GENE-BASED ASSOCIATION TESTING OF
DICHOTOMOUS TRAITS USING GENERALIZED
FUNCTIONAL LINEAR MIXED MODELS FOR
FAMILY DATA

by

Yingda Jiang

Master of Biotechnology, University of Pennsylvania, 2008
Bachelor of Science, Fudan University, China, 2006

Submitted to the Graduate Faculty of
the Graduate School of Public Health in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2015
UNIVERSITY OF PITTSBURGH
GRADUATE SCHOOL OF PUBLIC HEALTH

This dissertation was presented

by

Yingda Jiang

It was defended on

July 27th, 2015

and approved by

Daniel E. Weeks, PhD, Professor, Departments of Human Genetics and Biostatistics, Graduate School of Public Health, University of Pittsburgh

Wei Chen, PhD, Assistant Professor, Division of Pulmonary Medicine, Allergy and Immunology, Department of Pediatrics, Children’s Hospital of Pittsburgh of UPMC

George C. Tseng, ScD, Professor, Department of Biostatistics, Graduate School of Public Health, University of Pittsburgh

Leming Zhou, PhD, DSc, Assistant Professor, Department of Health Information Management, School of Health and Rehabilitation Sciences, University of Pittsburgh

Dissertation Director: Daniel E. Weeks, PhD, Professor, Departments of Human Genetics and Biostatistics, Graduate School of Public Health, University of Pittsburgh
GENE-BASED ASSOCIATION TESTING OF DICHTOMOUS TRAITS USING GENERALIZED FUNCTIONAL LINEAR MIXED MODELS FOR FAMILY DATA

Yingda Jiang, PhD

University of Pittsburgh, 2015

ABSTRACT
Gene-based association testing with rare variants requires arbitrarily aggregating or collapsing the information of the rare variants in genes into a single measure. As genotyping data can be viewed as a realization of a stochastic process that varies along the chromosome, it is more natural to summarize the genetic information using the approaches of functional data analysis. In functional data analysis, discrete genotypes are fitted by a continuous curve by using a collection of smooth basis functions. Existing generalized functional linear models (FLM) have been developed for unrelated samples to test for association between a dichotomous trait and genetic variants in a gene. In most situations, these models have higher power than well-known kernel-based methods (SKAT and SKAT-O). Here we extend this approach to accommodate family-based data using the GLOGS (genome-wide logistic mixed model/score test) approach developed by Stanhope and Abney, and develop family-based generalized functional linear mixed models (GFLMMs). This involves parallel computations to integrate out a multidimensional polygenic effect. Simulation results indicate that in most scenarios our new statistics are better than other similar statistics (famSKAT or F-SKAT), but not better than the retrospective kernel and burden statistics developed by Schaid and colleagues. We also embed FLM-smoothed genotypes in the retrospective statistics, improving the power of the kernel-based approach. We illustrate the behavior of these statistics by applying them to an age-related macular degeneration (AMD) family data set,
where, as expected, we observe strong association between AMD and $CFH$ and $ARMS2$, two known AMD susceptibility genes. Our proposed GFLMM provides a new tool for conducting family-based research studies in public health for complex or multifactorial diseases. The findings may improve the knowledge of existing AMD susceptibility genes and make a positive contribution to AMD treatment and prevention.

**Keywords:** Functional linear model (FLM), GWAS, AMD, Linkage, Association.
# TABLE OF CONTENTS

**PREFACE** ................................................................. xii

## 1.0 INTRODUCTION .................................................... 1

1.1 Research Aims ....................................................... 1

1.2 Motivation ........................................................... 2

1.3 Relevance to Public Health ........................................ 3

1.4 Existing Statistical Methodologies ................................. 4

1.4.1 Linkage analysis .................................................. 4

1.4.2 Association studies and retrospective regression techniques ... 6

1.5 Functional Linear Model ............................................ 9

1.6 A Brief Review of Age-related Macular Degeneration ............. 11

1.7 Exome Chip Arrays .................................................. 13

## 2.0 GENOME-WIDE LINKAGE ANALYSIS AND RETROSPECTIVE

REgression TESTS ......................................................... 15

2.1 Methods ............................................................... 15

2.1.1 Phenotypes ......................................................... 15

2.1.2 Sample Quality Investigation .................................. 16

2.1.3 Sample Identity and Relationship Examination ................. 16

2.1.4 Genotyping and Quality Control ................................ 17

2.1.5 Statistical Analyses .............................................. 18

2.1.5.1 Linkage analysis ............................................... 18

2.1.5.2 Gene-based association testing ............................... 19

2.2 Results and Conclusions ............................................ 19
3.0 GENERALIZED FUNCTIONAL LINEAR MIXED MODELS
(GFLMMS) IN FAMILY DATA ANALYSIS

3.1 Methods

3.1.1 GFLMM and Null Hypothesis for Testing

3.1.2 Likelihood Functions

3.1.3 Derivatives and Test Statistics

3.1.4 Computational Issues in Multidimensional Integration

3.1.5 Embed FLM-smoothed Genotypes in Retrospective Regression

3.1.6 Simulation Study

3.1.7 Real Data Analysis

3.2 Results and Conclusions

3.2.1 Simulation Results
## LIST OF TABLES

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Marker and sample failure and adjustment</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>LOD scores of top ten variants for single-point nonparametric linkage analysis</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>Suggestive linkage signals</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>Retrospective kernel-based approach and burden tests for a combination of common and rare variants</td>
<td>29</td>
</tr>
<tr>
<td>5</td>
<td>Single-variant association analysis for the common risk variants within \textit{CFH} and \textit{ARMS2}</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>Summary of the updates on the GLOGS program</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<td>Multidimensional integration in small samples</td>
<td>44</td>
</tr>
<tr>
<td>8</td>
<td>Summary of the notations in the plots</td>
<td>51</td>
</tr>
<tr>
<td>9</td>
<td>GFLMM and embedded approaches for a combination of common and rare variants</td>
<td>67</td>
</tr>
<tr>
<td>10</td>
<td>GFLMM and embedded approaches for rare variants</td>
<td>68</td>
</tr>
<tr>
<td>11</td>
<td>Single-variant association analysis for the rare risk variants within \textit{CFH}</td>
<td>69</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

1. Linkage signals identified on chromosome 6 ........................................... 23
2. Linkage region identified on chromosome 6 with support interval ............ 24
3. Multi-point LOD scores by chromosomes 1-6 ........................................... 25
4. Multi-point LOD scores by chromosomes 7-12 ....................................... 26
5. Multi-point LOD scores by chromosomes 13-18 ..................................... 27
6. Multi-point LOD scores by chromosomes 19-22 and X-chromosome .......... 28
7. Q-Q plots for the retrospective kernel and burden tests with a combination of common and rare variants ............................................................... 31
8. Q-Q plots for the retrospective kernel and burden tests with rare variants . 32
9. Type I error rates of rare risk variants (α = 0.01, 40% risk variants) ........... 54
10. Type I error rates of rare risk variants (α = 0.01, a re-sampled set of 40% risk variants) .............................................................. 55
11. Type I error rates of rare risk variants (α = 0.01, 30% risk variants) ........... 56
12. Type I error rates of a combination of common and rare risk variants (α = 0.01, 40% risk variants) .................................................. 57
13. Type I error rates of a combination of common and rare risk variants (α = 0.01, a re-sampled set of 40% risk variants) ................................ 58
14. Type I error rates of a combination of common and rare risk variants (α = 0.01, 30% risk variants) .................................................. 59
15. Power levels of rare risk variants (α = 0.01, 40% risk variants) ............... 62
16. Power levels of rare risk variants (α = 0.01, a distinct set of 40% risk variants). 63
17. Power levels of rare risk variants (α = 0.01, 30% risk variants) ............... 64
18  Q-Q plots of the GFLMM with the B-Spline (flmBS) and the Fourier basis
     (flmFR) with a combination of common and rare variants  . . . . . . . . . . . 70
19  Q-Q plots of the embedded kernel-based approach (flmBS_kernel) and burden
     tests (flmBS_burden) with a combination of common and rare variants  . . . 71
20  Q-Q plots of the GFLMM with the B-Spline (flmBS) and the Fourier basis
     (flmFR) with only rare variants  . . . . . . . . . . . . . . . . . . . . . . . . . . 72
21  Q-Q plots of the embedded kernel-based approach (flmBS_kernel) and burden
     tests (flmBS_burden) with only rare variants  . . . . . . . . . . . . . . . . . . . . . 73
A.1 Type I error rates of rare risk variants (\( \alpha = 0.05 \), 40\% risk variants)  . . . 83
A.2 Type I error rates of rare risk variants (\( \alpha = 0.05 \), a re-sampled set of 40\% risk
     variants)  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
A.3 Type I error rates of rare risk variants (\( \alpha = 0.05 \), 30\% risk variants)  . . . 85
A.4 Type I error rates of a combination of common and rare risk variants (\( \alpha = 0.05,
     40\% risk variants)  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
A.5 Type I error rates of a combination of common and rare risk variants (\( \alpha = 0.05,
     a re-sampled set of 40\% risk variants)  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
A.6 Type I error rates of a combination of common and rare risk variants (\( \alpha = 0.05,
     30\% risk variants)  . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
A.7 Power levels of rare risk variants (\( \alpha = 0.05 \), 40\% risk variants)  . . . . . . . 89
A.8 Power levels of rare risk variants (\( \alpha = 0.05 \), a re-sampled set of 40\% risk variants) 90
A.9 Power levels of rare risk variants (\( \alpha = 0.05 \), 30\% risk variants)  . . . . . . . 91
PREFACE

The research related to this dissertation was supported by the AMD grant, and was performed within the Departments of Biostatistics and Human Genetics, Graduate School of Public Health, University of Pittsburgh.

I would like to express my sincere thanks to Dr. Daniel E. Weeks, my advisor and committee chairperson, not only for his advising me on this Ph.D. dissertation, but also for his guidance during the past four years in my personality, scientific attitude, and career path, which I believe will be of benefit to my lifetime.

I would like to thank all my committee members, Dr. Wei Chen for his help on building the model and constructing the statistics, Dr. George C. Tseng for his suggestions on how to write a manuscript and present my work, and Dr. Leming Zhou for his advice on defining gene boundaries and validating results.

My grateful thanks also go to Dr. Ruzong Fan for his suggestions and comments on the GFLMM approach and, his R code for the beta-smooth only model.

I sincerely appreciate Dr. Qi Yan’s generating and willing to share the haplotype data for our simulation study.

I also gratefully acknowledge Drs. Yvette P. Conley and Michael B. Gorin for generating the AMD exome-chip data for our real data analysis work.

Finally, I would like to give very special thanks to my parents for understanding and supporting my decision to seek international studies at the University of Pittsburgh.
1.0 INTRODUCTION

It is not uncommon nowadays to perform a genome-wide association study (GWAS) in public health research. Researchers have generated reams of genotype data to study inherited diseases. Biostatisticians, not even enjoying a twinkle of delight in data wealth and availability, are bearing more responsibilities to develop novel methodologies to better characterize and model data structures. Challenging problems may not be just a limit to theoretical analysis of those proposed methods per se. Computational feasibility poses another inevitable hurdle to high-dimensional genotype data sets including large known pedigree structures and numerous genetic variants. Seeking promising solutions to these issues calls for time and effort put in statistical genetics. Throughout the following work, we expect to develop and present new promising statistics for analyzing dichotomous genetic traits with family-based genotype data.

1.1 RESEARCH AIMS

Our research study centered on two main aims. First, we applied existing statistical methodologies to exome chip data with family relatedness to identify age-related macular degeneration (AMD) related susceptibility loci by genome-wide linkage analysis and retrospective regression. Through these analysis procedures, novel risk variants or genes reaching genome-wide significance were reported and compared with both those already published in previous studies and the ones analyzed by our proposed methods. Second, we expounded the idea of the functional data analysis by demonstrating how to build family-based generalized functional linear mixed models (GFLMMs), and apply the new models to the analysis of
pedigree data. Our newly introduced methods also included the ones where we embedded FLM-smoothed genotypes in the retrospective kernel-based and burden tests. We then applied our newly-developed statistics to real exome chip data with known pedigree structures to identify AMD susceptibility genes.

1.2 MOTIVATION

Rare genetic variants with a minor allele frequency (MAF) less than 0.05 have triggered more research interests nowadays in testing association between a disease trait and contributive genes or loci [Lee et al., 2014]. However, the data sparsity of genetic variants set a stumbling block in the construction of the gene-based statistics. Situations may become even more complicated when we are wrestling with dichotomous traits and pedigree data. The past few years have witnessed the development of kernel-based approaches and burden tests broadly used in the rare variant analysis [Wu et al., 2011, Lee et al., 2012, Ionita-Laza et al., 2013]. Despite the emergence of methods that aggregate or collapse the rare variants, these methods, however, fail to sufficiently utilize genetic position and high-order linkage disequilibrium information. In 2013 and 2014, by using the techniques of functional data analysis, Fan et al. built functional regression models to test the association between a disease trait and multiple genetic variants [Fan et al., 2013, Fan et al., 2014]. The proposed approach yielded clues to the appropriate adjustment of high-order linkage disequilibrium by involving position information about genetic variants in the association analysis. Since their studies are focused on unrelated individuals, it brings about a natural question: Can we develop a family-base generalized functional linear model (FLM), and extend the promising approach to pedigree data analysis?

In this work, we were motivated by Fan’s earlier publications [Fan et al., 2014] and enhanced the idea to further extend the generalized FLM to dichotomous traits for related individuals within known pedigrees. More specifically, to adjust pedigree relatedness, we added an unobserved polygenic term into the Fan’s original beta-smooth only model. After applying functional weightings, we reduced the dimension by smoothing genotypes. To con-
struct the score statistics in the presence of nuisance parameters, we had to integrate out the polygenic term and maximized the likelihood. A numerical solution conductible to the challenging multidimensional integration in our study was propelled by the idea of tractability via parallel processing. A previously discussed mixed model-based system, the genome-wide LOGistic mixed model/Score test (GLOGS) program [Stanhope and Abney, 2012], partly met our requirement, and was modified and implemented for our parameter estimation.

Besides the elaboration on the research aims and motivation mentioned above, the following sections of this chapter will give an overall review on the existing methods for rare variant analysis in related individuals, introduce the concept of the FLM approach, and briefly discuss how the model works. In addition, since we also exploited the real exome chip data related to AMD in the UCLA/Pitt family-based study, sections will be reserved to discuss the exome chip and the genetics of AMD.

1.3 RELEVANCE TO PUBLIC HEALTH

Many research studies in public health genotype collect related individuals and pedigree structure information to investigate inherited complex diseases [Ott et al., 2011]. How to build a powerful mathematical model to better handle correlated data remains a problem in statistical genetics. In our study, we built the GFLMM to handle family data by adding an unobserved polygenic term into the existing generalized FLM approach to adjust pedigree relatedness. Our proposed model provides a new tool to conduct family-based research studies in public health with a concentration on the analysis of dichotomous disease traits. Furthermore, we applied our proposed GLFMM to the real exome chip data related to AMD, and the findings may improve the knowledge of existing AMD susceptibility genes and make a positive contribution to the etiology of AMD. We also suggested some promising findings which give clues to future studies in public health for AMD treatment and prevention.
1.4 EXISTING STATISTICAL METHODOLOGIES

1.4.1 Linkage analysis

Linkage, a term describing the coinheritance of alleles in close proximity during meiosis, constitutes one of the most consequential concepts in genome science [Lobo and Shaw, 2008]. The aim of genetic linkage analysis is to determine and quantitatively measure how risk alleles within a chromosome region cosegregate with a disease trait in a given pedigree [Ott et al., 2015]. As a traditional and once predominant tool, the backbone idea behind it comes from Mendelian recombination mapping and has now been enriched by advanced and sophisticated statistical methods. Compared with the association studies later discussed in Section 1.4.2, linkage analysis talks about using physically close genetic markers on a chromosome to investigate the tendency of coinheritance, and determine the position of disease susceptibility genes and how likely they are inherited together [Pulst, 1999]. Linkage occurs to two loci if the recombination fraction between them is less than 0.5. This kind of relation can be quantitatively constructed and expressed by a likelihood ratio from a statistical view of point. For parametric linkage analysis, the most popular and widely discussed approach is the logarithm of the odds (LOD) score method. Denoted by $Z$ by convention, LOD score is a parametric measure of the likelihood ratio [Morton, 1955], which is defined by

$$Z = \log_{10} \frac{L(\theta)}{L(\theta = 0.5)},$$

where $L$ denotes the likelihood function; $\theta$ is the recombinant fraction, the ratio of the number of recombinant gametes to the total number of gametes. For a human nuclear family consisting of father, mother, and their children, Equation 1.1 can be estimated by

$$Z = \log_{10} \frac{(1 - \theta)^{NR} \theta^R}{0.5^{NR+R}},$$

where $NR$ and $R$ represent the number of non-recombinant and recombinant offspring in the pedigree, respectively.

To conquer the computational difficulties related to parametric linkage analysis, several efficient algorithms have been broadly discussed to explore underlying genetic linkage during
the past decades [Kruglyak et al., 1995, Weeks and Lathrop, 1995, Kruglyak et al., 1996]. The Elston-Stewart algorithm and the Lander-Green algorithm are the two notable ones worth mentioning. Both of the two algorithms compute exact likelihoods for pedigrees. The Elston-Stewart algorithm estimates the joint probability of phenotypes for all individuals via conditional probabilities of genotype data. The computation tends to be time-consuming under a large number of variants for investigation [Elston and Stewart, 1971, Ziegler et al., 2010]. The Lander-Green algorithm establishes transitional probabilities for identity by descent (IBD) at arbitrary chromosomal locations, and estimates the likelihood conditional on IBD sharing under hidden Markov models. Its computational burden grows exponentially with the number of individuals for investigation [Lander and Green, 1987, Ziegler et al., 2010]. A notable property about the modern Lander-Green algorithm is that it is compatible with nonparametric linkage analysis. As a matter of factor, parametric linkage analysis relies heavily on a correct specification of a predefined genetic model [Clerget-Darpoux et al., 1986]. For most multifactorial diseases, it is unlikely to specify a priori true inheritance model, which may lead to invalid estimation of LOD scores. In contrast to model-based linkage analysis, nonparametric, or model-free linkage analysis estimates IBD, an allele-sharing statistic, at a locus between pairwise relatives to test whether an observed value is greater than expected [Kruglyak et al., 1996]. To meet the requirement of analyzing large numbers of single variants, a rapid computer program MERLIN was developed by Abecasis et al. through analyzing gene patterns in a binary tree with its nodes and branches representing meioses and gene transmissions, respectively [Abecasis et al., 2002]. The program is built on sparse binary trees instead of huge likelihood arrays in the traditional algorithms and the Idury-Elston algorithm, the one with the improvements on the Lander-Green algorithm in terms of simplifying transitional matrices and using founder reduction [Idury and Elston, 1997]. Via using symmetry to combine identical gene flow patterns, Abecasis et al. laid out demonstrative arguments that the MERLIN program outperforms Genehunter or Allegro in timing and marker handling capacity [Abecasis et al., 2002]. In view of this, for our applied study in Chapter 2, we take advantage of the efficiency of the MERLIN program in handling large number of markers genotyped for pedigrees, and conduct nonparametric linkage analysis to investigate any intriguing linkage signals.
Although linkage analysis may be complained about its power and high requirement in collecting large pedigrees enriched by affected cases, it does not lose its allure in locating the chromosome segments that carry disease susceptibility genes [Pulst, 1999]. Genomewide linkage study has been serving as a successful paradigm for identifying risk variants in Mendelian genetic disorders. To name a few for AMD for instance, at the beginning of this century, several AMD candidate genes were examined via targeted genotyping of genetic markers, which played as a prelude to genome-wide linkage analysis [Weeks et al., 2000]. The signals discovered on chromosomes 1q and 10q revealed the most two statistically significant regions amongst the early AMD studies [Klein et al., 2005, Jakobsdottir et al., 2005, Rivera et al., 2005]. These evidence provide us with genuine and reliable hits for comparison to guide later linkage or association studies in later investigations and discussions.

1.4.2 Association studies and retrospective regression techniques

It is a matter of fact that powerful genetic linkage mapping requires collecting data from related individuals and investigating the cosegregation of genetic variants and trait of interest. However, not all of the linkage studies end up with exciting results. Biostatisticians are acutely aware of some disadvantages or restrictions in terms of limited sample size to detect genes of modest effect size, low power, result interpretation, and genotype errors [Dawn Teare and Barrett, 2005]. In the purpose of gathering and analyzing more abundant population-based family data, biostatisticians turn their attention to association studies. The involvement of the unrelated individuals makes large samples possible to increase study power and investigate those genes with modest effect size. Unlike linkage analysis, association studies compare allele frequencies between disease cases and controls [Cordell and Clayton, 2005]. In drawing the inference on significant association signals, statistical analyses play a pivotal role. Traditional analysis methodologies put more weights on single variants via capturing common genetic variants independently based on the common disease common variant hypothesis [Gibson, 2011]. The restriction of these approaches lies in the ignorance of the influence of those rare variant with modest effect size on a disease trait.
With the advanced genotyping technologies gradually stepping into the mainstream, large-scale genotypic data are becoming accessible. A rise in GWASs has been found since 2005 [Hirschhorn and Daly, 2005], meanwhile facilitating the necessity in the analysis of numerous functional genetic variants with rare allele frequencies less than 0.05 [Gibson, 2011, Visscher et al., 2012]. Due to data sparsity, rare variants have to be combined into groups for association analyses, which may marginalize the outdated statistical methodologies only aimed at common variants [Lee et al., 2014]. To solve the issue by establishing the grouping, biostatisticians have been seeking the path to the identification of disease susceptibility genes by developing several modern statistical approaches. These strategies generally encompass kernel-based approach and burden tests [Drichel et al., 2014]. The main idea with kernel-based approach is to aggregate the association between genetic variants and disease traits via a kernel matrix with the structure specifying the pairwise genetic similarity between two individuals while treating covariates as fixed factors and genetic effect random factors [Lee et al., 2014]. In those early studies, the kernel-based approach was developed for quantitative traits, and assumed uncorrelated residuals. To extend the approach for pedigree data, Schifano et al. and Chen et al. introduced into their modeling scheme a random term to recover pedigree relatedness [Schifano et al., 2012, Chen et al., 2013]. More specifically, under the null hypothesis, they added in a variance component with random variation assumed to follow a multivariate normal distribution with mean 0 and variance-covariance matrix determined by inbreeding or kinship coefficients. Both sequence kernel association test (SKAT) and Schaids multiple genetic variant associating testing in 2013 served as two paradigm cases for population- and pedigree-based studies, respectively [Ionita-Laza et al., 2013, Schaid et al., 2013]. Compared to kernel-based approach, burden tests use a different strategy to create a variant-sum of burden as statistics for the variants within a genetic region [Lee et al., 2014]. Through the summation, rare variants are collapsed and represented by a single statistic, and can be used to test the association between a disease trait in a case-control study [Asimit and Zeggini, 2010]. Since the key idea behind burden tests is to collapse and cluster all rare variants into a single variable, a caution should be exercised that the variants collapsed by the variant-sum are assumed to share the same direction [Basu and Pan, 2011]. For burden tests, on the positive side, they facilitate the
analysis of rare variants with a low MAF by conquering the hurdle of data sparsity. On the negative side, however, the association may be neutralized with collapsing both risk and protective variants, resulting in of less power than kernel-based approach with the existence of a mixture of risk and protective variants within a gene region.

Although kernel-based approach and burden tests pave the way to the analysis of rare variants, a pivotal issue may lie in the non-random ascertainment of pedigree selection according to disease traits. Researchers tend to include more cases in pedigree selection process to assure an adequate power, which may introduce selection bias into the statistics for inference. Retrospective regression techniques find an unconventional approach to nicely get around the ascertainment sampling issue [Schaid et al., 2013]. As a consequential contributor to rare variant analysis for pedigree data, retrospective regression considers genotypes random factors, and regress them on a fixed disease trait. The kernel-based statistic proposed by Schaid et al., with retrospective regression behind it, takes a quadratic form \( Q = (Y - \hat{Y})^T H (Y - \hat{Y}) \), where \( Y - \hat{Y} \) represents the residuals of disease traits, and \( H \) is a positive-semidefinite kernel matrix estimated under a weighted linear kernel by \( H = G W W^T G^T \), where \( G \) is a genotype matrix denoting the copy numbers of minor alleles and \( W \) is a diagonal matrix with its elements specifying the weights for each corresponding variant [Schaid et al., 2013]. For the burden statistic with retrospective regression, Schaid et al. extended the single-variant based statistic initially suggested by Thornton et al. [Thornton and McPeek, 2010] to a gene-based burden one, and described on how to facilitate the estimation by using maximum likelihood estimates of expected IBD (EIBD) sharing.

In addition to resolving the ascertainment issue, the efficiency of conducting the retrospective regression is of another advantage. The well-built “pedgene” package (http://cran.r-project.org/web/packages/pedgene/index.html) in R [R Core Team, 2014] makes it possible for scanning whole genome within a reasonable time period. In Chapter 2, we will apply the retrospective kernel and burden tests to analyzing the real exome chip data to identify AMD risk genes. Furthermore, we will also discuss in later Chapter 3 on how to embed the FLM approach in this promising framework.
1.5 FUNCTIONAL LINEAR MODEL

As mentioned in preceding sections, kernel-based approach and burden tests have been developed during the past few years to overcome the difficulties of examining a single rare variant. Combined with retrospective regression techniques, these methods free themselves from modeling ascertainment process, and become more advanced and sophisticated. However, although these existing approaches have epitomized the paradigms in rare variants analysis, it is noteworthy that none of them employs the position information of genetic variants. This may bring about a potential weakness in a GWAS, that is, higher-order linkage disequilibrium fails to be taken into account [Slatkin, 2008]. An ensuing issue may be enlarged by the fact that the design of a modern GWAS nowadays is characterized by testing a high-density genetic markers or probes in a candidate region [Roberts et al., 2010], which would finally result in ambiguous association signals concerning whether a genetic variant is truly related to a trait or it is a false positive due to a non-random association. A direct perceived question is: Is there a better statistical methodology that enables the analysis of rare variants while appropriately accounting for high-order linkage disequilibrium? Such an ideal statistic is expected to incorporate variant positions the information which is always collected or accessible from web-based databases during GWASs. FLMs may hold the key to address this issue.

Functional data analysis originates in time series data modeling [Ullah and Finch, 2013]. The core content of this approach is to treat longitudinal measurements as a continuous curve [Müller and Stadtmüller, 2005]. Since in a GWAS genetic markers are tightly genotyped within a region of interest on a chromosome, such an idea can be borrowed to model the association between a trait and genetic variants by expanding predictor functions under some techniques to smooth and reduce the dimension of discrete data in a concentrated manner. Such sophisticated techniques previously discussed included the application of a series of orthogonal basis functions as smoothers and the approximation of the Karhunen-Loéven expansion [Fan et al., 2014, Karhunen, 1947, Loeve, 1977]. Particular for a gene-based association study, Fan et al. presented an approach to revise the original FLM by taking a direct usage of genotype data with the purpose to estimate the genetic effect functions.
In 2014, Fan’s group extended their method to a more general form to handle a dichotomous trait [Fan et al., 2014]. These pioneer and influential studies, by focusing on population data with unrelated individuals, enable the analysis of rare variants or common variants, or a combination of the two. Specifically in mathematical expressions, given a disease trait and observed genotype data, the generalized FLM can be defined as

\[ Y = g \left( \alpha + \int_0^1 X(t)\beta(t) \, dt \right) + \epsilon, \]  

where \( Y \) is a vector denoting a univariate dependent variable, either continuous or discrete, following a distribution within the exponential family; \( g(.) \) is an appropriate link function; \( \alpha \) is a constant vector; \( X(t) \) contains genotype data observed at a position \( t \); \( \beta(t) \) is a parameter function vector to be estimated for statistical inference; \( \epsilon \) denotes the vector of error terms [Müller and Stadtmüller, 2005]. For a dichotomous trait under a logistic regression framework, the model can be specified as

\[ p_i = \logit^{-1} \left( Z_i^T \alpha + \int_0^1 X_i(t)\beta(t) \, dt \right), \]  

where \( Z \) is the underlying covariate matrix, and \( p_i = P(Y_i = 1) \), the probability of disease.

Under a generalized FLM, discrete genotypes are treated as a representing function to be modeled. In practice of identifying the disease associated genes, the idea is no longer generating a kernel matrix or collapsing the rare variants in a biological region, but viewing genotypes as a stochastic process that varies along the chromosome [Fan et al., 2013, Ross, 1996]. The necessity to involve functional data analysis rests with the natural functionality in the objectives of analysis and statistical modeling process itself [Ramsay and Dalzell, 1991]. Particularly for genotype data, given that position intervals for genetic variants may vary, functional data analysis may outperform any other existing approach in modeling this unbalance. Another advantage of treating discrete genotypes as functional data is the competence to adjust high-order linkage disequilibrium (not just limited to pairwise linkage disequilibrium between adjacent variants), for it does not necessarily assume that the genotypes of a certain individual are independent of each other [Ullah and Finch, 2013], and all position information within a gene boundary is sufficiently used in smoothing genotype data.
According to the studies led by Fan et al., the generalized FLM was developed for modeling a disease trait in a population-based sample [Fan et al., 2014], naturally motivating an extension to a sample with relatedness. In Chapter 3 of this dissertation, we assimilate the functional data analysis into the generalized linear mixed model framework developed by Papachristou et al. [Papachristou et al., 2011] to better handle the higher-order linkage disequilibrium for sufficiently close genetic variants. Our proposed statistics integrate the beta-smooth only model propounded by Fan et al. [Fan et al., 2013], and are adapted to a dichotomous trait in a family-based association study with the inclusion of a kinship matrix to account for pedigree relatedness. Furthermore, inspired by the unique character of Schaid’s approach [Schaid et al., 2013] in modeling ascertainment mentioned in preceding Section 1.4.2, we embed our modeling strategy in retrospective regression to solve the potential ascertainment issue in our samples.

1.6 A BRIEF REVIEW OF AGE-RELATED MACULAR DEGENERATION

To evaluate the performance of our proposed model in reality, we exploits real exome chip data in the UCLA/Pitt family-based study. Besides assessing the performance, we also aim to draw a clearer picture of the genetic contributions to AMD, address uncertainty remained in the susceptibility variants previously identified but still open to debate, and shed light on the statistical methodologies applicable to family-based studies in human genetics. In this section, we give a brief review of the genetics of AMD.

