Carnegie Mellon University

SCALABLE PRIVACY-PRESERVING DATA SHARING METHODOLOGY FOR GENOME-WIDE ASSOCIATION STUDIES

A Dissertation Submitted to the Graduate School

in Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

in

Statistics

by

Fei Yu

Department of Statistics
Carnegie Mellon University
Pittsburgh, PA 15213

February 11, 2015

ABSTRACT

Rapid developments in whole-genome sequencing technologies in recent years have made the collection of high quality genetic data faster and more affordable. Because many types of genetic research can benefit from having a large amount of genetic data, the sharing of genetic data becomes crucial to propelling the quality of genetic studies. However, following the publication of a statistical attack on genome-wide association study (GWAS) data proposed by Homer et al. (PLoS Genetics 2008), protecting the privacy of individual-level information in GWAS databases has become a major concern for genetic researchers.

Traditional statistical methods for confidentiality and privacy protection of statistical databases do not scale well to deal with GWAS databases, especially in terms of guarantees regarding protection from linkage to external information. The more recent concept of differential privacy is an approach that provides a rigorous definition of privacy with meaningful privacy guarantees in the presence of arbitrary external information. Building on the notion of differential privacy, we explore methods for differentially private release of GWAS data. First, we describe methods for releasing single-nucleotide polymorphisms (SNPs) that are most relevant to a disease, which is one of the most common tasks in a GWAS. We design our methods for GWAS's that use χ^2 statistic as the measure of relevance, and we show that our methods can be extended to other measures of relevance. We present results of applying our methods for releasing the most relevant SNPs to a real human genetic dataset from a GWAS of Crohn's disease. Second, we describe methods for releasing regression coefficients. We propose a method that allows genetic researchers to perform a wide range of regression analyses, including ℓ_1 and ℓ_2 penalized logistic regressions. The major advantage of our method is that it incorporates regularization parameter selection, which ensures that our method mimics regression analyses done in a normal GWAS. By combing our methods for releasing the most relevant SNPs with our method for releasing regression coefficients, we have developed an end-to-end differentially private method for solving high-dimensional regression problems in a GWAS. We present results of applying our method for releasing regression coefficients to a simulated GWAS dataset.

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

[1] Fei Yu, Stephen E. Fienberg, Aleksandra B Slavković, and Caroline Uhler. "Scalable privacy-preserving data sharing methodology for genome-wide association studies." In: *Journal of Biomedical Informatics* 50C (Feb. 2014), pp. 133–141.

[2] Fei Yu, Michal Rybar, Caroline Uhler, and Stephen E. Fienberg. "Differentially-private logistic regression for detecting multiple-SNP association in GWAS databases." In: *Privacy in Statistical Databases*. Ed. by Josep Domingo-Ferrer. Vol. 8744. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 170–184.

[3] Fei Yu and Zhanglong Ji. "Scalable privacy-preserving data sharing methodology for genome-wide association studies: an application to iDASH healthcare privacy protection challenge." In: *BMC Medical Informatics and Decision Making* 14 Suppl 1 (Dec. 2014), S3.

[4] Aleksandra Slavkovic and Fei Yu. "Genomic privacy: risk and protection methods." In: *CHANCE* 28.2 (2015), Submitted.

Software implementations of some of the methods are available at:

- `https://github.com/fy/compare_dp_mechanisms`
- `https://github.com/fy/dp_penalized_logistic_regression`

ACKNOWLEDGEMENTS

I am grateful to my advisor Steve Fienberg for always showing me the bright side of graduate study. I would like to thank Steve Fienberg, Kathryn Roeder, Chad Schafer, Sesa Slavković, Jing Lei, and Bernie Devlin for their contributions to my intellectual and academic development. I would also like to acknowledge my collaborators, including Caroline Uhler, Michal Rybar, and Zhanglong Ji. I want to acknowledge the finanical support I received from NSF Awards EMSW2-RTG and BCS-0941518 for pursuing my dissertation research.

Thank you to my fellow graduate students for engaging me in stimulating conversations, joining me in thrilling intramural games, and making my time in Pittsburgh wonderful.

All that I am, I owe to my parents. I am eternally thankful to them.

Contents

List of Figures

List of Tables

List of Programs

ACRONYMS

WTCCC	Wellcome Trust Case Control Consortium
i.i.d.	independent and identically distributed
GWAS	genome-wide association study
SNP	single-nucleotide polymorphism
CDF	cumulative density function

Part I

INTRODUCTION

1 INTRODUCTION

1.1 Introduction

Agencies around the world have been putting considerable efforts into collecting clinical and genetic data, and building large databases (e.g., Database of Genotype and Phenotype[1] [dbGaP] at the U.S. National Library of Medicine) in an effort to support the personalized health care initiatives. Genetic researchers have made big strides in improving whole-genome sequencing technologies over the past decade. The collection of high quality genetic data is becoming faster and more affordable, enabling genetic researchers to gather more data and explore a wider range of research interests. Genome-wide association studies (GWAS's), in particular, reap great benefits from having a large DNA sample size. In a typical GWAS setting, researchers examine a large number of single-nucleotide polymorphisms (SNPs) and try to identify genetic factors that are associated with a phenotype (e.g., a common disease). Increasing the number of DNA samples available for analysis allows researchers to make more accurate statistical inference, fit more complex statistical models, and improve the overall quality of their analyses. Catalyzed by researchers' continually improving ability to collect high quality genetic data, growth in the number of GWAS publications per year is maintained at a remarkable pace: according to a report by the National Human Genome Research Institute (NHGRI; Welter et al.

1 http://www.ncbi.nlm.nih.gov/gap

[34]), the total number of GWAS published rose from less than 50 in 2005 to around 900 in 2010 and to almost 2,000 in 2013.

To take full advantage of the large amount of genetic data collected, it is imperative that data are shared among researchers. Not only is the sharing of genetic data essential for forming larger datasets for analysis, but it also makes resource allocation more efficient by reducing the number of duplicate experiments, and helps with data quality assessment as new experiments can use previously collected data as benchmarks. Despite the potential benefits, genetic data sharing is not a common practice among genetic researchers. While it is clear that individual-level genetic data deserve a high level of protection, for many years researchers believed that releasing statistics aggregated from thousands of individuals in a GWAS would not compromise the genetic study participants' privacy. Such a belief came under challenge with the publication of a statistical attack proposed by Homer et al. [14], which demonstrated that one can combine minor allele frequencies published in a GWAS and genetic data from publicly available sources, such as SNP data from the HapMap[2] project, to infer whether an individual has participated in that particular GWAS.

The publication of Homer et al. [14] drew widespread attention. Concerned with the potential breaches of genetic study participants' privacy, the National Institutes of Health (NIH) quickly responded to the Homer et al. [14] attack by removing all aggregate genetic data from open-access databases (e.g., see [8, 40]) and instituting an elaborate approval process that every researcher has to go through in order to gain access to aggregate genetic data. Other genetic data curating agencies, including the Broad Institute and the Wellcome Trust Centre for Human Genetics, also adopted NIH's policy. This NIH policy remains in effect today.

Tightened genetic data access policies set in motion two movements in the genetics research community: (1) studying potential privacy breaches associated with controlled release of genetic data; and (2) advocating full access to personal genetic data. Many

2 http://hapmap.ncbi.nlm.nih.gov/

publications have been devoted to refining and extending the potential privacy breaches initially proposed by Homer et al. [14], which exploit genetic information stored in an aggregate form through minor allele frequencies. For example, Clayton [7] proposed a Bayesian approach to testing the membership status of an individual in a particular sample, and Lumley and Rice [20] used regression results to predict a study participant's disease status. Some researchers harness genetic summaries to impute and recover individual level data (e.g., Zhou et al. [42]). We have also seen breaches that take advantage of the linkage between genetic data and metadata (e.g., Gymrek et al. [12]). For an overview of these and additional similar attacks, see Section 2.

On the other hand, some researchers advocate full openness to the personal genetic data (e.g., Personal Genome Project[3]), accepting that privacy in this setting may not be feasible. In July 2013, over 70 leading medical and research organizations from around the world, including the NIH and the Wellcome Trust, declared their intent to form a global alliance to build a framework for sharing genetic and clinical data they collect from genetic study participants. Therefore, it will be important and timely for us to understand the underlying privacy and confidentiality risks of genetic data sharing, and possible methods in statistics and computer science that will enable us to share usable genetic data while minimizing disclosure risk.

Researchers have started thinking about how to provide privacy protection while preventing linkage attacks on genetic data. Notable entities that have made significant contributions to developing these methods include the National Science Foundation (NSF) grant "Collaborative Research: Integrating Statistical and Computational Approaches to Privacy" shared by Carnegie Mellon University, Penn State University, and Cornell University, the *Integrating Data for Analysis, Anonymization and SHaring*[4] (iDASH) center at

3 `http://www.personalgenomes.org/`

4 `http://idash.ucsd.edu/`

University of California, San Diego, and the *Laboratory for Communications and Applications*[5].

One of the main difficulties of privacy protection is that it is almost impossible to control auxiliary information available to an attacker. Many attacks on genetic data rely on strong correlations between released databases, and the sparsity and high dimensionality of genetic data. While such strong correlations are often explicitly masked thanks to genetic data curating agencies' regulations (e.g., NIH's HIPAA Privacy Rule), these correlations can still be revealed in varying degrees by auxiliary information.

More recent research on privacy tries to take into account the possibility of unforeseeable availability of auxiliary information by being very precise about what kind of privacy guarantees can be offered. Out of various privacy protection approaches, differential privacy (Dwork et al. [10]) is quickly becoming a widely acceptable model for privacy protection. Suppose that a person is in the dilemma of participating in a study that sequences her genetic data and worrying that the data or results of the study can somehow be used in an unfavorable manner against her, such as denying her insurance. Differential privacy tries to alleviate the concerns of such a user. Any analysis carried out using differential privacy is endowed with the guarantee that, whether or not you decide to take part in the study, an intruder (e.g., an insurance company) will not learn anything more about you than what s/he already knows about you.

Such a strong privacy guarantee of course is not easy to provide, and it may come at a serious price in terms of data utility. Researchers are working on developing differentially private algorithms for genetic data sharing. For example, Uhler et al. [29], Yu et al. [37], and Johnson and Shmatikov [17] were the first to propose differentially private algorithms for releasing SNPs that are most strongly associated with a phenotype (e.g., a disease), which is a task commonly carried out in genome-wide association studies. There are recent extensions that release coefficients of penalized logistic regressions in this setting as well (e.g., Yu et al. [39]). These algorithms have been applied to a real human

5 `http://lca.epfl.ch/`

GWAS dataset and evaluated by analyses of the trade-off between privacy protection and statistical utility of the released data, and they have shown a great promise of supporting broader sharing of genetic data with rigorous privacy guarantees.

1.2 Thesis organization

This thesis is organized as follows:

- In Chapter 2, we begin with an in-depth analysis of the Homer et al. [14] attack, which is the most widely-discussed attack on genetic databases and prompted NIH to institute a more restrictive policy on genetic data access. By analyzing the results of applying the Homer et al. [14] attack to a real human GWAS dataset, we show that the attack has limited applicability. We then review representative statistical attacks on genetic database in the literature, giving insight into potential risks posed by these attacks and their limitations.
- In Chapter 3, we review definitions and notation related to differential privacy. We describe techniques for constructing differentially private algorithms and provide a toy example of differential privacy.
- In Chapter 4, we examine methods for privacy preserving release of the most relevant SNPs in a genome-wide association study. We review two general-purpose algorithms for releasing a subset from a large set of items differentially privately. Both algorithms require a score function. With a focus on GWAS, we describe in details two score functions: χ^2 statistic and Hamming distance score. We explain how to calculate the sensitivity of χ^2 statistic and how to calculate the Hamming distance scores. Lastly, we apply the differentially private release methods to a real human GWAS dataset and analyze how well the methods preserve statistical utility while they protect data privacy.

- In Chapter 5, we describe a differentially private method for releasing coefficients of penalized logistic regressions. Our method not only solves solving regression problems with any convex penalty functions, but also handles the selection of regularization parameters by cross-validation. We provide the exact form of the random noise used in the objective function perturbation mechanism and show that the perturbation noise can be efficiently sampled. We also apply the method to a simulated GWAS dataset and analyze the method's performance.

2

A REVIEW OF STATISTICAL ATTACKS ON GENETIC DATABASES

In this chapter, we review a wide range of statistical attacks on genetic databases. These attacks differ in many respects: some make different assumptions on the data, some define different hypothesis tests, some use different statistical techniques, and some address different privacy risks. In our review, we give insight into potential risks of sharing genetic data as well as limitations of statistical attacks on genetic databases.

The chapter begins with notation that we use for describing all the attacks. Then we provide an in-depth analysis of the Homer et al. [14] attack, which is the most widely-discussed attack on genetic databases and prompted NIH to institute a more restrictive policy on genetic data access. By analyzing the results of applying the Homer et al. [14] attack to a real human GWAS dataset, we show that the attack has limited applicability.

The Homer et al. [14] attack has spurred researchers' interest in genetic data privacy issues. Many publications have been devoted to refining and extending the potential privacy breaches initially proposed by Homer et al. [14], and analyzing genetic data privacy issues from different angles. In the remainder of the chapter, we review representative statistical attacks in the literature, giving insight into potential risks posed by these attacks and their limitations.

2.1 Notation

Suppose that we sample from the population $\mathbb{P}$ two independent collections of DNA samples, and we denote them by F and G. Let Y denote the DNA of an arbitrary individual sampled also from $\mathbb{P}$. Denote the genotype of Y at SNP$_i$ by y_i = 0, 0.5, or 1, which indicates that the number of minor alleles at SNP$_i$ is 0, 1, or 2, respectively. Denote the minor allele frequencies of F and G at SNP$_i$ by $f_i, g_i \in [0, 1]$, respectively. Suppose that there are N_F and N_G individuals in F and G, respectively, and that F, G, and Y contain information about a common set of SNPs of size M.

2.2 The Homer et al. [14] attack

The Homer et al. [14] attack is a statistical attack that makes inference on the *membership* of Y. The membership of Y is referred to as the DNA collection—F, G, or neither F nor G—that Y belongs to. In other words, the goal of the attack is to identify whether Y belongs to F, G, or neither. Homer et al. [14] argued that if Y belongs to one of the DNA collections, say F, then the genotypes of Y across all M SNPs are on average closer to the minor allele frequencies of F than G; on the other hand, if Y is in neither F nor G, then the genotypes of Y across all M SNPs are on average equally far away from the minor allele frequencies of either DNA collection. To contrast the similarity between Y and F and the similarity between Y and G at a particular SNP, Homer et al. [14] defined the following distance metric:

$$D_i(Y) = |y_i - f_i| - |y_i - g_i|,$$

where y_i, f_i and g_i are defined in Section 2.1. Therefore, if Y belongs to F but not G, then $D_i(Y)$ is more likely to be negative, as Y is more similar to F than G; if Y belongs to G but not F, then $D_i(Y)$ is more likely to be positive; and if Y belongs to neither F nor G, then $D_i(Y)$ is equally likely to be positive or negative.

To determine whether Y belongs to F, G, or neither, Homer et al. [14] derived a one-sample t-test statistic to test the null hypothesis that Y is in neither F nor G against one of the two alternative hypotheses: (i) Y is in either F or G, and (ii) Y is in F. The t-test statistic is based on the distance metric $D_i(Y)$. Let $\overline{D}$ and S^2 denote the sample mean and sample variance of the distances, $\{D_i\}_{i=1}^{M}$, of all M SNPs. Assuming that D_i's are i.i.d. with $\mathbb{E}[D_i] = \mu_0$, by the central limit theorem, we have

$$T_{\mu_0}(Y) = \frac{\overline{D} - \mu_0}{S/\sqrt{M}} \sim \mathcal{N}(0,1).$$

Under the null hypothesis that Y is in neither F nor G, we can assume $\mu_0 = 0$. Then

$$T(Y) = \frac{\overline{D}}{S/\sqrt{M}} \sim \mathcal{N}(0,1). \tag{2.1}$$

When we test the null hypothesis against the alternative hypothesis that Y is in either F or G, we reject the null hypothesis if $T(Y)$ is large in absolute value. For example, we reject the null hypothesis at the 95% significance level if $|T(Y)| > \Phi^{-1}(0.975)$, where Φ denote the CDF of the standard normal distribution. Similarly, when we test the null hypothesis against the alternative hypothesis that Y is in F, we reject the null hypothesis if $T(Y)$ is a large positive number.

By performing various experiments using publicly available human genetic data from *HapMap*[1], Homer et al. [14] showed that their method of inferring Y's membership status is highly effective. Homer et al. [14]'s experimental results were summarized as plots of the logarithm of p-values corresponding to all the individuals' t-statistics. We can see in Figure 3 of [14] that there is a clear separation between the p-values of individuals who are in F or G and those pertaining to individuals who are in neither F nor G. The experimental results of Homer et al. [14] therefore suggest that the attack is powerful, which allows an adversary to concede only a relatively small amount of accuracy of inferring Y's membership when Y belongs to F or G in order to reduce the chance of making a false inference on Y's membership when Y belongs to neither F nor G; in other

1 http://hapmap.ncbi.nlm.nih.gov/

words, controlling for the probability of making a wrong inference when Y belongs to neither F nor G, the probability that the adversary makes a correct inference when Y belongs to F or G is relatively high.

2.2.1 *A sensitivity analysis of Homer et al. [14]*

Braun et al. [3] argued that the key assumptions of the Homer et al. [14] attack are too stringent to be applicable in realistic settings. The most problematic assumptions are (i) that the SNPs are in linkage equilibrium and (ii) that the individual Y and the two DNA collections F and G are sampled from the same underlying population. Braun et al. [3] presented a sensitivity analysis of the key assumptions and showed that violation of the first assumption results in a substantial increase in variance and violation of the second assumption, together with the condition that F and G have different sample sizes, results in the distribution of the t-statistic deviating considerably from the standard normal distribution.

Here we describe the experiment carried out by Braun et al. [3] for their sensitivity analysis. Let F and G denote the cases and controls, respectively, in a case-control GWAS. Let W denote a group of individuals who have not participated in the GWAS but are ancestrally similar to those individuals in the GWAS. Randomly split F into subsets F_{in} and F_{out} so that $F = F_{in} \cup F_{out}$. Similarly, split G into subsets G_{in} and G_{out} randomly so that $G = G_{in} \cup G_{out}$. Now suppose that, instead of using F and G, we use F_{in} and G_{in} as cases and controls for the GWAS, then F_{out} and G_{out} consist of individuals who are sampled from the same underlying population as F_{in} and G_{in} but are in neither F_{in} nor G_{in}. For every individual in DNA collections F_{in}, F_{out}, G_{in}, G_{out}, and W, we calculate the Homer et al. [14] t-statistic with F_{in} as cases and G_{in} as controls. The distribution of the t-statistic of each DNA collection is then estimated and shown in Figure 2.1, a reproduction of Figure 1 in Braun et al. [3].

In Figure 2.1A and Figure 2.1B, we observe that, for every DNA collection, the distribution of the t-statistic spreads out more than the standard normal distribution does, which suggests that the sample variance S^2 underestimates the variance of the estimator $\overline{D}$ in Equation 2.1, which in turn is likely the result of linkage disequilibrium. We also observe that the means of the distributions for F_{out}, G_{out}, and W, which are collections of individuals who are in neither the case group (F_{in}) nor the control group (G_{in}), deviate from 0, which suggests that the expected value of the estimator $\overline{D}$ deviates from 0, which is likely the result of F and G not being from the same underlying population.

To examine the effect of linkage disequilibrium, Braun et al. [3] performed the same experiment on a random subset consisting of 50,000 SNPs, the size of which is about 1/9 of the original set of SNPs. In Figure 2.1C and 2.1D, we can clearly see that the variance of the distribution for each DNA collection is much closer to the variance of the standard normal distribution, which suggests that the sample variance is closer to the variance of the estimator for $\overline{D}$ when we reduce the effect of linkage disequilibrium. On the other hand, we can still observe that the means of the distributions for F_{out}, G_{out}, and W deviate from the mean of the standard normal distribution, which is likely the result of F and G not being from the same underlying population.

In Section 2.2.2, we perform analyses similar to Braun et al. [3]. We apply the Homer et al. [14] attack to the WTCCC data and demonstrate how the Homer et al. [14] attack is affected by the assumption that Y and the DNA collections F and G are from the same underlying population. To analyze the effect of linkage disequilibrium, not only do we carry out Braun et al. [3]'s experiment, but we also device a new experiment that ensures the cases and controls in the GWAS are indeed from the same underlying population and thus eliminates the potentially confounding effect due to the sampling assumption on the cases and controls.

