A STATISTICAL METHOD FOR GENOTYPIC ASSOCIATION THAT IS ROBUST TO SEQUENCING MISCLASSIFICATION

By

LISHENG ZHOU

A dissertation submitted to the

Graduate School - New Brunswick

And

The Graduate School of Biomedical Sciences

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Microbiology and Molecular Genetics

Written under the direction of

Tara Matise, Ph.D., Derek Gordon, Ph.D.

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

May 2017

ABSTRACT OF THE DISSERTATION

A Statistical Method for Genotypic Association that Is Robust to Sequencing Misclassification

By LISHENG ZHOU

Dissertation Directors:

Tara Matise, Ph.D., Derek Gordon, Ph.D.

In analyzing human genetic disorders, association analysis is one of the most commonly used approaches. However, there are challenges with association analysis, including differential misclassification in data that inflates the false-positive rate. In this thesis, I present a new statistical method for testing the association between disease phenotypes and multiple single nucleotide polymorphisms (SNPs). This method uses next-generation sequencing (NGS) raw data and is robust to sequencing differential misclassification. By incorporating expectation-maximization (EM) algorithm, this method computes the test statistic and estimates important parameters of the model, including misclassification. By performing simulation studies, I report that this method maintains correct type I error rates and may obtain high statistical power.

Acknowledgement

First and foremost, I wish to express my sincere gratefulness to my advisors, Dr. Tara Matise and Dr. Derek Gordon. Their excellent guidance, continual support, and encouragement enormously helped me through this remarkable journey of pursuing a Ph.D. I also wish to thank Dr. Steven Buyske and Dr. Jinchuan Xing for their valuable advice on my research while on my dissertation committee.

Current and former members in the Matise/Gordon Lab, Dr. Anthony Musolf, Dr. Alejandro Q. Nato, Dr. Douglas Londono, and Rasheeda Williams, I cherish the exciting time spent with all of you.

Moreover, to all of my friends in the Genetics Department, the program of Molecular Biosciences, the iJOBS program, and at Rutgers, I treasure the companion from you.

Special thanks to my boyfriend, Guanjie Huang, who makes me laugh since the first day we met.

In the end, my deepest gratitude goes to my parents, Weiji Zhou and Yuqing Chen, and my grandmother, Naiming Gong. The unconditional love and support from them make me a better me as time goes by.

Table of contents

List of tables

List of illustrations

Chapter 1 Introduction

Human genetic disorders are unusual traits that are inherited within human genomes. In general, there are three categories of genetic disorders, single gene disorders, complex-trait disorders and chromosomal disorders. In single gene disorders, mode of inheritance may generally be classified as either autosomal dominant disorders, autosomal recessive disorders, X-linked disorders or mitochondrial disorders. The most commonly used methods for analyzing these genetic disorders are linkage analysis and association analysis. In association analysis, existing designs include case-control, family-based and quantitative trait. For complex-trait disorder association studies, GWAS (genome-wide association studies) are widely applied today. However, there are challenges with GWAS, among them those dealing with statistical design and analysis. The purpose of this work is to address some of those statistical challenges.

1.1 Genetic disorders

1.1.1 What are genetic disorders?

Genetic disorders are disease traits that are caused by changes in the genome that result in abnormal expression or gain or loss of function of one or more genes. New variants are introduced in each generation [1]. Some of these variants may be deleterious in that they have a harmful effect on the organism. Such mutations may occur spontaneously during one's life span, or may be inherited from parents.

1.1.2 Single gene, complex, and chromosomal disorders

As noted above, genetic disorders can generally be classified into three groups. We provide more details on each group directly below.

1.1.2.1 Single gene disorder

The first type is the single gene disorder, that requires only one mutation in a single gene to trigger the expression of the corresponding disorder. This type is also called a Mendelian disorder. The occurrence of this type of disorder is rare in the general population. Single gene disorders usually have identifiable inheritance patterns [2]. Examples of single gene disorders are cystic fibrosis, fragile X syndrome, sickle-cell disease, and Huntington's disease.

1.1.2.2 Complex-trait disorder

The second type of genetic disorder is a complex-trait disorder, that requires multiple factors for the expression of a disease trait [2]. These factors include multiple genes and interactions with the environment, and therefore genetic disorders do not typically exhibit distinct inheritance patterns. Complex-trait disorders are often developed by the interaction

between genes and environment (G×E), where genetics plays a significant role. Variation in a single gene is not expected to be sufficient for the expression of a complex-trait disorder phenotype, even though the expression may be dependent upon the number of mutations in relevant disease genes [2]. Complex-trait disorders occur more frequently in the general population than do most single gene disorders. Examples of complex-trait disorders include heart disease, Alzheimer disease, Parkinson disease, and asthma.

1.1.2.3 Chromosomal disorder

The last type is a chromosomal disorder, where the disease trait is caused by abnormal chromosome structure or number. This may occur due to deletion of a chromosome region (or regions), such as Angelman syndrome, or due to the occurrence of an abnormal number of chromosomes, such as Down syndrome.

1.1.3 Single Gene Modes of inheritance

Mode of inheritance is the pattern by which a monogenic trait is transmitted in families. Most Mendelian traits follow one of four modes of inheritance: autosomal dominant, autosomal recessive, X-linked and mitochondrial [3].

1.1.3.1 Autosomal dominant

Autosomal dominant inheritance requires only one disease allele to cause an individual to be affected by the disease. If the disease allele is inherited from parents, instead of being a de novo mutation, assuming the disease has 100% penetrance, at least one of the parents of the affected individual also expresses the disease phenotype. Because the inheritance of the disease allele is through autosomal inheritance, male or female offspring share equal possibility of inheriting the disease allele. There is a special case existing in this category

where the homozygous state of an autosomal dominant mutation presents a lethal phenotype, for example, Autosomal dominant osteopetrosis type II (ADO2) [4].

1.1.3.2 Autosomal recessive

Autosomal recessive inheritance requires two trait alleles in the same gene to express the trait phenotype. If those alleles are inherited, and neither of the parents of the affected individual expresses the trait, then both parents are carriers of the trait allele. An individual carrying just one recessive allele but not showing expression of the phenotype is called a "carrier". The probability of inheriting one disease allele from both carriers (parents) at the same time and expressing the disease phenotype is 0.25. Again, because the trait is autosomal, the sex of offspring does not impact this probability.

1.1.3.3 X-linked

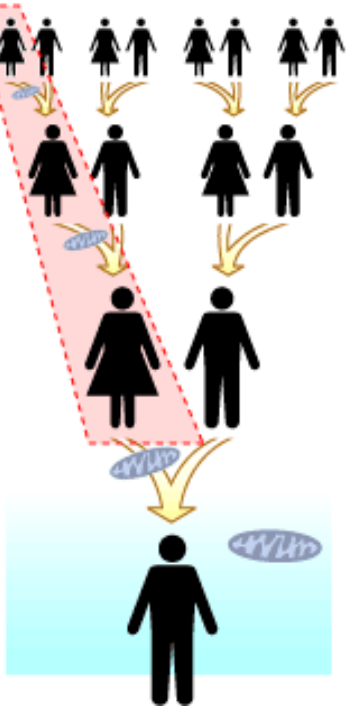
In X-linked inheritance, the disease locus is located on the X chromosome. This mode of inheritance may be further categorized into dominant or recessive inheritance. Males have only one X chromosome whereas females have two, therefore, X-linked traits occur in different proportions in males and females. With dominant disease alleles, the occurrence of the disease only requires one allele, resulting in that the probability of disease occurrence is usually the same in males and in females. For recessive alleles, only one disease allele is necessary for males to express the disease phenotype, but two are necessary in females. As a result, the probability of disease occurrence due to X-linked recessive inheritance is usually considerably higher in males than in females.

1.1.3.4 Mitochondrial

In mitochondrial inheritance, the trait allele is located in the mitochondrial DNA (mtDNA) and therefore is transmitted to offspring as cytoplasmic genes. mtDNA is strictly

maternally inherited [5, 6]. Though a few sperm mitochondrial DNA enter the egg, paternal mtDNA is not transmitted to offspring [7, 8]. Therefore, in mitochondrial inheritance, only the female parent transmits her mtDNA to all of her offspring [6, 7, 9] (Figure 1.1).

Figure 1.1 Mitochondrial DNA is only inherited from female parents



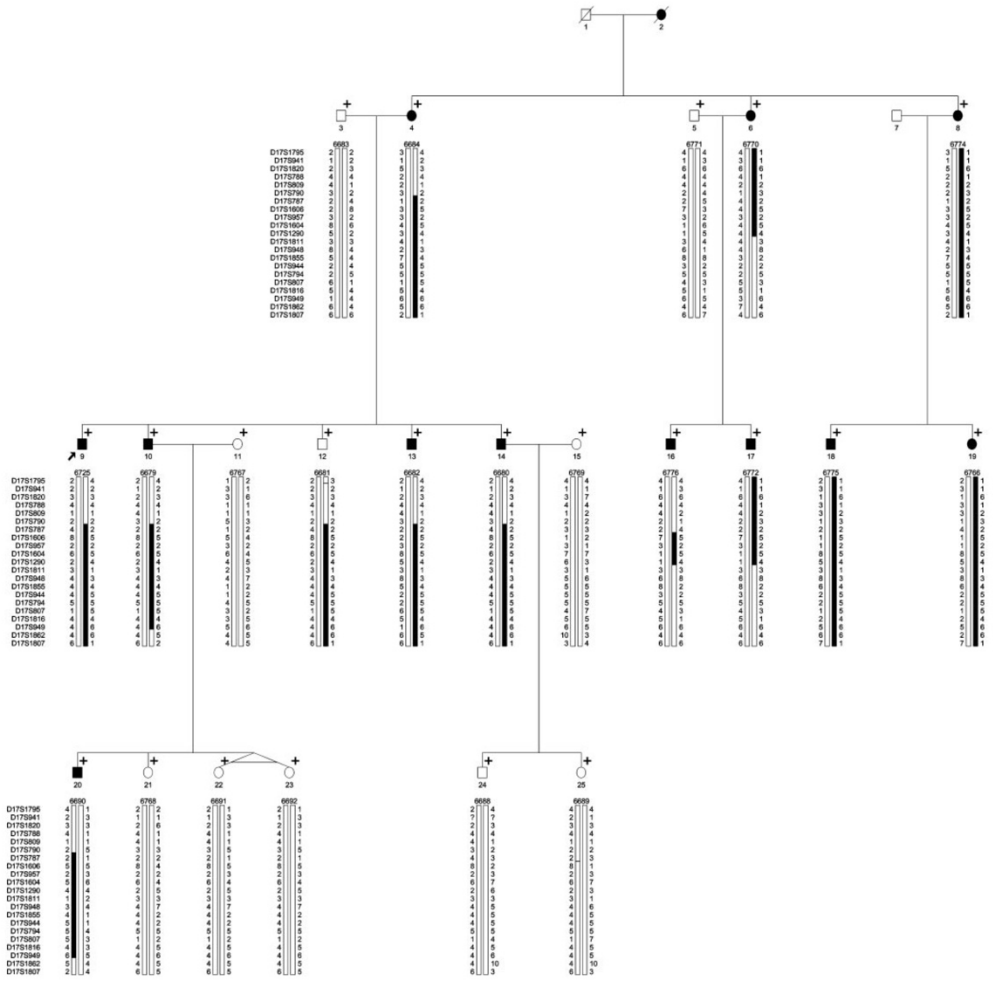
This figure is extracted from an online source [10].

1.2 Linkage analysis and association analysis

1.2.1 Linkage analysis

In human genetics, linkage analysis plays an important role in disease gene mapping. This strategy is achieved by estimating genetic distances from recombination events, by studying the co-inheritance of two loci (for example, a disease locus and a non-disease locus) within families from generation to generation [11]. In order to locate underlying disease loci, linkage analysis depends on the identification of recombinants; that is, recombinant haplotypes in children that are different from the parental haplotypes. In his book, *Analysis of Human Genetic Linkage*, Ott defines a haplotype as "the alleles (at different genes) received by an individual from one parent" [11]. For many single gene disorders, genes have been localized through application of linkage analysis to pedigrees with affected individuals. Recombination analysis is used to locate the disease gene (see example in Figure 1.2). A few examples of diseases caused by single underlying genes are: Cystic Fibrosis [12, 13], Tay Sachs [14], and Huntington's Disease [15-17]. However, linkage analysis has limitations: 1) it is most powerful for studying Mendelian monogenic disorders or oligogenic disorders (for complex-trait disorders, other strategies are required); 2) it requires family data to trace the recombination events [2].

Figure 1.2 Pedigree MYO-068 with familial high myopia

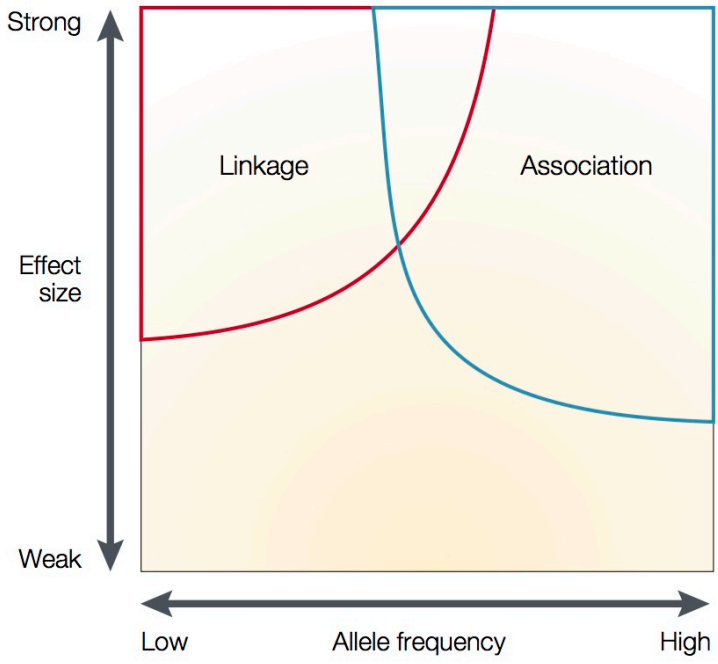


This figure and its legend are extracted from a published article [18]: Circles and squares: females and males, respectively; solid symbols: affected individuals. Diagonal lines through symbols: deceased individuals. The alleles for the most informative polymorphic markers are shown for each studied individual. Haplotypes were constructed based on the minimum number of recombinations between these markers. Solid bar: the chromosome assumed to carry the inherited disease allele; open bars: normal haplotypes. Nonparticipating family members are not shown. Only one of the monozygous twins 22 and 23 was used in the linkage analysis. Note that individuals 6, 16, and 17 are recombinant for the telomeric marker D17S1811. Individual 16 was recombinant for the centromeric marker D17S787.

1.2.2 Association analysis

When dealing with complex-trait disorders, association analysis is used to identify putative genes by testing the correlation between disease status and genetic variation [2]. Comment that methods for testing association (methods that incorporate linkage disequilibrium [LD] among loci; e.g., chi-square test of independence for alleles or genotypes, transmission disequilibrium test [TDT]) have been shown in some circumstance to be more powerful, statistically, for gene mapping than linkage analysis [19] (see Figure 1.3). These methods are potentially even more powerful with the advent of high-density single nucleotide polymorphisms (SNP) chip technology. With chips now containing 500K to 2.5million multiple SNPs per chip [20], virtually guaranteed to have LD present amongst markers (and that increases power of association methods). More details on genetic association analysis will be discussed in the next section.

Figure 1.3 The relative efficiency between linkage analysis and association analysis



This figure is extracted from a published review [21]. Association analysis is generally more powerful than linkage analysis when the allele of interest is frequent.

1.3 Genetic association analysis

1.3.1 What is genetic association analysis?

Unlike linkage analysis, that is often "scored over a limited number of observed generations" [22], association analysis utilizes a much larger number of generations to find the correlation between complex disease traits and genomic variants. Furthermore, in general, association analysis does not require any relationship between tested individuals.

1.3.2 Existing approaches for association analysis

1.3.2.1 Case-control test

One of the existing approaches of association analysis is the use of case-control studies. This approach is probably the approach most often observed in published reports of genetic association studies. In this kind of study, a set of individuals who are affected by the disease of interest are phenotyped and grouped as the case group, and a set of individuals who are not affected by the disease are phenotyped and grouped as the control group. Once the final set of cases and controls is determined, they are genotyped (typically using SNPs). Below, we mention some of the test statistics that are applied to case-control designs. Also, when conducting a case-control study, it is important to address potential confounding factors, such as population stratification [2].

1.3.2.2 Family-based association study

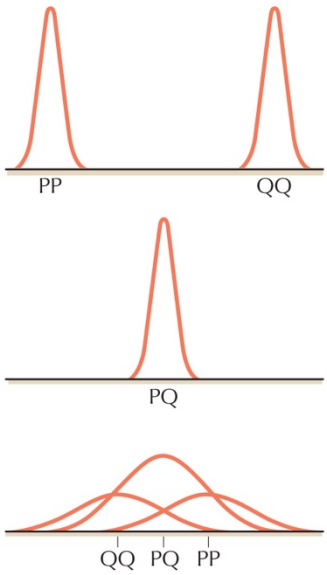
To avoid the potential type I error inflation caused by using stratified populations, the family-based association study approach was developed. It was first proposed by Falk and

Rubinstein (reference here) by testing the association between an affected child and the unaffected parents within the family [23]. The test statistic of this association was later defined as haplotype relative risk (HRR) [24]. After some statistical limitations were identified in HRR (e.g., the statistic may only be applied to simplex families (father, mother, affected child) [25], TDT was developed to test for the linkage between a marker and the disease, and applied to all SNPs whether or not previously identified for association [26, 27]. Apart from the original TDT test, there were several extensions of TDT [28-34], including one that allows for locus heterogeneity [30].

1.3.2.3 Quantitative traits association study

A quantitative disease trait exhibits a continuous distribution of disease phenotypic values (Figure 1.4). These values cannot be simply categorized and fit in Mendelian segregation ratios [35]. A general way to test the association between markers and quantitative traits is to take the mean train values into account [2, 36].

Figure 1.4 Example distribution of a quantitative trait



This figure and its legend are extracted from a book, *Evolution* [37]: The distribution of a quantitative trait in individuals with different marker genotypes (PP, PQ, QQ) in parental, F_1 and F_2 generations.

1.3.3 Genome-wide association studies (GWAS)

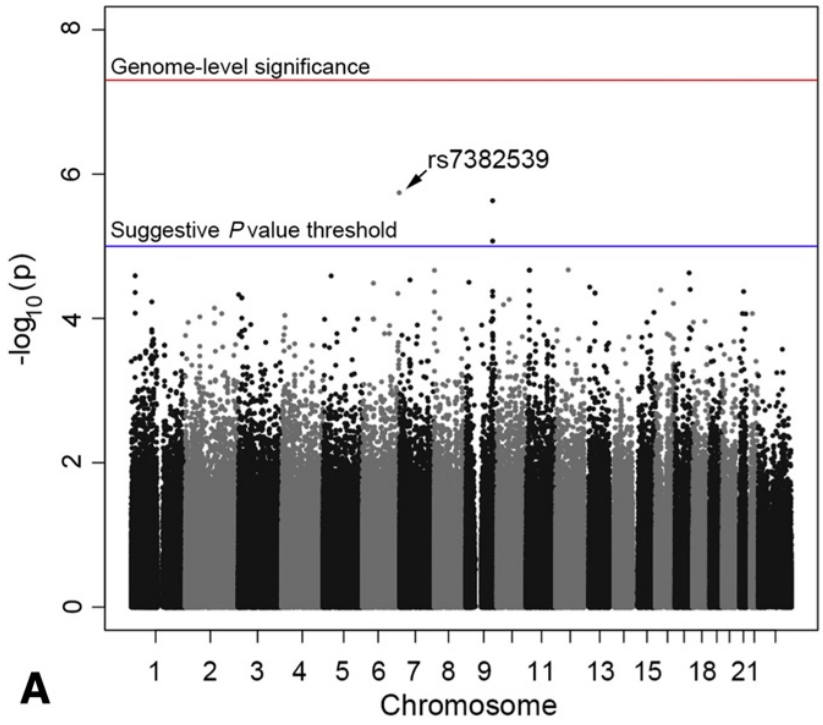
For complex diseases with causal genes contributing moderate effects to the disease risk, identifying candidate genes requires large-scale testing through an association study (Figure 1.3); one approach is GWAS [19]. In a GWAS study, typically thousands to tens of thousands of individuals carrying the disease of interest (as well as unmatched controls) are recruited and genotyped, and the genetic variants across genomes are then tested to identify the correlation between any of these variants and the disease [38, 39].

1.3.4 Significant genetic association

If a significant genetic association is found (see Figure 1.5 for example), there are three possible scenarios: 1) Direct association, meaning the genotyped SNPs identified are the

true causal variants of the disease of interest; 2) Indirect association, where the genotyped SNPs identified are adjacent markers and in LD with the true causal variant of the disease; 3) False-positive results, that may be caused by chance or systematic confounding (e.g. population stratification) [2]. A possible solution to distinguish disease causal variants from indirect association results is fine mapping [40].

Figure 1.5 Example of Manhattan plot showing all genotyped SNPs



This figure and its legend are extracted from a published article [41]: Manhattan plot showing all genotyped SNPs. X-axis: genomic coordinates of GWAS tested SNPs from chromosome 1 to X. Y-axis: significance level for each SNP on a $-\log_{10}$ scale. Genome-level significance, $P = 5 \times 10^{-8}$; suggestive P value threshold, $P = 1 \times 10^{-5}$.

1.3.5 Problems existing in case-control association analysis:

1.3.5.1 Quality control

Unlike family-based studies, case-control studies recruit participants with a variety of genetic backgrounds. After recruiting, samples from the participants are examined at different times and locations, and are often genotyped by different protocols and technicians. These variations may introduce uncontrollable bias in the data. Such biases may worsen if the individuals in the control group are recruited without confirming their negative status of disease affection [2]. Therefore, the qualities of the genomic data from study participants that are recruited separately may vary a great deal. These quality variations could lead to false-positive results of the analyses, that may eventually cost the time and money of a study, and possibly even negatively impact the health of the patients.

1.3.5.2 Departure from HWE

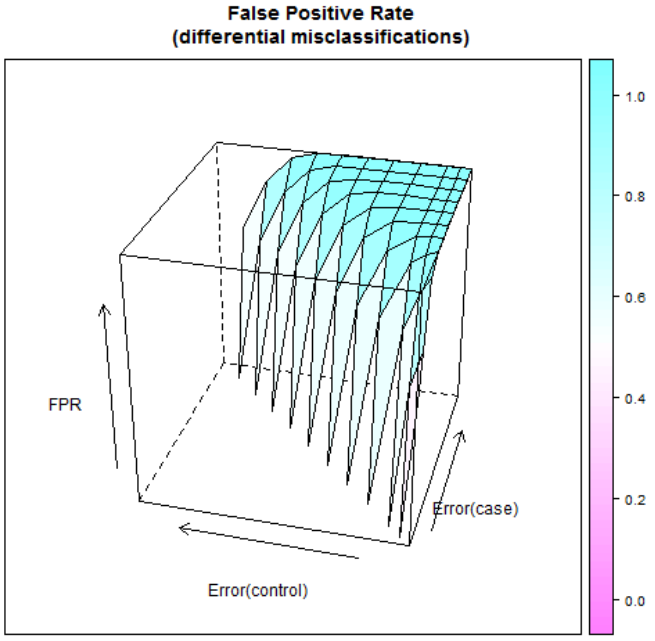
Hardy-Weinberg equilibrium (HWE) [42, 43] is assumed when testing the genetic association in case-control studies, and is used in association studies to control the quality of data. Any observation of departures from HWE in controls are associated with quality problems that may cause fluctuation in type I errors, especially in the presence of population stratification, genotyping errors and selection bias [44].

1.3.5.3 Genotyping misclassifications

Genotyping misclassification is also known as genotype error, and indicates that the true genotype is misclassified so that the reported genotype is different from the true underlying genotype. This may affect gene mapping and significantly decrease the power of the association test [45-47]. There are two types of misclassification, differential and non-differential. In the case of non-differential misclassification, the events that introduce

genotyping misclassifications are assumed to be the same between cases and controls. However, few real-world studies have this non-differential misclassification property. This non-differential misclassification, if applied improperly, may result in reduction of statistical power and biased estimates of parameters [48-52]. For differential misclassification, genotyping misclassification rates between cases and controls are different. These kinds of misclassifications are inevitable and could be introduced in every step of the study. However, even small differential genotyping misclassifications could result in significant problems in association analyses, such as type I error rate inflation and false-positive associations [48, 53, 54]. Figure 1.6 below shows the increase in false positive rate when differential misclassifications occur between cases and controls.

Figure 1.6 The distribution of false positive rate in differential misclassifications



Legend: FPR: false positive rate; Error(case): the misclassification rate in the case group; Error(control): the misclassification rate in the control group.

1.4 Motivation of our method

1.4.1 Direct association approach – identify causal variant

To solve the problem of decreased power in association studies with differential misclassifications, we developed a statistical test of association that is robust to differential misclassifications. In this method, we utilized next-generation sequencing raw data, instead of genotyping data, to discover the association between the disease of interest and causal variants in the genome.

1.4.2 Using NGS

With current technology, genotyping is generally designed to cover a subset of loci in the genome, that may cause missing data. Therefore, in an association study using genotyping data, the variants detected to be in association with the disease of interest may not be the causal variants, but merely variants in LD with the disease causal variants. If that is the case, a further approach of fine mapping is then required to identify causal variants. To avoid this kind of situation caused by missing genotypes, sequencing technology may be applied in association studies. Genome sequencing technology is able to reflect every variant in a genome region of interest if properly designed.

Next-generation sequencing (NGS), also known as high-throughput sequencing, was developed to meet demands for low-cost, high efficiency sequencing. Before the first appearance of parallelized sequencing technology of NGS in 2005, Sanger sequencing was used widely for nearly 30 years [55-58]. Compared to Sanger sequencing (that serves as gold standard sequencing with sequencing read lengths reaching 750 base pairs [bp]), NGS generates shorter reads (100 - 250 bp) and a higher number of sequencing reads from a single instrument run [55]. Moreover, NGS is able to sequence several samples at a single

run, that provides for a much shorter time in data generation for a large number of samples. However, the data quality from NGS is not always satisfactory, and the different algorithms in NGS downstream data processing may cause data quality variations between studies.

1.4.3 A test of association robust to differential misclassification

One feature of our method is its robustness to differential misclassifications. The misclassification we consider in our method is a combination of sequencing errors and systematic errors that exist between cases and controls in raw NGS data such as errors introduced by sample recruiting, sample preparation and downstream data processing. These misclassifications inevitably vary between cases and controls, and therefore, require our careful attention when designing an association test.

Reference:

1. McClellan, J. and M.C. King, Genetic heterogeneity in human disease. Cell, 2010. 141(2): p. 210-7.
2. Al-Chalabi, A. and L. Almasy, Genetics of Complex Human Diseases: A Laboratory Manual. 2009: Cold Spring Harbor Laboratory Press.
3. Board, P.C.G.E. NCI Dictionary of Genetics Terms: Mode of inheritance. The NCI Dictionary of Genetics Terms contains technical definitions for more than 200 terms related to genetics. These definitions were developed by the PDQ® Cancer Genetics Editorial Board to support the evidence-based, peer-reviewed PDQ cancer genetics information summaries.]. Available from: https://www.cancer.gov/publications/dictionaries/genetics-dictionary?cdrid=460196.
4. Alam, I., et al., Generation of the first autosomal dominant osteopetrosis type II (ADO2) disease models. Bone, 2014. 59: p. 66-75.
5. Pyle, A., et al., Extreme-Depth Re-sequencing of Mitochondrial DNA Finds No Evidence of Paternal Transmission in Humans. PLoS Genet, 2015. 11(5): p. e1005040.
6. Giles, R.E., et al., Maternal inheritance of human mitochondrial DNA. Proc Natl Acad Sci U S A, 1980. 77(11): p. 6715-9.
7. Zeviani, M. and S. Di Donato, Mitochondrial disorders. Brain, 2004. 127(Pt 10): p. 2153-72.
8. Schwartz, M. and J. Vissing, Paternal inheritance of mitochondrial DNA. N Engl J Med, 2002. 347(8): p. 576-80.
9. Ankel-Simons, F. and J.M. Cummins, Misconceptions about mitochondria and mammalian fertilization: implications for theories on human evolution. Proc Natl Acad Sci U S A, 1996. 93(24): p. 13859-63.
10. Marshalling the Evidence. Understanding Evolution. ; Available from: http://undsci.berkeley.edu/article/0_0_0/endosymbiosis_07.
11. Ott, J., Analysis of human genetic linkage. 3rd ed. 1999, Baltimore: Johns Hopkins University Press. xxiii, 382 p.
12. Eiberg, H., et al., Linkage relationships of paraoxonase (PON) with other markers: indication of PON-cystic fibrosis synteny. Clin Genet, 1985. 28(4): p. 265-71.
13. Tsui, L.C. and R. Dorfman, The cystic fibrosis gene: a molecular genetic perspective. Cold Spring Harb Perspect Med, 2013. 3(2): p. a009472.
14. Frisch, A., et al., Origin and spread of the 1278insTATC mutation causing Tay-Sachs disease in Ashkenazi Jews: genetic drift as a robust and parsimonious hypothesis. Hum Genet, 2004. 114(4): p. 366-76.
15. Bishop, D.T., et al., Strategies for efficient linkage analysis: example of Huntington's disease pedigrees. Genet Epidemiol Suppl, 1986. 1: p. 217-22.
16. Marazita, M.L. and M.A. Spence, Linkage analysis of G8 and Huntington's disease. Genet Epidemiol Suppl, 1986. 1: p. 247-50.

