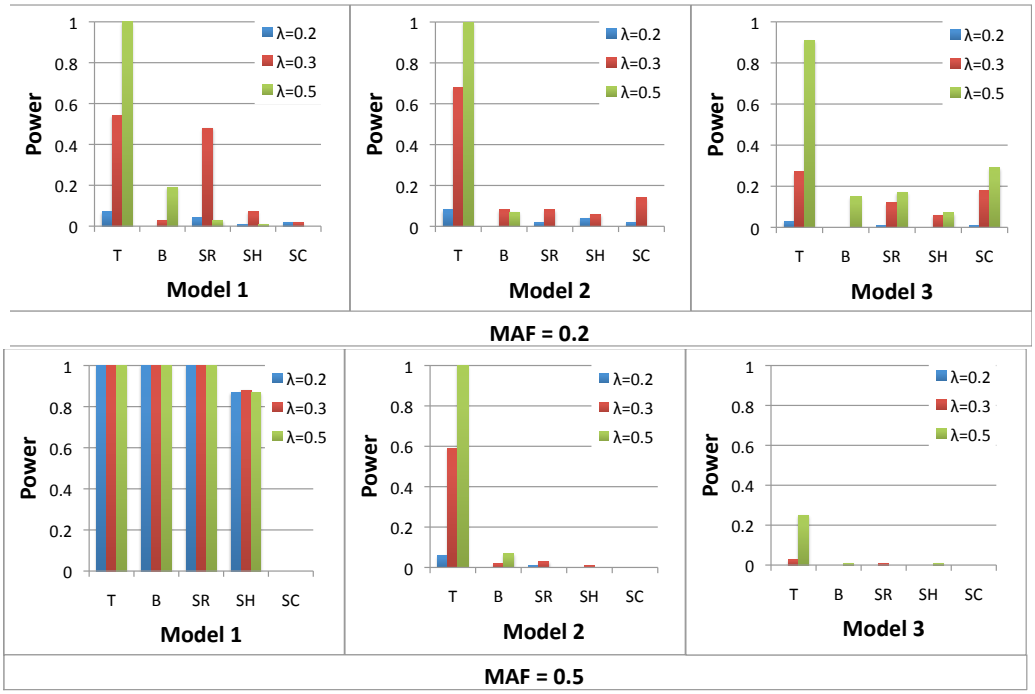

Figure 3.1: Power comparison under three main effect models. Each model has two MAF settings and three λ settings which control the main effect of the ground truth SNP. For each model, we generate 100 datasets. For each dataset, the sample size is 2,000 (1,000 cases and 1,000 controls) and the number of SNPs is 1,000. Abbreviations of the methods are: T (TEAM), B (BOOST), SR (SNPRuler), SH (SNPHarvester) and SC (Screen and Clean). The p-value for one-way ANOVA test is 0.0009.

3.6 Results

3.6.1 Model with main effect

The results here are obtained by using data generated in the first part of Section 3.4.1. Figure 3.1 shows that in each setting, TEAM outperforms all other methods. For the other four methods, different model settings lead to different rankings. For example, in Model 1 with λ=0.3, SNPRuler is second; in Model 2 with λ=0.5, Screen and Clean is second. The different performance of TEAM and BOOST is due to a key difference in defining the interaction effect. TEAM uses the χ^2 test to measure the significance of two-locus interactions and thus makes no assumption about the data. BOOST uses a log likelihood ratio test to

get the deviance difference between the log likelihood of the log-linear homogeneous association model and log-linear saturated model. BOOST performs well when the interaction terms contribute significantly to the model. However, when single-SNP association terms fit the model well and interaction terms do not contribute significantly, BOOST may not be able to detect the ground truth. This type of epistatic interactions is often referred as "statistical epistasis" [Cordell, 2002] and is widely accepted by the statistical community. SNPRuler is not an exhaustive method, but the test used is the same as that of TEAM. We set the pruning threshold to 0; thus it explores as many epistatic interactions as possible. Compared to TEAM, this method potentially misses true positives. The result of SNPHarvester is expected as its randomization technique makes it difficult to perform better than exhaustive search. Screen and Clean performs poorly, due to its numerous filtering steps in the two-stage design. In the screening step, before the main-effect lasso procedure starts, it includes only marginally significant and tag SNPs. After that, it still only considers n (set by the user) pairs of SNPs instead of all the possible pairs to continue the interaction model fitting procedure. In the cleaning step, the filtering test is applied to only a small number of SNP pairs, resulting in little power to detect the ground truth.

All five methods perform best on Model 1 compared to Model 2 and Model 3. This is because of the multiplicative effect between and within the two loci, making the epistatic interaction effect stronger and easier to detect. Model 2 only considers the multiplicative effect between two loci; the power to detect epistatic interactions drops obviously for all methods. The interaction effect of Model 3 is even weaker than Model 2, leading to the lowest power in all methods. It is also noted that the higher the main effect of the model, the easier it is for each method to detect epistatic interactions. However, SNPRuler and SNPHarvester do not follow this pattern because, when the main effect of the ground truth pair is large, these two methods prune such main effect SNPs at the filtering stage. This leads to the missing detection of ground truth.

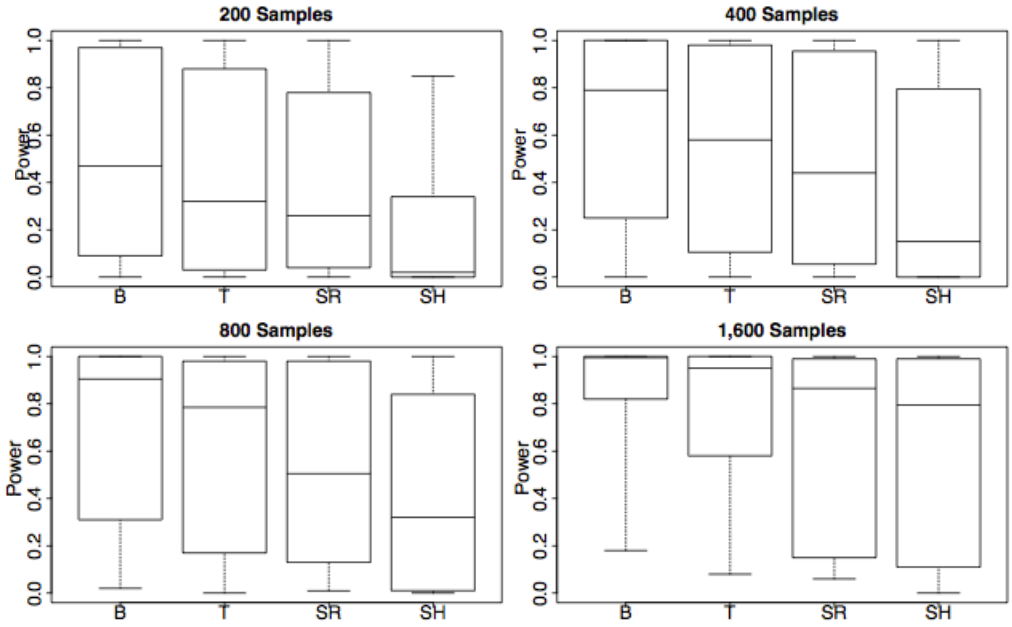

Figure 3.2: Power comparison under 70 models without main effect. For each model, we simulate data using four different sample sizes. These sizes simulate the study design from small scale to large scale. Abbreviations of the methods are: B (BOOST), T (TEAM), SR (SNPRuler), and SH (SNPHarvester).

3.6.2 Model without main effect

The results here are obtained using data generated in the latter part of Section 3.4.1. Screen and Clean is applicable only to data with main effect; thus we omit it here. Figure 3.2 gives an overall picture of the performance of the methods for each sample size, while Figure 3.3 and 3.4 gives the details. The median power of BOOST is the highest followed by TEAM. The performance of SNPRuler is close to that of an exhaustive method (TEAM) but is at a lower computational cost. BOOST performs the best in each setting and TEAM second; but the difference is not as obvious as that in data with main effect. SNPHarvester performs relatively poorly for each sample size. All methods perform well when heritability is high; when heritability reduces to 0.001, all methods have little power. Lescai and Franceschi [2010] point out in their study of neurological cancers that low heritability caused by phenocopy level (PE) is the main reason for the methods to lose power. We also notice that increasing the sample size

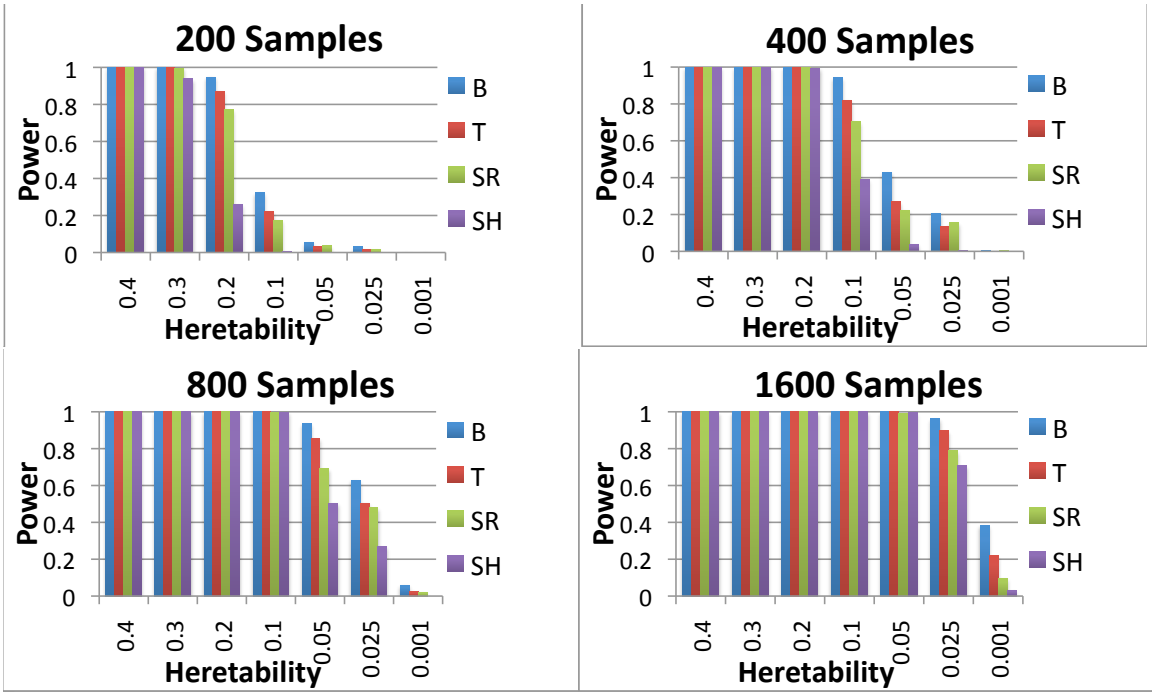

Figure 3.3: Detailed results of four methods on data without main effect for MAF 0.2. In particular, for models with heritability 0.001, MAF 0.2 and sample size 200, the results of these datasets were not reported previously; all four methods have zero power on them. This shows the limitations of purely statistical methods. The p-value for one-way ANOVA test is 0. 0997. Abbreviations of the methods are: B (BOOST), T (TEAM), SR (SNPRuler), and SH (SNPHarvester)

helps all these methods to improve their power in each heritability setting.

When we evaluate the four methods on data without main effect, we use all datasets that are publicly available. They include 70 models and 4 different sample sizes for each model. Part of these datasets are also used in BOOST, SNPRuler and SNPHarvester. BOOST does not include the results of 70 models for 200 samples. SNPRuler and SNPHarvester merely show results of 60 models and each model with 400 samples. Our reported results are consistent with previous reported results and are complementary to them. In particular, for those models with 0.001 heritability, 0.2 MAF and 200 samples, the results of these datasets were not reported previously; and all four methods have zero power (see Figure 3.3). This shows the limitations of purely statistical methods.

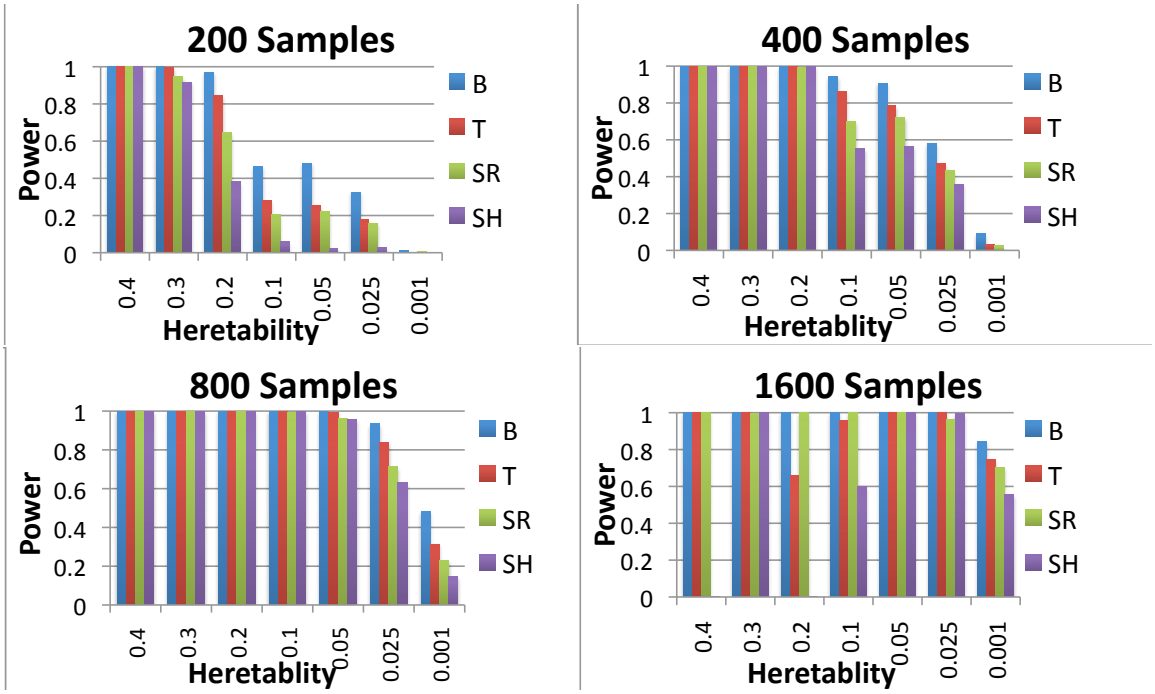

Figure 3.4: Detailed results of four methods on data without main effect for MAF 0.4. Abbreviations of the methods are: B (BOOST), T (TEAM), SR (SNPRuler), and SH (SNPHarvester)

3.6.3 Scalability

We apply all methods to datasets with 100, 1,000, 10,000, and 100,000 SNPs. From Table 3.5, BOOST is the fastest under the first three settings. This is due to its fast Boolean operation to collect contingency tables and upper-bound-pruning technique. When the SNP size grows to 100,000, it is much slower than the two non-exhaustive methods SNPHarvester and Screen and Clean. TEAM is the slowest in all settings for two reasons. First, the overall running time is only an order faster than that of a brute-force approach. Second, the permutation procedure makes it even more expensive, although traversing MST helps reduce the cost. SNPRuler cannot execute on the dataset with 100,000 SNPs because we get the "out of memory" error, even though we have set the heap size to 12.8G for the Java virtual machine, which is the maximum on our PC. SNPHarvester and Screen and Clean only identify a fixed number of candidate epistatic interac-

Table 3.5: Running time comparison of the five methods. Abbreviations of the methods are: SR (SNPRuler), SH (SNPHarvester), SC (Screen and Clean).

# SNPs	TEAM	BOOST	SR	SH	SC
100	58.23s	0.16s	2.43s	2.29s	7.39s
1,000	353.20s	2.47s	21.73s	22.33s	55.48s
10,000	7,406.29s	156.16s	1,097.65s	224.24s	626.96s
100,000	∼36 days	15,010.42s	NA	6,616.65s	5,858.34s

tions, and then fit them to a statistical model for follow-up analysis. Thus, their scalability is much better than the other three methods when SNP size grows.

3.6.4 Type-1 error

We define the type-1 error rate of a method as the proportion of datasets that the method reports the existence of significant epistatic interactions, out of the 1,000 datasets in which no epistatic interactions are actually embedded. The significance level is set to 0.05 after Bonferroni correction. The type-1 error rate for TEAM is 0.018, BOOST is 0.065, and SNPRuler and SNPHarvester both are 0.003. TEAM and BOOST have higher power thus their higher type-1 error rates are reasonable. Screen and Clean has problems controlling type-1 error, as its type-1 error rate is as high as 0.86.

3.6.5 Completeness

SNPRuler, SNPHarvester and BOOST use some pruning techniques to speed up the search. Hence they have better scalability than TEAM as shown in Table 3.5. The side effect of using pruning techniques is the loss of power—the most significant SNP pairs may be thrown away. To study the magnitude of this side effect, we pick the most significant SNP pair on each dataset and study how many of them are pruned. For each method, the most significant SNP pair is the SNP pair with the lowest p-value calculated using the statistical test used by the method. Thus, for BOOST, the most significant SNP pair is the SNP pair with the lowest p-value calculated using likelihood-ratio test. For the other two methods, the most significant SNP pair is the SNP pair with the lowest p-value calculated using the chi-square test. BOOST is a complete method in the sense

that there is no incorrect prunning for the most significant SNP pairs. Figure 3.5 also shows that the number of incorrectly pruned datasets of SNPRuler is smaller than that of SNPHarvester for both types of data. Correspondingly, the power of SNPRuler is higher than that of SNPHarvester.

3.7 Discussion

The five methods all demonstrate respective utilities through the experiment results above. No single method is simultaneously the most powerful, the most scalable, and has the lowest type-1 error rate in every setting. When users want powerful results and are not concerned with computation cost, we recommend using TEAM and BOOST. Compared with TEAM, BOOST uses a model-fitting procedure. If the data fits the model well, the result is usually good; otherwise, a model-free method may be the alternative choice. When users expect moderate running time and power, we recommend using SNPRuler. Its pruning technique helps reduce running time albeit at the risk of losing power. If users are conscious of computation cost and have to run very large datasets, we recommend using SNPHarvester because it only identifies a small number (40–50) of groups for the model-fitting procedure.

Our evaluations are based on simulation results. In a real study, users usually have no idea of the "ground truth" in the dataset. Hence it may not be sufficient to rely only on one method to obtain results. We suggest that, if time and computation resources permit, users try both the recommended model-free (i.e., TEAM) and model-fitting (i.e., BOOST) methods.

It is tempting to consider taking a "majority vote" of the results of two or more methods. For example, let every algorithm report their top-3 predictions. A SNP pair receives k votes if it is reported by k methods. We select the one with the highest vote as the final prediction. When there is a tie, we choose the one with the lowest P-value. Unfortunately, for both types of data tested, we find that an ensemble using such a strategy cannot increase power over using solely BOOST or TEAM. In Figure 3.6, we see that for data without main effect, BOOST's ground truth predictions highly overlap with the other three methods', so any ensemble cannot contribute a significant number of new ground truth

predictions. Specifically, the proportion of BOOST's ground truth predictions that are not predicted by the other three methods is 4.1%, while the proportion of the other methods' ground truth predictions not predicted by BOOST is 0.2%. Similarly, for data with main effect, no ensemble can outperform TEAM.

Our evaluations above only focus on two-locus epistatic interaction. Recently, Wang et al. [2010a] and Liu et al. [2011a] provide a general model that can be extended to n-locus epistasis. They also provide mathematical details of dissecting the χ^2 test into different epistatic components. For example, two-way epistatic interaction can be partitioned into four epistatic components: additive $\times$ additive, additive $\times$ dominant, dominant $\times$ additive and dominant $\times$ dominant. This helps characterize epistatic interactions in a more specific way and provides more physiological insights.

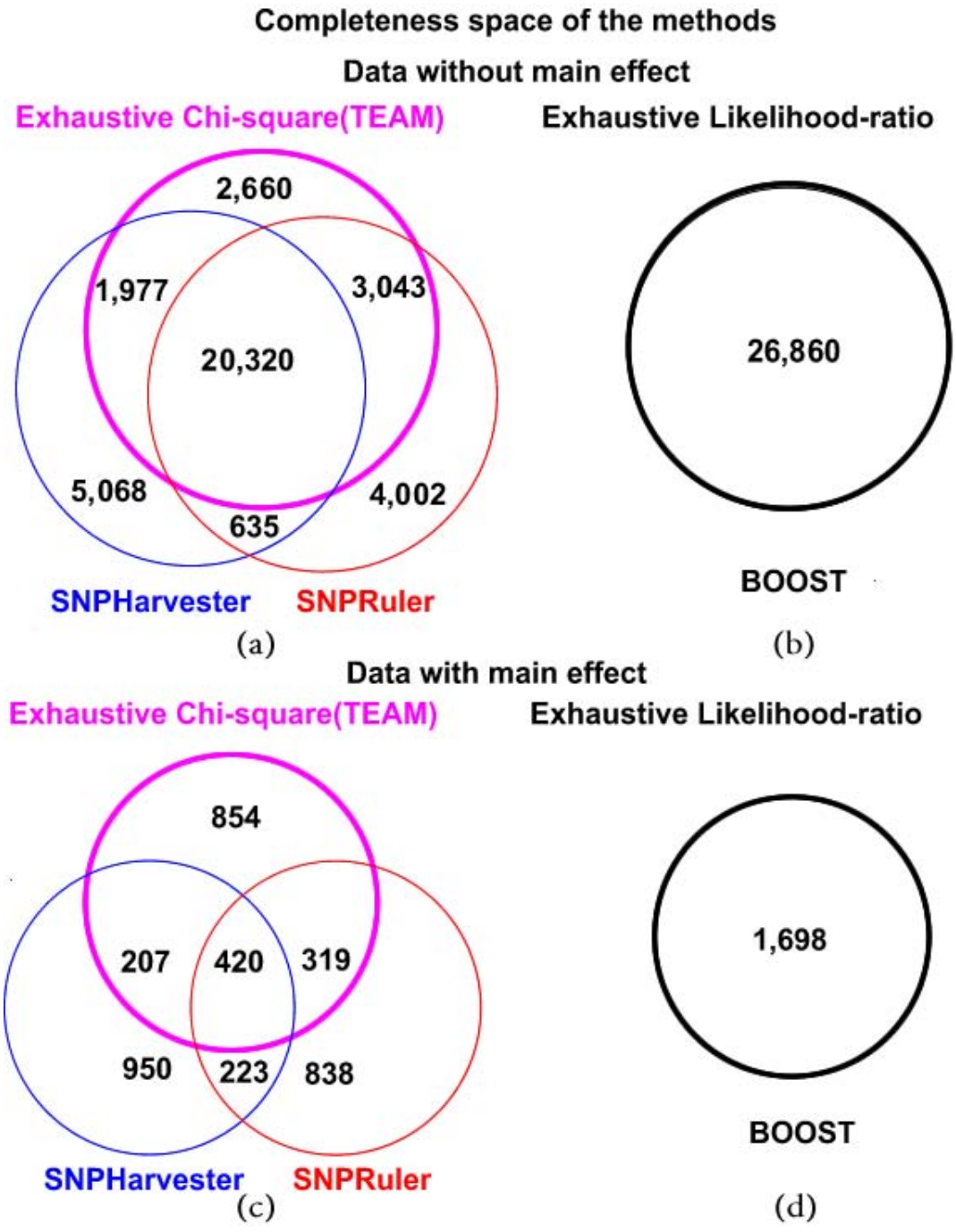

Figure 3.5: The completeness space for the four methods. As there are two types of datasets and two types of test statistics, four venn diagrams are drawn respectively. In Part (a), all three methods—TEAM, SNPRuler and SNPHarvester—use χ^2 test. TEAM's outputs represent the 28,000 (20,320 + 1,977 + 2,660+ 3,043) top significant SNP pairs in 28,000 datasets. SNPHarvester can identify 22,297 (20,320+1977) of them. Among the 28,000 top SNP pairs, 20,320 of them can be identified by all three methods. Parts (b), (c) and (d) follow similar explanations.

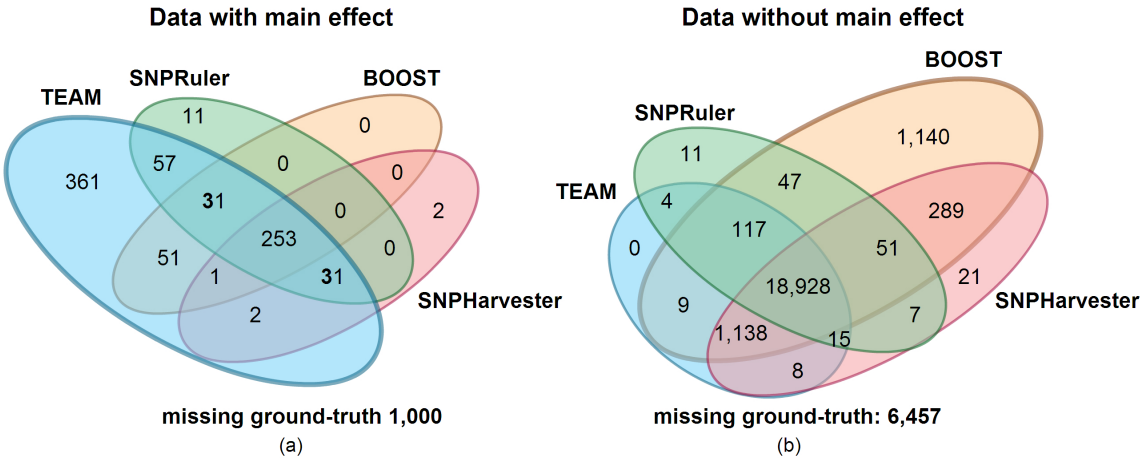

Figure 3.6: The power space for the four methods on data with and without main effect. In part (a), there are in total 1,800 datasets for 18 settings of the simulated datasets, which corresponds to 1,800 ground truth. Among these ground truth, only 800 of them can be detected by at least one of the four methods, while the best method—TEAM—identifies 787 ground truth out of 800. This explains why using ensemble methods cannot outperform TEAM. Similar observation is illustrated in Part (b).

Chapter 4

CEO: A Cloud Epistasis cOmputing model in GWAS

4.1 Introduction

It is becoming increasingly important and challenging in genome-wide association study (GWAS) to identify single nucleotide polymorphisms (SNPs) associated with phenotypes such as human diseases (e.g, breast cancer, diabetes and heart attacks). Traditionally, researchers focused on the association of individual SNPs with the phenotypes. Such methods can only find weak associations as they ignore the genomic and environmental context of each SNP [Moore and Williams, 2009]. However, SNPs may interact (known as *epistatic interaction*) and jointly influence the phenotypes. As such, there has been a shift away from the one-SNP-at-a-time approach towards a more holistic and significant approach that detects the association between a combination of multiple SNPs with the phenotypes [Moore *et al.*, 2010].

In the meantime, the number of discovered SNPs is becoming larger and larger. For example, the dataset from the Hapmap project[1]contains 3.1 million SNPs and the 1000 Genome project[2] provides approximately 15 million SNPs. From a computational perspective, it is very time consuming to determine the

[1] http://hapmap.ncbi.nlm.nih.gov/
[2] http://www.1000genomes.org/page.php

interactions of SNPs. Given n SNPs, the number of k-locus is ${}^nC_k = \frac{n!}{k!(n-k)!}$. This renders existing statistical modeling techniques (which work well for a small number of SNPs) [Wu et al., 2010, 2009; Yang*et al.*, 2010; Zhao and Chen, 2012] impractical. Likewise, techniques that enumerate all possible interactions [Wan et al., 2010a; Zhang et al., 2010] are not scalable for a large number of SNPs. To reduce the computation overhead, heuristics [Wu et al., 2010, 2009; Zhao and Chen, 2012] have also been developed. These schemes add a filtering step to select a fixed number of candidate epistatic interactions and fit them to a statistical model. However, these approaches risk missing potentially significant epistatic interactions that may have been filtered out, as concluded in the previous chapter. Therefore, a scalable and efficient approach becomes attractive for such a computationally intensive task in large-scale GWAS.

A promising solution to the computation challenge is to exploit parallel processing. There are a variety of high-performance computing solutions. For example, Ma et al. [2008] describes a tool for processing single-locus and two-locus SNPs analyses using a supercomputer. However, it is not easy for researchers to rewrite their own programs on specialized hardware. As another example, two-locus SNPs analysis is performed using graphics processing units (GPU) in Greene *et al.* [2010]. However, this requires the users to understand the GPU architecture well to fully exploit the computation power of GPU. Instead, we aim to develop a cloud-based solution which has a number of benefits. First, the MapReduce framework (available in most cloud services) offers high scalability, ease of programming and fault tolerance. Second, most software can be easily deployed on the cloud and made accessible to all. Third, there are already low-cost commercially available cloud platforms (e.g., Amazon Elastic Compute Cloud (Amazon EC2)). Fourth, the pay-as-you-use model of such commercial platforms also makes them attractive for end-users who need not own, manage, and (over-)provision any computational resources.

In this chapter, we propose a cloud-based epistasis computing (CEO) model to find statistically significant epistatic interactions. Our solution is based on the MapReduce framework [Dean and Ghemawat, 2004], and implemented on Hadoop[1], an open-source-equivalent implementation of the MapReduce frame-

[1] http://hadoop.apache.org/

work. We develop solutions for determining significant interactions for two-loci and three-loci as well as computing the top-k most significant answers efficiently. As a first cut, we have adopted a brute force approach that examines all possible interactions among the SNPs. This ensures that we will not miss any statistically significant interactions for two-loci and three-loci. Our method can be easily extended to deal with heuristics approaches. We validate our proposed CEO on a local cluster of more than 40 nodes. Our results show that our CEO is efficient, and that the MapReduce framework can be effectively deployed for bioinformatics research such as GWAS.

The rest of this chapter is organized as follows. Section 4.2 provides the problem formulation and reviews some background knowledge. In Section 4.3, we propose our CEO processing model for both two-loci and three-loci analysis. We also present an efficient approach to retrieve the top-k most significant answers. Section 4.4 reports results of a performance study on our own cluster. In Section 4.5, we provide additional discussion, and finally, we conclude the chapter in Section 4.6.

4.2 Problem formulation

Typically a GWAS uses two types of data—genotype data that codes the genetic information of each individual, and phenotype data that measures the individual's quantitative traits. For simplicity, we use bi-allelic genotype data (i.e., a locus has allele A and a which can form three types of genotypes, AA, Aa and aa.) and are encoded as 0, 1 and 2 in the raw data. For phenotype data, we consider the binary form (0 for control and 1 for case). Our model can handle other types of genotype and phenotype data also. The table on the left of Figure 4.1 shows an example of the raw data format for a dataset with m samples and 6 SNPs. Each row contains the raw data of an individual sample. The first and last columns are the sample id and phenotype. The rest of the columns are the genotype of each SNP.