AMD, a progressive neurodegenerative disease, constitutes one of the primary causes of visual impairment and irreversible blindness in the elderly of western countries. The statistics in 2004 showed that a significant increase occurs in the estimated prevalence with an increase in age for individuals of European descent [Friedman et al., 2004]. In the United States, approximately 10 million of age 40 years or older developed intermediate or advanced AMD in the mid 2000s [Friedman et al., 2004]. For people older than 80 years old, more than 15% were afflicted with AMD, and the estimated percentage is expected to rise dramatically up to 65% by 2020 [Parmeggiani et al., 2012]. The soaring morbidity rate of AMD in the
developed nations, especially causing blindness or low vision amongst the people of age 40 years or older, makes AMD a spotlight in the research of complex disease.

Understanding of the clarification and the pathogenesis of a disease per se would contribute to the advances in disease treatment, control, and prevention. AMD is not an exception. AMD of early stages is associated with the presence of atrophy of the retinal pigment epithelium, thickening of the Bruch membrane, and the appearance of large drusen [Ambati et al., 2013]. In later stages, despite an existing overlap, AMD can be characterized into choroidal neovasculariation (CNV) and geographic atrophy (GA), and dichotomized in pathology based on the presence of or absence of CNV [Ambati and Fowler, 2012]. Advanced AMD is associated with the degeneration of retinal photoreceptor, or CNV and retinal pigmented epithelium cells [Ambati et al., 2013, Ambati et al., 2003]. Previous studies have led to supportive evidence that the molecular pathogenesis of AMD involves para-inflammation, complement system mediated pathways, and the responses of immune system [Ambati and Fowler, 2012].

It has been clearly reported that genetic factors account for approximately 40% to 71% of the variation of AMD susceptibility [Seddon et al., 2005, Montezuma et al., 2007, Seddon et al., 2009, Gorin, 2012]. Besides the unveiling of the biological mechanism of AMD, the genetic susceptibility to AMD has been broadly discussed through publications of linkage analysis and association studies, the strategies biostatisticians have already developed to identify susceptibility genetic variants [Swaroop et al., 2007, Fritsche et al., 2014].

Probably the association studies on AMD are those of the most conspicuous exemplifications in the field of statistical genetics [Katta et al., 2009]. The rapid rise in the findings of AMD susceptibility genes during the past decade has stemmed from not only the advancement in genotyping arrays but also the refinement of the statistical approaches. The identification of association between AMD and complement factor H \((CFH)\) in 2005 marked a prelude to the subsequent GWASs aimed at AMD [Klein et al., 2005, Haines et al., 2005, Edwards et al., 2005]. This revelation also prompted biostatisticians to turn their focus toward extensive GWASs in AMD and conclude more exciting discoveries. There is general agreement that \textit{ARMS2} on chromosome 10 [Jakobsdottir et al., 2005, Rivera et al., 2005], together with \textit{CFH} region on chromosome 1, is another consequential AMD susceptibility
region. The subsequent findings of $C2/CFB$ region on chromosome 6 [Gold et al., 2006] and $C3$ region on chromosome 19 [Seddon et al., 2013] not only confirmed those promising findings from earlier linkage studies, but also shed fresh light on the indications of novel genes/loci associated with AMD. In 2013, AMD Gene Consortium identified seven new loci reaching genome-wide significance near $COL8A1/FILIP1L$, $IER3/DDR1$, $SLC16A8$, $TGFBR1$, $RAD51B$, $ADAMTS9/MIR548A2$, and $B3GALTL$ by investigating more than 77,000 cases and controls [Fritsche et al., 2013]. A more recent study conducted by the International AMD Genomics Consortium has exemplified a broader analysis on a combination of common and rare variants, unveiling sixteen novel loci associated with advanced AMD trait (Fritsche et al., in preparation; submitted to Nat. Genet.). It cannot be denied that to date more and more AMD associated susceptibility genes/loci have been discovered. However, most of these robust findings are based on the studies with unrelated individuals under a population design, which may evoke further investigations into the AMD related genes in family-based studies.

In summary, although molecular and immunological basis of AMD has well been laid, like many other genetic disorders, the etiology of AMD is complex, and determined by multigenic susceptibility including the complement pathway, a region of chromosome 10, lipid metabolism, extracellular matrix remodeling, and angiogenesis [Fritsche et al., 2014, Black and Clark, 2015, Cascella et al., 2014, Horie-Inoue and Inoue, 2014]. Against such a backdrop, we consider it a positive contribution to public health by examining both common and rare variants across human genome in exome chip data genotyped from related individuals. In the following Chapters 2 and 3 we report risk genes robustly associated with AMD by applying retrospective regression techniques, GFLMMs, and an embedded method combining the two approaches.

### 1.7 EXOME CHIP ARRAYS

Although whole-genome sequencing is no longer a utopia in GWASs, it is still cost intensive and time consuming to obtain measurements for every locus in a clinical study particu-
larly with a large number of participants recruited [Visscher et al., 2012]. It is also widely acknowledged that most findings in GWASs with traditional genotyping arrays are those variants with relatively small effect size, thus not well explaining genetic variations in the heritability of a complex disease [Cirulli and Goldstein, 2010, Robinson et al., 2014]. To address the issues, researchers have shifted their interests to those rare variants, and consider to genotype with a large sample the markers with low MAF [Wagner, 2013]. An alternative genotyping strategy is therefore developed to mainly target the exomes in human genome by capturing functional coding regions [Wang et al., 2013]. Unlike traditional genotyping arrays accommodating common variants, the design of exome chip arrays enriches for relatively rare variants, low frequency coding variants, non-synonymous variants, and other protein altering variants, technically juggling the common variants in associations analyses and the rare variants in sequencing studies (Exome Chip Design - Genome Analysis Wiki: http://genome.sph.umich.edu/wiki/Exome_Chip_Design). Exome chip arrays can also be customized with the incorporation of redundant probes to target those variants of high interest for a certain disease trait. These design properties make it possible for researchers to generate dense genotype data in a genetic region of interest, when they desire to test the hypothesis that rare variants with large effect sizes may decisively account for missing heritability of complex diseases [Manolio et al., 2009].

For a complex disease like AMD, clear evidence has been shown that rare coding variants may have play a pivotal role in explaining the genetic variation [Seddon et al., 2013, Zhan et al., 2013]. With the extensive coverage of known AMD susceptibility loci, the real Genetics of Age-related Macular Degeneration (GARM) data generated from the exome chip arrays enable a fine capture of the rare variants associated with AMD, and stand a chance of finding de novo functional coding variants remaining to be identified in our studies. These potential novel variants tend to be the rare ones that might not be widely discussed in the previous publications.
2.0 GENOME-WIDE LINKAGE ANALYSIS AND RETROSPECTIVE REGRESSION TESTS

In this chapter, we first describe the pipeline for the real GARM data cleaning. Then we elaborate on the analysis procedures via genome-wide linkage analysis and retrospective kernel-based approach and burden tests.

2.1 METHODS

2.1.1 Phenotypes

We initially attempted to collect 1,379 genotyped individuals containing 506 extended families. Among these study samples, we subset the 1,070 genotyped individuals of European ancestry for our further investigations. The AMD trait we were phenotyping and modeling was called “C_65”. The “C” diagnostic scheme established an inclusive and stringent definition of AMD affected status previously discussed [Weeks et al., 2000; Weeks et al., 2004]. It defined and included the patients in the following three categories with

1. those who were clearly diagnosed as having AMD based on the evidence of extensive and/or coalescent drusen, pigmentary changes and/or the presence of GA and/or CNV;
2. those who were likely to be affected, with clear pathological changes in drusen and/or pigment epithelium; and
3. those who could not be considered AMD-free for lack of sufficient medical records, or with the presence of end-stage degeneration.
The number “65” referred to the fact that an individual was a normal control if he or she was at least 65 years old at the last exam, which could ensure it would be unlikely for the controls to develop AMD during their lifetime. The controls were those who had no indications of any macular or other retinal pigment epithelial changes according to eye-care diagnostics.

### 2.1.2 Sample Quality Investigation

To check and assess the quality of samples and genotypes (see Section 2.1.4), we applied the data cleaning pipeline built in the Gene Environment Association (GENEVA) studies project ([http://www.genome.gov/27550876](http://www.genome.gov/27550876)) and the methodologies previously discussed by Laurie et al. [Laurie et al., 2010].

For the 1,070 genotyped individuals of European ancestry, we carried out sample quality checks by dividing each chromosome into twelve “bins”, and running B allele frequency variance analysis [Alkan et al., 2011]. The anomalous chromosome-sample pairs were identified with high B allele standard deviation more than four standard deviations from the mean B allele frequency standard deviation over all samples. Males and females were computed separately for X-chromosome. We identified 24 chromosome-sample pairs with entire or partial chromosome duplication or deletions by counting the number of bins with high standard deviation exceeding nine for each pair and gender. To further determine if there were any chromosome segment deletions or duplication, we plotted the genotyping intensities against the chromosome positions. The chromosome segments of 29 additional samples were removed due to fuzzy and noisy intensities.

### 2.1.3 Sample Identity and Relationship Examination

To determine whether there existed any potential gender errors, we did a misannotation gender check on the discrepancy between the annotated and genetic genders. Gender-specific means of the Y-chromosome intensity and X-chromosome heterozygosity were plotted versus X-chromosome intensity, X-chromosome heterozygosity, and autosomal heterozygosity, respectively. One sample was excluded due to a zero value of X-chromosome heterozygosity.
We further explored the pedigree relationships in the PREST software (http://www.utstat.toronto.edu/sun/Software/Prest/prest3.02/index.html) by estimating pair-wise kinship coefficients and IBD sharing. We flagged any paired samples with the middle p-value of the estimates of EIBD, identical by state (IBS), and adjusted IBS (AIBS) < 0.005, and considered a violation of a certain relationship within a family. We also estimated IBD sharing by using the Method of Moments (MoM) approach in PLINK (http://pngu.mgh.harvard.edu/~purcell/plink). To make the exclusion reason more persuasive, we ranked the missing call rate for each sample involved in the potential problematic relationships. A sample with a missing call rate > 0.01 was flagged for further consideration. Filtering decisions came out based on the middle PREST p-values < 0.005, the PLINK IBD estimates, and the missing call rates that 26 samples were excluded due to relationship misspecifications with 5 samples stayed on the top list of the missing call rates. We also altered two family structures from full siblings pairs to half siblings ones due to relationship misspecification.

With respect to all of the unrelated paired samples, we further investigated the connectivity to examine whether or not an unrelated individual would be related to too many others. The connectivity coefficients were computed and ranked in descending order. The top two unrelated samples with the highest connecting relatedness were excluded.

### 2.1.4 Genotyping and Quality Control

With 247,870 variants released by the Center for Inherited Disease Research (CIDR) at Johns Hopkins University (http://www.cidr.jhmi.edu), we first filtered out 1,794 variants with no genotyping data released, and 487 variants with annotated genotyping errors. Since nonpolymorphic variants contributed uninformatively to linkage or association analyses, we excluded 130,872 nonpolymorphic ones before any further data cleaning steps. Due to study design issues, we figured out that 401 pairs of probes were actually detecting the same position on a certain chromosome. We therefore excluded additional 401 variants associated with a lower call frequency rate.

We then checked the quality of the sample batches by calculating *missing.nI*, missing call rates per variant over all individuals and *missing.eI*, missing call rates per individual over
all variants. Next, for all variants with $missing.e1 < 0.05$, we computed $missing.n2$ over all individuals. Accordingly, for all variants with $missing.n2 < 0.05$, we computed $missing.e2$ per individual over all variants. The duplicate genotypes of 12 blind samples were excluded per a higher $missing.n2$, and in total 57 variants were dropped due to $missing.n2 > 0.05$.

Hardy-Weinberg equilibrium (HWE) may have an impact on the interpretation of linkage and association signals. To identify any variants with significant deviations of HWE, we ran the PEDSTATS software package (http://csg.sph.umich.edu/abecasis/Pedstats/index.html) to test each variant that passed the quality control. We flagged 90 variants with the p-values of the HWE test $< 10^{-7}$.

2.1.5 Statistical Analyses

We did linkage analysis and association testing for statistical analyses. In our analyses, genotypes were coded as 0, 1, and 2 to count the copy numbers of minor alleles for a certain genetic variant. Missing genotypes were coded as “NA”. We used the Mega2 program (https://watson.hgen.pitt.edu/docs/mega2_html/mega2.html) to convert our data into the proper formats required by different computing programs [Baron et al., 2014].

2.1.5.1 Linkage analysis We examined autosomal chromosomes for the underlying non-parametric linkage signals by using the MERLIN software package (http://csg.sph.umich.edu/abecasis/Merlin/index.html). Missing genotypes were retained in the data for the linkage analysis. To avoid time-consuming computations on large pedigrees, we subjected to the PedStr program (http://mga.bionet.nsc.ru/soft/index.html) two pedigrees with bit size exceeding 24 to reduce the family size, and split them into four sub-pedigrees [Kirichenko et al., 2009]. To detect the underlying linkage signal for each single variant, we carried out single-point linkage analysis by treating each variant independently and assuming HWE. With linkage disequilibrium modeled, multiple-point nonparametric linkage analysis was done with the exclusion of the variants violating HWE.

For the variants on X-chromosome, we ran the linkage analysis using MINX (MERLIN in X), a program built in the MERLIN package. We also figured out pseudoautosomal regions
(PARs) on X- (60,001-2,699,520 and 154,931,044-15,526,060) and Y-chromosome (10,001-2,649,520 and 59,034,050-59,363,566) (Human genome overview - Genome Reference Consortium: http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human). Since the genes within PARs behave like the ones on autosomes during meiosis, we analyzed them by MERLIN as we did for those autosomal variants.

2.1.5.2 Gene-based association testing We ran multiple genetic variant testing by computing kernel association statistic and burden statistic based on the methodologies proposed by Schaid et al. [Schaid et al., 2013]. To implement this, we pulled gene boundary information from the UCSC Genome Browser (https://genome.ucsc.edu), and used the “pedgene” package in R to compute the gene-based kernel and burden statistics. Gene boundaries were determined according to the reference assembly GRCH37. We did not include any variants in the upstream or downstream sequences. We took into account different transcripts of the same gene. The genes containing a single variant were excluded from the analyses in this section. For each variant, we applied the Madsen-Browning weights [Madsen and Browning, 2009], which were based on a function of MAF estimated by $1/\sqrt{\text{MAF}} \times (1 - \text{MAF})$.

Missing values constitute a practical issue in real data analysis. Although the “pedgene” package claimed the capability of finely handling missing genotypes, we recognized that out of an unbalanced design the missing genotypes would cause the inflation of p-values. There was a concern that it might be time consuming to impute the genotypes in a family-based study with a large number of genetic markers. Alternatively, we replaced the missing genotypes for a certain variant with a number sampled from \{0, 1, 2\}, the associated probability of which was determined by the non-missing genotype frequency.

2.2 RESULTS AND CONCLUSIONS

A total of 976 genotyped individuals of European ancestry passed the quality control procedure. To completely connect pedigrees, we had to create dummy parents and included
non-genotyped individuals who shared the same family with those 976 individuals. Our final cleaned sample data contained 2,727 samples, 1,275 with a known AMD trait (1031 cases and 244 controls). A total number of 111,547 variants were included in our study. Among them, 26,359 were common variants (MAF ≥ 0.05), and 85,188 were rare ones (MAF < 0.05). Full details of data cleaning were summarized in Table 1.

Table 1: Marker and sample failure and adjustment

<table>
<thead>
<tr>
<th>Marker filter criteria</th>
<th>Marker lost</th>
<th>Sample filter criteria</th>
<th>Sample lost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failed by CIDR (no genotypes)</td>
<td>1,794</td>
<td>Anomalous chromosomes</td>
<td>29</td>
</tr>
<tr>
<td>Annotated genotyping error</td>
<td>487</td>
<td>Gender misannotation</td>
<td>1</td>
</tr>
<tr>
<td>Nonpolymorphic variants</td>
<td>130,872</td>
<td>Relationship misspecification</td>
<td>26</td>
</tr>
<tr>
<td>Repeated measurement</td>
<td>401</td>
<td>High connecting relatedness</td>
<td>2</td>
</tr>
<tr>
<td>Missing call rates &gt; 5%</td>
<td>57</td>
<td>Fuzzy sample-chromosome pairs</td>
<td>24</td>
</tr>
<tr>
<td>Bad probes identified</td>
<td>106</td>
<td>Duplicated samples</td>
<td>12</td>
</tr>
<tr>
<td>Departure from HWE</td>
<td>90</td>
<td>Pedigree structure alteration ‡</td>
<td>2</td>
</tr>
<tr>
<td>Markers not on autosomes †</td>
<td>4,310</td>
<td>Markers remained in the study</td>
<td>111,547</td>
</tr>
<tr>
<td>Samples remained in the study</td>
<td></td>
<td>Samples remained in the study</td>
<td>976</td>
</tr>
</tbody>
</table>

† Markers on X-chromosome were analyzed by MINX for linkage signals.
‡ These two individuals with altered structures were kept in the analysis.

2.2.1 Linkage Signals

We performed single- and multi-point nonparametric linkage analysis in MERLIN. To model linkage disequilibrium, we set the threshold of pairwise correlation $r^2$ to be 0.10, such that the variants for which $r^2 > 0.10$ were clustered to be analyzed.

For single-variant linkage signals, we presented in Table 2 the top ten hits with the highest LOD scores. As a traditional rule, LOD = 3.0 is used as a genome-wide linkage significance threshold [Churchill and Doerge, 1994]. For our single-variant signals, rs2308655 within HLA-B on chromosome 6 had the highest single-point LOD score (2.922). Compared with the threshold of LOD = 3.0, this linkage signal failed to reach genome-wide significance. However, we still considered it a suggestive single-variant signal given that the association of AMD with HLA-A and HLA-B were previously discussed [Goverdhan et al., 2005]. In addition, we identified another promising linkage signal (LOD = 2.714) on rs86567 within
HLA-DOA related to HLA classes. HLA-DOA belongs to HLA class II alpha chain paralogues, and is also located on chromosome 6, close to HLA-B, which might suggest an interesting AMD linkage region. Besides HLA classes, we also observed rs3124299 on chromosome 9 with LOD = 2.709. The variant is within COL5A1, a gene significantly associated with central cornea thickness in European cohorts [Vitart et al., 2010]. In Table 2 we listed corresponding multi-point LOD scores for a comparison purpose. None of them reached a genome-wide linkage significance threshold.

Table 2: LOD scores of top ten variants for single-point nonparametric linkage analysis

<table>
<thead>
<tr>
<th>variant</th>
<th>gene</th>
<th>chr</th>
<th>position (bp)</th>
<th>major/minor allele</th>
<th>MAF</th>
<th>single-point LOD score</th>
<th>multi-point LOD score</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs2308655</td>
<td>HLA-B</td>
<td>6</td>
<td>31322303</td>
<td>[G/C]</td>
<td>0.4482</td>
<td>2.922</td>
<td>2.346</td>
</tr>
<tr>
<td>rs704124</td>
<td>C12orf12</td>
<td>12</td>
<td>90826178</td>
<td>[A/G]</td>
<td>0.2735</td>
<td>2.831</td>
<td>1.345</td>
</tr>
<tr>
<td>rs3815768</td>
<td>ELL2</td>
<td>5</td>
<td>95236459</td>
<td>[A/G]</td>
<td>0.2559</td>
<td>2.792</td>
<td>0.260</td>
</tr>
<tr>
<td>rs10799445</td>
<td>ZNF678</td>
<td>1</td>
<td>227911883</td>
<td>[T/G]</td>
<td>0.2036</td>
<td>2.746</td>
<td>0.869</td>
</tr>
<tr>
<td>rs86567</td>
<td>HLA-DOA</td>
<td>6</td>
<td>32976759</td>
<td>[T/G]</td>
<td>0.4148</td>
<td>2.714</td>
<td>1.050</td>
</tr>
<tr>
<td>rs3124299</td>
<td>COL5A1</td>
<td>9</td>
<td>137619195</td>
<td>[T/C]</td>
<td>0.4281</td>
<td>2.709</td>
<td>0.127</td>
</tr>
<tr>
<td>rs4774</td>
<td>CIITA</td>
<td>16</td>
<td>11000848</td>
<td>[G/C]</td>
<td>0.2969</td>
<td>2.564</td>
<td>0.808</td>
</tr>
<tr>
<td>rs6941421</td>
<td>JARID2</td>
<td>6</td>
<td>15089151</td>
<td>[A/G]</td>
<td>0.4092</td>
<td>2.564</td>
<td>0.078</td>
</tr>
<tr>
<td>rs1387144</td>
<td>BDNF</td>
<td>11</td>
<td>27635319</td>
<td>[A/C]</td>
<td>0.4078</td>
<td>2.554</td>
<td>0.803</td>
</tr>
<tr>
<td>rs6913635</td>
<td>COL11A2</td>
<td>6</td>
<td>33118937</td>
<td>[A/C]</td>
<td>0.0579</td>
<td>2.467</td>
<td>1.605</td>
</tr>
</tbody>
</table>

Top ten variants for single-point nonparametric linkage analysis were listed. Corresponding reference alleles, MAF, and multi-point LOD scores were also tabulated for comparison.

For multi-point linkage signals, the most convincing evidence of linkage was observed on chromosome 6 with the highest LOD = 3.867. Even compared to a more stringent threshold LOD = 3.3 suggested by Lander and Kruglyak [Lander and Kruglyak, 1995], the linkage we suggested here still reached genome-wide significance. To clearly demonstrate the strongest linkage peaks, we plotted the LOD scores in Figure 1 for all of the variants on chromosome 6, and particularly zoomed in the target region between 30,000,000 and 33,000,000 bp. To superimpose the known AMD risk genes, we referred to the National Center for Biotechnology Information (NCBI, http://www.ncbi.nlm.nih.gov), and searched for all the genes related to AMD within the genome of Homo sapiens. We noted that the transcripts within this region included those from C2, CFB, SKIV2L, ATF6B, FKBPL, NOTCH4, STK19, and TNXB, all
of which were AMD risk genes, giving supportive evidence that the region on chromosome 6 might be significantly linked to AMD susceptibility. To better describe the linkage area, we computed the support interval threshold by considering 1 unit below the maximum LOD score, or $\text{LOD}_{\text{support}} = \text{LOD}_{\text{max}} - 1$ [Conneally et al., 1985, Edwards, 1992]. In our analysis, the largest LOD score was 3.867, which led to $\text{LOD}_{\text{support}} = 2.867$. We plotted in Figure 2 by zooming in the region that contained all the variants associated with a LOD score greater than the $\text{LOD}_{\text{support}}$. By doing this, we achieved a narrow and precise region on chromosome 6 between 31,773,821 and 32,439,508 bp. The genes within this region were aligned below the LOD score curve to illustrate the position of the transcripts related to the linkage peaks.

Besides chromosome 6, we also plotted the multi-point linkage signals for all the other autosomes and X-chromosome in Figures 3, 4, 5, and 6 with previously reported AMD risk genes superimposed. To describe the suggestive linkage signals, we added a more inclusive threshold of $\text{LOD} = 2.0$ to better illustrate some genetic regions of particular interest but failing to reach a genome-wide significance threshold of $\text{LOD} = 3.0$. We identified suggestive linkage regions on Chromosomes 5, 8, 9, and 12 with corresponding multi-point LOD scores exceeding 2.0. We presented more details in Table 3.

### Table 3: Suggestive linkage signals with multi-point LOD scores $\geq 2.0$, but $< 3.0$

<table>
<thead>
<tr>
<th>Chromosome</th>
<th>Start (bp)</th>
<th>End (bp)</th>
<th>Region</th>
<th>Peak LOD score</th>
<th>Nearby risk genes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>73,932,199</td>
<td>75,306,838</td>
<td>5p15.33</td>
<td>2.488</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>100,182,333</td>
<td>118,165,247</td>
<td>8p23.3-8p23.2</td>
<td>2.899</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>116,856,481</td>
<td>118,741,214</td>
<td>9p24.3</td>
<td>2.602</td>
<td>$\text{C5}, \text{TLR4}$</td>
</tr>
<tr>
<td>12</td>
<td>88,440,676</td>
<td>88,589,256</td>
<td>12p13.33</td>
<td>2.015</td>
<td></td>
</tr>
</tbody>
</table>

Although linkage signals on chromosomes 5, 8, 12 were discussed by Weeks et al. as early as in 2000 [Weeks et al., 2000], our results in Table 3 gave suggestive signals on these chromosomes. No AMD risk genes were previously reported within or close to these regions, which indicated that these signals might call for further investigations. For 9p14.3 reaching a suggestive threshold, we identified two AMD risk genes $\text{TLR4}$ and $\text{C5}$ close to it. The association between AMD and these two genes were previously discussed by Zareparsi et al. [Zareparsi et al., 2005] and Mullins et al. [Mullins et al., 2000], respectively, which strengthened the evidence of a promising linkage region we suggested on chromosome 9.
Figure 1: Linkage signals identified on chromosome 6. Linkage signals identified on chromosome 6 with a highest estimated LOD score of 3.867. The region (31,883,679-32,439,508) included the transcripts from C2, CFB, and SKIV2L, three AMD susceptibility genes. A threshold of LOD = 3.0 (black line) and the support interval threshold (red line drawn by 1 unit below LOD$_{max}$ = 3.876) were superimposed onto the plot for comparison. The upper plot illustrated the entire chromosome 6, while the lower plot zoomed in a linkage region of interest between 30,000,000 and 33,000,000 bp.
Figure 2: Linkage region identified on chromosome 6 with support interval. Linkage region identified on chromosome 6 with multi-point LOD scores > support interval threshold 2.876 (red line), 1 unit below LOD\textsubscript{max} = 3.876. The genes within the region (31,773,821-32,439,508) were aligned below the linkage peaks. A threshold of LOD = 3.0 (black line) was superimposed onto the plot for comparison.
Figure 3: Multi-point LOD scores by chromosomes 1-6. Multi-point LOD scores were plotted by chromosomes 1-6. The known AMD susceptibility genes were labeled by vertical dashed lines (darkgreen). A genome-wide threshold of LOD = 3.0 (black horizontal line) and a suggestive threshold of LOD = 2.0 (pink horizontal line) were superimposed onto the plot for comparison.
Figure 4: Multi-point LOD scores by chromosomes 7-12. Multi-point LOD scores were plotted by chromosomes 7-12. The known AMD susceptibility genes were labeled by vertical dashed lines (dark green). A genome-wide threshold of LOD = 3.0 (black horizontal line) and a suggestive threshold of LOD = 2.0 (pink horizontal line) were superimposed onto the plot for comparison.
Figure 5: Multi-point LOD scores by chromosomes 13-18. Multi-point LOD scores were plotted by chromosomes 13-18. The known AMD susceptibility genes were labeled by vertical dashed lines (darkgreen). A genome-wide threshold of LOD = 3.0 (black horizontal line) and a suggestive threshold of LOD = 2.0 (pink horizontal line) were superimposed onto the plot for comparison.
Figure 6: Multi-point LOD scores by chromosomes 19-22 and X-chromosome. Multi-point LOD scores were plotted by chromosomes 19-22 and X-chromosome. The known AMD susceptibility genes were labeled by vertical dashed lines (darkgreen). A genome-wide threshold of LOD = 3.0 (black horizontal line) and a suggestive threshold of LOD = 2.0 (pink horizontal line) were superimposed onto the plot for comparison.
2.2.2 Genome-wide Association Signals

We carried out gene-based tests to investigate AMD susceptibility genes on autosomes by using retrospective kernel-based approach and burden tests. The boundaries for a certain gene might vary with the inclusion of different transcripts. Tests were conducted in two different ways. First we considered a combination of common and rare variants, and tested a total number of 16,844 autosomal genes. We controlled a genome-wide threshold of $2.97 \times 10^{-6}$ after Bonferroni correction. We next excluded all common variants, and focused on the genes having at least two polymorphic rare variants. A total number of 14,849 genes were tested, associated with a threshold of $3.37 \times 10^{-6}$ after Bonferroni correction. Those genes reaching genome-wide significance were summarized in Table 4.

For the analysis of the common and rare variants, we confirmed, as anticipated, that strong association was detected between AMD and \textit{CFH} and \textit{ARMS2}, two known AMD susceptibility genes. Both of the kernel and burden tests reached genome-wide significance (see the p-values in boldface associated with \textit{CFH} and \textit{ARMS2} in Table 4).

Table 4: Retrospective kernel-based approach and burden tests for a combination of common and rare variants

<table>
<thead>
<tr>
<th>chr</th>
<th>gene</th>
<th>nvariant</th>
<th>length (kb)</th>
<th>start (bp)</th>
<th>end (bp)</th>
<th>kernel_MB</th>
<th>burden_MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>\textit{CFH}</td>
<td>3</td>
<td>50</td>
<td>196,621,007</td>
<td>196,670,695</td>
<td>$1.537 \times 10^{-08}$</td>
<td>$1.477 \times 10^{-07}$</td>
</tr>
<tr>
<td>chr1</td>
<td>\textit{CFH}</td>
<td>15</td>
<td>96</td>
<td>196,621,007</td>
<td>196,716,634</td>
<td>$8.253 \times 10^{-12}$</td>
<td>$6.527 \times 10^{-10}$</td>
</tr>
<tr>
<td>chr10</td>
<td>\textit{ARMS2}</td>
<td>2</td>
<td>3</td>
<td>124,214,178</td>
<td>124,216,688</td>
<td>$6.536 \times 10^{-13}$</td>
<td>$1.118 \times 10^{-10}$</td>
</tr>
</tbody>
</table>

kernel\_MB and burden\_MB denote the retrospective kernel and burden tests with the Madsen-Browning weights, respectively. Genes reported with either kernel or burden, or both tests with a p-value $< 2.97 \times 10^{-6}$ (in boldface).