17. Sarfarazi, M., Report on genetic linkage analysis between Huntington's disease and the G8 DNA polymorphism. Genet Epidemiol Suppl, 1986. 1: p. 259-64.
18. Paluru, P., et al., New locus for autosomal dominant high myopia maps to the long arm of chromosome 17. Invest Ophthalmol Vis Sci, 2003. 44(5): p. 1830-6.
19. Risch, N. and K. Merikangas, The future of genetic studies of complex human diseases. Science, 1996. 273(5281): p. 1516-7.
20. Ha, N.T., S. Freytag, and H. Bickeboeller, Coverage and efficiency in current SNP chips. Eur J Hum Genet, 2014. 22(9): p. 1124-30.
21. Ardlie, K.G., L. Kruglyak, and M. Seielstad, Patterns of linkage disequilibrium in the human genome. Nat Rev Genet, 2002. 3(4): p. 299-309.
22. Ott, J., J. Wang, and S.M. Leal, Genetic linkage analysis in the age of whole-genome sequencing. Nat Rev Genet, 2015. 16(5): p. 275-84.
23. Rubinstein, P., et al., HLA antigens and islet cell antibodies in gestational diabetes. Hum Immunol, 1981. 3(3): p. 271-5.
24. Falk, C.T. and P. Rubinstein, Haplotype relative risks: an easy reliable way to construct a proper control sample for risk calculations. Ann Hum Genet, 1987. 51(Pt 3): p. 227-33.
25. Ott, J., Statistical properties of the haplotype relative risk. Genet Epidemiol, 1989. 6(1): p. 127-30.
26. Spielman, R.S. and W.J. Ewens, The TDT and other family-based tests for linkage disequilibrium and association. Am J Hum Genet, 1996. 59(5): p. 983-9.
27. Spielman, R.S., R.E. McGinnis, and W.J. Ewens, Transmission test for linkage disequilibrium: the insulin gene region and insulin-dependent diabetes mellitus (IDDM). Am J Hum Genet, 1993. 52(3): p. 506-16.
28. Deng, H.W. and W.M. Chen, The power of the transmission disequilibrium test (TDT) with both case-parent and control-parent trios. Genet Res, 2001. 78(3): p. 289-302.
29. Haldar, T. and S. Ghosh, Statistical equivalent of the classical TDT for quantitative traits and multivariate phenotypes. J Genet, 2015. 94(4): p. 619-28.
30. Londono, D., et al., TDT-HET: a new transmission disequilibrium test that incorporates locus heterogeneity into the analysis of family-based association data. BMC Bioinformatics, 2012. 13: p. 13.
31. Rice, J.P., et al., TDT with covariates and genomic screens with mod scores: their behavior on simulated data. Genet Epidemiol, 1995. 12(6): p. 659-64.
32. Sham, P.C. and D. Curtis, An extended transmission/disequilibrium test (TDT) for multi-allele marker loci. Ann Hum Genet, 1995. 59(Pt 3): p. 323-36.
33. Waldman, I.D., B.F. Robinson, and S.A. Feigon, Linkage disequilibrium between the dopamine transporter gene (DAT1) and bipolar disorder: extending the transmission disequilibrium test (TDT) to examine genetic heterogeneity. Genet Epidemiol, 1997. 14(6): p. 699-704.
34. Waldman, I.D., B.F. Robinson, and D.C. Rowe, A logistic regression based extension of the TDT for continuous and categorical traits. Ann Hum Genet, 1999. 63(Pt 4): p. 329-40.
35. St Clair, D.A., Quantitative disease resistance and quantitative resistance Loci in breeding. Annu Rev Phytopathol, 2010. 48: p. 247-68.

36. Boerwinkle, E., R. Chakraborty, and C.F. Sing, The use of measured genotype information in the analysis of quantitative phenotypes in man. I. Models and analytical methods. Ann Hum Genet, 1986. 50(Pt 2): p. 181-94.
37. Nicholas H. Barton, D.E.G.B., Jonathan A. Eisen, David B. Goldstein, and Nipam H. Patel, Evolution. 2007: Cold Spring Harbor Laboratory Press.
38. O'Brien, S.J., Stewardship of human biospecimens, DNA, genotype, and clinical data in the GWAS era. Annu Rev Genomics Hum Genet, 2009. 10: p. 193-209.
39. MacArthur, J., et al., The new NHGRI-EBI Catalog of published genome-wide association studies (GWAS Catalog). Nucleic Acids Res, 2016.
40. Devlin, B. and N. Risch, A comparison of linkage disequilibrium measures for fine-scale mapping. Genomics, 1995. 29(2): p. 311-22.
41. Tsai, E.A., et al., THBS2 Is a Candidate Modifier of Liver Disease Severity in Alagille Syndrome. Cell Mol Gastroenterol Hepatol, 2016. 2(5): p. 663-675 e2.
42. Hardy, G.H., Mendelian Proportions in a Mixed Population. Science, 1908. 28(706): p. 49-50.
43. Stern, C., The Hardy-Weinberg Law. Science, 1943. 97(2510): p. 137-8.
44. Minelli, C., et al., How should we use information about HWE in the meta-analyses of genetic association studies? Int J Epidemiol, 2008. 37(1): p. 136-46.
45. Gordon, D., et al., Increasing power for tests of genetic association in the presence of phenotype and/or genotype error by use of double-sampling. Stat Appl Genet Mol Biol, 2004. 3: p. Article26.
46. Gordon, D. and S.J. Finch, Factors affecting statistical power in the detection of genetic association. J Clin Invest, 2005. 115(6): p. 1408-18.
47. Gordon, D., et al., Power and Sample Size Calculations for Case-Control Genetic Association Tests when Errors Are Present: Application to Single Nucleotide Polymorphisms. Human Heredity, 2002. 54(1): p. 22-33.
48. Ahn, K., D. Gordon, and S.J. Finch, Increase of rejection rate in case-control studies with the differential genotyping error rates. Stat Appl Genet Mol Biol, 2009. 8: p. Article25.
49. Mote, V.L. and R.L. Anderson, An Investigation of the Effect of Misclassification on the Properties of Chi-2-Tests in the Analysis of Categorical Data. Biometrika, 1965. 52: p. 95-109.
50. Gordon, D., et al., A transmission disequilibrium test for general pedigrees that is robust to the presence of random genotyping errors and any number of untyped parents. Eur J Hum Genet, 2004. 12(9): p. 752-61.
51. Edwards, B.J., et al., Power and sample size calculations in the presence of phenotype errors for case/control genetic association studies. BMC Genet, 2005. 6: p. 18.
52. Ahn, K., et al., The effects of SNP genotyping errors on the power of the Cochran-Armitage linear trend test for case/control association studies. Ann Hum Genet, 2007. 71(Pt 2): p. 249-61.
53. Moskvina, V., et al., Effects of differential genotyping error rate on the type I error probability of case-control studies. Hum Hered, 2006. 61(1): p. 55-64.
54. Clayton, D.G., et al., Population structure, differential bias and genomic control in a large-scale, case-control association study. Nat Genet, 2005. 37(11): p. 1243-6.

55. Schuster, S.C., Next-generation sequencing transforms today's biology. Nat Methods, 2008. 5(1): p. 16-8.
56. Sanger, F., et al., Nucleotide sequence of bacteriophage phi X174 DNA. Nature, 1977. 265(5596): p. 687-95.
57. Smith, M., et al., DNA sequence at the C termini of the overlapping genes A and B in bacteriophage phi X174. Nature, 1977. 265(5596): p. 702-5.
58. Margulies, M., et al., Genome sequencing in microfabricated high-density picolitre reactors. Nature, 2005. 437(7057): p. 376-80.

Chapter 2 Methods

This chapter describes the development of a statistical method, $LRT_{ae,NGS}$, that tests for association between disease phenotypes and multiple single nucleotide polymorphisms (SNPs). Such association may indicate that certain multi-locus genotypes (MLGs) are "risk" genotypes. A novel feature of this method is its robustness to misclassification that may exist in next-generation sequencing (NGS) data. The data used in this approach are affection status, observed alternative read counts, and sequencing coverage values of multiple genetic loci (SNPs), from a group of phenotyped individuals.

The first section of this chapter describes the process of developing the statistical method. It gives readers step-by-step explanations of how this approach was created. At the beginning of the section, we clarify the terms and notation mentioned in the equations. Then we explain the key equations used in the algorithm, and how they interact and form the expectation-maximization (EM) algorithm. The second section of this chapter explains the process of how the observed datasets are simulated. These datasets are used to evaluate the performance of the method. We develop a software program implementing our method. The source code and instructions of the program are provided in the Appendix.

2.1 Key terms and notation used in this chapter

2.1.1 Definitions of terms used throughout this work

Allele frequency

An allele frequency is the proportion of a particular allele (at a given locus on a chromosome; from this point forward, any locus is assumed to be on a chromosome) in a population [2]. It is always expressed as a percentage. We may estimate the allele frequency from a finite sample by counting the number of copies of the particular allele, and dividing by the total number of copies of all alleles (at the locus) in the finite sample.

For example, consider a locus in a population, where the locus has two alleles, A and a. In the population, there are three possible genotypes: AA (homozygous genotype for *A* allele); Aa (heterozygous genotype); and aa (homozygous genotype for *a* allele). Now consider a random sample of ten individuals from this population. Among the ten individuals, we determine that, two of them are homozygous for the AA genotype, five are heterozygotes Aa, and three are homozygous for the aa genotype. The estimated frequency of allele A is $(2\times2+5)/(2\times10)=0.45$, and the estimated frequency of allele a is $(3\times2+5)/(2\times10)=0.55$.

Genotype frequency

As with the allele frequency, for a given locus, a genotype frequency is the proportion of a given genotype in a population, where the genotype is one of the possible genotypes at the locus [1]. It is expressed as a percentage. Analogous to the estimated allele frequencies, each estimated genotype frequency (denoted g_j) is given by:

$$g_j = \frac{N_j}{N}. \tag{2.1}$$

Here, j refers to the j^{th} genotype, $1 \leq j \leq n$. That is, there is an ordering to the genotypes, and there are n of them. Taking the example from the allele frequency, the frequency of genotype AA is $2/10 = 0.2$, the frequency of genotype Aa is $5/10 = 0.5$, and the frequency of genotype aa is $3/10 = 0.3$.

Multi-locus genotype (MLG)

A multi-locus genotype is the combination of specific genotypes across two or more loci.

Multi-locus genotype frequency

A MLG frequency is the proportion of a MLG in a population. It is always expressed as a percentage. MLG frequency is the number of individuals with a given MLG m (N_m) divided by the total number of individuals of population (N):

$$g_m = \frac{N_m}{N}. \tag{2.2}$$

Due to possible linkage disequilibrium, the MLG frequency may be different from the product of the genotype frequencies on those loci with given genotypes.

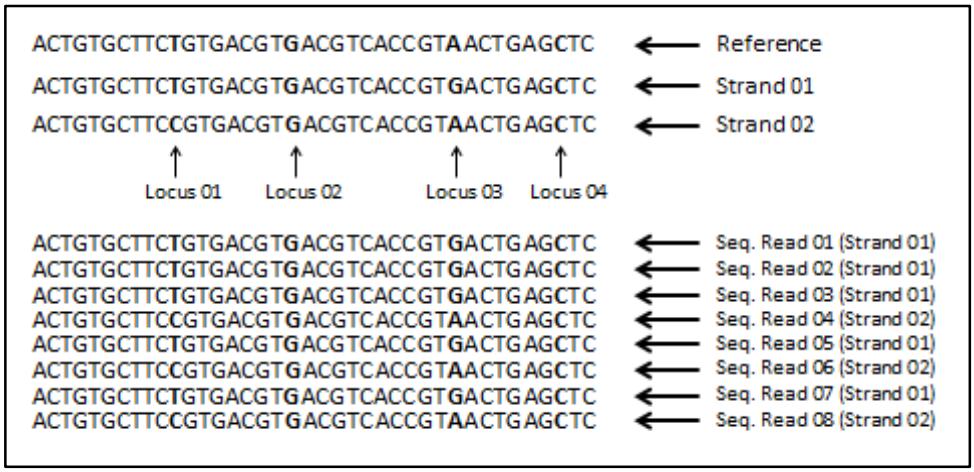
Sequencing coverage

Sequencing coverage is the number of times a base pair is observed for individual k at locus m for a given NGS sequencing experiment (Figure 2.1)[3, 4].

Alternative allele read count

Alternative allele read count is the number of times a given alternative allele is observed for individual k at locus m, for a given NGS sequencing experiment (number bounded between 0 and coverage) (see Figure 2.1 below) [5].

Figure 2.1 Example of sequencing coverage and alternative allele read count of an individual



Top panel: A region of the reference sequence and two DNA sequences for an individual (strand 01 and strand 02). As indicated, there are four markers sequenced (locus 1-4) in this region. The individual is heterozygous at locus 01 (genotype T/C), homozygous at locus 02 (genotype G/G), heterozygous at locus 03 (genotype G/A) and homozygous at locus 04 (genotype C/C). **Bottom panel:** The sequence reads consist of random selections of one or two strands. In this example, strand 01 is selected to be sequenced 5 out of 8 times and strand 02 is selected to be sequenced 3 out of 8 times. For these four markers, strand 01 consists of the reference alleles at locus 01, 02 and 04, while strand 02 consists of the reference allele at locus 02, 03 and 04. Finally, for this individual, we note that the sequencing coverage for this region of the genome is 8x (8 sequence reads), and the alternative allele read counts for locus 01 to 04 are 3, 0, 5 and 0.

Sequencing misclassification rate

Sequencing misclassification rate is the proportion of sequenced reads in which sequenced alleles are misclassified as an allele other than true allele in all sequenced reads. It is the number of sequenced reads with misclassified alleles divided by the total number of sequenced reads:

$$\varepsilon = \frac{number\ of\ sequence\ reads\ with\ misclassified\ alleles}{total\ number\ of\ sequenced\ reads}.$$

2.1.2 Notation

M = the number of SNPs considered when determining the multi-locus genotypes (MLGs) and their frequencies.

t(superscript) = indicates the true value of the variable.

$v_{m,k}^{t}$= sequencing coverage for individual k at locus m (no misclassification is assumed; note, $1 \leq m \leq M$).

$x_{m,k}$ = alternative allele read count for individual k at locus m.

i_k^t = phenotype value for individual k (no misclassification is assumed) = value indicating whether individual k is affected by the disease ($i_k^t = 1$) or unaffected ($i_k^t = 0$).

$j_{m,k}^{t}$ = genotype value for individual k at locus m (no misclassification is assumed) = value indicating genotype with homozygous reference alleles ($j_{m,k}^{t} = 0$), heterozygous reference allele/alternative allele ($j_{m,k}^{t} = 1$), and homozygous alternative alleles ($j_{m,k}^{t} = 2$). This value is a latent variable; that is, it is not part of the observed data.

$\varepsilon_{i_k^t}^{t}$ = misclassification rate of alternative and reference alleles (which varies with phenotype) for individual k.

2.1.3 Mathematical principles

Hardy-Weinberg equilibrium

In this work, we specify that single or multi-locus genotype frequencies follow Hardy-Weinberg Equilibrium (HWE) proportions. The Hardy-Weinberg Equilibrium is an important concept in population genetics in that it describes a condition under which genotype frequencies may be written as functions of allele frequencies [6, 7]. In a single locus case with two alleles, A and a, with the allele frequency of $f(A) = p$ and $f(a) = q$, the expected genotype frequencies are $f(AA) = p^2$ for the genotype AA; $f(Aa) = 2pq$ for the genotype Aa; and $f(aa) = q^2$ for the genotype aa. Consider a case with two loci, *Locus 1* with alleles A and a, and *Locus 2* with alleles B and b. Denote the allele frequency of $f(A) = p$ and $f(B) = q$. Under random mating and certain population assumptions, Hardy-Weinberg equilibrium will be achieved after one generation, as stated above. After

many of generations the joint genotypic frequencies at *Locus 1* and *Locus 2* will be independent. For instance:

Genotype	**Frequency**
$AABB$	$p^2 \times q^2$
$AABb$	$p^2 \times 2q(1-q)$
$AAbb$	$p^2 \times (1-q)^2$
$AaBB$	$2p(1-p) \times q^2$
etc	…

Contingency table

A contingency table displays the multivariate frequency distribution of a study's variables, in a matrix format [8, 9]. Differences in genotype frequencies may be indicative of a disease locus within close proximity of the sequenced marker. From the example of a study of genotype frequency differences in affection status (Table 2.1), there are two variables, affection status (affected or unaffected) and genotype (AA, Aa or aa), which are cross-classified. Table 2.1 gives an example of a contingency table. Suppose that 200 individuals (100 affected patients and 100 unaffected healthy people) are randomly sampled from a large population. Each has a genotype (here we specify that the typed locus has two alleles, A and a) that is determined by the underlying frequency distribution for the two affection-status groups (affected and unaffected). Table 2.1 provides the numbers of individuals who satisfy the following conditions:

Affected and with genotype AA (64), affected and with genotype Aa (32), affected with genotype aa (4), unaffected with genotype AA (36), unaffected with genotype Aa (48), and unaffected with genotype aa (16). The data is organized in such a way to test the null hypothesis that genotype frequencies are equal in affected individuals and unaffected individuals. In this example, the value of the chi-square test statistic is 18.24 with a p-value of 0.000109 (degrees of freedom = 2). Based on this information, one is able to determine whether or not to reject the null hypothesis (in this example, we reject the null hypothesis at the significance level of 0.05).

Table 2.1 Contingency table example of a study of genotype frequency differences

	AA	Aa	aa	Row Totals
Affected Individuals	64	32	4	100
Unaffected Individuals	36	48	16	100
Column Totals	100	80	20	200 (Grand Total)

Example of contingency table with genotypes *AA*, *Aa*, *aa* in a population of 200 individuals (100 affected patients and 100 unaffected healthy people).

2.1.4 Statistical terms

Likelihood and log-likelihood

The likelihood, L, of an observed data set is the hypothetical probability of a specific outcome from an event that has already occurred. While probability typically refers to the occurrence of future events, in this case, the likelihood refers to past events with known outcomes. In his book, *Likelihood*, Edwards provides the classic definition of likelihood [9]:

$$L(data\ variables \mid data) = K \times \Pr(\text{variables} \mid \text{data})$$

(K is some arbitrary constant)

The log-likelihood, $\ln(L)$, is the logarithm of the likelihood. It is used in statistical hypothesis testing.

Likelihood ratio

The likelihood ratio quantitatively measures how much more probable the observed data is under one set of parameter settings than another set [10, 11]. Our likelihood ratio test requires nested models (*e.g.*, the complete likelihood and the likelihood with observed data), where the more complex models can be transformed into simpler ones by applying a set of constraints on the parameters. For our work, we apply a specific set of parameter constraints (described below) to obtain our likelihood ratio test.

Log-likelihood ratio test (LRT) statistic

The log-likelihood ratio test statistic is:

$$\chi^2 = 2\ln(likelihood\ for\ alternative\ model) - 2\ln(likelihood\ for\ null\ model)$$

Under the null hypothesis that MLG frequencies are equal between cases and controls, this statistic is distributed as a central chi-square test statistic, with degrees of freedom equal to the difference in the number of parameters in the two models [12].

The likelihood ratio test rejects the null hypothesis if the p-value of the statistic χ^2 for a given data set is less than the user-specified value. This value is referred to as the significance level of the test.

Chi-square test of independence on genotypes

When assuming the null hypothesis is true, the probability distribution of the log-likelihood ratio statistic can be approximated using Wilk's theorem [13]:

As the sample size n approaches ∞, the test statistic $-2\log(\Lambda)$ for a nested model will be asymptotically a central χ^2-distribution with degrees of freedom equal to the difference in the number of parameters in the two models:

$$d.f. = d.f.(unconstrained\ model) - d.f.(null\ model)$$

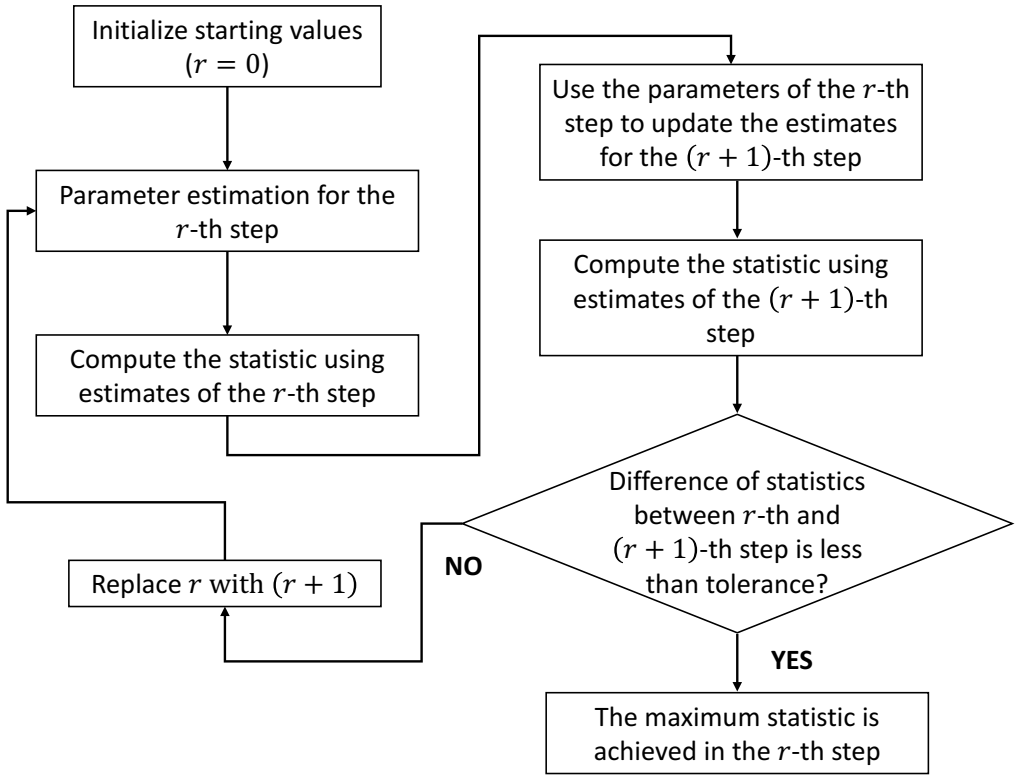
For example, if we consider 3 loci and each locus is bi-allelic, the degrees of freedom for the null model is $3^3 - 1 = 26$, while the degrees of freedom for the unconstrained model

is $2\times(3^3-1)=52$. Therefore, if the genotype frequencies between cases and controls are equal, the probability distribution of the log-likelihood ratio statistic is a central χ^2-distribution with $(52-26)=26$ degrees of freedom.

Expectation-Maximization algorithm

The Expectation-Maximization (EM) algorithm is a method to find maximum likelihood estimates of parameters by iteration of steps computing expectation and maximization [14-16]. The expectation step creates a function for the expectation of the log-likelihood evaluated using the current parameter estimates. The maximization step computes parameters maximizing the expected log-likelihood from the expectation step, where the parameters will be used in the next expectation step until the algorithm meets the maximized log-likelihood by reaching the tolerance. A general workflow of the EM algorithm is shown in Figure 2.2.

Figure 2.2 A general workflow of EM algorithm



Bayes' Rule

Bayes' rule describes the conditional probability of event A, given that event B is true. The conditional probability is equal to the probability of A multiplied by the probability of B (given that A is true), and then divided by the probability of B [17]:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)}. \tag{2.3}$$

For example, given a group of 100 individuals, 60 are unaffected (control) and 40 are affected by a disease (case). Among the controls, 30 of them have the genotype AA, and the other 30 have the genotype Aa. Among the cases, 15 of them has the genotype AA, and the other 25 has the genotype Aa. By applying Bayes' rule, we can therefore compute an individual's probability of being affected or unaffected by the disease with a given genotype. For instance,

$$P(\text{affected}|Aa) = \frac{P(\text{affected})P(Aa|\text{affected})}{P(Aa)} = \frac{\frac{40}{100}\times\frac{25}{40}}{\frac{30+25}{100}} \approx 0.45.$$

Bayesian posterior probability

Bayesian posterior probability is the probability of parameters θ given the relevant evidence X, $p(\theta|X)$ [18]. The posterior probability can then be described by Bayes' Rule as [19]:

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}. \tag{2.4}$$

Odds ratio

Odds ratio (OR) quantifies the association between the presence or absence of two properties in a given population [20-22]. Given two properties, A and B, a portion of the individuals who are affected by a disease were previously exposed to A (D_A), while the others were exposed to B (D_B). Also, some of the unaffected individuals were exposed to A (H_A), and the rest were exposed to B (H_B). We can then compute the odds ratio as,

$$OR = \frac{D_A/H_A}{D_B/H_B}. \tag{2.5}$$

2.2 Development of the likelihood ratio test

For the developed likelihood ratio test, $LRT_{ae,NGS}$, the null hypothesis states that there is no difference in the MLG frequencies between cases and controls, while the alternative hypothesis states that, for at least one MLG, the case and control frequencies are not equal.

2.2.1 Log-likelihood of the observed data

With the total number of markers M, the likelihood function of the multiple-locus alternative allele count data involves 3^M genotype configurations. With k indicating the k^{th} position in N sequenced individuals, with m being the marker number, if we set $\boldsymbol{x_k} = (x_{1,k}, \cdots, x_{M,k}), \boldsymbol{v_k^t} = (v_{1,k}^t, \cdots, v_{M,k}^t), \boldsymbol{G_k^t} = (j_{1,k}^t, \cdots, j_{M,k}^t)$, the complete likelihood function may be written as:

$$\begin{aligned} L_{c,k} &= \prod_{k=1}^{N} \Pr(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t) \\ &= \prod_{k=1}^{N} \prod_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \Pr(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t, \boldsymbol{G_k^t})^{I(\boldsymbol{G_k^t})}. \end{aligned} \tag{2.6}$$

$I(\ \ )$ is the indicator function indicating the MLG in $\Pr(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t, \boldsymbol{G_k^t})$ is $\boldsymbol{G_k^t}$. If the value of the $I(\ \ )$ is known, then this equation demonstrates the complete likelihood; in other words, each individual's $\boldsymbol{G_k^t}$ is known.

Upon finding the logarithm of both sides, Equation (2.6) can be rewritten as,

$$ln\left(L_{c,k}\right) = \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} I(\boldsymbol{G_k^t}) \times ln[\Pr(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t, \boldsymbol{G_k^t})]\,. \tag{2.7}$$

Considering the observed data, $(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)$, Equation (2.7) may be rewritten as:

$$\begin{aligned} &E\left[ln\left(L_{c,k}\right)\middle|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)\right] \\ &= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)] \\ &\times ln[\Pr(\boldsymbol{x_k}|\boldsymbol{v_k^t}, i_k^t, \boldsymbol{G_k^t}) \times \Pr(\boldsymbol{G_k^t}|\boldsymbol{v_k^t}, i_k^t) \times \Pr(\boldsymbol{v_k^t}|i_k^t) \times \Pr(i_k^t)]. \end{aligned} \tag{2.8}$$

Because the sequencing coverage vector $\boldsymbol{v_k^t}$ has no effect on the MLG $\boldsymbol{G_k^t}$ **,** and the affection status i_k^t has no effect on the sequencing coverage vector $\boldsymbol{v_k^t}$ (according to the sequencing technology characteristics, sequencing coverage value is independent of the affection status), Equation (2.8) can be rewritten as:

$$E\left[ln\left(L_{c,k}\right)\middle|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)\right]$$

$$= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)]$$

$$\times ln[\Pr(\boldsymbol{x_k}|\boldsymbol{v_k^t}, i_k^t, \boldsymbol{G_k^t}) \times \Pr(\boldsymbol{G_k^t}|i_k^t) \times \Pr(\boldsymbol{v_k^t}) \times \Pr(i_k^t)]$$

$$= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)] \times \ln[(1 - i_k^t)$$

$$\times \Pr(\boldsymbol{x_k}|\boldsymbol{v_k^t}, i_k^t = 0, \boldsymbol{G_k^t}) \times \Pr(\boldsymbol{G_k^t}|i_k^t = 0) \times \Pr(i_k^t = 0)$$

$$+ i_k^t \times \Pr(\boldsymbol{x_k}|\boldsymbol{v_k^t}, i_k^t = 1, \boldsymbol{G_k^t}) \times \Pr(\boldsymbol{G_k^t}|i_k^t = 1) \times \Pr(i_k^t = 1)]$$

$$+ \gamma, \tag{2.9}$$

where $\gamma = \sum_{k=1}^{N} \ln\left(\Pr(\boldsymbol{v_k^t})\right)$. The last equality follows from the fact that the phenotypes and genotypes are independent of the coverage.

Summing all true genotype vectors $\boldsymbol{G_k^t}$ is equivalent to summing each locus m's genotype value from 0 to 2. In this work, we specify that, conditional on the underlying data (including the genotype vector $\boldsymbol{G_k^t}$), the observed alternative allele counts are independent. We specify that, conditional on sequencing coverage, affection status and true underlying genotype, the observed alternative allele counts follow a binomial distribution. Written another way, the equation reads:

$$\Pr\left(\boldsymbol{x_k}\middle|\boldsymbol{v_k^t} = \left(v_{1,k}^t, \cdots, v_{M,k}^t\right), i_k^t, \boldsymbol{G_k^t} = \left(j_{1,k}^t, \cdots, j_{M,k}^t\right)\right) = \prod_{m=1}^{M} Bin(x_{m,k}; v_{m,k}^t; p(i_k^t, j_{m,k}^t)). \tag{2.10}$$

Before proceeding with the log-likelihood and the test statistics, the binomial probability mass function is computed. We compute the binomial probability mass function $Bin(x_{m,k}|v_{m,k}^t, i_k^t, j_{m,k}^t)$ for each single locus m to calculate the probability of alternative allele read count $x_{m,k}$ given sequencing coverage $v_{m,k}^t$, affection status i_k^t, and single locus genotype $j_{m,k}^t$. For example, in $Bin(x_{1,k}|v_{1,k}^t, i_k^t, j_{1,k}^t)$, the subscript "1" indicates that it is the first marker, while the subscript "k" indicates that it is the k-th individual in the sample. The superscript "t" indicates that it is the true value, which means no misclassification occurs for this value. The determination of the probability mass function is necessary for computation of the log-likelihoods.