For our scheme to work, the raw data has to be pre-processed to transform the SNP information into the following new data format: $\langle SNP_i,\ PT,\ GT\ ,\ list(sampleID)\rangle$where SNP_i, PT and GT are the i^{th} SNP, phenotype value and

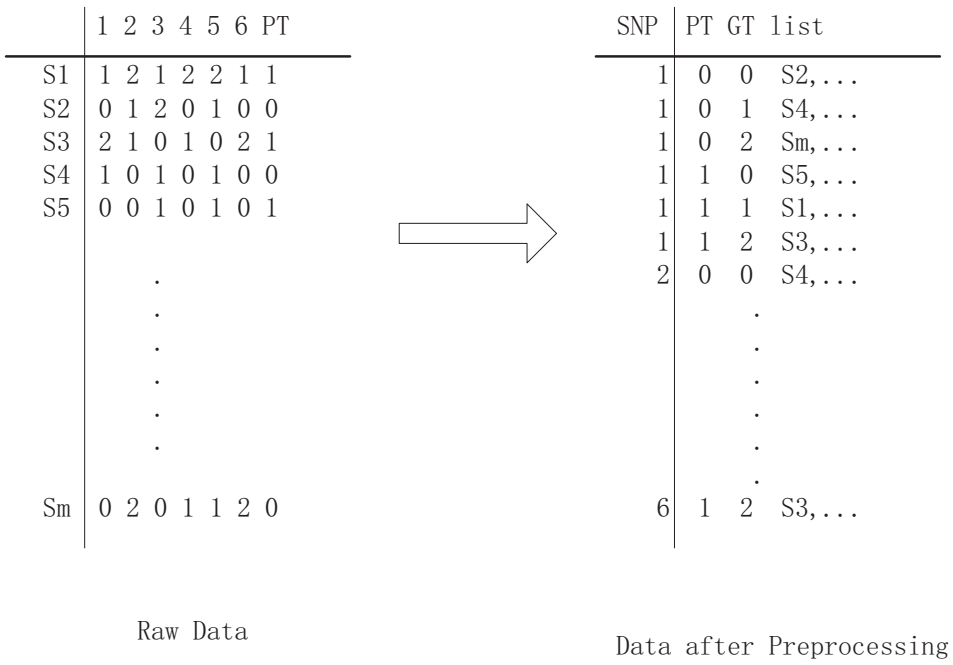

Figure 4.1: Data formats before and after preprocessing

the SNP genotype respectively. $list(sampleID)$ stores all the sample ids in the data set whose phenotype and SNP genotype on the SNP_i are PT and GT respectively. The table on the right of Figure 4.1 depicts the transformed data.

The pre-processing can be performed in one MapReduce job efficiently. For example, pre-processing 100,000 SNPs only takes 76 seconds on a 43-node cluster. As this is not the focus of our work, we shall not discuss this further. For the rest of this chapter, we assume that the input to our algorithm is the pre-processed data.

The goal of our research is to identify a set of most significant SNP pairs (epistatic interactions) that correlate to the phenotype. To measure the association between epistatic interaction and phenotype in our CEO model, we adopt the χ^2-test [Balding , 2006], which is widely used. Moreover, as our CEO framework assumes no statistical model fitting and thus parameter free, the χ^2-test is effective in capturing any order interactions.

Take two-locus epistatic interactions as an example. Let $n_{0(j,k)}$ denote the

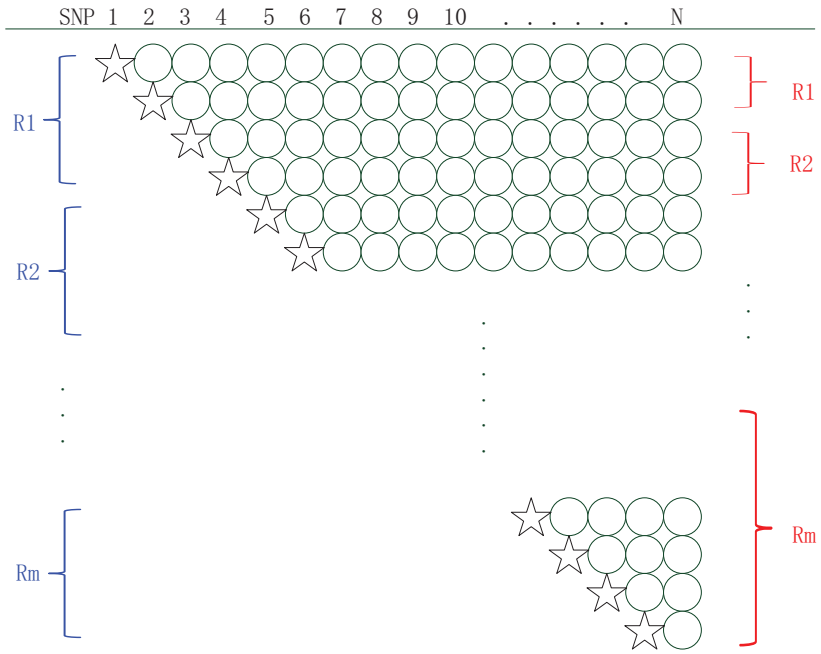

Figure 4.2: SNP-pairs representation and distribution to reducers

number of samples in the control group whose first locus's genotype code as 'j' and second locus's genotype code as 'k', where j and k take on values 0, 1 or 2. Likewise, we can denote $n_{1(j,k)}$ for the case group. For two-loci, we have 18 combinations (2 ×3 × 3). Moreover, let $n = \sum_{i=0}^{1}\sum_{j=0}^{2}\sum_{k=0}^{2} n_{i(j,k)}$, $n_i = \sum_{j=0}^{2}\sum_{k=0}^{2} n_{i(j,k)}$, and $n_{j,k} = \sum_{i=0}^{1} n_{i(j,k)}$. The null hypothesis behind the χ^2-test is that there is no association between the two-locus epistatic interaction and phenotype. We thus calculate the χ^2-test value of this epistatic interaction using the following formula:

$\chi^2 = \sum_{i=0}^{1}\sum_{j=0}^{2}\sum_{k=0}^{2} \frac{(n_{i(j,k)} - n_i n_{(j,k)}/n)^2}{n_i n_{(j,k)}/n}$

As the χ^2-test follows the χ^2 distribution, the corresponding significance level can be obtained after Bonferroni correction. The lower the value is, the more confident we are to reject the null hypothesis. The resultant p-value for the two-locus epistatic interaction can be obtained as $P(x > C)$ where C is the χ^2 value, and $P(x)$ is the probability at value x under the χ^2 distribution. The above expressions can be easily generalized for three-locus interaction.

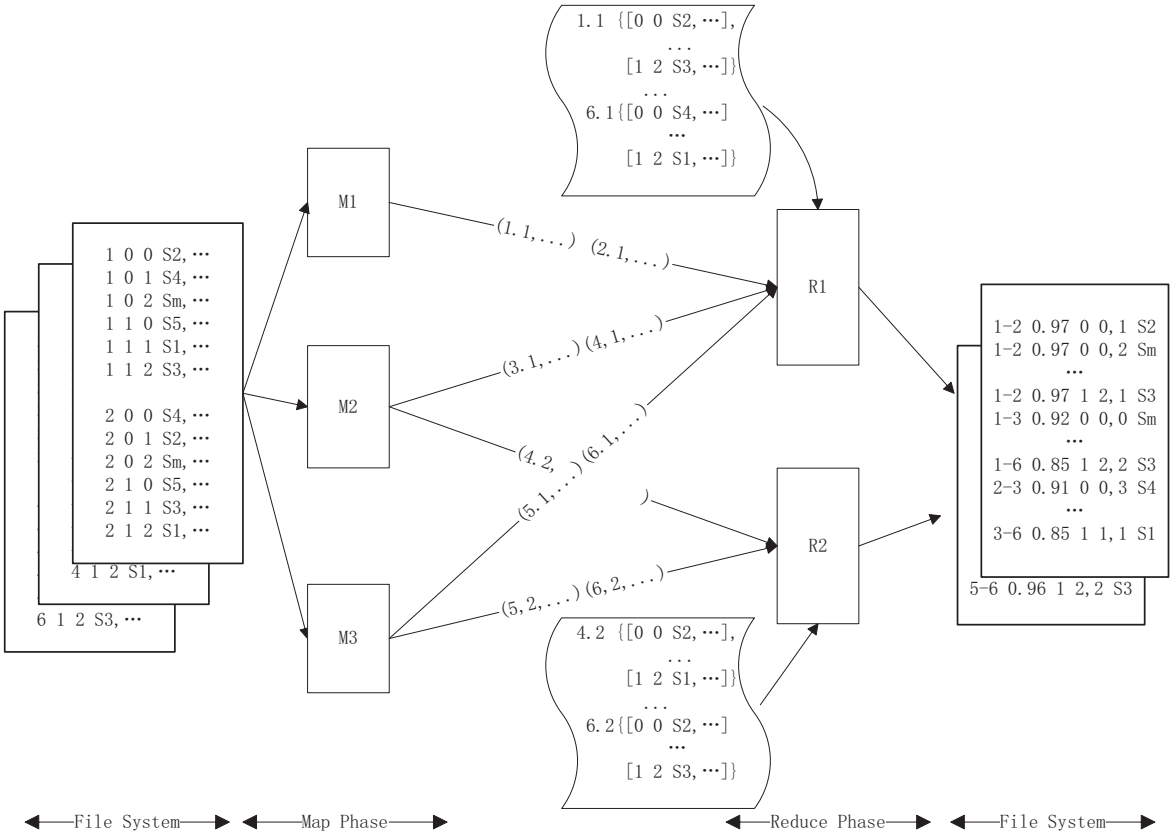

Figure 4.3: Two-locus epistatic analysis example with 6 SNPs

4.3 CEO processing model

In this section, we introduce our CEO processing model using MapReduce. In our system, we provide several components including two-locus epistatic analysis, three-locus epistatic analysis and top k retrieval. Moreover, users can choose to output all the epistatic interactions information or the ones whose statistical values are not less than a user-specified threshold. In the following section, we introduce them one by one.

4.3.1 Two-locus epistatic analysis

For two-locus epistatic analysis, we aim at finding statistically significant interaction between two SNPs among all SNP-pairs. For each two-locus SNP com-

bination, we can get contingency table and calculate its statistical value. Using the statistical value, we can derive the p-value to identify the significance of this combination. For N SNPs in the data set, we need to calculate $N(N-1)/2$ two-locus SNP combinations, as depicted in Figure 4.2. Each row represents a subset of SNP-pair computations where the starred node has to be paired up with a circled node. Thus, row 1 has (N-1) pairs, row 2 has (N-2) pairs and so on.

Our goal essentially is to split these $N(N-1)/2$ pairs of SNPs across all nodes to be processed in parallel. We have two issues to address here: (a) How do we split the SNP pairs across all nodes? (b) How to perform two-locus analysis under the MapReduce framework?

We shall first look at issue (a). Given N SNPs and M `reducers`, we consider the following two simple strategies:

Naive Model. The most straightforward approach is to simply distribute approximately equal number of rows to each reducer. This is depicted by the square brackets on the LHS of Figure 4.2 where the first $\frac{N}{M}$ rows are assigned to the first reducer, the next $\frac{N}{M}$ rows are assigned to the second reducer and so on. Here, the number of SNP-pairs can be easily determined without any additional meta-data, e.g., for row 1, we know that we need to pair up SNP1 (starred node) with all other remaining SNPs (circled node), resulting in (N-1) pairs.

Greedy Model. Under the naive model, some `reducers` are more heavily loaded than others, e.g., `reducer` one is likely to be a bottleneck. To achieve better load balancing, we also examine a greedy solution. Ideally, each `reducer` should process $N(N-1)/2M$ SNP pairs. Therefore, starting from the first row, we seek to allocate consecutive rows to a `reducer` such that the total number of SNP pairs for these rows is closest to $N(N-1)/2M$. In Figure 4.2, the square brackets on the RHS show that, under the greedy scheme, each `reducer` may be assigned different number of rows to process. However, the computation task in each `reducer` is about the same. Like the naive scheme, this method also requires minimum meta-data to be transferred.

As we shall see in our experimental study, in Section 4.4, it turns out that these schemes are surprisingly effective (in the sense that the processing cost is almost proportional to the number of pairs/triples to be computed).

We are now ready to look at issue (b). Without loss of generality, let us assume

we have M reducers. Under the MapReduce framework, the `mapper` essentially determines the `reducer` in which an SNP pair should be sent to, and the `reducer` computes the statistical significance of each SNP pair allocated. Figure 4.3 shows how the CEO model processes the data having 6 SNPs.

Map Phase : Each `mapper` reads a chunk of the input (pre-processed) data. For each SNP, it then determines the `reducers` which this SNP should be shuffled to. We shall discuss how the `reducers` are determined later. It suffices now to assume that this information is available to the `mapper`. We note that one SNP information may be shuffled to multiple different `reducers`. For example, in Figure 4.2, SNP_N needs to be shuffled to all the `reducers`. This, unfortunately, is not supported by the MapReduce framework which allows only one output $(key, value)$ pair emitted from a `mapper` to be shuffled to one `reducer`.

Shuffling Phase: We write our own partitioning function to parse the *reducer_marker* in the key and partition the emitted pairs to multiple reducers.

Reduce Phase: The MapReduce library sorts and merges the intermediate results based on the key. The $(key, value)$ pairs with the same key, are grouped together as $(key, set(values))$ pair where set(values) is a set of values for that key. The $(key, set(values))$ pairs are supplied to a user's reduce function in order. Because all the keys have same `reducer_marker` , the key will be sorted only based on the SNP_i. Thus, the data for SNP_i are sent to the `reduce` function before those for SNP_j where $i < j$. This means that the starred nodes are supplied earlier than the circled nodes. Therefore, in each `reducer`, only the starred nodes need to be cached in the main memory. As the circled nodes are received, they can be immediately paired up with the starred nodes to compute its p-value, after which the circled nodes can be discarded. As such, our CEO model significantly reduces the memory utilization.

In our processing model, the two-locus analysis can be finished in one MapReduce job.

4.3.2 Three-locus epistatic analysis

Three-locus epistatic analysis aims at finding statistically significant interaction between three SNPs. Here we propose one way of doing three-locus epistatic

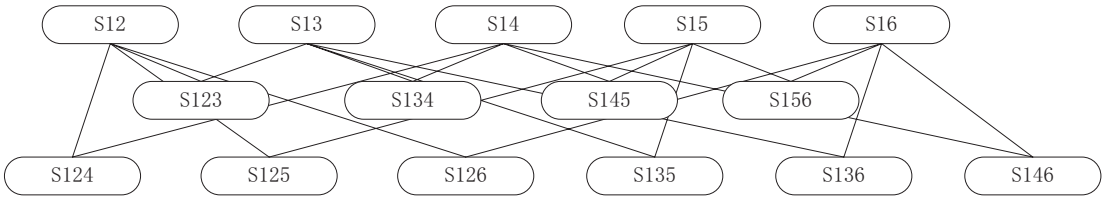

Figure 4.4: All the Three-locus SNPs having SNP1

analysis using the output of two-locus epistatic analysis. Note that the output data of two-locus analysis are written to the file system.

As what we have discussed before, from each row in Figure 4.2 , we can get all the needed two-locus SNP combinations involving the tree head SNPs (i.e., the first SNPs that are sent to reducers). Further, if we combine any two two-locus SNPs from one tree, we can get all possible three-locus SNPs involving the head node SNP. Figure 4.4 shows an example of finding all the three-locus SNPs with SNP_1 using the two-locus SNP information from the first row in the 6 SNPs example. In the same way, all the possible three-locus SNPs involving SNP_m can be generated from the combinations of two-locus SNPs of the row whose starred node is SNP_m.

For three-locus epistatic analysis, the same processing model can be adopted here. All the two-locus SNPs which are derived from the same computation tree need to be processed in the same `reducer` to get all the three-locus SNPs.

Map Phase: The two-locus SNP data is split into small chunks and each chunk is assigned to each `mapper` by the MapReduce library. As already mentioned above, the two-locus SNPs derived from the same computation tree must be shuffled to the same `reducer`. To achieve this, the key in the output $(key, value)$ pair from the Map phase is set as $SNP_i.SNP_j$ where SNP_i and SNP_j are the starred node and the circled node respectively.

The advantage of setting the output key in this format is that, after sorting the intermediate result according to the keys by the MapReduce library, all the two-locus SNPs in one tree can be grouped closely and fed into the reduce function continuously. In the reduce phase, after processing all the two-locus SNPs from one tree, the data can be discarded from the memory to minimize memory

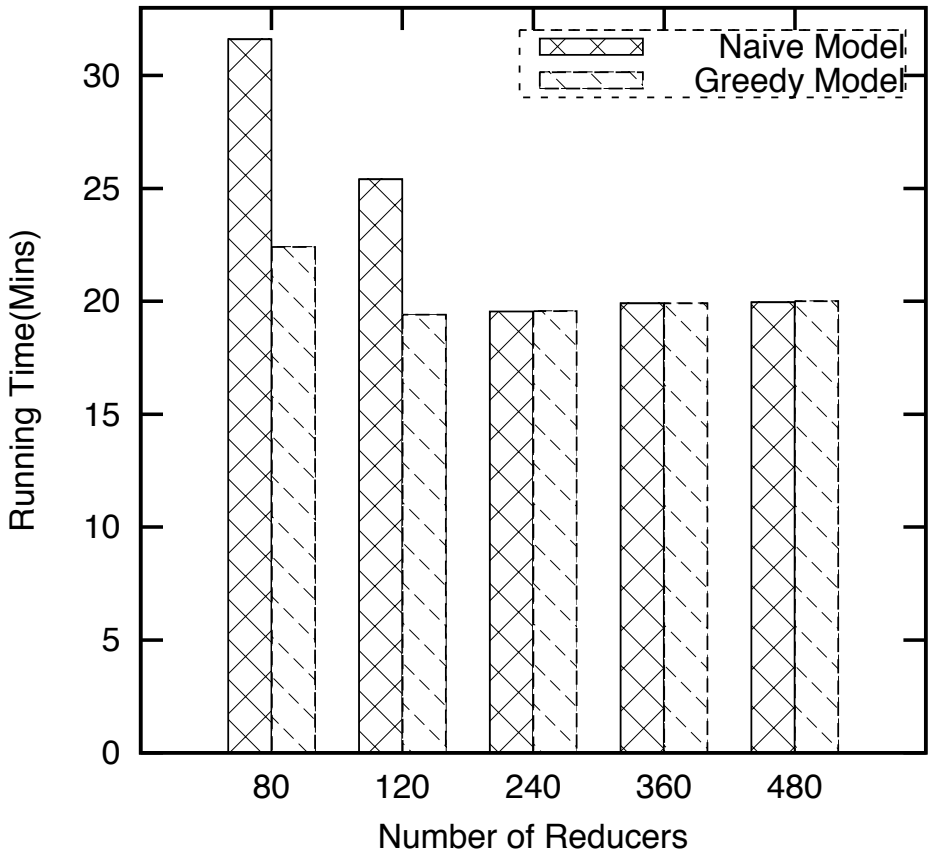

Figure 4.5: Dependence of Job Completion Time on Reducer Numbers

utilization.

Shuffling Phase: Our specified partitioning function is used to partition the pairs according to SNP_i value in the integer part of the key. The intermediate result from the `mappers` with the same SNP_i will be shuffled to the same `reducer`.

Reduce Phase: After sorting and merging the intermediate results, the two-locus SNP information with smaller starred node, will be supplied to `reduce` function earlier than the others. Combining any two two-locus SNPs at the `reducer`, we get the three-locus SNPs and calculate its statistical significance. The result is then output to the file system.

The load balancing algorithm can also be used here for optimization. Three-locus analysis can be performed using one MapReduce job using the two-locus SNP data.

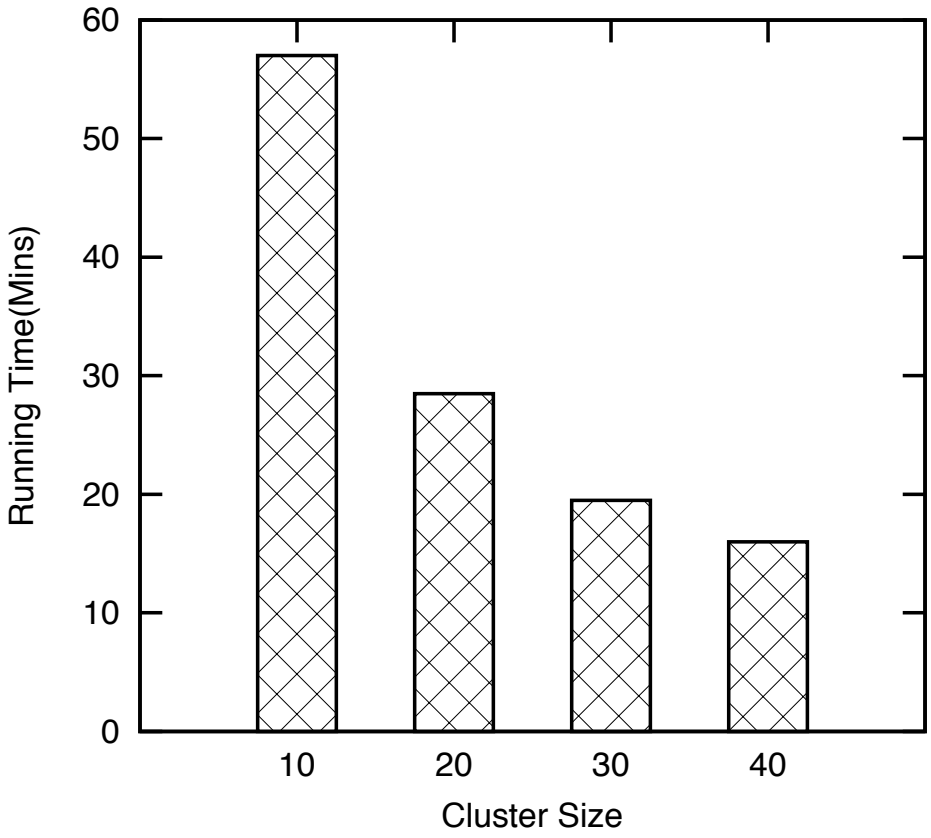

Figure 4.6: CEO Scalability and Performance Comparison

4.4 Experiments and results

Apache Hadoop is an open-source-equivalent implementation of the MapReduce framework, running on HDFS (Hadoop distributed file system). We run a series of experiments on our local cluster of more than 40 nodes to evaluate our model in Hadoop. Each node consists of a X3430 4(4) @ 2.4GHZ CPU running Centos 5.4 with 8GB memory and 2x 500G SATA disks. Moreover, since our tasks at hand are computationally intensive, we set the number of `reducers` per node to be equal to the number of cores at the node, which is 4. This guarantees that each `reducer` can get one core. Therefore, there are a total of 4*N `reducers` which can be run simultaneously on a N-node cluster.

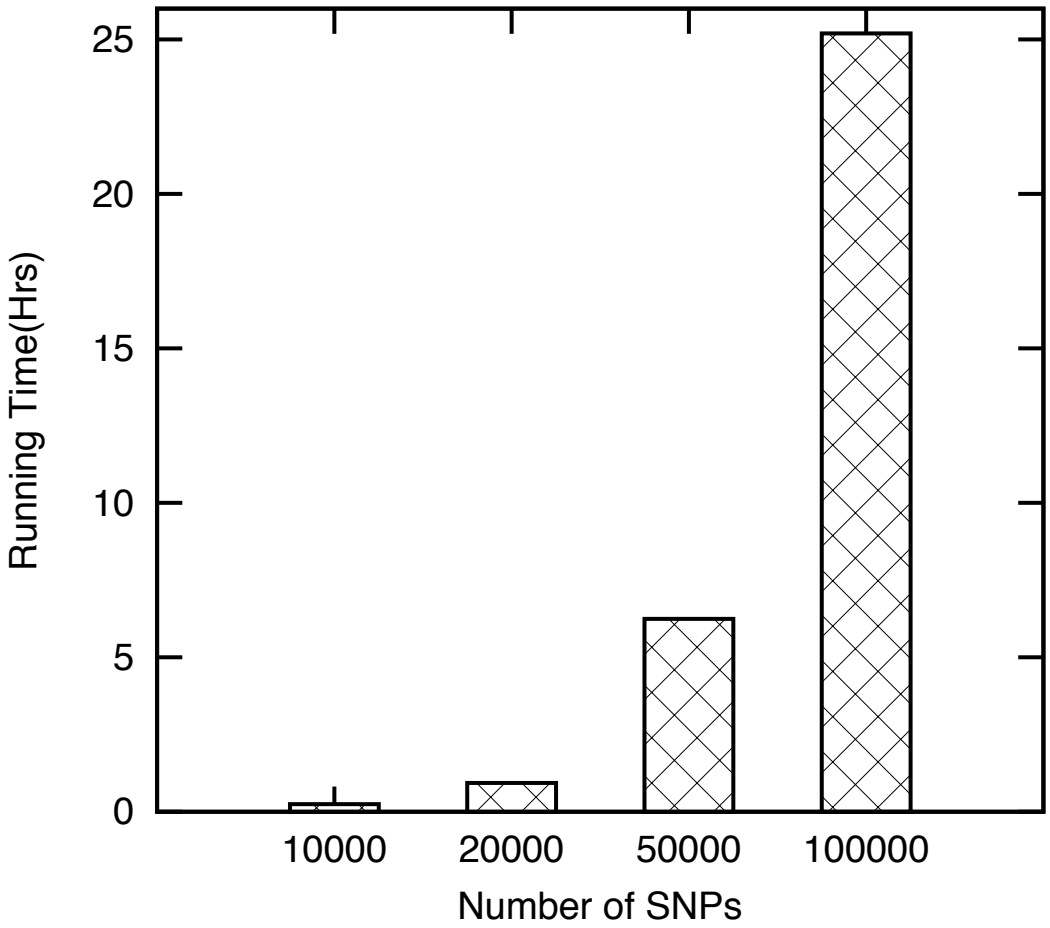

Figure 4.7: CEO Performance on Processing Different Number of SNPs on Local Cluster with 43 Nodes

Effect of number of reducers on completion time: For Hadoop application, a user can specify the number of `reducers` to be used in one job. Because we have preconfigured the total number of `reducers` to be 4*N for a N-node cluster, this may require multiple phases to complete a job. For example, if N=30, then by specifying 120 `reducers` in one job, we can complete it in 1 phase; with 360 `reducers`, it will then take 3 phases to complete the job. Our first experiment is to investigate the optimal number of `reducers` that should be set for one job based on a given cluster size. This experiment is conducted with a 10,000 SNPs dataset on a 30-node cluster. Note that all the datasets we used include 2000 samples. Figure 4.5 presents the running time for both the Naive and Greedy

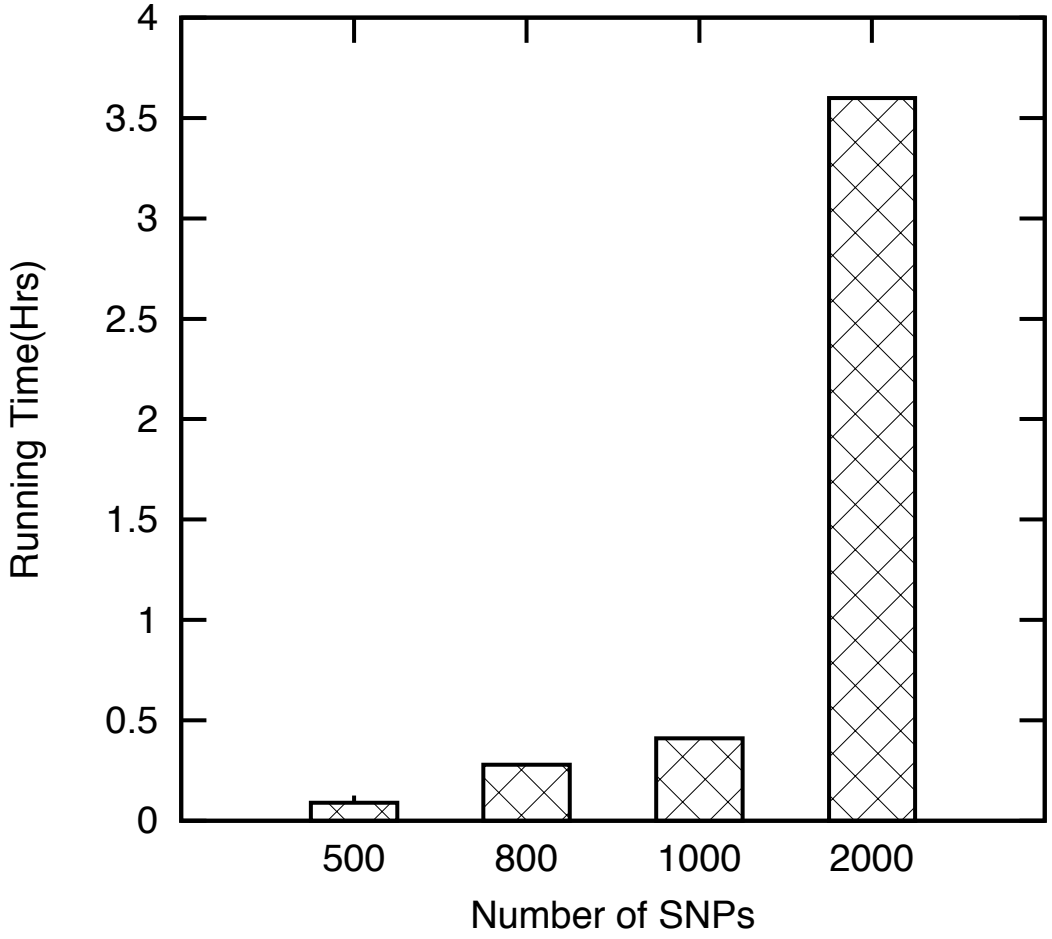

Figure 4.8: Three-locus Epistatic Analysis on Local Cluster with 43 Node

models. As shown, there is a certain optimal number of `reducers` that should be used. When the number of `reducers` is too small, the computation resources are not fully utilized. On the other hand, when the number of `reducers` is too large, the processing may require multiple phases which increases the communication overhead. We note that while the Greedy model is optimal when the number of `reducers` corresponds to the actual configured value (i.e., 120), the Naive model is optimal when a larger number of `reducers` is used (i.e., 240). This is because for the Naive model, a larger number of `reducers` means that the `reducer` with the most skewed load will be allocated smaller load. In fact, as the number of `reducers` increases, the Naive scheme performs as well as the Greedy model. Based on the above results, for the following experiments, we only use the Greedy

model.

Scalability : In this experiment, we study the scalability of the CEO model as the system resources increase. Figure 4.6 shows the completion time analyzing 10,000 SNPs as the cluster size increases from 10 to 40 nodes. The reducer numbers in each job are set as 40, 80, 120 and 160 respectively. From the result, we can see that when more nodes are added for processing, the completion time reduces. In fact, we observe a linear speedup in performance. When we double the resources, the execution time reduces to half, such as the execution time on 10/20/40-node clusters.

Two-locus analysis: In this experiment, we study the performance of the CEO model for two-locus analysis as we vary the number of SNPs processed. Figure 4.7 shows the processing time for the data sets with 10,000, 20,000, 50,000 and 100,000 SNPs on a 43-node cluster. As expected, the processing time is essentially proportional to the number of interacting SNP-pairs to be evaluated. We observe that even for 100,000 SNPs, the CEO model only takes 25 hours to complete the processing. This shows that our CEO model is effective.

Three-locus analysis: We also evaluate the performance of three-locus analysis on the 43-node cluster. The result is presented in Figure 4.8 for SNP size of 500, 800, 1,000 and 2,000. We observe that the running time is also proportional to the number of SNP-triples. This confirms that the CEO scheme can effectively balance the load across all nodes.