For \textit{CFH}, we discovered two transcripts reaching genome-wide significance. One region of 96 kb in length (196,621,007-196,716,634) with 15 variants contained the other one (196,621,007-196,670,695) with only 3 variants, which indicated that the transcripts of \textit{CFH} associated with AMD might be located within the smaller region between 196,621,007 and 196,670,695 bp on chromosome 1. However, for the analysis of the rare variants with MAF $< 0.05$, no genes reached the genome-wide significance level with either kernel-based or
burden statistics. For \textit{ARMS2}, actually both of the two variants in our exome chip data were common variants with MAF = 0.3933 and 0.1002, respectively, so they were not included in the rare variant analysis. For \textit{CFH}, the 96 kb region included 7 rare variants, but failed to reach the significance threshold, indicating that the common variants within \textit{CFH} might play a pivotal role in the significant association signal. In Table 5, we listed all common variants within in \textit{CFH} and \textit{ARMS2}, and examined their association on an individual level by conducting the \(M_{QLS}\) test suggested by Thornton and McPeek [Thornton and McPeek, 2007].

<table>
<thead>
<tr>
<th>variant</th>
<th>gene</th>
<th>chr</th>
<th>position (bp)</th>
<th>major/minor</th>
<th>MAF</th>
<th>(M_{QLS})</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs800292</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196642233</td>
<td>[T/C]</td>
<td>0.1584</td>
<td>(6.3 \times 10^{-06})</td>
</tr>
<tr>
<td>rs1329424</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196646176</td>
<td>[A/C]</td>
<td>0.4578</td>
<td>(1.1 \times 10^{-12})</td>
</tr>
<tr>
<td>rs10737680</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196679455</td>
<td>[T/G]</td>
<td>0.2735</td>
<td>(9.2 \times 10^{-11})</td>
</tr>
<tr>
<td>rs6677604</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196686918</td>
<td>[A/G]</td>
<td>0.1279</td>
<td>(1.7 \times 10^{-05})</td>
</tr>
<tr>
<td>rs1410996</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196696933</td>
<td>[T/C]</td>
<td>0.2729</td>
<td>(1.6 \times 10^{-10})</td>
</tr>
<tr>
<td>rs380390</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196701051</td>
<td>[C/G]</td>
<td>0.4310</td>
<td>(2.4 \times 10^{-10})</td>
</tr>
<tr>
<td>rs1329428</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196702810</td>
<td>[T/C]</td>
<td>0.2732</td>
<td>(1.8 \times 10^{-10})</td>
</tr>
<tr>
<td>rs1065489</td>
<td>\textit{CFH}</td>
<td>chr1</td>
<td>196709774</td>
<td>[A/C]</td>
<td>0.1587</td>
<td>(3.6 \times 10^{-01})</td>
</tr>
<tr>
<td>rs10490924</td>
<td>\textit{ARMS2}</td>
<td>chr10</td>
<td>124214448</td>
<td>[T/G]</td>
<td>0.3933</td>
<td>(5.3 \times 10^{-15})</td>
</tr>
<tr>
<td>rs10490923</td>
<td>\textit{ARMS2}</td>
<td>chr10</td>
<td>124214251</td>
<td>[A/G]</td>
<td>0.1002</td>
<td>(3.7 \times 10^{-01})</td>
</tr>
</tbody>
</table>

\(M_{QLS}\) tests for the common risk variants within \textit{CFH} and \textit{ARMS2} identified in Table 4. The \(M_{QLS}\) statistics associated with a significant p-value < 2.09 \times 10^{-6} were highlighted in boldface.

In addition to \textit{CFH} and \textit{ARMS2}, we did not discover any other gene with a significant association signal in our exome chip data. In Chapter 3, we would analyze the same exome chip data set by using our proposed approaches for comparison.

To investigate the behavior of the kernel-based and burden tests in real data analysis, we drew quantile-quantile (Q-Q) plots of the gene-based statistics (see Figures 7 and 8). It was clear that the kernel-based statistics resulted in a higher estimated \(\lambda_{GC}\) value than corresponding the burden statistics, regardless of a combination of common and rare variants (1.22 vs. 1.09) or only rare variants (1.25 vs. 1.08). This interesting observation would be compared with the embedded kernel-based and burden tests discussed in Chapter 3.
Figure 7: Q-Q plots for the retrospective kernel and burden tests with a combination of common and rare variants. Kernel and burden statistics were illustrated on the top and bottom row, respectively. Left and right column showed the genes with and without AMD risk genes, respectively. The genes contained a combination of common and rare variants. $\lambda_{GC}$ values were estimated and superimposed onto the plots to denote genomic inflation factor, indicating how far the points were away from the reference line. The 95% confidence interval was shaded in gray.
Figure 8: Q-Q plots for the retrospective kernel and burden tests with rare variants. Kernel and burden statistics were illustrated on the top and bottom row, respectively. Left and right column showed the genes with and without AMD risk genes, respectively. The genes contained only rare variants. \( \lambda_{GC} \) values were estimated and superimposed onto the plots to denote genomic inflation factor, indicating how far the points were away from the reference line. The 95% confidence interval was shaded in gray.
2.3 DISCUSSION

In this chapter, we conducted linkage analysis in MERLIN and gene-based association tests by using retrospective regression techniques on cleaned exome chip data. The two analyses approach the investigation of AMD etiology from different angles. Linkage analysis addresses the research question by looking at how risk alleles cosegregate and inherit in pedigrees, while association tests compare the frequency of risk alleles between cases and controls.

For the multi-point linkage analysis with linkage disequilibrium modeled, we identified a significant linkage region on chromosome 6 between 31,773,821 and 32,439,508 bp, which maps to the 6p21.31 band containing a total of 48 genes. Comparing with either a frequently used LOD = 3.0 or a more stringent threshold of LOD = 3.3, the linkage signal we suggested reached genome-wide significance with the highest LOD = 3.867. Several genes present in this linkage region were previously shown to be AMD susceptibility genes including C2, CFB, SKIV2L, and HLA-DRA belonging to HLA class II alpha chain paralogues. Actually, the linkage with AMD on chromosome 6 was reported and discussed in a couple of early studies. In 2003, Schick et al. observed a suggestive but less significant linkage peak on chromosome 6 [Seddon et al., 2003], but did not bound the region in detail. Later in 2007, Thompson et al. identified reported a significant region on 6q21-23 with evidence from two independent samples via investigating the rate of pigmentary abnormalities and geographic atrophy (PA/GA) [Thompson et al., 2007]. We have noted that both Schick and Thompson applied a significance threshold of p-value < 0.01, which in our opinion is less persuasive as genome-wide linkage significance. Here we provided sufficient evidence to argue that this linkage region would be a narrower band on chromosome 6 (6p21.31) with the variants contributive to the development of AMD all reaching a support interval threshold. Our findings provide one more important piece of evidence on the linkage with AMD on 6p21.

For the suggestive signals with multi-point LOD scores ≥ 2.0, we identified moderate linkage peaks on chromosomes 5, 8, 9, and 12 (see Table 3 for details). The association between AMD and 9q24 has been established by previous studies in terms of two susceptibility genes C5 [Baas et al., 2010] and TLR4 [Zareparsi et al., 2005], while the linkage of this region has never been clearly described. The region we discussed on chromosome 9 is
bounded between 116,856,481 and 118,741,214 bp, with multi-point LOD scores $\geq 2.0$ in our analysis. According to the assembly GRCh37 and the linkage region we discussed in Section 2.2.1, $C5$ is located on chromosome 9 between 123,714,613 and 123,812,554 bp, which is downstream of our suggested region. $TLR4$, also called $ARMD10$ as an alias, is located between 117,704,175 and 117,717,491 bp, within the proposed region. These evidence may support the authentication of the suggestive linkage on chromosome 9. For the other suggestive linkage signals on chromosomes 5, 8, and 12, however, no previous AMD susceptibility genes were reported within or close to the corresponding region. To further investigate these suggestive linkages, we may need to collect more data, especially genotype data within these regions of interest.

For the association testing, we recognized that the missing problem in our study with an unbalanced design would inflate p-values. We addressed the missing data problem by replacing them with values derived from the frequencies of non-missing genotypes. While it might not be an optimal way to handle the missing genotypes, it would be time-consuming to do an accurate pedigree-based imputation in a genome-wide association study. Since we randomly selected a value from $\{0, 1, 2\}$ to substitute for a missing genotype, chances were that Mendelian inheritance might be violated within a pedigree. However, such violations were uncommon to occur.

Assuredly, we confirmed significant associations between AMD and $CFH$ on chromosome 1 and $AMRS2$ on chromosome 10 with both kernel-based approach and burden tests. In our association analysis, we considered different transcripts by varying gene boundaries for definition. An advantage by doing this is that we are able to in-depth unveil the transcripts of a gene that contribute to AMD. For $CFH$, we defined two different gene regions containing 3 and 15 variants, respectively. The smaller region (196,621,007-196,670,695) is contained in the larger region (196,621,007-196,716,634) with the same starting position. According to the results in Table 4, both of the two regions reach genome-wide significance, indicating the transcripts of $CFH$ with in a small region between 196,621,007 and 196,670,695 bp on chromosome 1 may contribute to AMD. Besides examining a combination of common and rare variants with MAF $< 0.05$, we also investigated the effects of rare variants. When using only rare variants, no genes reached genome-wide significance. Both of rs10490923
and rs10490924 within ARMS2 are common variants, and are excluded in the rare variant analysis. For CFH, the 96 kb transcript contains 7 rare variants, but is not significantly associated with AMD based on the retrospective kernel or burden tests. Through the analysis, we have demonstrated that for both CFH and ARMS2, the common variants have an impact on the significant association signals. From Table 11, not surprisingly, we have identified and confirmed these common variants within CFH and ARMS2. In fact, the individual association of these common risk variants has clearly been reported and discussed in previous publications [Hughes et al., 2006, Chen et al., 2010, Kanda et al., 2007]. Our results in Tables 4 present one more solid piece of evidence to the AMD risk loci within CFH and ARMS2. Besides the two well-known AMD risk genes, we observed no other genes reaching genome-wide significance by using the retrospective kernel or burden tests.

For the Q-Q plots in Figures 7 and 8, we observed a slight discrepancy in the estimated $\lambda_{GC}$ value between kernel-based approach and burden tests. Since kernel-based approach resulted in a higher $\lambda_{GC}$ value than burden tests regardless of the rarity of the variants, it might trigger an argument that the p-values for kernel-based approach were inflated in the real data analysis, or the kernel test could be more powerful than the burden test. This potential issue, if true, has not been well noted or clearly described since the development of the approach, for Schaid et al. did not further apply their proposed statistics to a real data set to conduct a GWAS. Although a GWAS is not an indispensable requirement, it may help identify some practical issues that simulation studies may not be able to identify or solve. When the AMD risk genes were removed, we did not see an obvious decrease of the $\lambda_{GC}$ estimates for either the kernel-based approach or the burden tests.

In Chapter 3 (see Section 3.1.5), we introduced a novel idea that we used the B-Spline basis functions to smooth and thus reduce the dimension of genotype data, and embedded smoothed genotypes in the retrospective kernel and burden tests. Through dimension reduction, we furthered our discussions on the performance of the embedded approach and its comparison with the ordinary retrospective regression approaches. We also focused more attention on the Q-Q plots of the embedded kernel tests which would reflect the behavior of kernel-based approach in real data analysis.
In this chapter, we first develop gene-based tests for a dichotomous trait and related individuals in known pedigrees by extending the generalized FLM previously discussed for population data. With the idea borrowed from the retrospective regression, we then embed the FLM-smoothed genotypes in the kernel-based approach and burden tests previously developed by Schaid et al. To assess the behavior of our newly introduced statistics, we conduct simulation studies to evaluate type I error rates and power levels, and make a comparison with some other compelling statistical methods for family-based association tests. Finally, we apply our new statistics to a real exome chip data set to identify AMD associated susceptibility genes or loci.

3.1 METHODS

3.1.1 GFLMM and Null Hypothesis for Testing

Consider a family-based genome-wide study including $N$ participants with a dichotomous trait of interest coded as 1 and 0 denoting cases and controls, respectively. Let $n$ denote the $n^{th}$ individual with $m$ genetic variants genotyped within a chromosome region. The physical locations of the $m$ variants, denoted by $0 \leq t_1 \leq t_2 \leq \cdots \leq t_m$ in a sequence, are normalized on the unit region $[0, 1]$.

Based on the beta-smooth only model proposed by Fan et al. [Fan et al., 2013], we extend the approach to handle pedigree relatedness. Consider a logit link function on the probability
of affected status for an individual
\[
\log \left( \frac{p_n}{1 - p_n} \right) = \beta^T Z_n + \gamma^T \Phi(t) X_n(t) + \sigma a_n ,
\]  
(3.1)

where \( p_n \) is the probability that the \( n^{th} \) individual is an affected case; \( \beta, \gamma, \) and \( \sigma \) are the effects of fixed covariates, genetic variants, and an unobserved polygenic term, respectively; \( Z_n \) is a design vector of fixed effects with its first element equal to 1 indicating the baseline effect; \( \Phi(t) \) is a matrix containing the functions of the B-Spline or the Fourier basis; \( X_n(t) \) is estimated by \( X_n(t) = G_n(t) \), which takes values in \( \{0, 1, 2\} \) determined by the copy numbers of minor alleles; and \( a_n \) is an unobserved random polygenic effect, following a multivariate normal distribution.

For an illustrative purpose, consider a genetic region with four variants under a series of three basis functions, Equation (3.1) can be expanded and rewritten by
\[
p_n = \log^{-1} \left\{ \beta^T Z_n + \gamma^T \Phi(t) X_n(t) + \sigma a_n \right\} 
= \log^{-1} \left\{ \beta^T Z_n + \begin{pmatrix} \gamma_1 & \gamma_2 & \gamma_3 \end{pmatrix} \begin{pmatrix} \varphi_1(t_1) & \varphi_1(t_2) & \varphi_1(t_3) & \varphi_1(t_4) \\ \varphi_2(t_1) & \varphi_2(t_2) & \varphi_2(t_3) & \varphi_2(t_4) \\ \varphi_3(t_1) & \varphi_3(t_2) & \varphi_3(t_3) & \varphi_3(t_4) \end{pmatrix} \begin{pmatrix} g_n(t_1) \\ g_n(t_2) \\ g_n(t_3) \\ g_n(t_4) \end{pmatrix} + \sigma a_n \right\} .
\]

For the B-Spline basis, \( \varphi_k(t) = B_k(t) \), \( k = 1, 2, \) and 3. For the Fourier basis, \( \varphi_1(t) = 1, \varphi_2(t) = \cos (2\pi t) \), \( \varphi_3(t) = \sin (2\pi t) \), with an odd number of basis functions as required. After applying the beta-smooth only smoothing, we can observe that the dimension of the genetic effect is reduced by one in this illustrative example.

By extending Stanhope and Abney’s framework [Stanhope and Abney, 2012] to a more general situation, we desired to formulate a composite null hypothesis
\[
H_0 : \gamma^T = \gamma_0^T = 0
\]  
(3.2)
in the presence of \( \beta \) and \( \sigma \) as nuisance parameters.
3.1.2 Likelihood Functions

To facilitate the following presentation of the likelihoods, we let \( \theta = \{\beta, \gamma, \sigma\} \) and \( \eta = \{\beta, \sigma\} \). Then the parameter space can be redefined by \( \theta = \{\gamma, \eta\} \).

Given the unobserved polygenic effect and for Bernoulli random variables, the conditional log likelihood function can be denoted by

\[
\ell_a (\theta | a) = \sum_{n=1}^{N} y_n \log p_n + (1 - y_n) \log (1 - p_n) .
\]  

(3.3)

To estimate and maximize the likelihood, \( a \) would have to be integrated out in Equation (3.3). Then the likelihood function can be expressed by

\[
L(\theta) = \int \exp (\ell_a (\theta | a)) h(a) \, da ,
\]

(3.4)

where \( h(a) \) is the density function of the multivariate normal distribution \( \text{MVN}(0, \Sigma) \), where \( \Sigma \) is \( 2 \times \) global kinship matrix accounting for pedigree relatedness.

Let \( \hat{\theta} = \{\gamma_0, \hat{\eta}\} \) denoting the constrained maximum likelihood estimation under \( H_0 \). By cubature approximation, Equation (3.4) can approximated by

\[
\log L(\theta) \approx \log \left( \sum_{c=1}^{C} \exp \{\ell_{a_c}(\theta)\} w_c \right)
= \log \left( w * \sum_{c=1}^{C} \exp \{\ell_{a_c}(\theta)\} \right) ,
\]

(3.5)

(3.6)

where \( a_c \) represents the multivariate normal cubature points transformed from the individual points in the Sobol cubature, and \( w_c \) is the associated cubature weights updated after each iteration. If equal weights are chosen, \( w \) no longer depends on \( c \).
3.1.3 Derivatives and Test Statistics

The first derivative of the log likelihood function, or the score function, is a gradient vector computed by

\[ S(\theta) = (S(\gamma), S(\eta))^T \]

\[ \equiv \left( \frac{\partial}{\partial \gamma} \log L(\theta), \frac{\partial}{\partial \eta} \log L(\theta) \right)^T \]

\[ \approx \frac{\partial}{\partial \theta} \log \left( \sum_{c=1}^{C} \exp \{ \ell_{ac}(\theta) \} w_c \right) \quad \text{(by approximation)} \]

\[ \sum_{c=1}^{C} \frac{\partial}{\partial \theta} \ell_{ac}(\theta) \exp \{ \ell_{ac}(\theta) \} w_c \]

\[ = \frac{\sum_{c=1}^{C} \exp \{ \ell_{ac}(\theta) \} w_c}{\sum_{c=1}^{C} \exp \{ \ell_{ac}(\theta) \} w_c} \quad . \quad (3.7) \]

With equal weights, Equation (3.7) can be reduced by canceling \( w \)

\[ S(\theta) = \frac{\sum_{c=1}^{C} \frac{\partial}{\partial \theta} \ell_{ac}(\theta) \exp \{ \ell_{ac}(\theta) \}}{\sum_{c=1}^{C} \exp \{ \ell_{ac}(\theta) \}} \quad , \quad (3.8) \]

where

\[ \frac{\partial}{\partial \theta} \ell_{ac}(\theta) = \sum_{n=1}^{N} (y_n - p_n) z_{an}^\theta \quad , \quad (3.9) \]

where

\[ z_{an}^\theta = (\Phi(t)G_n(t), Z_n, a_c(n)) \quad . \quad (3.10) \]

Particularly, to get the score function for the genetic effect denoted by \( S(\gamma) \), we set

\[ z_{an}^\gamma = (\Phi(t)G_n(t))^T \quad (3.11) \]

to obtain the first derivative only with respect to \( \gamma \).
Similarly, the negative second derivative, or the observed Fisher information, is computed by

\[-I(\theta) \equiv \frac{\partial^2}{\partial \theta \partial \theta^T} \log L(\theta | a)\]

\[
\approx \sum_{c=1}^{C} \left( \frac{\partial^2}{\partial \theta \partial \theta^T} \ell_{ac}(\theta) + \frac{\partial}{\partial \theta} \ell_{ac}(\theta) \frac{\partial}{\partial \theta^T} \ell_{ac}(\theta) \right) \exp \{\ell_{ac}(\theta)\} w_c \]

\[
- \frac{\partial}{\partial \theta} \log (L(\theta)) \frac{\partial}{\partial \theta^T} \log (L(\theta)) ,
\]

which reduces to

\[-I(\theta) \approx \sum_{c=1}^{C} \left( \frac{\partial^2}{\partial \theta \partial \theta^T} \ell_{ac}(\theta) + \frac{\partial}{\partial \theta} \ell_{ac}(\theta) \frac{\partial}{\partial \theta^T} \ell_{ac}(\theta) \right) \exp \{\ell_{ac}(\theta)\} \]

\[
- \frac{\partial}{\partial \theta} \log (L(\theta)) \frac{\partial}{\partial \theta^T} \log (L(\theta))
\]

(3.12)

with equal weights, where

\[
\frac{\partial^2}{\partial \theta \partial \theta^T} \ell_{ac}(\theta) = \sum_{n=1}^{N} (p_n^2 - p_n) z_n^{\theta} z_n^{\theta T} .
\]

(3.14)

We recognize that a similar formula to Equation 3.12 was previously derived by Stanhope and Abney. However, there should be a “+” sign in the numerator of the first instead of a “−” sign as they derived in their Appendix 3 on page 9.

To construct the test statistic, we consider Rao’s score test. A general form of the test statistic can be formulated by \[ R = S(\theta)^T I(\theta)^{-1} S(\theta) \] [Rao, 1948, Rao, 2005]. Under the null hypothesis (3.2) in the presence of the estimated nuisance parameters \( \hat{\eta} \) under the maximum likelihood,

\[
R_r = S(\gamma_0)^T I(\gamma)^{-1} S(\gamma_0) \bigg|_{\beta = \hat{\beta}, \sigma = \hat{\sigma}} \sim \chi^2_{k_r} ,
\]

(3.15)

where \( \hat{\beta} \) and \( \hat{\sigma} \) are the maximum likelihood estimates of \( \beta \) and \( \sigma \), respectively. To make inference of the genetic effect from the score statistics, chi-squared distribution function is used to convert the score statistic to desired p-value under the null hypothesis.
To compute $I(\gamma)^{-1}$, note that the elements of $I(\theta)$ are determined by the derivatives with respect to the parameters of genetic effect $\gamma$ and the nuisance parameters $\eta$, and $I(\gamma)^{-1}$ can be determined by the manipulation of the block matrix $I(\theta)$. Here we define

$$I(\theta) \equiv -\frac{\partial^2}{\partial \theta \partial \theta^T} \log L(\theta) = \begin{bmatrix} I_{\gamma\gamma}(\theta) & I_{\gamma\eta}(\theta) \\ I_{\eta\gamma}(\theta) & I_{\eta\eta}(\theta) \end{bmatrix}. \quad (3.16)$$

Then

$$I(\gamma)^{-1} = \left[ I_{\gamma\gamma}(\theta) - I_{\gamma\eta}(\theta)I_{\eta\eta}(\theta)^{-1}I_{\eta\gamma}(\theta) \right]^{-1}. \quad (3.17)$$

### 3.1.4 Computational Issues in Multidimensional Integration

As mentioned in Section 3.1.2, to estimate the maximum likelihood we have to integrate out the unobserved polygenic effect. This integration is a multidimensional one, and its computational hurdle and efficiency may depend on the sample size and the complexity of pedigree structures. To conquer the technical challenges, we resorted to the GLOGS program (http://www.bioinformatics.org/~stanhope/GLOGS), and addressed the high-dimensional integration via a tractable and parallel implementation [Stanhope and Abney, 2012].

As previously discussed in detail, the GLOGS program features the estimation of parameters from a logistic regression model with a random polygenic effect term and the performance of score tests between a dichotomous disease trait and a single marker. In our study, we took full advantage of the first characteristic to achieve the aim of integration. However, rather than simply applying the relevant programs in the GLOGS package, we considered several modifications before adopting it to our work. The updates are summarized in Table 6 and discussed in detail in the following presentation.

#### 3.1.4.1 The Gauss-Newton algorithm with step-halving

The original GLOGS program uses the Gauss-Newton algorithm to update the parameter space by updating $\hat{\Theta}^{i+1} = \hat{\Theta}^i - D_{\Theta}^2 \log L^i(\hat{\Theta}^i)^{-1} D_{\Theta} \log L^i(\hat{\Theta}^i)$, where $\hat{\Theta}^i$ is the estimated parameter space update at the $i^{th}$ iteration, and $D_{\Theta}$ and $D_{\Theta}^2$ denote the first and second derivative with respect to the parameter space $\Theta$, respectively. Despite being a broadly accepted and applied technique,
it may result in the “downhill” (more negative in our case) estimations of the log likelihood in a few computational iterations. Since our aim is always maximizing the log likelihood, the algorithm without any modifications may be deviate and inefficient to achieve the maximization purpose. To improve the algorithm, we instead used the Gauss-Newton updates with a step-halving process to ensure a consistent uphill estimation of the log likelihood in each iteration by updating

\[ \hat{\Theta}^{i+1} = \hat{\Theta}^i - \alpha \cdot D^2 \Theta \log L^i(\hat{\Theta}^i)^{-1} D \Theta \log L^i(\hat{\Theta}^i), \tag{3.18} \]

where \( \alpha = (1/2)^n \), and \( n \) is a positive integer starting from 1. We compare the log likelihood in the \((i+1)^{th}\) iteration with the one in the \(i^{th}\) iteration to determined whether or not the step-halving needs to be invoked by applying an if condition loop:

```
if ( log L^i(\hat{\Theta}^{i+1}) - log L^i(\hat{\Theta}^i) < -10^{-8} )
    { invoke the step-halving procedures }
else
    { move on to the next iteration }
```

### 3.1.4.2 Stopping criterion

The stopping criterion in the original GLOGS program deals with a comparison of the Euclidean norms of the Jacobian matrix from the two adjacent iterations: \( \| D\Theta \log(L^i(\hat{\Theta}^i)) - D\Theta \log(L^i(\hat{\Theta}^{i+1})) \| < \delta \), where \( D\Theta \) denotes the first derivative with respect to the parameter space, and \( \delta \) is a small number. To the best of our knowledge, such a convergence criterion is seldom used in estimating parameters as a stopping criterion.
Actually it may result in some issues. Here is a straightforward counterexample that nullifies the stopping criteria in one-dimensional space. If we define
\[ f(x) = -(x - 1)^2 \quad (x \in \mathbb{R}) , \] (3.19)
it is obvious that \( f(x) \) is maximized at \( x = 1 \), and the first derivative of \( f(x) \) is
\[ \frac{df(x)}{dx} = 2 - 2x \quad (x \in \mathbb{R}) . \] (3.20)
Particularly, we let \( x^i = 0.5, x^{i+1} = 0.5000004 \), and \( \delta = 10^{-6} \). By applying the stopping criteria, we will obtain \( \|2 - 2x^i\| - \|2 - 2x^{i+1}\| = 8 \times 10^{-7} < \delta = 10^{-6} \). However, \( x^i = 0.5 \) is still far from 1, the value we anticipate to maximize \( f(x) \).

To address the problem, we suggested a different but more widely-acknowledged convergence rule in our improved program by considering the relative function criterion
\[ 2 \left| \log(L(\hat{\Theta}^{i+1})) - \log(L(\hat{\Theta}^i)) \right| \leq \text{FTOL} \ast \left( \left| \log(L(\hat{\Theta}^{i+1})) \right| + \left| \log(L(\hat{\Theta}^i)) \right| + \epsilon \right) , \] (3.21)
where \( \epsilon \) is a small number, and we set \( \text{FTOL} = 10^{-8} \) to speed up the convergence without significantly losing the estimation precision.

### 3.1.4.3 Cubature weights

To approximate the multidimensional integral, the original GLOGS program computes the cubature weights using a strategy called Bayesian updating after estimating the parameter space at a certain iteration. It also reassigns the cubature point weights near zero to be zero in order to accelerate the computation speed. However, this strategy may lead to the involvement of a single weight to approximate a high-dimensional integral in a large data set with more than 1,000 samples. Although the strategy appears to be working for the sample data sets included in the GLOGS program package, we suggest using equal weights to better approximate the desired integration instead of applying the Bayesian updating. The advantages of our suggested weightings are obvious. If \( C \) cubature points were applied to the approximation, the weight for each iteration would simply be \( 1/C \). Another nice aspect is that if equal weights are chosen, our proposed model can be reduced by canceling out the weighting term (see Equations 3.8 and 3.13), leading to a straightforward derivation and more convenient and efficient programming.
3.1.4.4 Small sample testing  We tested our modified GLOGS program in two small data sets with 7 and 25 individuals, respectively, to estimate parameters and maximum likelihoods. Then we used the “cubature” package (https://cran.r-project.org/web/packages/cubature/index.html) and the “R2Cuba” package (https://cran.r-project.org/web/packages/R2Cuba/index.html) to compute the likelihoods in R at the estimated parameter vector as returned by the GLOGS program. The results were summarized in Table 7. Compared with the results generated by the R packages, the original GLOGS program failed to approximate a likelihood value close to a true maximum likelihood estimate (see Table 7, A lines). After we modified the program, the results were consistent with those obtained by the existing R packages in our small testing data sets.

![Table 7: Multidimensional integration in small samples](image)

Multi-dimensional integration in small samples by using the (modified) GLOGS program, R “cubature” package, and R “R2Cuba” package. A: Original GLOGS program with no modifications; B: Original GLOGS program with equal weights; C: Our improved version with equal weights and corrected convergence criterion; D: Our version with equal weights, corrected convergence criterion, and step-halving procedure; (1) log likelihood: Estimated maximum log likelihood; (2) Inv_logL: Maximum log likelihood estimated by using inverse transformation techniques; (3) MVN_logL: Maximum log likelihood estimated by transforming cubature points to a multivariate normal distribution with the kinship covariance matrix, and then integrating on the [0,1] hypercube.

† The R “cubature” package cannot handle the adaptive integration of a function with a high dimension.
3.1.5 Embed FLM-smoothed Genotypes in Retrospective Regression

A random sampling process may have an influence on the bias of the statistics computed in Section 3.1.3. To sufficiently utilize genetic positions while eliminating ascertainment bias, we embed the smoothed genotypes in the retrospective regression techniques, and construct kernel and burden statistics with beta-smooth retrospective analysis. To do this in an efficient manner, we subject smoothed genotypes to the “pedgene” package developed in R. For this embedded approach, we only consider a smoothing process by using the B-Spline basis. We note that not all variants within a gene region can be reduced to a fixed arbitrarily small dimension, especially for those large genes containing a high number of genetic variants within them. This is because the the copy number of the minor allele for a particular variant cannot exceed 2, which means that its genotype data must satisfy \( \bar{G}/2 < 1 \), where \( \bar{G} \) is the mean of the genotypes for the variant across all of the individuals. To get around this issue and reduce more dimensions, we apply the smallest number of basis that can allow the analysis to run in the “pedgene” package. Since genetic variants within a gene are weighted by the B-Spline basis, we apply equal weights when computing the kernel and burden statistics.

3.1.6 Simulation Study

To evaluate the performance of our test statistics, we simulated data to estimate empirical type I error rates and power levels.