In general, at the m^{th} locus, and for the k^{th} individual, in the binomial distribution $Bin(x_{m,k}; v_{m,k}^t; p(i_k^t, j_{m,k}^t))$, the number of "successes" (i.e., observing the alternative allele instead of the reference allele) is $x_{m,k}$, the total number of experiments is $v_{m,k}^t$, and the probability of a success for any given experiment (i.e., reading a sequence) is $p(i_k^t, j_{m,k}^t) = \left(\frac{2-j_{m,k}^t}{2}\varepsilon_{i_k^t}^t + \frac{j_{m,k}^t}{2}(1-\varepsilon_{i_k^t}^t)\right)$.

Also, $\varepsilon_{i_k^t}^t$ is the probability in individual k of misclassifying alleles. We specify a symmetric error model in which, given the alternative allele A and the reference allele a, the misclassification probability of observing the alternative allele A when the true allele is a, equals the misclassification probability of observing the reference allele a when the true allele is A:

$$\varepsilon^t_{i^t_k,(a\to A)} = \varepsilon^t_{i^t_k,(A\to a)} = \varepsilon^t_{i^t_k}. \tag{2.11}$$

To demonstrate that the formula correctly computes the misclassification probabilities, consider the following: for genotype $j^t_{m,k} = 0$ and $j^t_{m,k} = 2$. For instance, when $j^t_{m,k} = 0$, the individual's genotype is a/a. Thus, every observed alternative allele A is really a reference allele a that has been misread. The number of misclassifications is $x_{m,k}$, so that $x_{m,k}$ follows a binomial distribution, with a probability of success for each experiment equal to $\varepsilon^t_{i^t_k} = \left(\frac{2-0}{2}\varepsilon^t_{i^t_k} + \frac{0}{2}(1-\varepsilon^t_{i^t_k})\right)$, given $v^t_{m,k}$ trials. Similarly, when $j^t_{m,k} = 2$, every read of alternative allele A is now a correct read from the genotype A/A. Thus, $x_{m,k}$ follows a binomial distribution with probability of success for each experiment equal to $\left(1-\varepsilon^t_{i^t_k}\right) = \left(\frac{2-2}{2}\varepsilon^t_{i^t_k} + \frac{2}{2}(1-\varepsilon^t_{i^t_k})\right)$, given $v^t_{m,k}$ trials. For the heterozygote genotype $j^t_{m,k} = 1$, the probability of alternative allele A being read on a single trial is the sum of two probabilities:

$$\begin{aligned}
&\Pr(\text{sequenced allele } = a)\times \Pr(\text{observed allele} = A \mid \text{sequenced allele} = a)\\
&+ \Pr(\text{sequenced allele} = A) \times \Pr(\text{observed allele} = A \mid \text{sequenced allele} = A),\\
&= \frac{1}{2}\varepsilon^t_{i^t_k} + \frac{1}{2}\left(1-\varepsilon^t_{i^t_k}\right),\\
&= \frac{1}{2}.
\end{aligned}$$

It follows that we can rewrite Equation (2.9) under the null hypothesis as:

$$L_{H_0} = E\left[\ln\left(L_{c,k}\right)|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)\right]$$

$$= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)]$$

$$\times \ln\left[(1 - i_k^t) \times \prod_{m=1}^{M} Bin\left(x_{m,k}; v_{m,k}^t; p\left(i_k^t = 0, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),*} \times \Pr(i_k^t = 0) + i_k^t\right.$$

$$\left.\times \prod_{m=1}^{M} Bin\left(x_{m,k}; v_{m,k}^t; p\left(i_k^t = 1, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),*} \times \Pr(i_k^t = 1)\right]$$

$$+ \gamma,$$

(2.12)

where $\boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),*}$ indicates that the frequencies of multi-locus genotype $\left(j_{1,k}^t, \cdots, j_{M,k}^t\right)$ are equal under different affection statuses (affected and unaffected).

Similarly, the alternative hypothesis is written as:

$$L_{H_1} = E\left[ln\left(L_{c,k}\right)|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)\right]$$

$$= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)]$$

$$\times \ln\left[(1 - i_k^t) \times \prod_{m=1}^{M} Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 0, j_{m,k}^t)) \times \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=0} \times \Pr(i_k^t = 0)\right.$$

$$+ i_k^t$$

$$\left.\times \prod_{m=1}^{M} Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 1, j_{m,k}^t)) \times \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=1} \times \Pr(i_k^t = 1)\right]$$

$$+ \gamma,$$

(2.13)

where $\boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=0}$ indicates the MLG frequencies in the unaffected population (controls) while $\boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=1}$ indicates the MLG frequencies in the affected population (cases) and they may not equal to $\boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=0}$.

2.2.2 Expectation-maximization algorithm estimates

We provide closed formula solutions of the $(r+1)^{\text{st}}$-step estimates of the parameters necessary to compute the log-likelihoods. The estimates are determined as a function of a vector of genotypes rather than a single genotype, so MLGs are considered all together. This approach is more efficient in comparison to considering only one single genotype at a time.

The posterior probability that individual k has genotype vector $\boldsymbol{G_k^t} = \left(j_{1,k}^t,\cdots,j_{M,k}^t\right)$ is calculated as:

$$
\begin{aligned}
\tau_{\boldsymbol{G_k^t},i_k^t}^{(r)} &= E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k},\boldsymbol{v_k^t},i_k^t)] \\
&= \Pr(\boldsymbol{G_k^t}|\boldsymbol{x_k},\boldsymbol{v_k^t},i_k^t) \\
&= \frac{\Pr(\boldsymbol{G_k^t},\boldsymbol{x_k},\boldsymbol{v_k^t},i_k^t)}{\Pr\left(\boldsymbol{x_k},\boldsymbol{v_k^t},i_k^t\right)} \\
&= \frac{\Pr(\boldsymbol{G_k^t}|i_k^t)\times\Pr(\boldsymbol{x_k}|\boldsymbol{v_k^t},i_k^t,\boldsymbol{G_k^t})\times\Pr(i_k^t)\times\Pr(\boldsymbol{v_k^t})}{\sum_{\boldsymbol{s_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)}\Pr(\boldsymbol{s_k^t}|i_k^t)\times\Pr(\boldsymbol{x_k}|\boldsymbol{v_k^t},i_k^t,\boldsymbol{s_k^t})\times\Pr(i_k^t)\times\Pr\left(\boldsymbol{v_k^t}\right)} \\
&= \frac{\boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t}\times\prod_{m=1}^{M}Bin\left(x_{m,k};v_{m,k}^t;p\left(i_k^t,j_{m,k}^t\right)\right)}{\sum_{\boldsymbol{s_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)}\boldsymbol{g}_{(s_{1,k}^t,\cdots,s_{M,k}^t),i_k^t}\times\prod_{m=1}^{M}Bin\left(x_{m,k};v_{m,k}^t;p\left(i_k^t,s_{m,k}^t\right)\right)}. \qquad (2.14)
\end{aligned}
$$

To compute the $(r+1)^{st}$-step estimates of the genotype frequencies under the alternative hypothesis, for a specific affection status (here we specify the affection status to be unaffected [controls], $i_k^t = 0$), we have,

$$\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \Pr\left(\boldsymbol{G_k^t} = \left(j_{1,k}^t, \cdots, j_{M,k}^t\right) \middle| i_k^t = 0\right) = 1, \tag{2.15}$$

which is,

$$\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \boldsymbol{g}_{\boldsymbol{G_k^t},0}^{(r)} = 1. \tag{2.16}$$

Therefore,

$$\boldsymbol{g}_{(2,2,\cdots,2),0}^{(r)} = 1 - \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,1)} \boldsymbol{g}_{\boldsymbol{G_k^t},0}^{(r)}. \tag{2.17}$$

Then Equation (2.13) can be rewritten as,

$$L_{H_1} = \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,1)} \tau_{\boldsymbol{G_k^t},0}^{(r)}$$

$$\times \ln\left[\prod_{m=1}^{M} Bin\left[\!\left[(x)\right]\!\right]_{m,k}; v_{m,k}^t; p\left(0, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),0}^{(r)} \times \Pr(i_k^t = 0)\right]$$

$$+\sum_{k=1}^{N}\tau_{(2,2,\cdots,2),0}^{(r)}\times\ln\left[\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\boldsymbol{g}_{(2,2,\cdots,2),0}^{(r)}\times\Pr(i_k^t=0)\right]+\gamma$$

$$=\sum_{k=1}^{N}\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,1)}\tau_{\boldsymbol{G_k^t},0}^{(r)}$$

$$\times\ln\left[\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\boldsymbol{g}_{(j_{1,k}^{t},\cdots,j_{M,k}^{t}),0}^{(r)}\times\Pr(i_k^t=0)\right]$$

$$+\sum_{k=1}^{N}\tau_{(2,2,\cdots,2),0}^{(r)}$$

$$\times\ln\left[\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\left(1-\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,1)}\boldsymbol{g}_{\boldsymbol{G_k^t},0}^{(r)}\right)\times\Pr(i_k^t=0)\right]$$

$$+\gamma.$$

(2.18)

Taking the partial derivative of $\boldsymbol{g}_{(0,0,\cdots,0),0}^{(r)}$, we compute:

$$\frac{\partial L_{H_1}}{\partial\boldsymbol{g}_{(0,0,\cdots,0),0}^{(r)}}=\frac{\sum_{k=1}^{N}\tau_{(0,0,\cdots,0),0}^{(r)}\times\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\Pr(i_k^t=0)}{\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\boldsymbol{g}_{(0,0,\cdots,0),0}^{(r)}\times\Pr(i_k^t=0)}$$

$$+\frac{\sum_{k=1}^{N}\tau_{(2,2,\cdots,2),0}^{(r)}\times(-1)\times\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\Pr(i_k^t=0)}{\prod_{m=1}^{M}Bin(x_{m,k};v_{m,k}^{t};p(0,j_{m,k}^{t}))\times\left(1-\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,1)}\boldsymbol{g}_{\boldsymbol{G_k^t},0}^{(r)}\right)\times\Pr(i_k^t=0)}$$

$$=\frac{\sum_{k=1}^{N}\tau_{(0,0,\cdots,0),0}^{(r)}}{\boldsymbol{g}_{(0,0,\cdots,0),0}^{(r)}}-\frac{\sum_{k=1}^{N}\tau_{(2,2,\cdots,2),0}^{(r)}}{\boldsymbol{g}_{(2,2,\cdots,2),0}^{(r)}}.\qquad(2.19)$$

To maximize the function, we set Equation (2.19) as:

$$\frac{\partial L_{H_1}}{\partial\boldsymbol{g}_{(0,0,\cdots,0),0}^{(r)}}=0.\qquad(2.20)$$

Therefore,

$$\boldsymbol{g}^{(r)}_{(0,0,\cdots,0),0} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(0,0,\cdots,0),0}}{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,2),0}} \times \boldsymbol{g}^{(r)}_{(2,2,\cdots,2),0}. \tag{2.21}$$

Similarly,

$$\boldsymbol{g}^{(r)}_{(0,0,\cdots,1),0} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(0,0,\cdots,1),0}}{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,2),0}} \times \boldsymbol{g}^{(r)}_{(2,2,\cdots,2),0},$$

$$\vdots$$

$$\boldsymbol{g}^{(r)}_{(2,2,\cdots,1),0} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,1),0}}{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,2),0}} \times \boldsymbol{g}^{(r)}_{(2,2,\cdots,2),0}. \tag{2.22}$$

Equation (2.16) can then be rewritten as,

$$\boldsymbol{g}^{(r)}_{(2,2,\cdots,2),0} \times \left(\frac{\sum_{k=1}^{N} \sum^{(2,2,\cdots,1)}_{\boldsymbol{G}^{t}_{k}=(0,0,\cdots,0)} \tau^{(r)}_{\boldsymbol{G}^{t}_{k},0}}{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,2),0}} + 1 \right) = 1. \tag{2.23}$$

Thus,

$$\boldsymbol{g}^{(r)}_{(2,2,\cdots,2),0} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,2),0}}{\sum_{k=1}^{N} \sum^{(2,2,\cdots,2)}_{\boldsymbol{G}^{t}_{k}=(0,0,\cdots,0)} \tau^{(r)}_{\boldsymbol{G}^{t}_{k},0}} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(2,2,\cdots,2),0}}{\sum_{k=1}^{N} (1 - i^{t}_{k})}. \tag{2.24}$$

Therefore the $(r+1)^{\text{st}}$-step estimates of the genotype frequencies are:

Under the alternative hypothesis, for the control population (unaffected),

$$\boldsymbol{g}^{(r)}_{(j^t_{1,k},\cdots,j^t_{M,k}),0} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(j^t_{1,k},\cdots,j^t_{M,k}),0}}{\sum_{k=1}^{N}(1-i^t_k)}; \tag{2.25}$$

For the case population (affected),

$$\boldsymbol{g}^{(r)}_{(j^t_{1,k},\cdots,j^t_{M,k}),1} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(j^t_{1,k},\cdots,j^t_{M,k}),1}}{\sum_{k=1}^{N} i^t_k}. \tag{2.26}$$

Under the null hypothesis,

$$\boldsymbol{g}^{(r)}_{(j^t_{1,k},\cdots,j^t_{M,k}),*} = \frac{\sum_{k=1}^{N} \tau^{(r)}_{(j^t_{1,k},\cdots,j^t_{M,k}),*}}{N}. \tag{2.27}$$

For the $(r+1)^{\text{st}}$-step estimates of the sequence error probabilities, we can rewrite Equation (2.13) as the following (here we specify the affection status to be unaffected [controls], $i^t_k = 0$):

$$L_{H_1} = \sum_{k=1}^{N} \sum_{\boldsymbol{G^t_k}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \tau^{(r)}_{\boldsymbol{G^t_k},0} \times \ln\left[(1-i^t_k)\times \prod_{m=1}^{M} Bin\left[\!\left[(x)\right]\!\right]_{m,k}; v^t_{m,k}; p\left(i^t_k=0, j^t_{m,k}\right)\right) \times \boldsymbol{g}_{(j^t_{1,k},\cdots,j^t_{M,k}),i^t_k=0} \times \Pr(i^t_k=0)\right] + \gamma. \tag{2.28}$$

Taking the partial derivative of $\varepsilon_{i_k^t=0}^t$, we compute:

$$\frac{\partial L_{H_1}}{\partial \varepsilon_{i_k^t=0}^t}$$

$$= \frac{\partial}{\partial \varepsilon_0^t}\left(\sum_{k=1}^{N}\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \tau_{\boldsymbol{G_k^t},0}^{(r)} \times \left[\sum_{m=1}^{M} ln\big(Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 0, j_{m,k}^t))\big)\right]\right)$$

$$= \sum_{k=1}^{N}\sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \tau_{\boldsymbol{G_k^t},0}^{(r)} \times$$

$$\left[\sum_{m=1}^{M} \frac{1}{Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 0, j_{m,k}^t))} \times \frac{\partial Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 0, j_{m,k}^t))}{\partial \varepsilon_0^t}\right]. \tag{2.29}$$

When $j_{m,k}^t = 0$, $p\big(i_k^t = 0, j_{m,k}^t = 0\big) = \varepsilon_0^t$,

$$\frac{\partial Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 0, j_{m,k}^t = 0))}{\partial \varepsilon_0^t}$$

$$= \frac{\partial}{\partial \varepsilon_0^t}\left[\binom{v_{m,k}^t}{x_{m,k}} \times {\varepsilon_0^t}^{x_{m,k}} \times (1-\varepsilon_0^t)^{v_{m,k}^t - x_{m,k}}\right]$$

$$= \binom{v_{m,k}^t}{x_{m,k}}\Big[x_{m,k} \times {\varepsilon_0^t}^{x_{m,k}-1} \times (1-\varepsilon_0^t)^{v_{m,k}^t - x_{m,k}} + {\varepsilon_0^t}^{x_{m,k}} \times (-1) \times \big(v_{m,k}^t - x_{m,k}\big)$$

$$\times (1-\varepsilon_0^t)^{v_{m,k}^t - x_{m,k}-1}\Big]$$

$$= \binom{v_{m,k}^t}{x_{m,k}} \times {\varepsilon_0^t}^{x_{m,k}-1} \times (1-\varepsilon_0^t)^{v_{m,k}^t - x_{m,k}-1} \times \big(x_{m,k} - v_{m,k}^t \times \varepsilon_0^t\big). \tag{2.30}$$

Therefore, when $j_{m,k}^t = 0$,

$$\frac{\partial L_{H_1}}{\partial \varepsilon_{i_k^t=0}^t} = \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \tau_{\boldsymbol{G_k^t},0}^{(r)} \times \left[\sum_{m=1}^{M} \frac{x_{m,k} - v_{m,k}^t \times \varepsilon_0^t}{\varepsilon_0^t \times (1-\varepsilon_0^t)} \right]. \tag{2.31}$$

Similarly, when $j_{m,k}^t = 2$, $p\left(i_k^t = 0, j_{m,k}^t = 2\right) = 1 - \varepsilon_0^t$,

$$\frac{\partial Bin(x_{m,k}; v_{m,k}^t; p(i_k^t = 0, j_{m,k}^t = 2))}{\partial \varepsilon_0^t}$$

$$= \frac{\partial}{\partial \varepsilon_0^t} \left[\binom{v_{m,k}^t}{x_{m,k}} \times (1-\varepsilon_0^t)^{x_{m,k}} \times {\varepsilon_0^t}^{v_{m,k}^t - x_{m,k}} \right]$$

$$= \binom{v_{m,k}^t}{x_{m,k}} \times (1-\varepsilon_0^t)^{x_{m,k}-1} \times {\varepsilon_0^t}^{v_{m,k}^t - x_{m,k}-1} \times \left(v_{m,k}^t - x_{m,k} - v_{m,k}^t \times \varepsilon_0^t \right). \tag{2.32}$$

Thus when $j_{m,k}^t = 2$,

$$\frac{\partial L_{H_1}}{\partial \varepsilon_{i_k^t=0}^t} = \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \tau_{\boldsymbol{G_k^t},0}^{(r)} \times \left[\sum_{m=1}^{M} \frac{v_{m,k}^t - x_{m,k} - v_{m,k}^t \times \varepsilon_0^t}{\varepsilon_0^t \times (1-\varepsilon_0^t)} \right]. \tag{2.33}$$

To achieve a maximum log-likelihood, we set Equation (2.33) as:

$$\frac{\partial L_{H_1}}{\partial \varepsilon_{i_k^t=0}^t} = 0.$$

The equation may be rewritten as:

$$\sum_{k=1}^{N} \sum_{\substack{\boldsymbol{G_k^t}=(0,0,\cdots,0) \\ j_{m,k}^t=0}}^{(2,2,\cdots,2)} \tau_{\boldsymbol{G_k^t},0}^{(r)} \times \left[\sum_{m=1}^{M} \frac{x_{m,k} - v_{m,k}^t \times \varepsilon_0^t}{\varepsilon_0^t \times (1-\varepsilon_0^t)} \right]$$

$$+ \sum_{k=1}^{N} \sum_{\substack{\boldsymbol{G_k^t}=(0,0,\cdots,0) \\ j_{m,k}^t=2}}^{(2,2,\cdots,2)} \tau_{\boldsymbol{G_k^t},0}^{(r)} \times \left[\sum_{m=1}^{M} \frac{v_{m,k}^t - x_{m,k} - v_{m,k}^t \times \varepsilon_0^t}{\varepsilon_0^t \times (1-\varepsilon_0^t)} \right] = 0. \tag{2.34}$$

Thus, for the $(r+1)^{\text{st}}$-step estimates of the sequence error probabilities, we have,

$$\varepsilon_{i_k^t=0}^{t,(r+1)} = \frac{\sum_{k=1}^{N} \left(\sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=0} \left(\tau_{\boldsymbol{G_k^t},0}^{(r)} x_{m,k} \right) + \sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=2} \left(\tau_{\boldsymbol{G_k^t},0}^{(r)} \left(v_{m,k}^t - x_{m,k} \right) \right) \right)}{\sum_{k=1}^{N} \left(\sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=0} \left(\tau_{\boldsymbol{G_k^t},0}^{(r)} v_{m,k}^t \right) + \sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=2} \left(\tau_{\boldsymbol{G_k^t},0}^{(r)} v_{m,k}^t \right) \right)}. \tag{2.35}$$

Similarly,

$$\varepsilon_{i_k^t=1}^{t,(r+1)} = \frac{\sum_{k=1}^{N} \left(\sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=0} \left(\tau_{\boldsymbol{G_k^t},1}^{(r)} x_{m,k} \right) + \sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=2} \left(\tau_{\boldsymbol{G_k^t},1}^{(r)} \left(v_{m,k}^t - x_{m,k} \right) \right) \right)}{\sum_{k=1}^{N} \left(\sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=0} \left(\tau_{\boldsymbol{G_k^t},1}^{(r)} v_{m,k}^t \right) + \sum_{m=1}^{M} \sum_{\boldsymbol{G_k^t}, j_{m,k}^t=2} \left(\tau_{\boldsymbol{G_k^t},1}^{(r)} v_{m,k}^t \right) \right)}. \tag{2.36}$$

We note that these probabilities are locus-independent, that is, the subscripts do not contain the individual locus number. However, the formulas indicate that the error probabilities are computed as a composite of the individual locus data values ($v_{m,k}^t$and $x_{m,k}$), suggesting an "average" of all the loci.

2.2.3 Derivation of test statistic

We use the log-likelihood for each hypothesis for a given iteration value r to ultimately determine the maximum log-likelihoods under each scenario (Null and Alternative). We then use these maximums to determine the value of the test statistic. The following notation is used:

$\ln(L_{H_d}) = \sum_{k=1}^{N} \ln[\Pr(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)]$ = Log-likelihood Equation of the observed data.

Note:

$d = 0$ for Null Hypothesis: $(H_0: \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=0} = \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=1})$.

There is no difference in the MLG frequencies between cases and controls.

$d = 1$ for Alternative Hypothesis: $(H_1: \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=0} \neq \boldsymbol{g}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t=1})$.

Therefore, there is a difference in the MLG frequencies between cases and controls.

$\ln(\widehat{L_{H_d}})$ = The maximum log-likelihood of the data for each hypothesis. This maximum is achieved by applying the EM algorithm in the following way (Figure 2.3):

1. Specify a certain number of starting points (randomly generated vector $\vec{\psi}$ of parameter settings for $\boldsymbol{g}^{(r=0)}_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t}$ and $\varepsilon^{t,(r=0)}_{i_k^t}$).

2. For each vector $\vec{\psi}$ in Item 1, update the log-likelihoods under each hypothesis until some specified stopping condition is satisfied, such as:

$$\left|\ln\left(L_{H_d}\right) \text{ of } (r+1)^{\text{st}} \text{ step} - \ln\left(L_{H_d}\right) \text{ of } (r)^{\text{th}} \text{ step}\right| < \delta.$$

In this work, we use $\delta = 0.00001$. The maximum log-likelihood is then the $(r)^{\text{th}}$ step of $\ln\left(L_{H_d}\right)$. This value is denoted by: $\ln\left(L_{H_d}\right)_{r(\vec{\psi})}$.

NOTE: For an arbitrary vector $\vec{\psi}$ in Item 1, if the stopping condition (2) is not met after the maximum number of steps, we define the log-likelihood as: $\ln\left(L_{H_d}\right)_{r_{\max}(\vec{\psi})}$, where $r_{\max}$ is the total number of steps specified for the EM algorithm. For example, in the Simulation section, $r_{\max} = 100$.

3. We define the maximum log-likelihood of the observed data, denoted $\ln\left(\hat{L}_{H_d}\right)$, as:

$$\ln\left(\hat{L}_{H_d}\right) = \max_{\vec{\psi}}\left(\ln\left(L_{H_d}\right)_{r(\vec{\psi})}\right). \tag{2.37}$$

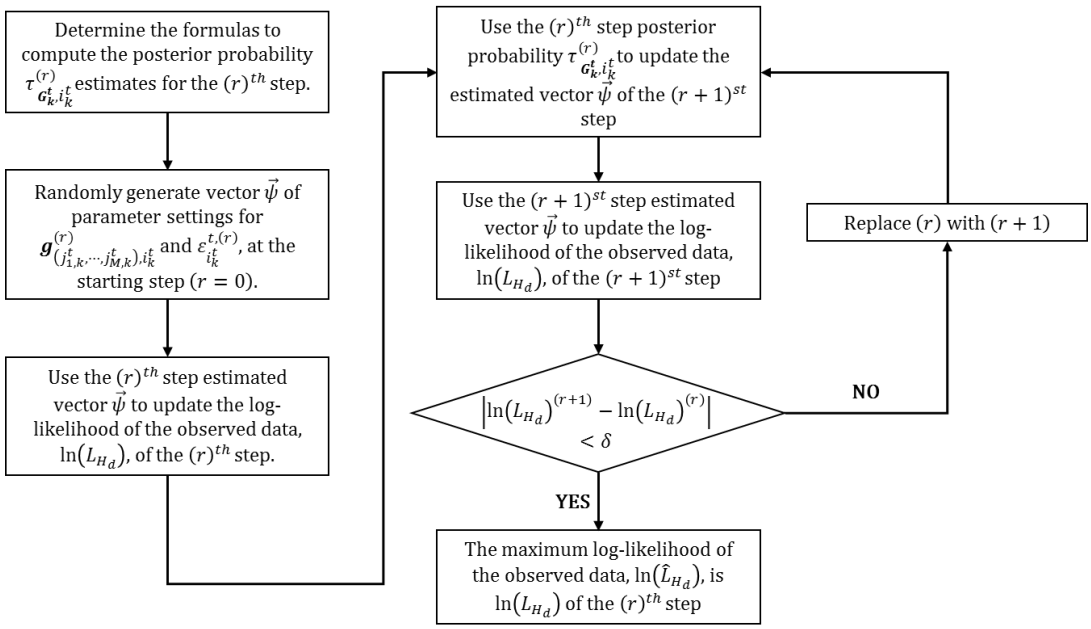
The test statistic is:

$$LRT_{ae,NGS} = 2\left[\ln\left(\hat{L}_{H_1}\right) - \ln\left(\hat{L}_{H_0}\right)\right]. \tag{2.38}$$

As noted above in item (3), the carat symbol $\widehat{\ }$ indicates that we have obtained the maximum log-likelihood of the data under the particular hypothesis. Note that $\boldsymbol{LRT_{ae,NGS}}$ is asymptotically a chi-square distribution with certain degrees of freedom. We consider two versions of the $\boldsymbol{LRT_{ae,NGS}}$ statistic. In this work, we allow for differential

misclassification in the computation of the $\boldsymbol{LRT_{ae,NGS}}$ statistic. Specifically, the two error model parameters are unconstrained (that is, it may be that $\boldsymbol{\varepsilon^{t}_{i^{t}_{k}=0}} \neq \boldsymbol{\varepsilon^{t}_{i^{t}_{k}=1}}$).

Figure 2.3 The workflow of the EM algorithm in obtaining the maximum log-likelihood of the observed data, $\ln(\hat{L}_{H_d})$.



2.3 Simulations of observed data for type I error rates and power evaluations

To evaluate the performance of the test statistic, we compute the empirical type I error rate and power. To do so, we must manipulate the true underlying MLG frequencies, conditional on different affection statuses. We performed simulation studies under different scenarios to evaluate the type I error rate under the null model, and power under the alternative model. In the null model, the MLG frequencies are equal in cases and controls; in the alternative model, the MLG frequencies differ between cases and controls.

From the power simulation, we created a factorial design to determine which factor(s) significantly alter(s) the test statistics. Factorial design is known as two-way design. It

considers all factors with equal interests, as well as the possibility of interaction between these factors. If two factors interact, their effect will not behave in an additive manner. There are two effects of high importance in the factorial design, main effects from single factors, and interaction effects from possible interactions between factors [23].

In the simulation, it is assumed that each locus has only two alleles. First, we randomly generate standardized MLG frequencies. Next, we compute the MLG frequencies conditional on affected status, by applying α (odds), odds-ratio (OR), mode of inheritance (MOI) and standardized MLG frequencies. Observed data is then simulated with the sequencing misclassification model incorporated.

2.3.1 How MLG frequencies are computed during simulation

To compute the conditional MLG frequencies for simulation, the MLG frequencies are conditional on affection status. The formulas provided below document how those MLG frequencies are computed.

Let say the disease MOI is dominant, only one alternative allele is necessary for the individual to be at increased risk of developing the disease. Here, we specify $\beta = \log(OR)$.