4.5 Top-K retrieval

In our system, we store the result of the two-locus and three-locus analysis in HDFS to allow users to do further analysis. One important function that we can further provide is to allow users to retrieve only the top-k most significant results with the lowest p-value. We have also provided such a capability in our system under the MapReduce framework. The basic idea is to split the output of the two/three-locus analysis into chunks. Each chunk is then assigned to one mapper. Next, each mapper will select the top-k most significant pairs/triples and shuffled these results to one reducer. Finally, the reducer can determine the global top-k answers based on all local top-k ones it receives. Our top-k scheme

is very efficient. For example, retrieving the top 10 most significant SNP pairs from the two-locus output of size 54GB only takes 132 seconds in the 43-node cluster.

4.6 Conclusion

According to Ma et al. [2008], it would require 1.2 years to do the pairwise epistasis testing of 500,000 SNPs using a serial program on a 2.66 GHz single processor without parallel processing. In this chapter, we have provided a cloud epistatic computing model (CEO) for large-scale epistatic interactions using the MapReduce framework. Our experimental results demonstrate the practical advantage of using the CEO model to exhaustively search two-locus epistatic interactions. We also provided a three-locus analysis approach as an example of k-locus analysis using our model. More importantly, by using the MapReduce framework, we have shown that large-scale data analysis in GWAS can be easily performed over commodity computers or cloud resources. The scalability of the MapReduce framework to thousands of machines with good fault tolerance will make such compute-intensive computations acceptable. Currently, we have used the popular χ^2 test to measure the interaction effect. Our CEO model can be easily adapted to handle other methods that utilize contingency table information to determine interaction effect (e.g., likelihood ratio, normalized mutual information, uncertainty coefficient). Also, for existing methods involving a filtering step and statistical model fitting step, our work can be used as a filtering step to retrieve the top-k most significant interactions for follow-up analysis. As future work, we plan to look at other strategies to allocate SNPs to nodes. For example, a best-fit model may assign the next available row to the `reducer` with the least number of SNP pairs. Alternatively, an ideal model may assign the SNP pairs in a round-robin fashion to the `reducers` so that every `reducer` will end up with the same number of SNP-pairs. We will explore such methods to study their effectiveness. The CEO source codes can be downloaded at `http://www.comp.nus.edu.sg/~wangzk/document/CEOSOURCECODE.tar.gz`

Chapter 5

eCEO: An efficient Cloud Epistasis cOmputing model in GWAS

5.1 Introduction

Finding significant epsitatic interactions of SNPs involve two major computational challenges:

1. Given a large number of combinations of SNPs, how can we distribute them across multiple processing nodes effectively to achieve load balancing?

2. Given a single combination of SNPs, how can we efficiently compute the significance of its association with the phenotype?

Chapter 4 describes the *CEO* model where we design the Greedy parallel distribution approach to tackle the first challenge and adopt a naive solution for the second challenge. Here, we propose an *e*fficient *C*loud-based *E*pistasis *C*omputing (*eCEO*) model to find statistically significant epistatic interactions. Our *eCEO* model is based on Google's MapReduce framework [Dean and Ghemawat, 2004], and is implemented on Hadoop, an open-source-equivalent implementation of the MapReduce framework. We develop solutions for the two computational challenges mentioned above. For the first challenge, we develop and study two ap-

proaches to distribute a large number of combinations of SNPs across processing nodes—a Greedy model and a Square-chopping model. For the second challenge, we adopt a Boolean operation approach, which is similar to the method used in [Wan et al., 2010a], and other optimizations.

We first apply our solutions for determining significant interactions for two-locus and three-locus as well as retrieving the top-k most significant answers. As a first cut, we have adopted a brute force approach that examines all possible interactions among the SNPs. This ensures that we do not miss any statistically significant interactions. Our method can be easily extended to deal with heuristics approaches. We validate our proposed *eCEO* model on our local cluster of over 40 nodes and on a public cloud (viz., Amazon EC2). Our results show that our *eCEO* model is efficient, and that the MapReduce framework can be effectively deployed for bioinformatics research such as the GWAS.

We then conduct an experimental study to show that *eCEO* outperforms *CEO* by a wide margin. For example, our experiment result shows that the execution time for processing 500,000 SNPs in a 43-node cluster is reduced from the 25 to 30 days using *CEO* model to the 9 hours using the *eCEO* model. Compared with *CEO*, our *eCEO* model supports four test statistics to measure the significance of the association between a combination of SNPs and the phenotype; and users can choose an appropriate one that meets their needs.

The rest of this chapter is organized as follows. Section 5.2 provides some background knowledge. In Sections 5.3, and 5.4, we present our solutions to address the two computational challenges for finding significant epistatic interactions. Section 5.5, we report results of a performance study on our own cluster and Amazon EC2. We also present an efficient approach to retrieve the top-k most significant answers. Finally, we discuss and conclude this chapter in Section 5.7.

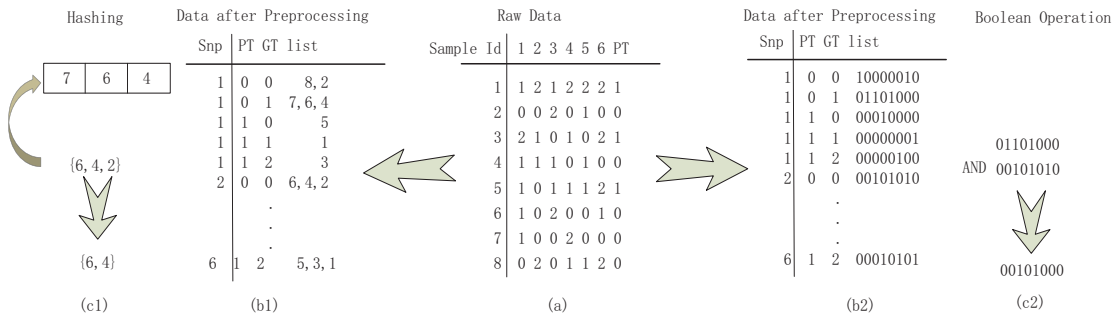

Figure 5.1: (a) is the raw data format with 6 SNPs from 8 individual samples; (b1) is the data format after pre-processing with sample id list in *CEO* model; (c1) illustrates the hashing method for finding the intersection between two lists of sample ids in *CEO* model- one is sample id list from the SNP 1 whose PT and GT are 0 and 1, the other is the sample id list from the SNP 2 whose PT and GT are 0 and 0 ; (b2) is the data format after pre-processing using bit strings representation in *eCEO* model; (c2) illustrates the way of finding the intersection from two lists with bit strings in *eCEO* model.

5.2 Background on statistical significance of SNP combinations

Typically a GWAS uses two types of data—genotype data that codes the genetic information of each individual, and phenotype data that measures the individual's quantitative traits. We use the same notation as Section 4.2.1 to describe the two types of data. Figure 5.1 (a) shows an example of the raw data format for a data set with 8 individual samples and 6 SNPs. Each row contains the individual sample information of raw data. The first and last columns are the sample id and phenotype. The rest of the columns are the genotype of each SNP.

The goal of our research is to identify a set of most significant combinations of multiple SNPs (epistatic interactions) that correlate to the phenotype. To measure the significance of the association between k-locus SNPs and phenotype in our *eCEO* model, we have implemented four test statistics—χ^2 test, likelihood ratio, normalized mutual information and uncertainty coefficient. Users can choose any of these tests based on their preferences and needs. As our *eCEO* framework assumes no statistical model fitting and is thus parameter free, these measures are effective in capturing interactions of arbitrary order. Without loss

of generality, we focus on the χ^2-test [Balding , 2006], which is used as the default in *eCEO*, in the following discussion.

5.3 Efficient algorithm for finding association significance

For our scheme to work, the raw data has to be pre-processed to collect the single SNP information in a new data format (to facilitate MapReduce processing). The straightforward way, which is adopted in our CEO model of the previous chapter [Wang et al., 2010b], is to reorganize the data into the format of $\langle SNP_i, PT, GT, list(sampleID)\rangle$ where SNP_i, PT and GT are the i^{th} SNP, phenotype value and the SNP genotype respectively. $list(sampleID)$ stores all the sample ids in the data set whose phenotype and SNP genotype on the SNP_i are PT and GT respectively. Figure 5.1(b1) depicts the transformed data from the raw data in Figure 5.1(a). The single SNP information can be sent to different processing nodes to calculate the χ^2-test value by collecting the contingency table from the combination of multiple SNPs. Let us still take two-locus as an example. In order to collect the contingency table, the first step is to calculate the $n_{i(j,k)}$ from the single SNP information. If we want to calculate the $n_{i(j,k)}$ for the pair of SNP x and SNP y, we need the information from $\langle x, i, j, list1(sampleID)\rangle$ in SNP x and $\langle y, i, k, list2(sampleID)\rangle$ in SNP y. We can derive $n_{i(j,k)}$ from the intersection between the two sample id lists. This can be easily done as follows: first, we build a hash table for the sample ids in the first list; second, we use the sample ids in the second list to probe the hash table for matching sample ids. For example, to get $n_{0(1,0)}$ for the pair of SNP 0 and SNP 1, we intersect the two sample lists as shown in Figure 5.1(c1).

However, our preliminary study suggests that using such an approach to collect the contingency table is computationally expensive.

In our *eCEO* model, we adopt an alternative approach. Instead of storing the sample ids in the list, we use a n-bit bit string to capture the sample ids, where n is the total number of samples. Each position in the bit string corresponds to a sample id. For example, from right to left, the first bit in the bit string

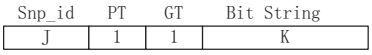

Figure 5.2: Data format in bytes. J, 1, 1, K bytes are used to store the SNP ID, phenotype, genotype and the bit string of the sample id list. User can choose the value of J and K according to their data size.

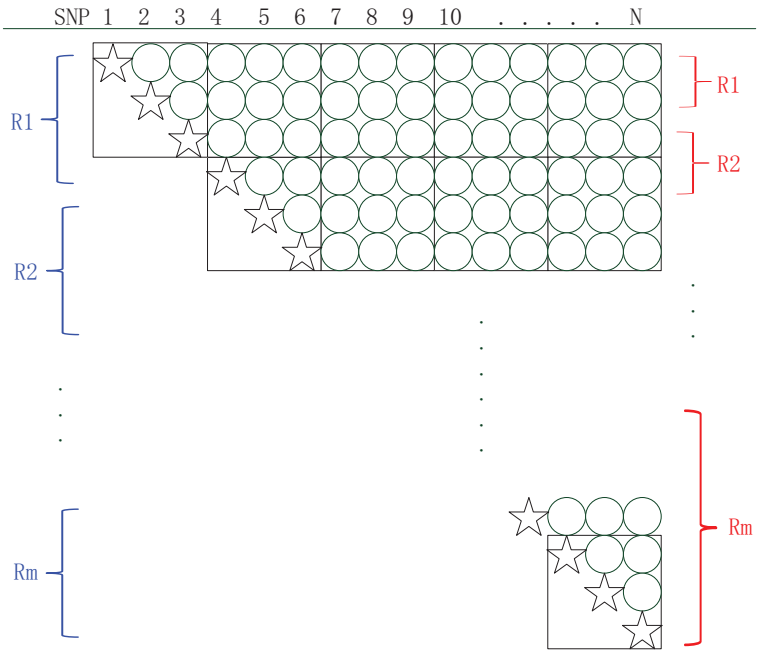

Figure 5.3: SNP-pairs representation and distribution to reducers

corresponds to the sample with id 1, the second bit corresponds to the sample with id 2 and so on. If the sample id is in the list, its corresponding position is set to 1, otherwise, it is set to 0. Therefore, in our system, the new transformed data is represented as bit strings as in Figure 5.1(b2). With such a representation, we can perform an AND operation on the two bit strings to find the intersection between them more efficiently. We can easily get the number of intersection samples from counting the 1's bits from the AND result. Figure 5.1(c2) depicts calculating $n_{0(1,0)}$ for the pair of SNP 0 and SNP 1 using bit strings. This method provides a more cpu-efficient way of collecting the contingency table.

To further improve performance, we incorporate several optimizations in our *eCEO* model. 1) We use the mutable decoding scheme in our system. From our observation, immutable decoding of objects from the key/values into Java

objects, used in the *CEO* model, is a time-consuming operation since it needs to create a unique Java object for each object in the key/values. For example, parsing ten objects in each record for one million records requires generating ten million immutable objects! With mutable decoding scheme, we can reuse ten mutable Java objects. Therefore, no matter how many records are decoded, only ten objects are created and reused. 2) We store all the data information into bytes including the SNP id, genotype, phenotype and bit strings as shown in the Figure 5.2. This follows from our observation that it is time-consuming to use Java string split function to split the objects in a record. We parse the objects in a record by directly fetching from the bytes records. 3) We write our own algorithm to count the intersection of the 1's bits without using the Java API.

5.4 Parallel distribution model

5.4.1 Two-locus epistatic analysis

For two-locus epistatic analysis, we aim at finding statistically significant interaction among all SNP pairs. For each pair of SNP combination, the p-value is computed (as described in Section 5.2) to determine its significance. For N SNPs in the data set, we need to calculate $\frac{N(N-1)}{2}$ two-locus SNPs combinations, as depicted in Figure 5.3. Each row represents a subset of SNP-pair computations where the starred node has to be paired up with a circled node. Thus, row 1 has (N-1) pairs, row 2 has (N-2) pairs and so on.

Our goal essentially is to split these $\frac{N(N-1)}{2}$ pairs of SNPs across all nodes to be processed in parallel. We have two issues to address here: (a) How do we split the SNP pairs across all nodes? (b) How to perform two-locus analysis under the MapReduce framework?

We shall first look at issue (a). Given N SNPs and M `reducers`, a naive approach is to simply distribute approximately equal number of rows to each `reducer`. This is depicted by the square brackets on the LHS of Figure 5.3 where the first $\frac{N}{M}$ rows are assigned to the first `reducer`, the next $\frac{N}{M}$ rows are assigned to the second `reducer` and so on. Here, the number of SNP-pairs can be easily determined without any additional meta-data, e.g., for row 1, we know that we

need to pair up SNP_1 (starred node) with all other remaining SNPs (circled node), resulting in (N-1) pairs. However, such a naive solution will result in load-imbalance as some `reducers` are more heavily loaded than others, e.g., `reducer` one is likely to be a bottleneck. To achieve better load balancing, we propose a new load-balanced solutions in this chapter:

Square-chopping Model. Under the Greedy model, the granularity of distribution of computation pairs is a single row. In some cases, if users have plenty of resources to use, they may want to reduce the number of computation pairs in each reducer further. Our Square-chopping model, which is an adaptation of the scheme in [Ma et al., 2008], can be used in these scenarios. Instead of sending the combination pairs according to rows, we distribute them by "Square-chopping" as shown in Figure 5.3. This can be achieved by dividing N SNPs into m subsets evenly. Each subset has n SNPs where n equals $\frac{N}{m}$. For simplicity, n is assumed to be integer. Then we assign any two subsets into one reducer. As shown in Figure 5.3, each off-diagonal reducer receives n^2 combination pairs and each diagonal reducer receives $\frac{n(n+1)}{2}$ combination pairs. Therefore, this scheme needs $\frac{m(m+1)}{2}$ reducers.

We are now ready to look at issue (b), i.e., performing two-locus analysis in MapReduce framework. The basic principles of *map*, *shuffle* and *reduce* phases are described in Section 4.3.1. However, some new technical solutions are mentioned below.

Map Phase: In Figure 5.3, SNP_N needs to be shuffled to all the `reducers`. This is not supported by the MapReduce framework which allows only one output $(key, value)$ pair emitted from a `mapper` to be shuffled to one `reducer`. We resolve this problem by replicating and emitting as many copies of a SNP as required. In addition, each such pair is "tagged" with the corresponding `reducer` identifier to distinguish the `reducer` that the pair should be shuffled to. In other words, for each `reducer` for which an SNP, SNP_i, should be shuffled to, we generate and emit a $(key, value)$ pair where `key` is set as $SNP_i.reducer\_marker$ ($reducer\_marker$ is the identifier of the `reducer` that this SNP_i should be shuffled to), as shown in the Figure 4.3 subgraph (2), and `value` contains the rest of the SNP information including the genotype, phenotype and the bit string representing the sample id list. In this way, all the output $(key, value)$ pairs with

the same *reducer_marker* are shuffled to the same `reducer`.

Reduce Phase: Because all the keys at the `reducer` have the same `reducer_marker`, the keys will be sorted only based on the SNP_i. Thus, the data for SNP_i are sent to the `reduce` function before those for SNP_j where $i < j$. For the Greedy model of previous chapter, this means that the starred nodes are supplied earlier than the circled nodes. Therefore, in each `reducer`, only the starred nodes need to be cached in the main memory. As the circled nodes are received, they can be immediately paired up with the starred nodes to compute its p-value, after which the circled nodes can be discarded. For the Square-chopping model of this chapter, this guarantees that information from one subset of SNPs will be supplied earlier than another. Therefore, we only need to keep one subset of the SNPs information in memory. As such, our *eCEO* model significantly reduces the memory utilization.

In our processing model, the two-locus analysis finishes in one MapReduce job.

5.4.2 Three-locus epistatic analysis

Three-locus epistatic analysis aims at finding statistically significant interaction between three SNPs. The way we conduct three-locus epistatic analysis is described in Section 4.3.2. However, the load balancing algorithm can also be used here for optimization.

5.5 Results

Apache Hadoop is an open source equivalent implementation of the MapReduce framework, running on HDFS (Hadoop distributed file system). We conduct a series of experiments on our local cluster with over 40 nodes, and a public cloud environment, Amazon Elastic Compute Cloud (Amazon EC2). For our local cluster, each node consists of a X3430 4(4) @ 2.4GHZ CPU running Centos 5.4 with 8GB memory and 2x 500G SATA disks. For Amazon EC2, we use 20 extra large instances, each with 8 EC2 Compute Units (4 virtual cores with 2 EC2 Compute Units each), 15 GB of memory, and 1690 GB of local instance

storage running on a 64-bit platform. Moreover, since our tasks at hand are computationally intensive, we set the number of `reducers` per node to be equal to the number of cores at the node, which is 4 in our local cluster and 8 in EC2 instances. This guarantees that each `reducer` can get one core. Therefore, there are a total of 4*N and 8*N `reducers` which can be run simultaneously on a N-node local cluster and EC2 clusters respectively.

Effect of number of reducers: For Hadoop application, a user can specify the number of `reducers` to be used in one job. Because we have preconfigured the total number of `reducers` to be 4*N for a N-node cluster, this may require multiple phases to complete a job. For example, if N=30, then by specifying 120 `reducers` in one job, we can complete it in 1 phase; with 360 `reducers`, it will then take 3 phases to complete the job. Our first experiment is to investigate the optimal number of `reducers` that should be set for one job based on a given cluster size. This experiment is conducted with a 50,000-SNPs dataset on a local 30-node cluster. Note that all the datasets we used include 2000 samples.

Figure 5.4 presents the running time for the Greedy model. As shown, there is a certain optimal number of `reducers` that should be used. When the number of `reducers` is too small, the computation resources are not fully utilized. On the other hand, when the number of `reducers` is too large, the processing may require multiple phases which increases the communication overhead. We note that the Greedy model is optimal when the number of `reducers` corresponds to the actual configured value (i.e., 120).

Figure 5.5 presents the running time for the Square-chopping model. Here, the 50,000-SNPs are evenly split into 10, 16, 20, 25 and 40 partitions corresponding to 55, 136, 210, 325 and 820 reducers needed. From the results, we observe that when the reducer number is close to a multiple of N, where N is the total number of reducers configured in the cluster, its performance is good; otherwise, (N - R%N) reducers in the last phase, where R is the reducer number set in the job, will be wasted.

Looking at the results for the Greedy and the Square-chopping models, we observe that the Square-chopping model is generally inferior to the Greedy model. This is because of wasted reducers in the last phase (as discussed above). Its performance, however, is closer to the Greedy model as the partition number in-

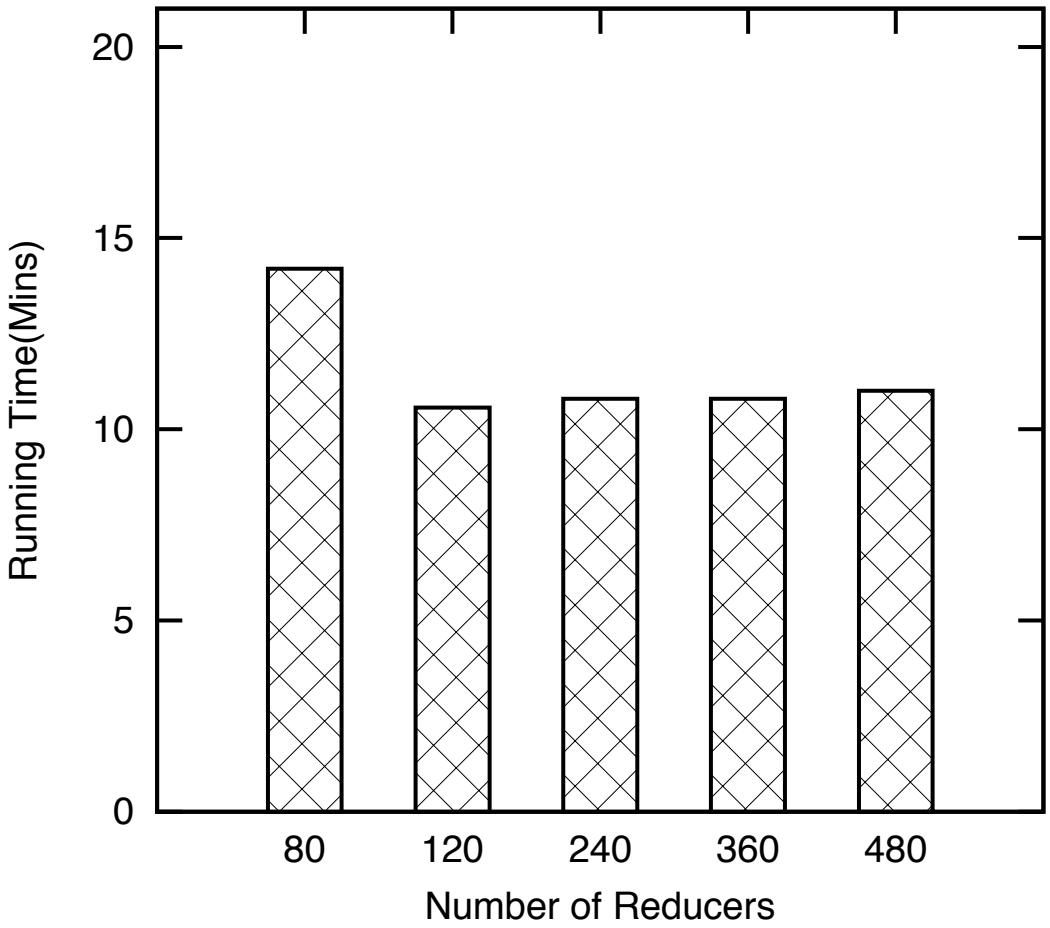

Figure 5.4: Effect of number of reducers for Greedy model

creases because the task in each reducer is smaller, and hence the wasted reducers in the last phase will not affect the total performance so much. Based on these results, for the subsequent experiments, we only use the Greedy model.

Scalability: First, we study the scalability of the *eCEO* model as the system resources increase. Figure 5.6 shows the completion time to analyze 50,000-SNPs as the cluster sizes increases from 10 to 40 nodes. The `reducer` numbers in each job are set as 40, 80, 120 and 160 respectively. From the result, we can see that completion time reduces with increasing number of nodes. In fact, we observe a (almost) linear speedup in performance. When we double the resources, the execution time reduces by half.

Now, let us consider the scalability of *eCEO* as the number of SNPs increases.

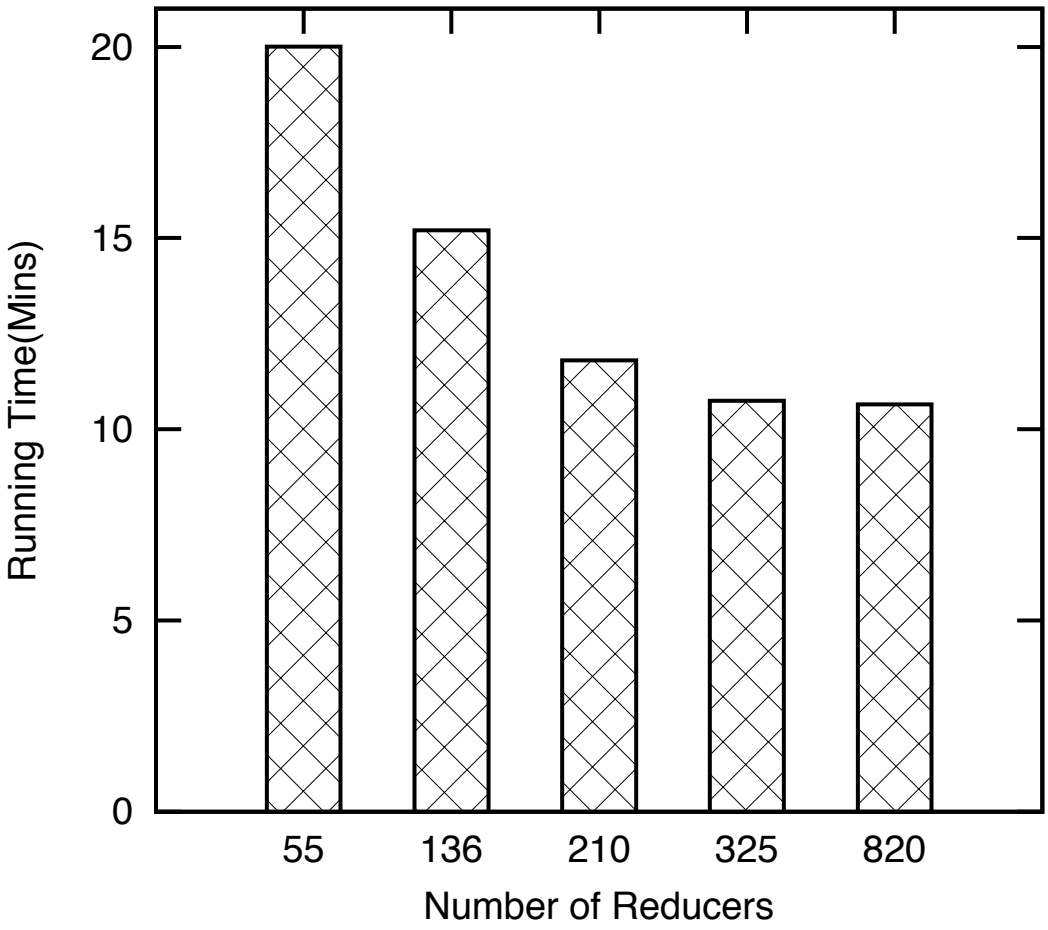

Figure 5.5: Effect of number of reducers for Square-chopping model

Figure 5.7 shows the processing time for exhaustively computing all the significant interactions for two-locus with 50,000, 100,000, 200,000 and 500,000 SNPs on a local 43-node cluster, and output the results whose p-values are smaller than 0.05. We make two interesting observations. First, the result shows that our *eCEO* offers a feasible and practical solution to perform pairwise epistasis for a large number of SNPs. According to [Ma et al., 2008], it would require 1.2 years to do the pairwise epistasis testing of 500,000-SNPs using the serial program on a 2.66 GHz single processor without parallel processing. Our eCEO model can accomplish this task in no more than 9 hours using only a 43-node cluster. Second, we note that the processing time is essentially proportional to the number of interacting SNP-pairs to be evaluated. For example, the number of SNP-pairs

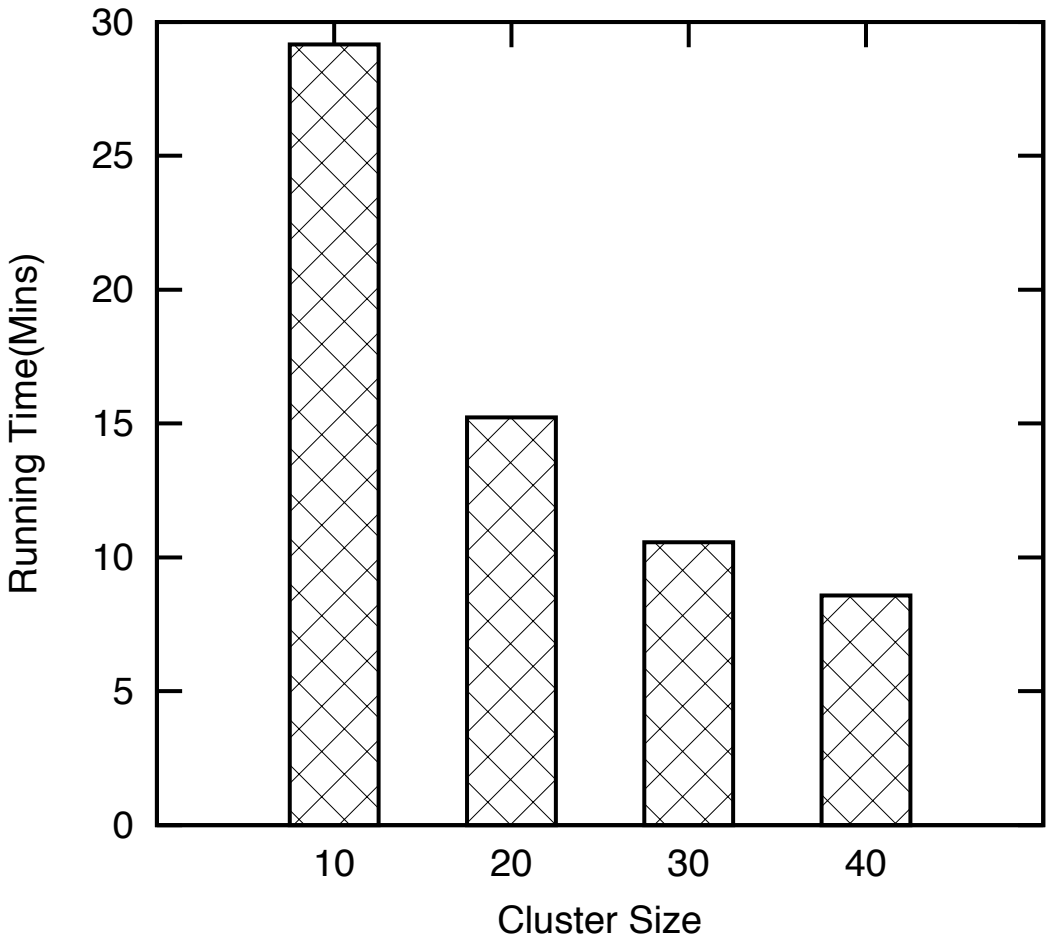

Figure 5.6: e*CEO* Scalability on different clusters

for the 500,000-SNPs dataset is 100 times more than that for the 50,000-SNPs dataset, and 6 times more than that for the 200,000-SNPs dataset. The running time for the 500,000-SNPs dataset (538 mins) is no more than 100 times that of the 50,000-SNPs dataset (7 mins), and is about 5 times more than the 200,000-SNPs dataset (109 mins).