Figure 2.1: A reproduction of Figure 1 in Braun et al. [3]

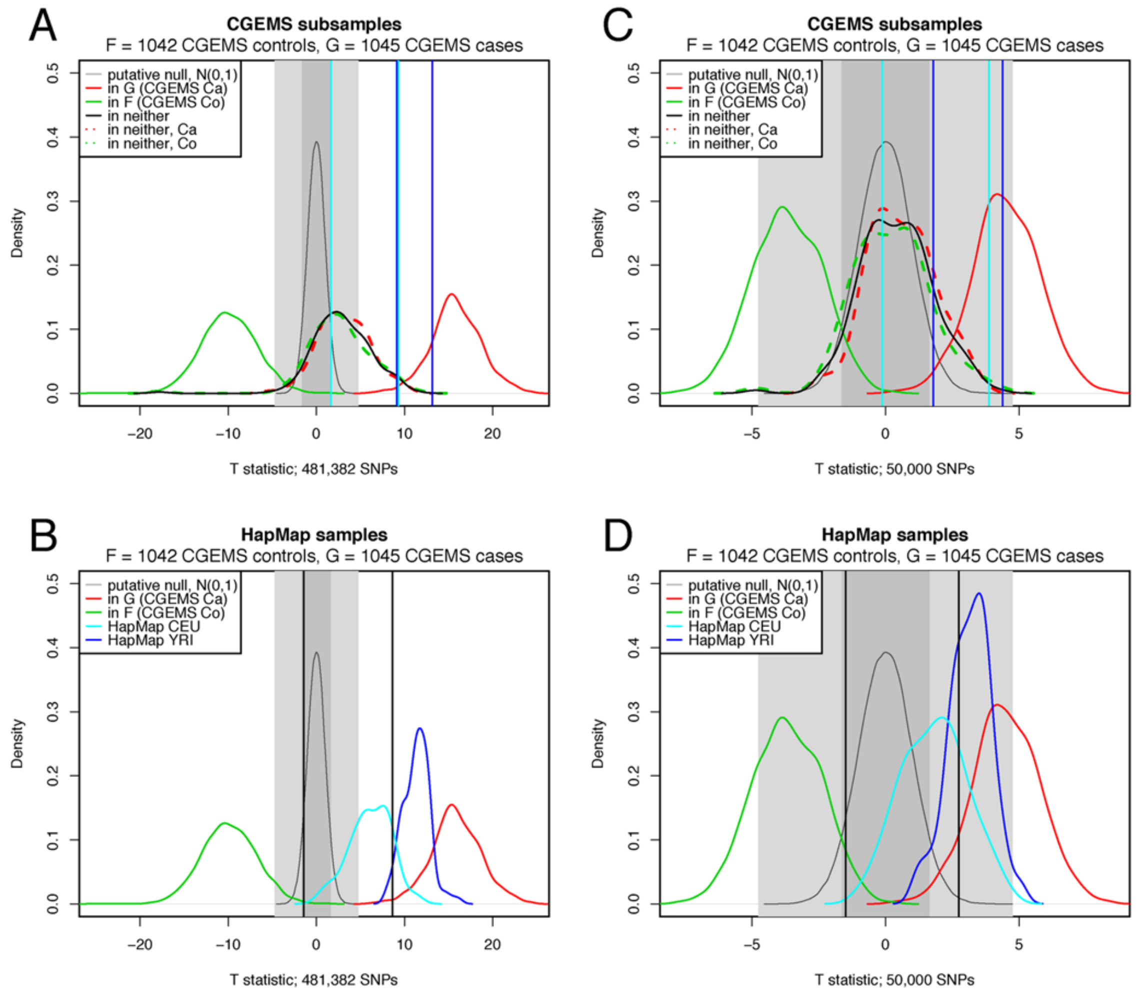


Figure 1. Comparison of *T* distributions. Comparison of T distributions for true positive and null samples versus putative null distribution, starting with 481,382 SNPs in (A,B) and 50,000 SNPs in (C,D). In all plots, true positive F (1042 CGEMS controls) is shown as a solid green curve, true positive G (1045 CGEMS cases) is shown as a solid red curve, and the putative null $N(0,1)$ is given as a thin grey curve. The dark and light grey regions represent the areas for which the null hypothesis would be accepted at $\alpha = 0.05$ and $\alpha = 10^{-6}$, respectively. In plots (A,C), CGEMS test samples in neither F nor G (100 CGEMS cases and 100 CGEMS controls) are given by a heavy black curve. The CGEMS case and CGEMS control distributions within this group are shown as dashed red and green lines, respectively. In plots (B,D), T distributions are given for HapMap CEPHs (cyan) and YRIs (blue). Vertical lines mark the 0.05 and 0.95 quantiles of the negative CGEMS samples (black), HapMap CEPHs (cyan), and HapMap YRIs (blue).
doi:10.1371/journal.pgen.1000668.g001

2.2.2 *Applying the Homer et al. [14] attack to the WTCCC data*

In this section, we first apply the Homer et al. [14] attack to the WTCCC data and perform the Braun et al. [3] experiment. We obtained results similar to Braun et al. [3], which confirms that the effectiveness of the Homer et al. [14] attack is affected by whether the SNPs are in linkage equilibrium and the assumption that F, G, and Y are from the same underlying distribution. We summarize our results in Figure 2.2, 2.3, and 2.4.

We further examine the implications and limitations of the Homer et al. [14] attack by devising a new experiment designed to satisfy the assumption that the cases and controls are from the same underlying population. In the new experiment, we divide the the controls' data into halves and let the two halves play the roles of cases and controls in a GWAS. Our new experiment allows us to show that, when F and G are from the same underlying distribution and Y is in neither F nor G, the expected value of the t-statistic of Y is close to 0, which confirms Homer et al. [14]'s claim that Y is on average equally far away from F and G when Y is in neither F nor G. Furthermore, by comparing results of our new experiment with results of the first experiment, in which we perform Braun et al. [3]'s analyses using the WTCCC data, we can delineate effects on the Homer et al. [14] attack imposed by linkage disequilibrium and the sampling assumption on the cases and controls have on the Homer et al. [14] attack. We summarize results of the new experiment in Figure 2.5, 2.6, and 2.7.

First experiment: The first experiment is set up in a fashion similar to Braun et al. [3].

In Wellcome Trust Case Control Consortium [33]'s GWAS of inflammatory bowel disease, the *IBD* dataset was used as cases and the *58C* and the *NBS* datasets were used as controls. In our experiment, we denote the *IBD* dataset by F, the *58C* by G, and the *NBS* dataset by W. We create a random subset, which we denote by F_{in}, containing 80% of the individuals in the *IBD* dataset and we treat F_{in} as cases in a GWAS. We denote the remaining 20% of individuals in *IBD* by F_{out} so that $F =$

$\mathsf{F}_{\mathrm{in}} \cup \mathsf{F}_{\mathrm{out}}$. Similarly, we denote a random subset containing 80% of the individuals in the *58C* dataset by G_{in} and we treat G_{in} as controls in a GWAS. We denote the remaining 20% of individuals in *58C* by $\mathsf{G}_{\mathrm{out}}$ so that $\mathsf{G} = \mathsf{G}_{\mathrm{in}} \cup \mathsf{G}_{\mathrm{out}}$. We then treat $\mathsf{F}_{\mathrm{out}}$, $\mathsf{G}_{\mathrm{out}}$ and *W* as as individuals who are sampled from the same underlying population as F_{in} and G_{in} but are in neither F_{in} nor G_{in}. We calculate the Homer et al. [14] t-statistic for each individual in each of the DNA collections F_{in}, G_{in}, $\mathsf{F}_{\mathrm{out}}$, $\mathsf{G}_{\mathrm{out}}$, and *W*. The distribution of the t-statistic of each DNA collection is then estimated and shown in Figure 2.2, 2.3, and 2.4.

In Figure 2.2, we calculate the t-statistic using all 415,196 SNPs. Figure 2.2 exhibits similar characteristics as Figure 2.1A and 2.1B, which are reproductions of Figure 1A and 1B in Braun et al. [3]: (i) the distribution of the t-statistic of any of the DNA collections spreads out more than the standard normal distribution does, suggesting that the sample variance S^2 underestimates the variance of the estimator $\overline{D}$, which is likely the result of linkage disequilibrium; (ii) the means of the distributions for $\mathsf{F}_{\mathrm{out}}$, $\mathsf{G}_{\mathrm{out}}$, and *W*, which are collections of individuals who are in neither the case group nor the control group, deviate from 0, suggesting that the expected value of the estimator $\overline{D}$ deviates from 0, which is likely the result of F and G not being from the same underlying population.

Figure 2.3 is comparable to Figure 2.1C and 2.1D, which are reproductions of Figure 1C and 1D in Braun et al. [3]. In Figure 2.3, we calculate each individual's t-statistic using a random subset consisting of 50,000 SNPs, the size of which is about $1/8$ of the full set of SNPs. We can observe in Figure 2.3 that, by using only a random subset of SNPs, which in effect reduces the effect of linkage disequilibrium, the sample variance is closer to the variance of the estimator $\overline{D}$ for each DNA collection.

To further reduce the effect of linkage disequilibrium, we calculate the t-statistics using only 1,000 SNPs, about $1/415$ as many SNPs as the full set of SNPs, and report the result in Figure 2.4. In Figure 2.4, distributions of the t-statistic for $\mathsf{G}_{\mathrm{out}}$ and *W* are almost indistinguishable from the distribution of $\mathtt{Normal}(0,1)$, which suggests that the variance of $\overline{D}$ can be properly estimated using the sample variance in the absence linkage disequilibrium.

By comparing Figure 2.2, 2.3, and 2.4, we observe that the deviations of the distributions for F_{in} and G_{in} from that of $\mathtt{Normal}(0,1)$ decreases as the number of SNPs used for calculating the t-statistic decreases. In particular, when the number of SNPs is small relative to the full set of SNPs, we observe in Figure 2.4 that there are substantial overlaps among distributions for F_{in}, G_{in}, and $\mathtt{Normal}(0,1)$. We can therefore conclude that the number of SNPs available for analysis affects the power of the Homer et al. [14] attack—that is, the more SNPs, the more the distribution of the t-statistic of F_{in} differs from the standard normal distribution, and thus the higher the probability that the Homer et al. [14] attack can correctly identify the membership of an individual when the individual belongs to F_{in}.

Second experiment: In the second experiment, we further examine how the assumption that the individual of interest, the cases, and the controls are sampled from the same underlying population affects the Homer et al. [14] attack. We divide the controls' data into halves and let the two halves play the roles of cases and controls in a GWAS, and therefore ensure that the cases and controls are sampled from the same underlying population. As a result, *58C* $= F_{in} \cup F_{out} \cup G_{in} \cup G_{out}$ and $W = \mathit{IBD} \cup \mathit{NBS}$ in this experiment. Similar to the first experiment, we obtain the distributions of the t-statistic using all 415,196 SNPs, a subset consisting of 50,000 SNPs, and a subset consisting of 1,000 SNPs. The results are summarized in Figure 2.5, 2.6, and 2.7, respectively.

Figure 2.5, 2.6, and 2.7 exhibit characteristics similar to those of the corresponding figures in the first experiment; the only noticeable difference is that in the second experiment, the means of the distributions for F_{out}, G_{out}, and W are very close to the mean of the null distribution $\mathtt{Normal}(0,1)$. The second experiment shows that when the cases and controls are in fact sampled from the same underlying population, we can expect that distributions of the t-statistic of F_{out}, G_{out}, and W are centered near 0. We can therefore conclude that in the first experiment, *IBD* and *58C*, which are treated as cases and controls in a GWAS, are not sampled from the same underlying population. Having deduced that individuals in *IBD* are not sampled from the same population as those in *58C* and observed that the mean of the distribution for W, which include individuals in *IBD*, is close to 0, we can conclude that, in order for the t-statistic to satisfy the $\mathrm{Normal}(0,1)$ null distribution, it is not necessary to assume that the individual of interest is sampled from the same underlying population as the cases and the controls. Therefore, we can relax the assumption that the individual of interest, the cases, and the controls are sampled from the same underlying population, and only require that the cases and the controls are sampled from the same underlying population.

To see how the assumption that the cases and the controls are sampled from the same underlying population affects the effectiveness of the Homer et al. [14] attack, let's consider the following analysis of Figure 2.3 from the first experiment and Figure 2.6 from the second experiment Following the Homer et al. [14] attack, which uses quantiles derived from $\mathrm{Normal}(0,1)$ to construct rejection regions, we can construct a rejection region that contains values smaller than the 5% quantile of $\mathrm{Normal}(0,1)$. In Figure 2.6 from the second experiment, in which the cases and the controls are indeed sampled from the same underlying population, the false positive rate, that is, the probability of falsely rejecting the hypothesis that an individual is in neither F_{in} nor G_{in} when the individual is indeed in neither F_{in} nor G_{in}, is higher than 5%, even though the expected false positive rate is 5%. On the other hand, in Figure 2.3 from the first experiment, in which, as we have concluded in this section, the assumption that the cases and the controls are sampled from the same underlying population does not hold, the false positive rate is not only higher than 5%, but it is also much higher than that of the second experiment, in which the assumption holds. We therefore conclude that, because in a real GWAS it is impossible to ensure that the cases and the controls are sampled from the same underlying population, the Homer et al. [14] attack has a high false positive rate and thus its applicability is limited.

2.3 Other attacks on genetic databases

2.3.1 *An extension of Homer et al. [14]*

Sampson and Zhao [26] attempted to relax Homer et al. [14]'s assumption that Y, F, and G are sampled from the same population by using an alternative method for estimating the allele frequencies in the samples, namely,

$$\hat{f}_i = f_i + \beta_i^F + \epsilon_i^F,$$

$$\hat{g}_i = g_i + \beta_i^G + \epsilon_i^G,$$

where β_i^F and β_i^G are SNP or platform specific biases for F and G, respectively, $\epsilon_i^F \sim \mathcal{N}(0, (\sigma_i^F id)^2)$, and $\epsilon_i^G \sim \mathcal{N}(0, (\sigma_i^G)^2)$. Under Sampson and Zhao [26]'s model, the distance metric becomes

$$D_i^{\ell_1}(Y) = |y_i - \hat{f}_i| - |y_i - \hat{g}_i|.$$

Sampson and Zhao [26] further proposed a new distance metric $D_i^{\ell_2}(Y) = (y_i - \hat{f}_i)^2 - (y_i - \hat{g}_i)^2$ and a new test statistic, $T_{\ell_2}^*$, that only requires the reference population to have the same ancestral structure as the individual of interest. Sampson and Zhao [26] showed that, by using $T_{\ell_2}^*$ instead of $D_i^{\ell_1}$ or $D_i^{\ell_2}$, not only can we make more robust inference, but we will also make the inference more powerful.

Figure 2.2: Density estimates of Homer et al. [14]'s t-statistic based on 415,196 SNPs for different groups of data. The dataset *IBD* is treated as cases, which we denote by F, and the dataset *58C* is treated as controls, which we denote by G, in a GWAS. The Homer et al. [14]'s t-statistics are calculated based on 80% random samples of the cases and controls, which we call F_{in} and G_{in}. Cases and controls not in F_{in} and G_{in} are denoted by F_{out} and G_{out}, respectively. Lastly, we denote the *NBS* dataset by W. Under the framework of Braun et al. [3], F_{out}, G_{out} and W consist of individuals who are sampled from the same underlying population as F_{in} and G_{in} but are in neither F_{in} nor G_{in}.

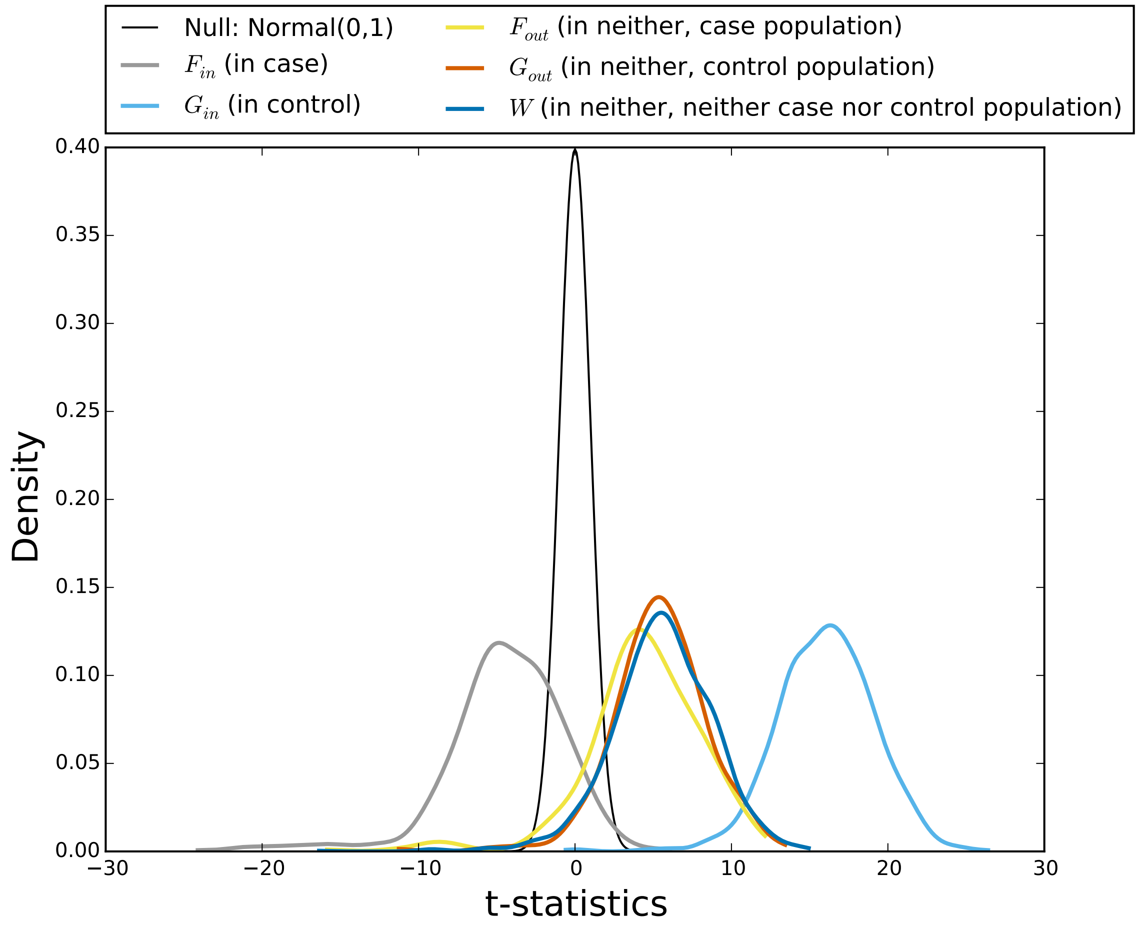


Figure 2.3: The same as Figure 2.2 except that only 50,000 randomly sampled SNPs are used to calculate the t-statistic.

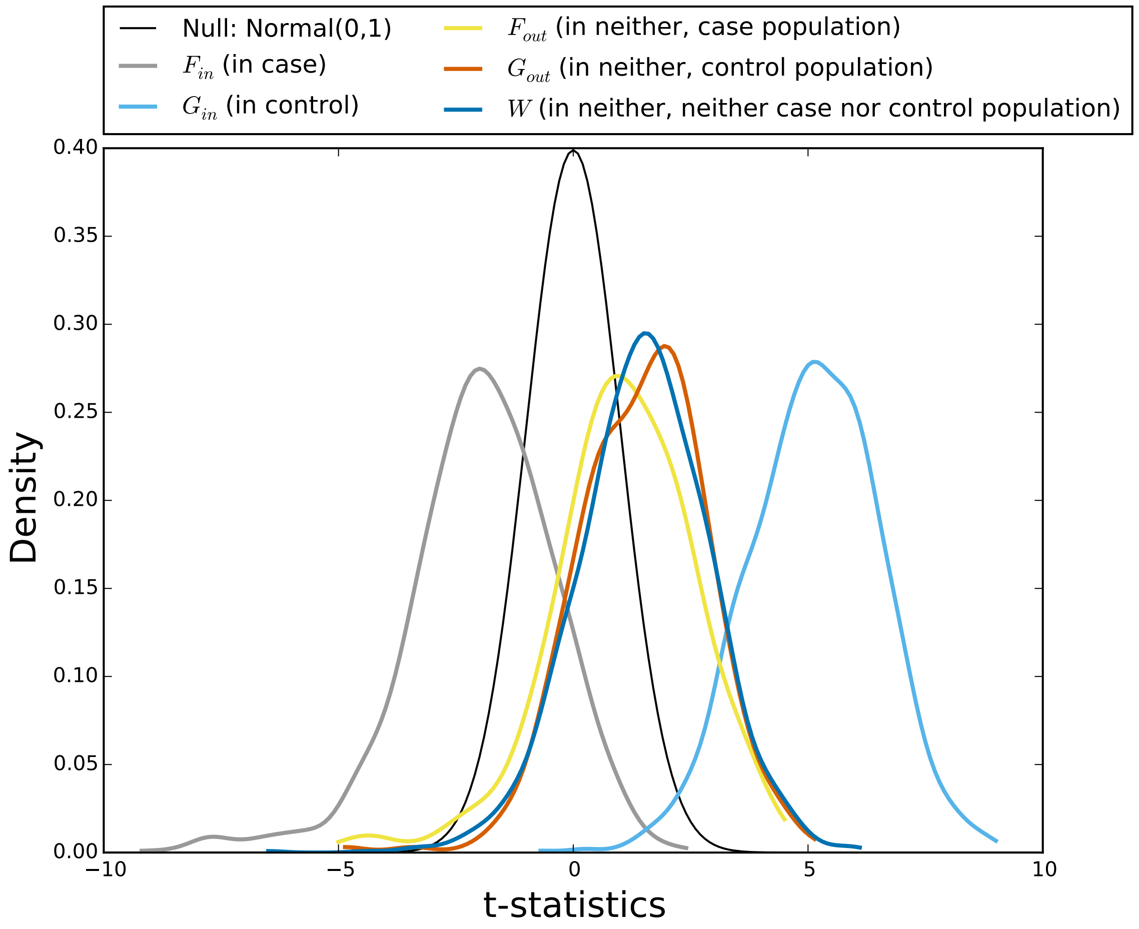


Figure 2.4: The same as Figure 2.2 except that only 1,000 randomly sampled SNPs are used to calculate the t-statistic.