3.1.6.1 Pedigrees We first simulated 25 families by randomly choosing progeny sizes from a negative binomial distribution [Cavalli-Sforza and Bodmer, 1999]. We assumed that each child within the second generation has a 25% chance of having offspring. Pedigree connection was checked by the PEDSTATS software to ensure each individual was correctly and completely connected. The final structure of the pedigrees included 228 individuals (119 males and 109 females; 70 founders and 158 nonfounders) within 25 families. The pedigree size ranged from 4 to 24 with an average value of 9.12.
3.1.6.2 Genetic variants  A pool of 10,000 haplotypes was generated by Dr. Qi Yan under the coalescent approach by using the COSI program [Schaffner et al., 2005]. The haplotypes were modeled on a European population and span a length of 200 kb on a region of chromosome 1. We considered 14 kb, a reported median of gene length [Lander et al., 2001], as a desired biological region that included 261 genetic variants. To target rare variants, we removed those variants with MAF ≥ 0.05.

To evaluate power levels, we sampled two haplotypes at random within the 14 kb region for each founder. For each nonfounder, we chose one haplotype at random from his or her parents. Genotypes were constructed by summing up two haplotypes for each individual to indicate the copy number of minor alleles.

3.1.6.3 Trait assignment and ascertainment  Trait status was determined for each individual based upon the genotypes. To do this, we considered a logistic regression genetic model to compute the probability of being affected for the $i^{th}$ individual:

$$ p_i = \frac{\exp \left( \beta_0 + \sum_j \log (\text{OR}_j) g_{ij} + \sigma a_i \right)}{1 + \exp \left( \beta_0 + \sum_j \log (\text{OR}_j) g_{ij} + \sigma a_i \right)}, $$

(3.22)

where we let $\beta_0 = -2.94$ and $\sigma = 0.2$ to set the disease prevalence to 5% and to set the polygenic effect, respectively; $g_{ij}$ is the copy number of minor alleles of the $j^{th}$ genetic variant for the $i^{th}$ individual; OR$_j$ is the odds ratio associated with the $j^{th}$ genetic variant [Larson and Schaid, 2014]. The odds ratios in the model were set in the following two ways. For constant OR models, all risk and protective variants were associated with OR = 2 and OR = 0.8, respectively. For MAF-dependent OR models, ORs were determined by $\text{OR}_j = \frac{\log c}{4} |\log_{10}\text{MAF}_j|$ as proposed by Wu et al., where we let $c = 3$, 4 or 10 to associate a rare variant with MAF equal to 0.0001 with OR = 3, OR = 4 or OR = 10, respectively [Wu et al., 2011]. After assigning the phenotype for each individual, we applied an ascertainment criterion to variants selection to ensure that each pedigree must contain at least 1 pair of affected siblings, either in the second or third generation, or both. Through the ascertainment, we anticipated to sample cases enriched for the dichotomous trait, thus weakening the influence of the polygenic effect.
3.1.6.4 Null genes To simulate null genes for type I error rates evaluation, we had to generate genotype data that was unrelated to the trait data. To do this while saving time and lowering the computational burden, we kept the trait variable as assigned in Section 3.1.6.3, but randomly assigned haplotype data for all the founders, transmitted the haplotypes to their offspring, and finally converted the haplotypes to the genotypes. Thus, we ensured that the trait was unrelated to the genotype data, and both the transmission of the traits and the genotypes followed the laws of Mendelian inheritance within each pedigree.

3.1.6.5 Simulation scenarios We did 500 and 1,000 replicates to estimate power and type I error rates, respectively. For each replicate, nonpolymorphic variants, if any, were removed from the subsequent analysis, for they were not informative in estimation. To simulate more practical situations in reality, we examined several distinct scenarios in which approximately 40% or 30% of the total number of variants were randomly assigned to be risk ones either with or without a mixture of 20% protective variants. To achieve and demonstrate more comparable scenarios, we designed two schemes. First, we particularly subset the 30% of the risk variants from those 40% ones, and kept the same set of 20% protective variants for the scenarios containing common and rare variants. Second, to further investigate the influence of rarity of the genetic variants on the MAF-dependent models, within the same 263 variants extracted from the pool we randomly assigned the other set of risk and protective variants by keeping the proportions identical to those in the first scheme.

While our statistics are developed mainly for rare variant analysis, we are still interesting in evaluating how the type I error rates would behavior if there is a combination of common and rare variants. Specifically, we simulated additional but similar scenarios as we did for rare variants, but allowed common variants included in the same genetic region as designated previously. Besides 263 rare variants, there were 41 common variants added in, resulting in a total number of 304 variants.

3.1.6.6 Choice of norder and nbasis There are two parameters we can specify when using the B-Spline or the Fourier basis to weight the genotypes. One parameter is norder, which determines the degree of basis functions. It shapes the form of the basis function by
the order of the curve (linear, quadratic, cubic, . . . ). The other parameter is \( nbasis \), which controls the number of nodes used in the polynomial interpolation. Finding an optimal combination of \( norder \) and \( nbasis \) in the analysis is a challenging procedure with the ad hoc solutions depending on the number and rarity of genetic variants.

In our simulation study for rare variants, we considered \( norder = 4 \) and \( nbasis = 5 \) to deeply reduce the dimension for both the GFLMM and the embedded approach. For a combination of common and rare variants, we kept using \( norder = 4 \) and \( nbasis = 5 \) for the GFLMM, but recognized that type I error rates were inflated in our embedded approach if we kept using the same parameters as we did for rare variants. To enable better calibration, we chose \( norder = 6 \) and \( nbasis = 19 \) for the B-Spline basis when smoothing genotypes in the embedded approach.

### 3.1.6.7 Other statistics for comparison

To further evaluate the performance of our proposed methods, we made a transverse comparison by exploring the other different statistics previously published. Besides the retrospective kernel and burden tests, we also included family-based SKAT (famSKAT) and F-SKAT tests in the simulation study. The famSKAT statistic is developed based on the idea of SKAT, with the capability to handle pedigree data. The extension lies in the inclusion of random effects for familial correlation [Chen et al., 2013]. Although famSKAT was initially described for the analysis of quantitative traits, we treated 0 and 1 as quantitative traits, and made famSKAT an appropriate approach for comparison. F-SKAT, on the other hand, can be considered another extension of SKAT to analyze related individuals with dichotomous traits [Yan et al., 2015]. It is based on a kernel machine regression, and within a framework of generalized linear mixed models. For all the statistics in comparison, we ran both weighted and unweighted versions. The weights were based on a function of MAF estimated by \( 1/\sqrt{\text{MAF} \times (1 - \text{MAF})} \) [Madsen and Browning, 2009]. Beta density weights were also used when retrospective kernel and burden statistics were computed [Wu et al., 2011].
3.1.7 Real Data Analysis

To assess the behavior of our statistics in real situations, we conducted genome-wide association analysis in the UCLA/Pitt family-based study. As discussed in Chapter 2, the same cleaned exome chip data set was used. Missing genotypes were replaced by 0, 1, or 2, denoting the copy number of minor alleles. The probabilities of the replacement with these values were determined by the non-missing genotype frequency. All those genes with at least two polymorphic variants within gene boundaries were included in the analysis. In accordance with the performance of the statistics in the simulation study, ad hoc analysis plans were made to better handle the genes with different number of variants.

3.1.7.1 Analysis of a combination of common and rare variants  With the inclusion of the common variants in the data set, we first focused on the combination of common and rare variants, and explored the AMD susceptibility genes by using our statistics discussed so far in this work. We drew Q-Q plots to examine the similarity of the observed and expected values.

According to the simulation results, our GFLMM performed well under $norder = 4$ and $nbasis = 5$ for both rare variants or the combination of common and rare variants. Correspondingly, for the analysis of the real exome chip data with a combination of common and rare variants, we kept using a constant set of $norder = 4$ and $nbasis = 5$. Particularly, for the genes with more than 5 polymorphic variants, we set $norder = 4$ and $nbasis = 5$, and for the genes with 2 to 5 polymorphic variants, we set $norder = 3$ and $nbasis = 3$ to reduce the dimension.

As for the embedded approach, we picked a more complicated strategy when running the analysis. Since only the B-Spline basis was used to smooth the genotypes, there was no restriction on the parity of $nbasis$. For the genes with 2, 3 to 6, and 7 to 24 polymorphic variants, we set $norder = nbasis = 2$, $norder = nbasis = 3$, and $norder = nbasis = 6$, respectively. For the genes with more than 25 polymorphic variants, we set $norder = 6$, and let $nbasis$ be the smallest integer $\geq 1/4 \times$ number of variants. Particularly, for those genes containing a large number of variants, we chosen the smallest $nbasis$ value that could
maximize the dimension reduction, but still retain $\bar{G}/2 < 1$, where $\bar{G}$ was the mean of the genotypes for the variant across all of the individuals as mentioned in Section 3.1.5.

3.1.7.2 Analysis of rare variants With the exclusion of the common variants, we were then interested in exploring the rare variants. We reran the analysis by using the GFLMM and embedded approaches. For the GFLMM, we used the same choice of $norder = 4$ and $nbasis = 5$ as we did for a combination of common and rare variants in Section 3.1.7.1. For the embedded approach, per the simulation study, we considered it appropriate to accomplish weighting by simply setting $norder = 4$ and $nbasis = 5$ for those genes containing more than 4 polymorphic variants. For those small-size genes with 2 to 3 polymorphic variants, we set $norder = nbasis = \text{number of variants}$.

3.1.8 Investigate the Behavior of Kernel-based Statistics in Real Data Analysis

In Chapter 2, we discovered and discussed the issue that kernel-based approach might generate smaller p-values than burden tests (see the Q-Q plots in Section 2.2.2 and the discussions in Section 2.3). To examine the similar issue for the GFLMM and the embedded approaches, we draw Q-Q plots similar to the ones we plotted in Section 2.2.2. For the GFLMM, we focused on the comparison between the usage of the B-Spline basis and the Fourier basis. For the embedded approach, we were interested in investigating the difference between the kernel-based and burden tests. We analyzed the scenarios of both common and rare variants and only rare variants separately.

3.2 RESULTS AND CONCLUSIONS

3.2.1 Simulation Results

Within 25 two- or third- generation families, we sampled a total of 228 related individuals. Type I error rates and power levels were estimated under both fixed and MAF-dependent OR models. For each model, we varied the proportion of risk variants by 40% or 30%, and
switched the inclusion of 20% protective variants. The empirical p-values were estimated at the nominal levels of 0.05 and 0.01, respectively.

In this section, we presented simulation results for the type I error rates and power levels by bar plots. The notations of the statistics and scenarios labeled in the plots and mentioned in the dissertation were defined in Table 8.

Table 8: Summary of the notations in the plots

<table>
<thead>
<tr>
<th>Notations</th>
<th>Descriptions and Interpretations</th>
</tr>
</thead>
<tbody>
<tr>
<td>flmBS</td>
<td>GFLMM approach with the B-Spline basis</td>
</tr>
<tr>
<td>flmFR</td>
<td>GFLMM approach with the Fourier basis</td>
</tr>
<tr>
<td>flmBS_kernel</td>
<td>embedded kernel-based approach with the B-Spline basis</td>
</tr>
<tr>
<td>flmBS_burden</td>
<td>embedded burden tests with the B-Spline basis</td>
</tr>
<tr>
<td>kernel_BT</td>
<td>retrospective kernel-based approach with weights based on Beta distribution</td>
</tr>
<tr>
<td>burden_BT</td>
<td>retrospective burden tests with weights based on Beta distribution</td>
</tr>
<tr>
<td>kernel_MB</td>
<td>retrospective kernel-based approach with Madsen-Browning weights</td>
</tr>
<tr>
<td>burden_MB</td>
<td>retrospective burden tests with Madsen-Browning weights</td>
</tr>
<tr>
<td>kernel_UW</td>
<td>retrospective kernel-based approach with equal weights</td>
</tr>
<tr>
<td>burden_UW</td>
<td>retrospective burden tests with equal weights</td>
</tr>
<tr>
<td>famSKAT_W</td>
<td>family-based SKAT with MAF-dependent weights</td>
</tr>
<tr>
<td>famSKAT_UW</td>
<td>family-based SKAT with equal weights</td>
</tr>
<tr>
<td>FSKAT_W</td>
<td>F-SKAT with MAF-dependent weights</td>
</tr>
<tr>
<td>FSKAT_UW</td>
<td>F-SKAT with equal weights</td>
</tr>
<tr>
<td>2.0/0.8</td>
<td>Fixed OR models with OR = 2.0 for risk variants, and 0.8 for protective variants.</td>
</tr>
<tr>
<td>m10</td>
<td>MAF-dependent OR models with OR = ( \frac{\log c}{\log_{10} \text{MAF}} ), where ( c = 10 ).</td>
</tr>
<tr>
<td>40%/0%</td>
<td>Models with 40% risk variants and no protective variants</td>
</tr>
<tr>
<td>40%/20%</td>
<td>Models with a mixture of 40% risk variants and 20% protective variants</td>
</tr>
</tbody>
</table>

3.2.1.1 Type I error rates  We ran 1,000 replicates for each scenario when simulating the type I error rates which were presented and illustrated by Figures A.1-A.9 (see Appendix A) and Figures 9-17. The estimated p-values for the different statistics were plotted in vertical bars with various colors by different scenarios. We denoted nominal levels of 0.05 and 0.01 by solid lines. Exact 95% confidence intervals were superimposed onto the plots using dashed lines to indicate the upper and lower bounds of the estimated p-values. Although rare variants were of particular interest to this dissertation, we also examined the behavior of
our proposed statistics with a combination of rare and common variants. We also noted that the genetic model described in Section 3.1.6.3 for simulation was appropriate for estimating type I error rates with a combination of rare and common variants.

For flmBS and flmFR derived from the GFLMM, Rao’s score test statistics showed conservative behaviors. Overall, the empirical type I error rates were within the 95% confidence interval of the nominal values, with a couple of exceptions. We noted that the potential inflation occurred in those scenarios with only risk variants under the MAF-dependent models where \( c = 10 \) or \( c = 4 \), regardless of rare or a combination of common and rare variants (m10-40%/0% and m4-40%/0% in Figures A.1, A.3, 9, 11, A.4, A.6, 12, and 14). These scenarios reflect the ones where a gene contains rare variants with extremely high ORs. Despite unlikely situations, to figure out the reasons, we simulated additional scenarios in which we retained the same genetic region, but re-sampled the risk and protective variants while keeping the same proportion as previously we did. For these scenarios subsequently simulated, the type I error rates showed no sign of inflation. Compared with the prior set of variants, the risk variants in the re-sampled set had lower mean MAF values, suggesting that for fixed \( norder = 4 \) and \( nbasis = 5 \), MAF of risk variants would have an impact on the performance of flmBS and flmFR.

For the embedded approaches, we chose different \( norder \) and \( nbasis \) as described in Section 3.1.7.1 when handling rare and a combination of common and rare variants. The behavior of flm_kernel and flm_burden for rare variants under \( norder = 4 \) and \( nbasis = 5 \) was conservative, with the type I error rates within the anticipated 95% confidence interval for most scenarios. With common variants mixed in, we recognized that the similar setting of \( norder = 4 \) and \( nbasis = 5 \) would result in the inflation of the type I error rates. Further examinations led us to higher order B-splines and more nodes with \( norder = 6 \) and \( nbasis = 19 \). Most of the type I error rates were within the anticipated range with one exception of m4-40%/0%, which was slightly high above the upper bound (Figures 12 and 13). There was a palpable difference in specifying \( norder \) and \( nbasis \) between rare variants and a combination of common and rare variants. When common variants mixed in, the embedded approaches called for a higher order B-Spline with more nodes for smoothing.
The other statistics for comparison, including kernel-based approach, burden tests, famSKAT, and F-SKAT, were supposed to well calibrate the type I error rates according to their original discussions, but surprisingly not F-SKAT. Based on our simulation results, the type I error rates estimated by FSKAT_W and FSKAT_UW showed significant inflation in most scenarios, regardless of with fixed or MAF-dependent ORs (as in Figures A.2 and 11), or for rare or a combination of common and rare variants (as in Figures A.2 and A.4). Retrospective techniques, for both kernel-based approach and burden tests, produced relatively stable type I error rates close to the nominal values, despite some slight underestimates for kernel-based approach in a few scenarios (as in Figures 9 and 12). While treating dichotomous traits as continuous ones, famSKAT resulted in conservative type I error rates in our simulation study, especially in those scenarios with fixed ORs and MAF-dependent ORs with a small $c$ value (as in Figures A.3 and A.6).
Figure 9: Type I error rates of rare risk variants ($\alpha = 0.01$, 40% risk variants). Type I error rates for 40% rare variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure 10: Type I error rates of rare risk variants ($\alpha = 0.01$, a re-sampled set of 40% risk variants). Type I error rates for a distinct set of 40% rare variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure 11: Type I error rates of rare risk variants ($\alpha = 0.01$, 30% risk variants). Type I error rates for 30% rare variants (a subset of those in Figure 9) with or without a mixture of 20% protective variants (the same ones as in Figure 9). Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure 12: Type I error rates of a combination of common and rare risk variants (\( \alpha = 0.01 \), 40% risk variants). Type I error rates for 40% combined common and rare variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure 13: Type I error rates of a combination of common and rare risk variants (\(\alpha = 0.01\), a re-sampled set of 40% risk variants). Type I error rates for a re-sampled set of 40% combined common and rare variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure 14: Type I error rates of a combination of common and rare risk variants ($\alpha = 0.01, 30\%$ risk variants). Type I error rates for 30% rare variants (a subset of those in Figure 12) with or without a mixture of 20% protective variants (the same ones as in Figure 12). Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
3.2.1.2 Power levels  We ran 500 replicates for each scenario when estimating empirical power levels which were presented and illustrated by Figures A.7-17. For comparison purposes, the standard error of each statistic at a 95% confidence level was superimposed onto the bar plots to define the upper and lower bounds of the estimated power levels. We labeled “T” or “D” at the bottom of each bar to indicate the inflation or deflation of its corresponding type I error rate, so that the calibration of the type I error rates could be reflected on the presentation and interpretation of the power levels. Power levels estimations were only done for rare variants.

Since F-SKAT’s type I error rates were way off the nominal values in quite a few scenarios, the measure of its power levels actually was not comparable to those of well-calibrated statistics. In general, both our family-based GFLMMs (flmBS and flmFR) and embedded approaches (flmBS_kernel and flmBS_burden), regardless of weighting by the B-Spline or the Fourier basis, outperformed famSKAT, despite the overlaps of the 95% confidence levels in the scenarios with lower $c$ values and the presence of protective variants, indicating a commensurate behavior of these statistics in m4- or m3- models with only rare variants.

Compared with the retrospective kernel-based approach or burden tests, neither flmBS nor flmFR exhibited an overall competitive behavior. Although our family-based GFLMMs might produce satisfying empirical power levels in the m10- or m4- models with only rare variants (Figures A.9, 16, and 17), evidence was lacking as to a more promising approach than the retrospective kernel-based approach or burden tests in most scenarios.

For the embedded approach, we were aware of the dissimilarity in power levels between the kernel-based approach and burden tests. After embedding the FLM-smoothed genotypes into the retrospective kernel-based approach, we might be able to increase power levels especially for the scenarios with fixed ORs (Figures A.8-17) or those including only risk variants with MAF-dependent ORs (as in Figures A.8 and 16). For burden tests, however, although they were competitive in fixed OR models or MAF-dependent models with only risk variants, weighting and smoothing genotypes failed to bring any advantages, or were even detrimental to power levels (as in Figures A.7 and A.8).

In summary, we estimated empirical power levels for rare variants with our proposed GFLMMs and the embedded approach, and compared them with the retrospective kernel-
based and burden test, famSKAT, and F-SKAT. Both our proposed method and embedded 
approach had an overall advantage over either famSKAT or F-SKAT, regardless of the genetic 
models we assumed. In contrast, neither flmBS nor flmFR outperformed the kernel-based or 
burden statistics. Besides, we considered it worthwhile to embed smoothed genotypes in the 
kernel-based approach, but not the burden tests, in view of the performance of flmBS_kernel 
than the corresponding kernel_BT, kernel_MB, or kernel_UW. By contrast, although the 
measure of flmBS_burden had nearly the same power levels as flmBS_kernel, we did not have 
sufficient evidence to argue that it was superior to the corresponding burden_BT, burden_MB, 
or burden_UW.
Figure 15: Power levels of rare risk variants (α = 0.01, 40% risk variants). Power levels for 40% rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. “I” or “D” at the bottom of each bar indicates the inflation or deflation of its corresponding type I error rate. Notations of statistics and scenarios are defined in Table 8.
Figure 16: Power levels of rare risk variants (α = 0.01, a distinct set of 40% risk variants). Power levels for a re-sampled set of 40% rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. “I” or “D” at the bottom of each bar indicates the inflation or deflation of its corresponding type I error rate. Notations of statistics and scenarios are defined in Table 8.
Figure 17: Power levels of rare risk variants \((\alpha = 0.01, 30\% \text{ risk variants})\). Power levels for 30\% rare risk variants (a subset of those in Figure 15) with or without a mixture of 20\% protective variants (the same ones as in Figure 15). Nominal type I error rate is 0.01. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. “I” or “D” at the bottom of each bar indicates the inflation or deflation of its corresponding type I error rate. Notations of statistics and scenarios are defined in Table 8.
3.2.2 Real Exome Chip Data Analyses on AMD

According to the simulation results, we applied our proposed GFLMM and the embedded approach to the analysis of a combination of common and rare variants with an ad hoc choice of \textit{norder} and \textit{nbasis} as discussed in Sections 3.1.7.1 and 3.1.7.2. We identified 16,844 genes containing at least two polymorphic variants. The results with at least one test statistic associated with a significant p-value were presented in Table 9. The corresponding p-values of the ordinary retrospective kernel and burden tests were also listed for a comparison purpose. We confirmed the strong association between AMD and \textit{CFH} and \textit{ARMS2}. Besides \textit{CFH} and \textit{ARMS2}, the other genes were not previously reported as AMD risk genes. Interestingly, \textit{CARD9} reached genome-wide significance only through solitary \textit{flmBS$_{kernel}$} \((p = 5.416 \times 10^{-10})\). This gene was previously shown to be involved in a pathway related to the mechanisms of sterile inflammation which played a pivotal role in the etiology of AMD [Strasser et al., 2012, Montezuma et al., 2007]. Another interesting significant association signal was related to \textit{ADAMTS6} by \textit{flmFR} \((p = 2.784 \times 10^{-6})\), a gene once reported with the expression in the retinal pigment epithelial cells, possibly indicating an influence on the etiology of AMD [Bevitt et al., 2003].

For the analysis of rare variants, we identified 14,849 genes containing at least two polymorphic variants. The results with at least one test statistic associated with a significant p-value were presented in Table 10. Unsurprisingly, \textit{CFH} and \textit{ARMS2} did not show up in the table, which was consistent with the results discussed in Section 2.2.2. We observed that \textit{ACO1} popped out only in the rare variant analysis associated with significant \textit{flmFR} \((p = 3.449 \times 10^{-17}\text{, see Table 10})\). \textit{ACO1}, also known as \textit{IRP1}, was previously associated with AMD through iron metabolism [Synowiec et al., 2012], suggesting supportive evidence of the association between AMD and \textit{ACO1}.

However, the results in Table 10 should be interpreted with caution, since all of the significant p-values were returned only by \textit{flmFR} (see the “\textit{flmFR}” column in Table 10). In addition, we also noticed that most of these genes included a relatively small number of variants ranging from 4 to 7, with the exception of \textit{CEP250} with 10 variants. This might lead to a suspicion that \textit{flmFR}, FLM-smoothed by the Fourier basis, may not be a reliable
method for the analysis of those genes that include only a few variants. Actually, smoothing genotype data might not be necessary for those genes containing the number of variants $\leq 7$, or even $\leq 10$, since the GFLMM, like the generalized FLM, assumes that genotype data within a certain genetic region on a chromosome are a realization of a stochastic process. Furthermore, if a large genetic region contains only a few variants, modeling them as a functional curve is not a reasonable analysis approach.

In our exome chip data set, we noted that one of the $CFH$ transcripts contained seven rare variants but failed to be identified in the rare variant analysis by any statistics. To further unveil these rare variants, we examined their association on an individual level by conducting the $M_{QLS}$ test [Thornton and McPeek, 2007]. According to the summaries in Table 11, we observed that none of these rare variants reached genome-wide significance, leading to a consistent conclusion drawn from the gene-based association analysis with different approaches and statistics.
Table 9: GFLMM and embedded approaches for a combination of common and rare variants

<table>
<thead>
<tr>
<th>chr</th>
<th>gene</th>
<th>nvariant</th>
<th>length (kb)</th>
<th>start (bp)</th>
<th>end (bp)</th>
<th>flmBS</th>
<th>flmFR</th>
<th>flmBS_kernel</th>
<th>flmBS_burden</th>
<th>kernel_MB</th>
<th>burden_MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr1</td>
<td>CFH</td>
<td>3</td>
<td>50</td>
<td>196621007</td>
<td>19667095</td>
<td>2.364E-07</td>
<td>2.313E-07</td>
<td>5.145E-09</td>
<td>8.490E-09</td>
<td>1.537E-08</td>
<td>1.477E-07</td>
</tr>
<tr>
<td>chr1</td>
<td>CFH</td>
<td>15</td>
<td>96</td>
<td>196621007</td>
<td>19671634</td>
<td>1.638E-08</td>
<td>2.219E-08</td>
<td>1.320E-12</td>
<td>3.266E-11</td>
<td>8.253E-12</td>
<td>6.527E-10</td>
</tr>
<tr>
<td>chr1</td>
<td>GBP3</td>
<td>4</td>
<td>16</td>
<td>89472359</td>
<td>89488549</td>
<td>2.060E-01</td>
<td>9.657E-05</td>
<td>1.250E-10</td>
<td>4.599E-06</td>
<td>7.883E-03</td>
<td>2.707E-01</td>
</tr>
<tr>
<td>chr1</td>
<td>MOB3C</td>
<td>5</td>
<td>7</td>
<td>47073386</td>
<td>47080805</td>
<td>5.016E-01</td>
<td>4.876E-01</td>
<td>5.930E-11</td>
<td>1.250E-06</td>
<td>4.405E-01</td>
<td>2.743E-01</td>
</tr>
<tr>
<td>chr1</td>
<td>MOB3C</td>
<td>5</td>
<td>9</td>
<td>47073386</td>
<td>47082563</td>
<td>5.016E-01</td>
<td>4.876E-01</td>
<td>5.930E-11</td>
<td>1.250E-06</td>
<td>4.405E-01</td>
<td>2.743E-01</td>
</tr>
<tr>
<td>chr2</td>
<td>CCDC142</td>
<td>7</td>
<td>10</td>
<td>74699958</td>
<td>74710357</td>
<td>1.844E-01</td>
<td>1.143E-06</td>
<td>5.810E-01</td>
<td>7.524E-01</td>
<td>3.415E-01</td>
<td>9.636E-01</td>
</tr>
<tr>
<td>chr5</td>
<td>ADAMTS6</td>
<td>9</td>
<td>333</td>
<td>64444562</td>
<td>64777704</td>
<td>1.952E-01</td>
<td>2.784E-06</td>
<td>4.449E-01</td>
<td>6.774E-01</td>
<td>1.142E-01</td>
<td>5.227E-01</td>
</tr>
<tr>
<td>chr8</td>
<td>GSDMD</td>
<td>6</td>
<td>5</td>
<td>144640476</td>
<td>144645231</td>
<td>8.442E-02</td>
<td>8.471E-13</td>
<td>1.356E-01</td>
<td>5.790E-01</td>
<td>1.143E-01</td>
<td>6.278E-01</td>
</tr>
<tr>
<td>chr8</td>
<td>GSDMD</td>
<td>6</td>
<td>10</td>
<td>144635556</td>
<td>144645231</td>
<td>8.442E-02</td>
<td>8.471E-13</td>
<td>1.356E-01</td>
<td>5.790E-01</td>
<td>1.143E-01</td>
<td>6.278E-01</td>
</tr>
<tr>
<td>chr8</td>
<td>PSD3</td>
<td>5</td>
<td>282</td>
<td>18384812</td>
<td>18666405</td>
<td>3.248E-04</td>
<td>7.246E-07</td>
<td>8.830E-03</td>
<td>1.456E-02</td>
<td>2.894E-04</td>
<td>5.794E-03</td>
</tr>
<tr>
<td>chr8</td>
<td>WDDYHV1</td>
<td>4</td>
<td>25</td>
<td>124428964</td>
<td>124454260</td>
<td>5.076E-01</td>
<td>1.281E-09</td>
<td>7.679E-01</td>
<td>7.586E-01</td>
<td>4.982E-01</td>
<td>8.667E-01</td>
</tr>
<tr>
<td>chr9</td>
<td>CARD9</td>
<td>5</td>
<td>10</td>
<td>139258407</td>
<td>139268133</td>
<td>2.504E-02</td>
<td>1.607E-07</td>
<td>5.416E-10</td>
<td>3.587E-05</td>
<td>1.215E-02</td>
<td>2.674E-01</td>
</tr>
<tr>
<td>chr10</td>
<td>ARMS2</td>
<td>2</td>
<td>3</td>
<td>124214178</td>
<td>124216868</td>
<td>7.274E-14</td>
<td>2.920E-12</td>
<td>3.047E-15</td>
<td>1.396E-14</td>
<td>6.536E-13</td>
<td>1.118E-10</td>
</tr>
<tr>
<td>chr11</td>
<td>OR9Q1</td>
<td>14</td>
<td>158</td>
<td>57791352</td>
<td>57949038</td>
<td>1.532E-01</td>
<td>1.575E-01</td>
<td>1.013E-10</td>
<td>7.047E-05</td>
<td>8.593E-02</td>
<td>2.733E-01</td>
</tr>
<tr>
<td>chr10</td>
<td>LIPN</td>
<td>7</td>
<td>17</td>
<td>90521162</td>
<td>90537999</td>
<td>4.346E-01</td>
<td>2.345E-06</td>
<td>6.830E-01</td>
<td>7.280E-01</td>
<td>3.653E-01</td>
<td>5.071E-01</td>
</tr>
<tr>
<td>chr10</td>
<td>PTPRE</td>
<td>4</td>
<td>38</td>
<td>129845812</td>
<td>129881464</td>
<td>9.188E-01</td>
<td>3.887E-07</td>
<td>8.629E-01</td>
<td>6.279E-01</td>
<td>5.146E-01</td>
<td>5.820E-01</td>
</tr>
<tr>
<td>chr11</td>
<td>CTTN</td>
<td>6</td>
<td>38</td>
<td>70244611</td>
<td>70282690</td>
<td>1.103E-02</td>
<td>1.556E-21</td>
<td>1.511E-09</td>
<td>9.454E-02</td>
<td>5.488E-03</td>
<td>2.188E-03</td>
</tr>
<tr>
<td>chr14</td>
<td>DNAAF2</td>
<td>5</td>
<td>10</td>
<td>50019891</td>
<td>50101948</td>
<td>1.043E-01</td>
<td>1.604E-06</td>
<td>7.573E-02</td>
<td>5.009E-02</td>
<td>6.711E-02</td>
<td>3.041E-02</td>
</tr>
<tr>
<td>chr14</td>
<td>PTGR2</td>
<td>4</td>
<td>34</td>
<td>74318620</td>
<td>74352168</td>
<td>2.114E-01</td>
<td>3.338E-01</td>
<td>5.701E-15</td>
<td>1.220E-11</td>
<td>1.338E-01</td>
<td>7.326E-02</td>
</tr>
<tr>
<td>chr14</td>
<td>PTGR2</td>
<td>4</td>
<td>34</td>
<td>74318533</td>
<td>74352168</td>
<td>2.114E-01</td>
<td>3.338E-01</td>
<td>5.701E-15</td>
<td>1.220E-11</td>
<td>1.338E-01</td>
<td>7.326E-02</td>
</tr>
<tr>
<td>chr14</td>
<td>PTGR2</td>
<td>4</td>
<td>34</td>
<td>74318546</td>
<td>74352168</td>
<td>2.114E-01</td>
<td>3.338E-01</td>
<td>5.701E-15</td>
<td>1.220E-11</td>
<td>1.338E-01</td>
<td>7.326E-02</td>
</tr>
<tr>
<td>chr17</td>
<td>UBE2Z</td>
<td>3</td>
<td>21</td>
<td>46985730</td>
<td>47006422</td>
<td>3.892E-02</td>
<td>4.073E-02</td>
<td>1.331E-07</td>
<td>5.365E-07</td>
<td>5.234E-03</td>
<td>4.513E-03</td>
</tr>
<tr>
<td>chr18</td>
<td>ZNF521</td>
<td>5</td>
<td>290</td>
<td>22641887</td>
<td>22932214</td>
<td>3.508E-02</td>
<td>6.043E-07</td>
<td>3.857E-02</td>
<td>1.839E-02</td>
<td>1.842E-03</td>
<td>2.971E-03</td>
</tr>
</tbody>
</table>