Performance Comparison between CEO and eCEO models: We also evaluate CEO's scalability with respect to the number of SNPs. The result is shown in Figure 5.7. Clearly, *eCEO* outperforms *CEO* by a wide margin. We did not run the experiments for 200,000 and 500,000 SNPs in the *CEO* model because it will take a long time—we estimated the execute time for 500,000 SNPs to be roughly 25∼30 days. But our *eCEO* model only needs 9 hours to complete. We expect

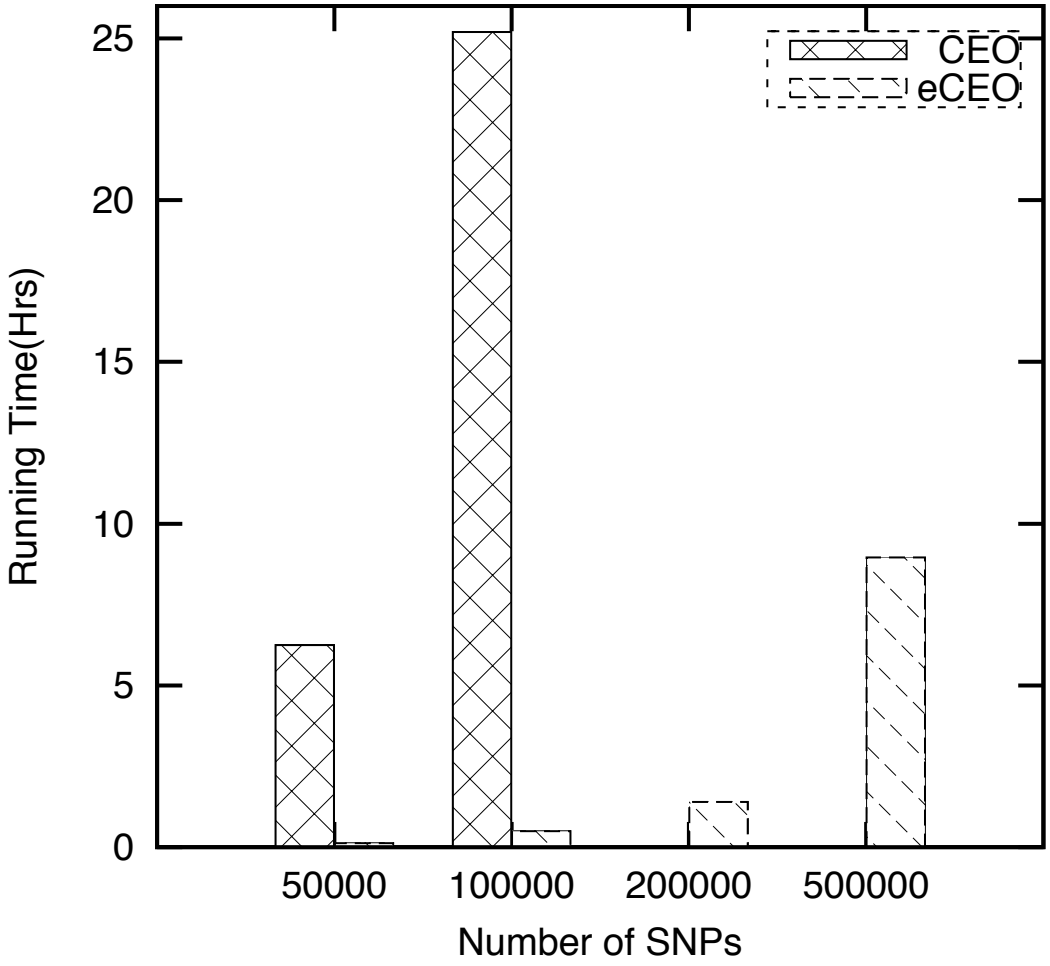

Figure 5.7: *CEO* and e*CEO* performance comparison

our *eCEO* model to be able to process 1 million SNPs in around ten hours on a 200-node cluster. The results confirm that our various optimizations are effective and efficient, and that our *eCEO* model is a practical and effective solution for processing large number of SNPs.

Three-locus analysis: We also evaluate the performance of three-locus analysis on a 43-node local cluster. We output all the two-locus analysis result and then perform the three-locus analysis. The result is presented in Figure 5.8 for SNP sizes of 2,000, 3,000, 4,000 and 5,000. We observe that the running time is also proportional to the number of SNP-triples. This confirms that the *eCEO* scheme can effectively balance the load across all nodes.

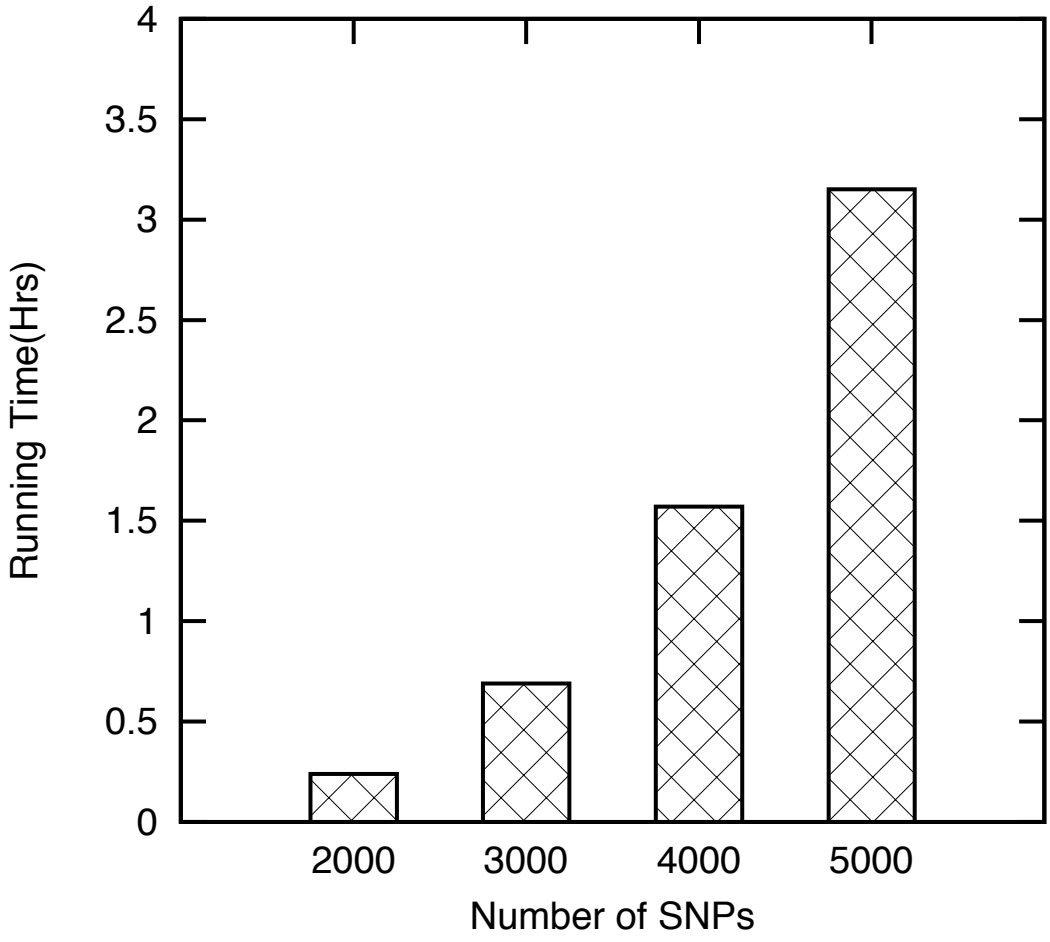

Figure 5.8: Three-locus epistatic analysis

Top-K Retrieval: In our system, we store the results of the two-locus and three-locus analyses in HDFS to allow users to do further analysis. One important function that we can further provide is to allow users to retrieve only the top-k most significant results with the lowest p-values. We have also provided such a capability in our system under the MapReduce framework. The basic idea is to split the output of the two/three-locus analysis into chunks. Each chunk is then assigned to one `mapper`. Next, each `mapper` will select the top-k most significant pairs/triples and shuffled these results to one `reducer`. Finally, the `reducer` can determine the global top-k answers based on all local top-k ones it receives. Our top-k scheme is very efficient. For example, retrieving the top 10 most significant SNPs from the two-locus output (with size of 56GB) only takes 145 seconds in

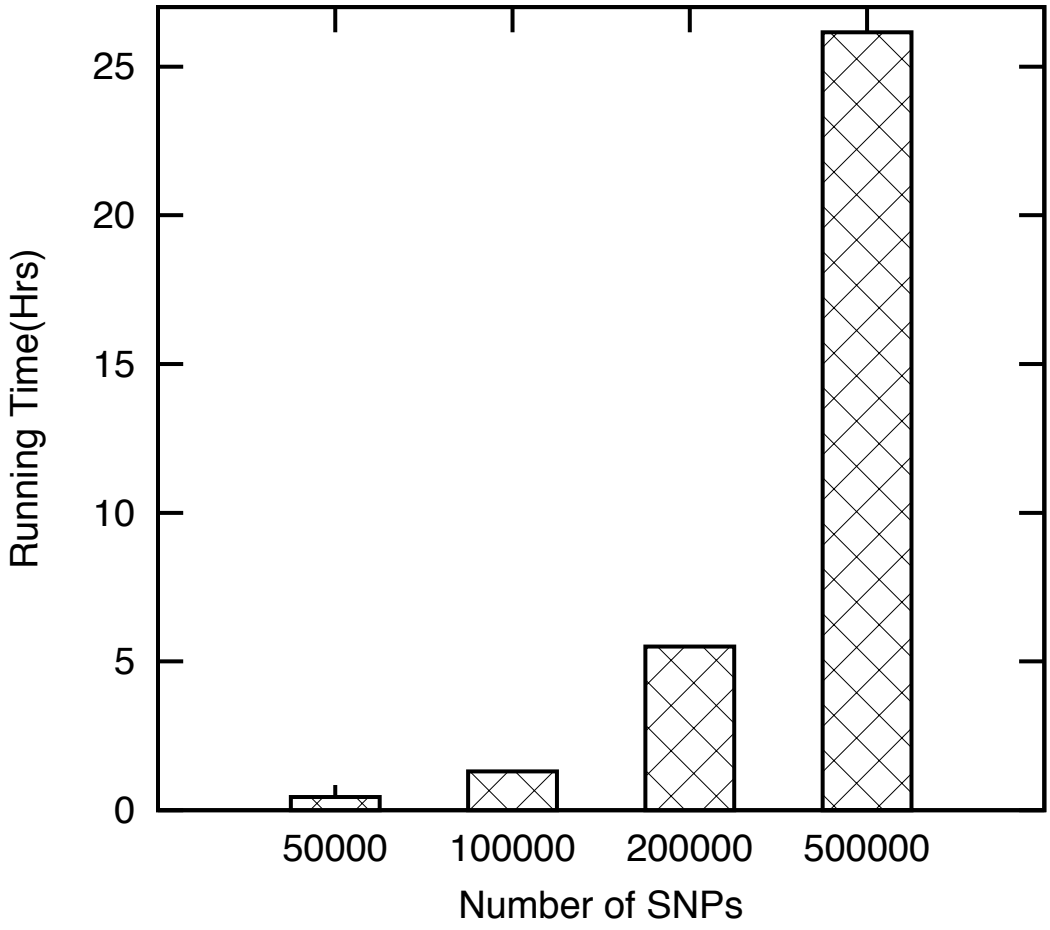

Figure 5.9: e*CEO* performance on EC2

the 43-node cluster.

Evaluation on and experience with a public cloud: eCEO is developed with the intention for users to exploit cloud computing for epistasis analysis. As such, we also evaluate our Greedy model on a public cloud, namely, Amazon EC2. Our quota of using Amazon EC2 instances in our research grant is 20. We use 20 extra-large instances in our experiments, including 1 master node and 19 slaves nodes. There are 19 computation nodes in this experiment. Figure 5.9 shows the execution time for two-locus analysis as we vary the number of SNPs from 50,000 to 500,000. From the results, we can see that the execution time is essentially proportional to the number of interacting SNP-pairs as we observed in our local cluster.

Our experience with Amazon EC2 shows the ease in which we can deploy our *eCEO* model. In fact, in the Hadoop package that we use, it provides tools to launch Amazon EC2 cluster with Hadoop directly. Therefore, we do not need to make any changes to our codes. We do not even need to set up Hadoop at all. Once we launch the cluster in Amazon EC2, we simply upload our *eCEO* program and run it. With many cloud providers offering services to use MapReduce program directly (such as Amazon EC2, Amazon Elastic MapReduce and so on), our *eCEO* model is an important tool for large-scale epistasis analysis on a public cloud.

5.6 Theoretical cost analysis and suggestion for a major improvement

5.6.1 Theoretical cost analysis

In the Amazon EC2, the cost that a user needs to pay is measured by how much instance hour is used. In our two proposed models, the time can be broken down into three parts given the same configuration of instances used in AWS:

1. data transfer from *map phase* to *reduce phase*;
2. data to be loaded into memory and
3. computation in the memory.

We use following symbols to derive the cost of each part:

- ns: the number of SNPs to be considered;
- nu: the number of SNP pairs that an instance on AWS is able to process within 1 instance hour and
- nn: the number of instances used in the AWS, $nn = ns * (ns - 1)/(2 * nu)$.

(1) How much data transfer is needed.

The data transfer can be divided into head SNP transfer and paired SNP transfer.

Head SNP files are copied once each, and paired SNP files are copied as many times as they are paired to head SNPs in a instance. For both the Greedy and the Square-chopping model, the number of times paired SNP files are transferred works out to nu times, where nu is the number of SNP pairs that can be computed in one instance hour on one instance. However, the number of times that head SNP files are transferred is different. For the Square-chopping model, there are $ns * (ns - 1)/2$ SNP pairs to be computed. Each node is processing nu pairs and we have nn nodes. At each of these nodes, $sqrt(nu)$ SNPs are paired to $sqrt(nu)$ SNPs. The number of times head SNP files are transferred is $sqrt(nu)$ for each instance. For the Greedy model, the number of times head SNP files are transferred is different for different instances, but the maximum number of head SNP is $sqrt(nu)$. Therefore, the data transfer cost overall is slightly less for the Greedy model than that for the Square-chopping model.

(2) How much data is loaded into memory.

For both the Greedy and Square-chopping models, the transferred data are loaded into memory. When the memory is full, the data to be loaded into memory reside at local disks of an instance. They are loaded after an instance finishes calculating the current SNP pairs test statistics and clears the current memory. Therefore the cost of loading data is a little less for the Greedy model than that for the Square-chopping model.

(3) SNP pair computing cost.

When a SNP pair is loaded into memory, it takes the same cost to calculate for both the Greedy and Square-chopping models. Since they calculate the same amount of SNP pairs ($ns * (ns - 1)/2$)) and we assume both models run in the same configuration of instances (nn and nu). The cost is the same for the Greedy and Square-chopping models.

Adding up the above three parts, the total cost for the Greedy model is slightly less than that for the Square-chopping model, which is shown in Figure 5.4 and Figure 5.5 (running time for the Greedy model is slightly less than that for the Square chopping model), assuming our local clusters simulate the behavior of AWS EC2.

5.6.2 Suggestion for a major improvement

In the current implementation, repetitive head SNP and paired SNP files are transferred and therefore repetitive costs are incurred. A better implementation can reduce cost by transferring the head SNP and paired SNP identifiers instead of actual SNP files, assuming there is a cache mechanism that can store all the preprocessed SNP genotyping files. Amazon EBS service provides such a cache mechanism in the sense that it loads once and only once the absent files when an instance tries to access them. After that the files are kept locally for accessing. Therefore, both models should mount the EBS volume on all EC2 instances so that SNP files (one file per SNP) are stored on the EBS. When an instance uses the SNP file of a paired SNP for the first time, the file is obtained and cached in that instance. The SNP file of this paired can be reused later since there is a cached copy, instead of getting another copy like our current implementation. Such an improvement can therefore reduce the cost of transferring data and loading data into memory.

Future work may investigate the exact amount of the reduction given the same configure of instances in AWS. The reduction in these costs is expected to be significant:

(1) How much data transfer is needed.

For the Greedy model, the cost of data transfer for each instance is now equal to the number of SNP files that this instance needs to access—i.e., the number of head SNPs and paired SNPs that it has to process. Specifically, instance$_1$ needs to see the files of ns SNPs, including those of the nu/ns head SNPs assigned to this instance; instance$_2$ needs to see the files of $(ns - nu/ns)$ SNPs, including those of the $nu/(ns - nu/ns))$ head SNPs; instance$_3$ needs to see the files of $(ns - nu/ns - nu/(ns - nu/ns))$ SNPs, including those of the $(nu/(ns - nu/ns - nu/(ns - nu/ns)))$ head SNPs; and so on, until instance$_{nn}$ which needs to see roughly the files of approximately the last $2 * sqrt(nu)$ SNPs, including those of approximately $sqrt(nu)$ head SNPs. Thus each instance in the Greedy model generally needs to see the files of at least $2 * sqrt(nu)$ SNPs. Since there are nn instances, the total number of SNP files to be transferred exceeds $nn * 2 * sqrt(nu)$, but is at most $nn * ns$. In contrast, for the Square-chopping model, each

instance has $sqrt(nu)$ head SNPs and $sqrt(nu)$ paired SNPs. Thus each instance needs $2 * sqrt(nu)$ SNP files. So the total number of files to be transferred is $nn * 2 * sqrt(nu)$. That is, with this modification to the implementation, the Square-chopping model needs much less data transfer than the Greedy model, reversing the situation in our current implementation. Furthermore, the amount of transfer needed in both models with this modification to the implementation is significantly less than that in the current implementation ($nn * 2 * sqrt(nu)$ versus $nn * nu$).

(2) How much data is loaded into memory.

In the current implementation, when a SNP is paired to $ns - 1$ head SNPs in an instance, its associated data is transferred and loaded $ns - 1$ times into the memory of that instance. With the improved implementation, it is loaded only once, so long as that instance has enough memory to hold the data of its entire head SNPs and one paired SNP in memory. Therefore, the total amount of data being loaded into memory for the Greedy model is at least $nn * 2 * sqrt(nu)$ and at most $nn * ns$, while that for the Square-chopping model is $nn * 2 * sqrt(nu)$. Again, with this modification to the implementation, the Square-chopping model becomes more efficient than the Greedy model. Furthermore, both the Square-chopping model and the Greedy model become more efficient than the current implementation ($nn * 2 * sqrt(nu)$ versus $nn * nu$).

(3) SNP pair computing cost.

The modification to the implementation does not change the SNP pair computing cost.

5.7 Conclusion

This chapter aims at providing an efficient epistasis computing model for large-scale epistatic interaction in GWAS which can be run on a computing cluster (local or cloud-based). We have proposed an efficient and feasible solution, called *eCEO* based on the MapReduce framework. As such, *eCEO* inherits the nice properties of MapReduce, which is high scalability and good fault tolerance. Moreover, it can leverage cloud computing with almost unlimited elastic computing resources. We have demonstrated the practical advantage of using *eCEO*

model to exhaustively search two-locus and three-locus epistatic interactions. Our *eCEO* model can also retrieve top-k most significant interactions. We have conducted extensive experimental study on a local cluster of over 40 nodes and on 20 instances on Amazon EC2. The results showed that our *eCEO* model is computationally efficient, flexible, scalable and practical. As future work, we plan to implement more test statistics. We also plan to explore the possibility of integrating *eCEO* as a filtering step to other methods, e.g., those based on statistical model fitting. Finally, we plan to develop pruning strategies based on domain knowledge, and integrate these into our scheme. For example, by knowing that certain SNPs do not interact, their computations can be avoided totally.

Chapter 6

Parallel random forest regression on Hadoop for multivariate quantitative trait mapping

6.1 Introduction

The last few years have seen extensive efforts to correlate human genetic and phenotypic variation. An increasing number of population GWAS have been carried out to discover causal associations between common genetic variations and complex human traits. These studies rely on high-throughput platforms that measure genetic changes at hundreds of thousands or even million single-nucleotide polymorphisms (SNPs) across the human genome in large random samples. Full sequencing of human genomes has shown that, in any given individual, there are on average approximately 4 million genetic variants [Frazer *et al.*, 2009]. The most common study design generally involves comparing a sample of healthy control subjects with a sample of diseased subjects, with the goal of identifying patterns of polymorphisms that vary systematically between these two populations and could, therefore, represent the effects of risk-enhancing alleles. Such abundance of genetic markers has now made it possible to identify quantitative trait loci (QTL), which are regions of a chromosome or even individual sequence variants that are responsible for trait variation. For many diseases, such as asthma or

attention deficit hyperactivity disorder (ADHD), investigators routinely measure multiple *endophenotypes* that are thought to be more proximal to the biological etiology of the clinical disorder [Ferreira and Purcell, 2009].

Traditional statistical genetics methodologies have started to be complemented with, or even replaced by, machine learning algorithms because they often make minimal assumptions about the underlying causal disease mechanism, which is generally unknown. Case-control studies can be analysed by performing feature (e.g. SNP) selection and ranking in the context of pattern classification. Random Forest (RF), which is amongst the top performing algorithms for supervised learning, has been found particularly promising in case-control studies [Boulesteix *et al.*, 2012; Goldstein *et al.*, 2010, 2011; Nicodemus, 2011]. Phenotypic variation in human populations is typically due to underlying genetic complexity from multiple interacting loci, with allelic effects that are sensitive to the environmental conditions each individual experiences. RF has also been regarded as a particularly powerful approach to capture gene-environment interactions and epistatic effects [Jiang *et al.*, 2009; Lunetta et al., 2004; Molinaro *et al.*, 2011; Sun, 2010].

Our interest in this work is on detecting genetic variants associated with quantitative, and possibly multivariate and very high-dimensional, traits. When a QTL is found to be linked to a causative marker locus, then individuals with different marker locus genotypes will have different mean values of the quantitative trait. In this respect, the QTL mapping problem can also be treated as a feature selection and ranking problem, albeit in a regression setting. Several studies mapping QTL that affect human diseases and complex traits have uncovered new loci. Although much emphasis has been placed on linkage mapping, or QTL mapping in families, there is now increasing interest for QTL mapping in unrelated individuals from the same population, or association mapping [Mackay *et al.*, 2009]. An instance of association mapping with very high-dimensional quantitative traits is found in the area of *imaging genetics*, an emerging field that is rapidly identifying genetic variants that influence the brain structure and function [Glahn *et al.*, 2007; Meyer-Lindenberg, 2012; Smit *et al.*, 2012]. Many research groups are now scanning unrelated individuals with structural and functional MRI (Magnetic Resonance Imaging), DTI (Diffusion Tensor Imaging) and other imaging modalities to characterise variability in the brain. In whole-brain

studies, an imaging phenotype may consist of thousands or millions of measurements in a 3D space, representing for instance gray matter intensities. These studies create important statistical challenges due to the very high dimensionality of the quantitative trait being observed. Power gains can be expected by analysing all these measurements jointly, rather than performing multiple independent analyses each involving a univariate response or using summary measures [Vounou *et al.*, 2012]. The use of such multivariate heritable imaging signatures of disease may increase the power in detecting causal variants, when compared with a simpler case-control status, since gene effects are expected to be more penetrant at the imaging level, rather than at the diagnostic level [Hibar *et al.*, 2011; Meyer-Lindenberg and Weinberger, 2006]. Although neuroimaging genetics studies have already identified coherent anatomical patterns of gene effects in three-dimensions using advanced statistical methods [Stein *et al.*, 2010a,b; Silver *et al.*, 2012; Vounou *et al.*, 2012], the potential of machine learning methods in that area has not yet been fully explored, and this may be due to the lack of scalable implementations.

We describe here a parallel implementation of RF for regression problems with multivariate responses. Our implementation has been specifically designed to run on large Hadoop clusters, including those available through cloud computing services such as Amazon Elastic MapReduce. The Hadoop ecosystem consists of a set of tools for building distributed systems, including tools for storage, data analysis, and coordination, thus enabling algorithms to be run on thousands of computational nodes. Hadoop was originally designed to address two main issues that arise when distributing data and computations across a very large cluster. First, the problem of hardware failure, which is addressed through replication; redundant copies of the data are kept by the system so that in the event of failure, there is always another copy available. Second, the problem of reliably combining the data resulting from various parallel computations from potentially many nodes and disks. The latter problem is addressed by adopting the MapReduce programming model [Dean and Ghemawat, 2004]. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. Hadoop is currently an Apache project, written in Java, and distributed under a free license.

This chapter is structured as follows. Section 6.2 provides a concise description of the RF algorithm, including an alternative node splitting criterion for tree building that is computationally convenient when the trait is high-dimensional, and a procedure for ranking SNPs in order of their predictive importance. We also present the strategy adopted to parallelize the algorithm using the MapReduce programming model. The motivating application and data set—an imaging genetics study of Alzheimer's disease—are described in Section 6.3. In Section 6.4 we report on experimental results aiming to showcase the scalability properties of the proposed software, and describe the real data analysis. We conclude by providing an overview of alternative parallel RF algorithms and some remarks on further work in Section 6.5.

6.2 Methods

6.2.1 Random forest regression

We call $\mathcal{D}$ the data set comprising N unrelated individuals or samples genotyped at P biallelic markers. For each individual, the markers are arranged in a data vector $\boldsymbol{x}_i = (x_{i1}, x_{i2}, ..., x_{iP})$, for $i = 1, \ldots, N$. Depending on the coding scheme, different genetic models can be applied. For instance, assuming an additive genetic model, each x_{ij} represents the count of minor alleles recorded at the j_{th} locus—homozygote of minor allele is 2, heterozygote is 1 and homozygote of major allele is 0. The associated quantitative trait for each subject is assumed to be a Q-dimensional real-valued vector which we denote as $\boldsymbol{y}_i = (y_{i1}, y_{i2}, ..., y_{iQ})$, with $i = 1, \ldots, N$. In imaging genetics study designs, for instance, it is common that the sample size N is much smaller than $\min\{P, Q\}$.

The RF algorithm builds an ensemble of regression trees, each one independently learned on a bootstrapped version of $\mathcal{D}$. The required number of trees in the forest, `Ntree`, is a user-defined parameter. The training data set for each tree is obtained by randomly sampling N subjects from $\mathcal{D}$ with replacement. The tree building process is accomplished by introducing a second layer of randomness and involves selecting a random subset of `Mtry` candidate SNPs at each node, among the P available SNPs, in order to reduce the correlation among trees. In

each tree, the best split at a node is determined by evaluating a split function for each value of a candidate SNP, and then selecting the SNP that maximises this function (see also Section 6.2.2). To reduce bias, the trees are grown to a maximum depth with no pruning or otherwise until a minimum sample size has been reached; by default, we set this value to 5 for univariate trait and 20 for multivariate traits. We only consider binary trees, although in principle multi-way splits could also be accommodated with minor changes.

For each tree, all the subjects in $\mathcal{D}$ that do not become part of the bootstrap sample used for training are collected together to form an *out-of-bag* (OOB) sample, which is used as a testing set. Approximately 63.2% of the subjects in $\mathcal{D}$ are utilised as training data, while the remaining subjects are OOB samples. Each OOB sample is used to obtain an estimate for the prediction error (PE) for its tree and these estimates are then averaged across all trees to provide an overall estimate [Breiman, 2001].

Although RF is deemed to be relatively insensitive to the choice of `Ntree` and `Mtry`, in practice, for a large-scale GWAS involving a massive number of predictors, and possibly multivariate responses, these parameters must be tuned to achieve an optimal predictive performance and increase the statistical power of the algorithm to detect the true causative SNPs. As the number of trees in the forest increases, the OOB error rate is expected to converge to a theoretical prediction error according to the law of large numbers [Breiman, 2001]. It is therefore important to select a sufficiently large number of trees to guarantee optimal performance and stable ranking.

6.2.2 Split functions for multivariate traits

The node splitting rule determines how each tree in the forest is built, and depends on the particular predictive task at hand. For each node j, two operations are performed: (a) every allowable split on each SNP is examined; (b) the best of these split is selected, and the left and right daughter nodes are created. The initial node is the root node, which contains the entire data set $\mathcal{D}$, and the two operations above are then applied repeatedly to each daughter node until no more splits can be obtained. During this process the value of a split function $\phi(s_m, j)$

is computed for every split at node j. In regression tasks with both univariate and multivariate responses, sum of squares functions are commonly used [Segal, 1992]. In what follows, we let $\mathcal{D}(j)$ denote the subset of samples associated with node j, and $\mathcal{M}_j$ the set of `Mtry` candidate SNPs that are available to split node j. Furthermore, the mean response vector observed in $\mathcal{D}(j)$ is denoted $\bar{\boldsymbol{y}}(j)$. With this notation in place, the total sum of squares at node j is

$$SS(j) = \sum_{i \in \mathcal{D}(j)} (\boldsymbol{y}_i - \bar{\boldsymbol{y}}(j))^T \boldsymbol{V}^{-1}(\boldsymbol{\Theta}, j)(\boldsymbol{y}_i - \bar{\boldsymbol{y}}(j)) \tag{6.1}$$

where $\boldsymbol{V}(\boldsymbol{\Theta}, j)$ is the $Q \times Q$ covariance matrix estimated from $\mathcal{D}(j)$, and depends on an unknown parameter vector $\boldsymbol{\Theta}$. A fully parametrized covariance matrix requires $Q(Q+1)/2$ parameters. With low sample sizes, a more parsimonious model is generally preferable so that $\boldsymbol{\Theta}$ has only a few dimensions. It is standard procedure to set the covariance matrices at the daughter nodes equal to the estimated covariance matrix at the parent node, in order to guarantee that the split function remains positive [Segal, 1992]. This procedure has the additional benefit of reducing the number of computations. We term the RF using the criterion in (6.1) as the Standard RF.

When a SNP is selected as a candidate to split node j into two daughter nodes, the total sum of squares computed at the left and right daughter nodes are $SS(j)_l$ and $SS(j)_r$, respectively. A suitable function in this case measures the reduction in the sum of squares due to the split, and is given by

$$\phi(j) = SS(j) - SS(j)_r - SS(j)_l. \tag{6.2}$$

Every candidate SNP is tested to split the node, and the one with the highest $\phi(j)$ is selected. Once the best split has been found, the daughter nodes become new parent nodes and the covariance matrices are estimated again.

Parsimonious parametrisation of the covariance matrices are required in order to keep the computational burden low. In high-dimensional settings, and especially when N is much smaller than Q, it is commonplace to assume that the covariance matrices are diagonal [Segal and Xiao, 2011; Vounou *et al.*, 2012]. For instance, typical whole-brain imaging genetics studies may involve a few hundred

thousands brain-wide measurements while the sample remain in the order of a few hundreds [Stein *et al.*, 2010b]. In this situation, the total sum of squares of Eq. (6.1) can be alternatively expressed in terms of all $N \times N$ squared inter-point Euclidean distances between all N samples [Minas *et al.*, 2011]. By rewriting Eq. (6.1) in an equivalent form, when $\boldsymbol{V}(\boldsymbol{\Theta}, j) = \text{diag}(1, \ldots, 1)$,

$$SS(j) = \frac{1}{2N(j)} \sum_{i \in \mathcal{D}(j)} \sum_{l \in \mathcal{D}(j)} d_E^2(\boldsymbol{y}_i, \boldsymbol{y}_l), \tag{6.3}$$

where $N(j)$ indicates the sample size at node j. This strategy provides an equivalent but computationally more efficient way of evaluating the split function of Eq. (6.2). The evaluation of each $SS(j)$ term has a cost complexity of $\text{O}(N(j)^2)$ instead of $N(j) \times Q$. We term the RF using the distance calculation in (6.3) as the Distance-based RF.