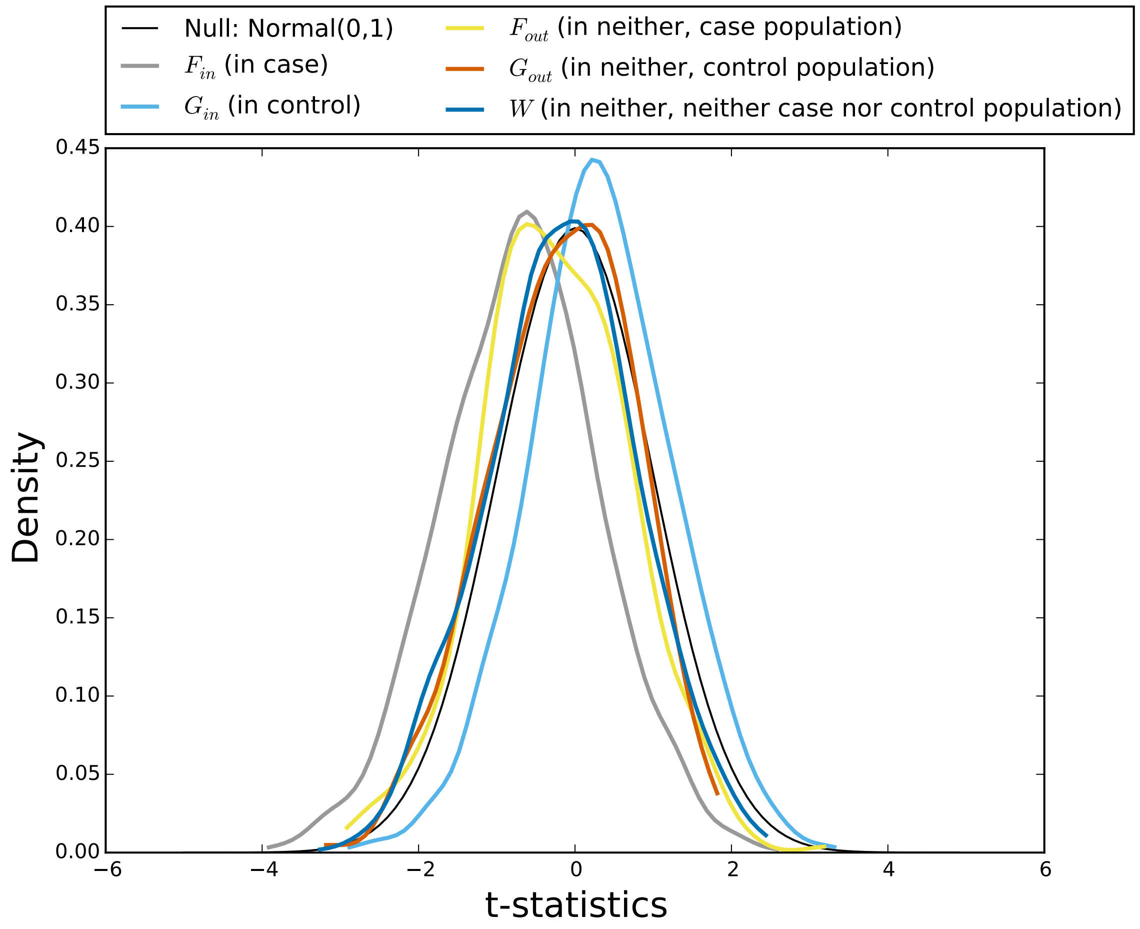


Figure 2.5: Density estimates of Homer et al. [14]'s t-statistic based on 415,196 SNPs for different groups of data. The dataset *58C* is divided into halves; one half is used as cases and the other half is used as controls in a GWAS. Similar to Figure 2.2 , we denote the cases by F and the controls by G. We calculate the Homer et al. [14]'s t-statistics based on 80% random samples of the cases and controls, which we call F_{in} and G_{in}. Cases and controls not in F_{in} and G_{in} are denoted by F_{out} and G_{out}, respectively. Lastly, we denote the *IBD* and *NBS* datasets by W. Under the framework of Braun et al. [3], F_{out}, G_{out} and W consist of individuals who are sampled from the same underlying population as F_{in} and G_{in} but are in neither F_{in} nor G_{in}. Indeed, by splitting the dataset *58C* into F_{in}, G_{in}, F_{out}, and G_{out}, we ensure that they are from the same underlying population.

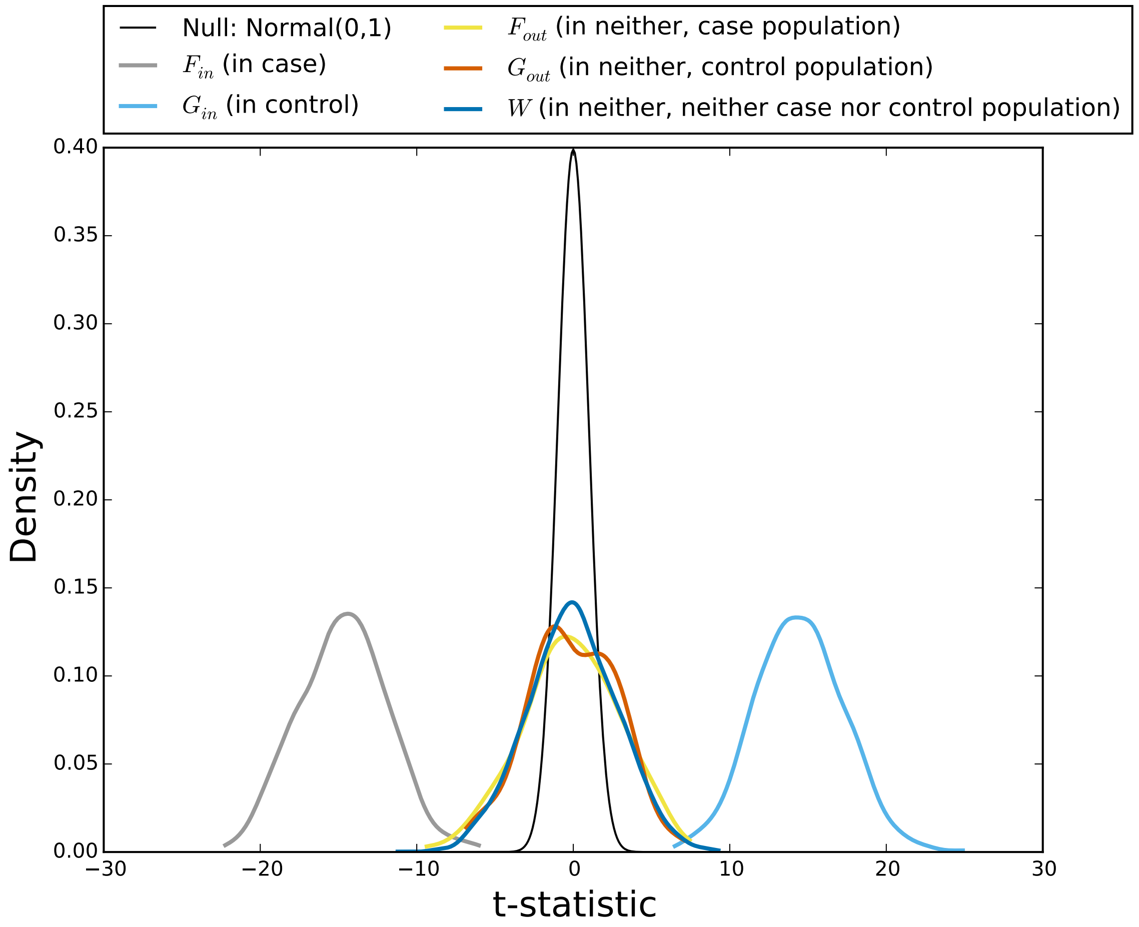


Figure 2.6: The same as Figure 2.5 except that only 50,000 randomly sampled SNPs are used to calculate the t-statistic.

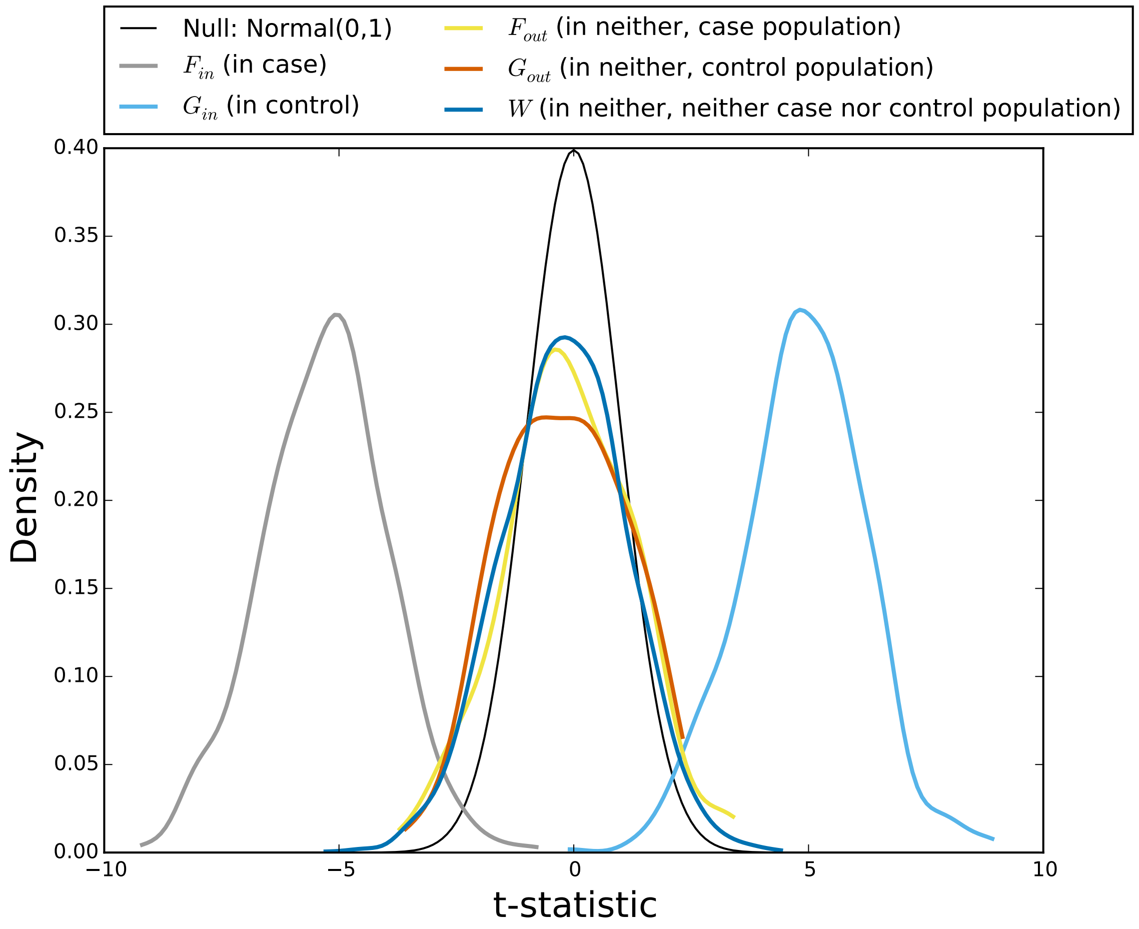
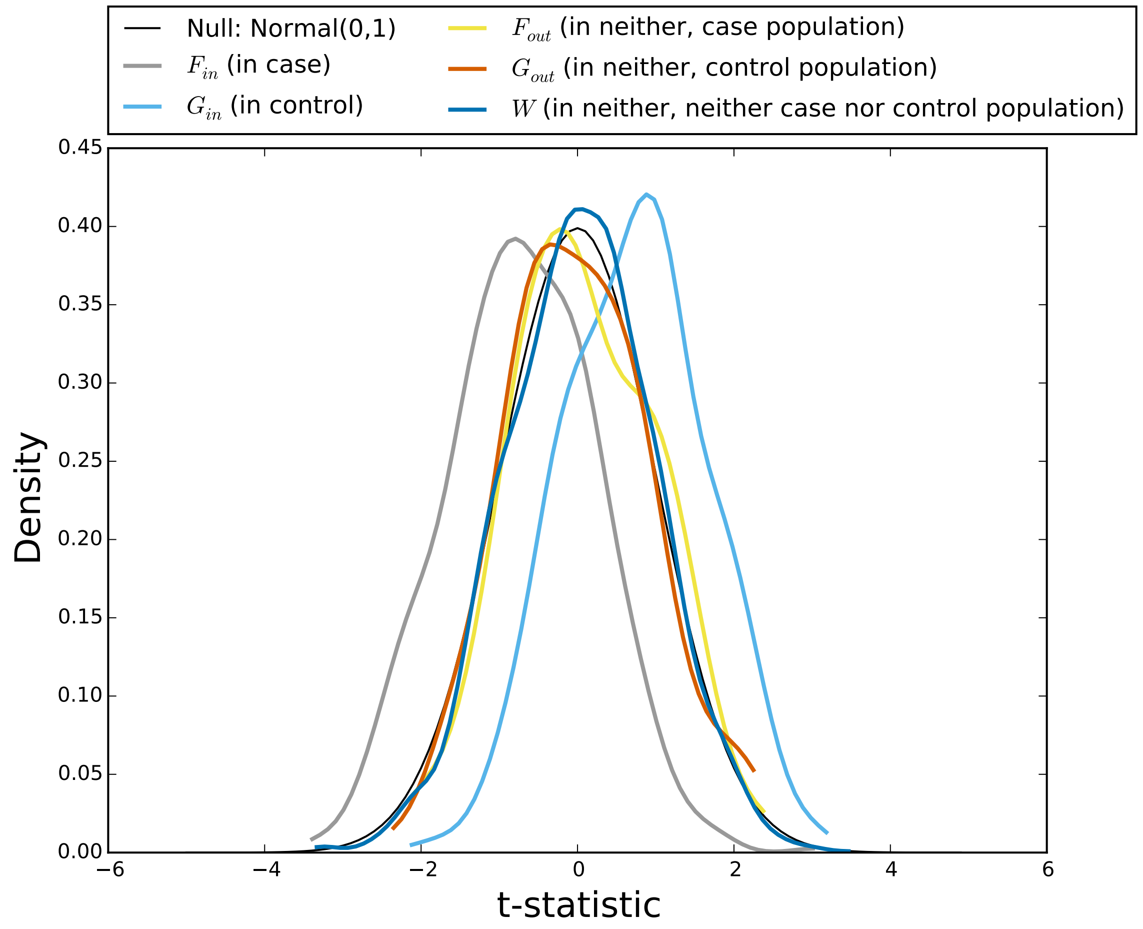


Figure 2.7: The same as Figure 2.5 except that only 1,000 randomly sampled SNPs are used to calculate the t-statistic.



2.3.2 *Likelihood ratio approaches*

Jacobs et al. [16] proposed a likelihood ratio approach to deriving a statistic for inferring the membership status of Y using minor allele frequencies. In Jacobs et al. [16], the membership status of Y is translated into four hypotheses: (1) H_0: Y is in neither F nor G; (2) H_1: Y is in F but not in G; (3) H_2: Y is in G but not in F; and (4) H_3: Y is in both F and G.

Define $r_i := \log L(y_i|H_*, F) - \log L(y_i|H_*, G)$, where $L(y_i|H_*, X)$ is the likelihood of individual Y having genotype y_i at SNP$_i$ given the DNA collection X under the hypothesis H_*. Jacobs et al. [16] treated the log-likelihood ratio r_i as the distance metric and defined the distance across all M SNPs by

$$d = \sum_{i=1}^{M} r_i = \sum_{i=1}^{M} \left[\log L(y_i|H_*, F) - \log L(y_i|H_*, G)\right].$$

Note that if we assume all the loci are in linkage equilibrium, then d becomes the log-likelihood ratio of observing Y given F or G under the hypothesis H_*:

$$d = \log\left(\prod_{i=1}^{M} \frac{L(y_i|H_*, F)}{L(y_i|H_*, G)}\right) = \log\left(\frac{L(Y|H_*, F)}{L(Y|H_*, G)}\right).$$

Sankararaman et al. [27] and Visscher and Hill [30]'s likelihood ratio attacks are formulated differently from that of [16]: instead of contrasting the similarity between Y and F and the similarity between Y and G, Sankararaman et al. [27] and Visscher and Hill [30] compared Y against F, assuming that Y and F are sampled from the same underlying population $\mathcal{P}$ and that the allele frequencies of $\mathcal{P}$ can be estimated from a representative DNA collection G. In Sankararaman et al. [27] and Visscher and Hill [30]'s frameworks, the null hypothesis then becomes H_0: Y and F are independently sampled from $\mathcal{P}$, and the alternative hypothesis becomes H_A: F consists of Y and $(N_F - 1)$ other individuals sampled from $\mathcal{P}$.

Jacobs et al. [16], Sankararaman et al. [27], and Visscher and Hill [30]'s attacks have been shown to be more powerful than the Homer et al. [14] attack, allowing us to make more accurate inference while we control for the false positive rate.

2.3.3 *A Bayesian approach*

The Homer et al. [14] attack is a frequentist approach. Clayton [7] provided a Bayesian interpretation of the problem of inferring whether Y is in F or not. The Bayes factor is

$$\log(\text{Bayes factor}) = \log \frac{L(y, f \mid H_1)}{L(y, f \mid H_0)},$$

where $H_0 : Y \notin F$ and $H_1 : Y \in F$. In Clayton [7]'s setting, $L(y, f \mid H_0) = L(f)L(y)$ and $L(y, f \mid H_1) = L(f|y, H_1)L(y)$. It then follows that

$$\log(\text{Bayes factor}) = \log \frac{L(y, f \mid H_1)}{L(y, f \mid H_0)} = \log\{L(f|y, H_1) - L(f)\}.$$

By appealing to the normal approximation for bi-allelic SNPs, Clayton [7] showed that, given μ, the population minor allele frequencies for the SNPs, and Σ, the covariance matrix, the Bayes factor can be written as

$$\log(\text{Bayes factor}) = \frac{M}{2} \log \frac{N_F}{N_F - 1} - \frac{1}{2} \left\{ \frac{N_F}{N_F - 1} (y - f)^T \Sigma^{-1} (y - f) - (y - \mu)^T \Sigma^{-1} (y - \mu) \right\}.$$

Moreover, Clayton [7] extended the method and considered situations in which (1) μ is known, (2) an informative prior is imposed on μ, and (3) an informative prior is imposed on μ, the mean of which comes from a hyperprior distribution.

By formulating the attack in a Bayesian framework, Clayton [7] was able to incorporate prior information (e.g., μ or a hyperprior on μ) into the attack, making the attack more flexible and potentially more powerful.