Penetrance of affection status i, conditional on MLG $\boldsymbol{j}$, can be calculated as:

$$f_{i,j} = \frac{\left(e^{\alpha + w_j \beta}\right)^i}{1 + e^{\alpha + w_j \beta}}, \tag{2.39}$$

where w_j is the weight corresponding to the MLG, $\boldsymbol{j}$. As aforementioned, a dominant weight parameterization is used. Specifically:

$$w_j = \begin{cases} 0, \boldsymbol{j} = (0,0,\ldots,0) \\ 1, \quad \text{otherwise} \end{cases}.$$

With known or randomly-generated standardized MLG frequencies, $g_{\boldsymbol{j},*}$, the probability of having MLG $\boldsymbol{j}$ and affection status i is,

$$\Pr(i,\boldsymbol{j}) = f_{\boldsymbol{i},\boldsymbol{j}} \times g_{\boldsymbol{j},*}, \tag{2.40}$$

which may be written as,

$$\sum_{\boldsymbol{j}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \Pr(i,\boldsymbol{j}) = \sum_{\boldsymbol{j}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} f_{\boldsymbol{i},\boldsymbol{j}} \times g_{\boldsymbol{j},*} = f_{\boldsymbol{i},*} \times \sum_{\boldsymbol{j}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} g_{\boldsymbol{j},*}\,. \tag{2.41}$$

As defined,

$$\sum_{\boldsymbol{j}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} g_{\boldsymbol{j},*} = 1. \tag{2.42}$$

Therefore, the prevalence of affection status i is,

$$f_{\boldsymbol{i},*} = \sum_{\boldsymbol{j}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} \Pr(i,\boldsymbol{j})\,. \tag{2.43}$$

According to Bayes' Rule, the MLG frequencies conditional on affection status can be calculated as,

$$g_{\boldsymbol{j},i} = \Pr(\boldsymbol{j}|i) = \frac{\Pr(i|\boldsymbol{j})\Pr(\boldsymbol{j})}{\Pr(i)} = \frac{\Pr(i,\boldsymbol{j})}{f_{\boldsymbol{i},*}} = \frac{f_{\boldsymbol{i},\boldsymbol{j}} \times g_{\boldsymbol{j},*}}{\sum_{\boldsymbol{j}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} f_{\boldsymbol{i},\boldsymbol{j}} \times g_{\boldsymbol{j},*}}. \quad (2.44)$$

To provide an example of the formulas used above, standardized MLG frequencies are randomly generated for each two-locus genotype, as shown in columns 1 and 2 in Table 2.2. The table provides two examples of how odds-ratio will affect the MLG frequencies conditional on affection status, using OR=1 and OR=2.

When the odds-ratio equals 1, $\beta = \log(OR) = 0$. The penetrance of affection status, $f_{\boldsymbol{i},\boldsymbol{j}}$, will not be affected by MLG $\boldsymbol{j}$. Consequently, the MLG frequencies are not altered by the affection status (i.e., data is simulated under the null). It should be noted that in Table 2.2 , for OR = 1, the MLG frequencies in cases and controls are equal for each MLG.

When OR does not equal 1, the penetrance of affection status is altered according to the MLG, based on the disease mode of inheritance. As a result, the MLG frequencies vary between different affection statuses, as seen in Table 2.2, under the heading "OR = 2".

Table 2.2 Computation of MLG frequencies conditional on affection status under different odds-ratios

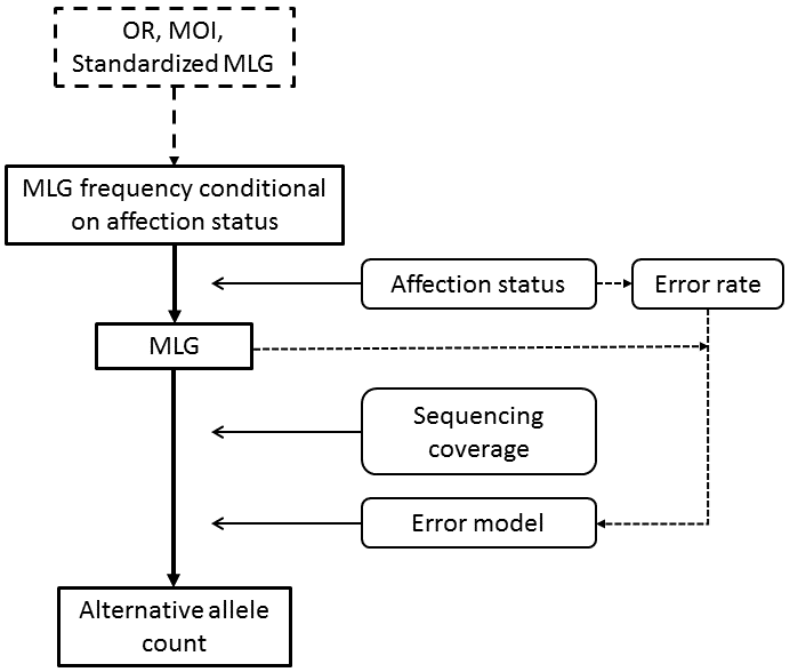
MLG	Standardized MLG Frequency	OR=1				OR=2			
		Controls		Cases		Controls		Cases	
		Penetrance	MLG Frequency	Penetrance	MLG Frequency	Penetrance	MLG Frequency	Penetrance	MLG Frequency
0, 0	0.0963	0.4750	0.0963	0.5250	0.0963	0.4750	0.1121	0.5250	0.0855
0, 1	0.0493	0.4750	0.0493	0.5250	0.0493	0.4011	0.0484	0.5989	0.0498
0, 2	0.0449	0.4750	0.0449	0.5250	0.0449	0.4011	0.0441	0.5989	0.0454
1, 0	0.0961	0.4750	0.0961	0.5250	0.0961	0.4011	0.0944	0.5989	0.0973
1, 1	0.1844	0.4750	0.1844	0.5250	0.1844	0.4011	0.1812	0.5989	0.1867
1, 2	0.1380	0.4750	0.1380	0.5250	0.1380	0.4011	0.1356	0.5989	0.1396
2, 0	0.1870	0.4750	0.1870	0.5250	0.1870	0.4011	0.1837	0.5989	0.1892
2, 1	0.0223	0.4750	0.0223	0.5250	0.0223	0.4011	0.0219	0.5989	0.0226
2, 2	0.1817	0.4750	0.1817	0.5250	0.1817	0.4011	0.1785	0.5989	0.1839

The odds (α) is 0.1 for all the columns.

2.3.2 Determination of data during simulation

Observed data can be simulated in null or alternative models regarding the MLG frequencies, conditional on affection status. In the simulation process, the MLG at a locus will be simulated first, based on its affection status and its frequency, conditional on the affection status. Then the alternative allele read count will be simulated based on the sequencing coverage and the sequencing error model at that locus, which is determined by the sequencing misclassification rate and the MLG. A workflow chart is demonstrated in Figure 2.4, and the formulas used will be shown.

Figure 2.4 Workflow for simulation on alternative allele read count



Here the notation for simulation parameters is specified:

$x_{m,k}$ = observed alternative allele read count for individual k at locus m (number bounded between 0 and sequencing coverage, $v_{m,k}^{t}$).

$v_{m,k}^{t}$= sequencing coverage for individual k at locus m (no misclassification is assumed).

i_k^t = affection status of individual k (no misclassification is assumed) = value indicating whether individual k is affected by the disease ($i_k^t = 1$) or not affected ($i_k^t = 0$).

$j_{m,k}^{t}$ = genotype value for individual k at locus m (no misclassification is assumed) = value indicating genotype with homozygous reference alleles ($j_{m,k}^{t} = 0$), heterozygous reference allele/alternative allele ($j_{m,k}^{t} = 1$), and homozygous alternative alleles ($j_{m,k}^{t} = 2$).

$g_{(j_{1,k}^{t},\cdots,j_{M,k}^{t}),i_{k}^{t}}$= frequency of MLG $\left(j_{1,k}^{t},\cdots,j_{M,k}^{t}\right)$ from locus 1 to locus M conditional on the affection status of individual k, i_{k}^{t}.

$\varepsilon_{i_{k}^{t}}^{t}$ = misclassification rate for individual k of alternative allele and reference allele, which varies with affection status.

2.3.2.1 Determination of an individual's simulated MLG

To simulate the observed alternative read count for each of the individuals, the true underlying MLG, affection status, sequencing coverage and misclassification rates (error rates) are used as simulation parameters. Among these parameters, sequencing coverage and misclassification rates are determined as input values. The affection status value is randomly assigned to be 0 (not affected) or 1 (affected) according to the number of cases and the number of controls in the data set.

The first step in simulating observed alternative read counts for one individual (for example, individual k) is to simulate true underlying MLG from locus 1 to locus M, $\left(j_{1,k}^{t},\cdots,j_{M,k}^{t}\right)$,

from the MLG frequencies, conditional on the affection status of that individual, $g_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t}$.

The MLG frequencies, conditional on a particular affection status sum to 1:

$$\sum_{(j_{1,k}^t,\cdots,j_{M,k}^t)=(0,0,\cdots,0)}^{(2,2,\cdots,2)} g_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t} = 1. \tag{2.45}$$

To obtain a simulated MLG on a given affection status, use the vector of MLG frequencies, $g_{(j_{1,k}^t,\cdots,j_{M,k}^t),i_k^t}$, as the vector of probability weights for random selection. The selected MLG, with the corresponding frequency, is then the simulated MLG on that given affection status. See Table 2.3 for illustration.

Table 2.3 Determination of an individual's simulated MLG

$(j_{1,k}^t, j_{2,k}^t)$	$g_{(j_{1,k}^t,j_{2,k}^t),i_k^t=1}$	Cumulative $g_{(j_{1,k}^t,j_{2,k}^t),i_k^t=1}$	Random Number
0, 0	0.0855	0.0855	
0, 1	0.0498	0.1353	
0, 2	0.0454	0.1808	
1, 0	0.0973	0.2780	

1, 1	0.1867	0.4647	
1, 2	0.1396	0.6043	
2, 0	0.1892	0.7935	
2, 1	0.0226	0.8161	0.7991
2, 2	0.1839	1.0000	

MLG (2, 1) is simulated for individual k who is affected by the disease ($i_k^t = 1$). In this example, two-locus genotype frequencies being affected by the disease, corresponding to particular two-locus genotypes, are listed in column 2, while column 3 lists the cumulative two-locus genotype frequencies. A random number (0.7991) is generated in column 4. This random number is greater than the cumulative two-locus genotype frequency of (2, 0) and less than that of (2, 1). Two-locus genotype (2, 1) is thus selected as the simulated genotype.

2.3.2.2 Determination of an individual's simulated vector of observed data

Observed alternative read counts can be simulated after the previous step using the simulated MLGs. The probability of each possible vector of alternative read counts, $\boldsymbol{x}_{m,k}$, is computed as the binomial probability product from all loci,

$$\prod_{m=1}^{M} Bin(x_{m,k}; v_{m,k}^t; p(i_k^t, j_{m,k}^t)).$$

The binomial distribution $Bin(x_{m,k}; v_{m,k}^t; p(i_k^t, j_{m,k}^t))$ at each locus (for instance, locus m) of individual k has the following properties:

1) The number of repeated trials is equal to the value of sequencing coverage at that locus.
2) The detection of the alternative allele from the sequencing reads is the success outcome, while on the contrary, the detection of the reference allele is the failure outcome. Therefore, the value of alternative read counts, is the number of success trials.
3) The probability of success (error model) is computed by the genotype at the locus, affection status, and the error rate of misclassification, conditional on the affection status:

$$p\left(i_k^t, j_{m,k}^t\right) = \frac{2 - j_{m,k}^t}{2} \varepsilon_{i_k^t}^t + \frac{j_{m,k}^t}{2} \left(1 - \varepsilon_{i_k^t}^t\right). \tag{2.46}$$

The probability of the vector of alternative read counts can then serve as the vector of probability weights for random selection of observed alternative read counts (Table 2.4).

With the simulation process established, we simulate data for our method. The parameter settings that we consider are:

Disease MOI:	Dominant
Number of loci tested:	3
Number of controls:	500, 1000
Number of cases:	500, 1000
Error rate in controls:	0.001, 0.05
Error rate in cases:	0.001, 0.05
Baseline odds-ratio (α):	0.1

OR: 1, 2, 4.

The test results using data simulated from the above parameter settings are reported in Chapter 3.

Table 2.4 Determination of an individual's simulated vector of observed data

$v_{1,k}^t$	$v_{2,k}^t$	$x_{1,k}$	$x_{2,k}$	$Bin(x_{1,k}; v_{1,k}^t; p(i_k^t = 1, j_{1,k}^t = 2))$	$Bin(x_{2,k}; v_{2,k}^t; p(i_k^t = 1, j_{1,k}^t = 1))$	Product	Cumulative	Random Number
3	3	0	0	0.0000	0.1250	0.0000	0.0000	
3	3	1	0	0.0026	0.1250	0.0003	0.0003	
3	3	2	0	0.0847	0.1250	0.0106	0.0109	
3	3	3	0	0.9127	0.1250	0.1141	0.1250	
3	3	0	1	0.0000	0.3750	0.0000	0.1250	
3	3	1	1	0.0026	0.3750	0.0010	0.1260	
3	3	2	1	0.0847	0.3750	0.0318	0.1577	
3	3	3	1	0.9127	0.3750	0.3423	0.5000	
3	3	0	2	0.0000	0.3750	0.0000	0.5000	
3	3	1	2	0.0026	0.3750	0.0010	0.5010	
3	3	2	2	0.0847	0.3750	0.0318	0.5327	
3	3	3	2	0.9127	0.3750	0.3423	0.8750	0.6997
3	3	0	3	0.0000	0.1250	0.0000	0.8750	

3	3	1	3	0.0026	0.1250	0.0003	0.8753	
3	3	2	3	0.0847	0.1250	0.0106	0.8859	
3	3	3	3	0.9127	0.1250	0.1141	1.0000	

Observed alternative read counts (3, 2) are simulated for individual k whose underlying two-locus genotype is (2, 1). This individual is affected by the disease ($i_k^t = 1$). The overall error rate of sequencing misclassification, $\varepsilon_{i_k^t=1}^t$, is 0.03 (not shown in the table). Columns 1 and 2 list the sequencing coverages from two loci, while columns 3 and 4 list all possible alternative read counts based on the sequencing coverages. Columns 5 and 6 show the computation results for binomial probabilities with all the parameters provided at locus 1 and locus 2, separately. Column 7 lists the products of binomial probabilities on both loci, of each possible vector of alternative read counts, while column 8 lists the cumulative products. A random number 0.6997 is generated in column 9. This random number is greater than the cumulative product of (2, 2) and less than that of (3, 2). Observed alternative read counts (3, 2) is thus selected as the simulated value.

Reference:

1. Brooker, R., E. Widmaier, and L. Graham, P. Stiling.(2011). Biology. New York, New York: The McGraw-Hill Companies Inc.
2. Gillespie, J.H., Population genetics: a concise guide. 2010: JHU Press.
3. illumina. Sequencing coverage. Available from: http://www.illumina.com/science/education/sequencing-coverage.html.
4. Sims, D., et al., Sequencing depth and coverage: key considerations in genomic analyses. Nature Reviews Genetics, 2014. 15(2): p. 121-132.
5. Kim, W., et al., Single-variant and multi-variant trend tests for genetic association with next-generation sequencing that are robust to sequencing error. Human Heredity, 2012. 74(3-4): p. 172-183.
6. Hardy, G.H., Mendelian Proportions in a Mixed Population. Science, 1908. 28(706): p. 49-50.
7. Stern, C., The Hardy-Weinberg Law. Science, 1943. 97(2510): p. 137-8.
8. Pearson, K., Davenport's 'Statistical Methods.'. Science, 1904. 20(518): p. 765.
9. Edwards, A.W.F., Likelihood. 1972: Cambridge [Eng.]University Press.
10. Mood, A. and F. Graybill, Introduction to the Theory of Statistics, 2nd edit. 1963. McGraw-Hill, New York.
11. Kendall, M., et al., Kendall's Advanced Theory of Statistics: Volume 2A– Classical Inference and and the Linear Model (Kendall's Library of Statistics). A Hodder Arnold Publication, 1999.
12. Ott, J., Analysis of human genetic linkage. 3rd ed. 1999, Baltimore: Johns Hopkins University Press. xxiii, 382 p.
13. Cox, D.R. and D.V. Hinkley, Theoretical Statistics. 1974: Chapman and Hall.
14. Dempster, A.P., N.M. Laird, and D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm. Journal of the royal statistical society. Series B (methodological), 1977: p. 1-38.
15. Hartley, H., Maximum likelihood estimation from incomplete data. Biometrics, 1958. 14(2): p. 174-194.
16. Wu, C.J., On the convergence properties of the EM algorithm. The Annals of statistics, 1983: p. 95-103.
17. Bayes, T. and R. Price, Philosophical Transactions of the Royal Society of London. Phil Trans R Soc, 1763. 53: p. 370-418.
18. Edwards, W., H. Lindman, and L.J. Savage, Bayesian statistical inference for psychological research. Psychological review, 1963. 70(3): p. 193.
19. Bishop, C.M., Pattern recognition. Machine Learning, 2006. 128.
20. Cornfield, J., A method of estimating comparative rates from clinical data; applications to cancer of the lung, breast, and cervix. J Natl Cancer Inst, 1951. 11(6): p. 1269-75.
21. Mosteller, F., Association and Estimation in Contingency Tables. Journal of the American Statistical Association, 1968. 63(321): p. 1-28.
22. Edwards, A.W.F., The Measure of Association in a 2 × 2 Table. Journal of the Royal Statistical Society. Series A (General), 1963. 126(1): p. 109-114.

23. G. E. P. Box, W.G.H. and J.S. Hunter, Statistics for experimenters : an introduction to design, data analysis, and model building. 1978, United States: Wiley, New York. 228-231.

Chapter 3 Results

In this chapter, we compute empirical type I error and power values for $LRT_{ae,NGS}$, by applying NGS data. Simulation programs developed by the author generate data used by the statistic. For a single run, user input to the program consists of a vector of 10 parameters: number of loci tested, number of cases, number of controls, sequencing coverage, misclassification rate in cases, misclassification rate in controls, odds, odds-ratio, mode of inheritance, and multi-locus genotype frequencies. The empirical type I error rate for a given vector (with odds ratio of 1) is the proportion of p-values among all simulated replicates that are less than a specified significance level. Similarly, the empirical power for a given vector (with odds ratio greater than 1) is the proportion of p-values among all simulated replicated that are less than a given significance level. We use the empirical powers in a factorial design to determine those factors (parameters) that most significantly alter the power of the test. At the end of the chapter, we also evaluate the performance of our method on misclassification rate estimation, and test it against the real data from the 1000 Genomes Project.

3.1 Likelihood ratio test calculations using factorial design

To find out the factors that most substantially affect the likelihood ratio test ($LRT_{ae,NGS}$) statistic when testing the association between multi-locus genotype (MLG) frequencies and the disease affection status, we apply a $2^5 \times 3$ factorial design on a total of 6 design factors (number of cases, number of controls, misclassification rate in cases, misclassification rate in controls, odds-ratio, and multi-locus genotype frequencies). We use the empirical power for each vector of settings in a linear regression. The input values are all main factors, and are all two-way iterations. We obtain coefficients for all input parameters, thereby determining a linear approximation to each empirical power value (see Chapter 2, *Simulations of observed data for type I error and power evaluation*). One factor considered is the frequency of the non-disease MLG, where the MLG consists of all single-locus genotypes that are homozygous for the wild-type allele. This parameter is considered as a setting because our previous work has shown that marker-allele or marker-genotype frequencies could significantly alter power of the test, or it may alter the sample size required to gain the expected power [5-7].

3.1.1 Calculations of empirical type I error rate and empirical power

To compute the empirical values for each parameter vector, for each of the 500 simulation replicates, an observed dataset is first generated based on the functions of the design factors. Random permutation is then applied to the affection statuses of the simulated dataset to generate 500 null permutation replicates.

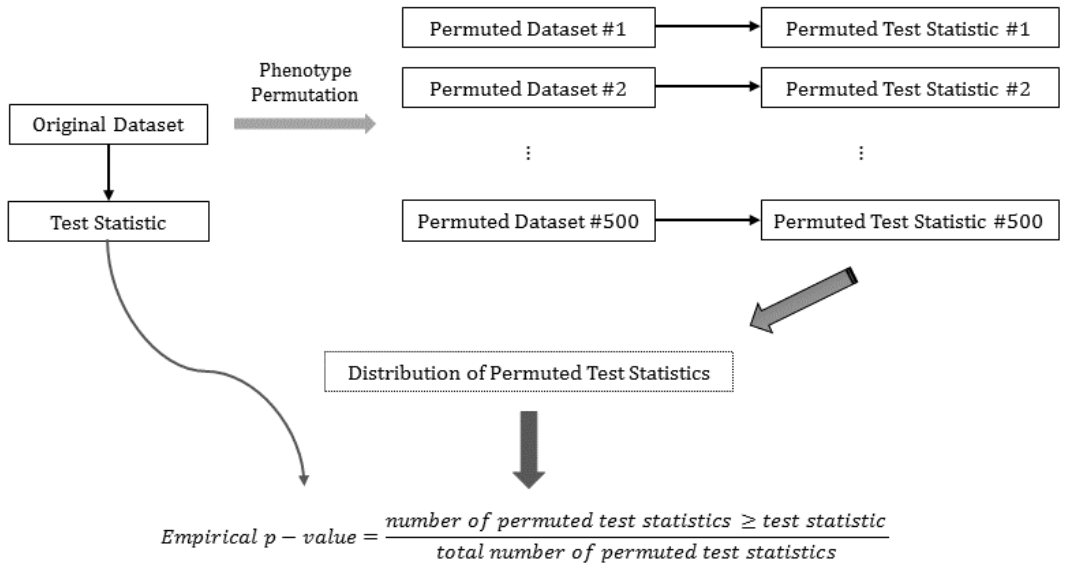
The empirical p-value is calculated by comparing the LRTs from the permuted datasets to the LRT from the original dataset:

$$\hat{p} = \frac{r}{n}.$$

r is the number of permutation replicates that produce an LRT greater than or equal to the calculated LRT for the original dataset. n is the number of replicates permuted (500 in this study). See Figure 3.1 for illustration.

With the empirical p-values generated for all simulation replicates, the empirical type I error rate, or empirical power for a vector at a given significance level, is the proportion of empirical p-values that reject the null hypothesis. The empirical type I error rate corresponds to the null model, whereas the empirical power corresponds to the alternative model.

Figure 3.1 Workflow for empirical p-value calculation



3.1.1.1 Null model – empirical type I error

Under the null hypothesis, all MLG frequencies among case and controls are equal ($MLGfreq_{case} = MLGfreq_{control}$). Therefore, in the null simulations, we set odds-ratio (OR) equal to 1; that is, the odds that an individual will become affected is independent of the MLG frequency [8-10]. The OR is calculated as the following:

$$OR = \frac{\#\ of\ cases(MLGfreq_{case}) \Big/ \#\ of\ controls(MLGfreq_{case})}{\#\ of\ cases(MLGfreq_{control}) \Big/ \#\ of\ controls(MLGfreq_{control})}.$$

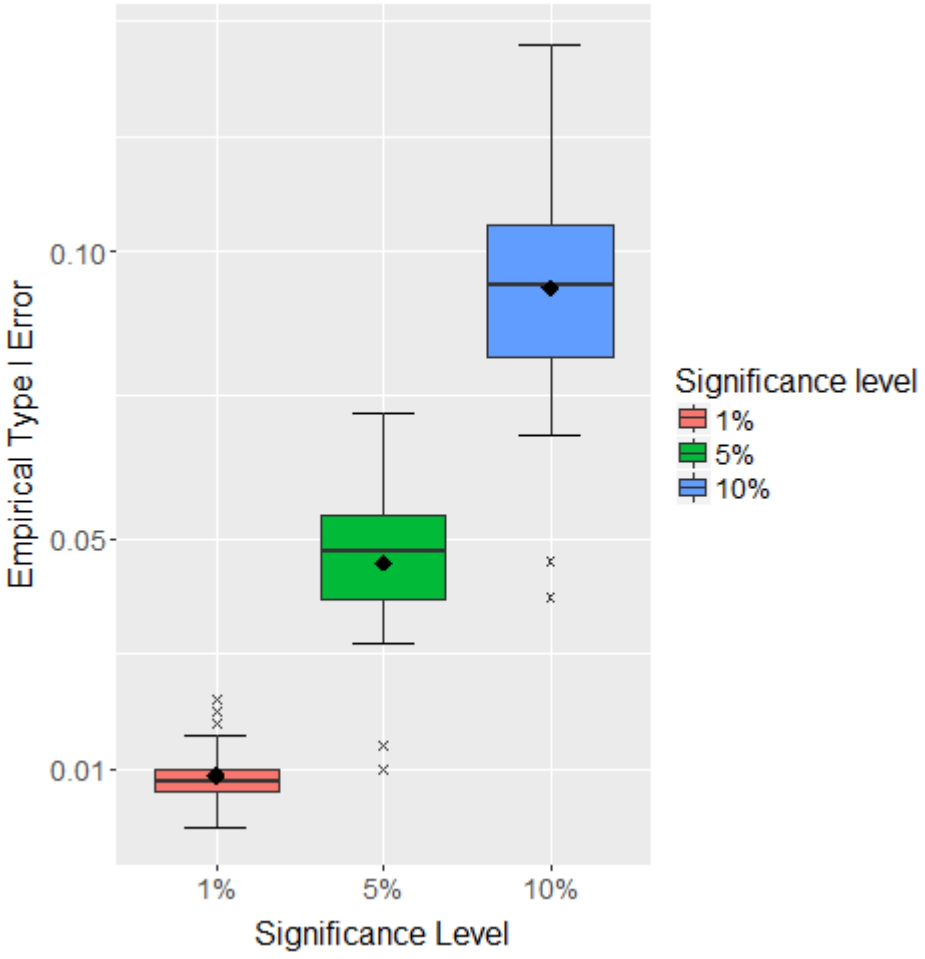
When OR equals 1, it means that the risk of developing the disease with MLG frequencies in cases is the same as that in controls, which describes our null hypothesis. On the other hand, if OR is not equal to 1, the disease risk with MLG frequencies in cases is different from that in controls, and therefore, the simulation model is alternative, rather than being null.

In this work, the non-disease MLG frequency is set as 0.5 or 0.95. Thus, $2^5 \times 3 = 96$ vectors of factor settings are used in the simulations for null and alternative models. For each of the vector settings, 500 simulation replicates are generated to compute the empirical type I error or empirical power.

Figure 3.2 shows boxplots of empirical type I error rates under three significance levels. Most of the empirical type I error rates fall in the corresponding significance levels, with the median and mean close to the values. The medians are 0.008, 0.048 and 0.094 for significance levels of 1%, 5% and 10%, respectively. The means are 0.009, 0.046 and 0.094

for significance levels of 1%, 5% and 10%, respectively. When the significance level is 1%, about 94% of the empirical type I errors are within the 95% confidence interval within the 500 simulations. The percentages for significance levels of 5% and 10% are 91% and 84%, respectively. The parameter settings that are within the 95% confidence intervals are listed in Table 3.1. These confidence intervals were computed in BINOM, a statistical genetics utility programs [11, 12].

Figure 3.2 Boxplots for empirical type I error rates



Legend (values are shown for 1%, 5% and 10%):

♦ (0.009, 0.046, 0.094): mean value of empirical type I error rate; Upper horizontal side of box (0.010, 0.054, 0.105): 3rd quartile ($3Q$) of values; Black horizontal line inside box (0.008, 0.048, 0.094): median value; Lower horizontal side of box (0.006, 0.040, 0.082): 1st quartile ($1Q$) of values; Upper line segment at top of "T"(0.016, 0.072, 0.136): upper whisker, maximum value for set of empirical type I error rates that is lower than or equal to $3Q + 1.5\delta$, $\delta = 3Q - 1Q = \text{Inter} - \text{quartile range}\ (IQR)$; Lower line segment at bottom of inverted "T" (0, 0.032, 0.068): lower whisker, minimum value for set of empirical type I error rates that is higher than or equal to $1Q - 1.5\delta$; ×: outlier.