6.2.3 Measure of variable importance for SNP ranking

One of the attractive features of RF for GWAS consists in its ability to perform SNP ranking by computing a measure of variable importance [Goldstein *et al.*, 2011]. A commonly used and computationally simple procedure for SNP ranking consists in monitoring the value of the split function $\phi(j)$ every time a particular SNP x_m has been selected as the best split, in each tree, and then add up all these cases as its importance score. The rationale is that the more chance a SNP is used to split a node, the more influential it is to the forest. We call this the *information gain based importance score*. This criterion is usually consistent with other variable importance measures that are based on permutations which are computationally more expensive [Breiman, 2001]. In the context of genetics studies, SNPs with the highest importance score are preferred candidates for further exploration. In the literature, this approach has been used as a prescreening step to choose informative SNPs for further analysis [De Lobel *et al.*, 2010].

6.2.4 Hadoop implementation

RF implementations generally build trees sequentially. However, a sequential approach is highly inefficient, especially when each tree involves a large number of SNPs, and many trees are needed in order to obtain reliable measures of SNP importance and PE estimates. RF can be parallelised because all trees are independently learned from randomised versions of the data. We describe here a parallel version of the RF regression algorithm that we have implemented using the MapReduce programming model for deployment on large Hadoop clusters. Broadly speaking, the approach consists in letting each node in the cluster build a certain number of trees in the forest, and then letting the system collect and aggregate the partial results from all trees in the ensemble, in an automated and fault-tolerant fashion.

The MapReduce model involves three phases: the *map* phase, the *shuffle* phase and the *reduce* phase. Each one of the *map* and *reduce* phase has key-value pairs as input and output. The *shuffle* phase shuffles the output of *map* phase to the input of *reduce* phase evenly using the MapReduce library. The *map* phase runs a user-defined *mapper* function on a set of key-value pairs $[k_j, v_j]$ taken as input, and generates a set of intermediate key-value pairs. In the map phase of our application, each input key corresponds to a unique tree ID and value is NULL since we load the full data set to build trees. A user-defined number of mappers, `nmap`, are executed whereby each mapper function learns one or more decision trees from bootstrapped versions of the data set. The output of the map phase consists of three types of information: (1) Sample identifier (key) and sample's regressed value from RF (value), which is used to estimate the OOB error rate at the reduce phase; (2) SNP identifier (key) and the decrease in sum-of-squares (value), which is used to obtain the SNP importance scores at the reduce phase; (3) Sample pair identifier (key) and its proximity (value)(`http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#prox`), which is used to produce the final proximity matrix extracted from RF. All these outputs from mappers are sorted, shuffled, and copied to reducers by Hadoop. An illustration of this initial process is given in Figure 6.1. The Hadoop job initially distributes the data set to each map task using a *DistributedCache* mechanism, which copies the read-only

files on to the slave nodes before launching a job. In our implementation, each map task loads the full version of the data set, which is then bootstrapped and used to learn each tree. In this illustration, `Ntree`=3, and each one of the three mappers builds one tree. When the number of required trees (`Ntree`) is larger than the total number of mappers `nmap`, each mapper builds $\frac{\texttt{Ntree}}{Nm}[+1]$ trees.

The output from all the mapper functions, consisting of key-value pairs, is then sorted, shuffled and copied to the *reduce* tasks, which receive their input in the form of $[k_j, [v_{j1}, v_{j2}, \ldots,]]$ pairs, where the first element can be SNP or sample identifier and the second element is a list of values associated with that SNP or sample. The reduce tasks run a user-defined reducer function, and generate an output again in the form of key-value pairs to be saved on file. The reduce tasks compute information gain importance score by summing up all the $\phi(j)$ evaluations obtained by the individual trees in the map phase. Again, the computations are equally distributed across reducers. For instance, in the bottom part of Figure 6.1, each reducer generates a partial list of key-value pairs containing SNP id and its information gain importance score. These lists are eventually saved on file and eventually they are joined to obtain the final output.

This parallel RF regression algorithm has been implemented in Java. It can be run in standalone, pseudo-distributed, fully distributed and commercial cloud service platform. The current version supports biallelic markers which use additive coding. The software gives users the options to calculate OOB, variable importance score, proximity matrix while building the standard or distance-based RF. For multivariate phenotypes, two options are available for modeling the covariance matrix (Euclidean distance and Mahalanobis distance).

6.3 Motivating application and data set

This work is originally motivated by experimental data produced by the Alzheimer's Disease Neuroimaging Initiative (ADNI). Alzheimer's Disease (AD) is a moderate to highly heritable condition, and a growing list of genetic variants have been associated with greater susceptibility to develop early- and late-onset Alzheimer's disease (AD) [Braskie *et al.*, 2011]. We have obtained genotypes for 464 unrelated subjects comprising 99 Alzheimer's diseases (AD), 154 elderly controls

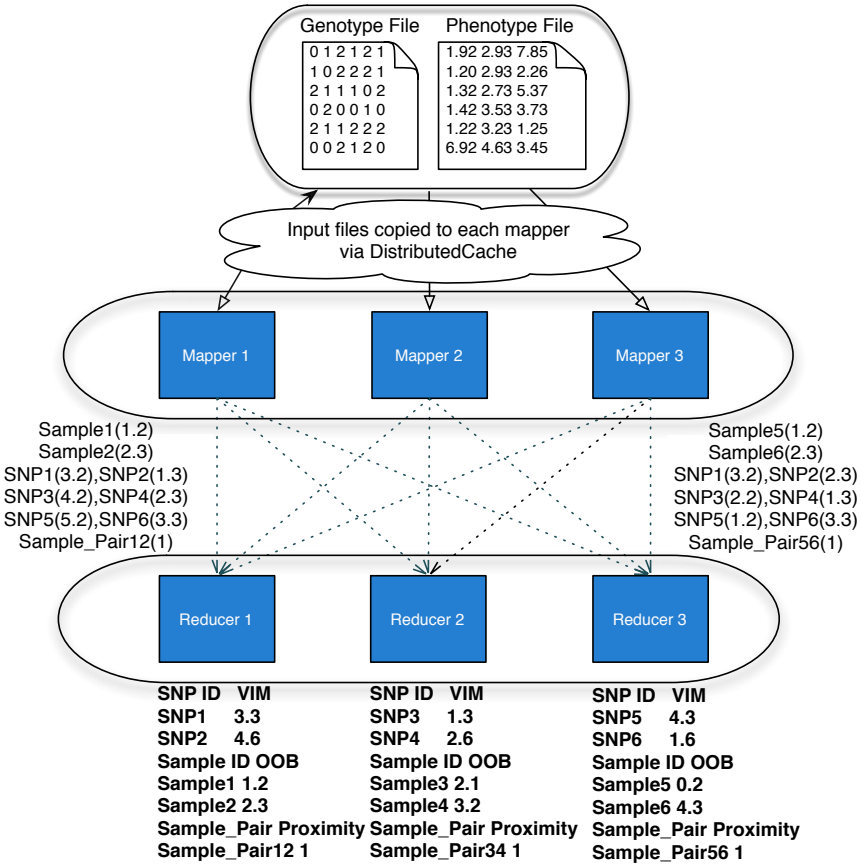

Figure 6.1: An illustration of the RF algorithm implemented according to the MapReduce model. In this example there are 6 SNPs observed on 6 samples, and the analysis is carried out using 3 mappers and 3 reducers. The RF parameters here are set to `Ntree`=3 and `Mtry`=3.

(CN) and 211 mild cognitive impairments (MCI). Genotyping was performed using the Human610-Quad Bead-Chip, which includes 620,901 SNPs [Saykin *et al.*, 2010]. Subjects are unrelated, and all of European ancestry, and passed screening for evidence of population stratification using the procedure described in Stein *et al.* [2010b]. For this study, we include only autosomal SNPs, and additionally exclude SNPs with a genotyping rate<95%, a Hardy-Weinberg equilibrium p-value<5×10^{7}, and a minor allele frequency<0.1. Since our RF implementation does not allow for missing SNP minor allele counts, missing genotypes were imputed as described by Vounou *et al.* [2010].

For each subject in this study, longitudinal brain scans at 6, 12 and 24 months

after the initial screening were available. Our multivariate quantitative trait provides a measure of structural change observed in the brain relative to baseline over the three time points. More specifically, each individual phenotype vector $\boldsymbol{y}_i$ consists of $148,023$ slope coefficients, one for each voxel, quantifying the temporal rate of linear brain tissue loss over time, and therefore providing an imaging signature of the disease. A more detailed description of the preprocessing steps and the procedure used to extract this imaging signatures can be found in Silver *et al.* [2012], where the same data has been analysed in search for biological pathways associated with AD.

6.4 Experiments and results

6.4.1 Simulations

6.4.1.1 Performance comparisons.

Monte Carlo simulation studies are designed to test the performance of our RF algorithm. In order to reduce the computational burden incurred in extensive and repeated simulations, we randomly extract 100 voxels out of the $148,023$ available, and $1,000$ randomly selected SNPs from $434,271$ available. First, we estimate the phenotype sample covariance matrix $\hat{\boldsymbol{V}}$ using all the available samples. Each simulated dataset consists of 464 samples represented by a genotype vector $\boldsymbol{x}_i$ consisting of $1,000$ SNPs. The use of real data is important as it preserves the original patterns of linkage disequilibrium and allele frequencies observed in the imaging phenotypes. Genetic effects are induced using an additive model involving 5 randomly selected causative SNPs with MAF 0.2. Each individual phenotype vector $\boldsymbol{y}_i$ is generated by randomly drawing from a multivariate normal distribution with covariance matrix $\hat{\boldsymbol{V}}$ and such that the genetic effects explain approximately 8% of phenotypic variability. Similar simulation procedures are described in Vounou *et al.* [2010].

We first report on simulation results meant to test and validate our RF implementation. Firstly, we report on the average OOB error obtained by our proposed *PaRFR* (Parallel Random Forest Regression) and a publicly available (sequential) implementation in R package *randomForest*. Since the R implementation

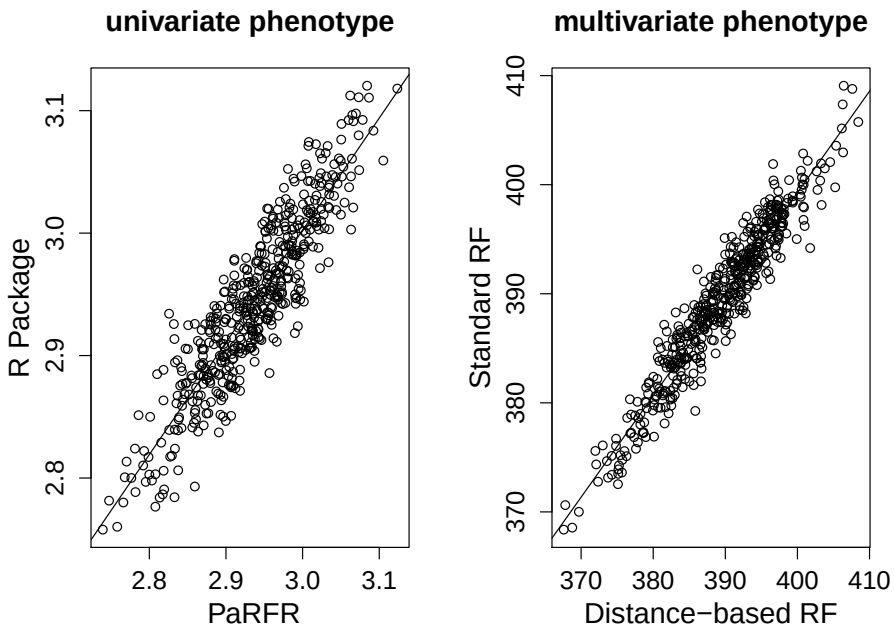

Figure 6.2: Left: OOB error comparison with the *randomForest* implementation; Right: OOB comparison between the two multivariate node splitting criteria. In each case, we use 500 simulated datasets.

only handles univariate responses, for this study we take the average simulated phenotype vector, $\bar{\boldsymbol{y}}$, as response. As can be seen in Figure 6.2 (left), the linear correlation coefficient between the two OOB errors is 0.91603 (p-value<0.001). Some discrepancy is expected due to the Monte Carlo error made during the tree building process. Secondly, we compare the two different node splitting criteria introduced in Section 6.2.2—the standard calculation on the sum of squares with diagonal covariance matrices, and the distance-based calculation using a pre-computed Euclidean distance matrix. Figure 6.2 (right) shows that the two criteria produce highly correlated OOB errors, with a correlation coefficient of 0.93055 (pvalue<0.001), with some difference in performance again explained by the randomness involved during the tree building process. In both cases, we use 500 simulated data sets.

We also assessed statistical power to detect the causal variants as a function of sample size. In Figure 6.3 we show ROC curves obtained from 500 simulated

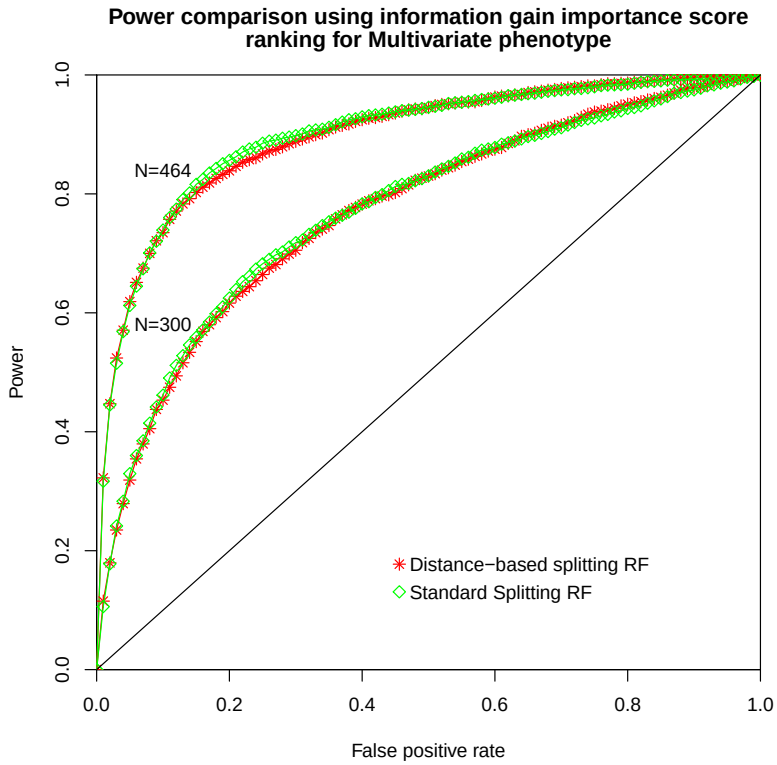

Figure 6.3: Power comparison between distance-based splitting RF and standard node splitting RF when sample size is 300 and 464, each dot in the figure is the average power over 500 simulations.

data sets, for two samples sizes ($N = 300$ and $N = 464$) and the two different node splitting rules that assume diagonal covariance. The power is defined as the number of causal SNPs detected by the algorithm and is reported for all false positives rates (FPR) along the ROC curve. As expected, for each FPR, power increases with sample size and both methods have the same power. These results confirm that the proposed distance-based criterion performs equally to the standard node splitting criterion when using the Euclidean distance.

6.4.1.2 Running time and scalability.

To assess the speedups that can be obtained by our implementation using the distance-based splitting function, we simulate a large dataset containing $1,000$

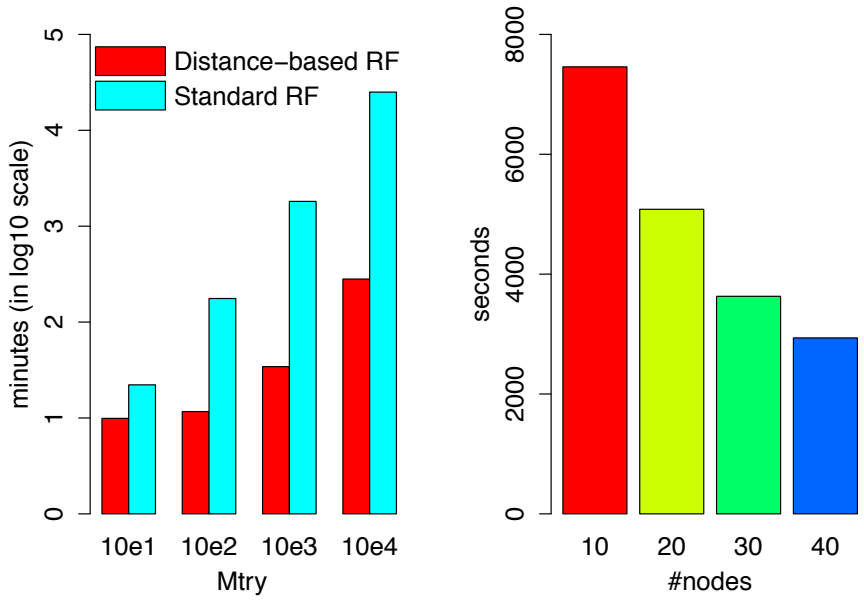

Figure 6.4: Left:runing time comparison using two different RF implementations for different `Mtry`; Right: the scalability test of Distance-based RF in local cluster.

independent samples, 1 million SNPs and a 10,000-dimensional phenotype. This analysis was run on a private cluster with 20 nodes. Each node has 24GB memory and 16 processors with Intel(R) Xeon(R) CPU 2.27GHz. We configured each *map* and *reduce* task to have 800MB memory, and the whole cluster capacity to run 400 *map* tasks and 100 *reduce* task. We compare the running time for different values of the `Mtry` parameter, ranging from 10 to 10,000. Figure 6.4 (left) shows the running time of the two methods on a log scale. When `Mtry`=10, the distance-based RF is only 2 times faster than the Standard RF because of the initial time required to launch the cluster and load the data. As the value of `Mtry` increases, we observe at least a 10-fold speedup, indicating that significant running time savings can be obtained by using distance-based RF for high-dimensional phenotypes.

To test the scalability of our implementation as the number of available nodes increases, we analyze the same size dataset with parameters `Mtry`=1,000 and `Ntree`=2,400. This analysis is run using our local cluster with 10, 20, 30 and 40 nodes, respectively. Figure 6.4 (right) shows that the running time decreases as more nodes are added and that, as expected, rough linear speedup is observed as

the number of nodes increases. For instance, doubling the number of nodes from 20 to 40 yields a 40% reduction in running time.

6.4.2 GWAS

For this study, the RF is applied to the analysis of 99 AD samples and 154 CN samples. MCI subjects are excluded from this analysis so as to detect AD-linked genes only. The analysis was run on our local cluster. Different `Mtry` and `Ntree` parameter values were initially tested to determine how sensitive the results were to changes in these values, and identify optimal values required to stabilise the error rate, and thus obtain a reliable and stable SNP ranking. In particular, 3 different `Mtry` values were used $(144,757; 72,379; 289,514$, each of the value represent the 1/3,1/6 and 2/3 of total SNP numbers), and a range of values from 10 to $50,000$ for `Ntree` [Goldstein *et al.*, 2010]. After initial testing, we settled for a total of $50,000$ trees (using `Mtry` $= 72,379$) to report the results, because Figure 6.5 shows that the agreement (Jaccard coefficient) of the top 5,000 SNP is relatively stable after building 40,000 trees. Figure 6.5 is obtained by first calculating the Jaccard coefficient of the top 5,000 SNPs ranked by *PaRFR* every time an additional 500 trees is added, and then the fitted smooth line is plotted. The horizontal line (corresponding to Jaccard coefficient= 0.88) shows that the fitted curve is almost horizontal after 40,000 trees. The total running time was 60 hours on a 20-node cluster. Since the process of inferring an individual tree took approximately 20 minutes, we estimate that a sequential implementation using the same machines would approximately take 700 days. Distance-based RF is at least 585 (148,023/253, because the evaluation of each $SS(j)$ term has a cost complexity of $O(N(j)^2)$ instead of $N(j) \times Q$, where here N(j) here is 253 and Q is 148,023) times faster than Standard RF.

In Table 6.1 we report the 10 top-ranked SNPs, using the information gain based importance score, and the corresponding mapped genes. Several well-validated genes that have been shown to be linked to AD are included in this list, including APOE, TOMM40, PVRL2, APOC1 and PICALM [Braskie *et al.*, 2011; Saykin *et al.*, 2010; Silver *et al.*, 2012]. To further validate the importance of the SNP rankings produced by *PaRFR*, we carried out a further non-parametric

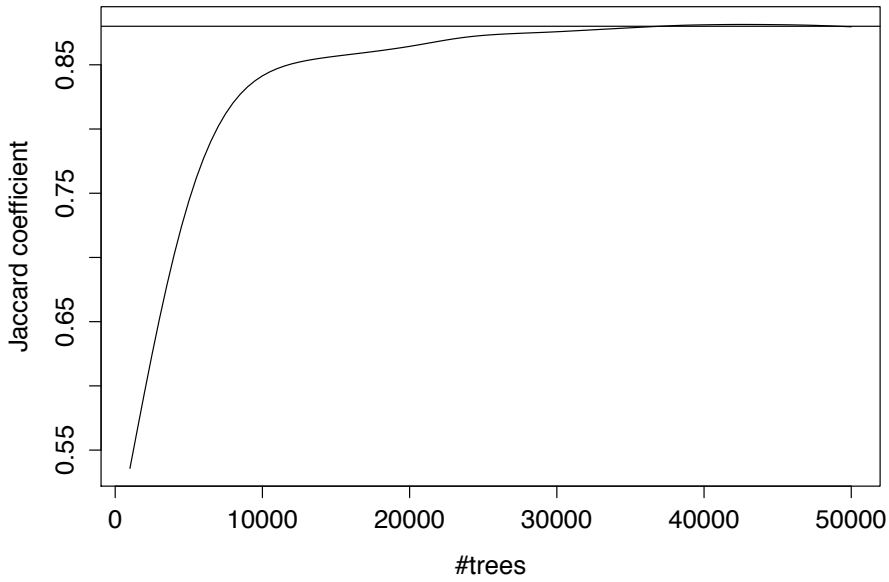

Figure 6.5: The Jaccard coefficient plot for the agreement of top 5,000 ranked SNPs with more trees added, the horizontal line is Jaccard coefficient = 0.88.

analysis using permutation testing. We extracted 47 AD-linked genes provided by Saykin *et al.* [2010], and assessed whether these genes rank high in our list, and whether such high rankings cannot be explained by chance only. For each AD-linked gene g, we first obtained an observed score, $\hat{s}_g$, by averaging the ranks of all SNPs that map onto that gene. Operating under the null hypothesis that the AD-genes are randomly ranked, we shuffled the SNP rankings $10,000$ times, and computed an empirical null distribution for each s_g. A p-value was then computed using this null distribution, the null distribution of top 2 genes is shown in Figure 6.6. Table 6.2 shows the 6 statistically significance genes (out of the 47 known AD-linked genes) and their scores, using a nominal significance level of 0.05 after multiple test correction by FDR.

Besides variable importance ranking, *PaRFR* also produces a matrix of pairwise genetic proximities from which we can gain further insights. The estimated proximity matrix is a $N \times N$ symmetric matrix where each element p_{ij} indicates the genetic similarity between any pair of samples, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Each p_{ij} value is obtained as the fraction of trees where sample $\boldsymbol{x}_i$ and sample $\boldsymbol{x}_j$ appear in the same leaf. A measure of genetic distance between any two samples is then simply

Table 6.1: ADNI: top 10 genes and corresponding SNPs, known AD-linked genes are in bold font.

Top 10 ranking genes	Corresponding SNP
TOMM40/APOE/APOC1	APOE4
PICALM	rs7938033
PVRL2	rs2075650
NTNG2	rs7862808
NTM	rs12293070
SLC12A1	rs6493311/rs1531916
MEF2D	rs1750304
CD109	rs9352023
UNC5B	rs10762435
DPYD	rs496179

obtained by taking $d_{ij} = 1 - p_{ij}$. It seems useful to compare the estimated genetic and phenotypic distances, as we would expect to identify the two large clusters of samples, AD patients and healthy controls. Figure 6.7 shows two-dimensional multidimensional scaling plots: the scatter plot (a) provides a 2D representation of the AD and CN samples obtained from the pair-wise genetic distances estimated by *PaRFR*, from which a clear separation between AD and CN can be noted; the scatter plot (b) provides a 2D representation of the AD and CN samples obtained from the pair-wise Euclidean distances of the multivariate neuroimaging phenotypes (148, 023 voxels), also showing a clear separation between the two phenotypically distinct groups. Most importantly, Figure 6.8 shows that genetic and phenotypic distances are almost linearly associated (the observed correlation coefficient for AD/AD and CN/CN plots are 0.8248, 0.8245, for pooled AD and CN groups is 0.7989). This result was further validated by performing a Mantel test of no association between the paired genetic and phenotypic distance matrices, which was significance at the 0.01 level in all three plots, using 10, 000 permutations. From the AD/AD plot there is a clear spread between AD/AD sample pair due to the spread in imaging signatures of AD group, and CN/CN plot shows more tight and smooth clustering. When looking the AD/AD plot and the CN/CN plot together, we see a more divergent phenotypic Euclidean distance in AD/AD plot than in CN/CN plot given the same genetic Euclidean distance (e.g. 0.3). This suggests that the SNPs chosen to build the RF (and

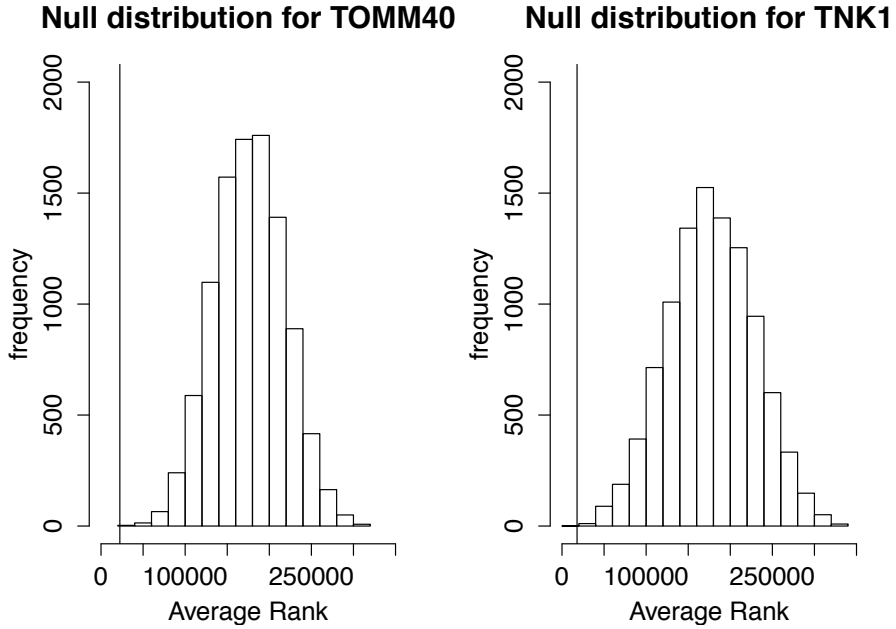

Figure 6.6: The Null distribution obtained by permuting 10,000 times the rank of SNPs harboured by the top 2 genes.

thus involved in defining the genetic Euclidean distance) are able to account for the larger spread of AD samples in Figure 6.7 (right). When the AD/CN plot is compared to the CN/CN and AD/AD plots, we see that at the same phenotypic Euclidean distance (e.g. 1,000), the genetic Euclidean distance is generally greater. This means that the SNPs involved in the RF exhibit a greater difference in their alleles for CN and AD samples.

Table 6.2: ADNI: AD-linked genes with significant ranks in the proposed Null Hypothesis.

AD-linked Genes	P-value	Average rank
TOMM40	<0.0001	22,371.83
TNK1	0.0022	17,890.00
NXPH1	0.0058	138,675.53
APOE	0.0088	32,035.50
ACE	0.0211	97,075.08
MYH13	0.0403	127,462.04

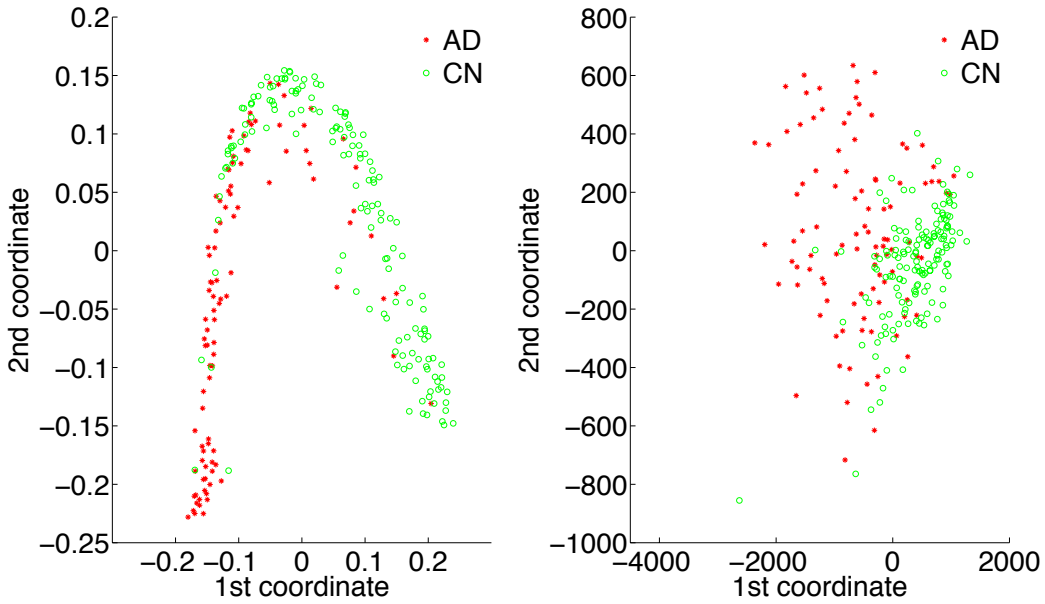

Figure 6.7: Two-dimensional multidimensional scaling plots: (a) 2D representation of the AD and CN samples obtained from the pair-wise genetic distances estimated by *PaRFR*; (b) 2D representation of the AD and CN samples obtained from the pair-wise Euclidean distances of the multivariate neuroimaging phenotypes (148, 023 voxels). Sample clustering can be seen in both plots.

6.4.3 Hypothesis testing on the quantitative phenotypes and genetic patterns

3 dimensional (3D) multidimensional scaling (MDS) plot is a set of related statistical techniques often used in information visualization for exploring similarities or dissimilarities in data. MDS is a special case of ordination[1]. The spread of the AD samples from 3D MDS plot of 148,023 voxels of 253 samples (Figure 6.9) motivates the following analysis.

We observe that the CN samples are tightly clustered while the AD samples are quite spread out in Figure 6.9. We believe that the spread of the AD samples in the figure reflects the different degrees of severity (i.e., different types and levels of cognitive impairment) of Alzheimer's disease in the AD samples. In particular, we make the hypothesis below.