2.3.4 *Regression approaches*

To make inference on whether an individual Y belongs to the DNA collection F, Visscher and Hill [30] analyzed the regression coefficient β in the simple linear regression model

$$(y_i - p_i) = \alpha + \beta(\hat{p}_i - p_i) + \epsilon_i,$$

where f_i is the allele frequency of Y at SNP i, p_i is the population allele frequency at SNP i, and $\hat{p}_i$ is the allele frequency of the collection F at SNP i. Visscher and Hill [30] constructed the statistic $t = (\beta - 1)^2/Var(\beta)$ and argued that $(t \mid H_0)$ is distributed as $\chi^2_{(1)}$ and $(t \mid H_1)$ is distributed as $\chi^2_{(1),\lambda}$, where H_0 is the null hypothesis that $Y \notin F$, H_1 is the alternative hypothesis that $Y \in F$, and λ is a non-centrality parameter for the χ^2 distribution.

Masca et al. [22] suggested that the regression model in Visscher and Hill [30] is unlikely to be correct because the dependent variable y_i is discrete before it is centered around p_i. Masca et al. [22] therefore proposed the use of the logistic regression model

$$\log\left(\frac{p_i}{1-p_i}\right) = \log\left(\frac{\hat{p}_i^{**}}{1-\hat{p}_i^{**}}\right) + \beta(\hat{p}_i - \hat{p}_i^{**}),$$

where $\hat{p}_i^{**}$ is an estimate of the population minor allele frequency at SNP i, $p_i = E[f_i|\hat{p}_i, \hat{p}_i^{**}]$, and assumed that $2f_i \sim \text{Bin}(2, p_i)$. Masca et al. [22] also advocated the use of generalized estimation equation as an alternative to the logistic regression model to account for linkage disequilibrium.

Im et al. [15] proposed another attack that uses regression coefficients produced in a GWAS. Suppose that F and G are the cases and controls, respectively, in a GWAS. Im et al. [15] defined the statistic

$$\hat{T}_Z = \frac{N_F}{M}\sum_{j=1}^{M}\hat{\beta}_j\left(X_{Z,j} - \overline{X}_j^G\right),$$

where $X_{Z,j} \in \{0, 1, 2\}$ is the genotype of Individual Z at SNP j, $\overline{X}_j^G$ is the mean allelic dosage (twice the allele frequency) of DNA samples in G, and $\hat{\beta}_j$ is the coefficient estimate of the simple linear regression model $W_i = \alpha_j + \beta_j X_{i,j} + \varepsilon_i$ for SNP j, where W_i is the phenotype of Individual i, with i indexing all DNA samples in F and G. Let μ and σ^2 denote the population mean and variance of the phenotype. Im et al. [15] argued that the means and variances of $\hat{T}_Y$ under the hypotheses $H_0 : Y \notin F$ and $H_1 : Y \in F$ are

$$\begin{aligned} E\left[\,\hat{T}_Y \,|\, X_Y, W_Y, H_0\,\right] &= 0, \qquad Var\left(\,\hat{T}_Y \,|\, X_Y, W_Y, H_0\,\right) = \sigma^2 \frac{N_F}{M}, \\ E\left[\,\hat{T}_Y \,|\, X_Y, W_Y, H_1\,\right] &= W_Y - \mu, \qquad Var\left(\,\hat{T}_Y \,|\, X_Y, W_Y, H_1\,\right) = \sigma^2 \frac{N_F}{M}. \end{aligned}$$

Lumley and Rice [20] also used regression results to construct an attack that breaches the study participants' privacy. In contrast to attacks in [30, 22, 15], which aimed to identify whether an individual belongs to a DNA collection or not, the Lumley and Rice [20] attack focuses on predicting a study participant's disease status. The results in Lumley and Rice [20] suggest that the attack has very high sensitivity and specificity.

2.3.5 *Imputation and recovery approaches*

Wen and Stephens [35] considered genetic data privacy risk in terms of an attacker's ability to accurately impute allele frequencies in a private database using a public database. Wen and Stephens [35]'s attack takes advantage of partially known genetic information about an individual and the correlations between SNPs to impute genetic information about the individual that is unknown to an intruder. Wen and Stephens [35] argued that, given a reference sample, it is possible to predict an individual's allele frequencies unknown to the attacker using the following linear predictor:

$$\hat{f}_u^{pred} = \hat{\mu}_u + \hat{\Sigma}_{ut} \left(\hat{\Sigma}_{tt} + \frac{\epsilon^2}{\sigma^2} \mathbb{I} \right)^{-1} (f_t^{obs} - \hat{\mu}_t),$$

where $\hat{f}_u^{pred}$ and f_t^{obs} are predicted and known, respectively, allele frequencies of an individual, $\hat{\mu}_u$ and $\hat{\mu}_t$ are estimated allele frequencies of the reference sample at SNPs that are unknown and known, respectively, to the attacker, $\hat{\Sigma}_{tt}$ is the estimated covariance matrix of allele frequencies among SNPs known to the attacker, and $\hat{\Sigma}_{ut}$ is the estimated covariance matrix of allele frequencies between SNPs known and unknown to the attacker, ϵ is the measurement error, and σ^2 is an over-dispersion parameter.

Wang et al. [31] and Zhou et al. [42] proposed an integer-programming attack that recovers the individuals' haplotypes using the r^2 statistic (see [13] for the definition of r^2 statistic), which is sometimes released in GWAS results to indicate linkage disequilibrium between a pair of SNPs. Having recovered the haplotypes, Wang et al. [31] and Zhou et al. [42]'s attack proceeds to derive the signs of r in the r^2 statistics. Wang et al. [31] and Zhou et al. [42] proposed a new statistic, T_r, for testing the null hypothesis that Y is not in the case group F against the alternative hypothesis that Y is in the case group F. T_r is defined as

$$T_r = \sum_{1 \leqslant i \leqslant j \leqslant M} \left(r_{ij}^F - r_{ij}^G\right)\left(Y_{ij}^{00} + Y_{ij}^{11} - Y_{ij}^{01} - Y_{ij}^{10}\right),$$

where r_{ij}^F and r_{ij}^G are the signed r statistic between SNP i and SNP j in F and G, respectively, $Y_{ij}^{ab} = 1$ if the individual Y's haplotype is a at SNP i and b at SNP j, and $Y_{ij}^{ab} = 1$ otherwise. Wang et al. [31] and Zhou et al. [42] argued that T_r is a more powerful test statistic than the one proposed by Homer et al. [14] because T_r takes advantage of the additional information on linkage disequilibrium. However, the accuracy of the linkage disequilibrium information depends heavily on the choice of the control group, and it may be unrealistic to assume that the control group matches perfectly with the case group. It is unclear how the power of the test degrades as the accuracy of the linkage disequilibrium information degrades. Furthermore, in order to evaluate the applicability of the attack, it would be useful to see the false positive rate of haplotype recovery—that is, the probability of the haplotype recovery algorithm giving a small p-value when the candidate haplotype is not in the case group.

2.3.6 *An attack using metadata*

Gymrek et al. [12] demonstrated how one can recover an individual's surname by analyzing the individual's short tandem repeats on the Y chromosome (Y-STRs) and combining the recovered surname with various types of metadata, such as year of birth and state of residency, to determine the identity of the individual in question. The surname recovery attack begins with querying public genetic genealogy databases to obtain a list of candidate surnames. Then a confidence score is assigned to each candidate surname according to a statistical model. Those surnames whose scores pass a threshold value will be retained. The results of the surname recovery attack show that the attack can only effectively recover rare surnames, which suggests that a simple way of guarding against this attack is to remove rare surnames from the database; nevertheless, Gymrek et al. [12]'s attack raises privacy concerns for releasing data from genetic genealogy services.

Part II

DIFFERENTIAL PRIVACY

3

DIFFERENTIAL PRIVACY

Traditional statistical methods for confidentiality and privacy protection of statistical databases do not scale well to deal with GWAS data, especially in terms of guarantees regarding protection from linkage to external information. The concept of *differential privacy*, recently introduced by the cryptographic community (e.g., Dwork et al. [10]), is an approach that provides a rigorous definition of privacy with meaningful privacy guarantees in the presence of arbitrary external information, although the guarantees may come at a serious price in terms of data utility. In this chapter, we start by reviewing definitions and notation related to differential privacy. In Section 3.2, we describe techniques for constructing differentially private algorithms. In Section 3.3, we present a toy example of differential privacy.

3.1 Definitions and notation

Let $\mathcal{D} = \{(X_1, \dots, X_n) : X_i \sim \mathcal{P}\}$ denote the set of all databases that consist of n individuals sampled independently from the same population $\mathcal{P}$. For $D, D' \in \mathcal{D}$, write $D \sim D'$ if D and D' differ in one individual; that is, there exists $i \in \{1, \dots, n\}$ such that $X_i \neq X'_i$ and $(\forall j \neq i) X_j = X'_j$, where $D = (X_1, \dots, X_n)$ and $D' = (X'_1, \dots, X'_n)$.

Definition 3.1 (Differential privacy). *A randomized mechanism $\mathcal{K}$ is ϵ-differentially private if, for all $D, D' \in \mathcal{D}$ such that $D \sim D'$ and for any measurable set $S \subset \mathbb{R}$,*

$$\frac{\Pr(\mathcal{K}(D) \in S)}{\Pr(\mathcal{K}(D') \in S)} \leqslant e^{\epsilon}.$$

$\mathcal{K}$ is (ϵ, δ)-differentially private *if, for all $D, D' \in \mathcal{D}$ such that $D \sim D'$ and for any measurable set $S \subset \mathbb{R}$,*

$$\Pr(\mathcal{K}(D') \in S) \leqslant e^{\epsilon} \Pr(\mathcal{K}(D) \in S) + \delta.$$

We get ϵ-differential privacy from (ϵ, δ)-differential privacy by setting $\delta = 0$. In (ϵ, δ)-differential privacy, δ can be thought of as the probability that an algorithm fails to satisfy ϵ-differential privacy. By allowing differential private algorithms to have a small chance of failure, we can sometimes construct algorithms that have much higher statistical utility than those that strictly enforce differential privacy. Meanwhile, ϵ serves as a tuning parameter for controlling how much privacy is preserved—the smaller ϵ is, the higher the level of privacy protection is guaranteed. In most cases increasing ϵ increases the statistical utility of the data, as the level of privacy protection decreases; however, we can see in Section 4.5.3 that in certain applications increasing ϵ doesn't increase statistical utility.

A statistical interpretation of differential privacy was given by Wasserman and Zhou [32] in the context of hypothesis testing. Suppose that an adversary has been given information about the data-generating distribution $\mathcal{P}$, the ϵ-differentially private randomized algorithm $\mathcal{K}(\cdot)$, and an instance of the output of $\mathcal{K}$, which we denote by Z. Wasserman and Zhou [32] showed that the power of any level α test of $H_0 : X_i = s$ versus $H_1 : X_i = t, t \neq s$ that is a functionof Z, $\mathcal{K}$, and $\mathcal{P}$ is bounded above by αe^{ϵ}. Furthermore, Wasserman and Zhou [32] pointed out that the Bayes factor of a Bayes test between H_0 and H_1 is bounded by $e^{-2\epsilon}$ and $e^{2\epsilon}$.

3.2 Techniques for constructing differentially private algorithms

Given a nonrandom function that takes a database as input, we want to design an mechanism that perturbs the output of the function so that the input database will be protected under the framework of differential privacy. Designing differentially private mechanisms often require knowledge of the targeted function's *sensitivity* , which is a notion of how much the output of the targeted function would change if one record in the input database were changed.

Definition 3.2 (Sensitivity)**.** *The* sensitivity *of a nonrandom function* $f : \mathcal{D} \times \mathbb{R}^d \to \mathbb{R}$ *is the smallest number* $S(f)$ *such that*

$$\sup_{x \in \mathbb{R}^d} |f(D, x) - f(D', x)| \leqslant S(f),$$

for all databases $D, D' \in \mathcal{D}$ *such that* $D \sim D'$*.*

Differentially private algorithms are sometimes complex; for example, Algorithm 1 and Algorithm 2 both consist of multiple steps. Two methods are often used as building blocks for constructing complex differentially private algorithms. One of the methods, due to Dwork et al. [10], is the *Laplace mechanism* (Definition 3.3). The Laplace mechanism generates differentially private results by adding random Laplace noise to the output of the original function. The other method, due to McSherry and Talwar [24], is the *exponential mechanism* (Definition 3.4). The exponential mechanism achieves differential privacy by randomly sampling from a distribution for which each possible outcome is weighted by a score function.

Definition 3.3 (Laplace mechanism)**.** *Releasing* $f(D) + b$*, where* $b \sim$ *Laplace* $\left(0, \frac{S(f)}{\epsilon}\right)$*, satisfies the definition of* ϵ*-differential privacy.*

To see that the Laplace mechanism is differentially private, consider the following. Given $\epsilon > 0$, let b and b' be independent Laplace random variables with $b, b' \sim \text{Laplace}\left(0, \frac{S(f)}{\epsilon}\right)$. Then for any $D \sim D'$,

$$\begin{aligned}
\frac{\mathbb{P}(f(D) + b = x)}{\mathbb{P}(f(D') + b' = x)} &= \frac{\exp\left(\frac{-|x - f(D)|}{S(f)/\epsilon}\right)}{\exp\left(\frac{-|x - f(D')|}{S(f)/\epsilon}\right)} \\
&\leqslant \exp\left(\frac{|f(D') - f(D)|}{S(f)/\epsilon}\right) \\
&\leqslant \exp(\epsilon).
\end{aligned}$$

Definition 3.4 (Exponential mechanism)**.** *Let* $q : \mathcal{D} \times \mathbb{R}^d \to \mathbb{R}$ *be a score function that assigns a score to an outcome* $x \in \mathbb{R}$ *given a database* $D \in \mathcal{D}$*. Define the random variable* ε_q^ϵ *by*

$$\mathbb{P}\left(\varepsilon_q^\epsilon(D) = x\right) = \frac{\exp\left(\frac{\epsilon q(D,x)}{2 S(q)}\right)}{\int_{\mathbb{R}^d} \exp\left(\frac{\epsilon q(D,t)}{2 S(q)}\right) dt}.$$

Then releasing ε_q^ϵ *satisfies the definition of* ϵ*-differential privacy.*

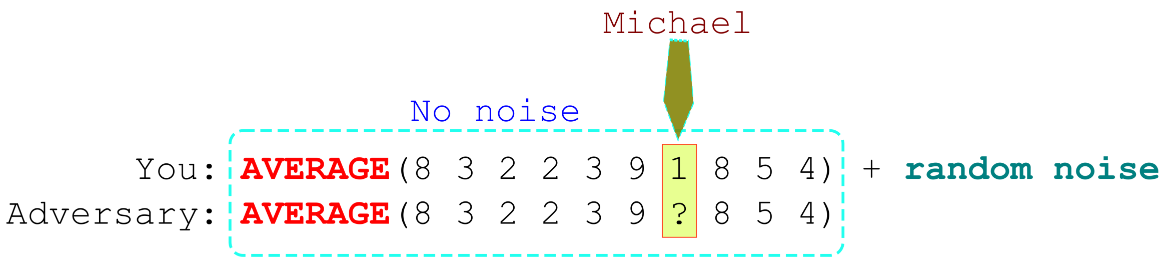
The exponential mechanism is differentially private because for a given ϵ and for any $D \sim D'$,

$$\begin{aligned}
\mathbb{P}\left(\varepsilon_q^\epsilon(D) = x\right) &= \frac{\exp\left(\frac{\epsilon q(D,x)}{2 S(q)}\right)}{\int_{\mathbb{R}^d} \exp\left(\frac{\epsilon q(D,t)}{2 S(q)}\right) dt} \\
&\leqslant \frac{\exp\left(\frac{\epsilon [q(D',x) + S(q)]}{2 S(q)}\right)}{\int_{\mathbb{R}^d} \exp\left(\frac{\epsilon [q(D',t) - S(q)]}{2 S(q)}\right) dt} \\
&= \frac{\exp\left(\frac{\epsilon}{2}\right) \exp\left(\frac{\epsilon q(D',x)}{2 S(q)}\right)}{\int_{\mathbb{R}^d} \exp\left(-\frac{\epsilon}{2}\right) \exp\left(\frac{\epsilon q(D',t)}{2 S(q)}\right) dt} \\
&= \exp(\epsilon) \mathbb{P}\left(\varepsilon_q^\epsilon(D') = x\right).
\end{aligned}$$

3.3 A toy example of differential privacy

The following toy example illustrates how differential privacy can be used to solve privacy protection problems in real life. Suppose that you teach a class with 10 seventh graders and the students have been given an exam. The exam is graded from 0 to 10. You want to let the parents know how well the whole class do in the exam by telling them the average score. Unfortunately, a resourceful adversary of yours, with an intention unknown to you, has obtained the scores of all of the students in the class except for one student's, whose name happens to be Michael[1]. If you were to release the average score without adding any noise, then you would in effect reveal Michael's score to your adversary. The difficult situation you are facing is depicted in Figure 3.1. How can you release the average score without compromising Michael's privacy?

Figure 3.1: A toy example of differential privacy.


Michael
No noise
You: AVERAGE(8 3 2 2 3 9 1 8 5 4) + random noise
Adversary: AVERAGE(8 3 2 2 3 9 ? 8 5 4)

Differential privacy can be applied in this toy example. Differential privacy requires that given two databases that differ by one record, the probabilities of a random function outputting the same result given either database as input differ by a factor no larger than e^{ϵ}. In the toy example, if you add Laplace noise with mean 0 and scale $\frac{1}{\epsilon}$ to the average score, then the resulting perturbed average score will conform with the definition of ϵ-differential privacy due to the Laplace mechanism (Definition 3.3), where the sensitivity of the unperturbed average score, μ, is 1:

$$\max\left\{\left|\mu - \frac{10\mu - x_{\text{unknown}} + 10}{10}\right|, \left|\mu - \frac{10\mu - x_{\text{unknown}} + 0}{10}\right|\right\} \leqslant 1,$$

where $x_{\text{unknown}} \in \{0, 1, \dots, 10\}$ is any possible score. As a result, your adversary will not be able to infer Michael's score using the perturbed average score that you release.

1 http://www.ssa.gov/oact/babynames/top5names.html

Part III

PRIVACY-PRESERVING DATA RELEASE METHODS

4

PRIVACY PRESERVING RELEASE OF THE MOST RELEVANT SNPS

In this chapter, we examine methods for privacy preserving release of the most relevant SNPs in a genome-wide association study (GWAS). We review two general-purpose algorithms for releasing a subset from a large set of items differentially privately in Section 4.2. One of the algorithms (Algorithm 1) is based on the Laplace mechanism (Definition 3.3) and the other algorithm (Algorithm 2) is based on the exponential mechanism (Definition 3.4). Both algorithms require a score function. With a focus on GWAS's, we describe in details two score functions: χ^2 statistic and Hamming distance score. We explain how to calculate the sensitivity of χ^2 statistic in Section 4.3 and how to calculate Hamming distance score in Section 4.4. Lastly, in Section 4.5, we apply the differentially private release methods to a real human GWAS dataset and analyze how well the methods preserve statistical utility while they protect data privacy.

4.1 Introduction

In a typical GWAS setting, the main goal of the study is to identify genetic variations that are associated with a phenotype. The phenotype of interest is often a disease, and genetic variations are usually described by singlue-nucleotide polymorphisms (SNPs). A GWAS database usually contains information on hundreds of thousands of SNPs from thousands of individuals.

Table 4.1: Genotype table comparing the frequencies of each genotype (0, 1, or 2) in the case group and the control group.

	# of minor alleles			
	0	1	2	Total
Case	r_0	r_1	r_2	R
Control	s_0	s_1	s_2	S
Total	n_0	n_1	n_2	N

One of the most commonly used measure of association between a phenotype and a single SNP is Pearson's χ^2 statistic: the larger the χ^2 statistic, the stronger the association. To calculate the χ^2 statistic for a single SNP, we first need to summarize the data for the SNP in the form of a contingency table. Following the notation in Devlin and Roeder [9], we can summarize the data for a single SNP in a case-control study with R cases and S controls using a 2×2 allelic table shown in Table 4.2 or a 2×3 genotype table shown in Table 4.1. We require the margins of a contingency table to be positive.

As was pointed out by Zheng et al. [41], researchers often use the χ^2 statistic based on a SNP's 2×2 allelic table to measure association when the genetic model of the phenotype is additive, and the χ^2 statistic based on the SNP's 2×3 genotype table when the genetic model is unknown. These χ^2 statistics are written as follows:

χ^2 statistic based on a 2×2 allelic table:

$$Y_A = \frac{2N}{RS(n_1+2n_2)(2n_0+n_1)}\left[N(s_1+2s_2) - S(n_1+2n_2)\right]^2,$$

χ^2 statistic based on a 2×3 genotype table:

$$Y = \frac{(r_0N - n_0R)^2}{n_0RS} + \frac{(r_1N - n_1R)^2}{n_1RS} + \frac{(r_2N - n_2R)^2}{n_2RS}.$$

Table 4.2: Allelic table comparing the frequencies of minor allele and major allele in the case group and the control group. See Table 4.1 for definitions of R, S, N, r_i, s_i, and n_i for $i \in \{0, 1, 2\}$.