GFLMM and embedded approaches for a combination of common and rare variants. Genes were reported with at least one test statistic associated with a p-value < 2.97 x 10^{-6} (highlighted in boldface). Known AMD risk genes were highlighted in boldface.
Table 10: GFLMM and embedded approaches for rare variants

<table>
<thead>
<tr>
<th>chr</th>
<th>gene</th>
<th>nvariant</th>
<th>length (kb)</th>
<th>start (bp)</th>
<th>end (bp)</th>
<th>flmBS</th>
<th>flmFR</th>
<th>flmBS_kernel</th>
<th>flmBS_burden</th>
<th>kernel_MB</th>
<th>burden_MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>chr2</td>
<td>CCDC142</td>
<td>7</td>
<td>10</td>
<td>74699958</td>
<td>74710357</td>
<td>1.844 × 10⁻⁰¹</td>
<td>1.143 × 10⁻⁰⁶</td>
<td>5.156 × 10⁻⁰¹</td>
<td>7.451 × 10⁻⁰¹</td>
<td>3.415 × 10⁻⁰¹</td>
<td>9.636 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr7</td>
<td>ADAM22</td>
<td>4</td>
<td>269</td>
<td>87563565</td>
<td>87832204</td>
<td>1.745 × 10⁻⁰¹</td>
<td>9.168 × 10⁻⁰⁷</td>
<td>5.709 × 10⁻⁰¹</td>
<td>9.601 × 10⁻⁰¹</td>
<td>1.798 × 10⁻⁰¹</td>
<td>1.250 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr7</td>
<td>ADAM22</td>
<td>4</td>
<td>248</td>
<td>87563565</td>
<td>87811428</td>
<td>1.745 × 10⁻⁰¹</td>
<td>9.168 × 10⁻⁰⁷</td>
<td>5.709 × 10⁻⁰¹</td>
<td>9.601 × 10⁻⁰¹</td>
<td>1.798 × 10⁻⁰¹</td>
<td>1.250 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr8</td>
<td>GSDMD</td>
<td>6</td>
<td>5</td>
<td>144640476</td>
<td>144645231</td>
<td>8.442 × 10⁻⁰²</td>
<td>8.471 × 10⁻¹³</td>
<td>1.386 × 10⁻⁰¹</td>
<td>6.219 × 10⁻⁰¹</td>
<td>1.143 × 10⁻⁰¹</td>
<td>6.278 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr8</td>
<td>GSDMD</td>
<td>6</td>
<td>10</td>
<td>144635556</td>
<td>144645231</td>
<td>8.442 × 10⁻⁰²</td>
<td>8.471 × 10⁻¹³</td>
<td>1.386 × 10⁻⁰¹</td>
<td>6.219 × 10⁻⁰¹</td>
<td>1.143 × 10⁻⁰¹</td>
<td>6.278 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr9</td>
<td>ACO1</td>
<td>6</td>
<td>66</td>
<td>32384600</td>
<td>32450832</td>
<td>6.670 × 10⁻⁰¹</td>
<td>3.449 × 10⁻¹⁷</td>
<td>6.867 × 10⁻⁰¹</td>
<td>5.639 × 10⁻⁰¹</td>
<td>5.199 × 10⁻⁰¹</td>
<td>4.760 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr10</td>
<td>LIPN</td>
<td>7</td>
<td>17</td>
<td>90521162</td>
<td>90537999</td>
<td>4.346 × 10⁻⁰¹</td>
<td>2.345 × 10⁻⁰⁶</td>
<td>6.573 × 10⁻⁰¹</td>
<td>7.158 × 10⁻⁰¹</td>
<td>3.653 × 10⁻⁰¹</td>
<td>5.071 × 10⁻⁰¹</td>
</tr>
<tr>
<td>chr11</td>
<td>CTTN</td>
<td>6</td>
<td>38</td>
<td>70244611</td>
<td>70282690</td>
<td>1.103 × 10⁻⁰²</td>
<td>1.556 × 10⁻¹¹</td>
<td>2.762 × 10⁻⁰¹</td>
<td>1.229 × 10⁻⁰¹</td>
<td>5.488 × 10⁻⁰³</td>
<td>2.188 × 10⁻⁰³</td>
</tr>
<tr>
<td>chr12</td>
<td>METTL25</td>
<td>4</td>
<td>121</td>
<td>82752275</td>
<td>82873016</td>
<td>3.200 × 10⁻⁰¹</td>
<td>2.097 × 10⁻⁰⁷</td>
<td>2.726 × 10⁻⁰¹</td>
<td>3.157 × 10⁻⁰¹</td>
<td>2.084 × 10⁻⁰¹</td>
<td>5.348 × 10⁻⁰²</td>
</tr>
<tr>
<td>chr20</td>
<td>CEP250</td>
<td>10</td>
<td>57</td>
<td>34043222</td>
<td>34099803</td>
<td>1.984 × 10⁻⁰¹</td>
<td>9.127 × 10⁻⁰⁷</td>
<td>1.836 × 10⁻⁰¹</td>
<td>2.650 × 10⁻⁰¹</td>
<td>4.106 × 10⁻⁰¹</td>
<td>5.183 × 10⁻⁰¹</td>
</tr>
</tbody>
</table>

GFLMM and embedded approaches for rare variants. Genes were reported with at least one test statistic associated with a p-value < 3.37 × 10⁻⁶ (highlighted in boldface). Known AMD risk genes were highlighted in boldface.
Table 11: Single-variant association analysis for the rare risk variants within \textit{CFH}

<table>
<thead>
<tr>
<th>variant</th>
<th>gene</th>
<th>chr</th>
<th>position (bp)</th>
<th>major/minor</th>
<th>MAF</th>
<th>$M_{\text{QLS}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs141336681</td>
<td>CFH</td>
<td>chr1</td>
<td>196642221</td>
<td>[A/C]</td>
<td>0.0013</td>
<td>0.035</td>
</tr>
<tr>
<td>rs143237092</td>
<td>CFH</td>
<td>chr1</td>
<td>196695675</td>
<td>[A/C]</td>
<td>0.0006</td>
<td>0.960</td>
</tr>
<tr>
<td>rs515299</td>
<td>CFH</td>
<td>chr1</td>
<td>196706677</td>
<td>[A/C]</td>
<td>0.0026</td>
<td>0.330</td>
</tr>
<tr>
<td>rs149474608</td>
<td>CFH</td>
<td>chr1</td>
<td>196709816</td>
<td>[T/G]</td>
<td>0.0058</td>
<td>0.160</td>
</tr>
<tr>
<td>rs145975787</td>
<td>CFH</td>
<td>chr1</td>
<td>196709833</td>
<td>[T/C]</td>
<td>0.0006</td>
<td>0.770</td>
</tr>
<tr>
<td>rs534399</td>
<td>CFH</td>
<td>chr1</td>
<td>196711067</td>
<td>[A/C]</td>
<td>0.0026</td>
<td>0.330</td>
</tr>
<tr>
<td>rs35274867</td>
<td>CFH</td>
<td>chr1</td>
<td>196712596</td>
<td>[A/T]</td>
<td>0.0097</td>
<td>0.710</td>
</tr>
</tbody>
</table>

$M_{\text{QLS}}$ tests for the rare risk variants within \textit{CFH} identified in Table 10. Interestingly, none of these rare variants reached a significant p-value $< 2.09 \times 10^{-6}$ for the single-variant analysis by using $M_{\text{QLS}}$.

Similar to what we did in Chapter 3, we assessed the behavior of the statistics for the real data analysis by drawing the Q-Q plots (Figures 18-21), separated by rarity of the variants within the genes in our analysis. The Q-Q plots were also drawn with those AMD risk genes removed for comparison. We dropped 10% of the most significant results when estimating the inflation factor. For the GFLMM approach, regardless of variant rarity, the B-Spline basis led to a little higher $\lambda_{GC}$ value than the Fourier basis, but we did not observe a huge behavior discrepancy (Figures 18 and 20). For the embedded approach, no inflation in the $\lambda_{GC}$ value was observed for the analysis of both common and rare variants (Figure 19) or for the analysis of only rare variants (Figure 21). Compared with the ordinary retrospective kernel test in Chapter 2 (see Figures 7 and 8), our results suggested that the embedded approach with the FLM-smoothed genotypes might lower $\lambda_{GC}$ values by reducing the dimension of genotype data in both kernel-based approach and burden tests. After we filtered out the AMD risk genes, the behavior of the Q-Q plots was similar in terms of the $\lambda_{GC}$ values.

Particularly, for the analysis of only rare variants by using flmBS and flmFR, we noticed an obvious difference between applying the B-Spline basis and the Fourier basis (see Figure 20). For flmBS, almost all of the points were within the 95% confidence interval, while for flmFR, quite a few points were highly above the reference line, even the risk AMD genes were excluded. This might give supportive evidence to indicate that smoothing by the Fourier basis might not work well, consistent with the results summarized in Table 10.
Figure 18: Q-Q plots of the GFLMM with the B-Spline (flmBS) and the Fourier basis (flmFR) for a combination of common and rare variants. Q-Q plots for the GFLMM with the B-Spline (flmBS) and the Fourier basis (flmFR) were illustrated on the top and bottom row, respectively. Left and right column showed the genes with and without AMD risk genes, respectively. The genes contained a combination of common and rare variants. The $\lambda_{GC}$ value was estimated and superimposed onto the plots to denote genomic inflation factor, indicating how far the points were away from the reference line.
Figure 19: Q-Q plots of the embedded kernel-based approach (flmBS\_kernel) and burden tests (flmBS\_burden) with a combination of common and rare variants. Q-Q plots for the embedded kernel-based approach (flmBS\_kernel) and burden tests (flmBS\_burden) were illustrated on the top and bottom row, respectively. Left and right column showed the genes with and without AMD risk genes, respectively. The genes contained a combination of common and rare variants. \( \lambda_{GC} \) value was estimated and superimposed onto the plots to denote genomic inflation factor, indicating how far the points were away from the reference line.
Figure 20: Q-Q plots of the GFLMM with the B-Spline (flmBS) and the Fourier basis (flmFR) for only rare variants. Q-Q plots for the GFLMM with the B-Spline (flmBS) and the Fourier basis (flmFR) were illustrated on the top and bottom row, respectively. Left and right column showed the genes with and without AMD risk genes, respectively. The genes contained only rare variants. The $\lambda_{GC}$ value was estimated and superimposed onto the plots to denote genomic inflation factor, indicating how far the points were away from the reference line.
Figure 21: Q-Q plots of the embedded kernel-based approach (flmBS_kernel) and burden tests (flmBS_burden) with only rare variants. Q-Q plots for the embedded kernel-based approach (flmBS_kernel) and burden tests (flmBS_burden) were illustrated on the top and bottom row, respectively. Left and right column showed the genes with and without AMD risk genes, respectively. The genes contained only rare variants. $\lambda_{GC}$ value was estimated and superimposed onto the plots to denote genomic inflation factor, indicating how far the points were away from the reference line.
3.3 DISCUSSION

In this chapter, we have developed the GFLMM to evaluate the association between dichotomous traits and disease susceptibility genes in related subjects. It is an important extension of the generalized FLM to the analysis of pedigree data, with the capability of analyzing both common and rare variants. To estimate the parameters of our proposed model and compute gene-based test statistics, we have integrated out the random polygenic effect term, and constructed Rao’s score test statistics in the presence of nuisance parameters. The parameter estimation procedures are an extension of the methods discussed by Stanhope and Abney [2012]. Their approach is limited to analyzing single variants, while our GFLMM is able to take multiple variants into account, and construct gene-based tests.

Previously, Stanhope and Abney developed the GLOGS program to estimate the parameters of generalized linear mixed models [Stanhope and Abney, 2012]. Based on our analyses and testing, we saw several flaws in the original program, and fixed them. First, we added a step-halving process into the Gauss-Newton algorithm. Sometimes the algorithm converges in a very slow manner, even our initial input value is close to the true one. Since we desire to maximize the likelihood under the parameter estimation procedure, we do not anticipate that the likelihood estimated by the current iteration is less than the one estimated in the preceding iteration. To make the program more efficient, we set up an if condition to control whether or not to invoke the step-halving method. Second, we consider the stopping criterion in the original program controversial, because it was determined by the absolute difference of first derivatives of the parameters estimated from two adjacent iterations. To clearly describe the problem, we simplified the discussion to one dimension, and gave a counterexample in Section 3.1.4.2. In general, the first derivative describes the local monotonicity of a function at a certain point, but may not be an appropriate stopping criterion for function optimization (maximization of the likelihood in our case). Third, the original GLOGS program uses Bayesian updates to approximate the integral. However, this results in a single weight in the example data provided by Stanhope and Abney. We argue that it may be problematic to use a single weight to approximate a multi-dimensional integral of a complicated function. We instead use equal weights that no longer depend on the iterations.
to improve the original GLOGS program, resulting in the cancellation of the weights in both numerators and denominators in Equation 3.7. By using two small sample data sets, we have tested our modifications, compared, and verified the integration results with two existing R packages. We have demonstrated in Table 7 that our modifications to the original GLOGS program are necessary and very important to approximate a multi-dimensional integral.

In terms of the parameter estimation, we observed that our modified GLOGS program tended to return a lower estimate of $\sigma$ than 0.2, a constant we set for the polygenic effect in the simulation study. Those estimates in several simulation scenarios can be as small as $10^{-4}$ or $10^{-5}$, indicating a trivial polygenic effect as compared with the genetic effect. This may be caused by two reasons. On one hand, there may be still an issue in the original GLOGS program when estimating the polygenic effect, and we failed to identify it in our study. On the other hand, while simulating pedigrees, we only allowed those pedigrees with at least 1 pair of affected siblings for the evaluation of type I error rates and power levels. This ascertainment criterion may make the genetic effect dominant, and thus marginalize the polygenic effect. It may be interesting to simulate pedigrees with a relatively large polygenic effect (but not as robust as the genetic effect) in future work, and track if our modified GLOGS program can more precisely estimate the polygenic parameter.

In addition to developing the GFLMM, we have also embedded the FLM-smoothed genotypes in the retrospective kernel and burden tests. The advantage in doing this is that we do not have to be concerned about the ascertainment issue, for it is within the retrospective regression framework, and meanwhile the genotypes in the model are FLM-smoothed. It should be noted that we do not apply any weightings like we did for kernel_MB or burden_MB, since the genotypes in the model have already been weighted by the B-Spline basis.

When it comes to the choice of $norder$ and $nbasis$, several factors may have an impact on it. First, the choice may be sensitive to different statistical methods. According to the simulation study with presence of both common and rare variants, the GFLMM can well calibrate type I error rates with the choice of $norder = 4$ and $nbasis = 5$ for most scenarios, while we have to specify a higher $norder = 6$ and $nbasis = 19$ for the embedded approach to analyze the same data sets simulated by an identical genetic model. Second, the choice may
be sensitive to the rarity of variants. For the embedded approach, with the common variants added in, a dramatic change in \textit{norder} (from 4 to 6) and \textit{nbasis} (from 5 to 19) is required to well calibrate type I error rates. Third, the choice may be sensitive to the number of variants within genes. This may not be exhibited in the simulation study due to the reason that we have fixed a genetic region of interest when simulating genotype data. However, it is highlighted in the real data analysis with genes of all kinds of length, thus containing a different number of variants. From the viewpoint of dimension reduction, it does not make much sense to keep \textit{nbasis} as a constant regardless of varying gene sizes. Thus, we have come up with an ad hoc rule for choosing \textit{nbasis} in the real data analysis.

We have simulated data to compare the GFLMM, the embedded approach, the ordinary retrospective kernel and burden tests, famSKAT, and F-SKAT. Commonalities of these statistical methodologies include modeling dichotomous traits, analyzing rare variants, and accounting for related people with known pedigree structures. The advantages of our suggested GFLMM are characterized by fitting discrete genotypes with a continuous curve by using a collection of smooth basis functions, and computing the exact likelihood of the model by conducting a multidimensional integration, which is different from approximating by the hierarchical likelihood or a quadratic form with the iterative weighted least squares (IWLS) algorithm. According to the simulation results, the GFLMM outperforms famSKAT and F-SKAT, but is not better than the retrospective kernel and burden statistics in terms of power levels. Inspired by the retrospective regression that can conquer the issue of modeling ascertainment [Schaid et al., 2013], we have embedded the FLM-smoothed genotypes in Schaid’s approach. For a given variant, its genotype data are coded by the copy number of minor alleles with an element from the set of \{0, 1, 2, NA\}, which means that after smoothing the genotypes, we cannot reduce the dimension to some extent such that an average number > 2 across all of the samples.

For the real data analysis, we have confirmed the association between AMD and \textit{CFH} and \textit{ARMS2} by using the GFLMM, regardless of applying the B-Spline or the Fourier basis. With those genes reaching genome-wide significance (flmBS and flmFR in Tables 9 and 10), not all of them were also seen to be significant for the retrospective kernel and burden tests (kernel\_MB and burden\_MB in Tables 9 and 10). We have noted a few inconsistencies where
the flmFR, with FLM-smoothed by the Fourier basis, returned significant p-values associated with the genes not identified by any other statistics in Table 10. These results may provoke some suspicions regarding smoothing by the Fourier basis, although the association of ACO1 has been discussed by some other literature. For the B-Spline smoother, this potential issue is not observed. All these observations lead us to recommend not using the Fourier basis to smooth genotype data when analyzing genes containing a small number of variants. If there are only a few variants located in a long genetic region, it may not be appropriate to assume these variants are a realization of a stochastic process and thus modeling them by a functional curve may not be a good idea.

Since the approximation of the multidimensional integral calls for intensive computations, it is time-consuming to build GFLMMs and estimate relevant the p-values. In our real data analysis, the computations were conducted on a high performance computing cluster. The cluster included 32 processors, each with a speed of 1,400 MHz. A total number of 64 genes were allowed to be analyzed simultaneously. The memory limitation was 2.6 GB for each computing core, and it took 18-20 minutes on a single processor to complete estimating the p-value for each gene, which is not desirable for a genome-wide study. The program we wrote in R to compute the score test statistics includes the part to read in the Sobol points to approximate the integral and estimate the first and second derivatives (see B.5.1). It takes time to read in the large vector containing tens of thousands of Sobol points used to approximate the multidimensional integral. The subsequent for loop executes the computation of the derivatives repeatedly given certain Sobol points. Since the number of Sobol points is huge, the estimation progress is slow. To speed up the process, we may consider to rewrite the code in C or consider to achieve the summation using matrix manipulation.

The statistics derived from the embedded approach are rapid to compute, which is an advantage when analyzing genome-wide data. The embedded approach is based on Schaid’s retrospective regression framework, and utilizes FLM-smoothed genotype data, which is a combination of retrospective and functional data analysis. However, we should proceed with caution. We have used the embedded approach to discover the association signal of CFH, ARMS2, and CARD9, which we are able to find some previous publications to support. However, besides these promising findings, the association of the other genes have never
been reported before. Given their relatively small number of variants, it reminds us to have a second thought about the smoothing and reducing the dimension of the small size genes containing only a few variants. As introduced early in Chapter 1, the idea behind the GFLMM comes from treating genotyping data as a realization of a stochastic process that varies along the chromosome. If a relative long genetic region contains very few variants, this realization may be questionable, which means that we may lack enough data points to fit a reasonable functional curve.

In Chapter 2, we have discussed a potential issue related to the smaller p-values in the kernel-based approach than in the burden tests. In this chapter, we have investigated the behavior of the statistics derived from the GFLMM and the embedded approach by drawing Q-Q plots. It is not beyond our expectation that the GFLMM or the embedded burden tests would not give rise to the obvious discrepancy of the p-values between the B-Spline basis and the Fourier basis, or the kernel-based and the burden statistics. Compared with the $\lambda_{GC}$ values estimated by the retrospective kernel-based approach and burden tests, the embedded methods have produced lower estimates, indicating that the embedment can attenuate the inflation related to the kernel-based test via reducing the dimension of genotype data, as long as we can find an optimal pair of $n_{order}$ and $n_{basis}$. 
4.0 FUTURE WORK

In this chapter, we talk about future studies we consider worthwhile to follow this dissertation.

Through the discussions in Chapter 2, we discovered an issue related to the retrospective kernel regression that kernel-based approach resulted in higher $\lambda_{GC}$ values than burden tests. By using our embedded approaches, the issue was attenuated because we managed to reduce the dimension of the genotype data from the number of variants to the number of basis functions we have specified. In our simulation studies (see 3.1.6), we chosen a null gene with a length of 14 kb, a median gene size. To better demonstrate the issue and our solutions, we plan to simulate more scenarios by varying gene sizes to include a different number of variants. The purpose for conducting this is to further investigate how GFLMMs perform in the genes containing different number of variants calling for a dimension reduction, especially by using the Fourier basis. For large-size genes, a reasonable choice may be 27 kb, a reported mean size of the genes from human genome [Lander et al., 2001]. For small-size genes, we can halve the length of 14 kb to include less variants in analysis. The simulation steps would be the same as we discussed in 3.1.6. We would mainly focus on the kernel-based approach and evaluate statistics and associated p-values for the ordinary retrospective regression approaches, the embedded approach we have suggested, and famSKAT as well.

Although we have presented the GFLMM allowing for covariates other than genetic effect, it is tricky and time-consuming to estimate parameters by using the GLOGS program in presence of other covariates to be adjusted in the model. The original GLOGS program limits itself to $\leq 4$ covariates, and invokes different functions when estimating the models containing different number of covariates. To modify more functions in the GLOGS program to well handle the models allowing for covariates is part of our future work.
As previously discussed in Section 3.1.6.6, our GFLMM relies on two parameters, \( n\text{order} \) and \( nbasis \). On one hand, we intend to reduce the dimension of genotype data controlled by \( nbasis \). On the other hand, we have to choose a certain pair of \( n\text{order} \) and \( nbasis \) to well calibrate type I error rates while reaching high power levels. Per the results for our simulation studies, the choice of \( n\text{order} \) and \( nbasis \) may be different between the analysis of common and rare variants and the analysis of only rare variants. To seek an optimal pair of \( n\text{order} \) and \( nbasis \), we need to simulate more scenarios with different pairs of \( n\text{order} \) and \( nbasis \). Based on our previous discussions and Fan’s early publication [Fan et al., 2014], we can set \( 4 \leq n\text{order} \leq 6 \) and \( 8 \leq nbasis \leq 21 \). For real data analysis, the choice may be more complicated. We can develop a more sophisticated ad hoc plan according to the size or the number of variants in all of the genes in human genome. Ideally, given an \( n\text{order} \), \( nbasis \) may be a function of the number of variants with a certain gene. Alternatively, we can also use a genotype data set related to all AMD risk genes as previously discussed and verified, and analyze by our GFLMM and embedded approach with different sets of \( n\text{order} \) and \( nbasis \) to see whether or not the results are significant as expected.

We recognize that the number of iterations we simulated for evaluating type I error rates was 1,000, not large enough to investigate the behavior of the statistics under extremely small nominal significance levels. For the GFLMM, the computation procedures are intensive and time-consuming, preventing us from running like 100,000 iterations. On the contrary, the computation can be rapid for the embedded approach, famSKAT, and F-SKAT, which makes it possible for us to repeat more iterations to simulate type I error rates under nominal \( \alpha = 0.005 \) or \( \alpha = 0.0001 \).

Although our study is mainly focused on rare variants, to make our arguments more persuasive and demonstrative, we may simulate some scenarios with common variants only. The genetic model (see Equation 3.22) we considered to assign the dichotomous trait in the simulation study is appropriate for rare variants or a combination of common and rare variants. To simulate the scenarios with only common variants, we have to use a different strategy to assign a disease trait.