Hypothesis A: The further away an AD sample is from the CN cluster in

[1] http://en.wikipedia.org/wiki/Multidimensional_scaling

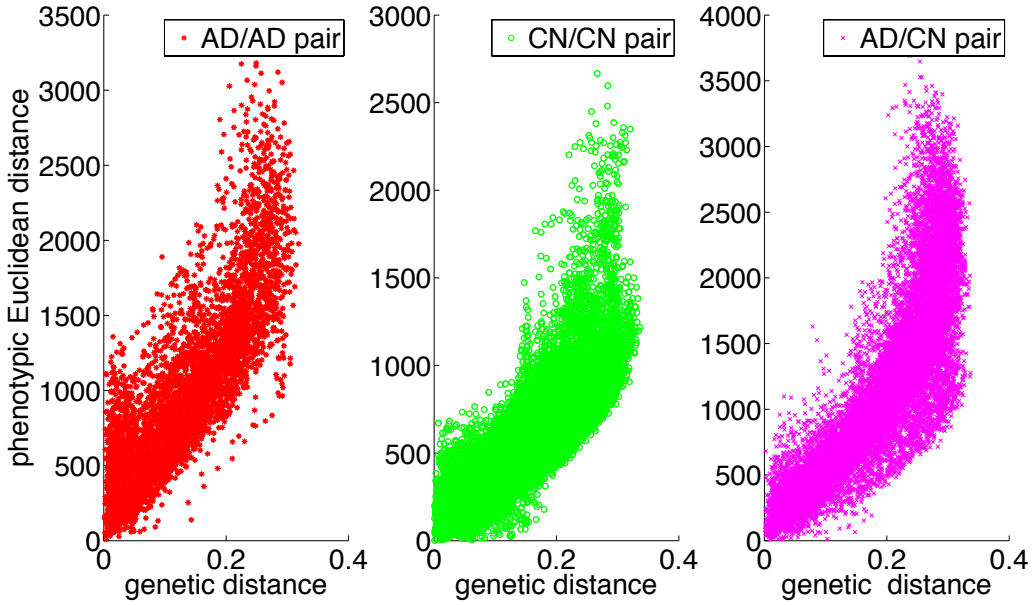

Figure 6.8: The three plots are the scatter plot of genetic Euclidean distance derived from Figure 6.7 left and phenotypic Euclidean distance derived from Figure 6.7 right for three types of sample pair. 4 outliers from CN groups are excluded.

Figure 6.9, the greater the severity of the Alzheimer's disease of the AD sample.

We believe there is a genetic explanation to the severity of the Alzheimer's disease of a patient and, thus, to the spread of the AD samples away from the CN cluster. In particular, we believe that Alzheimer's disease is the result of cumulative effect of the dysfunction of multiple genes due to mutations and that different combinations of mutations correspond to the different degrees of severity. That is, we make the following hypothesis.

Hypothesis B: The more number of certain combinations of mutations are observed in a person, the more severely he suffers from Alzheimer's disease.

We are not able to verify Hypothesis A as the clinical notes of the AD patients are not available to us. So we assume that Hypothesis A is true, and attempt to verify Hypothesis B with respect to it (i.e., use the spread distance of an AD sample from the CN cluster as an indication of the severity of the disease in that AD sample). Now, in order to test Hypothesis B, we need to (i) identify those combinations of mutations that are associated with Alzheimer's disease, and (ii)

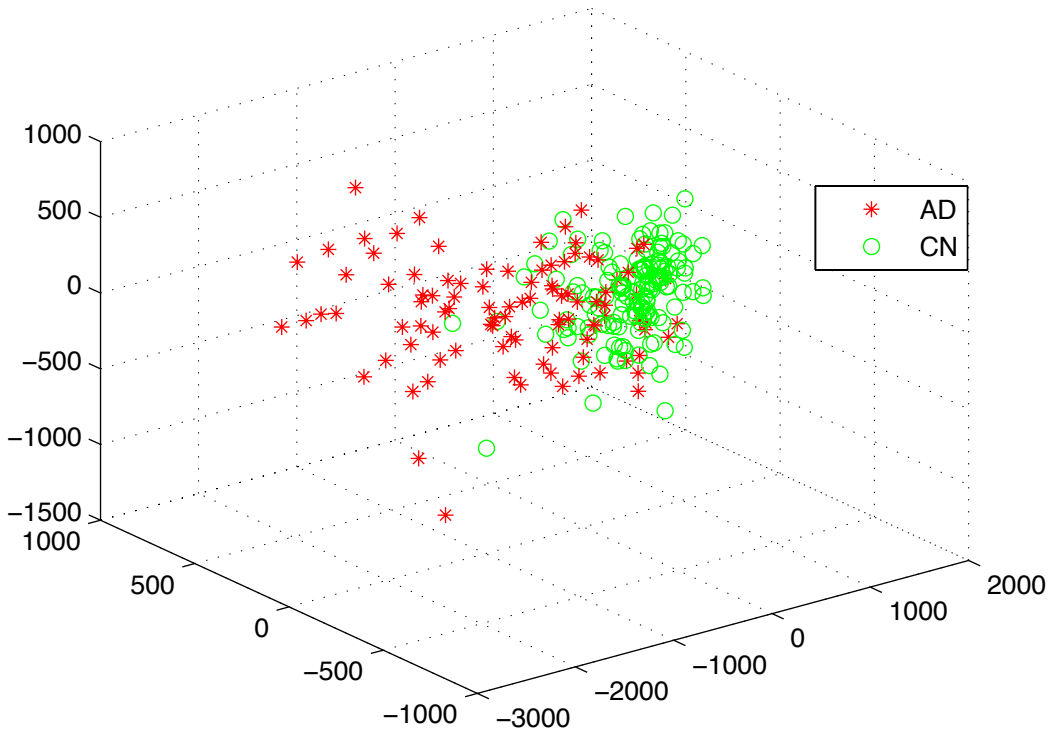

Figure 6.9: The 3D MDS plot of the 148,023 voxels from 253 ADNI samples. This plot is used to visualize the relative distance between different samples from high-dimensional space to 3 dimensions.

demonstrate the correlation of the number of these combinations of mutations with the spread (i.e., distance) away from the CN cluster.

In accordance to Hypothesis A, the samples furthest away from the CN samples are expected to be have the most severe form of Alzheimer's disease. In accordance to Hypothesis B, these samples should contain the most number of the combinations of mutations that correspond to the different types and severity of cognitive impairment in Alzheimer patients. Thus we can identify those combinations of mutations that are associated with Alzheimer's disease as those that are frequently observed in AD samples far away from the CN cluster and are infrequently observed in the CN cluster.

In order to identify samples that are far away from the CN cluster in an unbiased manner, we apply an unsupervised hierarchical clustering technique. This clustering procedure segments the AD and CN samples into four clusters

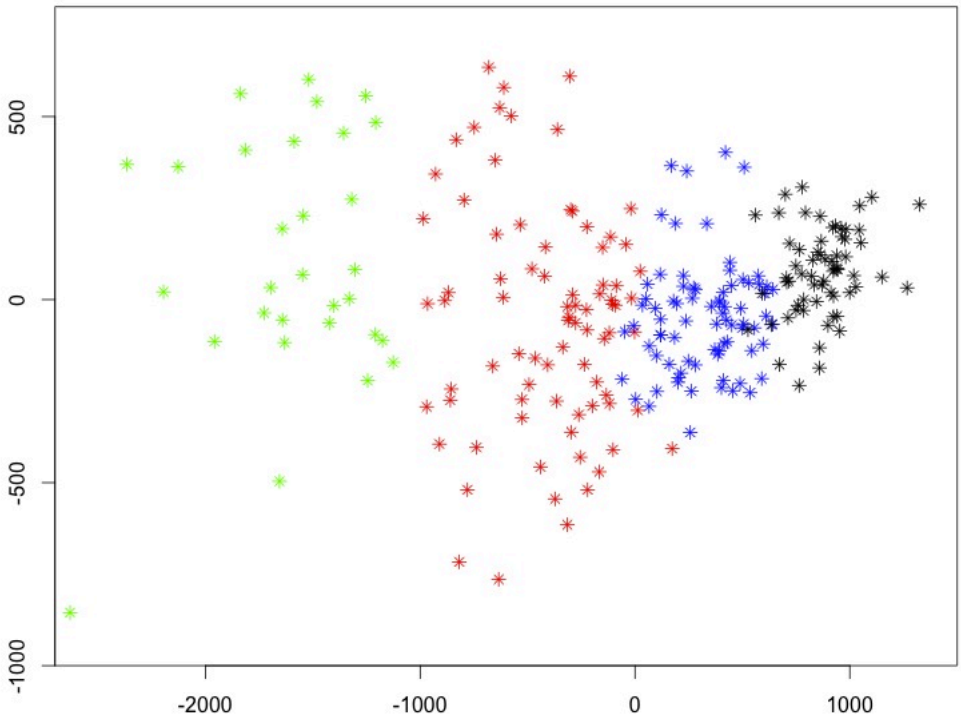

Figure 6.10: The 2D MDS plot of the hierarchical clustering of 253 ADNI samples. The four clusters, from right to left, are referred to in the main text as C1, C2, C3 and C4.

C1, C2, C3, and C4 as shown in Figure 6.10. The proportion of AD samples in these clusters are 11%, 16%, 63%, and 93% respectively. From Figure 6.10, it is also clear that C1 corresponds to the CN cluster and C4 corresponds to AD samples furthest away from the CN cluster.

Thus, to identify a combination of mutations (i.e., a mutation pattern) that is associated with the type and severity of cognitive impairment in Alzheimer's disease, we require a pattern to be "frequent" in C4 and "infrequent" in C1, C2 and C3. The frequency threshold of mutation patterns is set to 0.05 in our analysis. All homozygous two major allele genotypes are removed in the data because disease-causing mutation patterns are expected to occur in the homozygous two minor allele genotypes and heterozygous genotypes. We define "frequent" based on the null hypothesis that: The proportion of the patterns in "C4" is no differ-

ent from the proportion in the entire data set. If the null hypothesis is rejected and the pattern is above a specified frequency threshold (0.05), this pattern is considered as a "mutation pattern" since it exists mostly in C4, which has the majority of the AD samples. The procedure described above produces 17,000 mutation patterns. We considered these mutations to be associated with the type and severity of Alzheimer's disease. Then, if Hypothesis B is true, we should see more number of these patterns in AD samples that have more severe degree of Alzheimer's disease.

To define the severity of the sample, in accordance to Hypothesis A, we consider two different notions of distance. The first notion defines severity as the Euclidean distance between a sample and the centroid of C1. This is motivated by the radiation pattern from C1 to C4, which suggests the centroid of C1 as the representative radiation center. The second distance notion defines severity as the projected distance—on the axis of disease progression—of a sample to the centroid of C1. The axis of disease progression is defined as the line connecting the centroids of C1 (which contains mostly CN samples) and C4 (which contains mostly AD samples).

Figure 6.11 is obtained by plotting the correlation between the above defined mutation pattern counts and two types of distance to the C1 centroid. Both correlations are positive and statistically significant. This provides good support for Hypothesis B: for an AD patient, the more mutation patterns he/she carries, the more severe he/she has the disease. From the bottom left part of the two plots, the CN samples are tightly clustered and have less number of mutation patterns, this is also in consistent with our expectations. The plot shows obvious evidence for the hypothesis we propose and can be used to give us more insights on Alzheimer's disease.

6.5 Discussion

In this chapter, we have described a parallel RF algorithm for multivariate regression on Hadoop, which is tolerant to node failure and can be easily scaled up to very large clusters. Two other state-of-the-art Hadoop-based implementations of the RF are COMET [Basilico *et al.*, 2011] and *Mahout* (`http://mahout.apache.`

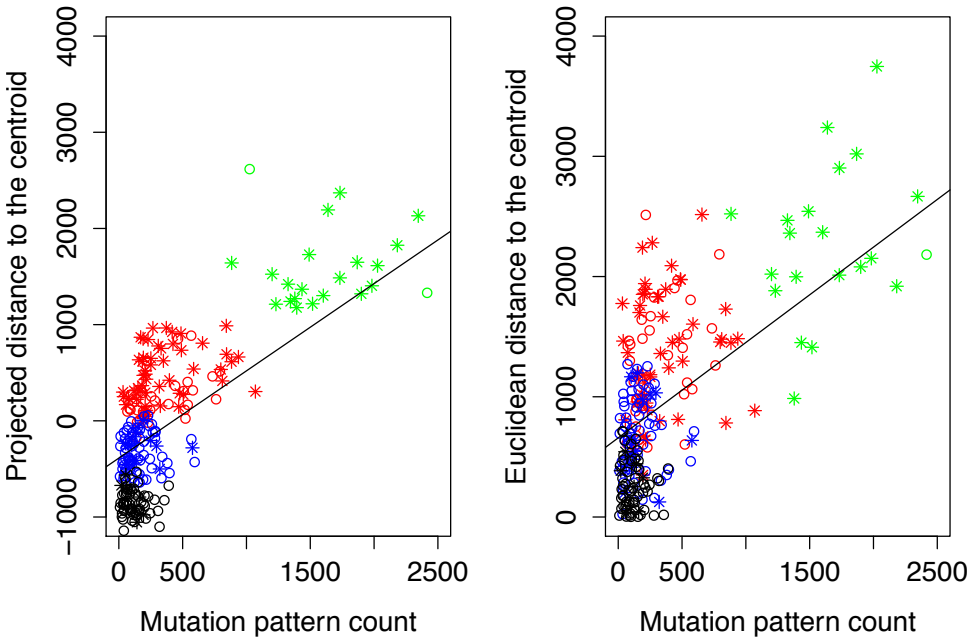

Figure 6.11: The correlation between the number mutation pattern and the distance to the healthy centroid (centroid of C1). Red stars are the AD samples and green circles are the CN samples. The fitted line are plotted because the beta coefficient of the line is statistically significant at p-value 0.05. The four clusters, from bottom left to top right, are referred to in the main text as C1, C2, C3 and C4. The star shape indicate a AD sample and circle shape indicate a CN sample.

org/). COMET is particularly suited for large data sets as it splits the input files into blocks of fixed size (e.g. 256MB). The RF is built using a bagging method *IVoting* within each block of data, and the output of different mappers results in a mega-ensemble RF which is sorted, shuffled and copied to reducers. However, this RF implementation only supports univariate classification and is not open source yet.

Mahout is an open-source machine learning library for large-scale data processing using Hadoop. Its latest version includes RF classification and regression for univariate. The parallelisation of the algorithm is done in two different approaches : the first each *map* task builds trees using partial data splits generated by Hadoop, and the second loads the full data into memory to build trees. However, *Mahout* is still evolving, and many features of RF, like variable importance and proximity matrix calculation, are not supported yet. Also, both Mahout and

COMET do not support high-dimensional quantitative response and variable importance score calculation.

Other variations of RF have been developed or supported for high performance computing (HPC). The SPRINT project [Hill *et al.*, 2008] supports Message Passing Interface (MPI) and provides a system for parallel computations by transparently distributing the computation to different processors among different nodes. Recently, the *randomForest* R package has also been included into this framework, although only univariate response is implemented. A different project also using MPI is Random Jungle, which re-implements RF efficiently and adds several features like variable backward elimination and conditional variable importance [Schwarz *et al.*, 2010]. It is not applicable to high-dimensional responses. MPI-based implementations are not able to scale to data larger than the total memory of the HPC system, and they have difficulty handling node failures. Highly parallel GPU-based implementations of RF have been proposed, such as CudaRF [Grahn *et al.*, 2011] and others [Sharp, 2008]. When each dataset is loaded into GPU memory, each thread builds a tree. However, they are difficult to scale to large datasets because of more restrictive GPU memory limitation. A table summarising the different parallel implementations and their main features can be found in the following table 6.3.

In future work, we intend to investigate the difference in power and in speed when building RF using partial data (*splits*) generated by Hadoop and full data which is used in the current implementation. We also plan to build a multi-pass *MapReduce* implementation which can monitor changes in OOB error as the forest grows. This feature will automatise parameter tuning and thus will save computational cost. A further generalisation of this approach may consist in replacing the pair-wise Euclidean distances with more general parameter-free distances that better capture the phenotypic variability observed in the populations. For instance, the imaging phenotypes may be brain connectivity networks, i.e. collections of nodes (vertices) and links (edges) between pairs of nodes. Such brain connectivity networks describe the set of connections in the neural system, or connectome, in which nodes could be neurons or cortical areas, and edges could be axons or fibre tracts. The edges could refer to the structural connectivity of a neural network [Rubinov and Sporns, 2010]. An additional benefit of this

distance-based formulation can be obtained by replacing the pair-wise Euclidean distances with more general parameter-free distances that capture the proximity between any two subject at the phenotypic level.

Although we are mostly motivated by applications in neuroimaging genetics, the algorithm we propose has wider scope and can be used for any QTL mapping study involving a very large number of genetic markers and high-dimensional responses. For instance, there is recent interest in detecting genetic variability associated with facial shape, which can be quantified using 3D phenotypes obtained from statistical shape analysis [Liu *et al.*, 2012]. Other multivariate traits arise, for instance, in eQTL mapping studies, where the phenotypes consist of gene expression abundances, or in longitudinal studies involving time series or repeated measurements [Xiao and Segal, 2009].

Table 6.3: Summary of the variables of different parallel RF

	SPRINT	Random Jungle	CudaRF	Mahout	COMET	PaRFR
Variable importance	√	√	×	×	×	√
Variable selection	√	×	×	×	×	×
Sample proximities	√	√	×	×	×	√
Classification	√	√	√	√	√	√
Univariate regression	√	√	×	√	×	√
Multivariate regression	×	×	×	×	×	√
Build trees using full/partial data	full	full	full	both	partial	full
Scale to GWAS data	×	×	×	√	√	√
Open-source	√	√	×	√	×	√
Programming language	C	C++	C	Java	Java	Java

Chapter 7

FastTagger: An efficient algorithm for genome-wide tag SNP selection using multi-marker linkage disequilibrium and its application in SNP imputation

7.1 Introduction

There are millions of SNPs in the human genome. Such an enormous number of SNPs presents a challenging problem for genome-wide association study. It has been observed that adjacent SNPs are often highly correlated. To reduce genotyping cost, many algorithms have been developed to select a smallest set of SNPs such that all the other SNPs can be inferred from them. The selected SNPs are called tag SNPs. Tag SNPs are useful because they are considered as the proxies of other SNPs when conducting SNP association analysis. The proxy property of Tag SNP can also be used for conducting SNP imputation. However, existing tag SNP selection algorithms are both time and memory consuming. To overcome these two computational challenges, we propose a fast and efficient genome-wide tag SNP selection algorithm (called FastTagger) using multi-marker linkage dis-

equilibrium. The algorithm can work on data with more than 100k SNPs that previous methods cannot handle. We further utilize the rules produced by FastTagger and develop a new tag-based imputation method called RuleImpute. This chapter is structured as two major sections. Section 7.2 and Section 7.3 describe the work on FastTagger and RuleImpute, respectively. Section 7.2.1 gives an overview of the existing tag SNP selection algorithms and their computational challenges. The proposed techniques to resolve these computational difficulties are introduced in Section 7.2.2. In Section 7.2.3, we compare the efficiency and speed with the state-of-the-art methods on benchmark Hapmap data. Section 7.3.1 gives a concise description of the SNP imputation problem. In Section 7.3.2, we describe the rule-based imputation method RuleImpute based on FastTagger and propose five different rule selection strategies. The best strategies are suggested in Section 7.3.3 according to the experimental results on Hapmap data.

7.2 FastTagger: Efficient tag SNP selection

7.2.1 Background

Existing tag SNP selection methods can be classified into two categories: block-based methods [Avi-Itzhak *et al.*, 2003; Gabriel *et al.*, 2002; Johnson *et al.*, 2001; Patil *et al.*, 2001; Sebastiani, 2003; Thompson *et al.*, 2003; Zhang *et al.*, 2004] and genome-wide approaches [Carlson, 2004; Halldorsson, 2004; Halperin, 2005; Liu *et al.*, 2007; Magi *et al.*, 2006; Qin *et al.*, 2006]. Block-based methods rely on a predefined haplotype block structure. The blocks are separated by recombination hot-spots, and there are few recombinations within a block. Thus the haplotypes within a block usually are of low diversity. These methods attempt to select a subset of SNPs that can discriminate all common haplotypes within each block.

In contrast, genome-wide tag SNP selection algorithms do not need to partition the whole chromosome into blocks, and they utilize linkage disequilibrium among nearby SNPs to find tag SNPs. Among the genome-wide approaches, those based on the r^2 linkage disequilibrium statistic have gained increasing popularity recently because r^2 is directly related to statistical power for detecting disease as-

sociations [Pritchard and Przeworski, 2001]. LD-select [Carlson, 2004] is the first algorithm using the r^2 LD statistic to select tag SNPs, and it employs a greedy approach to find tag SNPs. Following it, several other algorithms based on the r^2 statistic have been developed. FESTA [Qin *et al.*, 2006] breaks down large marker sets into disjoint pieces, where exhaustive searches can replace the greedy algorithm, thus leading to smaller tag SNP sets. MultiPop-TagSelect [Howie, 2006] and REAPER [Magi *et al.*, 2006] apply LD-select to multiple populations. LRTag [Liu *et al.*, 2007] uses a Lagrangian relaxation algorithm to find tag SNPs across multiple populations.

All these algorithms use pairwise LD between SNPs. Recent studies have shown that multi-marker LD can help further reduce the number of tag SNPs needed [Bakker *et al.*, 2005; Huang and Chao, 2008; Pe'er, 2006], and several algorithms have been developed to select tag SNPs based on multi-marker r^2 statistics [Hao, 2007; Hao *et al.*, 2007; Wang and Jiang, 2008]. These algorithms find association rules of the form $\{SNP_1, \cdots, SNP_k\} \rightarrow SNP_x$, where $k \leq 3$, $SNP_x \notin \{SNP_1, \cdots, SNP_k\}$ and the r^2 statistic between the left hand side and the right hand side of the rule is not less than a predefined threshold. Their results show that the multi-marker LD model can reduce the number of tag SNPs significantly compared with pairwise algorithms.

However, existing multi-marker based algorithms are both time and memory consuming. Most of the time is spent on calculating multi-marker r^2 statistics. Furthermore, an excess number of multi-marker association rules may be generated when $k \geq 3$, which incurs high memory consumption when using these rules to select tag SNPs. It takes hundreds of hours for the MultiTag algorithm [Hao, 2007; Hao *et al.*, 2007] to finish on chromosomes containing around 30k SNPs. The MMTagger algorithm [Wang and Jiang, 2008] needs several hours to finish, but it consumes more than 1GB memory. MMTagger cannot work on chromosomes with more than 100k SNPs when $k \geq 3$.

7.2.2 Methods

In this section, we first describe how to calculate multi-marker r^2 statistics, and then present the FastTagger algorithm. The FastTagger algorithm consists of two

steps. In the first step, it generates tagging rules, and in the second step, it uses a greedy approach to select tag SNPs using rules generated in the first step.

Multi-marker tagging rules

Most SNPs have only two alleles, so we consider only bi-allelic SNPs. Given a population, the allele with higher frequency in the population is called major allele, and the allele with lower frequency is called minor allele. We use uppercase letters to denote the major alleles of SNPs, and use lowercase letters to denote the minor alleles. SNPs that are far apart from each other usually are not linked. Here we require that the distance between every pair of SNPs in a rule must not exceed a predefined distance threshold max_dist.

Given k SNPs $S = \{SNP_1, SNP_2, \cdots, SNP_k\}$, there are 2^k possible haplotypes over the k loci. To calculate the r^2 statistic of rule $S \rightarrow SNP_x$, we need to divide the 2^k haplotypes into two non-empty groups and map the two groups to the two alleles of SNP_x. MultiTag [Hao, 2007] and MMTagger [Wang and Jiang, 2008] uses different methods to do the mapping.

The one-vs-the-rest model. MultiTag uses this model. There are totally $2^{2^k} - 2$ possible ways to group the 2^k haplotypes into two non-empty groups. MultiTag considers only 2^k ways such that one group contains only one haplotype, and the other group contains all the other haplotypes. It calculates the r^2 statistics for all the 2^k groupings, and then select the one with the highest r^2 statistic.

The co-occurrence model. MMTagger does the mapping based on the co-occurrences of the alleles of the SNPs on the left hand side and the alleles of the SNP on the right hand side. Let H be a haplotype over the SNP set S on the left hand side, A and a be the two alleles of SNP_x on the right hand side, and $f(H)$ be the frequency of H. We use $f(HA)$ to denote the frequency of H and $SNP_x = A$ occurring together, and $f(Ha)$ to denote the frequency of H and $SNP_x = a$ occurring together. If $f(HA) > f(Ha)$, we map haplotype H to allele A of SNP_x, otherwise we map haplotype H to allele a of SNP_x. Let H_A be the set of haplotypes mapped to allele A, and H_a be the set of haplotypes mapped to allele a. We convert the SNP set S to a bi-allelic marker with two "alleles" H_A

and H_a. Then we can calculate the r^2 statistic between S and SNP_x as follows.

$$r^2(S, SNP_x) = \frac{(P(H_A A) - P(H_A)P(A))^2}{P(H_A)P(H_a)P(A)P(a)} \tag{7.1}$$

where $P(H_A)$, $P(H_a)$, $P(A)$, $P(a)$ and $P(H_A A)$ are the relative frequencies of H_A, H_a, A, a and $H_A A$ respectively. We implemented both models in the FastTagger algorithm, and let users choose which model they want to use.

If the r^2 statistic between S and SNP_x is no less than a predefined threshold min_r2, we say that SNP_x can be tagged by S, and $R : S \rightarrow SNP_x$ is a tagging rule. With the increase of the size of S, the haplotypes of S partition the whole dataset into finer and finer groups. In an extreme case, every haplotype of S occurs at most once. In this case, the association between haplotypes of S and alleles of SNP_x becomes unreliable. To prevent over-fitting, we put a constraint on the size of S. The size of S should not exceeds a predefined threshold max_size.

The r^2 statistics can be calculated from phased haplotype data directly. If the SNP data are in the form of unphased genotype data, we can use existing haplotype inference algorithms such as PHASE [Stephens *et al.*, 2001] to convert genotype data into phased haplotype data. We can also estimate k-marker haplotype frequencies directly from genotype data without phasing using algorithms described in, e.g.,Hill [1974, 1975]. The second approach is used in the algorithm LD-select [Carlson, 2004].

Generating tagging rules

To generate all the tagging rules, we need to enumerate all the SNP sets that satisfy the maximum distance constraint and maximum size constraint, and then calculate the r^2 statistics between these SNP sets and their nearby SNPs. The search space can be enormously large when the number of SNPs is large. We use several techniques to reduce the number of rules to be tested.

Merging equivalent SNPs. Given two SNPs SNP_i and SNP_j, if $r^2(SNP_i, SNP_j) = 1$, which means that SNP_i and SNP_j can tag each other perfectly, then we say SNP_i and SNP_j are equivalent. Two equivalent SNPs always have the same r^2 statistics with other SNPs. Thus, the computation cost of the rules involving them can be shared by merging them together. For each group of merged equiv-

alent SNPs, a representative SNP is picked to represent this group. FastTagger generates tagging rules between representative SNPs only. The tagging rules generated in this way are called representative tagging rules. One representative tagging rule can actually represent multiple rules. Therefore, by merging equivalent SNPs, we are not only saving computation cost, but also reducing storage overhead. Note that not every rule represented by a representative tagging rule is valid. Some of them may not satisfy the distance constraint. Equivalent SNPs that are separated by more than max_dist bases cannot appear in the same rule, and merging them together can produce many false rules. To reduce the number of false rules, FastTagger only merges equivalent SNPs that are within a distance of max_dist.

Pruning redundant tagging rules. If a SNP SNP_x can be tagged by a SNP set S, then any rule $S' \rightarrow SNP_x$ such that S' is a proper superset of S is redundant. FastTagger generates only non-redundant tagging rules to reduce running time and memory consumption, and the definition of non-redundant rules is given as follows:

Definition 1 (Non-redundant tagging rule) *Given a rule $S \rightarrow SNP_x$ such that SNP_x can be tagged by S, if there does not exist another rule $S' \rightarrow SNP_x$ such that S' is a proper subset of S and SNP_x can be tagged by S', then $S \rightarrow SNP_x$ is called a non-redundant tagging rule.*

To prune redundant rules, before calculating the r^2 statistic between S and SNP_x, FastTagger checks whether there exists a subset S' of S such that SNP_x can be tagged by S'. FastTagger uses a depth-first strategy to enumerate SNP sets. This search strategy is adopted from a frequent generator mining algorithm [Liu *et al.*, 2008], and it ensures that all the tagging rules whose left hand side is a subset of S are generated before S is processed. There can be many tagging rules generated. To speed-up the check operation, FastTagger divides the generated tagging rules into groups based on their a right hand side SNP; that is, rules with the same right hand side SNP are in the same group. FastTagger then uses a hash map to index the rules in the same group, and the hashing key is the left hand side of the rules. To check whether $S \rightarrow SNP_x$ is redundant, FastTagger searches the hash map of SNP_x for the subsets of S. If there is a subset of S in

the hash map of SNP_x, the rule is redundant; otherwise, the r^2 statistic of the rule is calculated.

Skipping rules. Even though merging equivalent SNPs and removing redundant tagging rules can reduce the number of tagging rules significantly, it is still possible that a large number of tagging rules are generated in the first step, which incurs high memory consumption in the second step. FastTagger uses heuristics to further reduce the number of tagging rules generated: if a SNP SNP_x occurs at the right hand side of tagging rules enough number of times, then SNP_x will not be considered as right hand side candidate in future rule generation. The rationale behind this heuristics is that if a SNP can be tagged by many other SNPs, then during the tag SNP selection process, the SNP has a high probability to be covered by selected tag SNPs.

Selecting tag SNPs using a greedy approach

Finding the smallest set of tag SNPs is computationally expensive. FastTagger uses a greedy approach similar to the one proposed in [Carlson, 2004; Hao, 2007] to find a near optimal set of tag SNPs. Let C be the set of candidate tag SNPs, T be the set of tag SNPs selected, and V be the set of SNPs not being covered. A SNP is covered if either it is a tag SNP or it can be tagged by some SNP set S such that $S \subseteq T$. Initially, C and V contain all the SNPs, and T is empty. FastTagger first identifies those SNPs that do not appear at the right hand side of any tagging rules, and these SNPs must be selected as tag SNPs. FastTagger puts them into T and remove them from C. These SNPs are also removed from V. For the remaining SNPs in V, if they can be tagged by some SNP set S such that $S \subseteq T$, then they are removed from V too. Next, for each SNP $SNP_i \in C$, FastTagger finds the set of SNPs in V that are covered by SNP_i. A SNP SNP_j in V is covered by SNP_i if SNP_j is not tagged by any subsets of T and there exists a subset S of T such that SNP_j is tagged by $S \cup \{SNP_i\}$. FastTagger then picks a SNP from C that covers the largest number of SNPs in V as a tag SNP. This newly picked tag SNP is put into T and removed from C. All the SNPs that are covered by it including itself are removed from V. This process is repeated until V is empty, that is, all the SNPs have been covered. In each iteration, in order to find the set of SNPs covered by every candidate tag SNP in

C, FastTagger needs to keep the tagging rules in memory. However, the number of rules generated can be very large. It is possible that the total size of tagging rules is too large to fit into the main memory. To solve this problem, we can break the whole chromosome into several chunks such that the rules over every chunk can fit into the main memory. We then select tag SNPs within each chunk.

When selecting tag SNPs within each chunk, only those tagging rules whose SNPs all fall into this chunk are used. To also utilize the rules across chunks, we allow two adjacent chunks to have certain overlap. The length of the overlap is determined by the *max_dist* threshold. The SNPs in one chunk that are within *max_dist* bases away from the first SNP of the next chunk are included in the next chunk since they can tag or be tagged by SNPs in the next chunk. FastTagger finds tag SNPs from each chunk from left to right. The tag SNPs selected in the current chunk that also belong to the next chunk will be passed on to the next chunk as tag SNPs. Note that if the distance between two adjacent SNPs is larger than *max_dist*, then these two SNPs are used as a breakpoint even if there is enough memory. The reason being that if the distance between two adjacent SNPs is larger than *max_dist*, then the two SNPs cannot tag each other or each other's neighbors.