	Allele type		
	Minor	Major	Total
Case	$r_1 + 2r_2$	$2r_0 + r_1$	2R
Control	$s_1 + 2s_2$	$2s_0 + s_1$	2S
Total	$n_1 + 2n_2$	$2n_0 + n_1$	2N

4.2 Differentially private methods

Algorithm 1 and Algorithm 2 are differentially private algorithms for releasing the most relevant SNPs from a large set of candidate SNPs, in which every SNP is assigned a score that indicates relevance of the SNP. The basis of Algorithm 1 is the Laplace mechanism (Definition 3.3), and the basis of Algorithm 2 is the exponential mechanism (Definition 3.4). Though Algorithm 1 and Algorithm 2 are stated in terms of SNPs, they can be generalized to release the most relevant subset from a large set, of which every element is assigned a score indicating relevance of the element.

Theorem 4.1. *Algorithm 1 is ϵ-differentially private.*

Proof. See the proof of Algorithm 1 in Uhler et al. [29]. □

Theorem 4.2. *Algorithm 2 is ϵ-differentially private.*

Proof. See Appendix A.1.1.1. □

It is a common practice in GWAS's to use single-SNP association tests to identify SNPs that are most relevant to a disease. Researchers often use the χ^2 statistic based on the genotype table or the allelic table of a SNP as a measure of relevance. Usually, only

Algorithm 1 [2, 29, 37] ϵ-differentially private algorithm for releasing the K most relevant SNPs using the *Laplace mechanism*

Input: $\{q_i\}_{i=1}^{M}$, the scores (e.g. χ^2 statistic) of all M candidate SNPs; K, the number of SNPs that we want to release; s, the sensitivity of the score function; ϵ, the privacy budget.

Output: K SNPs.

```
1: Add independent Laplace noise with mean zero and scale $\frac{2Ks}{\epsilon}$ to each of the M SNPs' scores.
2: Choose the top K SNPs based on the perturbed scores.
```

Algorithm 2 [17, 37] ϵ-differentially private algorithm for releasing the K most relevant SNPs using the *exponential mechanism*.

Input: $\{q_i\}_{i=1}^{M}$, the scores (e.g. χ^2 statistic, Hamming distance) of all M candidate SNPs; K, the number of SNPs that we want to release; s, the sensitivity of the score function; ϵ, the privacy budget.

Output: K SNPs.

```
1: Set $w_i = \exp\left(\frac{\epsilon q_i}{2Ks}\right)$. Define $\Pr(\mathcal{T}(D) = i) = w_i \Big/ \sum_{j=1}^{M} w_j$.
2: Sample $I \sim \mathcal{T}(D)$. Record SNP$_I$. Set $q_I = -\infty$.
3: Repeat Step 1 and 2 until K SNPs have been recorded.
```

the most relevant SNPs—that is, SNPs that have the largest χ^2 statistics—are released. Therefore, χ^2 statistic is a natural choice as score function for Algorithm 1 and Algorithm 2 when we release the most relevant SNPs. In addition to χ^2 statistic, another statistic proposed by Johnson and Shmatikov [17] can also be used as score function. We describe Johnson and Shmatikov [17]'s statistic in more details in Section 4.4.

Here, we describe three methods for releasing the most relevant SNPs:

Method 1. This method is based on Algorithm 1 with χ^2 statistic as score function. Given the χ^2 statistics of all the candidate SNPs and the sensitivity of χ^2 statistic, Algorithm 1 adds independent Laplace noise to each one of the original χ^2

statistics and outputs the top SNPs ranked by the perturbed χ^2 statistics. Algorithm 1 was first proposed by Bhaskar et al. [2] to output frequent patterns differentially privately. It was then adapted by Uhler et al. [29] and applied to a simulated GWAS dataset. Uhler et al. [29] showed that the sensitivity of the χ^2 statistic based on a 2×3 contingency table with positive margins, $N/2$ cases and $N/2$ controls is $\frac{4N}{N+2}$. In Section 4.3, we extend Uhler et al. [29]'s method and derive an upper bound for the sensitivity of the χ^2 statistic based on a 2×3 contingency table with positive margins and arbitrary numbers of cases and controls. In addition, we derive the the sensitivity of the χ^2 statistic based on a 2×2 contingency table with positive margins in Section 4.3. Furthermore, we provide a new interpretation of the Method 1 by assuming that data for the controls are known.

Method 2. This method is based on Algorithm 2 with χ^2 statistic as score function. Given the χ^2 statistics of all the candidate SNPs and the sensitivity of χ^2 statistic, Algorithm 2 iteratively samples from the set of candidate SNPs whose sampling weights are associated with their respective χ^2 statistics. The sampling weights also depend on the sensitivity of χ^2 statistic. Sensitivity results of χ^2 statistic are discussed in Method 1, and their derivations are described in Section 4.3.

Method 3. This method is also based on Algorithm 2, but, in contrast to Method 2, it uses Hamming distance score proposed by Johnson and Shmatikov [17] as score function. Hamming distance score can be described as the smallest number of moves one must make so that the significance of a genotype table changes. A move, counted as one Hamming distance in the space of genotype tables, is defined as changing the genotype of one individual. Significance refers to whether the p-value of the χ^2 statistic of the SNP is smaller than a prespecified threshold value or not. Details about how Hamming distance score is calculated are given in Section 4.4.

4.3 SENSITIVITY OF χ^2 STATISTIC

In this section, we describe the derivations of sensitivity results for the χ^2 statistic based on a 2×3 genotype table or a 2×2 allelic table. As both Method 1 and Method 2 use χ^2 statistic as score function, we have to derive the sensitivity results before we can use these methods. We begin this section with a brief review of sensitivity results for the χ^2 statistic based on a 2×3 genotype table derived in Uhler et al. [29], which assumed that there are equal numbers of cases and controls in a GWAS. We relax Uhler et al. [29]'s assumption and derive sensitivity results, assuming that there are arbitrary numbers of cases and controls in a GWAS. In addition to sensitivity results for the χ^2 statistic based on a 2×3 genotype table, we also derive sensitivity results for the χ^2 statistic based on a 2×2 allelic table. Furthermore, we provide a new interpretation of GWAS data privacy protection by assuming that the controls' data are known. Our results were published in Yu et al. [37].

Under the assumption that there are equal numbers of cases and controls, Uhler et al. [29] derived sensitivity results for the χ^2 statistic based on a 2×3 contingency table, the corresponding p-value, and the projected p-value. For completeness, we briefly review these sensitivity results here.

Theorem 4.3 (Uhler et al. [29]). *Sensitivity of the χ^2 statistic based on a 2×3 table with positive margins and $N/2$ cases and $N/2$ controls is $\frac{4N}{N+2}$.*

Theorem 4.4 (Uhler et al. [29]). *Sensitivity of the p-values of the χ^2 statistic based on a 2×3 contingency table with positive margins and $N/2$ cases and $N/2$ controls is $\exp(-2/3)$, when the null distribution is a χ^2 distribution with 2 degrees of freedom.*

Corollary 4.5 (Uhler et al. [29]). *Projecting all p-values larger than $p^* = \exp(-N/c)$ onto p^* results in a sensitivity of $\exp(-N/c) - \exp\left(-\frac{N(2Nc-4N-4c+c^2)}{2c(Nc-2N-c)}\right)$ for any fixed constant $c \geqslant 3$, which is a factor of $N/2$.*

In the remainder of this section, we generalize these sensitivity results to allow for arbitrary numbers of cases and controls. The generalization makes the proposed methods applicable in typical GWAS settings, in which there are more controls than cases, as researchers often use data pertaining to other diseases or publicly available data (e.g., the HapMap [1] project) as controls to increase the statistical power.

Let's first consider the situation in which the adversary has complete information about the controls. In this scenario, it is only necessary to protect information about the cases.

Theorem 4.6. *Let $\mathcal{D}$ denote the set of all 2×3 genotype tables with positive margins, R cases and S controls. Suppose that s_0, s_1, and s_2, the numbers of all three genotypes for the controls, are known. Let $N = R + S$, and $s_{\max} = \max\{s_0, s_1, s_2\}$. Then the sensitivity of the χ^2 statistic based on a table in $\mathcal{D}$ is bounded above by $\frac{N^2}{RS}\frac{s_{\max}}{1+s_{\max}}$.*

Proof. See Appendix A.1.2.1. □

Theorem 4.6 gives an upper bound for the sensitivity of the χ^2 statistic based on a 2×3 genotype table with positive margins and known controls. In Corollary 4.7, we remove the assumption that the controls are known, but we include the assumption that $r_0 \geqslant r_2$ and $s_0 \geqslant s_2$, which reflects the definition of major and minor alleles. We show in Corollary 4.7 how the sensitivity of the χ^2 statistic based on a 2×3 genotype table with positive margins is attained.

Corollary 4.7. *Let $\mathcal{D}$ denote the set of all 2×3 genotype tables with positive margins, R cases and S controls. We further assume that for tables in $\mathcal{D}$, $r_0 \geqslant r_2$ and $s_0 \geqslant s_2$; i.e., in the case and the control populations, the number of individuals having two minor alleles is no greater than the number of individuals having two major alleles. The sensitivity of the χ^2 statistic based on a table in $\mathcal{D}$ is $\frac{N^2}{RS}\left(1 - \frac{1}{\max\{S,R\}+1}\right)$, where $N = R + S$.*

Proof. See Appendix A.1.2.2. □

1 http://hapmap.ncbi.nlm.nih.gov/

If we have no knowledge of either the cases or the controls, we should use the sensitivity result presented in Corollary 4.7; on the other hand, when the controls are known, we can use Theorem 4.6 to reduce the sensitivity. Notice that in Theorem 4.6, when we assume the controls' data (s_0, s_1, and s_2) are known, the upper bound for the sensitivity is a function of $s_{\max} = \max\{s_0, s_1, s_2\}$. Therefore, we can divide the SNPs into groups having different $s_{\max}$ and use different upper bounds for sensitivity for different groups of SNPs. Nevertheless, because in most GWAS's the number of controls, S, is large and $s_{\max} = \max\{s_0, s_1, s_2\} \geqslant S/3$, we will only achieve insignificant reduction in sensitivity by dividing the SNPs into groups rather than treating all the SNPs as one group, in which case $s_{\max} \approx S$; this is illustrated by the following analysis:

$$\begin{aligned}\left\{\frac{N^2}{RS}\,\frac{S}{1+S}\right\}\Big/\left\{\frac{N^2}{RS}\,\frac{s_{\max}}{1+s_{\max}}\right\} &\leqslant \left\{\frac{N^2}{RS}\,\frac{S}{1+S}\right\}\Big/\left\{\frac{N^2}{RS}\,\frac{S/3}{1+S/3}\right\}\\ &= \frac{S+3}{S+1}\\ &\approx 1.\end{aligned}$$

To improve on statistical utility by reducing the sensitivity of the score function, Uhler et al. [29] proposed projecting the p-values that are larger than a threshold value onto the threshold value itself. In Theorem 4.8, we generalize this result to nonnegative score functions, showing how to incorporate projections into the Laplace mechanism.

Theorem 4.8. *Given a nonnegative function* $f(d)$*, define* $h_C(d) = \max\{C, f(d)\}$*, with* $C > 0$*; i.e., we project values of* $f(d)$ *that are smaller than* C *onto* C*. Let* s *denote the sensitivity of* $h_C(d)$*, and suppose* $Y \sim \text{Laplace}(0, \frac{s}{\epsilon})$*. Then* $W(d) = \max\{C, Z(d)\}$*, with* $Z(d) = h_C(d) + Y$*, is* ϵ*-differentially private.*

Proof. See Appendix A.1.2.3. □

To use the idea of projection in Theorem 4.8, we can set C to be the χ^2 statistic that corresponds to a small p-value, and use an upper bound in place of the sensitivity of the projection function. A valid upper bound is

$$s_C = \min\left\{Y_{\max} - C, \quad \frac{N^2}{RS}\left(1 - \frac{1}{\max\{S, R\} + 1}\right)\right\}.$$

Theorem 4.9. *The sensitivity of the allelic test χ^2 statistic based on a 2×3 genotype table with positive margins,* R *cases and* S *controls is given by the maximum of*

$$\left\{ \begin{array}{l} \dfrac{8N^2S}{R(2S+3)(2S+1)}, \\ \dfrac{4N^2[(2R^2-1)(2S-1)-1]}{RS(2R+1)(2R-1)(2S+1)}, \\ \dfrac{8N^2R}{S(2R+3)(2R+1)}, \\ \dfrac{4N^2[(2S^2-1)(2R-1)-1]}{RS(2S+1)(2S-1)(2R+1)} \end{array} \right\}.$$

Proof. See A.1.2.4. □

4.4 Hamming distance score

In this section, we introduce Hamming distance score, which is originally proposed by Johnson and Shmatikov [17]. Hamming distance score is used as score function in Method 3 for releasing the most relevance SNPs. We explain how Hamming distance score is defined and the difficulty in calculating it. Then we propose an efficient way of calculating Hamming distance score by appealing to a graphical interpretation of the χ^2 statistic based on a 2×2 allelic table. Results in this section were published in Yu and Ji [38].

Johnson and Shmatikov [17]'s *Hamming distance* is a distance metric defined in the space of genotype tables with positive margins and fixed numbers of cases and controls. In this section, when we say the Hamming distance from one database to another, we mean the Hamming distance from the genotype table of a particular SNP in one database to the genotype table of that same SNP in the other database. Then loosely speaking, the Hamming distance between a database D and another database D' is the number of individuals in D whose data one must replace so that D becomes D' after the replacement.

Denote the space of genotype tables by $\mathcal{D}$. *Hamming distance score* is based on the shortest Hamming distance between the database, D_0, that we observe and a set of databases, $\mathcal{D}^* \subset \mathcal{D}$, that satisfy the following condition: given an evaluation function $f : \mathcal{D} \to \mathcal{R}$ and a threshold value c^*, for each $D \in \mathcal{D}^*$,

$$\mathbb{1}_{\{f(D_0)>c^*\}} \times \mathbb{1}_{\{f(D)<c^*\}} + \mathbb{1}_{\{f(D_0)<c^*\}} \times \mathbb{1}_{\{f(D)>c^*\}} = 1.$$

In other words, f evaluated at D_0 and any database in $\mathcal{D}^*$ will not both be greater than c^* or smaller than c^*.

In the context of GWAS, we can use χ^2 statistic as evaluation function. Let $\bar{F}_{\chi^2}$ denote the p-value of a χ^2 statistic. Given a threshold value c^*, we call a database D *significant* if $\chi^2(D) = c_0 > c^*$, and we call D *insignificant* if $\chi^2(D) = c_0 < c^*$; equivalently, let $p^* = \bar{F}_{\chi^2}(c^*)$, then D is significant if $\bar{F}_{\chi^2}(c_0) < p^*$ and insignificant if $\bar{F}_{\chi^2}(c_0) > p^*$. There is not a rigorous way of choosing c^*, or, equivalently, p^*, but Johnson and Shmatikov [17] suggested that it may be reasonable to let p^* be the p-value derived from the Bonferroni correction of an α-level hypothesis test. For example, when there are M SNPs in the database, we may set $p^* = \alpha/M$ according to the Bonferroni correction. However, as we can see in Section 4.5, the choice of threshold value p^* affects the performance of Method 3 because the Hamming distance scores for the SNPs may change as p^* changes. Therefore, a more rigorous way of choosing p^* is needed to optimize the performance of Method 3.

4.4.1 *A naïve way of finding Hamming distance score*

Figure 4.1 is an illustration of how to find the Hamming distance between a significant database D and some insignificant database D'. Starting at D, we find that the χ^2 statistic of D is c_0 and the corresponding p-value is p_0. Because we assume that D is significant, thus $c_0 > c^*$ and, equivalently, $p_0 < p^*$. We replace the data of one individual in the cases or the controls so that D becomes D_1. If $c_1 = \chi^2(D_1) < c^*$, then D_1 becomes insignificant and we say that the Hamming distance is 1; otherwise, D_1 is still significant and we replace the data of one individual in the cases or the controls so that D_1 becomes D_2. More generally, we perform the following procedure: at Step n, if D_n is insignificant, then we will stop and say that the Hamming distance is n; otherwise, we replace the data of one individual in the cases or the controls of D_n so that D_n becomes D_{n+1} and continue to Step $n+1$. If D is insignificant, the Hamming distance between D and some significant database D' can be obtained in a fashion similar to that described in Figure 4.1; the only difference is that the procedure will keep incrementing n until D_n becomes significant.

Figure 4.1: An illustration of how to find the Hamming distance between a significant database D_0 and some insignificant database.

$$\begin{tikzcd}
\text{Database}: & D_0 \arrow[r, phantom, "\sim"] \arrow[d, Rightarrow] & D_1 \arrow[r, phantom, "\sim \cdots \sim"] \arrow[d, Rightarrow] & D_{n-1} \arrow[r, phantom, "\sim"] \arrow[d, Rightarrow] & D_n \arrow[d, Rightarrow] \\
\chi^2(\text{p-value}): & c_0\ (p_0) \arrow[d, Rightarrow] & c_1\ (p_1) \arrow[r, phantom, "\ldots"] \arrow[d, Rightarrow] & c_{n-1}\ (p_{n-1}) \arrow[d, Rightarrow] & c_n\ (p_n) \arrow[d, Rightarrow] \\
> c^*(< p^*)? & \text{Yes} & \text{Yes} & \text{Yes} & \text{No} \\
 & \text{(sig.)} & \text{(sig.)} & \text{(sig.)} & \text{(insig.)}
\end{tikzcd}$$

Figure 4.1 describes a path from a significant database D to an insignificant database in the corresponding space of genotype tables with positive margins and fixed numbers of cases and controls. To calculate the shortest Hamming distance, we consider all paths from D to any insignificant database and find the length of the shortest path. (The shortest path may not be unique.) In the same spirit, if D is insignificant, we find the shortest Hamming distance by examining all paths from D to any significant database. Recall that we denote the evaluation function by f and the threshold value by c^*. Let d denote the shortest Hamming distance for the database D. Then following the convention in Johnson and Shmatikov [17], we define the Hamming distance score, h, by

$$h = \begin{cases} -d, & \text{if } f(D) < c^*, \\ d-1, & \text{if } f(D) \geqslant c^*. \end{cases}$$

The sensitivity of Hamming distance score is 1 because changing one individual in a database changes the shortest Hamming distance by at most 1; a detailed proof was given in Johnson and Shmatikov [17]. Finding the sensitivity of an arbitrary score function is sometimes difficult; for example, when we use χ^2 statistic as score function, we have to perform elaborate analyses shown in Section 4.3 in order to find the sensitivity. Therefore, it is often more convenient to use Hamming distance score as score function, as there will be no need to calculate its sensitivity, which is always 1.

4.4.2 *Issues with Hamming distance score*

Despite the advantages, which includes eliminating the need to calculate the sensitivity, using Hamming distance score as score function is not without drawbacks. The biggest drawback is that it is impossible to compute the shortest Hamming distance naïvely. An naïve approach to computing the shortest Hamming distance of a significant database D is to consider all paths from D to any insignificant database and find the length of the shortest path. Such approach is computationally prohibitive as the number of paths to examine grows rapidly as the number of individuals in a database increases.

To address the computational difficulty with Hamming distance score, Johnson and Shmatikov [17] and Yu et al. [37] used proxies for the shortest Hamming distance. For a significant (insignificant) database D_0, Yu et al. [37] used a proxy that is the length of the path of greatest descent (ascent), which is constructed in the following manner: for $n \geqslant 0$, we change one individual in D_n so that $D_{n+1} - D_n$ is maximized (minimized).

While Johnson and Shmatikov [17] and Yu et al. [37] provided *ad-hoc* methods for computing approximations to the shortest Hamming distance, neither publications justified that the *ad-hoc* methods produce good approximations to the shortest Hammingdistance. Mostly importantly, it is unclear whether the proxies proposed by Johnson and Shmatikov [17] and Yu et al. [37] preserve the sensitivity of Hamming distance score. If sensitivities of the proxies are greater than the sensitivity of the shortest Hamming distance score, ϵ-differentially private mechanisms that use any of these proxies as score function will no longer be ϵ-differentially private! There are two solutions to this problem, we can either calculate the sensitivities of the proxies, which seems difficult, or we do not use the proxies at all. In Section 4.4.4, we present a computationally efficient way of finding the shortest Hamming distance when we use the χ^2 statistic based on a 2×2 allelic table as evaluation function. We prove that our method in Section 4.4.4 indeed produces the shortest Hamming distance and hence the sensitivity of the resulting score function is 1.