To push forward our discussions even further, we can derive the statistic from likelihood ratio test (LRT). In our study, we intended to compute the exact marginal likelihood by in-
tegrating out the random term, and then construct the score test statistics in the GFLMM. The theoretical computation is precise, while to get around the integral takes time and effort. The logic behind LRT in our case is to approximate the likelihood of the GFLMM by using some numerical algorithms, and maximize the likelihood of the model with genetic effect and the one under the null hypothesis. The parameter estimation and likelihood optimization can be efficiently completed by several existing R packages or functions like “lme4” (https://cran.r-project.org/web/packages/lme4/index.html) and the “glmmPQL” function in “MASS” (https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/glmmPQL.html).
APPENDIX A

ADDITIONAL SIMULATION PLOTS

The following plots illustrate the type I error rates and the empirical power levels simulated under a nominal $\alpha = 0.05$. In general, the patterns of these plots are similar to those depicted in Section 3.2.1.
Figure A.1: Type I error rates of rare risk variants ($\alpha = 0.05$, 40% risk variants). Type I error rates for 40% rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure A.2: Type I error rates of rare risk variants ($\alpha = 0.05$, a re-sampled set of 40% risk variants). Type I error rates for a re-sampled set of 40% rare variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure A.3: Type I error rates of rare risk variants ($\alpha = 0.05$, 30% risk variants). Type I error rates for 30% rare risk variants (a subset of those in Figure A.1) with or without a mixture of 20% protective variants (the same ones as in Figure A.1). Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure A.4: Type I error rates of a combination of common and rare risk variants ($\alpha = 0.05$, 40% risk variants). Type I error rates for 40% combined common and rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure A.5: Type I error rates of a combination of common and rare risk variants (α = 0.05, a re-sampled set of 40% risk variants). Type I error rates for a re-sampled set of 40% combined common and rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure A.6: Type I error rates of a combination of common and rare risk variants (α = 0.05, 30% risk variants). Type I error rates for 30% rare risk variants (a subset of those in Figure A.4) with or without a mixture of 20% protective variants (the same ones as in Figure A.4). Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. Notations of statistics and scenarios are defined in Table 8.
Figure A.7: Power levels of rare risk variants ($\alpha = 0.05$, 40% risk variants). Power levels for 40% rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. “I” or “D” at the bottom of each bar indicates the inflation or deflation of its corresponding type I error rate. Notations of statistics and scenarios are defined in Table 8.
Figure A.8: Power levels of rare risk variants (\(\alpha = 0.05\), a re-sampled set of 40% risk variants). Power levels for a re-sampled set of 40% rare risk variants with or without a mixture of 20% protective variants. Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. “I” or “D” at the bottom of each bar indicates the inflation or deflation of its corresponding type I error rate. Notations of statistics and scenarios are defined in Table 8.
Figure A.9: Power levels of rare risk variants ($\alpha = 0.05$, 30% risk variants). Power levels for 30% rare risk variants (a subset of those in Figure A.7) with or without a mixture of 20% protective variants (the same ones as in Figure A.7). Nominal type I error rate is 0.05. OR for risk variants is 2.0 or MAF-dependent, and OR for protective variants is 0.8 or MAF-dependent. “I” or “D” at the bottom of each bar indicates the inflation or deflation of its corresponding type I error rate. Notations of statistics and scenarios are defined in Table 8.
APPENDIX B

R CODES: SIMULATION STUDIES AND REAL DATA ANALYSES

B.1 GENERATE PEDIGREE STRUCTURE

We wrote a function called "simuPed" in R to generate related individuals with known pedigree structures for the simulation study in Section 3.1.6. The function took two arguments: \textit{Nped} specified number of pedigrees we desired to generate, and \textit{Maxsib} specified the maximum siblings within a certain generation.

```r
# Simulate pedigree structure
# Compute kinship coefficients
# Generate a seed file including 1,000 numbers
# Clean the haplotype data
rm(list=ls())
require(kinship2)
DIR <- "~/Volumes/Macintosh HD 2/Dissertation/Programs/Simulation/"
setwd(DIR)
kinMatName <- "simu_chol_cor.dat"
ITER <- 1000 # number of iterations
snpS <- 1000 # starting point of a gene
snpL <- 14*1000 # median 14 kb, mean 27 kb
RARE <- 0.05 # filter for rare variants
CAUSAL <- 0.4 # percent of the variants that are deleterious
PROTECT <- 0.2 # percent of the variants that are protective
ORrisk <- 2.0 # risk variants
ORprotect <- 0.8 # protective variants

# write a function to generate pedigree structure
simuPed <- function(Nped, Maxsib=7) {
  if (missing(Nped) | Nped %% 1 !=0)
    stop("missing or an invalid number of pedigrees")
  if (missing(Maxsib) | Maxsib %% 1 !=0 | Maxsib < 2)
    stop("missing or an invalid number of sibling size")
  ...
```

92
if (Maxsib >= 15)
    stop("sibling size too large")
if (Nped >= 1500)
    warning("specify a large number of pedigrees or sibling size, simulation may be slow", immediate.= TRUE)
### parameters input for simulations ###
N <- Nped # number of pedigrees, must be greater than 1#
Nparent <- 2*N # number of parents#
NEXTProb <- 1/4 # proportion of the third generations#
### determine the progeny size by a negative distribution ###
r <- c(2:Maxsib); n <- 2.84; P <- 0.93; Q <- 1+P
Pr <- choose(n+r -1,r)*(P/Q)^r*Q^(-n)
SIBSIZE <- matrix(c(r,Pr), byrow = TRUE, nrow = 2)
### choose sibship size ###
sibsize <- sample(SIBSIZE[,1], N, replace= TRUE, prob= SIBSIZE[,2])
pedsiz e <- sibsize + 2 # add in parents#
### generate data under simulation model ###
Ped <- c(rep(1:N,pedsiz e))
Per <- unlist(sapply(pedsiz e,seq))
simuData <- data.frame(cbind(Ped,Per))
colnames(simuData) <- c("Ped","Per")
NN <- nrow(simuData) # total number of subjects#
simuData$Father <- 1
simuData$Mother <- 2
simuData$Sex <- 1 # male#
simuData[simuData$Per == 2, "Sex"] <- 2 # female founder#
simuData[simuData$Per == 1 | simuData$Per == 2, "Father"] <- 0
simuData[simuData$Per == 1 | simuData$Per == 2, "Mother"] <- 0
simuData[simuData$Per != 1 & simuData$Per != 2, "Sex"] <- sample(c(1,2), NN-Nped*2, replace = TRUE)
### generate the third generations ###
simuData$index <- 1:NN
subFamID <- simuData$[simuData$Father != 0, "index"]
subFamID <- sample(subFamID, size = floor(length(subFamID)*NEXTProb), replace= FALSE)
if (length(subFamID) != 0) {
    nextGen <- simuData$[simuData$index %in% subFamID, ]
    nextGen$couple <- (NN+1):nrow(nextGen)-1
    nextGen$Sex2 <- as.numeric(paste0("2",nextGen$couple))
    nextGen$Sex <> ifelse(nextGen$Sex == 1, 2, 1)
    nextGen[.(c("Ped","Per","Father","Mother","Sex"))]
    nextGen2 <- nextGen[.(c("Ped","couple","Sex2"))]
    nextGen2$Father <- nextGen2$Mother <- 0
colnames(nextGen2) <- c("Ped","Per","Sex","Father","Mother")
simuData2 <- rbind(nextGen1, nextGen2)
simuData2$matchID <- rep(1:nrow(nextGen1), times = 2)
simuData2 <- simuData2$[order(simuData2$matchID),]
}
N2 <- length(unique(simuData2$matchID))
sibsize2 <- sample(SIBSIZE[,1], N2, replace= TRUE, prob= SIBSIZE[,2])
matchID <- c(rep(unique(simuData2$matchID),sibsize2))
Per2 <- unlist(sapply(sibsize2, seq))
Per2 <- (NN+nrow(nextGen2)+length(matchID)-1)
Per2 <- as.numeric(paste0("3",Per2))
gen3Data <- data.frame(cbind(matchID,Per2))
gen3Data$Ped <- 0
gen3Data$Father <- gen3Data$Mother <- -1
gen3Data$Sex <- sample(c(1,2), nrow(gen3Data), replace= TRUE)
colnames(gen3Data)[colnames(gen3Data) == "Per2"] <- "Per"
simuData2 <- rbind(simuData2, gen3Data)
simuData2 <- simuData2$[order(simuData2$matchID),]
for (j in matchID) {
```r
  tmp <- simuData2[matchID == j,]
  tmp <- tmp[1:2,]
  ped <- tmp$Ped
  father <- tmp$Sex == 1, Per]
  mother <- tmp$Sex == 2, Per]
  simuData2[matchID == j & simuData2$Ped == 0, "Ped"] <- ped
  simuData2[matchID == j & simuData2$Father == -1, "Father"] <- father
  simuData2[matchID == j & simuData2$Mother == -1, "Mother"] <- mother
}
simuData2$matchID <- NULL; simuData$index <- NULL
simuData <- rbind(simuData, simuData2)
simuData <- simuData[!duplicated(simuData),]
simuData <- simuData[order(simuData$Ped, simuData$Per),]
} else {
simuData$index <- NULL
}
return(simuData)
```

# ###########################################################
## generate the pedigree structure and write out
# ###########################################################

SEED <- 25
set.seed(SEED)
pedstr <- simuPed(Nped = 25, Maxsib = 7)
pedstr$ID <- 1:nrow(pedstr)
save(pedstr, file = "pedstr.RData")

# ###########################################################
## compute kinship coefficients and write out
# ###########################################################

kinObj <- pedigree(id = pedstr$Per, dadid = pedstr$Father, momid = pedstr$Mother, sex = pedstr$Sex, famid = pedstr$Ped)
kinMatrix <- kinship(kinObj)
S <- 2*kinMatrix
S <- as.matrix(S)
save(S, file = "kinCoef.RData")
cholS <- chol(S) # cholesky decomposition of kinship matrix #
cholS <- t(cholS) # lower triangle matrix #
colnames(cholS) <- rownames(cholS) <- pedstr$ID
write.table(cholS, kinMatName, sep = "\t", quote = FALSE, col.names = TRUE, row.names = TRUE)

# ###########################################################
## generate a seed file including 1000 numbers
# ###########################################################

seedfile <- sample.int(1e8, ITER, replace = FALSE)
save(seedfile, file = "seedFile.RData")

# ###########################################################
## clean the haplotype data generated by Qi
# ###########################################################

## read in haplotype data simulated by Qi ##
hap <- read.table("Haplotype.txt", header = FALSE, sep = " ", stringsAsFactors = FALSE)
snp <- read.table("SNPInfo.txt", header = TRUE, sep = " ", stringsAsFactors = TRUE)
subsnp <- snp
subsnp$rs <- paste0("rs", subsnp$rs)
colnames(hap) <- subsnp$rs

## get a subregion with a length of 14 kb, and save the subsets ##
```
```
subsn <- subsnp[subsn$position >= snpS & subsnp$position <= snpS+snpL, ]
subhap <- hap[,subsn$rs]

## filter out common variants ##
maf <- colMeans(subhap)
subsn <- subsnp[maf > 0 & maf < RARE, ]
subhap <- subhap[,subsn$rs]

save(subhap, file= "Haplotype_sub.RData")
save(subsn, file= "SNPInfo_sub.RData")

## sample ~% causal variants ##
NRhap <- nrow(subhap)
NChap <- ncol(subhap)
Ncausal <- ceiling(CAUSAL*NChap)
Nprotect <- ceiling(PROTECT*NChap)

## 1. 40 % causative with fixed OR ##
logORrisk1 <- log(c(rep(ORrisk, Ncausal), rep(1, NChap-Ncausal)))
logORrisk1 <- sample(logORrisk1)
save(logORrisk1, file= "logOR_2_40.RData")

## 2. 40 % causative 20 % protective with fixed OR ##
logORrisk2 <- logORrisk1
index0 <- which(logORrisk1 == 0)
indexRisk <- which(logORrisk1 > 0)
sampleIndex0 <- sample(index0, Nprotect, replace= FALSE)
logORrisk2[sampleIndex0] <- log(ORprotect)
save(logORrisk2, file= "logOR_2_40_08_20.RData")

## 3. 40 % causative with MAF dependent OR ##
submaf <- colMeans(subhap)
submaf <- as.vector(submaf)
# -----------------------------------------------------------#
logORmaf <- log(10)/4 * abs(log10(submaf))
logORrisk3 <- logORmaf
logORrisk3[index0] <- 0
save(logORrisk3, file= "logOR_m_40.RData")

logORrisk3_m3 <- logORmaf3
logORrisk3_m3[index0] <- 0
save(logORrisk3_m3, file= "logOR_m3_40.RData")

logORrisk3_m4 <- logORmaf4
logORrisk3_m4[index0] <- 0
save(logORrisk3_m4, file= "logOR_m4_40.RData")

#logORrisk4_m5 <- log(5)/4 * abs(log10(submaf))
#save(logORrisk4_m5, file= "logOR_m5_40.RData")
# -----------------------------------------------------------#

## 4. 40 % causative 20 % protective with MAF dependent OR ##
# -----------------------------------------------------------#
logORrisk4 <- logORrisk2
indexPos <- which(logORrisk2 > 0)
logORrisk4[indexPos] <- logORmaf[indexPos]
indexNeg <- which(logORrisk2 < 0)
logORrisk4[indexNeg] <- -logORmaf[indexNeg]
save(logORrisk4, file= "logOR_m_40_m_20.RData")
```
logORrisk4_m3 <- logORrisk2
indexPos <- which(logORrisk2 > 0)
logORrisk4_m3[indexPos] <- logORmaf3[indexPos]
indexNeg <- which(logORrisk2 < 0)
logORrisk4_m3[indexNeg] <- -logORmaf3[indexNeg]
save(logORrisk4_m3, file = "logOR_m3_40_m3_20.RData")

logORrisk4_m4 <- logORrisk2
indexPos <- which(logORrisk2 > 0)
logORrisk4_m4[indexPos] <- logORmaf4[indexPos]
indexNeg <- which(logORrisk2 < 0)
logORrisk4_m4[indexNeg] <- -logORmaf4[indexNeg]
save(logORrisk4_m4, file = "logOR_m4_40_m4_20.RData")

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

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

## select 1/4 causative variants from 40 % ones ##
Nnull <- floor(1/4 * Ncausal)
nullVariant <- sample(indexRisk, Nnull)

## 5. 30 % causative with fixed OR, subset ##
logORrisk5 <- logORrisk1
logORrisk5[nullVariant] <- 0
save(logORrisk5, file = "logOR_2_30.RData")

## 6. 30 % causative 20 % protective with fixed OR, subset ##
logORrisk6 <- logORrisk2
logORrisk6[nullVariant] <- 0
save(logORrisk6, file = "logOR_2_30_08_20.RData")

## 7. 30 % causative with MAF dependent OR, subset ##
logORrisk7 <- logORrisk3
logORrisk7[nullVariant] <- 0
save(logORrisk7, file = "logOR_m_30.RData")

logORrisk7_m3 <- logORrisk3_m3
logORrisk7_m3[nullVariant] <- 0
save(logORrisk7_m3, file = "logOR_m3_30.RData")

logORrisk7_m4 <- logORrisk3_m4
logORrisk7_m4[nullVariant] <- 0
save(logORrisk7_m4, file = "logOR_m4_30.RData")

## 8. 30 % causative 20 % protective with MAF dependent OR, subset ##
logORrisk8 <- logORrisk4
logORrisk8[nullVariant] <- 0
save(logORrisk8, file = "logOR_m_30_m_20.RData")

logORrisk8_m3 <- logORrisk4_m3
logORrisk8_m3[nullVariant] <- 0
save(logORrisk8_m3, file = "logOR_m3_30_m3_20.RData")

logORrisk8_m4 <- logORrisk4_m4
logORrisk8_m4[nullVariant] <- 0
save(logORrisk8_m4, file = "logOR_m4_30_m4_20.RData")

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

## 9. Simulate an uncorrelated data set ##
Ncausal <- length(which(logORrisk1 > 0))
logORrisk9 <- log(c(rep(ORrisk, Ncausal), rep(1, NChap - Ncausal)))
logORrisk9 <- sample(logORrisk9)
save(logORrisk9, file = "logOR_2_40_new.RData")

## 10. 40 % causative 20 % protective with fixed OR ##
B.2 SIMULATE GENOTYPES AND ASSIGN TRAITS

We ran the following program on a high performance computing cluster to simulate genotype data in Section 3.1.6. For each iteration, it assigned haplotypes to the founders in a pedigree, transmitted to offspring, converted to genotypes, and smoothed the data by invoking basis functions. It also assigned traits to all simulated individuals and generated input files for
the GLOGS program.

```r
### This program will generate smoothed genotype data for type I error and power
estimations.
rm(list=ls())

args= (commandArgs(TRUE))
if (length(args) == 0) {
  print("No arguments supplied.")
  seedIndex = 1
} else {
  for (i in 1:length(args)) {
    eval(parse(text=args[[i]]))
  }
}

RpkgDIR <- "/home/dweeks/yij5/Dissertation/Simulation/Rpackage/
require("MASS",lib.loc= RpkgDIR)
require("fda",lib.loc= RpkgDIR)

DIRcurrent <- paste0("/home/dweeks/yij5/Dissertation/Simulation/Iterations/iter_",seedIndex,"/")
DIRinput <- "/home/dweeks/yij5/Dissertation/Simulation/Input/
DIRoutput <- "/home/dweeks/yij5/Dissertation/Simulation/Output/

## define file names ##
phenoName <- "simu_data.dat"
uMatFile <- "u_matrix_star.dat"
ORfileName <- "logOR_m3_40_new.RData"

## define parameters ##
PREV <- 0.05 # disease prevalence
POLYGEN <- 0.2 # polygenic effect

order <- 4
basis <- 5 # number of basis functions
beta0 <- log(PREV/(1 - PREV)) # intercept adjusting population prevalence

## read in haplotype data simulatd by Qi ##
setwd(DIRinput)
hap <- get(load("Haplotype_sub.RData"))
snp <- get(load("SNPInfo_sub.RData"))
kinCoef <- get(load("kinCoef.RData"))
seeds <- get(load("seedFile.RData"))
SEED <- seeds[seedIndex]
set.seed(SEED)

## sample causal variants ##
NRhap <- nrow(hap)
NChap <- ncol(hap)
logORrisk <- get(load(ORfileName))

## Simulate and acertain 25 pedigrees with at least 2 cases. ##
## Transmit haplotypes to offspring. 
## Convert haplotypes to genotypes, and assign phenotypes. 

## read in the pedigree structure ##
pedstr <- get(load("pedstr.RData"))
pedstr$trait <- -1 # preassign the trait 
family <- unique(pedstr$Ped)
```
## identify pedigree "founders" and sample haplotypes ##

```R
pedstr$m2 <- pedstr$m1 <- 0
founderID <- pedstr[pedstr$Father == 0, "ID"]
Nfounder <- length(founderID)
pedstr[pedstr$ID %in% founderID, "m1"] <- sample(c(1:NRhap), Nfounder, replace = TRUE)
pedstr[pedstr$ID %in% founderID, "m2"] <- sample(c(1:NRhap), Nfounder, replace = TRUE)
```

## assign haplotypes and genotypes ##

```R
powerPed <- data.frame()
powerGeno <- data.frame()
NCgeno <- NChap
system.time(
  for (f in family) {
    substr <- pedstr[pedstr$Ped == f, ]
    founderID <- substr[substr$Father == 0, "ID"]
    Nfounder <- length(founderID)
    nonFounderID <- substr[substr$Father != 0, "ID"]
    index <- 0 # ascertainment index #
    while (index == 0) {
      ## sample haplotypes for founders ##
      substr[substr$ID %in% founderID, "m1"] <- sample(c(1:NRhap), Nfounder, replace = TRUE)
      substr[substr$ID %in% founderID, "m2"] <- sample(c(1:NRhap), Nfounder, replace = TRUE)
      ## sample haplotypes and convert to genotypes ##
      for (j in nonFounderID) {
        fatherPer <- substr[substr$ID == j, "Father"]
        fatherPool <- as.vector(substr[substr$Per == fatherPer, c("m1", "m2")])
        substr[substr$ID == j, "m1"] <- sample(fatherPool, 1, replace = TRUE)
        motherPer <- substr[substr$ID == j, "Mother"]
        motherPool <- as.vector(substr[substr$Per == motherPer, c("m1", "m2")])
        substr[substr$ID == j, "m2"] <- sample(motherPool, 1, replace = TRUE)
      }
      NRsubstr <- nrow(substr)
      geno <- hap[substr$m1, ] + hap[substr$m2, ]
      colnames(geno) <- colnames(hap)
      ## simulate polygenic effect ##
      muMean <- rep(0, NRsubstr)
      kinStart <- substr[1, "ID"]
      kinEnd <- substr[NRsubstr, "ID"]
      corSigma <- kinCoef[kinStart:kinEnd, kinStart:kinEnd]
      a <- mvnorm(n = 1, mu = muMean, Sigma = corSigma, tol = 1e-16, empirical = FALSE, EISPACK = FALSE)
      ## compute the Bernoulli probability of affected status conditional on genotype ##
      ## based on Schaid's proposed model ##
      for (k in 1:NRsubstr) {
        g <- geno[k, ]
        A <- beta0 + sum(logORrisk*g) + POLYGEN*a[k]
        mu <- exp(A) / (1 + exp(A))
        substr[k,"trait"] <- sample(c(1,0), 1, replace = FALSE, prob = c(mu, 1-mu))
      }
      ## Ascertain at least 2 affected siblings in the pedigree ##
      affectMomID <- substr[substr$trait == 1, "Mother"]
      dup <- affectMomID[duplicated(affectMomID)]
      if (sum(dup) != 0) {
        powerPed <- rbind(powerPed, substr)
        powerGeno <- rbind(powerGeno, geno)
        index <- 1
      }
    }
  }
)
```

99
powerPed$m1 <- powerPed$m2 <- NULL

# Compute MAF, and keep the markers in (0, 0.05)
maf <- colSums(powerGeno) / nrow(powerGeno)
powerGeno <- powerGeno[, maf > 0]
# add in if condition here: no markers selected#

# Invoke the GLOGS program in R.

## input files for GLOGS ##
setwd(DIRcurrent)
phenoCov <- powerPed[, c("ID","trait")]
write.table(phenoCov, phenoName, sep = "\t", quote = FALSE, col.names = FALSE, row.names = FALSE)

## invoke GLOGS in R ##
## estimate initial non-genotype covariate effect ##
NRpowerPed <- nrow(powerPed)
glmEst <- glm(formula = powerPed$trait ~ 1, family = "binomial")
initial <- summary(glmEst)$coefficients[1]

## Weight the variants by the beta-smooth only model

pos <- snp[, snp$rs %in% colnames(powerGeno), "position"] # already sorted#
nsample <- nrow(powerGeno)
nsnp <- ncol(powerGeno)

definition: normalize position ##
if (max(pos) > 1) {
pos <- (pos - min(pos)) / (max(pos) - min(pos))
}

## generate basis functions ##
betabasis <- create.bspline.basis(norder = order, nbasis = basis)
fourierbasis <- create.fourier.basis(c(0,1), nbasis = basis)
B <- eval.basis(pos, betabasis)
FR <- eval.basis(pos, fourierbasis)
powerGeno <- as.matrix(powerGeno)
UJ_power <- powerGeno %*% B
UJ_power_FR <- powerGeno %*% FR

## Output the pedigree structure with simulated trait.
## Output the two unweighted genotype files.

save(powerGeno, file = paste0("powerGeno_", seedIndex, ".RData"))
save(UJ_power, file = paste0("UJ_power_BS_", seedIndex, ".RData"))
save(UJ_power_FR, file = paste0("UJ_power_FR_", seedIndex, ".RData"))
write.table(initial, file = paste0("initial_", seedIndex, ".txt"), row.names = FALSE, col.names = FALSE, quote = FALSE)
save(powerPed, file = paste0("pedstrTrait_", seedIndex, ".RData"))
B.3 COMPUTE STATISTICS AND P-VALUES IN THE SIMULATION STUDY

B.3.1 GFLMM

We ran the following program on a high performance computing cluster to estimate type I error rates and power levels in Section 3.1.6. For each iteration, it read in the pedigree and genotype data, approximated the integral by using Sobol points produced by the GLOGS program, computed Rao’s score test statistics, and estimated and output p-values.

```r
### This program will estimate type I error rates and power levels for GFLMM.

rm(list=ls())

args = commandArgs(TRUE)
if (length(args) == 0) {
  print("No arguments supplied.")
  seedIndex = 1
} else {
  for (i in 1:length(args)) {
    eval(parse(text=args[[i]]))
  }
}

DIRcurrent <- paste0("/home/dweeks/yij5/Dissertation/Simulation/Iterations/iter_",seedIndex,
        "/")
DIRinput <- "/home/dweeks/yij5/Dissertation/Simulation/Input"
DIRoutput <- "/home/dweeks/yij5/Dissertation/Simulation/Output/"
RpkgDIR <- "/home/dweeks/yij5/Dissertation/Simulation/Rpackage/"
require("MASS", lib.loc= RpkgDIR)

# define file names#
phenoName <- "simu_data.dat"
uMatFile <- "u_matrix_star.dat"

# define parameters#
Z <- 1 # covariate vector
C <- 400000 # number of cubature sizes
W <- 1/C # equal weights

setwd(DIRinput)
uMat <- scan(uMatFile, sep = "", quiet= TRUE)
setwd(DIRcurrent)
Gpower <- get(load(paste0("UJ_power_BS_", seedIndex, ".RData")))
Gpower_FR <- get(load(paste0("UJ_power_FR_", seedIndex, ".RData")))
phenoCov <- read.table(phenoName, header= FALSE, sep= "\t", stringsAsFactors= FALSE)
colnames(phenoCov) <- c("ID","trait")
NRpowerPed <- nrow(phenoCov)
```

101
### Get the final estimated parameters and compute p-value.

```r
#rmCMD <- paste("rm -f", uMatFile, sep = " ")
#system(rmCMD, wait= FALSE)
estParameters <- scan("estParameters.txt", quiet= TRUE)
estBetaNull <- estParameters[1]
estSigma <- estParameters[2]
lnL <- estParameters[3]
y <- phenoCov$trait
numG_power <- ncol(Gpower) # number of variants #
numG_power_FR <- ncol(Gpower_FR)
Gpower <- as.data.frame(Gpower)
Gpower_FR <- as.data.frame(Gpower_FR)
Gpower$Z <- 1
Gpower_FR$Z <- 1
S_theta_numerator_power <- S_theta_numerator_power_FR <- 0
I_theta_numerator_power <- I_theta_numerator_power_FR <- 0
S_theta_denominator <- exp(lnL - log(W))
I_theta_denominator <- S_theta_denominator

for (c in 1:C) {
  ## compute p ##
  u_vec <- uMat[((c-1)*NRpowerPed+1):(c* NRpowerPed)]
  B <- estBetaNull + estSigma * u_vec
  p <- exp(B) / (1 + exp(B))
  ## compute likelihood given c: exp(l_a_c) ##
  l_c <- sum(y* log(p) + (1-y)* log(1-p))
  exp_l_c <- exp(l_c)
  Gpower$u_vec <- u_vec
  Gpower_FR$u_vec <- u_vec
  t_z_theta_power <- as.matrix(Gpower)
  t_z_theta_power_FR <- as.matrix(Gpower_FR)
  y_p <- matrix((y-p), nrow = 1, ncol = NRpowerPed, byrow = TRUE)
  Dl_theta_power <- (y_p) %*% t_z_theta_power
  Dl_theta_power_FR <- (y_p) %*% t_z_theta_power_FR
  ## assume equal weights ##
  S_theta_numerator_power <- S_theta_numerator_power + Dl_theta_power * exp_l_c
  S_theta_numerator_power_FR <- S_theta_numerator_power_FR + Dl_theta_power_FR * exp_l_c
  ## D2l_theta = sum ((p^2-p)*z*z') ##
  z_theta_power <- t(t_z_theta_power)
  z_theta_power_FR <- t(t_z_theta_power_FR)
  row_z_theta_power <- nrow(z_theta_power)
  row_z_theta_power_FR <- nrow(z_theta_power_FR)
  p2_p <- p^2 - p
  p2_p_power <- matrix(rep(p2_p, row_z_theta_power), nrow = row_z_theta_power, ncol = NRpowerPed, byrow= TRUE)
  p2_p_power_FR <- matrix(rep(p2_p, row_z_theta_power_FR), nrow = row_z_theta_power_power, ncol = NRpowerPed, byrow= TRUE)
}````
D2l_theta_power <- (p2_p*z_theta_power) %*% t_z_theta_power
D2l_theta_power_FR <- (p2_p*z_theta_power_FR) %*% t_z_theta_power_FR
I_theta_numerator_power <- I_theta_numerator_power + (D2l_theta_power + t(Dl_theta_power))*exp_l_c
I_theta_numerator_power_FR <- I_theta_numerator_power_FR + (D2l_theta_power_FR + t(Dl_theta_power_FR))*exp_l_c

## write a function to compute p-value of the score test ##
scoreP <- function(S_theta_numerator, I_theta_numerator, numG) {
  S_theta <- S_theta_numerator / S_theta_denominator
  I_theta <- -(I_theta_numerator/I_theta_denominator - t(S_theta)) %*% S_theta
  S_gamma <- S_theta[1:numG, drop = FALSE]
  I_gamma_gamma <- I_theta[1:numG,1:numG, drop = FALSE]
  I_gamma_eta <- I_theta[1:numG,(numG+1):ncol(I_theta), drop = FALSE]
  I_eta_gamma <- I_theta[(numG+1):ncol(I_theta),1:numG, drop = FALSE]
  inv_I_eta_eta <- ginv(I_eta_eta)
  I_gamma <- I_gamma_gamma - I_gamma_eta %*% inv_I_eta_eta %*% I_eta_gamma
  inv_I_gamma <- ginv(I_gamma)
  R <- S_gamma %*% inv_I_gamma %*% t(S_gamma)
  R <- c(R)
  pValue_R <- pchisq(R, df = numG, lower.tail = FALSE)
  return(pValue_R)
}

pValue_R_power <- scoreP(S_theta_numerator_power, I_theta_numerator_power, numG_power)
pValue_R_power_FR <- scoreP(S_theta_numerator_power_FR, I_theta_numerator_power_FR, numG_power_FR)

## read out the results ##
results <- c(pValue_R_power, pValue_R_power_FR)
save(results, file = paste0("simu_result_", seedIndex, ".RData"))

### This program will embed FLM-smoothed genotypes in Schaid’s kernel and burden tests. The B-Spline basis was used to smooth the genotypes.