Using the above method, FastTagger can work on chromosomes containing more than 100k SNPs with as less as 50MB memory, while existing algorithm consumes more than 1GB memory even on chromosomes containing around 30k SNPs.

7.2.3 Results and discussion

In this section, we study the performance of FastTagger. We conducted the experiments on a PC with 2.33Ghz Intel(R) Core(TM) Duo CPU and 3.25GB memory running Fedora 7. All codes were compiled using g++. We obtained the datasets from HapMap release 21 (ftp://ftp.ncbi.nlm.nih.gov/hapmap/phasing/2006-07_phaseII/phased/) and project ENCODE (ftp://ftp.ncbi.nlm.nih.gov/hapmap/phasing/2005-03_phaseI/ENCODE/). There are 4 populations and 10 regions in the ENCODE project. Here, we report the overall results on the ten regions for each population. From HapMap release 21, we selected 6 chromosomes: chr1,

chr2, chr3, chr19, chr21 and chr22, and used the Han Chinese plus Japanese population. Table 7.1 shows the number of SNPs with $MAF \geq 5\%$ on the datasets. In all the experiment, we set max_dist to 100k, and select only those SNPs with $MAF \geq 5\%$.

Table 7.1: The "#Rep SNPs" column is the number of representative SNPs with merging window size of 100k. CEU, HCB, JPT, YRI datasets are from ENCODE project

datasets	#SNPs	#Rep SNPs	datasets	#SNPs	#Rep SNPs
CEU	7,221	2,484	chr2	169,905	85,807
HCB	6,430	2,286	chr3	135,058	71,244
JPT	6,216	2,196	chr19	28,931	17,807
YRI	7,963	4,408	chr21	28,914	15,644
chr1	149,716	78,893	chr22	26,595	15,553

The first experiment is to compare FastTagger with LRTag [Liu *et al.*, 2007], MMTagger [Wang and Jiang, 2008] and MultiTag [Hao, 2007]. LRTag uses only pair-wise LD to find tag SNPs, and it has been shown to outperform LD-select and FESTA. Hence we choose LRTag as a representative of the pairwise algorithms. MMTagger and MultiTag both use multi-marker LD to find tag SNPs. We obtained the programs from their respective authors. FastTagger used all the techniques described previously except the skipping rules technique. LRTag takes pre-computed pairwise r^2 statistics as input, so the running time of LRTag includes only tag SNP selection time. We report the results at min_r2=0.95 here. For all the four algorithms, the selected tag SNPs can cover the whole region of interest.

We first compare FastTagger with LRTag and MultiTag on using pairwise LD to find tag SNPs. Table 7.2 and 7.3 show the running time and the number of tag SNPs selected by the three algorithms. The running time is measured in minutes. FastTagger is several times faster than LRTag even though LRTag only needs to pick tag SNPs from pre-computed pairwise r^2 statistics while FastTagger needs to compute pairwise r^2 statistics as well as selecting tag SNPs. Both algorithms are orders of magnitude faster than MultiTag.

Among the three algorithms, LRTag produces the smallest number of tag SNPs, but the difference is very small. Overall, FastTagger generates 0.31% more tag SNPs than LRTag when min_r2=0.95. MultiTag generates 1.77% more tag

Table 7.2: Comparison of running time when pairwise LD are used

	min_r2	Running time (minutes)		
		FastTagger	LRTag	MultiTag
ENCODE CEU	0.95	0.003	0.016	10.4
ENCODE HCB	0.95	0.003	0.014	7.5
ENCODE JPT	0.95	0.003	0.013	6.6
ENCODE YRI	0.95	0.004	0.008	41.6
chr1	0.95	0.076	0.242	26.2
chr2	0.95	0.088	0.293	30.2
chr3	0.95	0.070	0.222	25.1
chr19	0.95	0.015	0.032	3.6
chr21	0.95	0.015	0.040	6.0
chr22	0.95	0.014	0.033	7.9

The running time of LRTag includes only tag SNP selection time, while the running time of FastTagger and MultiTag includes both rule generation time and tag SNP selection time. MMTagger is excluded from this table because the MMTagger program provided by its authors cannot use pairwise LD to find tag SNPs.

SNPs than FastTagger when min_r2=0.95. LRTag uses a Lagrangian relaxation algorithm to select tag SNPs instead of a greedy approach used in other algorithms. That is why it generates less tag SNPs than other algorithms.

Table 7.4 and 7.5 show the running time and the number of tag SNP selected by the FastTagger, MMTagger and MultiTag when multi-marker LD are used. We implemented both models in FastTagger, and denote them as Fast-COOC (the co-occurrence model) and Fast-1vsR (the one-vs-the-rest model). MultiTag took extremely long time to finish on the 6 chromosomes when max_size=3, so its results are not reported on the 6 chromosomes when max_size=3. When max_size=2, we divided chr1, chr2 and chr3 into 20 chunks, chr19, chr21 and chr22 into 5 chunks, and then ran MultiTag on each chunk and combined the results. MMTagger terminated abnormally on chr1, chr2 and chr3 when $max\_size = 3$ because too many rules were generated. To solve this problem, we divided the three chromosomes into 10 chunks, and then ran MMTagger on each chunk and combined the results together.

Table 7.5 shows that the multi-marker model can reduce the number of tag SNPs significantly under the same min_r2 threshold compared with the pairwise model (Table 7.3). The number of tag SNPs is reduced by more than 30% when max_size=2. When max_size=3, the number of tag SNPs is reduced by more

Table 7.3: Comparison of number of tag SNPs selected when pairwise LD are used

	min_r2	#tag SNPs		
		FastTagger	LRTag	MultiTag
ENCODE CEU	0.95	2144	2127	2136
ENCODE HCB	0.95	2065	2055	2061
ENCODE JPT	0.95	1996	1990	1996
ENCODE YRI	0.95	4115	4107	4109
chr1	0.95	62190	61988	63391
chr2	0.95	66026	65822	67236
chr3	0.95	55895	55713	56972
chr19	0.95	14777	14744	15014
chr21	0.95	12455	12435	12658
chr22	0.95	12690	12652	12932

than 40%. However, calculating multi-marker r^2 statistics is much more expensive than computing pairwise r^2. FastTagger is more than 10 times slower when max_size=2, and hundreds of times slower when max_size=3.

On ENCODE regions, FastTagger and MMTagger take similar time to finish when max_size=2; when max_size=3, FastTagger is 2-3 times faster than MMTagger. On the 6 chromosomes, FastTagger is 2-6 times faster than MMTagger. Both algorithms are orders of magnitude faster than MultiTag. The number of tag SNPs selected by FastTagger under the co-occurrence model is smaller than that selected by MMTagger and MultiTag. When max_size=3, FastTagger produces 3%-9% less tag SNPs than MMTagger.

Table 7.6 shows the maximum memory usage of FastTagger and MMTagger with $max\_r2 = 0.95$ and $max\_size = 3$. MMTagger consumes much more memory than FastTagger, that is why it cannot work on large chromosomes such as chr1, chr2 and chr3 when $max\_size = 3$.

Table 7.5 also shows that the co-occurrence model generates smaller set of tag SNPs than the one-vs-the-rest model. The reason being that more rules are generated under the co-occurrence model as shown in Table 7.7. When max_size=2, the two models generate similar number of rules, so does the number of tag SNPs. When $max\_size = 3$, the co-occurrence model generates 3-4 times more rules than the one-vs-the-rest model, hence it can use much less tag SNPs to tag all the other SNPs. The co-occurrence model also consumes much more memory when max_size=3 as shown in the last two columns of Table 7.7.

Table 7.4: Comparison of running time when multi-marker LD are used.

	max_size	*min_r2*	Running time (minutes) Fast-COOC	MMTagger	Fast-1vsR	MultiTag
ENCODE CEU	2	0.95	0.038	0.041	0.048	≥10 hours
ENCODE HCB	2	0.95	0.032	0.032	0.042	≥10 hours
ENCODE JPT	2	0.95	0.029	0.028	0.038	≥10 hours
ENCODE YRI	2	0.95	0.181	0.188	0.245	≥60 hours
chr1	2	0.95	1.13	5.84	1.40	≥7 days
chr2	2	0.95	1.32	7.21	1.63	≥7 days
chr3	2	0.95	1.14	5.11	1.41	≥7 days
chr19	2	0.95	0.176	0.343	0.218	≥30 hours
chr21	2	0.95	0.287	0.473	0.359	≥60 hours
chr22	2	0.95	0.370	0.567	0.468	≥100 hours
ENCODE CEU	3	0.95	1.28	3.69	1.85	≥50 hours
ENCODE HCB	3	0.95	1.26	3.40	1.93	≥80 hours
ENCODE JPT	3	0.95	1.06	2.74	1.60	≥50 hours
ENCODE YRI	3	0.95	11.6	36.7	17.4	≥14 days
chr1	3	0.95	34.9	137.3	49.6	-
chr2	3	0.95	42.9	166.9	60.8	-
chr3	3	0.95	39.3	154.6	55.5	-
chr19	3	0.95	4.34	16.6	6.25	-
chr21	3	0.95	9.91	37.7	14.4	-
chr22	3	0.95	16.5	65.3	24.4	-

Fast-COOC represents the FastTagger algorithm using the co-occurrence model, and Fast-1vsR represents the FastTagger algorithm using the one-vs-the-rest model. *max_size* is the maximum number of SNPs on the left hand side of a tagging rule. For the MMTagger algorithm, we divided chr1, chr2 and chr3 into 10 chunks when *max_size*=3, and ran MMTagger on each chunk, and then combined the results. For the MultiTag algorithm, we divided chr1, chr2 and chr3 into 20 chunks, chr19, chr21 and chr22 into 5 chunks when *max_size*=3. When *max_size*=3, MultiTag took too long to finish on the 6 chromosomes, so we did not get its results on the 6 chromosomes.

The effectiveness of the techniques used in FastTagger

This experiment studies the effectiveness of the techniques used by FastTagger in reducing running time and memory consumption. We used the co-occurrence model in this experiment because it generates more rules and is more memory demanding than the one-vs-the-rest model. The baseline FastTagger algorithm in this experiment uses two techniques as in the previous experiment: merging equivalent SNPs and pruning redundant tagging rules. The running time and memory consumption of the baseline algorithm, and the number of tag SNPs and tagging rules generated by the baseline algorithm on chr19, chr21 and chr22 when *max_size*=3 and *min_r2*=0.95 is shown in Table 7.8.

Table 7.5: Comparison of number of tag SNPs selected when multi-marker LD are used.

	max_size	*min_r2*	#tag SNPs			
			Fast-COOC	MMTagger	Fast-1vsR	MultiTag
ENCODE CEU	2	0.95	1282	1282	1291	1371
ENCODE HCB	2	0.95	1305	1328	1308	1424
ENCODE JPT	2	0.95	1234	1258	1240	1349
ENCODE YRI	2	0.95	2575	2618	2579	2770
chr1	2	0.95	43202	43483	43306	43462
chr2	2	0.95	44135	44556	44225	49289
chr3	2	0.95	37881	38206	37952	39300
chr19	2	0.95	11151	11192	11160	11747
chr21	2	0.95	8543	8627	8564	9103
chr22	2	0.95	8970	9025	8993	9533
ENCODE CEU	3	0.95	972	1017	1151	1244
ENCODE HCB	3	0.95	1003	1034	1170	1170
ENCODE JPT	3	0.95	958	1002	1129	1244
ENCODE YRI	3	0.95	1848	1927	2165	2516
chr1	3	0.95	35556	38185	40534	-
chr2	3	0.95	35502	38372	41129	-
chr3	3	0.95	30695	33041	35305	-
chr19	3	0.95	9444	10032	10546	-
chr21	3	0.95	6929	7404	7935	-
chr22	3	0.95	7327	7788	8392	-

Fast-COOC represents the FastTagger algorithm using the co-occurrence model, and Fast-1vsR represents the FastTagger algorithm using the one-vs-the-rest model. *max_size* is the maximum number of SNPs on the left hand side of a tagging rule. For the MMTagger algorithm, we divided chr1, chr2 and chr3 into 10 chunks when *max_size*=3, and ran MMTagger on each chunk, and then combined the results. For the MultiTag algorithm, we divided chr1, chr2 and chr3 into 20 chunks, chr19, chr21 and chr22 into 5 chunks when *max_size*=3. When *max_size*=3, MultiTag took too long to finish on the 6 chromosomes, so we did not get its results on the 6 chromosomes.

The "#Rep SNPs" column in Table 7.1 shows the number of representative SNPs after merging equivalent SNPs using window size of 100k. The number of SNPs is reduced by around a half. We have tried using a larger window size to merge equivalent SNPs, and the results show that larger window sizes do not achieve much further reduction. The reduction in the number of SNPs greatly reduces the number of rules to be tested. Table 7.9 shows the performance of FastTagger without merging equivalent SNPs. Without merging equivalent SNPs, FastTagger generates an excessive number of tagging rules, e.g., around 20 times more than that of merging equivalent SNPs, thus taking much longer time and

Table 7.6: Memory usage of FastTagger and MMTagger

	FastTagger	MMTagger		FastTagger	MMTagger
chr1	94.41MB	-	chr19	30.29MB	657MB
chr2	287.50MB	-	chr21	74.99MB	1210MB
chr3	119.72MB	-	chr22	50.20MB	1216MB

The co-occurrence model is used in FastTagger. min_r2=0.95, max_size=3.

Table 7.7: The number of tagging rules generated under the two models using the FastTagger algorithm (min_r2=0.9)

	max	#rules		memory	
	$\_size$	Fast-COOC	Fast-1vsR	Fast-COOC	Fast-1vsR
chr19	2	121,122	120,627	6.63MB	6.63MB
chr21	2	169,864	168,936	11.43MB	11.43MB
chr22	2	156,134	155,223	8.14MB	8.13MB
chr19	3	1,421,519	377,773	38.69MB	13.29MB
chr21	3	2,713,338	657,767	101.11MB	29.92MB
chr22	3	2,590,826	573,738	67.28MB	19.21MB

consuming much more memory. There is also a slight increase in the number of tag SNPs selected.

Table 7.10 shows the performance of FastTagger without pruning redundant rules. Pruning redundant rules can reduce the number of rules generated by 3 times, thus reducing the maximum memory usage of FastTagger by more than a half. Although identifying redundant rules can reduce the search space, it also incurs some overhead. Hence the running time of FastTagger does not decrease when it uses the pruning redundant rules technique.

Table 7.11 shows performance of FastTagger when the skipping rules technique is used. Here if a SNP appears in the right hand side no less than 5 times, the SNP will not be considered as right hand side any more. By using this technique, the number of rules generated is reduced by more than a half. The running time and memory usage of FastTagger is also reduced. The number of tag SNPs selected increases slightly, but it is still smaller than that generated by the MMTagger algorithm.

We also tested FastTagger under a memory constraint. The maximum memory can be used by FastTagger is limited to 50MB. We used the three large chromosomes, chr1, chr2 and chr3, in this experiment. All the three chromosomes

Table 7.8: Baseline algorithm: merging equivalent SNPs and pruning redundant rules, no skipping rules. The co-occurrence model is used. max_size=3, min_r2=0.95.

	time	#tag SNPs	mem	#rules
chr19	4.34	9444	30.29MB	*951,392*
chr21	9.91	6929	74.99MB	*1,747,900*
chr22	16.5	7327	50.20MB	*1,658,769*

Table 7.9: Baseline algorithm without merging equivalent SNPs. The co-occurrence model is used. max_size=3, min_r2=0.95.

	time	#tag SNPs	mem	#rules
chr19	31.4	9476	209.83MB	*17,798,798*
chr21	72.3	6959	555.42MB	*35,278,021*
chr22	90.5	7342	340.59MB	*30,954,495*

contain more than 100k SNPs. Table 7.12 shows even with as little as 50MB memory, FastTagger can still work on chromosomes with 100k SNPs. There is only a tiny increase in its running time and the number of tag SNPs generated by the baseline algorithm on chr19, chr21 and chr22 when max size=3 and min r2=0.95 is shown in Table 7.8.

7.3 RuleImpute: An application of FastTagger in SNP imputation

7.3.1 Background

SNP imputation and tag SNP selection are like two sides of one coin. Tag SNP selection tries to find a smaller number of tag SNPs to be genotyped by using the correlation between SNPs while SNP imputation uses typed SNPs and a reference panel to impute those missing/untyped SNPs. This process has several benefits:

Table 7.10: Baseline algorithm without pruning redundant rules. The co-occurrence model is used. max_size=3, min_r2=0.95.

	time	#tag SNPs	mem	#rules
chr19	4.24	9439	75.70MB	*3,048,090*
chr21	9.60	6942	191.86MB	*5,643,004*
chr22	15.8	7327	130.19MB	*5,563,473*

Table 7.11: Baseline algorithm with skipping rules: if a SNP appears in the right hand side no less than 5 times, the SNP will not be considered as right hand side any more. The co-occurrence model is used. max_size=3, min_r2=0.95.

	time	#tag SNPs	mem	#rules
chr19	3.66	9550	18.61MB	*461,139*
chr21	8.06	7086	40.74MB	*754,084*
chr22	13.5	7447	28.62MB	*755,309*

Table 7.12: Performance of Fast-COOC when memory size is restricted to 50MB ($max\_size = 3$, min_r2=0.95)

	No memory constraint			mem=50MB		
	time	#tag SNPs	mem	time	#tag SNPs	#chunks
chr1	34.9	35556	94.41MB	35.14	35561	16
chr2	42.9	35502	287.50MB	43.14	35518	21
chr3	39.3	30695	119.72MB	39.3	30706	15

(1) Spencer*et al.* [2009] show that imputation improves up to 10% power in association analysis.

(2) It provides a finer resolution of certain associated disease region and thus increases the chances of uncovering novel casual variants.

(3) It gives a unifying way to conduct meta-analysis for multiple GWAS. Since different GWAS use different genotype chips, imputation analysis can make each study has an equal number of SNPs given the same reference panel. Figure 7.1 from Marchini and Howie [2010] is used to illustrate how SNP imputation works. Before conducting an imputation, researchers merely use a study sample (part a) to do association analysis, and there may be few statistically significant SNPs (part b). Then researchers prepare the reference haplotypes (part d) such as HapMap and phased study sample (part c) for imputation. Different algorithms differ in how to make use of reference panel information but the majority of algorithms involve phasing the typed SNPs in the study sample (part c). The ultimate outputs of different imputation algorithms complete the missing parts of the original study sample (part e). Using the "complete" study sample to conduct the association analysis again, researchers may find some untyped SNPs having significant signals (part f).

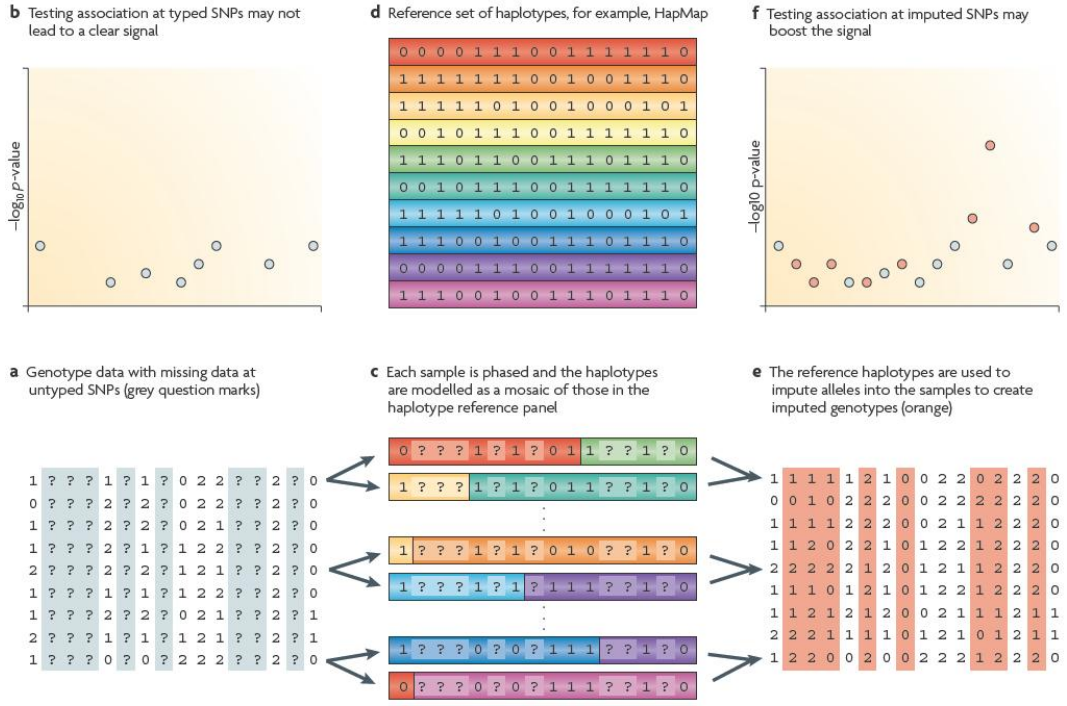

Figure 7.1: Illustration of the general workflow of SNP imputation

7.3.2 Methods

In the literature, there are mainly two types of methods for SNP imputation analysis: tagging-based methods and hidden markov model (HMM)-based methods.

Tagging-based methods

Plink [Purcell et al., 2007], TUNA [Nicolae, 2006] and our RuleImpute use tag SNPs to impute the missing SNPs. The typed SNPs are tagging SNPs and the missing SNPs are to be tagged. These methods search for tagging correlations from a reference panel, and then apply them to the study samples. The differences between these methods are in how they define the correlations between tagging SNPs and tagged SNPs and how they find the tagging correlations. The methods are usually simple and fast, however they do not make full use of all the available information in the reference panel and study sample. Their imputation accuracy is lower compared with methods based on HMM. Next, we introduce

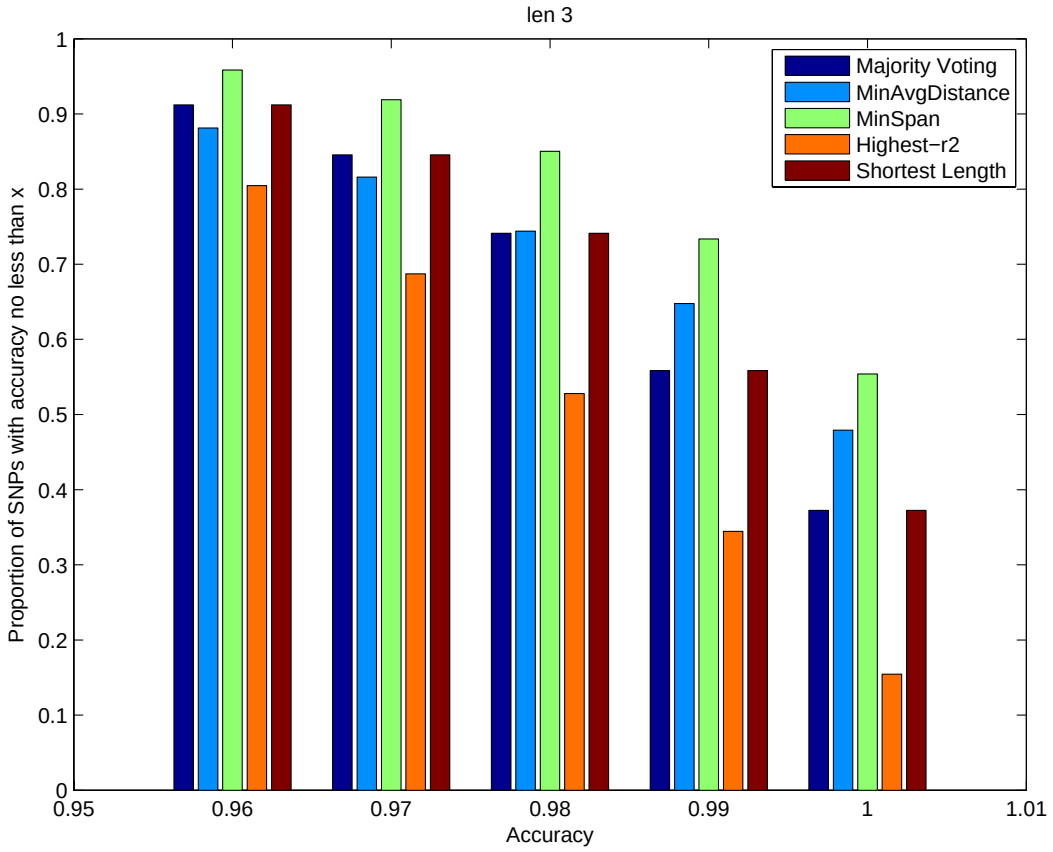

Figure 7.2: Performance comparison of five different ruler selection strategies.

our RuleImpute method which is derived from above-mentioned FastTagger.

RuleImpute. The RuleImpute algorithm is based on multi-marker tagging. It consists of two steps. In the first step, RuleImpute generates tagging rules of the form: S $\rightarrow$ SNP$_{untyped}$, where S $=\{$ SNP$_{typed_1}$, SNP$_{typed_2}$,$\cdots$, SNP$_{typed_k}$ $\}$ is a SNP set. The linkage disequilibrium of the rules is measured using the r^2 statistic, and the r^2 statistics of the rules are calculated using the method described in Equation 7.1. The generated rules must satisfy two constraints: 1) The span of a rule must be no larger than a threshold *max_span*. Here the span of a rule is defined as the maximal distance between every pair of SNPs in the rule. 2) The r^2 statistic of a rule must be no less than a *threshold* so that the alleles of SNP$_{untyped}$ can be inferred from the alleles of the SNPs on the left hand side reliably.

In the second step, RuleImpute infers the missing values using tagging rules. It is highly possible that some untyped SNP can be tagged by multiple rules, and we need to decide which rule should be used for imputation. Intuitively, rules with higher r^2 value, shorter length and smaller span should be selected first. To find out which metric should be given the highest priority, we designed five rule selection strategies:

(1) **Majority Voting**: We use every tagging rule of the untyped SNP to impute, and then use the alleles with the highest frequency as the final prediction.

(2) **MinAvgDistance**: For every rule, we calculate the average distance between the SNPs on the left hand side and the SNP on the right hand side, then select the SNPs with the smallest average distance.

(3) **MinSpan**: The rule with the smallest span is selected.

(4) **Highest-r^2**: The rule with the highest r^2 value is selected.

(5) **Shortest Length**: The rule with the shortest length is selected.

HMM-based methods

IMPUTE [Howie *et al.*, 2009], MACH [Li and Abecasis, 2006] and BIMBAM [Servin and Stephens, 2007] are the methods based on the variants of HMM. Instead of only using the local tagging relationship between several SNPs, these methods incorporate haplotype structures into some HMM. The basic components of these methods include states, observations, state transition probabilities and emission probabilities. Each SNP is modeled by several states, representing possible haplotypes at the SNP. According to this model, a haplotype is generated by walking the state chain. These methods differ in how to define the states for a SNP and how to learn transition probabilities and emission probabilities. IMPUTE and MACH use a large state set to represent all the possible haplotypes in the reference panel. However, BIMBAM uses clusters to represent hidden states, based on the observation that haplotypes have cluster structures. Users can set the parameter K to represent the number of clusters. As for the parameter learning of the HMM, IMPUTE and BIMBAM use a Markov Chain Monte Carlo method to perform the integration while MACH uses a local search

method.

7.3.3 Results and discussion

We use release 22 from the HapMap Project in our experiments (ftp://ftp.ncbi.nlm.nih.gov/hapmap/phasing/2007-08_rel22/). From release 22, we pick chromosome 22 of U.S. residents of northern and western European ancestry (CEU), which consists of 33,815 SNPs and 60 individuals. We partition the data into four folds evenly, and each test fold contains approximately 25% of individuals. We picked every other SNP as typed SNPs. We set the length of tagging rules to 3, the r^2 threshold to 0.9, and the maximum span threshold to 100,000.

We define the accuracy of an untyped SNP as the proportion of alleles of this SNP that are correctly predicted, and the accuracy of each untyped SNP is averaged over four folds. We then calculate the proportion of SNPs with accuracy no less than a certain threshold for the five selection strategies. Figure 7.2 shows that among the five strategies the MinSpan strategy achieves the best results. The MinSpan rules contains SNPs that are physically close to each other, and these SNPs have higher chance to inherit with each other from ancient population. Therefore their tagging relationship is more reliable compared to other strategies. However, the Highest r^2 strategy gives the worst performance. This is probably because longer rules are less reliable than shorter rules even if they have higher r^2 value.

Based on the above results, in our RuleImpute algorithm, we should first select rules based on their spans, then based on their lengths, and lastly based on their r^2 values.

Chapter 8

Conclusion

By first conducting an independent third-party experiments on state-of-the-art methods in detecting epistatic interaction for GWAS, we have shown the importance of enhancing the computational power to analyze the data. The computational challenge is an unavoidable obstacle in GWAS, although different heuristic techniques are used. We have thus proposed the first-ever cloud computing model (CEO) using big data technologies to detect the epistatic interactions. Experimental results show this brand new computing model can easily handle current GWAS data in a reasonable amount of time on an affordable cluster size. We have further proposed an improved computing model (eCEO) by introducing a Boolean representation and bit operation on the data, which is motivated by Wan et al. [2010a]. This optimization technique can be applied to different test statistics that require the construction of contingency tables. Therefore, we can easily extend the computing model to incorporate those test statistics. The performance study of eCEO shows further speedup against CEO. The CEO and eCEO point to a new direction for computational analysis of the GWAS data that is computationally efficient, flexible, scalable and practical. However, CEO and eCEO are only limited to the analysis of case-control GWAS data. Recently, more challenging high-dimensional quantitative phenotype data in GWAS are available because they are believed to be closer to characterizing some diseases than coarse case-control phenotypes. Potkin *et al.* [2009] also show that using quantitative trait has higher power than using case-control trait. MRI imaging data is a good example of this kind, which is used as a de facto data type for studying the relation

between genotype and high-dimensional imaging phenotype. Noticing the gap between the demand of analyzing complex high-dimensional phenotype GWAS data and the available computational power, we have proposed a powerful parallel distance-based random forest algorithm to ease the computational challenge. Unlike traditional GWAS analysis which only reports top-ranked associated SNPs, we have extracted more insights by proposing two hypotheses:

Hypothesis A: The further away an AD sample is from the CN cluster in Figure 6.8, the greater the severity of the Alzheimer's disease of the AD sample.

Hypothesis B: The more number of certain combinations of mutations are observed in a person, the more severely he suffers from Alzheimer's disease.

The validation of hypothesis A is not possible since we have no clinical information of the AD patients. So we assume that Hypothesis A is true, and attempt to verify Hypothesis B with respect to it. We first define the severity of patients using two types of distance to the centroid of the healthy cluster, and then mine "frequent" mutation patterns. By analyzing the relation between the severity of the disease and the count of such statistically frequent mutation patterns among 253 study samples, we show that Hypothesis B is well supported. The analytical results deepen our understanding of the disease: The more mutation patterns a sample carries, the more severe disease a sample has. This analysis opens a new door to uncover the etiology of complex diseases using GWAS data. We expect more results to come out in the future.