Table 3.1 The parameter settings and the empirical type I errors that are within the upper and lower whisker range

n_Controls	n_Cases	ε_0^t	ε_1^t	OR	Non-disease MLG Freq	Empirical Type I Error		
						1%	5%	10%
500	500	0.001	0.001	1	0.5	0.008	0.06	0.102
500	500	0.05	0.05	1	0.5	0.002	0.032	0.08
500	500	0.001	0.05	1	0.5	0.01	0.042	0.088
500	500	0.05	0.001	1	0.5	0.002	0.054	0.104
1000	1000	0.001	0.001	1	0.5	0.008	0.048	0.088
1000	1000	0.05	0.05	1	0.5	0.008	0.05	0.104
1000	1000	0.001	0.05	1	0.5	0.006	0.04	0.074
1000	1000	0.05	0.001	1	0.5	0.016	0.046	0.102
500	1000	0.001	0.001	1	0.5	0.008	0.052	0.106
500	1000	0.05	0.05	1	0.5	0.01	0.054	0.11

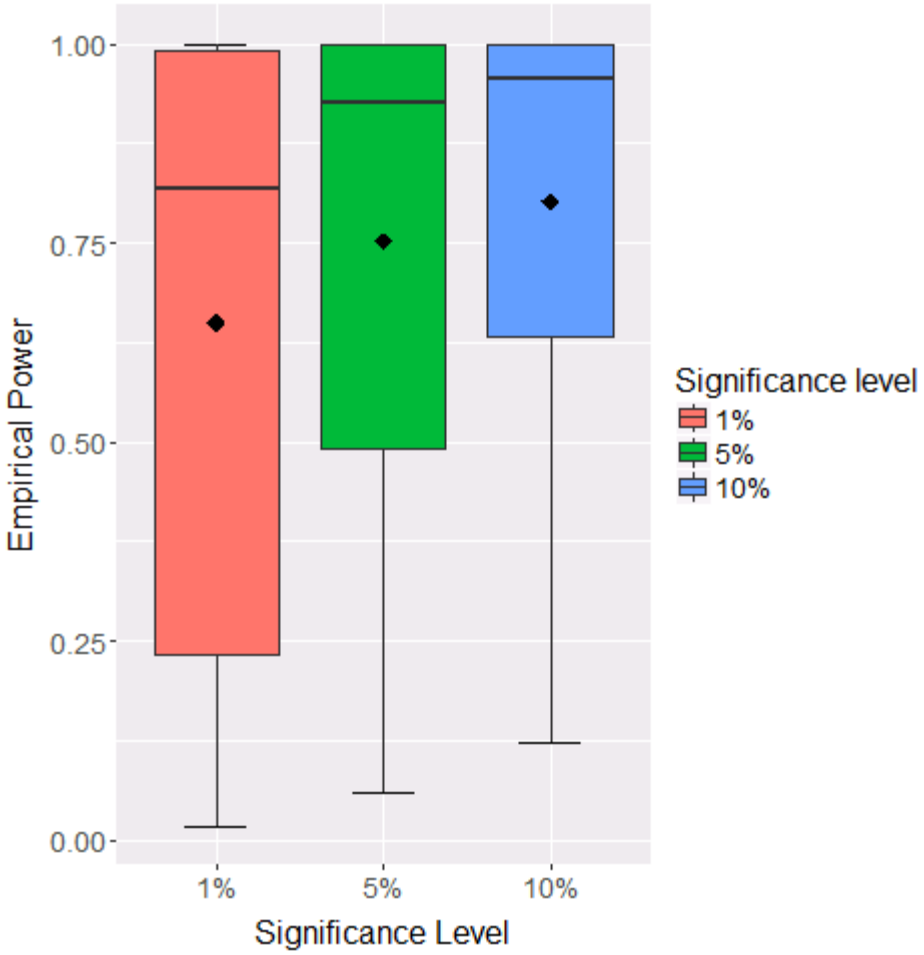
500	1000	0.001	0.05	1	0.5	0.01	0.048	0.096
500	1000	0.05	0.001	1	0.5	0.006	0.036	0.078
1000	500	0.001	0.001	1	0.5	0.002	0.042	0.09
1000	500	0.05	0.05	1	0.5	-	0.068	0.116
1000	500	0.001	0.05	1	0.5	0.008	0.044	0.098
1000	500	0.05	0.001	1	0.5	0.01	0.048	0.096
500	500	0.001	0.001	1	0.95	0.01	0.056	0.106
500	500	0.05	0.05	1	0.95	0.008	0.058	0.096
500	500	0.001	0.05	1	0.95	0.01	0.036	0.084
500	500	0.05	0.001	1	0.95	0.012	0.04	0.08
1000	1000	0.001	0.001	1	0.95	0.01	0.054	-
1000	1000	0.05	0.05	1	0.95	0.002	0.048	0.092
1000	1000	0.001	0.05	1	0.95	0.004	0.05	-
1000	1000	0.05	0.001	1	0.95	0.012	0.038	0.078
500	1000	0.001	0.001	1	0.95	0.008	0.044	0.082
500	1000	0.05	0.05	1	0.95	0.006	0.034	0.092
500	1000	0.001	0.05	1	0.95	0.018	0.06	-
500	1000	0.05	0.001	1	0.95	-	-	-
1000	500	0.001	0.001	1	0.95	0.01	0.036	0.084
1000	500	0.05	0.05	1	0.95	0.014	0.052	0.128
1000	500	0.001	0.05	1	0.95	0.004	-	-
1000	500	0.05	0.001	1	0.95	0.02	-	0.122

Legend**:** n_Controls = number of controls; n_Cases = number of cases; ε_0^t = misclassification rates in controls; ε_1^t = misclassification rates in cases; OR = odds ratio; Non-disease MLG Freq = the frequency of the non-disease MLG.

3.1.1.2 Alternative model – empirical power

In the alternative hypothesis, the MLG frequencies differ between cases and controls. Therefore, in the simulations for the alternative model, we set OR not equal to 1 to measure power of the test. To test in different situations, OR was set to be 2 or 4. See Figure 3.3 for the boxplot of empirical power on different significance levels.

Figure 3.3 Boxplots for empirical power



Legend (values are shown for 1%, 5% and 10%):
♦ (0.650, 0.752, 0.802): mean value of empirical type I error rate; Upper horizontal side of box (0.991, 0.999, 1): 3rd quartile ($3Q$) of values; Black horizontal line inside box (0.817, 0.926, 0.957): median value; Lower horizontal side of box (0.234, 0.493, 0.633): 1st quartile ($1Q$) of values; Upper line segment at top of "T"(1, 1, -): upper whisker, maximum value for set of empirical type I error rates that is lower than or equal to $3Q + 1.5\delta$, $\delta = 3Q - 1Q = \text{Inter} - \text{quartile range}\ (IQR)$; Lower line segment at bottom of inverted "T" (0.018, 0.06, 0.122): lower whisker, minimum value for set of empirical type I error rates that is higher than or equal to $1Q - 1.5\delta$.

The empirical power from all of the parameter settings are within the upper and lower whiskers range (reported in Table 3.2).

Table 3.2 The parameter settings and the empirical power that are within the upper and lower whisker range

n_Controls	n_Cases	ε_0^t	ε_1^t	OR	Non-disease MLG Freq	Empirical Power		
						1%	5%	10%
500	500	0.001	0.001	2	0.5	0.714	0.896	0.944
500	500	0.05	0.05	2	0.5	0.704	0.868	0.938
500	500	0.001	0.05	2	0.5	0.704	0.866	0.922
500	500	0.05	0.001	2	0.5	0.696	0.874	0.932
1000	1000	0.001	0.001	2	0.5	0.988	0.998	1
1000	1000	0.05	0.05	2	0.5	0.986	0.996	0.998
1000	1000	0.001	0.05	2	0.5	0.978	0.994	0.998
1000	1000	0.05	0.001	2	0.5	0.982	0.998	0.998
500	1000	0.001	0.001	2	0.5	0.874	0.966	0.98
500	1000	0.05	0.05	2	0.5	0.886	0.966	0.982
500	1000	0.001	0.05	2	0.5	0.884	0.97	0.984
500	1000	0.05	0.001	2	0.5	0.858	0.958	0.978
1000	500	0.001	0.001	2	0.5	0.888	0.968	0.99
1000	500	0.05	0.05	2	0.5	0.854	0.964	0.988
1000	500	0.001	0.05	2	0.5	0.856	0.962	0.98
1000	500	0.05	0.001	2	0.5	0.898	0.98	0.984
500	500	0.001	0.001	4	0.5	1	1	1
500	500	0.05	0.05	4	0.5	1	1	1

500	500	0.001	0.05	4	0.5	1	1	1
500	500	0.05	0.001	4	0.5	1	1	1
1000	1000	0.001	0.001	4	0.5	1	1	1
1000	1000	0.05	0.05	4	0.5	1	1	1
1000	1000	0.001	0.05	4	0.5	1	1	1
1000	1000	0.05	0.001	4	0.5	1	1	1
500	1000	0.001	0.001	4	0.5	1	1	1
500	1000	0.05	0.05	4	0.5	1	1	1
500	1000	0.001	0.05	4	0.5	1	1	1
500	1000	0.05	0.001	4	0.5	1	1	1
1000	500	0.001	0.001	4	0.5	1	1	1
1000	500	0.05	0.05	4	0.5	1	1	1
1000	500	0.001	0.05	4	0.5	1	1	1
1000	500	0.05	0.001	4	0.5	1	1	1
500	500	0.001	0.001	2	0.95	0.062	0.174	0.274
500	500	0.05	0.05	2	0.95	0.056	0.216	0.3
500	500	0.001	0.05	2	0.95	0.052	0.206	0.326
500	500	0.05	0.001	2	0.95	0.046	0.138	0.198
1000	1000	0.001	0.001	2	0.95	0.192	0.404	0.522
1000	1000	0.05	0.05	2	0.95	0.162	0.362	0.496
1000	1000	0.001	0.05	2	0.95	0.174	0.372	0.5
1000	1000	0.05	0.001	2	0.95	0.096	0.262	0.394
500	1000	0.001	0.001	2	0.95	0.046	0.15	0.268

500	1000	0.05	0.05	2	0.95	0.062	0.19	0.31
500	1000	0.001	0.05	2	0.95	0.084	0.242	0.362
500	1000	0.05	0.001	2	0.95	0.018	0.06	0.122
1000	500	0.001	0.001	2	0.95	0.148	0.326	0.454
1000	500	0.05	0.05	2	0.95	0.17	0.36	0.464
1000	500	0.001	0.05	2	0.95	0.1	0.254	0.354
1000	500	0.05	0.001	2	0.95	0.166	0.338	0.45
500	500	0.001	0.001	4	0.95	0.448	0.686	0.79
500	500	0.05	0.05	4	0.95	0.49	0.696	0.788
500	500	0.001	0.05	4	0.95	0.484	0.708	0.802
500	500	0.05	0.001	4	0.95	0.362	0.58	0.71
1000	1000	0.001	0.001	4	0.95	0.886	0.95	0.978
1000	1000	0.05	0.05	4	0.95	0.886	0.94	0.972
1000	1000	0.001	0.05	4	0.95	0.894	0.948	0.978
1000	1000	0.05	0.001	4	0.95	0.808	0.93	0.95
500	1000	0.001	0.001	4	0.95	0.45	0.73	0.84
500	1000	0.05	0.05	4	0.95	0.528	0.746	0.84
500	1000	0.001	0.05	4	0.95	0.596	0.812	0.89
500	1000	0.05	0.001	4	0.95	0.248	0.522	0.67
1000	500	0.001	0.001	4	0.95	0.826	0.908	0.95
1000	500	0.05	0.05	4	0.95	0.792	0.922	0.964
1000	500	0.001	0.05	4	0.95	0.734	0.86	0.898
1000	500	0.05	0.001	4	0.95	0.774	0.91	0.944

Legend: n_Controls = number of controls; n_Cases = number of cases; ε_0^t = misclassification rates in controls; ε_1^t = misclassification rates in cases; OR = odds ratio; Non-disease MLG Freq = the frequency of the non-disease MLG.

3.1.2 ANOVA for effects on power

Despite remaining high level, the empirical powers are varying between wide ranges, especially in the significance levels of 1% and 5%. Therefore, we utilized factorial designs to determine the factors that alter statistical powers significantly for the significance levels of 1%, 5% and 10%.

Analysis of Variance (ANOVA) was conducted to determine the factors of empirical power for the significance levels of 1%, 5% and 10%. The results are reported in Table 3.3, Table 3.4 and Table 3.5, separately. The factors are sorted based on the F-statistics, from the largest to the least. The value φ^2, the respective factor's proportion of the overall Sum of Squares, was reported. Specifically, $\varphi^2 = \frac{SSQ_{\text{Factor}}}{SSQ_{\text{Total}}}$.

Table 3.3 ANOVA for main effects and all two-way interactions on the significance level of 1%

Factor	**Df**	**SSQ_{Factor}**	**F Statistic**	**φ^2**
Non-disease MLG Freq	1	5.012	815.038	0.601
OR	1	1.83	297.579	0.219

OR× Non-disease MLG Freq	1	0.624	101.554	0.075
n_Controls	1	0.373	60.659	0.045
n_Controls × Non-disease MLG Freq	1	0.111	18.06	0.013
n_Cases	1	0.037	6.042	0.004
n_Controls × OR	1	0.019	3.086	0.002
n_Controls × ε_1^t	1	0.009	1.522	0.001
n_Cases × OR	1	0.009	1.506	0.001
n_Cases × ε_1^t	1	0.008	1.266	0.001
n_Cases × Non-disease MLG Freq	1	0.008	1.252	0.001
ε_1^t× Non-disease MLG Freq	1	0.008	1.369	0.001
ε_1^t	1	0.006	1.047	0.001
ε_0^t× ε_1^t	1	0.006	0.933	0.001
ε_1^t× OR	1	0.005	0.803	0.001
ε_0^t	1	0.004	0.725	0
n_Cases× ε_0^t	1	0.004	0.682	0
ε_0^t× Non-disease MLG Freq	1	0.004	0.61	0
n_Controls × ε_0^t	1	0.002	0.318	0

ε_0^t× OR	1	0.002	0.27	0
n_Controls × n_Cases	1	0	0.031	0
Residuals	42	0.258		
SSQ_{Total}		8.339		

Legend is the same as the legend in Table 3.1.

Table 3.4 ANOVA for main effects and all two-way interactions on the significance level of 5%

Factor	**Df**	$\boldsymbol{SSQ_{\text{Factor}}}$	**F Statistic**	$\boldsymbol{\varphi^2}$
Non-disease MLG Freq	1	3.205	1850.939	0.525
OR	1	1.431	826.435	0.234
OR × Non-disease MLG Freq	1	1.005	580.119	0.164
n_Controls	1	0.212	122.601	0.035
n_Controls × Non-disease MLG Freq	1	0.113	65.491	0.018
n_Cases	1	0.01	5.862	0.002
n_Controls × ε_1^t	1	0.01	5.632	0.002
ε_1^t × Non-disease MLG Freq	1	0.01	6.038	0.002

ε_1^t	1	0.008	4.6	0.001
n_Controls × ε_0^t	1	0.006	3.358	0.001
n_Cases × ε_0^t	1	0.006	3.673	0.001
ε_0^t× ε_1^t	1	0.006	3.673	0.001
ε_0^t	1	0.005	2.973	0.001
ε_0^t× Non-disease MLG Freq	1	0.005	2.651	0.001
n_Cases × ε_1^t	1	0.004	2.31	0.001
n_Cases × OR	1	0.001	0.326	0
n_Controls × n_Cases	1	0	0	0
n_Controls × OR	1	0	0.073	0
n_Cases × Non-disease MLG Freq	1	0	0.153	0
ε_0^t × OR	1	0	0.172	0
ε_1^t× OR	1	0	0.126	0
Residuals	42	0.073		
SSQ_{Total}		6.11		

Legend is the same as the legend in Table 3.1

Table 3.5 ANOVA for main effects and all two-way interactions on the significance level of 10%

Factor	**Df**	**SSQ_{Factor}**	**F Statistic**	**φ^2**
Non-disease MLG Freq	1	2.1897	2227.036	0.472
OR	1	1.1486	1168.255	0.248
OR×Non-disease MLG Freq	1	0.9424	958.441	0.203
n_Controls	1	0.1457	148.221	0.031
n_Controls×Non-disease MLG Freq	1	0.0978	99.482	0.021
n_Controls× ε_1^t	1	0.0091	9.227	0.002
ε_1^t× Non-disease MLG Freq	1	0.0087	8.844	0.002
ε_1^t	1	0.008	8.102	0.002
ε_0^t×ε_1^t	1	0.0078	7.921	0.002
n_Cases	1	0.0069	7.049	0.001
ε_0^t	1	0.0059	5.991	0.001
n_Controls× ε_0^t	1	0.0059	5.991	0.001
ε_0^t× Non-disease MLG Freq	1	0.0059	5.991	0.001

n_Cases×ε_0^t	1	0.005	5.091	0.001
n_Cases× ε_1^t	1	0.004	4.069	0.001
n_Controls×OR	1	0.0031	3.161	0.001
n_cases× Non-disease MLG Freq	1	0.0005	0.55	0
n_Cases×OR	1	0.0002	0.237	0
ε_1^t×OR	1	0.0002	0.207	0
n_Controls× n_Cases	1	0.0001	0.061	0
ε_0^t×OR	1	0	0.023	0
Residuals	42	0.0413		
SSQ_{Total}		4.6368		

Legend is the same as the legend in Table 3.1.

In the tables above (Table 3.3, Table 3.4 and Table 3.5), there are five factors that most substantially affect the power of the association test, based on the F-statistics and the φ^2 values. These factors are (in order of the F-statistic values), frequency of the non-disease MLG, odds-ratio, odds-ratio × frequency of the non-disease MLG, number of controls, number of controls × frequency of the non-disease MLG. All of these factors account for 95.3%, 97.6% and 97.6% of the total Sum of Squares (SSQ_{Total}) in the significance levels of 1%, 5% and 10%, respectively.

Using the results in the above three tables, we selected the four main-effect terms and their two-way interactions (if applicable) to perform a regression analysis. The results are displayed in Table 3.6, Table 3.7 and Table 3.8.

Table 3.6 Linear regression analysis coefficients for the three most significant factors from Table 3.3, and their two-way interaction terms (significance level of 1%)

Variable	Factor	Coefficient Estimate	Standard Error	t-statistic
	(Intercept)	0.81	0.05	17.58
x_1	n_Controls = 1000	0.04	0.05	0.93
x_2	OR = 4	0.12	0.05	2.56
x_3	Non-disease MLG Freq = 0.95	-0.83	0.05	-17.21
x_1x_2	n_Controls = 1000, OR = 4	0.07	0.04	1.76
x_1x_3	n_Controls = 1000, Non-disease MLG Freq = 0.95	0.17	0.04	4.25
x_2x_3	OR = 4, Non-disease MLG Freq = 0.95	0.4	0.04	10.08

Legend is the same as the legend in Table 3.1.

Table 3.7 Linear regression analysis coefficients for the three most significant factors from Table 3.4, and their two-way interaction terms (significance level of 5%)

Variable	Factor	Coefficient Estimate	Standard Error	t-statistic
	(Intercept)	0.93	0.02	37.98
x_1	n_Controls = 1000	0.03	0.03	1.33
x_2	OR = 4	0.05	0.03	2.05
x_3	Non-disease MLG Freq = 0.95	-0.79	0.03	-30.88
x_1x_2	n_Controls = 1000, OR = 4	0.01	0.02	0.27
x_1x_3	n_Controls = 1000, Non-disease MLG Freq = 0.95	0.17	0.02	8.1
x_2x_3	OR = 4, Non-disease MLG Freq = 0.95	0.5	0.02	24.09

Legend is the same as the legend in Table 3.1.

Table 3.8 Linear regression analysis coefficients for the three most significant factors from Table 3.5 and their two-way interaction terms (significance level of 10%)

Variable	Factor	Coefficient Estimate	Standard Error	t-statistic

	(Intercept)	0.96	0.02	52.1
x_1	n_Controls = 1000	0.03	0.02	1.76
x_2	OR = 4	0.05	0.02	2.36
x_3	Non-disease MLG Freq = 0.95	-0.7	0.02	-36.5
x_1x_2	n_Controls = 1000, OR = 4	-0.03	0.02	-1.78
x_1x_3	n_Controls = 1000, Non-disease MLG Freq = 0.95	0.16	0.02	9.97
x_2x_3	OR = 4, Non-disease MLG Freq = 0.95	0.49	0.02	30.96

Legend is the same as the legend in Table 3.1.

From the above Table 3.6, Table 3.7 and Table 3.8, we computed the fitted functions under different significance levels as the following:

$$\widehat{power}_{1\%} = 0.81 + 0.04x_1 + 0.12x_2 - 0.83x_3 + 0.07x_1x_2 + 0.17x_1x_3 + 0.4x_2x_3,$$

$$\widehat{power}_{5\%} = 0.93 + 0.03 + 0.05x_2 - 0.79x_3 + 0.01x_2 + 0.17x_1x_3 + 0.5x_2x_3,$$

$$\widehat{power}_{10\%} = 0.96 + 0.03x_1 + 0.05x_2 - 0.7x_3 - 0.03x_1x_2 + 0.16x_1x_3 + 0.49x_2x_3,$$

where,

$$x_1 = \begin{cases} 1, \text{n\_Controls } = 1000 \\ 0, \text{n\_Controls } = 500 \end{cases},$$

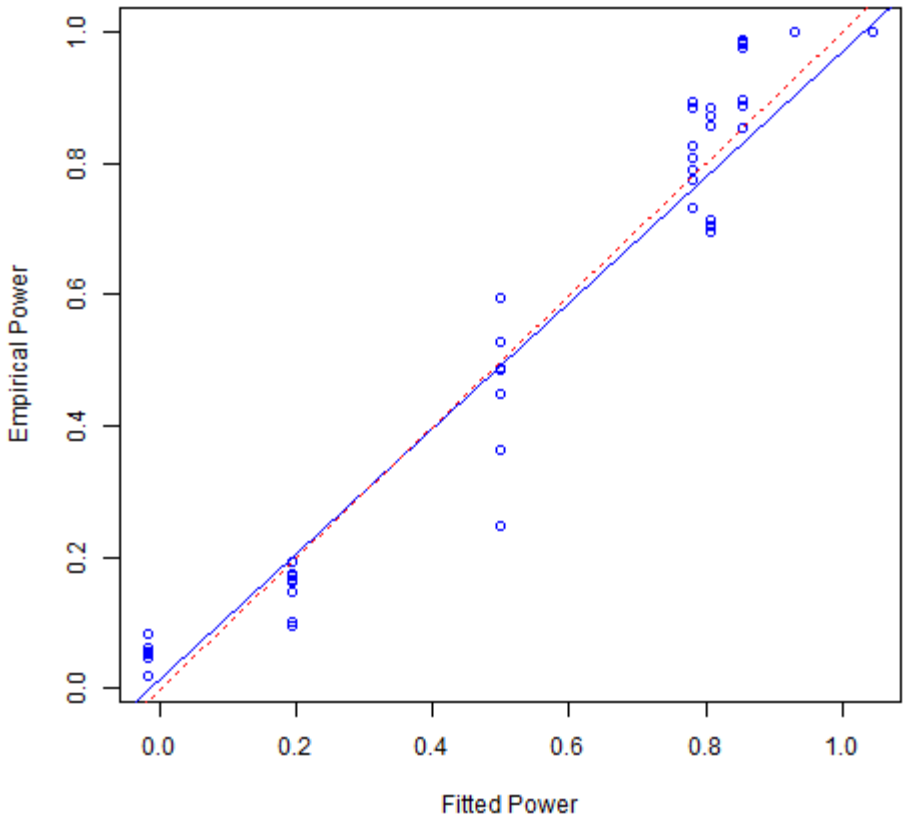
$$x_2 = \begin{cases} 1, \text{OR} = 4 \\ 0, \text{OR} = 2 \end{cases},$$

$$x_3 = \begin{cases} 1, \text{Non} - \text{disease MLG Freq } = 0.95 \\ 0, \text{Non} - \text{disease MLG Freq } = 0.5 \end{cases}.$$

Upon examining the above equations, we note that two factors play a significant role in altering the power of the test. Taking the significance level of 1% for example: Increasing the non-disease MLG frequency from 0.5 to 0.95 produces a substantial decrease in power of 0.83(coefficient for variable x_3), while increasing it from 0.5 to 0.95 and jointly increasing the odds-ratio from 2 to 4 produces a power increase of approximately 0.4 (coefficient for variable x_2x_3).

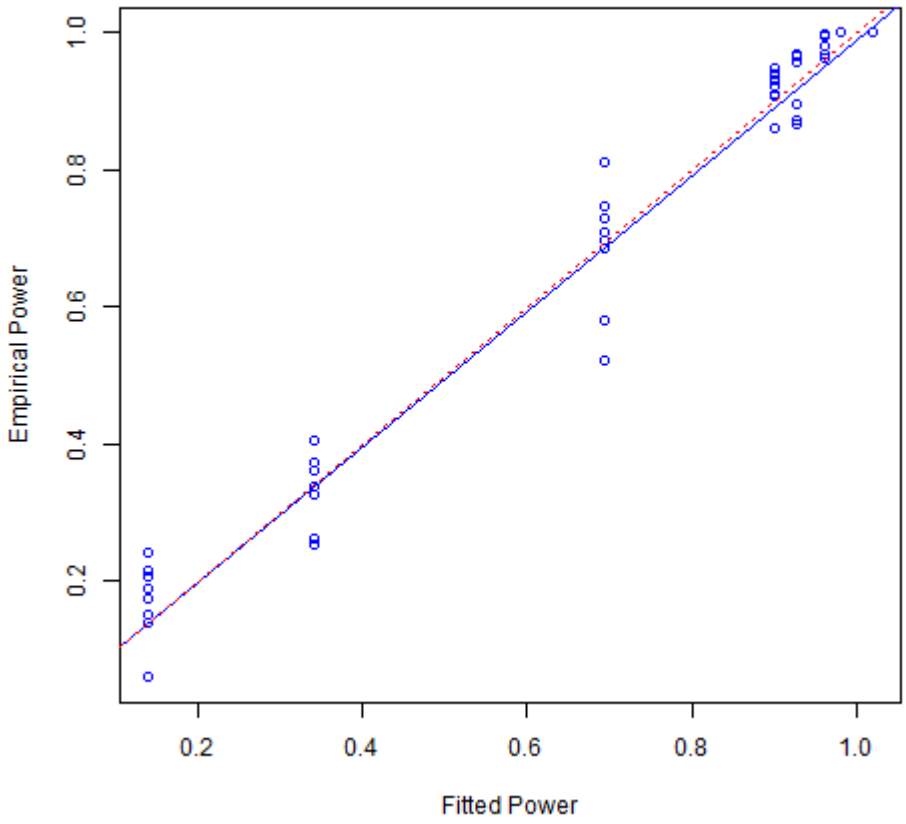
In Figure 3.4, Figure 3.5 and Figure 3.6, we plot the fitted values (using the above equations) versus the empirical powers. The coefficients of the trend line, computed using a generalized linear model in R, are consistent with the finding that the empirical powers are accurately represented by a linear combination of the three variables (x_1, x_2 and x_3) and their two-way interactions. We may conclude that for the parameter settings, only three of the six factors are needed to approximate the empirical power. These factors are the number of controls, odds-ratio and the non-disease MLG frequency. Moreover, apart from these three factors, values of the misclassification rate (in cases or in controls) do not affect the power significantly.

Figure 3.4 Scatter plot of empirical power versus fitted power using 64 vectors of factor settings (significance level: 1%)



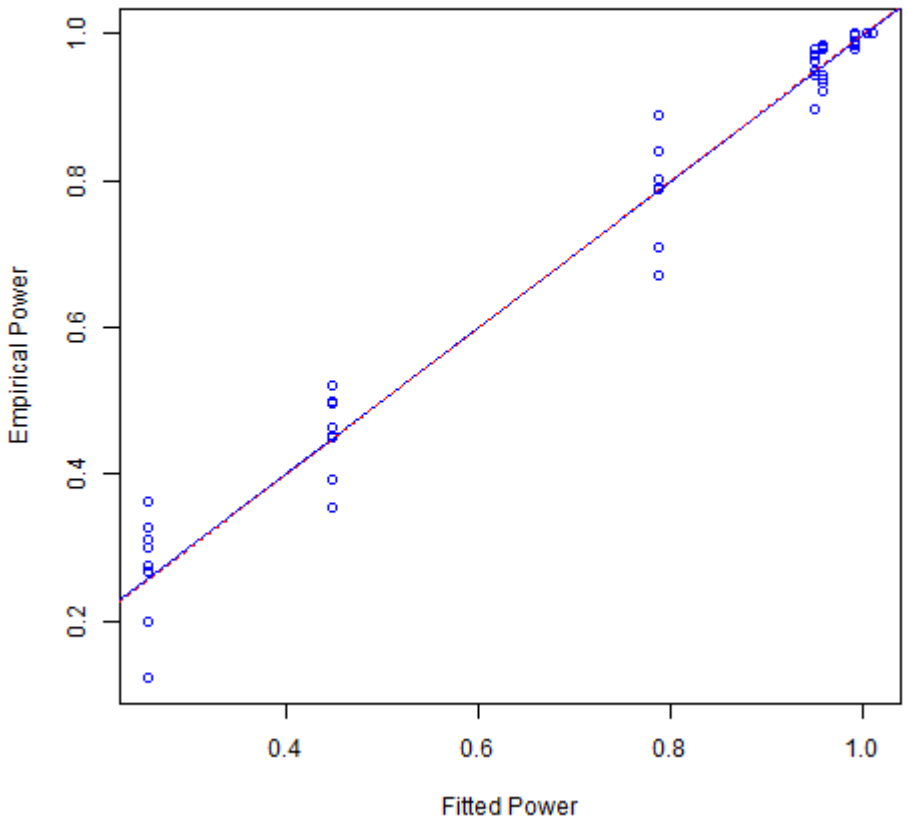
The trend line intercept is 0.0167 and the slope is 0.954 (the multiple R-squared value is 0.954). Blue dots: Data points. Blue Line: Fitted trend line of the data points. Red dotted line: It's slope equals 1.

Figure 3.5 Scatter plot of empirical power versus fitted power using 64 vectors of factor settings (significance level: 5%)



The trend line intercept is 0.0196 and the slope is 0.989 (the multiple R-squared value is 0.976). Blue dots: Data points. Blue Line: Fitted trend line of the data points. Red dotted line: It's slope equals 1.

Figure 3.6 Scatter plot of empirical power versus fitted power using 64 vectors of factor settings (significance level: 10%)



The trend line intercept is 0.0016 and the slope is 0.996 (the multiple R-squared value is 0.976). Blue dots: Data points. Blue Line: Fitted trend line of the data points. Red dotted line: It's slope equals 1.

3.2 Performance evaluation on misclassification estimates

3.2.1 Testing on Simulated Data

Since the true misclassification rates cannot be observed directly to determine the correctness from our method's misclassification estimation, we simulate NGS data with known underlying misclassification rates, which are differential according to affection status. The data simulation is processed through the simulation computer program we developed.