#### B.3.2 Embedded approaches

The following program embedded FLM-smoothed genotypes in retrospective kernel-based approach and burden tests. The B-Spline basis was used to smooth the genotypes.
require(pedgene); require(fda)
order <- 4

# DIRmain <- "/Volumes/Macintosh HD 2/Dissertation/Programs/Simulation/otherStatistics/"
DIRmain <- "/Users/yjiang/Dissertation/Simulation/otherStatistics/"
#MODEL <- "m_40_m_20"
dataDIR <- paste0(DIRmain, "oriGenoPedtrait_", MODEL ,"/")
flmDIR <- paste0(DIRmain, "flmGenoSimuData_", MODEL , "/")
ORname <- paste0("logOR_", MODEL , ".RData")

# Dirектория, в которой находятся данные
DIRmain <- "/Users/yjiang/Dissertation/Simulation/otherStatistics/"
MODEL <- "m_40_m_20"

# Data directory
dataDIR <- paste0 ( DIRmain , " oriGenoPedtrait_ ", MODEL , " /")
flmDIR <- paste0 ( DIRmain , " flmGenoSimuData_ ", MODEL , " /")
ORname <- paste0 ( " logOR_ ", MODEL , " .RData " )

# Read in simulated data sets.
setwd(DIRmain)
seeds <- get ( load ( " seedFile . RData ") )
snp <- get ( load ( " SNPInfo_sub . RData ") )
logORrisk <- get ( load ( ORname ) )
riskVariantList <- snp [ which ( logORrisk > 0 ) , "rs"]
protectVariantList <- snp [ which ( logORrisk < 0 ) , "rs"]

NFILES <- 500
for ( f in 1 : NFILES ) {
  SEED <- seeds [ f ]
powerGeno <- get ( load ( paste0 ( dataDIR , " powerGeno_ ", f , " . RData " ) ) )
pedstr <- get ( load ( paste0 ( dataDIR , " pedstrTrait_ ", f , " . RData " ) ) )
NCpowerGeno <- ncol ( powerGeno )
N <- nrow ( pedstr )
row.names ( powerGeno ) <- 1 : N

  pos <- snp [ snp$rs %in% colnames ( powerGeno ) , "position"]
  if ( max ( pos ) > 1 ) {
    pos <- ( pos - min ( pos ) ) / ( max ( pos ) - min ( pos ) )
  }

  ## generate basis functions ##
  if ( NCpowerGeno > order ) { # number of non-polymorphic variants > 4 #
    betabasis <- create.bspline.basis ( norder = order , nbasis = NCpowerGeno )
  } else {
    betabasis <- create.bspline.basis ( norder = NCpowerGeno , nbasis = NCpowerGeno )
  }

  B <- eval.basis ( pos , betabasis )
powerGeno <- as.matrix ( powerGeno )
UJ <- powerGeno %*% B

  weight1 <- rep ( 1 , NCpowerGeno )
schaidPed <- pedstr
schaidPed$ID <- NULL
colnames ( schaidPed ) <- c ( "ped" , "person" , "father" , "mother" , "sex" , "trait" )
pedPer <- schaidPed [ , c ( "ped" , "person" ) ]
schaidPowerG <- cbind ( pedPer , UJ )

  schaid_power <- pedgene ( schaidPed , schaidPowerG , male.dose = 2 , checkpeds = FALSE , weights = weight1 , weights.mb = NULL , method = "kounen" , acc.davies = 1e-9 )
pKernelPower <- schaid_power$pgdf$pdval.kernel
pBurdenPower <- schaid_power$pgdf$pdval.burden

  result <- c ( pKernelPower , pBurdenPower , SEED )
otherStats <- as.data.frame ( t ( result ) )
outputFileName <- paste0 ( "powerSchaidFLM_num_ ", MODEL , ".txt" )
write.table ( otherStats , file = outputFileName , append = TRUE , quote = FALSE ,
  sep = "\t", row.names = FALSE , col.names = FALSE )
B.3.3 Other statistics for comparison

The following program computed the statistics derived from the retrospective kernel and burden tests, famSKAT, and F-SKAT, which were used to compare with the GFLMM and the embedded approach developed in Chapter 3.

```r
# This program will compute other statistics for a comparison purpose.
### Schaid's kernel and burden tests
### famSKAT treating binary 0/1 as quantitative trait
### Qi's method

rm(list=ls())

args = (commandArgs(TRUE))
if (length(args) == 0) {
  print("No arguments supplied.")
  MODEL = "2.40"
} else {
  for (i in 1:length(args)) {
    eval(parse(text=args[[i]]))
  }
}

DIRmain <- "~/Users/yjiang/Dissertation/Simulation/otherStatistics/"
MODEL <- "2.40"
dataDIR <- paste0(DIRmain, "oriGenoPedtrait_", MODEL,"/")
ORname <- paste0(\"logOR_\", MODEL, ".RData\")

# Read in simulated data sets.

setwd(DIRmain)
install.packages("kinship", lib= "./kinship", type= "source", repos= NULL)
#powerGeno <- get(load("powerGeno.RData"))
#alphaGeno <- get(load("alphaGeno.RData"))
#pedstr <- get(load("pedstrTrait.RData"))
seeds <- get(load("seedFile.RData"))
snp <- get(load("SNPInfo_sub.RData"))
logORrisk <- get(load(\"ORname\"))
riskVariantList <- snp[which(logORrisk > 0), "rs"]
protectVariantList <- snp[which(logORrisk < 0), "rs"]

NFILES <- 500
for (f in 1:NFILES) {
  SEED <- seeds[f]
powerGeno <- get(load(paste0(dataDIR,"powerGeno_",f,".RData")))
pedstr <- get(load(paste0(dataDIR,"pedstrTrait_",f,".RData")))
NCpowerGeno <- ncol(powerGeno)
N <- nrow(pedstr)
row.names(powerGeno) <- 1:N
riskGeno <- intersect(colnames(powerGeno),riskVariantList)
numRisk <- length(riskGeno)
```

105
protectGeno <- intersect(colnames(powerGeno), protectVariantList)
numProtect <- length(protectGeno)

# Compute Schaid's kernel and burden statistics.
# pedigree package in R
require("pedgene")
weight1 <- rep(1, NCpowerGeno)
schaidPed <- pedstr
schaidPed$ID <- NULL
colnames(schaidPed) <- c("ped","person","father","mother","sex","trait")
schaidPowerG <- cbind(schaidPed[, c("ped","person")], powerGeno)
schaid_powerBT <- pedgene(schaidPed, schaidPowerG, male.dose = 2, checkpeds= FALSE, weights = NULL, weights.mb= FALSE, method= "kounen", acc.davies= 1e-9)
pKernelPowerBT <- schaid_powerBT$pgdf$pval.kernel
pBurdenPowerBT <- schaid_powerBT$pgdf$pval.burden

schaid_powerMB <- pedgene(schaidPed, schaidPowerG, male.dose = 2, checkpeds= FALSE, weights = NULL, weights.mb= TRUE, method= "kounen", acc.davies= 1e-9)
pKernelPowerMB <- schaid_powerMB$pgdf$pval.kernel
pBurdenPowerMB <- schaid_powerMB$pgdf$pval.burden

schaid_power1 <- pedgene(schaidPed, schaidPowerG, male.dose = 2, checkpeds= FALSE, weights = weight1, weights.mb= TRUE, method= "kounen", acc.davies= 1e-9)
pKernelPower1 <- schaid_power1$pgdf$pval/kernel
pBurdenPower1 <- schaid_power1$pgdf$pval/burden

# Adjust ID coding for the following statistics computation.
pedIDstr <- pedstr
IDindex <- with(pedIDstr, pedigree(id= Per, dadid= Father, momid= Mother, sex= Sex, famid= Ped))
pedIDstr$fa <- IDindex$findex
pedIDstr$mo <- IDindex$mindex
pedIDstr$ID <- pedIDstr$trait

# Compute famSKAT statistics.
# http://www.hsph.harvard.edu/han-chens/FamSKAT/
detach("package:pedgene")
detach("package:kinship2")
require("kinship", lib.loc= ".:/kinship")
source("famSKAT_v1.8.04052013.R")
y <- pedIDstr$trait
fskPowerG <- as.matrix(powerGeno)
id <- pedIDstr$ID
kin <- makekinship(pedIDstr$Ped, pedIDstr$ID, pedIDstr$fa, pedIDstr$mo)

fSKATpowerW <- famSKAT(phenotype= y, genotypes= fskPowerG, id= id, fullkins= kin)
pFamSKATpowerW <- fSKATpowerW$pvalue # by default: Wu weights SKAT#

fSKATpower1 <- famSKAT(phenotype= y, genotypes= fskPowerG, id= id, fullkins= kin, sqrtweights= weight1)
pFamSKATpower1 <- fSKATpower1$pvalue

# Compute Qi's statistics.
# http://www.pitt.edu/~qiy17/Softwares.html
source("./Qi_FSKAT/glmPQL.s")
source("./Qi_FSKAT/FSKAT.R")
qiPedstr <- pedIDstr[ ,c("ID","fa","mo","trait")]
qiPedstr[qiPedstr$fa == 0, "fa"] <- NA
qiPedstr[qiPedstr$mo == 0, "mo"] <- NA
qiPowerG <- as.data.frame(t(powerGeno))
IDname <- colnames(qiPowerG)
qiPowerG$snp <- row.names(qiPowerG)
qiPowerG$gene <- "simuGene"
qiPowerG <- qiPowerG[ ,c("gene","snp", IDname)]
pQiPowerW <- FSKAT(phenotype = y, genotypes = qiPowerG, id= qiPedstr$ID, fa= qiPedstr$fa, mo= qiPedstr$mo, family = "binomial", covariates = NULL, weights = NULL, acc= 1e-9)
pQiPower1 <- FSKAT(phenotype = y, genotypes = qiPowerG, id= qiPedstr$ID, fa= qiPedstr$fa, mo= qiPedstr$mo, family = "binomial", covariates = NULL, weights = qiWeight1, acc= 1e-9)
otherStats <- as.data.frame(t(result))
outputFileName <- paste0(" powerOther_", MODEL, ".txt")
write.table(otherStats, file = outputFileName, append= TRUE, quote= FALSE, sep = "\t", row.names= FALSE, col.names= FALSE)

B.4 PLOT TYPE I ERROR RATES AND POWER LEVELS

We used the following codes to collect the simulation results and plot type I error rates and power levels.

# ##############################################################
## Draw barplots of power and type I error rates .
# ##############################################################
rm(list=ls())
require("binom")
require("ggplot2")
require("RColorBrewer")
require("grid")
```r
Scenario <- c("2.0/0.8\n30%/0%","2.0/0.8\n30%/20%","m10\n30%/0%","m10\n30%/20%","m4\n30%/0%","m4\n30%/20%","m3\n30%/0%","m3\n30%/20%")
Model <- c("2.30","2.30_08_20","m_30","m_30_m_20","m4_30","m4_30_42","m_30","m3_30_m_20")
fileName5 <- "power_30_5.pdf"; fileName1 <- "power_30_1.pdf"
fileName5_alpha <- "alpha_30_5.pdf"; fileName1_alpha <- "alpha_30_1.pdf"

# Scenario <- c("2.0/0.8\n40%/0%","2.0/0.8\n40%/20%","m10\n40%/0%","m10\n40%/20%","m4\n40%/0%","m4\n40%/20%","m3\n40%/0%","m3\n40%/20%")
# Model <- c("2_40","2_40_08_20","m_40","m_40_m_20","m4_40","m4_40_m4_20","m3_40","m3_40_m3_20")
# fileName5 <- "power_40_5.pdf"; fileName1 <- "power_40_1.pdf"
# fileName5_alpha <- "alpha_40_5.pdf"; fileName1_alpha <- "alpha_40_1.pdf"

LEVEL1 <- 0.05; LEVEL2 <- 0.01; ITER <- 500
mainDIR <- "/Volumes/Macintosh HD 2/Dissertation/Programs/Simulation/
otherDIR <- paste0(mainDIR,"otherStatistics/result/"
typeIDIR <- paste0(mainDIR,"type_I_error/result/"

# Analyze power simulation results from other methods.

setwd(otherDIR)
powerTable5 <- powerTable1 <- data.frame(stringsAsFactors=FALSE)
for (m in Model) {
  scenario <- Scenario[which(Model == m)]
  resultDIR <- paste0(mainDIR,"result_",m,"/"
  resultFlm <- data.frame(flmBS = numeric(0), flmFR = numeric(0), stringsAsFactors=FALSE)
  R <- c(1:ITER)
  for (j in R) {
    r <- get(load(paste0(resultDIR,"simu_result_",j,".RData")))
    resultFlm[j,] <- r
  }
  otherPowerName <- paste0("powerOther_",m,".txt")
schaidFlmName <- paste0("powerSchaidFLM_",m,".txt")
  other <- read.table(otherPowerName, header=FALSE, sep="\t", stringsAsFactors=FALSE)
  colnames(other) <- c("kernel_BT","burden_BT","kernel_MB","burden_MB","kernel_UW","burden_UW","famSKAT_W","famSKAT_UW","FSKAT_W","FSKAT_UW","numTotal","numRisk","numProtect","seed")
  flmSchaid <- read.table(schaidFlmName, header=FALSE, sep="\t", stringsAsFactors=FALSE)
  colnames(flmSchaid) <- c("flmBS_kernel","flmBS_burden","seed")
  otherPower <- merge(flmSchaid, other, by="seed", all=TRUE, sort=FALSE)
  otherPower$numTotal <- otherPower$numRisk <- otherPower$numProtect <- otherPower$seed <- NULL
  Power <- cbind(resultFlm, otherPower)
  Nrow <- nrow(Power)
  Ncol <- ncol(Power)
  methods <- colnames(Power)
  scenarioTable5 <- data.frame(stringsAsFactors=FALSE)
  scenarioTable1 <- data.frame(stringsAsFactors=FALSE)
  for (i in 1:Ncol) {
    p <- Power[,i]
    p5 <- length(p[p < LEVEL1])
  }
}
```
p1 <- length(p[p < LEVEL2])
power5 <- p5/Nrow
power1 <- p1/Nrow
ci5 <- binom.confint(p5, Nrow, conf.level = 1-LEVEL1, methods = "exact")
lower5 <- ci5$lower ; upper5 <- ci5$upper
lower1 <- ci1$lower ; upper1 <- ci1$upper
testResult5 <- c(power5, lower5, upper5)
testResult1 <- c(power1, lower1, upper1)
scenarioTable5 <- rbind(scenarioTable5, testResult5)
scenarioTable1 <- rbind(scenarioTable1, testResult1)
}
colnames(scenarioTable5) <- c("power","lower","upper")
colnames(scenarioTable1) <- c("power","lower","upper")
scenarioTable5$method <- scenarioTable1$method <- methods
scenarioTable5$scenario <- scenarioTable1$scenario <- scenario
powerTable5 <- rbind(powerTable5, scenarioTable5)
powerTable1 <- rbind(powerTable1, scenarioTable1)
}
powerTable5$flag <- powerTable1$flag <- "okay"
methodMatch <- unique(powerTable5$method)

# ##############################################################
## Indicate the inflation or deflation of Type I errors.
# ##############################################################
setwd(typeIDIR)
alphaTable5 <- alphaTable1 <- data.frame(stringsAsFactors= FALSE)
for (n in Model) {
  scenario <- Scenario[which(Model == n)]
  otherName <- paste0("typeIother_logOR_", n, ".txt")
  other <- read.table(otherName, header = FALSE, sep = "\t", stringsAsFactors= FALSE)
colnames(other) <- c("kernel_BT","burden_BT","kernel_MB","burden_MB","kernel_UW","burden_UW","famSKAT_W","famSKAT_UW","FSKAT_W","FSKAT_UW","seed")
flmName <- paste0("typeIflm_logOR_", n, ".txt")
flm <- read.table(flmName, header = FALSE, sep = "\t", stringsAsFactors= FALSE)
colnames(flm) <- c("flmBS","flmFR","flmBS7","flmFR7","flmBS9","flmFR9","numRisk","numProtect","seed")
flm$flmBS7 <- flm$flmFR7 <- flm$flmBS9 <- flm$flmFR9 <- NULL
# flm3 <- read.table("typeIflm_3.txt", header = FALSE, sep = "\t", stringsAsFactors= FALSE)
# colnames(flm3) <- c("BS3alpha","FR3alpha","causal","seed")
error <- flm
error <- merge(other, flm, by = "seed", all= TRUE, sort = FALSE)
#error <- merge(error, flm3, by = "seed", all= TRUE, sort = FALSE)
flmEmbedName <- paste0("typeIschaidFlm_logOR_", n, ".txt")
flmEmbed <- read.table(flmEmbedName, header = FALSE, sep = "\t", stringsAsFactors= FALSE)
colnames(flmEmbed) <- c("flmBS3_kernel","flmBS3_burden","flmBS_kernel","flmBS_burden","flmBS7_kernel","flmBS7_burden","flmBS9_kernel","flmBS9_burden","seed")
flmEmbed5 <- flmEmbed[,c("flmBS_kernel","flmBS_burden","seed")]
error <- merge(error, flmEmbed5, by = "seed", all= TRUE, sort = FALSE)
error$seed <- error$numRisk <- error$numProtect <- NULL
NError <- ncol(error)
NError <- nrow(error)
ci5data <- ci1data <- data.frame(stringsAsFactors= FALSE)
for (i in 1:NError) {
  nAlpha <- error[,i]
x5 <- length(nAlpha[nAlpha < LEVEL1])
x1 <- length(nAlpha[nAlpha < LEVEL2])
ci5 <- binom.confint(x5, NRerror, conf.level = 1-LEVEL1, methods = "exact")
ci1 <- binom.confint(x1, NRerror, conf.level = 1-LEVEL1, methods = "exact")
ci5data <- rbind(ci5data, ci5)
ci1data <- rbind(ci1data, ci1)

method <- colnames(error)[1:NCerror]
.ci5data$method <- ci1data$method <- method
.ci5data$flag <- ci1data$flag <- "okay"
.ci5data[ci5data$lower > LEVEL1, "flag"] <- "I"
.ci5data[ci5data$upper < LEVEL1, "flag"] <- "D"
.ci1data[ci1data$lower > LEVEL2, "flag"] <- "I"
.ci1data[ci1data$upper < LEVEL2, "flag"] <- "D"

stat <- Scenario[which(Model == n)]

inflate5 <- ci5data[ci5data$flag == "I","method"]
deflate5 <- ci5data[ci5data$flag == "D","method"]
powerTable5[(powerTable5$scenario == stat) & (powerTable5$method %in% inflate5), "flag"] <- "I"
powerTable5[(powerTable5$scenario == stat) & (powerTable5$method %in% deflate5), "flag"] <- "D"
inflate1 <- ci1data[ci1data$flag == "I","method"]
deflate1 <- ci1data[ci1data$flag == "D","method"]
powerTable1[(powerTable1$scenario == stat) & (powerTable1$method %in% inflate1), "flag"] <- "I"
powerTable1[(powerTable1$scenario == stat) & (powerTable1$method %in% deflate1), "flag"] <- "D"

ci5data <- ci5data[match(methodMatch, ci5data$method),]
ci1data <- ci1data[match(methodMatch, ci1data$method),]

upper5 <- directory; cldata$scenario <- scenario; cldata$scenario <- scenario
alphaTable5 <- rbind(alphaTable5, ci5data)
alphaTable1 <- rbind(alphaTable1, ci1data)

powerTable5[powerTable5$flag == "okay", "flag"] <- ""
powerTable1[powerTable1$flag == "okay", "flag"] <- ""
powerTable5$order <- factor(powerTable5$scenario, unique(powerTable5$scenario))
powerTable1$order <- factor(powerTable1$scenario, unique(powerTable1$scenario))

alphaTable5 <- alphaTable5[,c("mean", "method", "scenario")]
alphaTable1 <- alphaTable1[,c("mean", "method", "scenario")]

bound5 <- binom.confint(LEVEL1*NRerror, NRerror, conf.level = 1-LEVEL1, methods = "exact")
UP5 <- bound5$upper; LO5 <- bound5$lower
bound1 <- binom.confint(LEVEL2*NRerror, NRerror, conf.level = 1-LEVEL1, methods = "exact")
UP1 <- bound1$upper; LO1 <- bound1$lower

# 30% or 40% models with alpha = 0.05

setwd(mainDIR)
palette <- colorRampPalette(brewer.pal(6, "Paired"))
limits <- aes(ymin = powerTable5$lower, ymax = powerTable5$upper)
powerPlot <- ggplot(powerTable5, aes(x = factor(method, levels = methods), y = power))
powerPlot <- powerPlot + ggtitle("Power Estimation by Different Statistics")
powerPlot <- powerPlot + theme(plot.title = element_text(size = 18, face = "bold", vjust = 2))
powerPlot <- powerPlot + geom_bar(aes(fill = factor(method, levels = methods), order = factor(method, levels = methods)), position = "dodge", stat = "identity")
powerPlot <- powerPlot + facet_wrap(~ order, ncol = 2)
powerPlot <- powerPlot + xlab("Statistics with alpha = 0.05")
powerPlot <- powerPlot + theme(text = element_text(size = 16), axis.title.x = element_text(vjust = -2))
powerPlot <- powerPlot + geom_text(aes(x = method, y = 0.01, label = flag, position = "dodge"), size = 4)
powerPlot <- powerPlot + ylab("Power")
powerPlot <- powerPlot + ylim(0, 1.0)
powerPlot <- powerPlot + theme(text = element_text(size = 16), axis.title.y = element_text(vjust = 3))
powerPlot <- powerPlot + theme(axis.ticks = element_blank(), axis.text.x = element_blank(), strip.text.x = element_text(size = 16, colour = "darkblue"))
powerPlot <- powerPlot + geom_errorbar(limits, width = 0.25)
powerPlot <- powerPlot + guides(fill = guide_legend(title = "Methods", keywidth = 1.62, keyheight = 1))
powerPlot <- powerPlot + guides(colour = guide_legend(title.hjust = 0.5))
powerPlot <- powerPlot + ylim(0, 1.0)
powerPlot <- powerPlot + theme(plot.background = element_rect(fill = "lightgrey"), plot.margin = unit(c(2, 1.5, 2, 1.5), "cm"))
ggsave(filename = fileName5, plot = powerPlot, width = 16, height = 20)

alphaPlot <- ggplot(alphaTable5, aes(x = factor(method, levels = methods), y = mean))
a因子Plot <- alphaPlot + ggtitle("Type I Error Estimation by Different Statistics")
a因子Plot <- alphaPlot + theme(title = element_text(size = 18, face = "bold", vjust = 2))
a因子Plot <- alphaPlot + geom_bar(aes(fill = factor(method, levels = methods), order = factor(method, levels = methods)), position = "dodge", stat = "identity")
a因子Plot <- alphaPlot + facet_wrap(~ order, ncol = 2)
a因子Plot <- alphaPlot + xlab("Statistics with alpha = 0.05")
a因子Plot <- alphaPlot + theme(text = element_text(size = 16), axis.title.x = element_text(vjust = -2))
a因子Plot <- alphaPlot + ylab("Type I error rates")
#alphaPlot <- alphaPlot + ylim(0, max(alphaTable5$mean))
a因子Plot <- alphaPlot + geom_hline(yintercept = c(0.05, 0.05), linetype = "dashed", colour = "brown")
a因子Plot <- alphaPlot + geom_hline(yintercept = 0.05, linetype = "dashed", colour = "black")
a因子Plot <- alphaPlot + theme(title = element_text(size = 16), axis.title.y = element_text(vjust = 3))
a因子Plot <- alphaPlot + guides(fill = guide_legend(title = "Methods", keywidth = 1.62, keyheight = 1))
a因子Plot <- alphaPlot + guides(colour = guide_legend(title.hjust = 0.5))
a因子Plot <- alphaPlot + scale_fill_manual(values = palette(length(methods)))
a因子Plot <- alphaPlot + theme(plot.background = element_rect(fill = "lightgrey"), plot.margin = unit(c(2, 1.5, 2, 1.5), "cm"))
ggsave(filename = fileName5_alpha, plot = alphaPlot, width = 16, height = 20)

parpowerPlot <- powerPlot + theme(text = element_text(size = 16), axis.title.x = element_text(vjust = -2))
powerPlot <- powerPlot + geom_text(aes(x = method, y = 0.01, label = flag, position = "dodge"), size = 4)

# alphaPlot <- alphaPlot + ylim(0, max(alphaTable5$mean))

alphaPlot <- alphaPlot + geom_hline(yintercept = c(UP5, LO5), linetype = "dashed", colour = "brown")
alphaPlot <- alphaPlot + geom_hline(yintercept = 0.05, linetype = "dashed", colour = "black")
alphaPlot <- alphaPlot + theme(text = element_text(size = 16), axis.title.y = element_text(vjust = 3))
alphaPlot <- alphaPlot + theme(axis.ticks = element_blank(), axis.text.x = element_blank(), strip.text.x = element_text(size = 16, colour = "darkblue"))
alphaPlot <- alphaPlot + guides(fill = guide_legend(title = "Methods", keywidth = 1.62, keyheight = 1))
alphaPlot <- alphaPlot + guides(colour = guide_legend(title.hjust = 0.5))
alphaPlot <- alphaPlot + scale_fill_manual(values = palette(length(methods)))
alphaPlot <- alphaPlot + theme(plot.background = element_rect(fill = "lightgrey"), plot.margin = unit(c(2, 1.5, 2, 1.5), "cm"))
ggsave(filename = fileName5_alpha, plot = alphaPlot, width = 16, height = 20)

# setwd(mainDIR)

# palette <- colorRampPalette(brewer.pal(6, "Paired"))

# limits <- aes(ymin = powerTable1$lower, ymax = powerTable1$upper)

powerPlot <- ggplot(powerTable1, aes(x = factor(method, levels = methods), y = power))
powerPlot <- powerPlot + ggtitle("Power Estimation by Different Statistics")
powerPlot <- powerPlot + theme(plot.title = element_text(size = 18, face = "bold", vjust = 2))
powerPlot <- powerPlot + geom_bar(aes(fill = factor(method, levels = methods), order = factor(method, levels = methods)), position = "dodge", stat = "identity")
powerPlot <- powerPlot + facet_wrap(~ order, ncol = 2)
powerPlot <- powerPlot + xlab("Statistics with alpha = 0.01")
powerPlot <- powerPlot + theme(title = element_text(size = 16), axis.title.x = element_text(vjust = -2))
powerPlot <- powerPlot + geom_text(aes(x = method, y = 0.01, label = flag, position = "dodge"), size = 4)

powerPlot <- powerPlot + ylab("Power")
powerPlot <- powerPlot + ylim(0, 1.0)

powerPlot <- powerPlot + geom_text(aes(x = method, y = 0.01, label = flag, position = "dodge"), size = 4)
B.5 REAL DATA ANALYSIS

B.5.1 GFLMM approach

We ran the following codes on the Ubuntu PC operating system (GNU/Linux 3.2.0-51-generic x86_64) to conduct a GWAS by using the GFLMM developed in Section 3.1.1.

### This program would compute p-values for each gene in GARM data
```
rm(list=ls())
args = commandArgs(TRUE)
if (length(args) == 0) {
  print("No arguments supplied.")
}
geneNum = 1
```
for (i in 1:length(args)) {
    eval(parse(text=args[[i]]))
}

DIR <- "/mnt/clusterfs/users/yij5/Input"
Rpkg <- "/mnt/clusterfs/users/yij5/Rpkg/"
Uname <- "u_matrix.star.RData"
phenoFileName <- "pheno_data.txt"

setwd(DIR)
require(fda, lib.loc = Rpkg)
require(MASS, lib.loc = Rpkg)
require(Matrix, lib.loc = Rpkg)

## parameters ##
order <- 4; basis <- 5
order3 <- 3; basis3 <- 3 # if number of variants <= 5 #

NonmissingPheID <- get(load("NonmissingPheID.RData"))
genoData <- get(load("exomeAMD_cleaned_noMissing.RData"))
snpAnnot <- get(load("snpAnnotDataV5.RData"))

## generate genotypes for a certain gene ##
geno <- genoData[NonmissingPheID,]
geno$FID <- geno$MAT <- geno$SEX <- geno$PHENOTYPE <- NULL
colnames(geno) <- sub("_.*","",colnames(geno))

## extract rare variants for analysis ##
rareMarker <- snpAnnot[snpAnnot$MAF < 0.05, "marker"]
rareMarker <- intersect(rareMarker, colnames(geno))
geno <- geno[, rareMarker]

refGene$name <- refGene$cdsStart <- refGene$cdsEnd <- NULL
refGene <- refGene[ncol(refGene$chrom) <= 5,]
refGene <- refGene[refGene$chrom != "chrX" & refGene$chrom != "chrY",]
refGene <- refGene[!duplicated(refGene),]

## get positions of variants within a gene ##
posAnnot <- snpAnnot[, c("marker", "rsid_Hum", "pos_Hum")]
posAnnot <- posAnnot[posAnnot$pos_Hum >= testGene$txEnd & posAnnot$pos_Hum <= testGene$txEnd & posAnnot$chrom %in% testGene$chrom,]
posAnnot <- posAnnot[order(posAnnot$pos_Hum, decreasing = FALSE),]

results <- data.frame(chr = character(0), gene = character(0), nvariants = numeric(0),
                      start = numeric(0), end = numeric(0), pFourier = numeric(0),
                      pBspline = numeric(0), geneID = numeric(0), stringsAsFactors = FALSE)

if (length(selectMarker) > 1) {
    geno <- geno[, selectMarker]
snpNumber <- ncol(geno)
}
for (k in 1:snpNumber) {
    marker <- geno[, k]
genotype0 <- sum(marker == 0, na.rm = TRUE)
genotype2 <- sum(marker == 2, na.rm = TRUE)
if (genotype0 < genotype2) {
geno[,k] <- 2-geno[,k]

# IMPORTANT: rare allele coded as 2 #

## Apply Fan's beta-smooth only model ##
# geno[is.na(geno)] <- 0 # code missing genotype to 0 #
pos <- pos&&not(pos&&not$marker %in% selectMarker, "pos_Hum")

## QR decomposition. We do not conduct this. ##
# dqr <- qr(geno)
# index <- dqr$qr[1:dqr$rank]
# geno <- geno[,index]

## normalize position ##
if (max(pos) > 1) {
  pos <- (pos - min(pos)) / (max(pos) - min(pos))
}

## generate basis functions ##
if (nsnp > basis) { # number of non-polyomrophic variants > 5 #
betabasis <- create.bspline.basis(norder = order, nbasis = basis)
  fourierbasis <- create.fourier.basis(c(0,1), nbasis = basis)
} else {
  betabasis <- create.bspline.basis(norder = order3, nbasis= basis3)
  fourierbasis <- create.fourier.basis(c(0,1), nbasis = basis3)
}
B <- eval.basis(pos, betabasis)
FR <- eval.basis(pos, fourierbasis)

## make sure UJ has full rank of bbasis or fbasis ##
# UJdqr <- qr(UJ)
# UJdqr_FR <- qr(UJ_FR)
# UJindex <- UJdqr$rank
# UJindex_FR <- UJdqr_FR$rank
# UJ <- UJ[,UJindex]
# UJ_FR <- UJ_FR[,UJindex_FR]

estSigma <- 1.866244e-01; estBetaNull <- 1.449921e+00; lnL <- -6.224243e+02
Z <- 1 # covariate vector #
C <- 350000 # number of cubature sizes #
W <- 1/C # equal weights #

phenofile <- read.table(phenofileName, header = FALSE, sep = "\t", stringsAsFactors = FALSE)
colnames(phenofile) <- c("ID","trait")
NRpheno <- nrow(phenofile) # 1275 #
y <- pheno$trait
uMat <- get(load(Unnamed))
numG <- ncol(UJ)
numG_FR <- ncol(UJ_FR)
G <- as.data.frame(UJ)
G_FR <- as.data.frame(UJ_FR)
G$Z <- 1
G_FR$Z <- 1
S_theta_numerator <- S_theta_numerator_FR <- 0
I_theta_numerator <- I_theta_numerator_FR <- 0
S_theta_denominator <- exp(lnL - log(W))
I_theta_denominator <- S_theta_denominator

for (c in 1:C) {
  ## compute p ##
  u_vec <- uMat[((c-1)*NRpheno+1):(c*NRpheno)]
  B <- estBetaNull + estSigma*u_vec
  p <- exp(B)/(1+exp(B))

  ## compute likelihood given c: exp(1-a_c) ##
  l_c <- sum(y*log(p)+(1-y)*log(1-p))
  exp_l_c <- exp(l_c)

  G$u_vec <- u_vec
  G_FR$u_vec <- u_vec

  t_z_theta <- as.matrix(G)
  t_z_theta_FR <- as.matrix(G_FR)

  ## compute z= (G ,1 ,a(n))' and Dl_theta= sum((y-p)*z) ##
  y_p <- matrix((y-p), nrow = 1, ncol = NRpheno, byrow = TRUE)
  Dl_theta <- (y_p) %*% t_z_theta
  Dl_theta_FR <- (y_p) %*% t_z_theta_FR

  ## assume equal weights ##
  S_theta_numerator <- S_theta_numerator + Dl_theta*exp_l_c
  S_theta_numerator_FR <- S_theta_numerator_FR + Dl_theta_FR*exp_l_c

  ## D2l_theta= sum((p^2-p)*z*z') ##
  p2_p <- p^2-p
  p2_p <- matrix(rep(p2_p, row_z_theta), nrow = row_z_theta, ncol = NRpheno, byrow = TRUE)
  p2_p_FR <- matrix(rep(p2_p, row_z_theta_FR), nrow = row_z_theta_FR, ncol = NRpheno, byrow = TRUE)

  D2l_theta <- (p2_p*z_theta) %*% t_z_theta
  D2l_theta_FR <- (p2_p_FR*z_theta_FR) %*% t_z_theta_FR

  I_theta_numerator <- I_theta_numerator + (D2l_theta + t(Dl_theta) %*% Dl_theta)*exp_l_c
  I_theta_numerator_FR <- I_theta_numerator_FR + (D2l_theta_FR + t(Dl_theta_FR) %*% Dl_theta_FR)*exp_l_c
}