We also address two other research problems in GWAS: tag SNP selection and SNP imputation. Considered the connection of the two problems, we first present an efficient algorithm called FastTagger for genome-wide tag SNP selection using multi-marker LD. FastTagger uses several techniques to reduce running time and memory consumption. Our experiment results show that FastTagger is several times faster than existing tag SNP selection algorithms using multi-marker models, and it consumes much less memory at the same time, which enables FastTagger to work on chromosomes containing more than 100k SNPs when existing algorithms using multi-marker models usually fail. FastTagger also selects

less tag SNPs than existing algorithms using multi-marker LD. Our experiment results also show that merging equivalent SNPs together is the most effective technique in reducing running time and memory consumption. We provide two multi-marker models in FastTagger. The one-vs-the-rest model generates rules with higher average r^2 and higher average accuracy than the co-occurrence model under the same parameter settings. However, it generates much fewer length-3 rules than the co-occurrence model, thus requiring more tag SNPs to cover all the other SNPs. We then apply FastTagger to develop a SNP imputation method called RuleImpute, which uses the rules produced by FastTagger to impute missing SNPs. Facing a large number of generated rules, we propose five different rule selection strategies to prioritize rules to be used for imputation. Our experimental results suggest that MinSpan rules should be selected at first to get better accuracy.

Bibliography

R. Agrawal, R. Srikant (1994). Fast algorithms for mining association rules in large databases. In *Procceding of the 20th VLDB conference*, pages 487–499. 41

A. Agresti (2002). *Categorical Data Analysis*. John Wiley & Sons, Inc., 2002. 44

A. Albrechtsen, S. Castella, et al. (2006). A Bayesian multilocus association method: allowing for higher-order interaction in association studies. *Genetics*, **176**(2), 1197–1208. 26

H. I. Avi-Itzhak, X. Su, et al. (2003). Selection of minimum subsets of single nucleotide polymorphisms to capture haplotype block diversity. In *In Pac Symp Biocomput*, pages 466–477. 121

P. D. Bakker, R. Yelensky, et al. (2005). Efficiency and power in genetic association studies. *Nat Genet*, **37**, 1217–1223. 122

D. J. Balding (2006). A tutorial on statistical methods for population association studies. *Nat Rev Genet* ,**7**(10), 781-791. 61, 76

J. D. Basilico, M. A. Munson, et al. (2011). COMET: A recipe for learning and using large ensembles on massive data. In *Proceedings of 2011 IEEE International Conference on Data Mining*, pages 41–50. 115

W. Bateson (1909). *Mendel's Principles of Heredity*. University Press, 1909. 2, 3, 27, 38

Y. Benjamini, Y. Hochberg (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J R Stat Soc Series B Stat Methodol*, **57**(1), 289–300. 44

A. L. Boulesteix, A. Bender, et al. (2012). Random forest Gini importance favours SNPs with large minor allele frequency: Impact, sources and recommendations. *Brief Bioinform*, **13**(3), 292–304. 94

M. N. Braskie, J. M. Ringman, et al. (2011). Neuroimaging measures as endophenotypes in Alzheimer's disease. *Int J Alzheimers Dis*, **2011**, 490140. 101, 107

L. Breiman (2001). Random forests. *Machine Learning*, **45**(1):5–32. 4, 39, 97, 99

C. C. Buchanan, E. S. Torstenson, et al. (2012). A comparison of cataloged variation between International HapMap Consortium and 1000 Genomes Project data. *J Am Med Inform Assoc*, **19**(2), 289–294. 18

P. R. Burton, D. G. Clayton, et al. (2007). Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, **447**(7145), 661–678. 2

W. S. Bush, J. Haines (2001). *Overview of Linkage Analysis in Complex Traits*. John Wiley & Sons, Inc. 18

W. S. Bush, S. M. Dudek, et al. (2006). Parallel multifactor dimensionality reduction: A tool for the large-scale analysis of gene-gene interactions. *Bioinformatics*, **22**(17), 2173–2174. 29

C. Carlson (2004). Selecting a maximally informative set of single-nucleotide polymorphisms for association analyses using linkage disequilibrium. *Am J Hum Genet*, **74**(1), 106–120. 121, 122, 124, 126

A. Chakravarti (1998). Population genetics—making sense out of sequence. *Nat Genet*, **21**(Suppl 1), 56–60. 18

P. Chanda, L. Sucheston, et al. (2008). AMBIENCE: A novel approach and efficient algorithm for identifying informative genetic and environmental associations with complex phenotypes. *Genetics*, **180**(2), 1191–1210. 39

Y. Chung, S. Y .Lee, et al. (2006). Odds ratio based multifactor-dimensionality reduction method for detecting gene-gene interactions. *Bioinformatics*, **23**(1), 71–76. 29

N. R. Cook, R. Y. L. Zee, et al.(2004). Tree and spline based association analysis of gene–gene interaction models for ischemic stroke. *Stat in Med*, **23**(9), 1439–1453. 4

H. J. Cordell (2002). Epistasis: What it means, what it doesn't mean, and statistical methods to detect it in humans. *Hum Mol Genet*, **11**(20), 2463–2468. 27, 28, 49

R. Culverhouse (2007). The use of the restricted partition method with case-control data. *Hum Hered*, **63**(2), 93–100. 39

L. A. Cupples, H. T. Arruda, et al. (2007). The Framingham Heart Study 100K SNP genome-wide association study resource: Overview of 17 phenotype working group reports. *BMC Med Genet*, **8**(Suppl 1), S1. 20

H. Dai, M. Bhandary, et al. (2012) Global tests of p-values for multifactor dimensionality reduction models in selection of optimal number of target genes. *BioData Min*, **5**(1), 3. 29

J. Dean, S. Ghemawat (2004). Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating Systems Design & Implementation*, pages 137–149. 5, 30, 59, 73, 95

E. E. Eichler, J. Flint, G. Gibson, et al. (2010). Missing heritability and strategies for finding the underlying causes of complex disease. *Nat Rev Genet*, **11**(6), 446–450. 2

T. Emahazion, L. Feuk, et al. (2001). SNP association studies in Alzheimer's disease highlight problems for complex disease analysis. *Trends Genet*, **17** (7), 407–413. 37

M. Emily (2012). IndOR: A new statistical procedure to test for SNP-SNP epistasis in genome-wide association studies. *Stat Med.* **31**(21), 2359–2373. 28

K. Estrada, M. Krawczak, et al. (2009). A genome-wide association study of northwestern Europeans involves the C-type natriuretic peptide signaling pathway in the etiology of human height variation. *Hum Mol Genet*, **18**(18), 3516–3524. 20

Y. H. Fang and Y. F. Chiu (2012). SVM-based generalized multifactor dimensionality reduction approaches for detecting gene-gene interactions in family studies. *Genet Epidemiol*, **36**(2), 88–98. 29

M. A. R. Ferreira, S. Purcell (2009). A multivariate test of association. *Bioinformatics*, **25**(1), 132–133. 94

T. M. Frayling, M. I. McCarthy (2007). Genetic studies of diabetes following the advent of the genome-wide association study: Where do we go from here? *Diabetologia*, **50**(11), 2229–2233. 29

K. A. Frazer, S. S. Murray, et al. (2009). Human genetic variation and its contribution to complex traits. *Nat Rev Genet*, **10**(4), 241–251. 93

S. B. Gabriel, S. F. Schaffner, et al. (2002). The structure of haplotype blocks in the human genome. *Science*, **296**(5576), 2225–2229. 121

S. Ghemawat, H. Gobioff, et al. (2003). The Google File System. In *Proceedings of 19th ACM Symposium on Operating Systems Principles*, pages 1–15. 5, 31

D. C. Glahn, P. M. Thompson, et al. (2007). Neuroimaging endophenotypes: strategies for finding genes influencing brain structure and function. *Hum Brain Mapp*, **28**(6), 488–501. 94

B. A. Goldstein, A. E. Hubbard, et al. (2010). An application of random forests to a genome-wide association dataset: Methodological considerations & new findings. *BMC Genet*, **11**:49. 94, 107

B. A. Goldstein, E. C. Polley, et al. (2011). Random forests for genetic association studies. *Stat Appl Genet Mol Biol*, **10**(1). Art. 32 94, 99

H. Grahn, N. Lavesson, et al. (2011). CudaRF: A CUDA-based implementation of random forests. In *Proceedings of the 9th IEEE/ACS International Conference on Computer Systems and Applications*, pages 95–101. 117

C. S. Greene, N. A. Sinnott-Armstrong, et al. (2010). Multifactor dimensionality reduction for graphics processing units enables genome-wide testing of epistasis in sporadic ALS. *Bioinformatics*, **26**(5), 694–695. 59

B. Halldorsson. (2004). Optimal haplotype block-free selection of tagging snps for genome-wide association studies. *Genome Res*, **14**, 1633–1640. 121

E. Halperin. (2005). Tag SNP selection in genotype data for maximizing SNP prediction accuracy. *Bioinformatics*, **21**, 195–203. 121

K. Hao (2007). Genome-wide selection of tag SNPs using multiple-marker correlation. *Bioinformatics*, **23**(23), 3178–3184. 122, 123, 126, 128

K. Hao, X. Di, et al. (2007). LDcompare: Rapid computation of single and multiple marker r^2 and genetic coverage. *Bioinformatics*, **23**(2), 252–254. 122

I. H. Herskowitz(1977). *Principles of Genetics*. New York, Macmillan. 16

D. P. Hibar, O. Kohannim, et al. (2011). Multilocus genetic analysis of brain images. *Frontiers in Genetics*, **2**(73), 1–12. 95

W. Hill (1974). Estimation of linkage disequilibrium in randomly mating populations. *Heredity*, **33**(2), 229–239. 124

W. Hill (1975). Tests for association of gene frequencies at several loci in random mating diploid populations. *Bioinformatics*, **31**(4), 881–888. 124

J. Hill, M. Hambley, et al. (2008). SPRINT: A new parallel framework for R. *BMC Bioinformatics*, **9**:1. 117

C. J. Hoggart, J. C. Whittaker, et al. (2008). Simultaneous analysis of all SNPs in genome-wide and re-sequencing association studies. *PLoS Genet*, **4** (7), e1000130. 1

D. W. Hosmer (2000). *Applied Logistic Regression.* John Wiley & Sons, Inc., 2000. 39

B. Howie (2006). Efficient selection of tagging single-nucleotide polymorphisms in multiple populations. *Hum Genet*, **120**, 58–68. 122

B. N. Howie, P. Donnelly, et al. (2009). A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. *PLoS Genet*, **5**(6), e1000529. 21, 138

Y. T. Huang and K. M. Chao(2008). A new framework for the selection of tag SNPs by multimarker haplotypes. *J Biomed Inform*, **41**(6), 953–961. 122

R. Jiang, W. W. Tang, et al. (2009). A random forest approach to the detection of epistatic interactions in case-control studies. *BMC Bioinformatics*, **10**(Suppl 1), S65. 4, 94

G. C. Johnson, L. Esposito, et al. (2001). Haplotype tagging for the identification of common disease genes. *Nat Genet*, **29**, 233–237. 121

L. Jourdren, M. Bernard, et al. (2012). Eoulsan: A cloud computing-based framework facilitating high throughput sequencing analyses. *Bioinformatics*, **28**(11), 1542–1543. 7

R. Kilpikari, M. J. Sillanp (2003). Bayesian analysis of multilocus association in quantitative and qualitative traits. *Genet Epidemiol*, **25**(2),122–135. 26

S. Kim, N. J. Morris, et al. (2010). Single-marker and two-marker association tests for unphased case-control genotype data, with a power comparison. *Genet Epidemiol*, **34**(1), 67–77. 26

R. J. Klein (2005). Complement factor H polymorphism in age-related macular degeneration. *Science*, **308**(5720), 385–389. 2, 37

L. C. Kwee, D. Liu, et al. (2008). A powerful and flexible multilocus association test for quantitative traits. *Am J Hum Genet*, **82**(2), 386–397. 26

E. S. Lander (1996). The new genomics: Global views of biology. *Science*, **274**(5287), 536–539. 18

E. S. Lander (2001). On the allelic spectrum of human disease. *Trends Genet*, **17**(9), 502–510. 18

B. Langmead, K. D. Hansen, et al. (2010). Cloud-scale RNA-sequencing differential expression analysis with Myrna. *Genome Biol*, **11**(8), R83. 7

B. Langmead, C. Trapnell, et al. (2009). Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol*, **10**(3), R25. 7

S. Y. Lee, Y. Chung, et al. (2007). Log-linear model-based multifactor dimensionality reduction method to detect gene gene interactions. *Bioinformatics*, **23**(19), 2589–2595. 29

F. Lescai, C. Franceschi (2010). The impact of phenocopy on the genetic analysis of complex traits. *PLoS One*, **5**(7), e11876. 50

D. Levy, G. B. Ehret, et al. (2009). Genome-wide association study of blood pressure and hypertension. *Nat Genet*, **41**(6), 677–687. 20

Z. Li, T. Zheng, et al. (2006). Pattern-based mining strategy to detect multi-locus association and gene × environment interaction. *BMC Proceedings*, **10**(Suppl 1), S16. 41

Y. Li, G. R. Abecasis (2006). Mach 1.0: Rapid haplotype reconstruction and missing genotype inference. *Annu Meet Am Soc Hum Genet*, abstract 2290. 138

W. Li, J. Reich (2000). A complete enumeration and classification of two-locus disease models. *Hum Hered*, **50**(6), 334–349. 45

Q. Li, G. Zheng, et al. (2009). Robust tests for single-marker analysis in case-control genetic association studies. *Ann Hum Genet*, **73**(2), 245–252. 25

C. Libioulle, E. Louis, et al. (2007). Novel Crohn disease locus identified by genome-wide association maps to a gene desert on 5p13.1 and modulates expression of PTGER4. *PLoS Genet*, **3**(4), e58. 29

A. M. Lindenberg (2012). The future of fMRI and genetics research. *NeuroImage*, **62**(2), 1286–1292. 94

A. M. Lindenberg, D. R. Weinberger (2006). Intermediate phenotypes and genetic mechanisms of psychiatric disorders. *Nat Rev Neurosci*, **7**(10), 818–827. 95

F. Liu, F. v. d. Lijn, et al. (2012). A genome-wide association study identifies five loci influencing facial morphology in Europeans. *PLoS Genet*, **8**(9), e1002932. 118

Z. Liu, Y. Shen, et al. (2011). Multilocus association mapping using generalized ridge logistic regression. *BMC Bioinformatics*, **12**(1), 384. 26

T. Liu, A. Thalamuthu, et al. (2011). Asymptotic distribution for epistatic tests in case-control studies. *Genomics*, **98**(2), 145–151. 55

L. Liu, Y. Wu, et al. (2007). Efficient algorithms for genome-wide tag SNPs selection across populations via linkage disequilibrium criterion. In *Proceedings of 6th Annual International Conference on Computational Systems Bioinformatics*, pages 67–78. 121, 122, 128

G. Liu, J. Li, et al. (2008). A new concise representation of frequent itemsets using generators and a positive border. *Knowl. Inf. Syst.*, **17**(1), 35–56. 125

B. A. Logsdon, C. L. Carty, et al. (2012). A novel variational Bayes multiple locus Z-statistic for genome-wide association studies with Bayesian model averaging. *Bioinformatics*, **28**(13), 1738–1744. 27

Q. Long, Q. Zhang, et al. (2009). Detecting disease-associated genotype patterns. *BMC Bioinformatics*, **10**(Suppl 1), S75. 4, 41

L. D. Lobel, P. Geurts, et al. (2010). A screening methodology based on Random Forests to improve the detection of gene-gene interactions. *Eur J Hum Genet*, **18**(10), 1127–1132. 99

K. L. Lunetta, L. B. Hayward, et al. (2004). Screening large-scale association study data: Exploiting interactions using random forests. *BMC Genet*, **5**(32),1127–1132. 4, 94

J. L. Stein, Alzheimer's Disease Neuroimaging Initiative, et al. (2010a). Genome-wide analysis reveals novel genes influencing temporal lobe structure with relevance to neurodegeneration in Alzheimer's disease. *Neuroimage*, **51**(2), 542–554. 95

J. L. Stein, X. Hua, et al. (2010b). Voxelwise genome-wide association study (vGWAS). *NeuroImage*, **53**(3), 1160–1174. 20, 95, 99, 102

L. Ma, H. B. Runesha, et al. (2008). Parallel and serial computing tools for testing single-locus and epistatic SNP effects of quantitative traits in genome-wide association studies. *BMC Bioinformatics*, **9**(315). 4, 40, 59, 72, 79, 83

T. F. C. Mackay, E. A. Stone, et al. (2009). The genetics of quantitative traits: Challenges and prospects. *Nat Rev Genet*, **10**(8), 565–577. 94

R. Magi, L. Kaplinski, et al. (2006). The whole genome tagsnp selection and transferability among hapmap populations. *Pac Symp Biocomput*, **11**, 535–543. 121, 122

T. A. Manolio, F. S. Collins, et al. (2009). Finding the missing heritability of complex diseases. *Nature*, **461**, 747–753. 2

J. Manyika, M. Chui, et al. (2011). Big data: The next frontier for innovation, competition, and productivity. Technical report, McKinsey Global Institute, May 2011. 5, 6

J. Marchini, P. Donnelly, et al. (2005). Genome-wide strategies for detecting multiple loci that influence complex diseases. *Nat Genet*, **37**(4), 413–417. 45

J. Marchini and B. Howie. (2010). Genotype imputation for genome-wide association studies. *Nat Rev Genet*, **11**(7), 499–511. 135

P. Marttinen, J. Corander (2010). Efficient Bayesian approach for multilocus association mapping including gene-gene interactions. *BMC Bioinformatics*, **11**(1), 443. 26

H. Matsuda (2000). Physical nature of higher-order mutual information: Intrinsic correlations and frustration. *Phys Rev E*, **62**(3 Pt A), 3096-3102. 44

J. Millstein, D. V. Conti, et al. (2006). A testing framework for identifying susceptibility genes in the presence of epistasis. *Am J Hum Genet*, **78**(1), 15–27. 39

C. Minas, S. J Waddell, et al. (2011). Distance-based differential analysis of gene curves. *Bioinformatics*, **27**(22), 3135–3541. 99

A. M. Molinaro, N. Carriero, et al. (2011). Power of data mining methods to detect genetic associations and interactions. *Hum Hered*, **72**(2), 85–97. 94

J. H. Moore and S. M. Williams (2009). Epistasis and its implications for personal genetics. *Am J Hum Genet*, **85**(3), 309-320. 58

J. H. Moore, F. W. Asselbergs, et al. (2010). Bioinformatics challenges for genome-wide association studies. *Bioinformatics*, **26**(4), 445-455. 58

J. H. Moore (2005). A global view of epistasis. *Nat Genet*, **37**(1), 13–14. 27

A. A. Motsinger, M. D. Ritchie, et al. (2006). The effect of reduction in cross-validation intervals on the performance of multifactor dimensionality reduction. *Genet Epidemiol*, **30**(6), 546–555, 2006. 29

A. A. Motsinger-Reif, S. M. Dudek, et al. (2008). Comparison of approaches for machine-learning optimization of neural networks for detecting gene-gene interactions in Genet Epidemiology. *Genet Epidemiol*, **32**(4), 325–340. 38

A. A. Motsinger-Reif, D. M. Reif, et al. (2008). A comparison of analytical methods for genetic association studies. *Genet Epidemiol*, **32**(8), 767–778. 38, 39

A. A. Motsinger-Reif, D. M. Reif, et al. (2006). Understanding the evolutionary process of grammatical evolution neural networks for feature selection in Genet Epidemiol. In *Proccedings of IEEE Symposium Computational Intelligence in Bioinformatics and Computational Biology*, pages 1–8. 39

NCI-NHGRI Working Group on Replication in Association Studies (2007). Replicating genotype-phenotype associations. *Nature*, **447**, 655–660. 29

K. K. Nicodemus (2011). Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures. *Brief Bioinform*, **12**(4), 369–373. 94

D. Nicolae (2006). Quantifying the amount of missing information in genetic association studies. *Genet Epidemiol*, **30**(8), 703–717. 136

M. Niemenmaa, A. Kallio, et al. (2012). Hadoop-BAM: Directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, **28**(6), 876–877. 7

M. Y. Park, T. Hastie (2008). Penalized logistic regression for detecting gene interactions. *Biostatistics*, **9**(1), 30–50. 42

N. Patil, A. J. Berno, et al. (2001). Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, **294**(5547), 1719–1723. 121

K. A. Pattin, B. C. White, et al. (2009). A computationally efficient hypothesis testing method for epistasis analysis using multifactor dimensionality reduction. *Genet Epidemiol*, **33**(1), 87–94. 29

I. Pe'er (2006). Evaluating and improving power in whole-genome association studies using fixed marker sets. *Nat Genet*, **38**, 663–667. 122

M. A. Pericak-Vance (2001). *Analysis of Genetic Linkage Data for Mendelian Traits*. John Wiley & Sons, Inc. 18

P. C. Phillips (1998). The language of gene interaction. *Genetics*, **149**(3), 1167–1171. 38

P. C. Phillips (2008). Epistasis—the essential role of gene interactions in the structure and evolution of genetic systems. *Nat Rev Genet*, **9**(11), 855–867. 2, 38

S. G. Potkin, J. A. Turner, et al. (2009). Genome-wide strategies for discovering genetic influences on cognition and cognitive disorders: Methodological considerations. *Cogn Neuropsychiatry*, **14**(4-5), 391–418. 140

J. K. Pritchard and M. Przeworski (2001). Linkage disequilibrium in humans: Models and data. *Am J Hum Genet*, **69**(1), 1–14. 122

J. K. Pritchard, M. Stephens, et al. (2000). Inference of population structure using multilocus genotype data. *Genetics*, **155**(2), 945–959. 22

Psychiatric GWAS Consortium Coordinating Committee, et al. (2009). Genome-wide association studies: History, rationale, and prospects for psychiatric disorders. *Am J Psychiatry*, **166**(5), 540–556. 1, 2

S. Purcell, B. Neale, et al. (2007). PLINK: A tool set for whole-genome association and population-based linkage analyses. *Am J Hum Genet*, **81**(3), 559–575. 21, 39, 136

Z. Qin, S. Gopalakrishnan, et al. (2006). An efficient comprehensive search algorithm for tagsnp selection using linkage disequilibrium criteria. *Bioinformatics*, **22**(2), 220–225. 121, 122

N. Risch, K. Merikangas (1996). The future of genetic studies of complex diseases. *Science*, **273**(5281), 1516–1517. 18

M. D. Ritchie, L. W. Hahn, et al. (2001). Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *Am J Hum Genet*, **69**(1), 138–147. 28, 38, 39, 41

M. Rubinov, O. Sporns (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, **52**(3), 1059–1069. 117

A. J. Saykin, L. Shen, et al. (2010). Alzheimer's disease neuroimaging initiative biomarkers as quantitative phenotypes: Genetics core aims, progress, and plans. *Alzheimers Dement*, **6**(3), 265–273. 102, 107, 108

M. C. Schatz, B. Langmead, et al. (2010). Cloud computing and the DNA data race. *Nat Biotechnol*, **28**(7), 691–693. 7

D. F. Schwarz, I. R. König, et al. (2010). On safari to random jungle: A fast implementation of random forests for high-dimensional data. *Bioinformatics*, **26**(14), 1752–1758. 117

P. Sebastiani (2003). Minimal haplotype tagging. *Proc Natl Acad Sci*, **100**, 9900–9905. 121

M. Segal (1992). Tree-structured methods for longitudinal data. *J Am Stat Assoc*, **87**(418), 407–418. 98

M. Segal, Y. Y. Xiao (2011). Multivariate random forests. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **1**(1), 80–87. 98

B. Servin and M. Stephens (2007). Imputation-based analysis of association studies: Candidate regions and quantitative traits. *PLoS Genet*, **3**(7), e114. 138

T. Sharp (2008). Implementing decision trees and forests on a GPU. In *Proceedings of 2008 European Conference on Computer Vision*, pages 595–608. 117

M. Silver, E. Janousova, et al. (2012). Identification of gene pathways implicated in Alzheimer's disease using longitudinal imaging phenotypes with sparse regression. *NeuroImage*, **63**(2012), 1681–1694. 95, 103, 107

X. Sim, R. T. H. Ong, et al. (2011). Transferability of type 2 diabetes implicated loci in multi-ethnic cohorts from Southeast Asia. *PLoS Genet*, **7**(4), e1001363. 29

T. Slavin, T. Feng, et al. (2011). Two-marker association tests yield new disease associations for coronary artery disease and hypertension. *Hum Genet*, **130**(6), 725–733. 26

D. J. A. Smit, D. v. Ent, et al. (2012). Neuroimaging and genetics: Exploring, searching, and finding. *Twin Res Hum Genet*, **15**(3), 267–272. 94

C. C. A. Spencer, Z. Su, et al. (2009). Designing genome-wide association studies: Sample size, power, imputation, and the choice of genotyping chip. *PLoS Genet*, **5**(5), e1000477. 135

L. D. Stein (2004). Human genome: End of the beginning. *Nature*, **431**(3), 915–916. 14

M. Stephens, D. J. Balding (2009). Modelling: Bayesian statistical methods for genetic association studies. *Nat Rev Genet*, **10**(10), 681–690. 27

M. Stephens, N. Smith, et al. (2001). A new statistical method for haplotype reconstruction from population data. *Am J Hum Genet*, **68**, 978–989. 124

Z. Su, N. Cardin, et al. (2009). A Bayesian method for detecting and characterizing allelic heterogeneity and boosting signals in genome-wide association studies. *Statistical Science*, **24**(4), 430–450. 26

L. Sucheston, P. Chanda, et al. (2010). Comparison of information-theoretic to statistical methods for gene-gene interactions in the presence of genetic heterogeneity. *BMC Genomics*, **11**(487). 38, 39

Y. V. Sun (2010). Multigenic modeling of complex disease by random forests. *Adv Genet*, **72**, 73–99. 94

Y. Y. Teo (2008). Common statistical issues in genome-wide association studies: a review on power, data quality control, genotype calling and population structure. *Curr Opin Lipidol*, **19**(2), 133–143. 20

D. Thompson, D. Stram, et al. (2003). Haplotype tagging single nucleotide polymorphisms and association studies. *Hum Hered*, **56**, 48–55. 121

D. R. Velez, B. C. White, et al. (2007). A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. *Genet Epidemiol*, **31**(4), 306–315. 29

P. M. Visscher, M. A. Brown, et al. (2012). Five years of GWAS discovery. *Am J Hum Genet*, **90**(1), 7–24. 2

M. Vounou, T. Nichols, et al. (2010). Discovering genetic associations with high-dimensional neuroimaging phenotypes: A sparse reduced-rank regression approach. *NeuroImage*, **53**(3), 1147–1159. 102, 103

M. Vounou, Alzheimer's Disease Neuroimaging Initiative, et al. (2012). Sparse reduced-rank regression detects genetic associations with voxel-wise longitudinal phenotypes in Alzheimer's disease. *Neuroimage*, **60**(1), 700–716. 95, 98

X. Wan, C. Yang, et al. (2010). Predictive rule inference for epistatic interaction detection in genome-wide association studies. *Bioinformatics*, **26**(1), 30–37. 28, 38, 39, 41

X. Wan, C. Yang, et al. (2010). BOOST: A fast approach to detecting gene-gene interactions in genome-wide case-control studies. *Am J Hum Genet*, **87**(3), 325–340. 28, 38, 39, 43, 59, 74, 140

K. Wang, D. Abbott (2008). A principal components regression approach to multilocus genetic association studies. *Genet Epidemiol*, **32**(2), 108–118. 26

W. B. Wang, and T. Jiang (2008). A new model of multi-marker correlation for genome-wide tag snp selection. In *Proc. of the International Conference on Genome Informatics*. 122, 123, 128

X. Wang, N. J. Morris, et al. (2012). Power of single- vs. multi-marker tests of association. *Genet Epidemiol*, **36**(5), 480–487. 1

Z. Wang, D. Agrawal, et al. (2012). COSAC: A framework for COmbinatorial Statistical Analysis on Cloud. *IEEE Trans Knowl Data Eng*, preprint. 7

Z. Wang, Y. Wang, et al. (2010). CEO: A Cloud Epistasis cOmputing model in GWAS. In *Proceedings of 4th IEEE International Conference on Bioinformatics and Biomedicine*, pages 85–90. 7, 76

Z. Wang, T. Liu, et al. (2010). A general model for multilocus epistatic interactions in case-control studies. *PLoS One*, **5**(8), e11384. 55

Z. Wang, Y. Wang, et al. (2011). eCEO: An efficient Cloud Epistasis cOmputing model in genome-wide association study. *Bioinformatics*, **27**(8), 1045–1051. 7, 41

P. Wegner (1960). A technique for counting ones in a binary computer. *Commun ACM*, **3**(5), 322. 43

R. A. Wilke, R. K. Mareedu, et al. (2008). The pathway less traveled: Moving from candidate genes to candidate pathways in the analysis of genome-wide

data from large scale pharmacogenetic association studies *Curr. Pharmacogenomics Person Med*, **6**(3), 150–159. 3

J. Wu, B. Devlin, et al. (2010). Screen and Clean: A tool for identifying interactions in genome-wide association studies. *Genet Epidemiol*, **34**(3),275–285. 38, 39, 42, 59

T. T. Wu, Y. F. Chen, et al. (2009). Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, **25**(6), 714–721. 26, 38, 59

Y. Y. Xiao, M. R. Segal (2009). Identification of yeast transcriptional regulation networks using multivariate random forests. *PLoS Comput Biol*, **5**(6), e1000414. 118

F. Xue, S. Li, et al. (2012). A latent variable partial least squares path modeling approach to regional association and polygenic effect with applications to a human obesity study. *PLoS ONE*, **7**(2), e31927. 26

C. Yang, Z. He, et al. (2009). SNPHarvester: A filtering-based approach for detecting epistatic interactions in genome-wide association studies. *Bioinformatics*, **25**(4), 504–511. 4, 38, 39, 42

C. Yang, X. Wan, et al. (2010). Identifying main effects and epistatic interactions from large-scale SNP data via adaptive group lasso. *BMC Bioinformatics*, **11**(Suppl 1), S18. 59

M. Yeager, N. Orr, et al. (2007). Genome-wide association study of prostate cancer identifies a second risk locus at 8q24. *Nat Genet*, **39**(5), 645–649. 29

L. S. Yung, C. Yang, et al. (2011). GBOOST: A GPU-based tool for detecting gene-gene interactions in genome-wide case control studies. *Bioinformatics*, **27**(9), 1309–1310. 44

F. Zhang, X. Guo, et al. (2011). Multilocus association testing of quantitative traits based on partial least-squares analysis. *PLoS ONE*, **6**(2), e16739. 26

X. Zhang, S. Huang, et al. (2010). TEAM: Efficient two-locus epistasis tests in human genome-wide association study. *Bioinformatics*, **26**(12), 217–227. 38, 39, 44, 59

Y. Zhang, J. S. Liu (2007). Bayesian inference of epistatic interactions in case-control studies. *Nat Genet*, **39**(9), 1167–1173. 39

K. Zhang, Z. S. Qin, et al. (2004). Haplotype block partitioning and tag SNP selection using genotype data and their applications to association studies. *Genome Res*, **14**, 908–916. 121

J. Zhao, Z. Chen (2012). A two-stage penalized logistic regression approach to case-control genome-wide association studies. *J Probab Stat*, **2012**(2012) 642403. 26, 59

M. Zhu, S. Zhao (2007). Candidate gene identification approach: Progress and challenges. *Int J of Biol Sci*, **3**(7), 420–427. 18