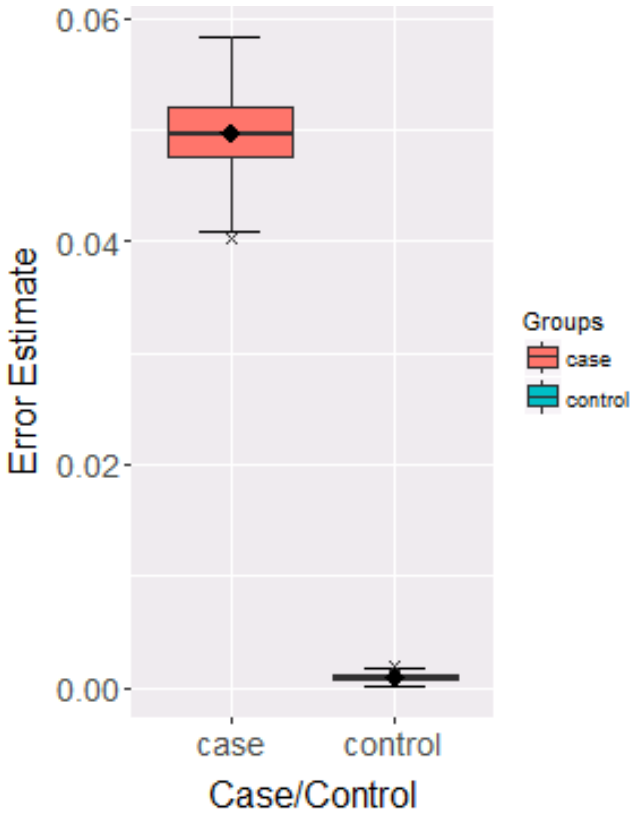
We simulated a dataset of 1000 controls and 500 cases. The simulated dataset was generated by a fixed sequencing coverage of 4 on each tested locus and other known parameters (see simulation notation in Chapter 2):

Disease MOI:	Dominant
Number of loci tested:	3
Number of controls:	1000
Number of cases:	500
Error rate in controls:	0.001
Error rate in cases:	0.05
α:	0.1
OR:	1
Frequency of non-disease MLG	0.95.

Note that the misclassification rates are 0.05 in cases and 0.001 in controls. This dataset is simulated under the constrained (null) model, of which the MLG frequencies are equal between cases and controls. The simulated dataset is tested using our method for misclassification rate estimation.

We generated 500 replicates of the simulation and estimation process, and the results are shown in a boxplot in Figure 3.7. The mean estimates for misclassification rate are 0.05 in cases, and 0.001 in controls (the medians are 0.05 and 0.001 in cases and in controls, respectively). The boxplot indicates that our method is able to correctly estimate the true underlying differential misclassification rates of the observed data.

Figure 3.7 Boxplot of misclassification estimates from simulated data



Legend:
♦ : mean value of empirical type I error rate; Upper horizontal side of box: 3rd quartile ($3Q$) of values; Black horizontal line inside box: median value; Lower horizontal side of box: 1st quartile ($1Q$) of values; Upper line segment at top of "T: upper whisker, maximum value for set of empirical type I error rates that is lower than or equal to $3Q + 1.5\delta$, $\delta = 3Q - 1Q = \text{Inter} - \text{quartile range}\ (IQR)$; Lower line segment at bottom of inverted "T": lower whisker, minimum value for set of empirical type I error rates that is higher than or equal to $1Q - 1.5\delta$; ×: outlier.

3.2.2 Testing on real data: the 1000 Genomes Project data

In order to test the performance of our method on a real-world situation, we tested it on the real data that we extracted from the 1000 Genomes Project [1-4, 13].

First, we downloaded the available exome sequencing data in BAM (Binary Sequence Alignment/Map) [14] format on chromosome 20 from 2,504 individuals in the 1000 Genomes Project Phase 3 archive (ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/phase3/data/). To make our process computationally efficient, we only kept the exome sequenced regions on chromosome 20, from base pair position 60897487 to position 60908969 for each individual [15]. After sorting and indexing the data on the selected region, we used the option "mpileup" in Samtools (version 1.3.1) [14, 16] and the option "call -m" in Bcftools (version 1.3.1) [16, 17] for variant calling and converted the data into VCF (Variant Call Format) [18]. With the variants called for every individual, we filtered out variants with QUAL (quality) lower than 100. QUAL is the Phred-scaled probability indicating the existence of a variant [18].

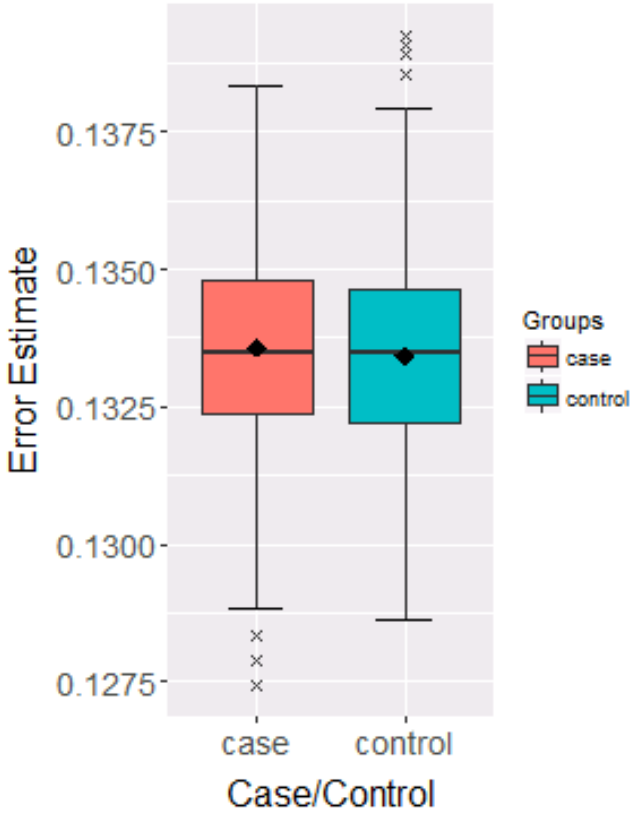
To make the data set comparable to the imputed data sets, we selected three genetic positions (loci) from the extracted region: 60907675, 60908964 and 60908969. Among all

2,504 individuals, we kept those individuals who are heterozygotes or alternative allele homozygotes at all three of these loci. 1,314 individuals are kept in the end. Their non-reference allele counts (sum of forward non-reference and reverse non-reference alleles, labeled "DP4" in VCF format) and raw sequencing coverages (labeled "DP" in VCF format) on each of the three loci are extracted using a computer program developed by the author. In the extracted region, the non-reference allele counts are in a range of 4-84, with a mean of 20.8 and a median of 18; the raw sequencing coverages are in a range of 5-158, with a mean of 28.4 and a median of 26.

On each of these individuals, the affection status is assigned randomly with 657 affected (cases) and 657 unaffected (controls). The preprocessed dataset is tested using our method for misclassification rate estimation.

The affection assigning step in the process is repeated 500 times to generate permutation replicates and the results are shown in a boxplot in Figure 3.8. The mean misclassification estimates are 0.134 in cases, and 0.133 in controls (medians are 0.133 and 0.133 in cases and in controls, respectively). The estimated misclassification rates are inflated considerably from the previous published error rates in the NGS platforms: Illumina HiSeq (0.34%), Ion Torrent PGM (1.9%) and Complete Genomics (2.4%) [19].

Figure 3.8 Boxplot of misclassification estimates from 1000 Genomes Project data



Legend:
♦ : mean value of empirical type I error rate; Upper horizontal side of box: 3rd quartile ($3Q$) of values; Black horizontal line inside box: median value; Lower horizontal side of box: 1st quartile ($1Q$) of values; Upper line segment at top of "T: upper whisker, maximum value for set of empirical type I error rates that is lower than or equal to $3Q + 1.5\delta$, $\delta = 3Q - 1Q = \text{Inter} - \text{quartile range}\ (IQR)$; Lower line segment at bottom of inverted "T": lower whisker, minimum value for set of empirical type I error rates that is higher than or equal to $1Q - 1.5\delta$; ×: outlier.

3.2.3 Testing on simulated data with high misclassification rates:

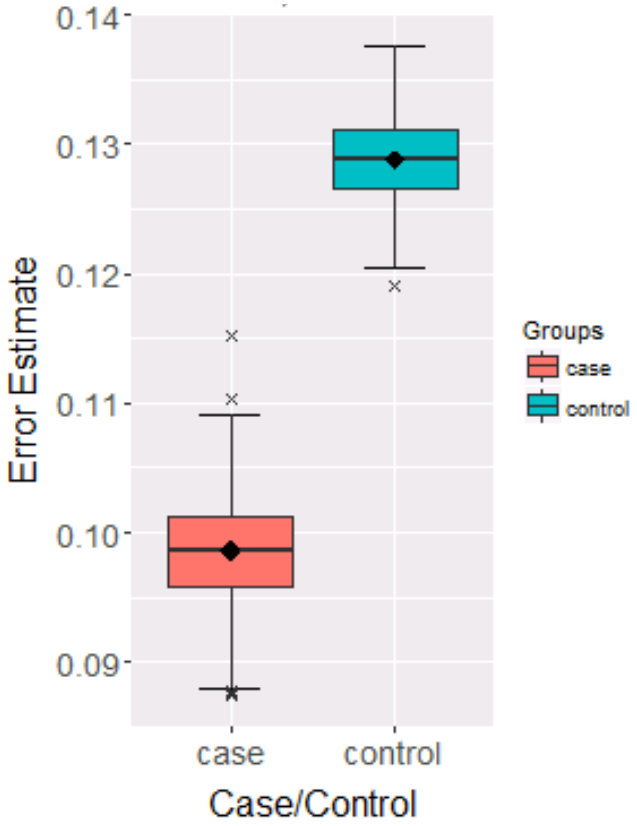
The surprisingly inflated estimates of misclassification rates in the real-world data raised some concerns. Is this inflation possibly caused by the fact that our method is not able to handle higher differential misclassification rates? Or, does our method only work on

simulated data, instead of real NGS-processed data? To clarify the answer, we further tested the performance of our method, on higher misclassification rates.

We simulated a new dataset, generated by the same known parameters and under the same constrained model as in *Section 3.2.1. Testing on Simulated Data*, except for that the misclassification rates are set to 0.1 in cases and 0.13 in controls. The newly simulated dataset is then tested with our method under the same process with 500 replicates.

The results are shown in a boxplot in Figure 3.9. The mean estimates for misclassification rate are 0.099 in cases, and 0.129 in controls (the medians are 0.099 and 0.129 in cases and in controls, respectively). The boxplot indicates that our method is able to correctly estimate the higher true underlying differential misclassification rates of the dataset.

Figure 3.9 Boxplot of misclassification estimates from simulated data



Legend:
♦ : mean value of empirical type I error rate; Upper horizontal side of box: 3rd quartile ($3Q$) of values; Black horizontal line inside box: median value; Lower horizontal side of box: 1st quartile ($1Q$) of values; Upper line segment at top of "T: upper whisker, maximum value for set of empirical type I error rates that is lower than or equal to $3Q + 1.5\delta$, $\delta = 3Q - 1Q = \text{Inter} - \text{quartile range}$ (IQR); Lower line segment at bottom of inverted "T": lower whisker, minimum value for set of empirical type I error rates that is higher than or equal to $1Q - 1.5\delta$; ×: outlier.

3.2.4 Testing on real data of high quality

In the boxplot in Figure 3.7, we see that our method is able to correctly estimate the higher differential misclassification rates from the simulated data. This raises another question: Why is our method not working with the real data extracted from the 1000 Genomes Project?

Is this problem caused by the quality of the data? Thus, we attempted testing our method again in estimating the misclassification rates in the 1000 Genomes Project data, by replacing raw sequencing coverage with sequencing coverage from high quality bases. The raw sequencing coverage (labeled "DP" in VCF format) is the number of detected sequencing reads that are covering a position of interest, while sequencing coverage with higher quality is the value of the sum of high quality bases at a position, including forward reference alleles, reverse reference alleles, forward non-reference alleles and reverse non-reference alleles (labeled "DP4" in VCF format). This high-quality sequencing coverage excludes the count of low-quality bases so it is equal to, or lower than, the raw sequencing coverage. The low quality on bases might be caused by bases being misaligned to the position [13].

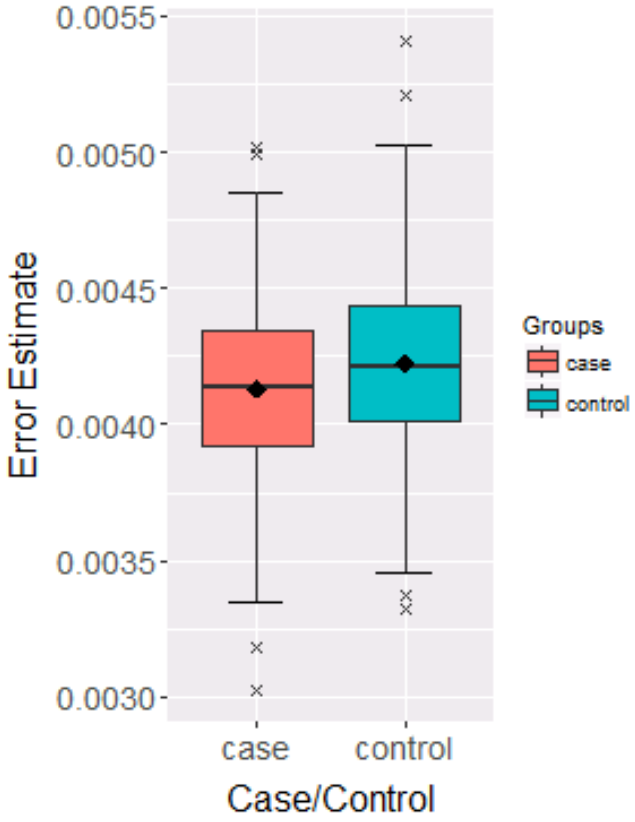
To make a dataset comparable to the previously tested 1000 Genomes Project dataset, we use the same pipeline for variant calling, and selected the same three loci from chromosome 20 on positions 60907675, 60908964, and 60908969. The same 1314 individuals are kept. We extract their non-reference allele counts (labeled "DP4" in VCF format) and compute high-quality sequencing coverages (as the sum of high-quality base counts from the 4 values labeled in "DP4" in VCF format) on each of the three loci, using another computer program developed by the author. In the extracted region, the high-quality sequencing coverages are in a range of 4-134, with a mean of 24.8 and a median of 23.

The affection status is once again assigned randomly with half of the individuals being affected and the other half being unaffected (657 in cases and 657 in controls). The newly preprocessed dataset is tested using our method for misclassification rate estimation.

The affection assigning is repeated 500 times and the results are shown in a boxplot in Figure 3.10. The mean misclassification estimates are 0.004 in cases, and 0.004 in controls (the medians are 0.004 and 0.004 in cases and in controls, respectively). The boxplot indicates that the misclassification rates from the 1000 Genomes Project data with high-quality sequencing coverage is around 0.004, which is much lower than the previous estimates on the same set of individuals and genetic positions. These misclassification rates also match the published rate range of 0.4-3% [19].

Given the estimation performance on the simulated misclassification and the actual data, it is reasonable to conclude that our method is able to estimate the underlying misclassification of the sequencing data.

Figure 3.10 Boxplot of misclassification estimates from 1000 Genomes Project data with sequencing coverage from high quality bases



Legend:
♦ : mean value of empirical type I error rate; Upper horizontal side of box: 3rd quartile ($3Q$) of values; Black horizontal line inside box: median value; Lower horizontal side of box: 1st quartile ($1Q$) of values; Upper line segment at top of "T: upper whisker, maximum value for set of empirical type I error rates that is lower than or equal to $3Q + 1.5\delta$, $\delta = 3Q - 1Q = \text{Inter} - \text{quartile range}$ (IQR); Lower line segment at bottom of inverted "T": lower whisker, minimum value for set of empirical type I error rates that is higher than or equal to $1Q - 1.5\delta$; ×: outlier.

Reference:

1. Genomes Project, C., et al., A map of human genome variation from population-scale sequencing. Nature, 2010. 467(7319): p. 1061-73.
2. Genomes Project, C., et al., An integrated map of genetic variation from 1,092 human genomes. Nature, 2012. 491(7422): p. 56-65.
3. Genomes Project, C., et al., A global reference for human genetic variation. Nature, 2015. 526(7571): p. 68-74.
4. Sudmant, P.H., et al., An integrated map of structural variation in 2,504 human genomes. Nature, 2015. 526(7571): p. 75-81.
5. Gordon, D., et al., Increasing power for tests of genetic association in the presence of phenotype and/or genotype error by use of double-sampling. Statistical Applications in Genetics and Molecular Biology, 2004. 3: p. Article26.
6. Kang, S.J., D. Gordon, and S.J. Finch, What SNP genotyping errors are most costly for genetic association studies? Genet Epidemiol, 2004. 26(2): p. 132-41.
7. Ahn, K., et al., The effects of SNP genotyping errors on the power of the Cochran-Armitage linear trend test for case/control association studies. Ann Hum Genet, 2007. 71(Pt 2): p. 249-61.
8. Edwards, A.W.F., The Measure of Association in a 2 × 2 Table. Journal of the Royal Statistical Society. Series A (General), 1963. 126(1): p. 109-114.
9. Mosteller, F., Association and Estimation in Contingency Tables. Journal of the American Statistical Association, 1968. 63(321): p. 1-28.
10. Cornfield, J., A method of estimating comparative rates from clinical data; applications to cancer of the lung, breast, and cervix. J Natl Cancer Inst, 1951. 11(6): p. 1269-75.
11. Ott, J., Analysis of human genetic linkage. 3rd ed. 1999, Baltimore: Johns Hopkins University Press. xxiii, 382 p.
12. Ott, J. Statistical Genetics Utility programs. Available from: http://www.jurgott.org/linkage/util.htm.
13. Calling SNPs/INDELs with SAMtools/BCFtools. 2010.
14. Li, H., et al., The Sequence Alignment/Map format and SAMtools. Bioinformatics, 2009. 25(16): p. 2078-9.
15. Kim, W., et al., Single-variant and multi-variant trend tests for genetic association with next-generation sequencing that are robust to sequencing error. Human Heredity, 2012. 74(3-4): p. 172-183.
16. Li, H., A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. Bioinformatics, 2011. 27(21): p. 2987-93.
17. Danecek P., S.S., Durbin R. Multiallelic calling model in bcftools (-m) 2016; Available from: http://samtools.github.io/bcftools/call-m.pdf.
18. The Variant Call Format (VCF) Version 4.2 Specification. 2015; Available from: https://samtools.github.io/hts-specs/VCFv4.2.pdf.
19. Ross, M.G., et al., Characterizing and measuring bias in sequence data. Genome Biol, 2013. 14(5): p. R51.

Chapter 4 Discussion

Here, we discuss some possible improvements for our method. One improvement is the extension to allow for locus-specific sequence error. Another improvement may be speeding up the computer program, so that our computer program may perform association tests on a greater number of genetic loci jointly. The current computer time under various conditions (such as the number of loci or sequencing coverage) is discussed in this chapter. We also discuss the potential to enhance the statistical power of our method by applying a double-sampling approach to a subset of sequenced individuals, meaning re-sequencing a small set of samples through another sequencing technology. In the end, we discuss the advancements in high-throughput technologies, that may give inspiration to readers for research in a similar field.

4.1 Summary

The method developed in this work is a likelihood-ratio approach ($LRT_{ae,NGS}$), designed to detect the association between genetic variants and genetic disorders using NGS data. We extend this approach to multiple genetic loci, which allows users to test all their genetic loci of interest at once. In our simulation results, our method maintains correct type I error rates for the null hypothesis, and has both a wide range and high level of powers for the alternative hypothesis. By applying factorial designs, we detect three factors altering test power significantly, including the number of controls, odds ratio and the most common, multi-locus genotype frequency. By using the expectation-maximization algorithm, we compute our test statistic and estimate differential misclassification rates from the observed data. By comparing the misclassification rate estimates to their true values from simulation studies, our method shows its robustness and accuracy in estimating differential misclassification rates.

4.2 Locus-specific misclassification rates

In our method, the misclassification rate is dependent on the affection status. In other words, our method allows for the possibility that misclassification probabilities are different between cases and controls. For this version of our statistical method, we specify that the misclassification probabilities (case or control) remain constant across all tested loci. However, this model may not hold for actual data. A more robust model is one that allows

for locus-specific error rates in cases and controls. In terms of notation, we extend our current notation to be: $\varepsilon^{t}_{m,i^{t}_{k}}$ (notation for other parameters, see Chapter 2).

Therefore, for the k^{th} individual, the probability of observing alternative allele counts for a total of M loci, conditional on sequencing coverage, affection status and the true underlying genotype (Chapter 2, Equation 2.10), can be written as:

$$\Pr\left(\boldsymbol{x_k}\middle|\boldsymbol{v_k^t} = \left(v^t_{1,k},\cdots,v^t_{M,k}\right), i^t_k, \boldsymbol{G_k^t} = \left(j^t_{1,k},\cdots,j^t_{M,k}\right)\right) = \prod_{m=1}^{M} Bin\left(x_{m,k}; v^t_{m,k}; p\left(m, i^t_k, j^t_{m,k}\right)\right). \tag{4.1}$$

In the above binomial probability mass function, $Bin\left(x_{m,k}; v^t_{m,k}; p\left(m, i^t_k, j^t_{m,k}\right)\right)$, the probability of a success in observing an alternative allele instead of a reference allele is $p\left(m, i^t_k, j^t_{m,k}\right) = \left(\frac{2-j^t_{m,k}}{2}\varepsilon^{t}_{m,i^t_k} + \frac{j^t_{m,k}}{2}\left(1-\varepsilon^{t}_{m,i^t_k}\right)\right)$. As in our original statistic, here, the error model is specified to be symmetric.

The log-likelihood of the observed data $(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i^t_k)$ for N individuals over M loci, under the null hypothesis (Chapter 2, Equation 2.12) may be rewritten as:

$$
\begin{aligned}
&L_{H_0} \\
&= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)] \\
&\times ln\Bigg((1 - i_k^t) \times \prod_{m=1}^{M} Bin\left(x_{m,k}; v_{m,k}^t; p\left(m, i_k^t = 0, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t, \cdots, j_{M,k}^t),*} \times \Pr(i_k^t = 0) \\
&+ i_k^t \times \prod_{m=1}^{M} Bin\left(x_{m,k}; v_{m,k}^t; p\left(m, i_k^t = 1, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t, \cdots, j_{M,k}^t),*} \times \Pr(i_k^t = 1) \Bigg) \\
&+ \gamma.
\end{aligned}
\tag{4.2}
$$

Similarly, for the alternative hypothesis (Chapter 2, Equation 2.13), the equation may be rewritten as:

$$
\begin{aligned}
&L_{H_1} \\
&= \sum_{k=1}^{N} \sum_{\boldsymbol{G_k^t}=(0,0,\cdots,0)}^{(2,2,\cdots,2)} E[I(\boldsymbol{G_k^t})|(\boldsymbol{x_k}, \boldsymbol{v_k^t}, i_k^t)] \\
&\times ln\Bigg((1 - i_k^t) \times \prod_{m=1}^{M} Bin\left(x_{m,k}; v_{m,k}^t; p\left(m, i_k^t = 0, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t, \cdots, j_{M,k}^t), i_k^t = 0} \times \Pr(i_k^t = 0) \\
&+ i_k^t \times \prod_{m=1}^{M} Bin\left(x_{m,k}; v_{m,k}^t; p\left(m, i_k^t = 1, j_{m,k}^t\right)\right) \times \boldsymbol{g}_{(j_{1,k}^t, \cdots, j_{M,k}^t), i_k^t = 1} \times \Pr(i_k^t = 1) \Bigg) \\
&+ \gamma.
\end{aligned}
\tag{4.3}
$$

4.3 Computer program execution time

To calculate the test statistics, and to estimate the MLG frequencies and the misclassification rates in the case-control datasets more efficiently, we developed a C computer program that implements our method. For data simulation, we have developed another program using both C and R. The source codes for these two programs may be found in the Appendix. Comments are provided to help the user understand how the codes implement the simulations, compute the EM-algorithms and statistics, and produce the output.

To evaluate the computer performance of our program, we measured the execution time for computing on various datasets with different numbers of loci, or different levels of sequencing coverage. The datasets tested were simulated by our simulation program. In the simulated datasets, we generally set the simulation parameters as the following (see simulation notation in Chapter 2), except for those listed particularly under each test:

Disease MOI:	Dominant
Number of controls:	1000
Number of cases:	500
Error rate in controls:	0.13
Error rate in cases:	0.1
α:	0.1
OR:	1
Frequency of non-disease MLG:	0.95.

For each of the tests below, we performed 500 runs.

4.3.1 Computer time on different number of loci tested

We tested the effect of loci number on computer time by running our program with a number of loci ranging from one to four. Four sets were tested with the sequencing coverage set to be 4. As illustrated in Figure 4.1, the time (measured in seconds) increases exponentially with the increased number of loci.

4.3.2 Computer time of different sequencing coverage on a single locus

With a single locus, we tested the effect of sequencing coverage on time. The number of sequencing coverage varies from 4x to 40x (4 sets). The time of execution increases as the sequencing coverage increases, but not exponentially. See Figure 4.2.

Figure 4.1 Computer program execution time on different number of loci

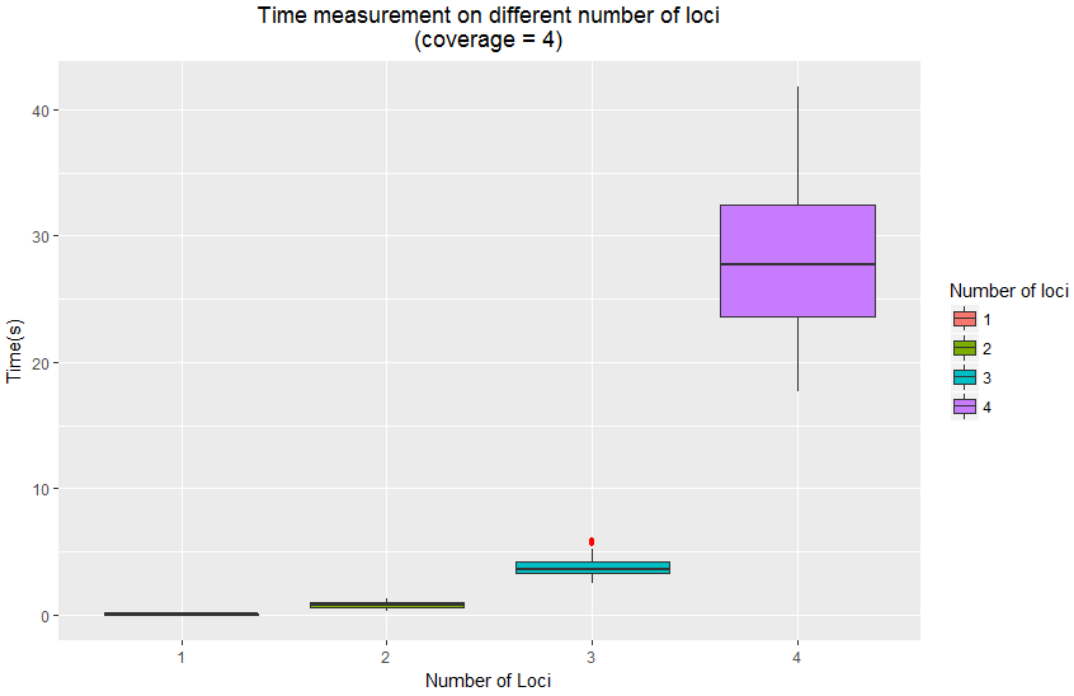
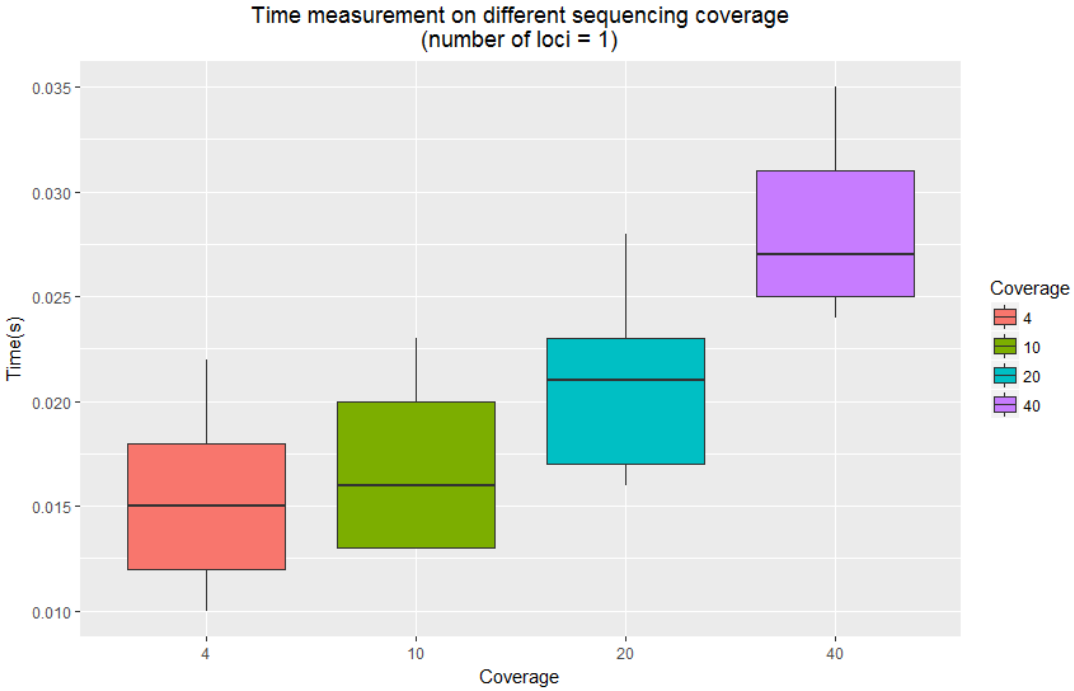


Figure 4.2 Computer program execution time on different sequencing coverage



4.3.3 Computer time of different sequencing coverage on two loci

To test whether the execution time acts similarly with more than one locus, we tested the program performance on two-locus datasets. The settings for sequencing coverage is the same as in Figure 4.2. From our results (Figure 4.3), we found that the time for execution in a two-locus setting decreases exponentially with the increase of sequencing coverage. This finding can be explained by the following: The likelihood of the MLG is a product of the individual binomial distribution. With a reasonably high coverage, the binomial probability for a more likely genotype from the observed data is much higher than that for a less likely genotype, even with the presence of sequencing error. Therefore, the binomial probabilities product for the less likely genotypes becomes practically 0 during the first few steps of a run, which decreases the number of steps required for the convergence to correct MLG. This outcome can be observed by noting the number of steps those tests took to achieve the maximum likelihood in Figure 4.4.