Besides the difficulty with computing the shortest Hamming distance, using Hamming distance score as score function has other issues: (i) the choice of threshold value c^* affects the performance of the exponential mechanism because the SNPs' Hamming distance scores change as c^* changes; (ii) ranking the SNPs by Hamming distance score may result in an ordering that is different from that resulting from ranking the SNPs by the evaluation function (e.g. χ^2 statistic). These issues are discussed in more details in Section 4.5.

4.4.3 *A graphical interpretation of χ^2 statistic for the allelic table*

Before we present our efficient method of finding the shortest Hamming distance in Section 4.4.4, we will first consider a graphical interpretation of the χ^2 statistic based on a 2×2 allelic table, as the graphical interpretation proves instrumental in justifying the validity of our method. Throughout this section, we refer to the χ^2 statistic based on a 2×2 allelic table simply as χ^2 statistic.

Let's refer to the case group's data and the control group's data collectively as a database and call the data for an individual a record. We can think of the number of cases, R, and the number of controls, S as fixed. Let's assume that the control group's data are known to the public, which is a plausible assumption as genetic researchers sometimes use publicly available genetic data as controls. Then for a given genotype table, we assume that s_0, s_1, and s_2 are fixed. Recall that a 2×2 allelic table can be derived from a 2×3 genotype table, and the χ^2 statistic based on a 2×2 allelic table (Table 4.1) is

$$\begin{aligned} Y_A &= \frac{2N}{RS(n_1 + 2n_2)(2n_0 + n_1)} \left[N(s_1 + 2s_2) - S(n_1 + 2n_2)\right]^2 \\ &= \frac{2N\left[(2r_0 + r_1)S - (2s_0 + s_1)R\right]^2}{RS(2r_0 + r_1 + 2s_0 + s_1)(2N - 2r_0 - r_1 - 2s_0 - s_1)}. \end{aligned}$$

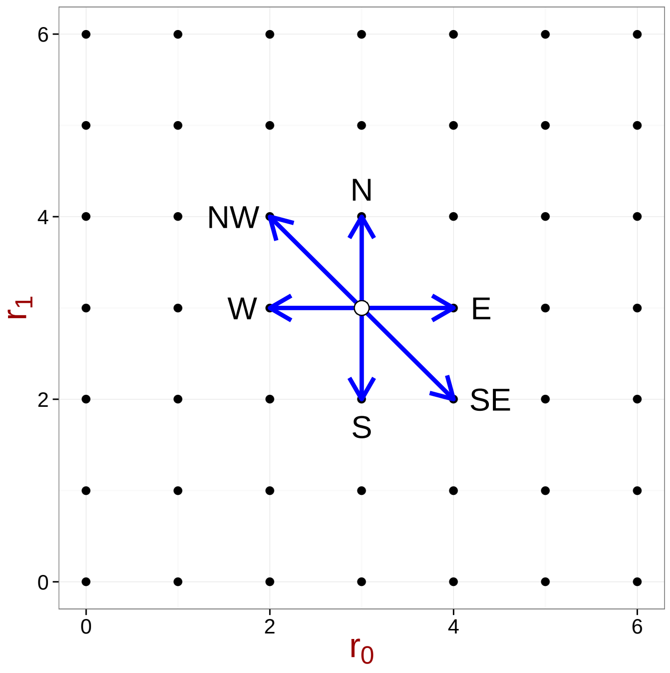
Therefore, with s_0, s_1, and s_2 fixed, Y_A is only a function of r_0 and r_1.

How χ^2 statistic changes when we change one record in the database is illustrated in Figure 4.2. In Figure 4.2, each dot represents the χ^2 statistic given r_0 and r_1. When we change one record in the case group, there are 6 possible changes to the genotype table: $(r_0 \to r_0 + 1, r_1 \to r_1)$, $(r_0 \to r_0 + 1, r_1 \to r_1 - 1)$, $(r_0 \to r_0, r_1 \to r_1 - 1)$, $(r_0 \to r_0 - 1, r_1 \to r_1)$, $(r_0 \to r_0 - 1, r_1 \to r_1 + 1)$, and $(r_0 \to r_0, r_1 \to r_1 + 1)$; that is, r_0 and r_1 cannot both increase or decrease by 1. The possible changes are shown as arrows in Figure 4.2. A change in the genotype table results in a change in the allelic table, and we get a new χ^2 statistic based on the new allelic table.

Let p^* denote a pre-specified threshold p-value. Let c^* denote the χ^2 statistic corresponding to p^*, with the χ^2 statistic having the χ^2 distribution with 1 degree of freedom. Then for a given SNP in the pool of candidate SNPs, the genotype table of which we denote by D, the shortest Hamming distance is the smallest number of sequential changes made to D such that the resulting genotype table, D', satisfies $Y_A(D') \geqslant c^*$ if $Y_A(D) < c^*$ or $Y_A(D') < c^*$ if $Y_A(D) \geqslant c*$; in other words, if we call c^* the significance threshold, then the goal is to make changes to the "insignificant" ("significant") table D so that the χ^2

statistic of D' goes above (below) the significance threshold c^*, and D' becomes a "significant" ("insignificant") table. Let d denote the shortest Hamming distance, then Hamming distance score is defined as $h = d - 1$ if $Y_A(D) \geqslant c^*$ and $h = -d$ if $Y_A(D) < c^*$.

Figure 4.2: Legal moves in the space of genotype tables with fixed R, S, s_0, s_1, and s_2.



Let's consider the space of genotype tables, $\mathcal{B}_D$, defined by a genotype table D: for all $D' \in \mathcal{B}_D$, D' shares the same values of s_0, s_1, s_2, R, S, and N with D, but D' does not necessarily have the same values of r_0, r_1, and r_2 as D. Let $n_{10} = 2s_0 + s_1$ denote the number of major alleles in the control group, and let $x = 2r_0 + r_1$ denote the number of major alleles in the case group, then we can write the χ^2 statistic based on a genotype table $D' \in \mathcal{B}_D$ as a function of x:

$$\begin{aligned} Y_A(D'; \mathcal{B}_D) &= Y(r_0, r_1; D) = Y_A(x; D) \\ &= \frac{2N\,(\,xS - n_{10}R\,)^2}{RS(x + n_{10})(2N - x - n_{10})}, \end{aligned}$$

where r_0, r_1 and x are derived from D', and n_{10}, R, S and N are the same for D and D'. For notational convenience, when r_0 and r_1 are also derived from D, we will simply write $Y_A(D; \mathcal{B}_D) = Y_A(D)$.

4.4.4 *An efficient way of finding the shortest Hamming distance*

Lemma 4.10. Y_A *is an increasing function of* x *when* $xS - n_{10}R > 0$*, and it is a decreasing function of* x *when* $xS - n_{10}R < 0$*.*

Proof. See A.1.3.1. □

To understand the implication of Lemma 4.10, let's consider Figure 4.3. In Figure 4.3, each dashed line represents a value of x, which is written as $x = 2r_0 + r_1$. Because we can consider each dot in Figure 4.3 to be a unique genotype table in the space of genotype tables with fixed control data and a fixed number of cases, those tables that lie on the same dashed line will have the same χ^2 statistic. Furthermore, because $0 \leqslant r_0 + r_1 \leqslant R$, $r_0 \geqslant 0$, and $r_1 \geqslant 0$, the space of genotype tables, represented as dots, fall within a triangle in Figure 4.3.

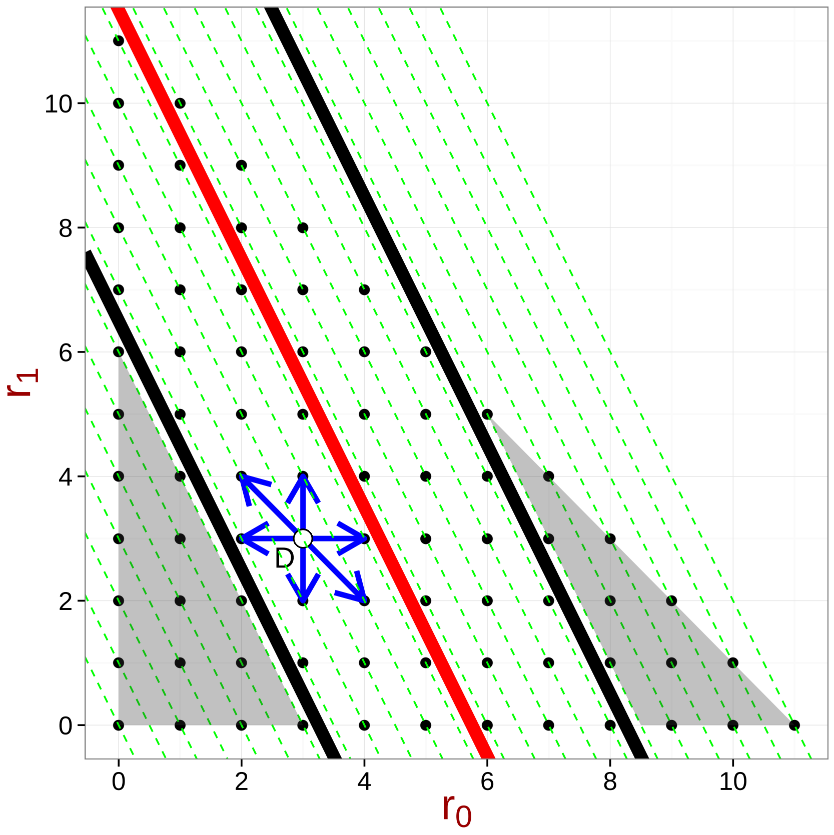
For the moment, let's treat r_0 and r_1 as continuous values. In Figure 4.3, the red solid line represents the line $2r_0 + r_1 = x = n_{10}R/S$, and the two solid black lines represent the lines $2r_0 + r_1 = x_1$ and $2r_0 + r_1 = x_2$, with $Y_A(x_1; \mathcal{B}_D) = Y_A(x_2; \mathcal{B}_D) = c$. There are two black lines because by Lemma 4.10, $Y_A(x)$ is an increasing function when $x > n_{10}R/S$, and it is a decreasing function when $x < n_{10}R/S$; that is, there could be two values of x, say x_1 and x_2, such that $Y_A(x_1; \mathcal{B}_D) = Y_A(x_2; \mathcal{B}_D)$ and $x_1 < n_{10}R/S < x_2$. However, because it is possible that

$$\begin{aligned}\max_x Y_A(x; D) &\leqslant \max\{Y_A(0; \mathcal{B}_D), Y_A(2R; \mathcal{B}_D)\} \\ &= \max\left\{\frac{2NRn_{10}}{S(2N - n_{10})}, \frac{2NR(2S - n_{10})}{S(2R + n_{10})}\right\} < c,\end{aligned}$$

thus there could be genotype tables for which only one black line exists or no black line exists at all; in such cases, we will substitute a black line with $0 = 2r_0 + r_1$ or $2R = 2r_0 + r_1$, whichever is appropriate.

In Figure 4.3, the genotype table D is insignificant and its χ^2 statistic is below the threshold value. By Lemma 4.10, we know that the χ^2 statistics of genotype tables, which are

Figure 4.3: An example of a genotype table, D, in the space of genotype tables with fixed R, S, s_0, s_1, and s_2. Each dot represent a genotype table. Each dashed line has $slope = -2$, representing the lines $x = 2r_0 + r_1$. The red line is $x = (2s_0 + s_1)R/S = 2r_0 + r_1$, and the two black lines correspond to values of $(2r_0 + r_1)$ such that $Y_A(r_0, r_1; \mathcal{B}_D) = c$, where c is a pre-specified significance threshold value.



represented by the dots on Figure 4.3, are greater than c when they are inside the shaded area (outside of the area between two black lines) and smaller than c when they are inside the non-shaded area (inside of the area between two black lines). Therefore, finding the shortest Hamming distance for D is equivalent to finding the shortest Hamming distance from the genotype table D to genotype tables in the shaded areas.

For genotype tables that are significant, they will fall into the shaded areas in Figure 4.3. Then finding the shortest Hamming distance for a significant genotype table is equivalent to finding the shortest Hamming distance from the genotype table to genotype tables in the non-shaded area.

Proposition 4.11. *Given a significance threshold value* c *and an insignificant genotype table* D *(i.e.,* $Y_A(D) < c$*), if there exists* $D' \in \mathcal{B}_D$ *such that* $Y_A(D'; \mathcal{B}_D) \geqslant c$*, then the shortest Hamming distance is* $\min\{H_1, H_2\}$*, where* H_1 *and* H_2 *are defined as follows:*

(i) H_1 *is the number of changes made to* D *in the following manner: (1) keep decreasing* r_0 *until the new genotype table,* D'*, becomes significant (i.e.,* $Y_A(D', \mathcal{D}) > c$*); (2) when* r_0 *is minimized but the new table is still insignificant, keep decreasing* r_1 *until the new table becomes significant.*

(ii) H_2 *is the number of changes made to* D *in the following manner: (1) keep increasing* r_0 *until the new genotype table becomes significant; (2) if* r_0 *can no longer be increased without decreasing* r_1 *and the new table is still insignificant, increase* r_0 *and decrease* r_1 *in each change until the new table becomes significant.*

If for all $D' \in \mathcal{B}_D, Y_A(D'; \mathcal{B}_D) < c$*, then we define the shortest Hamming distance as* $\min\{H_1', H_2'\}$*, where* H_1' *and* H_2' *are defined as follows:*

(i) When r_0 *and* r_1 *are both minimized but the new table is still insignificant, set* H_1' *to* $1 + d_1$*, where* d_1 *is smallest the number of changes needed to minimize* r_0 *and* r_1*.*

(ii) When r_0 *and* r_1 *are both maximized but the new table is still insignificant, set* H_2' *to* $1 + d_2$*, where* d_2 *is smallest the number of changes needed to maximize* r_0 *and* r_1*.*

Proof. See A.1.3.2. □

Proposition 4.12. *Given a significance threshold value* c *and a significant genotype table* D *(that is,* $Y_A(D) \geqslant c$*), the shortest Hamming distance is* $\min\{H_1, H_2\}$*, where* H_1 *and* H_2 *are defined as follows:*

(i) If $2r_0 + r_1 > (2s_0 + s_1)R/S$*, set* $H_1 = \infty$*; otherwise,* H_1 *is the number of changes made to* D *in the following manner: keep decreasing* r_0 *until the new genotype table,* D'*, becomes insignificant (i.e.,* $Y_A(D', \mathcal{D}) < c$*).*

(ii) If $2r_0 + r_1 < (2s_0 + s_1)R/S$, *set* $H_2 = \infty$*; otherwise,* H_2 *is the number of changes made to* D *in the following manner: keep decreasing* r_0 *until the new genotype table becomes insignificant.*

Proof. The proof is similar to the proof of Proposition 4.11. □

Definition 4.1 (Hamming distance score)**.** *Given a threshold* χ^2 *statistic value* c *and a genotype table* D*, the Hamming distance score of* D *is*

$$h = \begin{cases} -d^-, & \text{if } Y_A(D) < c, \\ d^+ - 1, & \text{if } Y_A(D) \geqslant c, \end{cases}$$

where d^- *is found using Proposition 4.11 and* d^+ *is found using Proposition 4.12.*

Corollary 4.13. *The sensitivity of Hamming distance score as defined in Definition 4.1 is 1.*

4.5 An application to real human genetic data

In this section, we apply Method 1, Method 2, and Method 3 to a GWAS dataset containing real human DNA samples from the Wellcome Trust Case Control Consortium (WTCCC). We use the differentially private methods described in Section 4.2 to release the most relevant SNPs, and we evaluate the trade-off between data utility and privacy risk of the differentially private methods.

Compared to similar analyses done by Uhler et al. [29] and Johnson and Shmatikov [17], our analysis is different in the following respects:

(i) Improving upon Uhler et al. [29]'s sensitivity results, which are based on the assumption that the numbers of cases and controls are the same, we use the new sensitivity results derived in Section 4.3 to allow for arbitrary numbers of cases and controls.

(ii) Uhler et al. [29] only analyzed Method 1, which is Algorithm 1 (Laplace mechanism) with χ^2 statistic as score function. Our analysis does not only cover Method 1, but it also includes results of applying Method 2 and Method 3, which are Algorithm 2 (exponential mechanism) with χ^2 statistic and Hamming distance score as score functions.

(iii) In contrast to Johnson and Shmatikov [17], which used the G-test to construct Hamming distance score, we use χ^2 statistic instead.

4.5.1 *Dataset from WTCCC: Crohn's disease*

We use a real dataset that was collected by the WTCCC and intended for genome-wide association studies of Crohn's disease. The dataset consists of DNA samples from 3 cohorts, the subjects of which all lived within Great Britain and identified themselves as white Europeans: *1958 British Birth Cohort* (58C), *UK Blood Services* (NBS), and *Crohn's disease* (CD). In the original study [33] DNA samples from the 58C and NBS cohorts were treated as controls and those from the CD cohort as cases.

The data were sampled using the Affymetrix GeneChip 500K Mapping Array Set. The genotype data were called by an algorithm named CHIAMO (see [33]), which WTCCC developed and deemed more powerful than Affymetrix's BRLMM genotype calling algorithm. According to the WTCCC analysis, some DNA samples were contaminated or came from non-Caucasian ancestry. In addition, they indicated that some SNPs did not pass quality control filters. Finally, WTCCC [33] removed additional SNPs from their analysis by visually inspecting cluster plots.

4.5.2 *Our re-analysis of the WTCCC data*

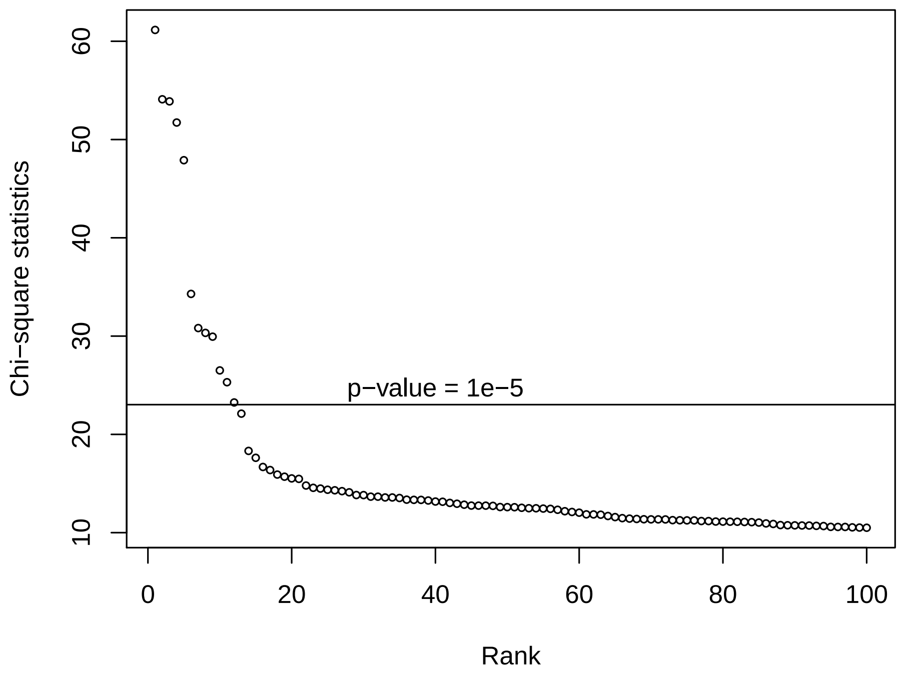
Wellcome Trust Case Control Consortium [33] mainly used the χ^2 statistic based on a 2×2 allelic table or a 2×3 genotype table to find SNPs with strong associations with Crohn's disease. Wellcome Trust Case Control Consortium [33] reported the most relevant SNPs' χ^2 statistics and the corresponding p-values. In general, Wellcome Trust Case Control Consortium [33] considered a SNP significant if the p-value corresponding to the SNP's χ^2 statistic is smaller than 10^{-5}. In the supplementary material of [33], the authors reported 26 significant SNPs, 6 of which were imputed. Per [33], imputing SNPs that do not exist in the WTCCC databases does not affectc calculation of the χ^2 statistics of SNPs that are already in the WTCCC databases; therefore, we disregard the imputed SNPs in our analysis and retain 20 significant SNPs.