## write a function to compute p-value of the score test ##
scoreP <- function(S_theta_numerator, I_theta_numerator, numG) {
  S_theta <- S_theta_numerator / S_theta_denominator
  I_theta <- -(I_theta_numerator/I_theta_denominator - t(S_theta)) %*% S_theta
  S_gamma <- S_theta[1:numG, drop = FALSE]
  I_gamma_eta <- I_theta[1:numG,(numG+1):ncol(I_theta)]
  I_eta_gamma <- I_theta[(numG+1):ncol(I_theta),1:numG]
  I_eta_eta <- I_theta[(numG+1):ncol(I_theta),(numG+1):ncol(I_theta)]
  inv_I_eta_eta <- ginv(I_eta_eta)
  return(scoreP)
}
B.5.2 Embedded method

We ran the following codes on the Ubuntu PC operating system (GNU/Linux 3.2.0-51-generic x86_64) to conduct a GWAS by using the embedded method developed in Section 3.1.5.

```r
### This program would compute p-values for each gene in GARM data
### Embed FLM in the retrospective regression, variant-dependent basis
rm(list=ls())
args = (commandArgs(TRUE))
if (length(args) == 0) {
  print("No arguments supplied.")
  geneNum = 2
} else {
  for (i in 1:length(args)) {
    eval(parse(text=arg[[i]]))
  }
}

DIR <- "/mnt/clusterfs/users/yij5/_input"
Rpkg <- "/mnt/clusterfs/users/yij5/Rpkg/"
phenoFileName <- "pheno_data.txt"
setwd(DIR)
require("fda", lib.loc= Rpkg)
require("survey",lib.loc= Rpkg)
require("quadprog", lib.loc= Rpkg)
require("CompQuadForm", lib.loc= Rpkg)
require("kinship2", lib.loc= Rpkg)
```
require("pedgene", lib.loc= Rpkg)
#require(fda);require(pedgene)

## parameters ##
order <- 4; Nbasis5 <- 5
NonmissingPheID <- get(load("NonmissingPheID.RData"))
genoData <- get(load("exomeAMD_cleaned_noMissing.RData"))
refGene <- read.table("refseq_genes.txt", header= TRUE, stringsAsFactors= FALSE)
snpAnnot <- get(load("snpAnnotDataV5.RData"))
snpAnnot <- snpAnnot[snpAnnot$filterReason == "okay",]
#snpAnnot <- snpAnnot[snpAnnot$FILTER == "PASS" & snpAnnot$filterReason == "okay",]

## generate genotypes for a certain gene ##
geno <- genoData[NonmissingPheID,]

## generate pedigree structure ##
schaidPed <- geno[,c("FID","IID","PAT","MAT","SEX","PHENOTYPE")]
colnames(schaidPed) <- c("ped","person","father","mother","sex","trait")
pedPer <- schaidPed[,c("ped","person")]
geno$PAT <- geno$MAT <- geno$SEX <- geno$PHENOTYPE <- NULL
geno$FID <- geno$IID <- NULL
colnames(geno) <- sub("_.*","",colnames(geno))

## extract rare variants for analysis ##
rareMarker <- snpAnnot[snpAnnot$MAF < 0.05, "marker"]
rareMarker <- intersect(rareMarker,colnames(geno))
geno <- geno[,rareMarker]
refGene$name <- refGene$cdsStart <- refGene$cdsEnd <- NULL
refGene <- refGene[ncol(refGene$chrom) <= 5,]
refGene <- refGene[refGene$chrom != "chrX" & refGene$chrom != "chrY",]
refGene <- refGene[! duplicated(refGene),]

testGene <- refGene[geneNum,]

## get positions of variants within a gene ##
posAnnot <- snpAnnot[,c("marker","rsid_Hum","pos_Hum","chrom")]
posAnnot <- posAnnot[posAnnot$pos_Hum >= testGene$txStart & posAnnot$pos_Hum <= testGene$txEnd & posAnnot$chrom %in% testGene$chrom,]
posAnnot <- posAnnot[order(posAnnot$pos_Hum, decreasing= FALSE,],]
selectMarker <- intersect(colnames(geno),posAnnot$marker)
results <- data.frame(chr= character(0), gene= character(0), nvariants= numeric(0), start= numeric(0), end= numeric(0), pBspline= numeric(0), pFourier= numeric(0), geneID= numeric(0), stringsAsFactors= FALSE)
if (length(selectMarker) > 1) {
geno <- geno[,selectMarker]
}
snpNumber <- ncol(geno)
for (x in 1:snpNumber) {
  marker <- geno[,x]
genotype0 <- sum(marker == 0, na.rm= TRUE)
genotype2 <- sum(marker == 2, na.rm= TRUE)
if (genotype0 < genotype2) {
geno[,x] <- 2-geno[,x]
  } # IMPORTANT: rare allele coded as 2 #
}

## Apply Fan's beta-smooth only model ##
maf <- colMeans(geno, na.rm= TRUE)
pos <- posAnnot[posAnnot$marker %in% selectMarker, "pos_Hum"]
if (length(pos) > 2) { # at least 2 non-polymorphic variants #
95 geno <- geno[, maf > 0]  # remove nonpolymorphic variants #
96 nsample <- nrow(geno)
97 nsnps <- ncol(geno)
98
99 ## normalize position ##
100 if (max(pos) > 1) {
101   pos <- (pos - min(pos)) / (max(pos) - min(pos))
102 }
103
104 geno <- as.matrix(geno)
105 ## generate basis functions ##
106 if (nsnps > order) {  # >= 5 polymorphic variants #
107   betabasis <- create.bspline.basis(norder= order, nbasis= Nbasis5)
108   B <- eval.basis(pos, betabasis)
109   meanUJ <- colMeans(UJ/2, na.rm= TRUE)
110   meanTRUE <- all(meanUJ < 1)
111   while (!meanTRUE) {
112     Nbasis5 <- Nbasis5 + 1
113     betabasis <- create.bspline.basis(norder= order, nbasis= Nbasis5)
114     B <- eval.basis(pos, betabasis)
115     meanUJ <- colMeans(UJ/2, na.rm= TRUE)
116     meanTRUE <- all(meanUJ < 1)
117   }
118 } else {
119   Nbasis5 <- nsnps
120   betabasis <- create.bspline.basis(norder= nsnps, nbasis= Nbasis5)
121   B <- eval.basis(pos, betabasis)
122   meanUJ <- colMeans(UJ/2, na.rm= TRUE)
123   meanTRUE <- all(meanUJ < 1)
124   while (!meanTRUE) {
125     Nbasis5 <- Nbasis5 + 1
126     betabasis <- create.bspline.basis(norder= nsnps, nbasis= Nbasis5)
127     B <- eval.basis(pos, betabasis)
128     meanUJ <- colMeans(UJ/2, na.rm= TRUE)
129   }
130 }
131
132 weightBS <- rep(1, ncol(UJ))
133
134 schaidG_BS <- cbind(pedPer, UJ)
135 schaid_BS_result <- pedgene(schaidPed, schaidG_BS, male.dose= 2, checkpeds= FALSE,
136                              weights= weightBS, weights.nb= TRUE, method= "kounen", acc.davies=1e-9)
137 pKernel_BS <- schaid_BS_result$pdf$pval.kernel
138 pBurden_BS <- schaid_BS_result$pdf$pval.burden
139
140 ## read out the results ##
141 chr <- testGene$chrom
142 gene <- testGene$name2
143 start <- testGene$txStart
144 end <- testGene$txEnd
145 nvariants <- nsnps
146 results[1, ] <- c(chr, gene, nvariants, start, end, pKernel_BS, pBurden_BS, geneNum)
147 write.table(results, file= "/mnt/clusterfs/users/yij6/Output/okay_results_rare_flmSchaid_5.txt",
148            append= TRUE, row.names= FALSE, col.names= FALSE, quote= FALSE)
APPENDIX C

C CODES: A MODIFIED FUNCTION FOR THE GLOGS PROGRAM

We modified the “ml_gaussnewton_zero.c” function in the original GLOGS program in terms of adding the step-halving strategy to the Gauss-Newton algorithm, applying a new stopping criterion, and using equal cubature weights to approximate the integral (see details in 3.1.4).

```c
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <mpi.h>
#include <zlib.h>
#include "external_functions.h"

typedef struct {
  double L;
  double DL_theta, DL_s;
  double D2L_theta, D2L_s;
  double D2L_theta_s;
} derivative_return;

derivative_return
derivative_binomial_glmm(double theta, double s, int nobs, double *Y, int ncube, double *u_vec, double *pu_vec, int *u_index_vec)
{
  double u, pu;
  int u_index;
  double z, p;
  double L;
  double Dlz_theta, Dlz_s;
  double D2lz_theta, D2lz_s;
  double D2lz_theta_s;
  double prod_L;
  double sum_Dlz_theta, sum_Dlz_s;
  double sum_D2lz_theta, sum_D2lz_s;
  double sum_D2lz_theta_s;
  double L_acc;
  double DL_theta_acc, DL_s_acc;
  double D2L_theta_acc, D2L_s_acc;
  double D2L_theta_s_acc;
```
double pl_pu;
derivative_return ret_struct;
int i, c;
L_acc = 0;
DL_theta_acc = 0;
DL_s_acc = 0;
D2L_theta_acc = 0;
D2L_s_acc = 0;

for (c = 0; c < ncube; c++) {
    pu = pu_vec[c];
    prod_L = 1;
    sum_Dlz_theta = 0;
    sum_Dlz_s = 0;
    sum_D2lz_theta = 0;
    sum_D2lz_s = 0;
    sum_D2lz_theta_s = 0;
    u_index = u_index_vec[c];

    for (i = 0; i < nobs; i++) {
        u = u_vec[(u_index * nobs) + i];
        if (exp(z) == INFINITY) {
            p = 1;
        } else {
            p = exp(z) / (1 + exp(z));
        }

        L = pow(p, Y[i]) * pow(1-p, 1-Y[i]);
        prod_L = prod_L * L;
        Dlz_theta = Y[i] - p;
        Dlz_s = u * Dlz_theta;
        D2lz_theta = p * p - p;
        D2lz_theta_s = u * D2lz_theta;
        D2lz_s = u * D2lz_theta_s;
        sum_Dlz_theta = sum_Dlz_theta + Dlz_theta;
        sum_Dlz_s = sum_Dlz_s + Dlz_s;
        sum_D2lz_theta = sum_D2lz_theta + D2lz_theta;
        sum_D2lz_s = sum_D2lz_s + D2lz_s;
        sum_D2lz_theta_s = sum_D2lz_theta_s + D2lz_theta_s;
    }

    pL_pu = prod_L * pu;
    L_acc = L_acc + pL_pu;
    DL_theta_acc = DL_theta_acc + sum_Dlz_theta * pL_pu;
    DL_s_acc = DL_s_acc + sum_Dlz_s * pL_pu;
    D2L_theta_acc = D2L_theta_acc + (sum_D2lz_theta + sum_Dlz_theta * sum_Dlz_theta) * pL_pu;
    D2L_s_acc = D2L_s_acc + (sum_D2lz_s + sum_Dlz_s * sum_Dlz_s) * pL_pu;
    D2L_theta_s_acc = D2L_theta_s_acc + (sum_D2lz_theta_s + sum_Dlz_theta * sum_Dlz_s) * pL_pu;
}

ret_struct.L = L_acc;
ret_struct_DL_theta = DL_theta_acc;
ret_struct_DL_s = DL_s_acc;
ret_struct_D2L_theta = D2L_theta_acc;
ret_struct_D2L_s = D2L_s_acc;
ret_struct_D2L_theta_s = D2L_theta_s_acc;

return (ret_struct);
}

derivative_return
gather_nodes(derivative_return d_ret, int totnodes)
{ MPI_Status stat;
 double dtool;
 int r;

 for (r = 1; r < totnodes; r++) {
   MPI_Recv(&dtool, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
   d_ret.L = d_ret.L + dtool;
   MPI_Recv(&dtool, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
   d_ret.DL_theta = d_ret.DL_theta + dtool;
   MPI_Recv(&dtool, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
   d_ret.DL_s = d_ret.DL_s + dtool;
   MPI_Recv(&dtool, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
   d_ret.D2L_theta = d_ret.D2L_theta + dtool;
   MPI_Recv(&dtool, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
   d_ret.D2L_s = d_ret.D2L_s + dtool;
   d_ret.D2L_theta_s = d_ret.D2L_theta_s + dtool;
 }
 return (d_ret);
}

void calculate_derivatives(derivative_return d_ret, double *L, double *DlnL, double *D2lnL)
{
 double DL_theta, DL_s;
 double D2L_theta, D2L_s;
 double D2L_theta_s;
 double DlnL_theta, DlnL_s;
 double D2lnL_theta, D2lnL_s;
 double D2lnL_theta_s;

 *L = d_ret.L;
 DL_theta = d_ret.DL_theta;
 DL_s = d_ret.DL_s;
 D2L_theta = d_ret.D2L_theta;
 D2L_s = d_ret.D2L_s;
 D2L_theta_s = d_ret.D2L_theta_s;
 DlnL_theta = DL_theta / *L;
 DlnL_s = DL_s / *L;
 DlnL[0] = DlnL_theta;
 DlnL[1] = DlnL_s;
 D2lnL_theta = (D2L_theta / *L) - DlnL_theta * DlnL_theta;
 D2lnL_s = (D2L_s / *L) - DlnL_s * DlnL_s;
 D2lnL_theta_s = (D2L_theta_s / *L) - DlnL_theta * DlnL_s;
 D2lnL[0] = -D2lnL_theta;
 D2lnL[3] = -D2lnL_s;
}

void calculate_parameters(double theta, double s, double *DlnL, double *D2lnL,
 double *theta_star, double *s_star, double alpha)
{
 double x11, x12, x22;
 double z1, z2;
 double y1, y2;
 double denom;
x11 = D2lnL[0];
x12 = D2lnL[1];
x22 = D2lnL[3];
z1 = DlnL[0];
z2 = DlnL[1];
denom = x11 * x22 - x12 * x12;
if (denom != 0) {
    y1 = (x22 * z1 - z2 * x12) / denom;
    y2 = (-x12 * z1 + x11 * z2) / denom;
    *theta_star = theta + alpha * y1;
    *s_star = s + alpha * y2;
} else {
    *theta_star = theta;
    *s_star = s;
}

/* Equal weights, no updates */

int update_pu(double theta, double s, int nobs, double *Y, int ncube,
               double *u_vec, double *pu_vec, int *u_index_vec, double L,
               int node, int totnodes, double thresh) {
    double u, pu;
    int u_index;
    double z, p;
    double L_u, L_prod, pu_0;
    int nonzero_cube;
    double pu_acc;
    int i, c, r, inttool;
    double dttool;
    int zero_index, nonzero_index;
    MPI_Status stat;
    nonzero_cube = 0;
    for (c = 0; c < ncube; c++) {
        pu = pu_vec[c];
        u_index = u_index_vec[c];
        L_prod = 1;
        for (i = 0; i < nobs; i++) {
            u = u_vec[(u_index * nobs) + i];
            z = theta + s * u;
            if (exp(z) == INFINITY) {
                p = 1;
            } else {
                p = exp(z) / (1 + exp(z));
            }
            L_u = pow(p, Y[i]) * pow(1 - p, 1 - Y[i]);
            L_prod = L_prod * L_u;
        }
        pu_0 = (L_prod * pu) / L;
        if ((pu_0 < 0) || (pu_0 > 1)) {
            printf("Warning: pu_0= %le\n", pu_0);
            printf("Diagnostic: L_prod= %le pu= %le L= %le\n", L_prod, pu, L_prod * pu, L);
        }
        if (pu_0 > 0) {
            nonzero_cube++;
        }
    }
    if (node == 0) {
        ...
for (r = 1; r < totnodes; r++) {
    MPI_Recv(&inttool, 1, MPI_INT, r, r, MPI_COMM_WORLD, &stat);
    nonzero_cube = nonzero_cube + inttool;
}
thresh = thresh / (double)(nonzero_cube);
for (r = 1; r < totnodes; r++) {
    MPI_Send(&thresh, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
} else {
    MPI_Send(&nonzero_cube, 1, MPI_INT, 0, node, MPI_COMM_WORLD);
    MPI_Recv(&thresh, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
}
pu_acc = 0;
nonzero_cube = 0;
for (c = 0; c < ncube; c++) {
    if (pu_vec[c] < thresh) {
    } else {
        pu_acc = pu_acc + pu_vec[c];
        nonzero_cube++;
    }
}
zero_index = 0;
nonzero_index = 0;
do {
    while ((zero_index < ncube) && (pu_vec[zero_index] != 0)) {
        zero_index++;
    }
    if (zero_index < ncube) {
        if (nonzero_index < zero_index)
            nonzero_index = zero_index;
        while ((nonzero_index < ncube) && (pu_vec[nonzero_index] == 0)) {
            nonzero_index++;
        }
        if (nonzero_index < ncube) {
            inttool = u_index_vec[zero_index];
            u_index_vec[zero_index] = u_index_vec[nonzero_index];
            u_index_vec[nonzero_index] = inttool;
        } else {
            zero_index = nonzero_index;
        }
    }
    while (zero_index < ncube);
    if (node == 0) {
        for (r = 1; r < totnodes; r++) {
            MPI_Recv(&dtool, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
            pu_acc = pu_acc + dtool;
        }
        for (r = 1; r < totnodes; r++) {
            MPI_Send(&pu_acc, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
        }
    } else {
        MPI_Send(&pu_acc, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
        MPI_Recv(&pu_acc, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    }
    return (nonzero_cube);
}
void
write_pu(int ncube, int nobs, double *pu_vec, double *u_vec, int *u_index_vec,
    char *u_outputfile, char *pu_outputfile, char *write_append)
{
    int c, r, u_index;
    FILE *fptr_pu;
gzFile fptr_u;

int main(int argc, char *argv[])
{
    int node, totnodes;
    int nobs, ngene, ncube_node_base;
    double *phenotypes;
    double *u, pu_base, *pu;
    int *u_index_vec;
    double theta_base, s_lo, s_hi, s_step, s_curr;
    double cull_thresh;
    int *id, *u_id, haltcrit;
    int ncube_node;
    double theta, s;
    derivative_return d_ret;
    double L, DlnL_distance, DlnL[2], D2lnL[4];
    double theta_0, s_0;
    double *pu_store;
    int ncube_node_store;
    int * u_index_vec_store;
    double s_eval_code, stop_code, update_pu_code, pu_store_code, pu_write_code;
    int retcode;
    double D2lnLinv[4];
    int i, j, r;
    char c, fname[256], fname2[256];
    FILE *fptr;
    double dtool;
    MPI_Status stat;
    double lnL_old;
    double lnL_new;
    double L_distance2;
    double L_compare;
    double FTOL;
    double EPS;
    double alpha;
    double stop_step;
    EPS = pow(10, -10);

    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &totnodes);
    MPI_Comm_rank(MPI_COMM_WORLD, &node);

    if (argc != 10) {
        if (node == 0) {
            printf("Required arguments: <pheno/cov file> <nobs> <ncube_node> <theta effect>
<s effect lo> <s effect hi> <s effect step> <stopping crit> <point culling threshold >\n");
        }
        MPI_Finalize();
        exit(0);
    }
    sscanf(argv[2], "%i", &nobs);
    id = (int *)malloc(nobs * sizeof(int));
    phenotypes = (double *)malloc(nobs * sizeof(double));
    fptr = fopen(argv[1], "r");
    for (i = 0; i < nobs; i++) {
        fscanf(fptr, "%i %lf", &(id[i]), &(phenotypes[i]));
    }
    fclose(fptr);
sscanf(argv[3], "%i", &ncube_node_base);
    u_id = (int *)malloc(nobs * sizeof(int));
    u = (double *)malloc(ncube_node_base * nobs * sizeof(double));
    sprintf(fname, "u_matrix_%i.dat", node);
    fptr = fopen(fname, "r");
    for (i = 0; i < nobs; i++) {
        if (fscanf(fptr, "%i", &(u_id[i])) == EOF) {
            if (node == 0) {
                printf("Too small cubature file. Quitting.
";
            }
            MPI_Finalize();
            exit(0);
        }
    }
    for (i = 0; i < ncube_node_base * nobs; i++) {
        if (fscanf(fptr, "%lf", &(u[i])) == EOF) {
            if (node == 0) {
                printf("Too small cubature file. Quitting.
";
            }
            MPI_Finalize();
            exit(0);
        }
    }
    fclose(fptr);
    u_index_vec = (int *)malloc(ncube_node_base * sizeof(int));
    for (i = 0; i < ncube_node_base; i++) {
        u_index_vec[i] = i;
    }
    u_index_vec_store = (int *)malloc(ncube_node_base * sizeof(int));
    pu = (double *)malloc(ncube_node_base * sizeof(double));
    pu_store = (double *)malloc(ncube_node_base * sizeof(double));
    pu_base = 1.0 / (double)(ncube_node_base * totnodes);
    haltcrit = 0;
    for (i = 0; i < nobs; i++) {
        if (id[i] != u_id[i])
            haltcrit = 1;
    }
    if (haltcrit == 1) {
        if (node == 0) {
            printf("Nonmatching phenotype/covariate and cubature person IDs. Quitting.
";
        }
        MPI_Finalize();
        exit(0);
    }
    sscanf(argv[4], "%le", &theta_base);
    sscanf(argv[5], "%le", &s_lo);
    sscanf(argv[6], "%le", &s_hi);
    sscanf(argv[7], "%le", &s_step);
    sscanf(argv[8], "%le", &FTOL);
    sscanf(argv[9], "%le", &cull_thresh);
    if (node == 0) {
        s_eval_code = 1;
        s_curr = s_lo;
        while (s_eval_code == 1) {
            for (r = 1; r < totnodes; r++) {
                MPI_Send(&s_eval_code, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
            }
            printf("s = %le evaluation.\n", s_curr);
            theta = theta_base;
            s = s_curr;
            for (i = 0; i < ncube_node_base; i++) {
                pu[i] = pu_base;
            }
        }
    }
stop_code = 0;

do {
    for (r = 1; r < totnodes; r++) {
        MPI_Send(&stop_code, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
    }
    for (r = 1; r < totnodes; r++) {
        MPI_Send(&theta_0, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
        MPI_Send(&s_0, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
    }
    d_ret_0 = derivative_binomial_gllmm(theta_0, s_0, nobs, phenotypes, ncube_node, u, pu, u_index_vec);
    d_ret_0 = gather_nodes(d_ret_0, totnodes);
    calculate_derivatives(d_ret_0, &L_0, DlnL_0, D2lnL_0);
}

/* Gauss-Newton with step-halving */
if (log(L_0) - log(L) < -1e-8) {
    stop_step = 0;
    alpha = alpha / 2;
} else {
    stop_step = 1;
}

while (stop_step == 0);

stop_step = 1;
for (r = 1; r < totnodes; r++) {
    MPI_Send(&stop_step, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
}

/* New stopping criteria: Likelihood comparison */
DlnL_distance_0 = sqrt(pow(DlnL_0[0], 2) + pow(DlnL_0[1], 2));
difference = DlnL_distance - DlnL_distance_0;
lnL_new = log(L_0);
lnL_old = log(L);
L_distance2 = 2 * fabs(lnL_new - lnL_old);
L_compare = FTOL * (fabs(lnL_new) + fabs(lnL_old) + EPS);

if (L_distance2 > L_compare) {
    theta = theta_0;
    s = s_0;
    L = L_0;
    DlnL_distance = DlnL_distance_0;
    DlnL[0] = DlnL_0[0];
    DlnL[1] = DlnL_0[1];
D2lnL[0] = D2lnL_0[0];
D2lnL[1] = D2lnL_0[1];
D2lnL[2] = D2lnL_0[2];
D2lnL[3] = D2lnL_0[3];

update_pu_code = 1;
for (r = 1; r < totnodes; r++) {
    MPI_Send(& update_pu_code , 1, MPI_DOUBLE , r, 0, MPI_COMM_WORLD);
    MPI_Send(&theta_0 , 1, MPI_DOUBLE , r, 0, MPI_COMM_WORLD);
    MPI_Send(&s_0 , 1, MPI_DOUBLE , r, 0, MPI_COMM_WORLD);
    MPI_Send(&L_0 , 1, MPI_DOUBLE , r, 0, MPI_COMM_WORLD);
}

ncube_node = update_pu(theta_0 , s_0 , nobs , phenotypes ,
                        ncube_node , u , pu , u_index_vec , L_0 ,
                        node , totnodes , cull_thresh);

stop_code = 0;
}

if ((L_0 > L) { 
    theta = theta_0 ;
    s = s_0 ;
    L = L_0 ;
    DlnL_distance = DlnL_distance_0;
    DlnL[0] = DlnL_0[0];
    DlnL[1] = DlnL_0[1];
    D2lnL[0] = D2lnL_0[0];
    D2lnL[1] = D2lnL_0[1];
    D2lnL[2] = D2lnL_0[2];
    D2lnL[3] = D2lnL_0[3];
}

stop_code = 1;
}

while (stop_code == 0);

for (r = 1; r < totnodes; r++) {
    MPI_Send(& stop_code , 1, MPI_DOUBLE , r, 0, MPI_COMM_WORLD);
}

if ((s_curr == s_lo) || (L > L_store)) { 
    pu_store_code = 1;
    for (r = 1; r < totnodes; r++) {
        MPI_Send(&pu_store_code , 1, MPI_DOUBLE , r, 0, MPI_COMM_WORLD);
    }
    theta_store = theta;
    s_store = s;
    L_store = L;
    DlnL_store[0] = DlnL[0];
    DlnL_store[1] = DlnL[1];
    D2lnL_store[0] = D2lnL[0];
    D2lnL_store[1] = D2lnL[1];
    D2lnL_store[2] = D2lnL[2];
    D2lnL_store[3] = D2lnL[3];
ncube_node_store = ncube_node;
for (i = 0; i < ncube_node; i++) {
    pu_store[i] = pu[i];
    u_index_vec_store[i] = u_index_vec[i];
}
else {
    pu_store_code = 0;
    for (r = 1; r < totnodes; r++) {
        MPI_Send(&pu_store_code, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
    }
}

printf("Curr Parameters :");
printf(" theta = %le s= %le ", theta, s);
printf(" lnL = %le\n", log(L));
printf("Best Parameters :");
printf(" theta = %le s = %le ", theta_store, s_store);
printf(" lnL= %le\n", log(L_store));

s_curr = s_curr + s_step;
if (s_curr > s_hi)
    s_eval_code = 0;
for (r = 1; r < totnodes; r++) {
    MPI_Send(&s_eval_code, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
}

printf("Final Parameters :");
printf(" theta = %le s= %le\n", theta_store, s_store);
printf(" lnL= %le\n", log(L_store));
printf("DlnL= %le\n", DlnL);
printf("D2lnL= "
);
printf("%le %le\n", D2lnL[0], D2lnL[1]);
printf("%le %le\n", D2lnL[2], D2lnL[3]);
recode = invert_matrix_2x2(D2lnL_store, D2lnLinv);
printf("%le %le\n", D2lnLinv[0], D2lnLinv[1]);
printf("%le %le\n", D2lnLinv[2], D2lnLinv[3]);
write_pu(ncube_node_store, nobs, pu_store, u, u_index_vec_store,
    " u_matrix_star.dat", " pu_matrix_star.dat", "w");
for (r = 1; r < totnodes; r++) {
    MPI_Send(&pu_write_code, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
    MPI_Recv(&pu_write_code, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
}
}

while (s_eval_code == 1) {
    for (i = 0; i < ncube_node_base; i++) {
        pu[i] = pu_base;
    }
    ncube_node = ncube_node_base;
    MPI_Recv(&stop_code, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    while (stop_code == 0) {
        MPI_Recv(&theta, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
        d_ret = derivative_binomial_glm(theta, s, nobs, phenotypes, ncube_node, u, pu,
            u_index_vec);
        MPI_Send(&(d_ret.L), 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
        MPI_Send(&(d_ret.D_theta), 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
        MPI_Send(&(d_ret.D_s), 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
        MPI_Send(&(d_ret.D2L_theta), 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
        MPI_Send(&(d_ret.D2L_s), 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
    }
}

write_pu(ncube_node_store, nobs, pu_store, u, u_index_vec_store,
    " u_matrix_star.dat", " pu_matrix_star.dat", "w");
for (r = 1; r < totnodes; r++) {
    MPI_Send(&pu_write_code, 1, MPI_DOUBLE, r, 0, MPI_COMM_WORLD);
    MPI_Recv(&pu_write_code, 1, MPI_DOUBLE, r, r, MPI_COMM_WORLD, &stat);
}
}
MPI_Recv(&stop_step, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
while (stop_step == 0) {
    MPI_Recv(&theta_0, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    MPI_Recv(&s_0, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    d_ret_0 = derivative_binomial_glmm(theta_0, s_0, nobs, phenotypes, ncube_node,
                                       u, pu, u_index_vec);
    MPI_Send(&d_ret_0.L, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
    MPI_Send(&d_ret_0.DL_theta, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
    MPI_Send(&d_ret_0.DL_s, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
    MPI_Send(&d_ret_0.D2L_theta, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
    MPI_Send(&d_ret_0.D2L_s, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
    MPI_Send(&d_ret_0.D2L_theta_s, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);

    MPI_Recv(&stop_step, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
}

MPI_Recv(&update_pu_code, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
if (update_pu_code == 1) {
    MPI_Recv(&theta, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    MPI_Recv(&s, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    MPI_Recv(&L, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
    ncube_node = update_pu(theta, s, nobs, phenotypes, ncube_node,
                            u, pu, u_index_vec, L, node, totnodes, cull_thresh);
}

MPI_Recv(&stop_code, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);

MPI_Recv(&pu_store_code, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);
if (pu_store_code == 1) {
    ncube_node_store = ncube_node;
    for (i = 0; i < ncube_node; i++) {
        pu_store[i] = pu[i];
        u_index_vec_store[i] = u_index_vec[i];
    }
}

MPI_Recv(&s_eval_code, 1, MPI_DOUBLE, 0, 0, MPI_COMM_WORLD, &stat);

MPI_Send(&pu_code, 1, MPI_DOUBLE, 0, node, MPI_COMM_WORLD);
MPI_Finalize();
BIBLIOGRAPHY