Figure 4.3 Computer program execution time on different sequencing coverage

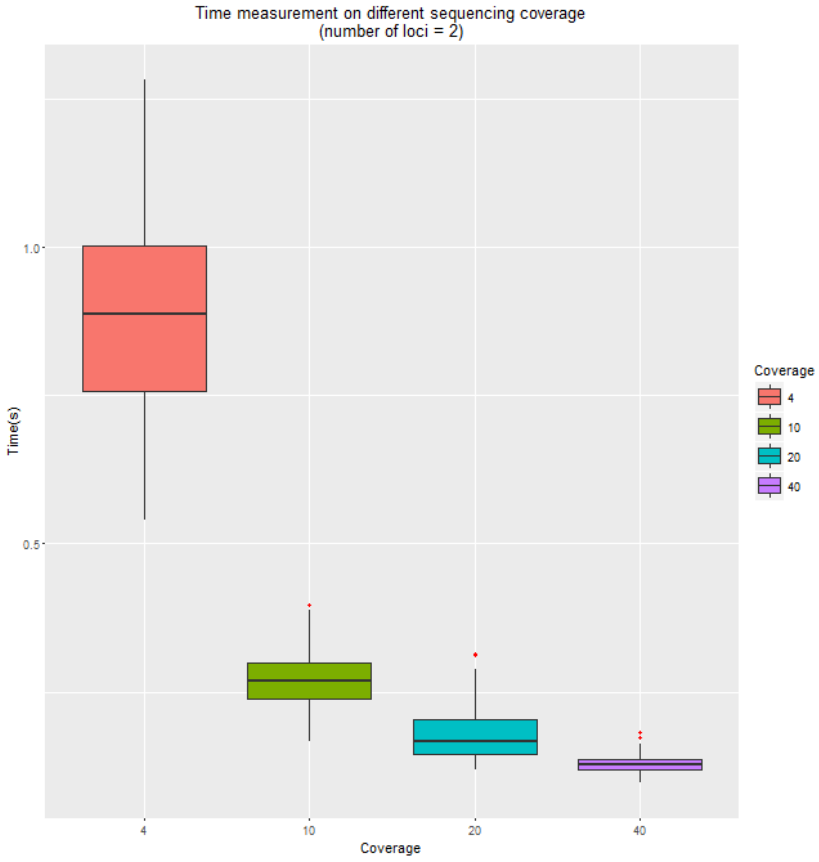
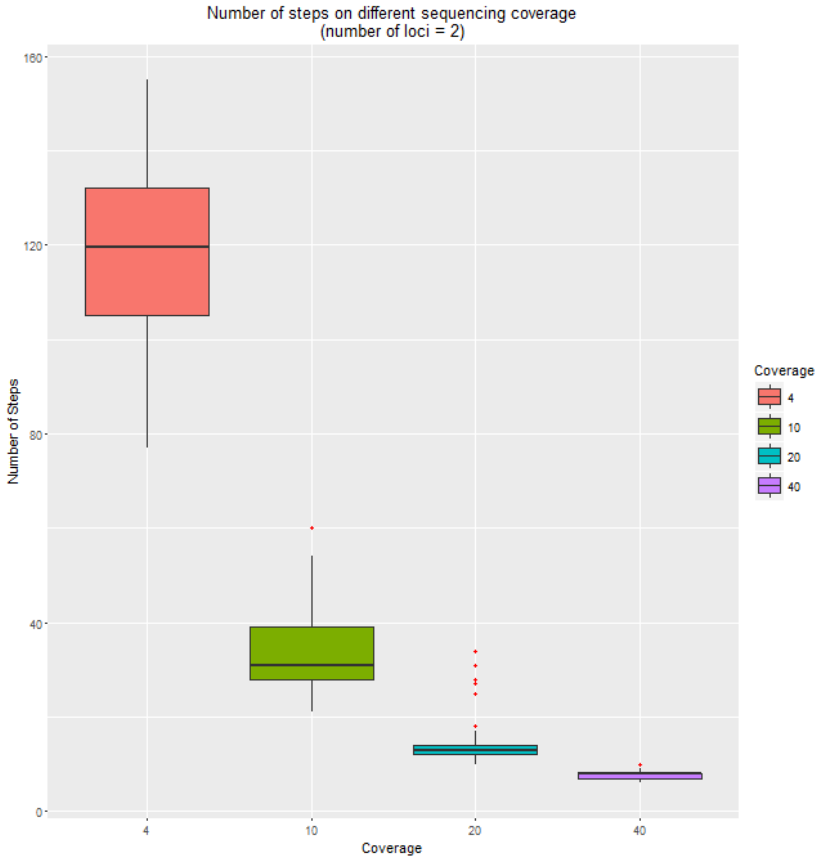


Figure 4.4 Number of steps to achieve maximum likelihood on different sequencing coverage



4.3.4 Computer time on real data: the 1000 Genomes Project data

We also tested the speed of our program on the real exome sequencing data from the 1000 Genomes Project. As described in *Section 3.2.4 Testing on real data of high quality*, we extracted four sequenced loci from chromosome 20 on positions 60905878, 60907675, 60908964 and 60908969. The number of individuals tested is 1006. In the extracted region, the non-reference allele counts are in a range of 3-89, with a mean of 23 and a median of 20; the high-quality sequencing coverages are in a range of 3-134, with a mean of 27.4 and a median of 25. We tested our program on this dataset with simulated affected status for

100 times, and average computer time is 2.198 seconds, with a maximum time of 6.075 seconds and minimum of 1.061 seconds.

4.4 Using double-sampling to increase genetic association test power

Among the different sequencing approaches, Sanger sequencing has a much higher accuracy rate (99.99%) compared to NGS. Thus, the Sanger approach may serve as a "gold standard" sequencing method, and may provide confirmation for NGS results [1]. If Sanger sequencing or some other highly accurate MLG classification method is available for a subset of individuals, we may extend our statistic test by using double-sampling.

In double-sampling procedures, samples are sequenced by one of two methods: a method that is cost-effective but "fallible" – with lower accuracy; or the other method that is "infallible" and has higher accuracy than the first, but is more expensive and may not be feasible for an entire study [2]. In the case of our study, the samples sequenced by NGS are considered "fallible" samples, while those sequenced by the Sanger method (for example) may be considered "infallible" samples. Because NGS is more economical than Sanger sequencing in genome sequencing [3], it is reasonable to assume that researchers would sequence all samples through NGS, and only double-sample a few through Sanger. The previously developed LRT_{ae} method can then be applied to the double-sampled data to gain higher test power [2]. Another example of double sampling is to sequence samples in a large cohort at low coverage (when the sequencing coverage is low, the sequenced outputs are generally considered "fallible") and combine with a subset of samples sequenced at high coverage ("infallible" method).

4.5 Advancement in high-throughput technologies

NGS technology is favored for its low cost and efficiency in population-scale sequencing [4], however, the short reads generated from these sequencing platforms makes it difficult apply in analyses of larger structural variations [5]. Also, *de novo* genome assembly using reads from NGS outputs could be problematic, because this could lead to missing key portions in the genome, or difficulty in identifying the position or number of repeats due to presence of repeating regions [6-9].

Thanks to the advancement of technology, the existing problems of *de novo* genome assembly that come from NGS, could possibly be solved by using the third-generation DNA sequencing technology. The currently available third-generation sequencing platforms can produce average read length of more than 10,000bp, with a few even reaching 100,000bp [5]. These commercial platforms include [5]: Pacific Biosciences (PacBio) Single Molecule Real Time (SMRT) sequencing [10], the Illumina Tru-seq Synthetic Long-Read technology [11] and the Oxford Nanopore Technologies sequencing platform [12]. With longer sequencing reads, the outputs from the third-generation sequencing technology generally span larger regions of the genome. It is believed that examination of these regions allows for the identification of more structural variations [13], such as insertions, deletions and translocations [5], as well as missing regions in genomes [14]. Moreover, the third-generation sequencing technology does not require synchronization, which eliminates the errors introduced by PCR (polymerase chain reaction) amplification and dephasing as in NGS process [15].

Sequencing data generated from these platforms should be able to be applied to our method with proper extensions on algorithms and equations. This data could also be applied to the aforementioned double-sampling approach (See Chapter 4, *Using double-sampling to increase genetic association test power*) with proper design.

Reference:

1. *Thermo Fisher Scientific. "Seq It Out"*. Available from: https://www.thermofisher.com/blog/behindthebench/when-do-i-use-sanger-sequencing-vs-ngs-seq-it-out-7/.
2. Gordon, D., et al., *Increasing power for tests of genetic association in the presence of phenotype and/or genotype error by use of double-sampling.* Stat Appl Genet Mol Biol, 2004. 3: p. Article26.
3. Schuster, S.C., *Next-generation sequencing transforms today's biology.* Nat Methods, 2008. 5(1): p. 16-8.
4. Mardis, E.R., *The impact of next-generation sequencing technology on genetics.* Trends in genetics, 2008. 24(3): p. 133-141.
5. Lee, H., et al., *Third-generation sequencing and the future of genomics.* 2016.
6. Schatz, M.C., A.L. Delcher, and S.L. Salzberg, *Assembly of large genomes using second-generation sequencing.* Genome research, 2010. 20(9): p. 1165-1173.
7. International, R.G.S.P., *The map-based sequence of the rice genome.* Nature, 2005. 436(7052): p. 793.
8. Lander, E.S., et al., *Initial sequencing and analysis of the human genome.* Nature, 2001. 409(6822): p. 860-921.
9. Li, R., et al., *The sequence and de novo assembly of the giant panda genome.* Nature, 2010. 463(7279): p. 311-317.
10. *Pacific Biosciences of California, Inc.*; Available from: http://www.pacb.com/smrt-science/smrt-sequencing/.
11. *Illumina, Inc.*; Available from: http://www.illumina.com/technology/next-generation-sequencing/long-read-sequencing-technology.html.
12. *Oxford Nanopore Technologies*. Available from: https://nanoporetech.com/.
13. Chaisson, M.J., et al., *Resolving the complexity of the human genome using single-molecule sequencing.* Nature, 2015. 517(7536): p. 608-611.
14. Ross, M.G., et al., *Characterizing and measuring bias in sequence data.* Genome biology, 2013. 14(5): p. 1.
15. Pareek, C.S., R. Smoczynski, and A. Tretyn, *Sequencing technologies and genome sequencing.* J Appl Genet, 2011. 52(4): p. 413-35.

Appendix 1. Source code for the statistical test (in C)

```
#define _GNU_SOURCE

#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include "BinomDist.h"
#include "chisqr.h"
#include "mapping_func.h"
#include "split.h"
/*
Last update:    21JAN2017
Created by:     Lisheng Zhou

This program will do the followings:
* Calculation of LRT (test statistic)
* Estimation of multilocus genotype frequencies, misclassification rates (error rates) using
EM algorithm
* Generate random starting points
```

```
MAX EM steps:        200
MAX locus: 10
MAX individuals: 1000

input file:       multi_locus_dataset.csv
(this input file will be generated from the Simulation program)

*/

int main()
{
        // BLOCK START: Read in data in the input file
        // number of tested loci will be calculated from the input file
        FILE *infile;
        char infileName[80] = "multi_locus_dataset.csv";
        infile = fopen(infileName, "r");
        if (infile == NULL)
        {
                fprintf(stderr, "Cannot open input file %s!\n", infileName);
                exit(1);
        }

        // count:  how many loci are tested in the dataset
```

```
int nloci;
char term[100];
fscanf(infile, " %100[^\n]", term);
char **list;
list = split(term, ",");

int n = 0;
while(list[n])
{
        n++;
}
nloci = (n - 2) / 2;

free(list);
free(list[0]);

// count: number of individuals
rewind(infile);
char test_count[100];
int num_ind = 0;
while(fscanf(infile, " %100[^\n]", test_count) != EOF){
        num_ind++;
}
```

```
int ind[num_ind];
int pheno[num_ind];
int x[num_ind][nloci];
int v[num_ind][nloci];

int case_count = 0;
int cont_count = 0;

char * line = NULL; //line read in
size_t len = 0;
ssize_t read;  // length of line

int order = 0;
rewind(infile);
while ((read = getline(&line, &len, infile)) != -1)
{
        char **array;
        array = split(line, ",");
        sscanf(array[0], "%i", &ind[order]);
        sscanf(array[1], "%i", &pheno[order]);

        if(pheno[order] == 0)
```

```
            {
                cont_count++;
            } else if (pheno[order] == 1)
            {
                case_count++;
            }

            for (int dat_v = 0; dat_v < nloci; dat_v++)
            {
                sscanf(array[dat_v + 2], "%i", &v[order][dat_v]);
            }
            for (int dat_x = 0; dat_x < nloci; dat_x++)
            {
                sscanf(array[dat_x + 2 + nloci], "%i", &x[order][dat_x]);
            }
            order++;
            free(array[0]);
            free(array);
    }
    fclose(infile);
    // BLOCK END: Read in data -- with number of loci not specified
```

```
/* BLOCK START: Initialization*/
double tolerance = 0.00001;              // difference tolerance to stop EM
double diff = 100;               // initialized different

double q_0 = (double)cont_count / (double)num_ind; // control rate
double q_1 = (double)case_count / (double)num_ind; // case rate

// total number of multilocus genotype (MLG): 3^nloci
int total_MLE = (int)pow(3, nloci);

double *err;
err = (double *)malloc(sizeof(double) * 2);
double *NullGenoFreq;
NullGenoFreq = (double *)malloc(sizeof(double) * total_MLE);
double *RandGeno;
RandGeno = (double *)malloc(sizeof(double) * total_MLE);

// Generate random multilocus geneotype (MLG) frequencies and error rates
FILE *ConstantFile;
char ConstantFileName[80] = "Constant.in";
ConstantFile = fopen(ConstantFileName, "r");
if (ConstantFile == NULL){
```

```
                fprintf (stderr, "Cannot open constant input file %s!\n",
ConstantFileName);
                exit(1);
        }
        long int seed;
        fscanf(ConstantFile, "%li\n", &seed);
//      double srand48();
        srand48(seed);
        fclose(ConstantFile);

        FILE *ConstantFileOut;
        ConstantFileOut = fopen(ConstantFileName, "w");
        double drand48();
        int tempSeed = drand48() * 100000000 ;
        fprintf(ConstantFileOut, "%i\n", tempSeed);
        fclose(ConstantFileOut);

        double randSum = 0;
        double drand48();

        // generate random starting points for error
        err[0] = drand48() * 0.03;
        err[1] = drand48() * 0.03;
```

```
		for (int i_hap = 0; i_hap < total_MLE; i_hap++)
		{
			RandGeno[i_hap] = drand48();
			randSum += RandGeno[i_hap];
		}
//		printf("starting points:\n");
		for (int i_hap2 = 0; i_hap2 < total_MLE; i_hap2++)
		{
			NullGenoFreq[i_hap2] = RandGeno[i_hap2] / randSum;

		}
		/* BLOCK END: Initialization*/

		/* BLOCK START: EM steps -- calculate NULL model likelihood*/
		int step = 0;

		double *pre_geno_freq; // to store the genoype frequencies of the previous step
		pre_geno_freq = (double *)malloc(sizeof(double) * total_MLE);

		double *pre_err; // to store the error rates of the previous step
		pre_err = (double *)malloc(sizeof(double) * 2);
```

```
double pre_lnSum; // to store the log(sum) of the previous step

double *inter_geno_freq;
inter_geno_freq = (double *)malloc(sizeof(double) * total_MLE);

double inter_error0, inter_error1, lnSum;

// while loop starts from here
// to initialize the parameters
while ((diff >= tolerance) && (step <= 200))
{
        // START: store previous values
        for (int init_geno_i = 0; init_geno_i < total_MLE; init_geno_i++)
        {
                pre_geno_freq[init_geno_i] = NullGenoFreq[init_geno_i];
                if (step > 0)
                {
                        NullGenoFreq[init_geno_i] =
inter_geno_freq[init_geno_i];
                }
        }
```

```
pre_err[0] = err[0];
pre_err[1] = err[1];

if (step > 0)
{
        err[0] = inter_error0;
        err[1] = inter_error1;

        pre_lnSum = lnSum;
}
// END: store previous values
for (int inter_geno_j = 0; inter_geno_j < total_MLE; inter_geno_j++)
{
        inter_geno_freq[inter_geno_j] = 0;

}

lnSum = 0;
double errorSum_num_cont = 0;
double errorSum_num_case = 0;
double errorSum_den_cont = 0;
double errorSum_den_case = 0;
```

```
for (int i = 0; i < num_ind; i++)
{
    // success rate for binomial probability
    double p[3];
    p[1] = 0.5;
    if (pheno[i] == 0)
    {
        p[0] = err[0];
        p[2] = 1 - err[0];
    } else if (pheno[i] == 1)
    {
        p[0] = err[1];
        p[2] = 1 - err[1];
    }

    // calculate binomial probability for each genotype
    double binom[nloci][3];
    for (int binom_loc = 0; binom_loc < nloci; binom_loc++)
    {
        binom[binom_loc][0] = binomial(x[i][binom_loc],
v[i][binom_loc], p[0]);
        binom[binom_loc][1] = binomial(x[i][binom_loc],
v[i][binom_loc], p[1]);
```

```
                    binom[binom_loc][2] = binomial(x[i][binom_loc],
v[i][binom_loc], p[2]);
                }

                // caulculate tau (posterior probability) numerator
                double *tau_num;
                tau_num = (double *)malloc(sizeof(double) * total_MLE);
                for (int tau_num_i = 0; tau_num_i < total_MLE;
tau_num_i++)
                {
                    tau_num[tau_num_i] = NullGenoFreq[tau_num_i];
                    int tau_num_remain = 0;
                    int tau_num_temp = tau_num_i;
                    int tau_num_geno;


                    for (int tau_num_loc = 0; tau_num_loc < nloci;
tau_num_loc++)
                    {
                        if (tau_num_loc != nloci - 1)
                        {
                            tau_num_remain =
tau_num_temp % (int)pow(3, (nloci - 1 - tau_num_loc));
```

```
                                                tau_num_geno =
(tau_num_temp - tau_num_remain) / (int)pow(3, (nloci - 1 - tau_num_loc));
                                                tau_num_temp =
tau_num_remain;
                                        }else{
                                                tau_num_geno =
tau_num_remain;
                                        }
                                        //printf("%i-%i:%lf,", tau_num_loc,
tau_num_geno, binom[tau_num_loc][tau_num_geno]);
                                        tau_num[tau_num_i] *=
binom[tau_num_loc][tau_num_geno];
                                }

                        }

                        // calculate tau denominator
                        double tau_sum = 0;
                        double temp_sum = 0; // calculate the sum of pre-LN for each
line
                        for (int tau_den_i = 0; tau_den_i < total_MLE; tau_den_i++)
                        {
                                tau_sum += tau_num[tau_den_i];
```

```
                    if (pheno[i] == 0)
                    {
                        temp_sum += tau_num[tau_den_i] * q_0;
                    //                  temp_sum                  +=
tau_num[tau_den_i];
                    }else if (pheno[i] == 1)
                    {
                    //      temp_sum += tau_num[tau_den_i];
                        temp_sum += tau_num[tau_den_i] * q_1;
                    }
                }

                // sum of LN
                lnSum += log(temp_sum);

                // calculate tau
                double *tau;
                tau = (double *)malloc(sizeof(double) * total_MLE);
                for (int tau_i = 0; tau_i < total_MLE; tau_i++)
                {
                    tau[tau_i] = tau_num[tau_i] / tau_sum;
                    inter_geno_freq[tau_i] += tau[tau_i]; // intermediate
genotype frequency of a specific locus
```

```
                    }
                    // error calculation -- numerator
                    double error_num = 0;

                    for (int error_num_i = 0; error_num_i < total_MLE;
error_num_i++)
                    {
                            int error_num_remain = 0;
                            int error_num_temp = error_num_i;
                            int error_num_geno;

                            for (int error_num_loc = 0; error_num_loc < nloci;
error_num_loc++)
                            {
                                    if (error_num_loc != nloci - 1)
                                    {
                                            error_num_remain =
error_num_temp % (int)pow(3, (nloci - 1 - error_num_loc));
                                            error_num_geno =
(error_num_temp - error_num_remain) / (int)pow(3, (nloci - 1 - error_num_loc));
                                            error_num_temp =
error_num_remain;
```

```
                                        }else{
                                                error_num_geno                  =
error_num_remain;
                                        }
                                        if (error_num_geno == 0)
                                        {
                                                error_num += tau[error_num_i]
* (double)x[i][error_num_loc];
                                        } else if (error_num_geno == 2)
                                        {
                                                error_num += tau[error_num_i]
* (double)(v[i][error_num_loc] - x[i][error_num_loc]);
                                        }
                                }

                        }

                        // error calculation -- denominator
                        double error_den = 0;

                        for (int error_den_i = 0; error_den_i < total_MLE;
error_den_i++)
                        {
```

```
int error_den_remain = 0;
int error_den_temp = error_den_i;
int error_den_geno;


for (int error_den_loc = 0; error_den_loc < nloci;
error_den_loc++)
{
    if (error_den_loc != nloci - 1)
    {
        error_den_remain =
error_den_temp % (int)pow(3, (nloci - 1 - error_den_loc));
        error_den_geno =
(error_den_temp - error_den_remain) / (int)pow(3, (nloci - 1 - error_den_loc));
        error_den_temp =
error_den_remain;
    }else{
        error_den_geno =
error_den_remain;
    }
    if (error_den_geno == 0)
    {
```

```
                                        error_den += tau[error_den_i] *
(double)v[i][error_den_loc];
                                    } else if (error_den_geno == 2)
                                    {
                                        error_den += tau[error_den_i] *
(double)v[i][error_den_loc];
                                    }
                                }

                            }
                            if (pheno[i] == 0)
                            {
                                errorSum_num_cont += error_num;
                                errorSum_den_cont += error_den;
                            }else if (pheno[i] == 1)
                            {
                                errorSum_num_case += error_num;
                                errorSum_den_case += error_den;
                            }

                            free(tau_num);
                            free(tau);
```

```
                }
                /* calculate INTERMEDIATE ERROR RATES */
                inter_error0 = errorSum_num_cont / errorSum_den_cont;
                inter_error1 = errorSum_num_case / errorSum_den_case;
                //          printf("%lf, %lf\n", inter_error0, inter_error1);
                //          printf("%lf\n", lnSum);

                /* calculate INTERMEDIATE GENOTYPE FREQUENCIES */
                for (int inter_geno_freq_i = 0; inter_geno_freq_i < total_MLE;
inter_geno_freq_i++)
                {
                        inter_geno_freq[inter_geno_freq_i]                    =
inter_geno_freq[inter_geno_freq_i] / (double)num_ind;
                        //          printf("%lf\n", inter_geno_freq[inter_geno_freq_i]);
                }

                if (step > 0)
                {
                        diff = lnSum - pre_lnSum;
                }
                step++;
        }
        double H0_LN = pre_lnSum;
```

```
printf ("EM ends...\nNull\nsteps: %i\n", step);
printf ("LN_H0: %lf\n", H0_LN);
printf ("MLG freq:\n");


for (int pre_geno_freq_i = 0; pre_geno_freq_i < total_MLE; pre_geno_freq_i++)
{
        printf("%lf,", pre_geno_freq[pre_geno_freq_i]);
//      fprintf(bootstrap, "%lf,", pre_geno_freq[pre_geno_freq_i]);
}
printf("\n");
printf ("Error: %lf, %lf\n", pre_err[0], pre_err[1]);
//  fprintf(bootstrap, "\n%lf, %lf\n", pre_err[0], pre_err[1]);


/* BLOCK START:EM steps -- calculate ALTERNATIVE model likelihood*/
int alt_step = 0;

// MLG freq and errors during calculation
double *err_alt;
err_alt = (double *)malloc(sizeof(double) * 2);
double *CaseGenoFreq;
CaseGenoFreq = (double *)malloc(sizeof(double) * total_MLE);
```

```
double *ContGenoFreq;
ContGenoFreq = (double *)malloc(sizeof(double) * total_MLE);

// to store MLG freq of the previous step for case and control
double *pre_geno_freq_case;
pre_geno_freq_case = (double *)malloc(sizeof(double) * total_MLE);
double *pre_geno_freq_cont;
pre_geno_freq_cont = (double *)malloc(sizeof(double) * total_MLE);

// to store error rates of the previous step -- alternative
double *pre_err_alt;
pre_err_alt = (double *)malloc(sizeof(double) * 2);

// to store log of sum of the previous step -- alternative
double pre_lnSum_alt;

// intermediate MLG freq
double *inter_geno_freq_case;
inter_geno_freq_case = (double *)malloc(sizeof(double) * total_MLE);
double *inter_geno_freq_cont;
inter_geno_freq_cont = (double *)malloc(sizeof(double) * total_MLE);

// intermediate error rates
```

```
double inter_error0_alt, inter_error1_alt, lnSum_alt;


double diff_alt = 100;
// while loop
// use the values from null
while ((diff_alt >= tolerance) && (alt_step <= 200))
{
        // START: use the MLG freq from null and store previous values
        for (int init_geno_m = 0; init_geno_m < total_MLE; init_geno_m++)
        {
                if ( alt_step == 0 ) {
                        pre_geno_freq_case[init_geno_m]         =
pre_geno_freq[init_geno_m];
                        pre_geno_freq_cont[init_geno_m]         =
pre_geno_freq[init_geno_m];

                        CaseGenoFreq[init_geno_m]               =
pre_geno_freq[init_geno_m];
                        ContGenoFreq[init_geno_m]               =
pre_geno_freq[init_geno_m];

                }
                else if ( alt_step > 0 ) {
```

```
                                        pre_geno_freq_case[init_geno_m]          =
CaseGenoFreq[init_geno_m];
                                        pre_geno_freq_cont[init_geno_m]          =
ContGenoFreq[init_geno_m];

                                        CaseGenoFreq[init_geno_m]                =
inter_geno_freq_case[init_geno_m];
                                        ContGenoFreq[init_geno_m]                =
inter_geno_freq_cont[init_geno_m];

                        }
                }
                //pre_err_alt[0] = pre_err[0];
                //pre_err_alt[1] = pre_err[1];

                if (alt_step > 0)

                {
                        pre_err_alt[0] = err_alt[0];
                        pre_err_alt[1] = err_alt[1];
                        err_alt[0] = inter_error0_alt;
                        err_alt[1] = inter_error1_alt;
                        pre_lnSum_alt = lnSum_alt;
```

```
} else if ( alt_step == 0 ) {
        pre_err_alt[0] = pre_err[0];
        pre_err_alt[1] = pre_err[1];
        err_alt[0] = pre_err[0];
        err_alt[1] = pre_err[1];
}
        // END: store previous values

for (int inter_geno_n = 0; inter_geno_n < total_MLE; inter_geno_n++)
{
        inter_geno_freq_case[inter_geno_n] = 0;
        inter_geno_freq_cont[inter_geno_n] = 0;
}

lnSum_alt = 0;

double errorSum_num_cont_alt = 0;
double errorSum_num_case_alt = 0;
double errorSum_den_cont_alt = 0;
double errorSum_den_case_alt = 0;

for (int m = 0; m < num_ind; m++)
{
```

```
// success rate for binomial probability
double p_alt[3];
p_alt[1] = 0.5;
if (pheno[m] == 0)
{
        p_alt[0] = err_alt[0];
        p_alt[2] = 1 - err_alt[0];
}else if (pheno[m] == 1)
{
        p_alt[0] = err_alt[1];
        p_alt[2] = 1 - err_alt[1];
}

// binomial probability
double binom_alt[nloci][3];
for (int binom_loc_alt = 0; binom_loc_alt < nloci; binom_loc_alt++)
{
        binom_alt[binom_loc_alt][0] = binomial(x[m][binom_loc_alt], v[m][binom_loc_alt], p_alt[0]);
        binom_alt[binom_loc_alt][1] = binomial(x[m][binom_loc_alt], v[m][binom_loc_alt], p_alt[1]);
```

```
                    binom_alt[binom_loc_alt][2] =
binomial(x[m][binom_loc_alt], v[m][binom_loc_alt], p_alt[2]);
                }

                // tau -- numerator
                double *tau_num_case;
                tau_num_case = (double *)malloc(sizeof(double) *
total_MLE);
                double *tau_num_cont;
                tau_num_cont = (double *)malloc(sizeof(double) *
total_MLE);

                for (int tau_num_m = 0; tau_num_m < total_MLE;
tau_num_m++)
                {
                    tau_num_case[tau_num_m] =
CaseGenoFreq[tau_num_m];
                    tau_num_cont[tau_num_m] =
ContGenoFreq[tau_num_m];
                    int tau_num_remain_alt = 0;
                    int tau_num_temp_alt = tau_num_m;
                    int tau_num_geno_alt;
                    /*
```

```
                    if (( m == 0 ) && ( tau_num_m == 1 )) {
                        printf( "Case: %lf\n",
tau_num_case[tau_num_m] );
                        printf( "Cont: %lf\n",
tau_num_cont[tau_num_m] );
                    }
                    */

                    for (int tau_num_loc_alt = 0; tau_num_loc_alt <
nloci; tau_num_loc_alt++)
                    {
                        if (tau_num_loc_alt != nloci - 1)
                        {
                            tau_num_remain_alt =
tau_num_temp_alt % (int)pow(3, (nloci - 1 - tau_num_loc_alt));
                            tau_num_geno_alt =
(tau_num_temp_alt - tau_num_remain_alt) / (int)pow(3, (nloci - 1 - tau_num_loc_alt));
                            tau_num_temp_alt =
tau_num_remain_alt;
                        }else{
                            tau_num_geno_alt =
tau_num_remain_alt;
                        }
```

```
                                        if (pheno[m] == 1)
                                        {
                                                tau_num_case[tau_num_m]   *=
binom_alt[tau_num_loc_alt][tau_num_geno_alt];
                                        }else
                                        {
                                                tau_num_cont[tau_num_m]   *=
binom_alt[tau_num_loc_alt][tau_num_geno_alt];
                                        }
                                }
                        }



                        // tau --denominator
                        double tau_sum_case = 0;
                        double tau_sum_cont = 0;
                        double temp_sum_alt = 0;
                        for (int tau_den_m = 0; tau_den_m < total_MLE;
tau_den_m++)
                        {

                                if (pheno[m] != 1)
                                {
```

```
                                        tau_sum_cont                    +=
tau_num_cont[tau_den_m];
                                        temp_sum_alt                    +=
tau_num_cont[tau_den_m] * q_0;
                                //      temp_sum_alt                    +=
tau_num_cont[tau_den_m];
                                }else if (pheno[m] == 1)
                                {
                                        tau_sum_case                    +=
tau_num_case[tau_den_m];
                                //      temp_sum_alt                    +=
tau_num_cont[tau_den_m];
                                        temp_sum_alt                    +=
tau_num_case[tau_den_m] * q_1;
                                }
                        }

                        // sum of LN
                        lnSum_alt += log(temp_sum_alt);

                        // tau
                        double *tau_case;
                        tau_case = (double *)malloc(sizeof(double) * total_MLE);
```

```
                double *tau_cont;
                tau_cont = (double *)malloc(sizeof(double) * total_MLE);
                for (int tau_m = 0; tau_m < total_MLE; tau_m++)
                {
                        tau_case[tau_m]  =  tau_num_case[tau_m]  /
tau_sum_case;
                        tau_cont[tau_m]  =  tau_num_cont[tau_m]  /
tau_sum_cont;
                        if ( pheno[m] == 1 ) {
                                inter_geno_freq_case[tau_m]          +=
tau_case[tau_m];
                        } else if ( pheno[m] == 0 ) {
                                inter_geno_freq_cont[tau_m]          +=
tau_cont[tau_m];
                        }
                }


                //error calculation -- numerator
                double error_num_alt = 0;

                for (int error_num_m = 0; error_num_m < total_MLE;
error_num_m++)
```

```
{
    int error_num_remain_alt = 0;
    int error_num_temp_alt = error_num_m;
    int error_num_geno_alt;

    for (int error_num_loc_alt = 0; error_num_loc_alt < nloci; error_num_loc_alt++)
    {
        if (error_num_loc_alt != nloci - 1)
        {
            error_num_remain_alt = error_num_temp_alt % (int)pow(3, (nloci - 1 - error_num_loc_alt));
            error_num_geno_alt = (error_num_temp_alt - error_num_remain_alt) / (int)pow(3, (nloci - 1 - error_num_loc_alt));
            error_num_temp_alt = error_num_remain_alt;
        }else{
            error_num_geno_alt = error_num_remain_alt;
        }
        if (error_num_geno_alt == 0)
```

```
                {
                        if (pheno[m] == 0)
                        {
                                error_num_alt        +=
tau_cont[error_num_m] * (double)x[m][error_num_loc_alt];
                        }else if (pheno[m] == 1)
                        {
                                error_num_alt        +=
tau_case[error_num_m] * (double)x[m][error_num_loc_alt];
                        }
                } else if (error_num_geno_alt == 2)
                {
                        if (pheno[m] == 0)
                        {
                                error_num_alt        +=
tau_cont[error_num_m] * (double)(v[m][error_num_loc_alt] - x[m][error_num_loc_alt]);
                        }else if (pheno[m] == 1)
                        {
                                error_num_alt        +=
tau_case[error_num_m] * (double)(v[m][error_num_loc_alt] - x[m][error_num_loc_alt]);
                        }
                }
        }
```

```
                }

                // error calculation -- denominator
                double error_den_alt = 0;

                for (int error_den_m = 0; error_den_m < total_MLE;
error_den_m++)
                {
                        int error_den_remain_alt = 0;
                        int error_den_temp_alt = error_den_m;
                        int error_den_geno_alt;

                        for (int error_den_loc_alt = 0; error_den_loc_alt <
nloci; error_den_loc_alt++)
                        {
                                if (error_den_loc_alt != nloci - 1)
                                {
                                        error_den_remain_alt =
error_den_temp_alt % (int)pow(3, (nloci - 1 - error_den_loc_alt));
                                        error_den_geno_alt =
(error_den_temp_alt - error_den_remain_alt) / (int)pow(3, (nloci - 1 - error_den_loc_alt));
```

```
                                        error_den_temp_alt              =
error_den_remain_alt;
                                    }else{
                                        error_den_geno_alt              =
error_den_remain_alt;
                                    }
                                    if (error_den_geno_alt == 0)
                                    {
                                        if (pheno[m] == 0)
                                        {
                                            error_den_alt         +=
tau_cont[error_den_m] * (double)v[m][error_den_loc_alt];
                                        }else if (pheno[m] == 1)
                                        {
                                            error_den_alt         +=
tau_case[error_den_m] * (double)v[m][error_den_loc_alt];
                                        }
                                    } else if (error_den_geno_alt == 2)
                                    {
                                        if (pheno[m] == 0)
                                        {
                                            error_den_alt         +=
tau_cont[error_den_m] * (double)v[m][error_den_loc_alt];
```

```
                                                }else if (pheno[m] == 1)
                                                {
                                                        error_den_alt        +=
tau_case[error_den_m] * (double)v[m][error_den_loc_alt];
                                                }
                                        }
                                }


                        }
                        if (pheno[m] == 0)
                        {
                                errorSum_num_cont_alt += error_num_alt;
                                errorSum_den_cont_alt += error_den_alt;
                        }else if (pheno[m] == 1)
                        {
                                errorSum_num_case_alt += error_num_alt;
                                errorSum_den_case_alt += error_den_alt;
                        }


                        free(tau_num_case);
                        free(tau_num_cont);
                        free(tau_case);
                        free(tau_cont);
```

```
                }

                /* INTERMEDIATE ERROR RATES */
                inter_error0_alt = errorSum_num_cont_alt / errorSum_den_cont_alt;
                inter_error1_alt = errorSum_num_case_alt / errorSum_den_case_alt;

                /* INTERMEDIATE GENOTYPE FREQUENCIES */
                for (int inter_geno_freq_m = 0; inter_geno_freq_m < total_MLE;
inter_geno_freq_m++)
                {
                        inter_geno_freq_case[inter_geno_freq_m]                 =
inter_geno_freq_case[inter_geno_freq_m] / (double)case_count;
                        inter_geno_freq_cont[inter_geno_freq_m]                 =
inter_geno_freq_cont[inter_geno_freq_m] / (double)cont_count;

                }

        //      printf("CaseFreq: %lf\n", inter_geno_freq_case[0]);
        //      printf("ContFreq: %lf\n", inter_geno_freq_cont[0]);

                if (alt_step > 0)
                {
                        diff_alt = lnSum_alt - pre_lnSum_alt;
```

```
                }
                alt_step++;
        }

        double H1_LN = pre_lnSum_alt;

        printf("Alternative\nsteps: %i\n", alt_step);
        printf("LN_H1: %lf\n", H1_LN);
        printf("MLG freq:\nCase:\n");

        for (int pre_geno_freq_m1 = 0; pre_geno_freq_m1 < total_MLE;
pre_geno_freq_m1++)
        {
                printf("%lf,", pre_geno_freq_case[pre_geno_freq_m1]);
        }
        printf("\n");
        printf("Control\n");
        for (int pre_geno_freq_m2 = 0; pre_geno_freq_m2 < total_MLE;
pre_geno_freq_m2++)
        {
                printf("%lf,", pre_geno_freq_cont[pre_geno_freq_m2]);
        }
        printf("\n");
```

```
printf ("Error: %lf, %lf\n", pre_err_alt[0], pre_err_alt[1]);

/* ALTERNATIVE - END */
double LRT = 2 * (H1_LN - H0_LN);
// int df = total_MLE + 1;
printf("LRT:\t%lf\n", LRT);

free(inter_geno_freq);
free(pre_geno_freq);
free(pre_err);

free(err);
free(NullGenoFreq);
free(RandGeno);

free(inter_geno_freq_case);
free(inter_geno_freq_cont);
free(pre_geno_freq_case);
free(pre_geno_freq_cont);
free(pre_err_alt);
```

```
        free(err_alt);
        free(CaseGenoFreq);
        free(ContGenoFreq);

        return 0;
}
```