We follow the filtering process in [33] closely and remove DNA samples and SNPs that [33] deemed contaminated. However, we have not removed SNPs with poor cluster plots. We verify that our processing of the raw genotype data leads to the same results as those published in the supplementary material of [33]: our calculations for 16 of the 20 reported significant SNPs match those in [33], deviating no more than 2% in χ^2 statistic. However, we find that a number of significant SNPs are not reported by Wellcome Trust Case Control Consortium [33]. Our correspondence with one of the principal authors of [33] confirms that [33] also found those SNPs to be significant. However, Wellcome Trust Case Control Consortium [33] did not report those SNPs because they suffered from poor calling quality according to visual inspection of the cluster plots, a procedure that we have not implemented. We therefore exclude from our analysis SNPs that have significant p-values but are not reported by the WTCCC.

In Figure 4.4 we plot in descending order the unperturbed χ^2 statistics based on genotype tables resulting from our analysis. We observe that there is a large gap between the 5th and the 6th largest χ^2 statistics. This turns out to be an important observation for the risk-utility analysis in Section 4.5.3: because of the nature of the distribution of the top χ^2 statistics in this dataset, it is easier to recover all top 5 SNPs in the perturbed data than it is to recover all top K SNPs for $K < 5$ or $K > 5$, as is evident in Figure 4.5.

To summarize, we are able to reproduce a high percentage of significant SNPs from Wellcome Trust Case Control Consortium [33]. Therefore, we are confident that our data

Figure 4.4: The top 100 unperturbed χ-statistics based on 2×3 genotype tables plotted in descending order.



processing procedure is sound and the χ^2 statistics that we obtain from the data are comparable with those produced in a high quality GWAS.

4.5.3 *Risk-utility analysis*

In this section, we use χ^2 statistics obtained from the WTCCC dataset using processes described in Section 4.5.1 and 4.5.2 to analyze the statistical utility of releasing differentially private χ^2 statistics for various privacy budgets. With 1748 cases and 2938 controls in the WTCCC dataset, we use Corollary 4.7 to obtain an upper bound, which is 4.27, for the sensitivity of the χ^2 statistic based on genotype tables.

We define *statistical utility* as follows: let $\mathcal{S}_0$ be the set of top K SNPs ordered according to their true χ^2 statistics and let $\mathcal{S}$ be the set of top K SNPs chosen after perturbation (by Method 1, 2, or 3). Then utility as a function of ϵ is

$$u(\epsilon) = \frac{|\mathcal{S}_0 \cap \mathcal{S}|}{|\mathcal{S}_0|}.$$

We perform the following procedure to approximate the expected utility $\mathbb{E}[u(\epsilon)]$: (i) assign a score to each SNP; (ii) denote the set of top K SNPs chosen according to the scores by $\mathcal{S}_0$; (iii) output the top K SNPs using Algorithm 1 or Algorithm 2 and denote the output by $\mathcal{S}$; (iv) calculate $u(\epsilon) = \frac{|\mathcal{S}_0 \cap \mathcal{S}|}{|\mathcal{S}_0|}$; (v) repeat (iii) 50 times for a fixed ϵ and report the average utility $\overline{u(\epsilon)}$. For Method 1, we use χ^2 statistic as score in Step (i) and use Algorithm 1 in Step (iii); for Method 2, we use χ^2 statistic as score in Step (i) and use Algorithm 2 in Step (iii); and lastly, for Method 3, we use Hamming distance scores in Step (i) and use Algorithm 1 in Step (iii).

The runtimes of the different methods vary considerably (see Table 4.3). They are reported based on experiments run on a PC with Intel i5-3570K CPU, 32 GB of RAM and the Ubuntu 13.04 operating system. Calculating the χ^2 statistics from genotype tables is a trivial task and takes very little time. Calculating Hamming distance scores, on the other hand, can be a daunting task if one cannot find a clever simplification. Hamming distance score relies on finding the shortest Hamming distance between the original genotype table and the set of genotype tables in which the significance of the p-value of each genotype table differ from that of the original genotype table. Thus, without any simplification, one would need to search the entire space of genotype tables in order to find the shortest Hamming distance. In our implementation of searching for the shortest Hamming distance, we greedily follow the path of maximum change—that is, greatest ascent or descent—of the χ^2 statistic until we find a genotype table with altered significance.

Note that the path found using the greedy method is only an approximation to the shortest Hamming distance, which should raise concerns about the sensitivity of the score function not being 1 and whether differential privacy is preserved. We addressed these concerns in Yu and Ji [38] and showed that if χ^2 statistics based on allelic tables are used instead of χ^2 statistics based on genotype tables, we can use the result in Section 4.4.4 to find the exact shortest Hamming distance efficiently.

Table 4.3: Comparison of runtimes for simulations in Section 4.5.3. The number of repetitions is 50, the number of different values for K is 4, the number of different values for ϵ is 15, and the number of SNPs is around 4000. $\mathcal{S}$ is the set of SNPs to be released after the perturbation.

Method	Time spent on generating $\mathcal{S}$ (in minutes)	Time spent on calculating the scores (in minutes)
Algorithm 1 (Laplace with χ^2)	0.04	≈ 0
Algorithm 2 (exponential with χ^2)	1.53	≈ 0
Algorithm 2 (exponential with Hamming)	2.00	3.50

In Figure 4.5, we compare the performance of Method 1, Method 2, and Method 3. It is clear that when $\epsilon = 1$, the Hamming distance score method (Method 3) outperforms the other methods by achieving the highest utility. Nevertheless, we note a few undesirable features regarding the performance of the Hamming distance score method.

- When $K = 3$, the utility of the Hamming distance score method does not exceed 0.67 even as ϵ continues to increase. This artifact is due to the fact that ranking the SNPs based on Hamming distance score results in an ordering that is different from the ordering based on χ^2 statistic. Table 4.4 lists the top 6 SNPs ranked by their χ^2 statistics and the SNPs' Hamming distance scores. For all threshold p-values, the Hamming distance score of the 4th SNP is larger than that of the 2nd SNP. As a result, when ϵ is large, the ordering of the top SNPs according to their Hamming distance scores does not change much after perturbation, and the Hamming distance score method will almost always output the 1st, 3rd, and 4th SNPs. Consequently, when $K = 3$, the utility for the Hamming distance score method will not reach 1 even as ϵ continues to increase.

- The Hamming distance score method is sensitive to the choice of threshold p-value. This becomes apparent in the plots for $K = 15$ in Figure 4.5. The risk-utility curves of the Hamming distance score method tend to have lower utility for the same ϵ when the threshold p-value is larger.

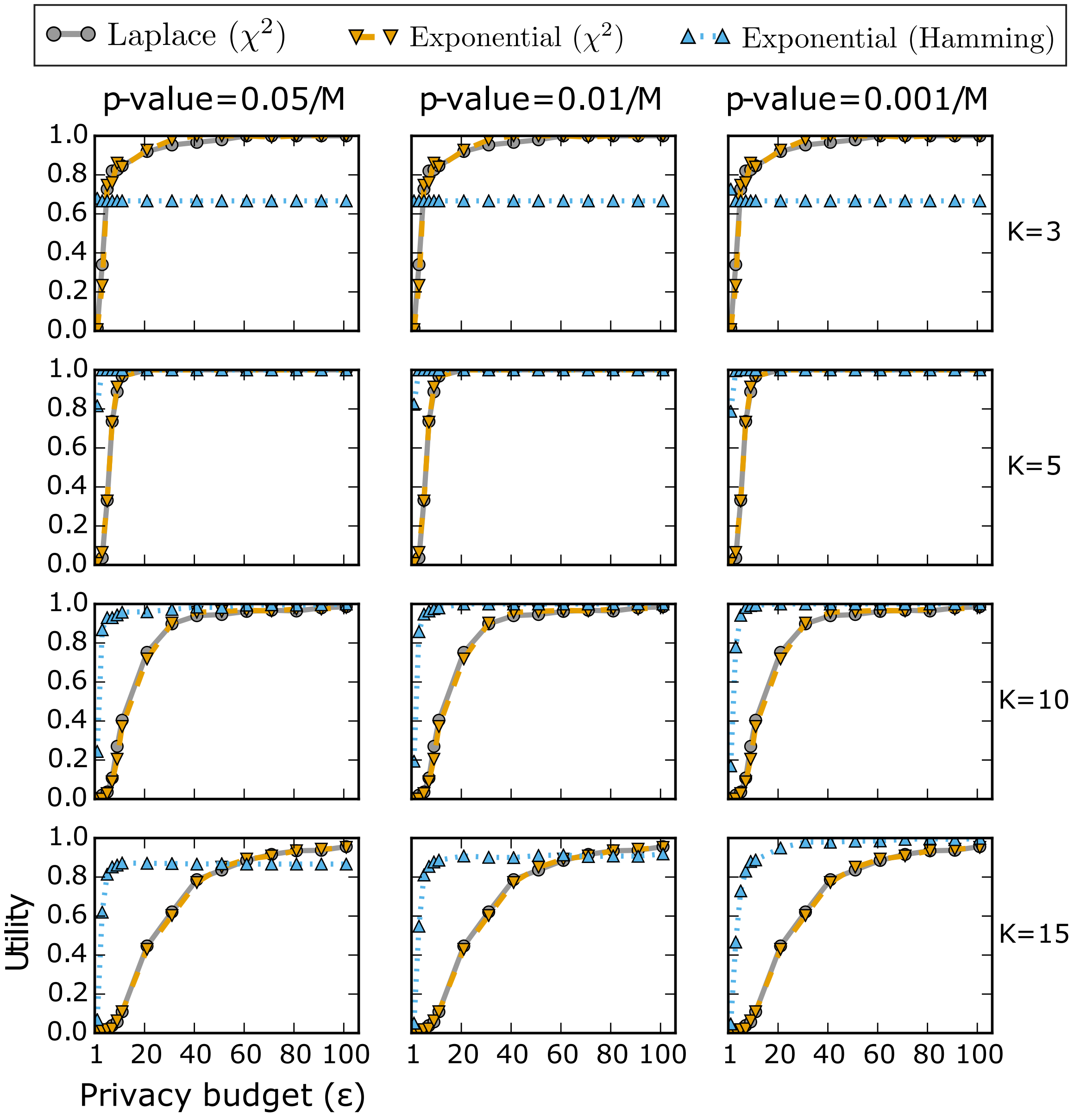
- Neither methods that use χ^2 statistic as score function (Method 1 and Method 2) perform as well as the Hamming distance score method for small values of ϵ. However, the χ^2 statistic methods do not suffer from the aforementioned issues. Furthermore, we can see in Figure 4.5 that both χ^2 statistic methods have similar performance, as they achieve approximately the same utility for each value of ϵ.

Table 4.4: χ^2 statistics and Hamming distance scores of the top 6 SNPs. M denotes the total number of SNPs.

Scoring scheme	Threshold p-value	Score (nearest integer)					
		1st	2nd	3rd	4th	5th	6th
χ^2 statistic	-	61	54	54	52	48	34
Hamming distance score	$0.001/M$	51	31	37	33	25	6
Hamming distance score	$0.01/M$	61	38	47	41	33	13
Hamming distance score	$0.05/M$	69	43	55	48	38	18

To summarize, the Laplace mechanism (Algorithm 1) and the exponential mechanism (Algorithm 2) have similar performance when the same score function is used. The method based on the Hamming distance score has the best performance for small values of ϵ, but it shows problematic behaviors when ϵ increases. Finally, the Hamming distance score

Figure 4.5: Performance comparison of Method 1 (Algorithm 1 (Laplace mechanism) with χ^2 statistic as score function), Method 2 (Algorithm 2 (exponential mechanism) with χ^2 statistic as score function), and Method 3 (Algorithm 2 with Hamming distance score as score function). Each row corresponds to a fixed K, the number of most relevant SNPs to release. Each column corresponds to a fixed threshold p-value, which is relevant only to Method 3, the Hamming distance score method. Data used to generate this figure consist of SNPs whose p-values are smaller than 10^{-5} and a randomly chosen 1% sample of SNPs whose p-values are larger than 10^{-5}; the total number of SNPs used for calculation is $M = 3882$.



method comes at a much higher computational cost than the other two algorithms and might not be computationally feasible for some datasets.

5 PRIVACY PRESERVING RELEASE OF REGRESSION COEFFICIENTS

When we consider a phenotype as the result of complex relationships between multiple SNPs, GWAS becomes a very high dimensional problem because the number of pairwise interaction terms grows quadratically with the number of SNPs. A popular approach to solving this high-dimensionality problem is a two-step procedure. In the first step, all SNPs are screened and a subset of SNPs are selected based on single-SNP association tests such as the χ^2 test for association between a SNP and the phenotype. In the second step, the selected subset of SNPs are tested for multiple-SNP association using penalized logistic regressions. Penalized regression approaches, such as ℓ_2 penalized logistic regression [25] and elastic-net penalized logistic regression [6, 1], have been shown to be effective in overcoming the challenges posed by the high-dimensional nature of genetic data.

To construct a differentially private mechanism for the two-step procedure, we can appeal to the sequential composition property of differential privacy [23] and use a differentially private release method in each step. For the first step, in which we select a subset of the most relevant SNPs, we can use one of the differentially private methods described in Chapter 4. In this chapter, we describe differentially private methods for the second step, in which we release coefficients of penalized logistic regressions.

5.1 Penalized logistic regression

In a GWAS setting, the logistic loss function $l(\theta; x, y)$ is given by

$$l(\theta; x, y) = \log\left(1 + \exp\left(-y\theta^T x\right)\right),$$

where y denotes the disease status—$y = 1$ if diseased and $y = -1$ if non-diseased, $x = (1, x_1, \ldots, x_M)$ is a feature vector, with the first entry corresponding to the intercept and $x_i \in \{0, 1, 2\}$ indicating the number of minor alleles at SNP_i, and $\theta \in \mathbb{R}^{M+1}$ is a vector of coefficients. Let N denote the total number of individuals and let $r(\theta)$ denote the penalization term. The regression coefficients of the penalized logistic regression problem are obtained by minimizing the loss function

$$L(\theta) = \sum_{i=1}^{N} \log\left(1 + \exp\left(-y_i\theta^T x_i\right)\right) + r(\theta)$$

with respect to θ. For ℓ_2 penalized regression, $r(\theta) = \frac{1}{2}\lambda\|\theta\|_2$, where λ is a tuning parameter that controls the extent to which the penalization term affects the loss function. For elastic-net penalized regression, $r(\theta) = \frac{1}{2}\lambda(1-\alpha)\|\theta\|_2 + \lambda\alpha\|\theta\|_1$, where λ is a tuning parameter that controls the extent to which the penalization terms affect the loss function and α is a tuning parameter that controls the sparsity of the resulting model.

5.2 Output perturbation

In this section, we describe a method for releasing regression coefficients differentially privately by perturbing the coefficients. However, we do not go into much details about this method as it is not the focus of this thesis.

The framework for constructing a differentially private method using the Laplace mechanism, which is outlined in Dwork et al. [10], suggests that we can release regression coefficients differentially privately by adding Laplace-like noise to coefficients that optimize the objective function. Let $\mathcal{T}(D) = \arg\min L(\theta; D)$, we can extend the notion of sensitivity by Definition 5.1:

Definition 5.1. *The* ℓ_p-sensitivity *of a function* $f : \mathcal{D} \times \mathbb{R}^d \to \mathbb{R}^k$ *is the smallest number* $S_p(f)$ *such that*

$$\sup_{x \in \mathbb{R}^d} \|f(D, x) - f(D', x)\|_p \leqslant S_p(f),$$

for all databases $D, D' \in \mathcal{D}$ *such that* $D \sim D'$.

The following is an example of constructing a differentially private algorithm by perturbing the output of $\mathcal{T}(D)$: suppose we know $S_1(\mathcal{T})$, the ℓ_1-sensitivity of $\mathcal{T}(D)$, then we can release the regression coefficients while satisfying ϵ-differential privacy simply by adding independent Laplace noise to each coefficient in $\mathcal{T}(D)$, where the scale of the Laplace noise is a function of $S_1(\mathcal{T})$, ϵ, and the dimension of θ.

In practice, however, releasing regression coefficients differentially privately by perturbing the output of the regression problem is difficult to implement. Firstly, it is often hard to calculate $S_p(\mathcal{T})$. Furthermore, as was pointed out by Chaudhuri et al. [4], $S_p(\mathcal{T})$ can be very large relative to the values of the coefficients, which leads to high variance for the noise-generating distribution and consequently severely degrades the statistical utility of the differentially private output.

5.3 Objective function perturbation

5.3.1 *Existing methods*

Inspired by the exponential mechanism proposed by McSherry and Talwar [24], Chaudhuri et al. [4] proposed a differentially private release mechanism that adds perturbation noise to the objective function and returns coefficients that optimize the perturbed objective function. In contrast to the output perturbation method described in Section 5.2, which adds Laplace-like perturbation noise to coefficients that optimize the unperturbed objective function, Chaudhuri et al. [4]'s mechanism eliminates the need to find the sensitivity of the objective function, which is often a difficult if not impossible task. Furthermore, as is shown later in this chapter, Chaudhuri et al. [4]'s mechanism and the extensions can be easily implemented.

Since the publication of Chaudhuri et al. [4], a series of papers have extended the idea of constructing differentially private mechanisms via objective function perturbation. Chaudhuri et al. [4] proposed a differentially private mechanism that deals with strongly convex optimization problems; for example, the loss function of ℓ_2 penalized logistic regression is strongly convex. Kifer et al. [18] extended Chaudhuri et al. [4] to deal with any convex optimization problems (see Algorithm 5), which include elastic-net penalized logistic regression. The key idea of Kifer et al. [18]'s extension is dubbed "successive approximation", a method that constructs a differentially private mechanism by taking the limit of a sequence of differentially private mechanisms.

Because the performance of penalized logistic regression depends heavily on the choice of penalization parameters, it is necessary to address the problem of how to choose penalization parameter differentially privately. Chaudhuri and Vinterbo [5] proposed a differentially private mechanism for selecting the penalization parameters via cross-validation by using a stability-based argument (see Definition 5.2). Chaudhuri and Vinterbo [5]'s mechanism is applicable to strongly convex optimization problems (e.g., ℓ_2 penalized logistic regression). In this chapter, we extend Chaudhuri and Vinterbo [5] so that the mechanism is applicable to any convex optimization problems, including elastic-net penalized logistic regression (see Theorem 5.1). Our extension was published in Yu et al. [39].

5.3.2 *Definitions and notation*

Let $l : \mathbb{R}^s \times \mathcal{D} \to \mathbb{R}$ denote the loss function, $r : \mathbb{R}^s \to \mathbb{R}$ the regularization function, and $h : \mathbb{R}^s \times \mathcal{D} \to \mathbb{R}$ the validation function. Let $T \in \mathcal{D}^n$ be a training dataset of size n drawn from $\mathcal{D}$ and $V \in \mathcal{D}^m$ a validation dataset of size m also drawn from $\mathcal{D}$. Let $b \in \mathbb{R}^s$, $b \sim \mathcal{G}$ denote the noise used to perturb the penalized loss function. Then we denote by $\mathcal{T}(\lambda, \epsilon; T, l, r, b)$ the differentially private procedure to produce parameter estimates from the training data T given the regularization parameter λ, the privacy budget $\epsilon > 0$, the loss function l, the regularization function r, and the random noise b. We assign a score to a vector of regression coefficients θ that results from the random procedure $\mathcal{T}(\lambda, \epsilon; T, l, r, b)$ using the validation data V and the validation score function $q(\theta; V) = -\frac{1}{m} \sum_{d \in V} h(\theta; d)$.