Appendix 2. Source code for the simulation process

2.1. Generate input file for the simulation program (in C)

```
/*

Date:          June 06, 2016

Created by:    Lisheng Zhou

This program will generate an input file Sim_parameter.in

for data simulation program in the Appendix Section 2.2 Simulation program

Input file:

1.    Vector_setting_fixed.in

      Format of this input file:

      Line 1: number of locus

      Line 2: number of controls

      Line 3: number of cases

      Line 4: sequencing coverage value

      Line 5: misclassification in controls

      Line 6: misclassification in cases

      Line 7: based-line odds-ratio

      Line 8: odds-ratio

      Line 9: mode of inheritance (dominant only for this version, use "1")
```

From Line 10: each line contains one of the population MLG frequencies value (Line 10 should be the non-disease MLG frequency)

2. Constant.in: a file containing random number

Output file:

1. Sim_parameter.in

*/

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "mapping_func.h"

int main()
{
	FILE *infile;
	char infileName[80] = "Vector_setting_fixed.in";
	infile = fopen(infileName, "r");
	if (infile == NULL){
		fprintf(stderr, "Cannot open infile file %s!\n", infileName);
```

```
        exit(1);
}

// Read in parameters
// line #1. number of loci
int nloci;
fscanf(infile, "%i", &nloci);

int nMLE = pow(3, nloci);

// line #2. number of controls
int ncontrol;
fscanf(infile, "%i", &ncontrol);

// line #3. number of cases
int ncase;
fscanf(infile, "%i", &ncase);

// line #4. coverage
int cvrg;
fscanf(infile, "%i", &cvrg);

// line #5. error rate for control
```

```
        double err_cont;
        fscanf(infile, "%lg", &err_cont);

        // line #6. error rate for case
        double err_case;
        fscanf(infile, "%lg", &err_case);

        // line #7. disease prevalence
        double alpha;
        fscanf(infile, "%lg", &alpha);

        // line #8. odds ratio
        double OR;
        fscanf(infile, "%lg", &OR);

        // line #9. model
        // 1 --> dominant
        int model;
        fscanf(infile, "%i", &model);


//      printf("%i, %lg, %lg, %i\n", nloci, prevalence, OR, model);
```

```
        // END Read in parameters


        // Computation
        // beta
        double beta = log(OR);

//      printf("%lg\n", beta);


        // weight w_j
        double *w;
        w = (double *) malloc(sizeof(double) * nMLE);


        if (model == 1)
        // dominant model, this is the only model considered in this program
        {
                w[0] = 0;
                for (int w_i = 1; w_i < nMLE; w_i++)
                {
                        w[w_i] = 1;
                }
        }
        /*
        // generate randomized data
```

```
double *tempMLE;
tempMLE = (double *) malloc(sizeof(double) * nMLE);
double tempSum = 0;

for(int temp_i = 0; temp_i < nMLE; temp_i++)
{
        FILE *ConstantFile;
        char ConstantFileName[80] = "Constant.in";
        ConstantFile = fopen(ConstantFileName, "r");
        if (ConstantFile == NULL){
                fprintf (stderr, "Cannot open constant input file %s!\n",
ConstantFileName);
                exit(1);
        }
        long int seed;
        fscanf(ConstantFile, "%li\n", &seed);
        double srand48();
        srand48(seed);
        fclose(ConstantFile);

        FILE *ConstantFileOut;
        ConstantFileOut = fopen(ConstantFileName, "w");
        double drand48();
```

```
                long int tempSeed = drand48() * 100000000 ;
                fprintf(ConstantFileOut, "%li\n", tempSeed);
                fclose(ConstantFileOut);

                double drand48();

                tempMLE[temp_i] = drand48();
                tempSum += tempMLE[temp_i];
        }
        */

        // population MLE
        double *popMLE;
        popMLE = (double *) malloc(sizeof(double) * nMLE);

        for (int pop_i = 0; pop_i < nMLE; pop_i++)
        {
                fscanf(infile, "%lg", &popMLE[pop_i]);
        //      popMLE[pop_i] = tempMLE[pop_i] / tempSum;
//              printf("%lg\n", popMLE[pop_i]);
        }

        fclose(infile);
```

```
// Pr(aff|MLG)
// control
double *Pr0_j;
Pr0_j = (double *) malloc(sizeof(double) * nMLE);
// case
double *Pr1_j;
Pr1_j = (double *) malloc(sizeof(double) * nMLE);

// Pr(aff,MLG)
// control
double *Pr0MLG;
Pr0MLG = (double *) malloc(sizeof(double) * nMLE);
// case
double *Pr1MLG;
Pr1MLG = (double *) malloc(sizeof(double) * nMLE);


// MLEs
// control
double *MLE_0;
MLE_0 = (double *) malloc(sizeof(double) * nMLE);
// case
```

```
double *MLE_1;
MLE_1 = (double *) malloc(sizeof(double) * nMLE);

double prevalence_unaff = 0;
double prevalence_aff = 0;

for (int pr_i = 0; pr_i < nMLE; pr_i++)
{
        Pr0_j[pr_i] = 1 / ( 1 + exp( alpha + beta*w[pr_i] ));
        Pr1_j[pr_i] = exp( alpha + beta*w[pr_i] ) / ( 1 + exp( alpha + beta*w[pr_i] ));
        Pr0MLG[pr_i] = Pr0_j[pr_i] * popMLE[pr_i];
        prevalence_unaff += Pr0MLG[pr_i];
        Pr1MLG[pr_i] = Pr1_j[pr_i] * popMLE[pr_i];
        prevalence_aff += Pr1MLG[pr_i];
}

for (int MLEi = 0; MLEi < nMLE; MLEi++)
{
        MLE_0[MLEi] = Pr0MLG[MLEi] / prevalence_unaff;
        MLE_1[MLEi] = Pr1MLG[MLEi] / prevalence_aff;
}
```

```
FILE *outfile;
char outfileName[80] = "Sim_parameter.in";
outfile = fopen(outfileName, "w");
fprintf( outfile, "%i\n", nloci );
fprintf( outfile, "%i\n", ncontrol );
fprintf( outfile, "%i\n", ncase );
fprintf( outfile, "%i\n", cvrg );
fprintf( outfile, "%lg\n", err_cont );
fprintf( outfile, "%lg\n", err_case );


for (int i1 = 0; i1< nMLE; i1++)
{
        fprintf( outfile, "%.16lg\n", MLE_0[i1] );
}
fprintf( outfile, "-99\n" );
for (int i2 = 0; i2 < nMLE; i2++)
{
        fprintf( outfile, "%.16lg\n", MLE_1[i2] );
}

fclose(outfile);
```

```
        free(MLE_1);
        free(MLE_0);
        free(Pr1MLG);
        free(Pr0MLG);
        free(Pr1_j);
        free(Pr0_j);
        free(popMLE);
//      free(tempMLE);
        free(w);

        return 0;
}
```

2.2. Simulation program (in C)

```
/*
Date:           May 27, 2016
Created by:     Lisheng Zhou

This program is to do simulations according to
an input file:
1. Sim_parameter.in
```

(this input is generated by the program described in the Appendix Section *2.1 Generate input file for the simulation program*)

Output file:

1. multi_locus_dataset.csv

****

Updated: 06/02/2016 by Lisheng Zhou

* Format of the input file is updated

- Sim_parameter.in

****

*/

```
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "BinomDist.h"
#include "split.h"
#include "mapping_func.h"
```

```
#include "x_mapping.h"

typedef double error[2]; // double: list to store error rates
typedef double array3[3]; // double: array with 3 elements
int main()
{
        FILE *infile;
        char infileName[80] = "Sim_parameter.in";
        infile = fopen(infileName, "r");
        if (infile == NULL){
                fprintf(stderr, "Cannot open parameter input file %s!\n", infileName);
                exit(1);
        }

        // Read in parameters from input file

        // line #1. number of loci

        int nloci;
        fscanf(infile, "%i", &nloci);

        int nMLE = pow(3, nloci); // number of MLEs
```

```
// line #2. number of controls
int ncontrol;
fscanf(infile, "%i", &ncontrol);

// line #3. number of cases
int ncase;
fscanf(infile, "%i", &ncase);

// line #4. coverage
int cvrg;
fscanf(infile, "%i", &cvrg);

// line #5. error rate for control
error err;
fscanf(infile, "%lg", &err[0]);

// line #6. error rate for case
fscanf(infile, "%lg", &err[1]);

// from line #7: MLEs

// memory allocation for MLEs
double *MLE;
```

```
MLE = (double *)malloc(sizeof(double) * nMLE);

for (int scan_i = 0; scan_i < nMLE; scan_i++)
{
        fscanf(infile, "%lg", &MLE[scan_i]);
}

int check;
fscanf(infile, "%i", &check);
if (check != -99)
{
        fprintf(stderr, "Number of lines for MLEs is not correct!\n");
        exit(1);
}

double *MLE_alt;
MLE_alt = (double *)malloc(sizeof(double) * nMLE);

for (int scan_alt = 0; scan_alt < nMLE; scan_alt++)
{
        fscanf(infile, "%lg", &MLE_alt[scan_alt]);
//      printf("%.16lg\n", MLE_alt[scan_alt]);
}
```

```
fclose(infile);

// print out inputs
//		printf("%i\n%i\n%i\n%i\n", nloci, ncontrol, ncase, cvrg);
//		printf("%lf\n%lf\n", err[0], err[1]);
/*
printf("MLEs:\n");
for (int i = 0; i < nMLE; i++)
{
		printf("%lf\n", MLE[i]);
}
*/
// ---------------------------------DATA-SIMULATION-----------------------------------
---

FILE *outfile;
char outfileName[80] = "multi_locus_dataset.csv";
outfile = fopen(outfileName, "w");

int Y; // phenotype
// MLG
int *MLG;
```

```
MLG = (int *)malloc(sizeof(int) * nloci);
// success rate
double* successrate;
successrate = (double *)malloc(sizeof(double) * nloci);
// x
int* xarray;
xarray = (int *)malloc(sizeof(int) * nloci);

for (int individual = 0; individual < ncontrol+ncase; individual++)
{
        fprintf(outfile, "%i,", individual);

        if (individual < ncontrol){
            Y = 0;
    } else {
            Y = 1;
    }

        fprintf(outfile, "%i,", Y);

        for (int vi = 0; vi < nloci; vi++)
        {
                fprintf(outfile, "%i,", cvrg);
```

```
}

// [1] simulate multi-locus genotype
// generate random number 1 for genotype simulation
FILE *ConstantFile;
char ConstantFileName[80] = "Constant.in";
ConstantFile = fopen(ConstantFileName, "r");
if (ConstantFile == NULL){
        fprintf (stderr, "Cannot open constant input file %s!\n",
ConstantFileName);
        exit(1);
}
long int seed;
fscanf(ConstantFile, "%li\n", &seed);
double srand48();
srand48(seed);
fclose(ConstantFile);

FILE *ConstantFileOut;
ConstantFileOut = fopen(ConstantFileName, "w");
double drand48();
long int tempSeed = drand48() * 100000000 ;
fprintf(ConstantFileOut, "%li\n", tempSeed);
```

```
fclose(ConstantFileOut);


double drand48();
double r_g;
r_g = drand48();

//      printf("random number 1: %.9lf\n", r_g);


double g_sum = 0;
int tempMLG;
double tempMLE;

for ( int i_g = 0; i_g < nMLE; i_g++ )
{
        if ( Y == 0 )
        {
                tempMLE = MLE[i_g];
        } else if ( Y == 1 )
        {
                tempMLE = MLE_alt[i_g];
        }

        if (( r_g > g_sum ) && ( r_g <= (g_sum + tempMLE) ))
```

```
                {
                        tempMLG = i_g;
                        break;
                }
                g_sum += tempMLE;

        }
        //      printf( "%i\n", tempMLG);

        // convert the number into multi-locus genotype
        map_in( tempMLG, nloci, MLG );

/*
        printf("selection of MLG-");
        for (int printMLGi = 0; printMLGi < nloci; printMLGi++ )
        {
                printf("%i", MLG[printMLGi]);
        }
        printf("\n");

        printf("Error rates: %0.3f, %0.3f\n", err[0], err[1]);
*/
        // [2] simulate alternative read counts (x)
```

```
// generate random number 2 for x simulation
FILE *ConstantFile2;
ConstantFile2 = fopen(ConstantFileName, "r");
if (ConstantFile2 == NULL){
        fprintf (stderr, "Cannot open constant input file %s!\n",
ConstantFileName);
        exit(1);
}
long int seed2;
fscanf(ConstantFile2, "%li\n", &seed2);
//double srand48();
srand48(seed2);
fclose(ConstantFile2);

FILE *ConstantFileOut2;
ConstantFileOut2 = fopen(ConstantFileName, "w");
double drand48();
long int tempSeed2 = drand48() * 100000000 ;
fprintf(ConstantFileOut2, "%li\n", tempSeed2);
fclose(ConstantFileOut2);

double drand48();
double r_x;
```

```
r_x = drand48();



// success rate of binomial distribution
array3 p;


p[0] = err[Y];
p[1] = 0.5;
p[2] = 1 - err[Y];


// success rates according to MLG
for (int success_i = 0; success_i < nloci; success_i++)
{
        successrate[success_i] = p[MLG[success_i]];
        //          printf("%lf\n", successrate[success_i]);
}


// a table of X probabiluty according to the multi-locus genotype


// number of total possible x
int total_x_order = (int) pow( cvrg+1, nloci );
// sum of probabilities of x
double sum_x = 0;
```

```
for ( int x_i = 0; x_i < total_x_order; x_i++ )
{
        x_map_in( x_i, nloci, xarray, cvrg );
        double temp_sum = 1;
        for ( int xnloci = 0; xnloci < nloci; xnloci++ )
        {
//              printf("%0.9lf,", binomial(xarray[xnloci], cvrg,
successrate[xnloci]));
                temp_sum *= binomial(xarray[xnloci], cvrg,
successrate[xnloci]);
        }
//      printf("\nPr(x1,x2,x3,x4): %0.9lf\n", temp_sum);
//      printf("cumulative probability: %0.9lf\n", sum_x+temp_sum);
        if ((r_x > sum_x) && (r_x <= sum_x + temp_sum))
        {
                break;
        } else {
                sum_x += temp_sum;

        }
}
```

```
//          printf("\nrandom number 2: %.9lf\nselected x:", r_x);
            for ( int print_x = 0; print_x < nloci; print_x++ )
            {
//                  printf("%i,", xarray[print_x]);
                    fprintf(outfile, "%i,", xarray[print_x]);
            }
//          printf("\n");
            fprintf(outfile, "\n");

        }

        fclose(outfile);

//          free(x);
        free(xarray);
        free(successrate);
        free(MLG);
        free(MLE_alt);
        free(MLE);

        return 0;
}
```

Appendix 3. Source code for the permutation step (in R)

```
## Date: 03/30/2016

## Created by: Lisheng Zhou

## Purpose: data simulation for bootstrap based on estimated parameters


## ==============================READ IN DATA


##setwd("C:/Users/zhou/Desktop/Today/R bootstrap simulation")

ori_data<-read.csv("multi_locus_dataset.csv",header=F)


## number of loci

n.loci=(dim(ori_data)[2]-2)/2

## number of individuals

n.k=dim(ori_data)[1]



## individual list

ind=ori_data[,1]

## phenotype list

Y=ori_data[,2]

## coverage matrix

V=data.matrix(ori_data[,3:(dim(ori_data)[2]-n.loci)])
```

```
## causal variant counts
X=data.matrix(ori_data[,(dim(ori_data)[2]-n.loci+1):dim(ori_data)[2]])
##================================END OF DATA READ IN


n0=length(which(Y==0))
n1=length(which(Y==1))
N=n0+n1
Y=sample(c(rep(0,n0),rep(1,n1)),N)


dat=data.frame(IND=ind,Y=Y,V=V,X=X)
write.table(dat,
file="multi_locus_dataset.permuted.csv",col.names=F,row.names=F,sep=",")
```

Appendix 4. Source code for utility functions

4.1. Binomial Distribution

```
#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <string.h>


/*Binomial Distribution Function (probability mass function, not cummulative*/

#include "BinomDist.h"


//factor function

double fact(int x)

{

 double i;

 double f = 1;

 for (i = x; i > 1; i--)

 {

  f = f * i;

 }

 return f;

} // end of factor function


//Binomial distribution
```

```
double binomial(short x, short n, float p)
{
        double pmf = (fact(n)/(fact(x)*fact(n-x)))*pow(p,x)*pow((1-p),(n-x));
//      double pmf = rfact(x,n)/fact(x)*pow(p,x)*pow((1-p),(n-x));
        return pmf;
} //end of Binomial distribution
```

4.2. Mapping Function

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "mapping_func.h"

// from number to vector

void map_in(int InNum, int NumLocus, int GenomArray[]){


        int remain = 0;
        int TempIn = InNum;
        int TempGeno = 0;
```

```
        for (int i = NumLocus-1; i >= 0; i--){
                remain = TempIn % (int)pow(3, i);
                TempGeno = (TempIn - remain)/(int)pow(3,i);
                TempIn = remain;
                GenomArray[i] = TempGeno;
        }
}

int map_out(int GenomArray[], int NumLocus){
        int sum = 0;
        for (int i = 0; i < NumLocus; i++){
                sum += GenomArray[i] * (int)pow(3,i);
        }
        return sum;
}
```

4.3. Splitting function

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#include "split.h"
```

```
char **split ( const char *s1, const char *s2) {

    char **lista;
    char *aux = (char*)malloc(strlen(s1) + 1);
    strcpy(aux, s1);
    char *token_Ptr;
    int i = 0;

    lista = (char **) malloc (sizeof (char *));
    token_Ptr = strtok(aux, s2);
    lista[i] = token_Ptr;
    i++;
    while(token_Ptr != NULL)
    {
        lista = (char **)realloc(lista, sizeof(char*) * (i + 1));
        token_Ptr = strtok(NULL, s2);
        lista[i] = token_Ptr;
        i++;
    }
    return lista;
}
```

Appendix 5. Instruction for running a simulation test

Following this instruction, reads may run the simulation program, calculate the test statistic and misclassification estimates from the simulated data, and even perform a permutation on the individuals' affection statuses in the simulated data. Before running the programs listed above, reads need to compile the source codes if they are written in C (this step will not be provided in this work).

5.1. Simulate NGS raw data

5.1.1. Data preparation

Readers must generate a proper formatted input file for the simulation program. The program that generates the right-formatted input file is provided in the Appendix Section *2.1 Generate input file for the simulation program (in C)*. However, this program requires two input files, a file containing all parameters (File "Vector_setting_fixed.in"), and a file containing a random number (File Constant.in). Here is the example file "Vector_setting_fixed.in":

Line 1: number of locus (e.g. 2)

Line 2: number of controls (e.g. 500)

Line 3: number of cases (e.g. 500)

Line 4: sequencing coverage value (e.g. 4)

Line 5: misclassification in controls (e.g. 0.01)

Line 6: misclassification in cases (e.g. 0.05)

Line 7: based-line odds-ratio (e.g. 0.1)

Line 8: odds-ratio (e.g. 1)

Line 9: mode of inheritance (dominant only for this version, use "1") (e.g. 1)

From Line 10: each line contains one of the population MLG frequencies value (Line 10 should be the non-disease MLG frequency)

e.g. Line 10: 0.75

Line 11: 0.006342811

Line 12: 0.0320819

Line 13: 0.03839475

Line 14: 0.045684489

Line 15: 0.038255781

Line 16: 0.037676154

Line 17: 0.008587083

Line 18: 0.042977033

5.1.2. Data Simulation

Use the output file "Sim_parameter.in" generated from the above step as the input in the simulation program described in *Appendix Section 2.2 Source code for the statistical test (in C)*. Run the simulation program and reads will get an output file named "multi_locus_dataset.csv".

5.2. Calculate test statistic and misclassification estimates

Use the simulated dataset "multi_locus_dataset.csv" as an input for the program described in *Appendix 1 Source code for the statistical test (in C)*. Run the program and the test statistic and misclassification estimate will be printed out to the screen as outputs.

5.3. Permutation program

Use the simulated dataset "multi_locus_dataset.csv" as an input and run the R script described in *Appendix 3 Source code for the permutation step (in R).* A Permuted file

named “multi_locus_dataset.permuted.csv” will then be generated, that may be used for further testing.