Definition 5.2 ((β_1, β_2, δ)-stability [5]). *A validation score function q is said to be $(\beta_1, \beta_2, \delta)$-stable with respect to a training procedure $\mathcal{T}$, a set of candidate regularization parameters Λ, and the privacy budget ϵ if there exists a set $E \subset \mathbb{R}^s$ such that $\mathbb{P}(b \in E) \geqslant 1 - \delta$ and, when $b \in E$, the following conditions hold:*

1. **Training stability:** *for all $\lambda \in \Lambda$, for all validation datasets $V \in \mathcal{D}^m$, and all training dataset $T, T' \in \mathcal{D}^n$ with $T \sim T'$,*

$$|\, q(\mathcal{T}(\lambda, \epsilon; T, l, r, b); V) - q(\mathcal{T}(\lambda, \epsilon; T', l, r, b); V) \,| \leqslant \frac{\beta_1}{n}.$$

2. **Validation stability:** *for all $\lambda \in \Lambda$, for all training datasets $T \in \mathcal{D}^n$, and all validation datasets $V, V' \in \mathcal{D}^m$ with $V \sim V'$,*

$$|\, q(\mathcal{T}(\lambda, \epsilon; T, l, r, b); V) - q(\mathcal{T}(\lambda, \epsilon; T, l, r, b); V') \,| \leqslant \frac{\beta_2}{m}.$$

Algorithm 3 Validate($\Theta, \mathcal{T}, \mathcal{G}, T, V, \beta_1, \beta_2, \alpha_1, \alpha_2$), a differentially private procedure for selecting regularization parameter [5]

```
Input: Parameter list $\Theta = \{\theta_1, \dots, \theta_k\}$, training procedure $\mathcal{T}$, noise distribution $\mathcal{G}$, validation score function $q$, training dataset $T$ of size $n$, validation dataset $V$ of size $m$, stability parameters $\beta_1$ and $\beta_2$, training privacy budget $\alpha_1$, validation privacy budget $\alpha_2$.
1: for $i = 1, \dots, k$ do
2:         Draw $b_i \sim \mathcal{G}$. Compute $h_i = \mathcal{T}(\theta_i, \alpha_1; T, b_i)$.
3:         Let $\beta = \max(\frac{\beta_1}{n}, \frac{\beta_2}{m})$.
4:         Let $t_i = q(h_i; V) + 2\beta Z_i$, where $Z_i \sim \exp\left(\frac{1}{\alpha_2}\right)$.
5: end for
Output: $i^* = \arg\max_i t_i$.
```

Algorithm 4 End-to-end differentially private training and validation procedure [5]

```
Input: Parameter list $\Theta = \{\theta_1, \dots, \theta_k\}$, training procedure $\mathcal{T}$, noise distribution $\mathcal{G}$, validation score function $q$, training dataset $T$ of size $n$, validation dataset $V$ of size $m$, stability parameters $\beta_1$ and $\beta_2$, training privacy budget $\alpha_1$, validation privacy budge $\alpha_2$.
1: $i^* =$ Validate($\Theta, \mathcal{T}, \mathcal{G}, T, V, \beta_1, \beta_2, \alpha_1, \alpha_2$), where the function Validate() is defined in Algorithm 3.
2: Draw $b \sim \mathcal{G}$.
Output: $h = \mathcal{T}(\theta_{i^*}, \alpha_1; T, b)$.
```

5.3.3 *Generalized regularization parameter selection*

Chaudhuri and Vinterbo [5] showed that we can choose the best regularization parameter in a differentially private manner using Algorithm 3 and Algorithm 4, which are restatements of Algorithm 1 and Algorithm 2 in Chaudhuri et al. [4], as long as the validation score function q is $(\beta_1, \beta_2, \delta)$-stable for some $\beta_1, \beta_2, \delta > 0$ with respect to the procedure $\mathcal{T}$, the set of candidate regularization parameters Λ, and the privacy budget ϵ. Chaudhuri and Vinterbo [5] gave conditions under which a validation score function is $(\beta_1, \beta_2, \delta)$-stable when the regularization function is differentiable. As an extension of Chaudhuri and Vinterbo [5], in Theorem 5.1, we specify the conditions under which a validation score function is $(\beta_1, \beta_2, \delta)$-stable when the regularization function is convex and not necessarily differentiable.

For notational convenience, in Theorem 5.1, we embed the regularization parameters into the regularization function to form a vector of candidate regularization functions $r = (r_1, \dots, r_t)$. Then selecting a regularization parameters is equivalent to selecting a linear combination of r_i's in r. For example, with $r(\theta) = \left(\frac{\lambda_1}{2}\|\theta\|_2^2, \dots, \frac{\lambda_k}{2}\|\theta\|_2^2\right)$, choosing a parameter from $\Lambda = \{e_1, \dots, e_k\}$, where e_i is a k-dimensional vector that is 1 in the ith entry and 0 everywhere else, results in Theorem 4 in Chaudhuri and Vinterbo [5]. Thus, Theorem 5.1 generalizes Theorem 4 in Chaudhuri and Vinterbo [5].

Theorem 5.1. *Let* $r = (r_1, \dots, r_t)$ *be a vector of convex regularization functions with* $r_i : \mathbb{R}^s \to \mathbb{R}$ *that are minimized at 0. Let* $\Lambda = \{\lambda_1, \dots, \lambda_k\}$ *be a collection of regularization vectors, where* λ_i *is a* t*-dimensional vector of 0's and 1's. Let* T *denote the training dataset and let* V *denote the validation dataset. Denote by* $c_{min} := \sup_c\{\forall\lambda \in \Lambda, \lambda^T r$ *is* c*-strong convex*$\}$*. Let* $h(\theta; d)$ *be a validation score function that is non-negative and* ψ*-Lipschitz in* θ*. Denote* $\max_{d\in\mathcal{D},\theta\in\mathbb{R}^s} h(\theta; d)$ *by* h^**. Let* $l(\theta; d)$ *be a convex loss function that is* γ*-Lipschitz in* θ*. Finally, let* $\xi \in \mathbb{R}$ *such that* $\mathbb{P}(\|b\|_2 > \xi) \leqslant \delta/k$ *for some* $\delta \in (0, 1)$*. Then the validation score* $q(\theta; V) = -\frac{1}{m}\sum_{d\in V} h(\theta; d)$ *is* $(\beta_1, \beta_2, \delta/k)$*-stable with respect to* $\mathcal{T}$*,* ϵ *and* Λ*, where*

$$\mathcal{T}(\lambda, \epsilon; T, l, r, b) := \arg\min_\theta L(\theta; \lambda, \epsilon),$$

with

$$L(\theta; \lambda, \epsilon) = \frac{1}{n}\sum_{d\in T} l(\theta; d) + \lambda^T r(\theta) + \frac{\max\{0,\ c^* - c_{\min}\}}{2}\|\theta\|_2^2 + \frac{\phi}{\epsilon n} b^T\theta,$$

$$\beta_1 = \frac{2\gamma\psi}{\max\{c^*, c_{min}\}}, \qquad \beta_2 = \min\left\{h^*, \frac{\psi}{\max\{c^*, c_{min}\}}\left(\gamma + \frac{\phi\xi}{\epsilon n}\right)\right\}.$$

Proof. See Appendix A.2.1. □

The term $\frac{\max\{0, c^*-c_{\min}\}}{2}\|\theta\|_2^2$ in Theorem 5.1 ensures that $L(\theta;\lambda,\epsilon)$ is at least c^*-strongly convex. This is a necessary condition for ensuring that the objective function perturbation algorithm (Algorithm 5) is differentially private. The value of ξ in Theorem 5.1 depends on the distribution of the perturbation noise b. In Section 5.3.4, we analyze two different distributions for the perturbation noise.

Algorithm 5 is a reformulation of Algorithm 1 in Kifer et al. [18], a differentially private generalized objective function optimization algorithm. Algorithm 5 incorporates the use of different perturbation noise distributions described in Section 5.3.4. Moreover, the objective function $L(\theta; D, \lambda, b)$ in Algorithm 5 is formulated in such a way that it is compatible with the regularization parameter selection procedure described in Theorem 5.1.

Theorem 5.2. *Algorithm 5 is ϵ-differentially private.*

Proof. See A.2.2. □

Algorithm 5 Generalized objective perturbation mechanism

Input: Dataset $D = \{d_1, \dots, d_n\}$; a convex domain $\Theta \subset \mathbb{R}^s$; privacy parameter ϵ; c^*-strongly convex regularizer r; convex loss function $l(\theta; d)$ with rank-1 continuous Hessian $\nabla^2 l(\theta; d)$, an upper bound c on the maximal singular value of $\nabla^2 l(\theta; d)$ and upper bounds κ_j on $\|\nabla l(\theta; d)\|_j$ for $j \in \{1, 2\}$ that hold for all $d \in D$ and all $\theta \in \Theta$. It is also required that $\phi \geqslant 2\kappa_j$ and $c^* \geqslant \frac{c}{n(e^{\epsilon/4}-1)}$.

Output: A differentially-private parameter vector θ^*.

```
1: Sample $b \in \mathbb{R}^s$ according to noise distribution $B_j$, $j \in \{1, 2\}$.
2: return $\theta^* = \arg\min_\theta L(\theta; D, b)$, where
```

$$L(\theta; D, b) = \frac{1}{n}\sum_{d \in D} l(\theta; d) + r(\theta) + \frac{\phi}{\epsilon n} b^T \theta.$$

5.3.4 *Distributions for the perturbation noise*

In this section, we analyze two perturbation noise distributions used in Algorithm 5. We show that both perturbation noise distributions can be generated easily. We also compare the performance of Algorithm 5 under different noise distributions.

Let B_2 be a random variable with density function

$$f_{B_2}(b) \propto \exp\left(-\frac{\|b\|_2}{2}\right).$$

Chaudhuri et al. [4] and Kifer et al. [18] showed that using B_2 as perturbation noise in the procedure $\mathcal{T}(\lambda, \epsilon; T, l, r, B_2)$ produces ϵ-differentially private regression coefficients. In Proposition 5.3, we describe an efficient method for generating such perturbation noise. Furthermore, we show that under slightly stronger conditions the procedure $\mathcal{T}(\lambda, \epsilon; T, l, r, B_1)$ is differentially private when we use perturbation noise B_1 with density function

$$f_{B_1}(b) \propto \exp\left(-\frac{\|b\|_1}{2}\right),$$

which is simpler to generate than B_2.

Proposition 5.3. *The random variable* $X = \frac{W}{\|W\|_2}Y$, *where* $W \sim \mathcal{N}(0, I_s)$ *and* $Y \sim \chi^2(2s)$, *has density function* $f_X(x) \propto \exp\left(-\frac{\|x\|_2}{2}\right)$.

Proof. See Appendix A.2.3. □

According to Proposition 5.3, $B_2 \sim \frac{W_s}{\|W_s\|_2}Y_{2s}$, with $W_s \sim \mathcal{N}(0, I_s)$ and $Y_{2s} \sim \chi^2(2s)$. On the other hand, B_1 can be viewed as the joint distribution of s independent Laplace random variables with mean $= 0$ and scale $= 2$. In order to specify the stability parameter β_2 in Theorem 5.1, we need to find $\xi \in \mathbb{R}$ such that $P(\|b\|_2 \geqslant \xi) \leqslant \delta/k$. The following propositions enable us to find ξ for the perturbation noises B_1 and B_2.

Proposition 5.4. $\mathbb{P}\left(\|B_1\|_1 \geqslant 2s\log(sk/\delta)\right) \leqslant \delta/k$.

Proof. See Lemma 17 in Chaudhuri and Vinterbo [5]. □

Proposition 5.5. $\mathbb{P}\left(\|B_2\|_2 \geqslant \left(\sqrt{s}+\sqrt{\log(k/\delta)}\right)^2 + \log(k/\delta)\right) \leqslant \delta/k.$

Proof. Note that $\|B_2\|_2 = \|\frac{W_s}{\|W_s\|_2}Y_{2s}\|_2 = Y_{2s}$, where $Y_{2s} \sim \chi^2(2s)$. The proof is completed by invoking Lemma 1 in Laurent and Massart [19]. □

Because $\mathbb{P}(\|B_1\|_1 \geqslant \xi) \geqslant \mathbb{P}(\|B_1\|_2 \geqslant \xi)$, Proposition 5.4 and Proposition 5.5 enable us to find $\xi \in \mathbb{R}$ such that $\mathbb{P}(\|b\|_2 \geqslant \xi) \leqslant \delta/k$. When the density function of b is $f(b) \propto \exp\left(\frac{\|b\|_1}{2}\right)$, $\xi = 2s\log(sk/\delta)$ by Proposition 5.4. On the other hand, when the density function of b is $f(b) \propto \exp\left(\frac{\|b\|_2}{2}\right)$, $\xi = \left(\sqrt{s}+\sqrt{\log(k/\delta)}\right)^2 + \log(k/\delta)$ by Proposition 5.5.

5.3.4.1 *Comparison of performance under different noise distributions*

Note that we can always bound $\|\nabla l(\theta; d)\|_2$ above by $\|\nabla l(\theta; d)\|_1$ and hence $\kappa_2 \leqslant \kappa_1$ in Algorithm 5. However, as we show in this section, results from Algorithm 5 are more accurate when we sample noise from B_1 instead of B_2. To compare the performance of Algorithm 5 under noise sampled from B_1 and B_2, we follow the algorithm performance analysis in Chaudhuri et al. [4] and analyze $\mathbb{P}(J(\theta_b) - J(\theta^*) > c)$, where

$$J(\theta) \;=\; \frac{1}{n}\sum_{d\in D} l(\theta; d) + r(\theta)$$

with l and r as defined in Algorithm 5, $\theta^* = \arg\min_\theta J(\theta)$, and

$$\theta_b = \arg\min_\theta \left[J(\theta) + \frac{\phi}{\epsilon n} b^T\theta\right] \;=\; \arg\min_\theta L(\theta; b).$$

In other words, $J(\theta_b) - J(\theta^*)$ measures how much the objective function deviates from the optimum due to the added noise. Given random noise $b \in \mathbb{R}^s$,

$$J(\theta_b) + \frac{\phi}{\epsilon n} b^T\theta_b \leqslant J(\theta^*) + \frac{\phi}{\epsilon n} b^T\theta^*.$$

Hence,

$$J(\theta_b) - J(\theta^*) \leqslant \frac{\phi}{\epsilon n} b^T(\theta^* - \theta_b) \leqslant \frac{\phi}{\epsilon n}\|b\|_2\|\theta^* - \theta_b\|_2.$$

Let E denote the event that $\{\|b\|_2 \leqslant \xi\}$, where $\xi = \frac{\epsilon n}{\phi}\sqrt{\lambda c}$. When E holds, $\frac{\phi}{\epsilon n} b^T\theta$ is $\frac{\phi\xi}{\epsilon n}$-Lipschitz. Hence, with $G(\theta) = J(\theta)$ λ-strongly convex, $g_1(\theta) = \frac{\phi}{\epsilon n} b^T\theta$ and $g_2 = 0$, we can invoke Lemma A.1 to obtain

$$\|\theta^* - \theta_b\|_2 \leqslant \frac{\phi\xi}{\lambda\epsilon n}.$$

Therefore, when E holds,

$$J(\theta_b) - J(\theta^*) \leqslant \frac{\phi}{\epsilon n}\|b\|_2\|\theta^* - \theta_b\|_2 \leqslant \frac{\phi}{\epsilon n}\ \xi\ \frac{\phi\xi}{\lambda\epsilon n} = c.$$

Thus $\mathbb{P}(J(\theta_b) - J(\theta^*) > c) \leqslant 1 - \mathbb{P}(E) = \mathbb{P}(\|b\|_2 > \xi)$ when the random noise b is sampled from B_1 or B_2. $\|B_1\|_1$ is the sum of s independent exponential random variables with mean $= 2$ and thus $\|B_1\|_1 \sim Gamma(s, 2)$. On the other hand, $\|B_2\|_2 \sim \chi^2(2s)$. But because $\chi^2(2s) \sim Gamma(s, 2)$, thus

$$\mathbb{P}(\|B_1\|_2 > \xi) \leqslant \mathbb{P}(\|B_1\|_1 > \xi) = \mathbb{P}(\|B_2\|_2 > \xi).$$

Therefore, sampling the perturbation noise from B_1 in Algorithm 5 produces more accurate results.

5.3.5 *Example: differentially private elastic-net penalized logistic regression*

In this section we demonstrate how to apply the results in Section 5.3.3 to elastic-net penalized logistic regression. The logistic loss function $l(\theta; x, y)$ is given by

$$l(\theta; x, y) \ = \ \log\left(1 + \exp(-y\,\theta^T x)\right),$$

where $y \in \{-1, 1\}$. The first and second derivatives with respect to θ are

$$\begin{aligned}\nabla l(\theta; x, y) \ &= \ -\frac{1}{1 + \exp(y\,\theta^T x)} yx \\ \nabla^2 l(\theta; x, y) \ &= \ \frac{1}{1 + \exp(-y\,\theta^T x)} \frac{1}{1 + \exp(y\,\theta^T x)} xx^T.\end{aligned}$$

It can be easily shown that the logistic loss function satisfies the following properties: (i) $l(\theta; x, y)$ is convex; (ii) $\nabla^2 l(\theta; x, y)$ is continuous; and (iii) $\nabla^2 l(\theta; x, y)$ is a rank-1 matrix.

We denote by $\|M\|_1$ the nuclear norm of the matrix M and we choose κ so that $\|x\|_j \leqslant \kappa$ for all x, where $j \in \{1, 2\}$. Then

$$\begin{aligned}\|\nabla^2 l(\theta; x, y)\|_1 \ &\leqslant \ \|xx^T\|_1 \ = \ \|x\|_2^2 \ \leqslant \ \|x\|_j^2 \ \leqslant \ \kappa^2, \quad \text{for } j \in \{1, 2\}, \\ \|\nabla l(\theta; x, y)\|_j \ &\leqslant \ \|x\|_j \ \leqslant \ \kappa,\end{aligned}$$

Thus we can apply Algorithm 5 to output differentially private coefficients for elastic-net penalized logistic regression. Moreover, the logistic loss function satisfies the conditions in Theorem 5.1 because $l(\theta; x, y)$ is Lipschitz, as there exists a parameter θ^* such that, for any θ_1 and θ_2,

$$|l(\theta_1; x, y) - l(\theta_2; x, y)| \leqslant \|\nabla l(\theta^*; x, y)\|_2 \|\theta_1 - \theta_2\|_2 \leqslant \kappa \|\theta_1 - \theta_2\|_2.$$

Thus we can apply the stability argument in Theorem 5.1 to select the best regularization parameters in a differentially private way. In Section 5.4 we show how well this method performs on a GWAS dataset.

5.4 An application to elastic-net penalized logistic regression

We now evaluate the performance of the proposed method based on a GWAS dataset. We analyze a binary phenotype such as a disease. Each SNP can take the values 0, 1, or 2. This represents the number of minor alleles at that site. A large SNP dataset is freely available from the HapMap project({`http://hapmap.ncbi.nlm.nih.gov/`}). It consists of SNP data from 4 populations of 45 to 90 individuals each, but does not contain any phenotypic information about the individuals. HAP-SAMPLE [36] can be used to generate SNP genotypes for cases and controls by resampling from HapMap. This ensures that the simulated data show linkage disequilibrium (i.e., correlations among SNPs) and minor allele frequencies similar to real data.

For our analysis we use the simulations from Malaspinas and Uhler [21]. The simulated datasets consist of 400 cases and 400 controls each with about 10,000 SNPs per individual (SNPs were typed with the Affymetrix CHIP on chromosome 9 and chromosome 13 of the Phase I/II HapMap data). For each dataset two SNPs with a given minor allele frequency (MAF) were chosen to be causative. We will analyze the results for minor allele frequency (MAF) $= 0.25$. The simulations were performed under the multiplicative effects model: Denoting the two causative SNPs by X and Y and the disease status by D (i.e., $X, Y \in$

$\{0, 1, 2\}$ and $D \in \{-1, 1\}$, where 1 describes the diseased state), then the multiplicative effects model can be defined through the odds of having a disease:

$$\frac{\mathbb{P}(D = 1 \mid X, Y)}{\mathbb{P}(D = -1 \mid X, Y)} = \epsilon\, \alpha^X \beta^Y \delta^{XY}.$$

This model corresponds to a log-linear model with interaction between the two SNPs. For our simulations we chose $\epsilon = 0.64$, $\alpha = \beta = 0.91$ and $\delta = 2.73$. This results in a sample disease prevalence of 0.5 and effect size of 1, which are typical values for association studies. See Malaspinas and Uhler [21] for more details.

In the first step, we screen all SNPs and select a subset of SNPs with the highest χ^2-scores based on a simple χ^2-test for association between each single SNP and the phenotype. Various approaches for performing the screening in a differentially private manner were discussed and analyzed in Uhler et al. [29], Johnson and Shmatikov [17], and Yu et al. [37]; We concentrated on the second step and did not employ the differentially private screening approaches in this paper. The second step of the two-step procedure consists of performing penalized logistic regression with elastic-net regularization on the selected subset of SNPs and choosing the best regularization parameters in a differentially private manner. In the following, we analyze the statistical utility of the second step and show how accurately our end-to-end differentially private penalized logistic regression method is able to detect the causative SNPs and their interaction.

The elastic-net penalty function has the form $\frac{1}{2}\lambda(1-\alpha)\ell_2 + \lambda\alpha\ell_1$, where α controls the sparsity of the resulting model and λ controls the extent to which the elastic-net penalty affects the loss function. In the simulation, we apply a threshold criterion to the terms in the model so that we exclude from the model the ith term if its regression coefficient, θ_i, satisfies $|\theta_i| / \max_i\{|\theta_i|\} < r$, where $\max_i\{|\theta_i|\}$ is the largest coefficient in absolute value and r is a thresholding ratio, which we set to 0.01.

In our experiments, we selected $M = 5$ SNPs with the highest χ^2-scores, which include the two causative SNPs, for further analysis. We denote by ϵ the privacy budget, by α the sparsity parameter in the elastic-net penalty function, and by "*convex_min*" the condition of strong convexity imposed on the objective function (see Theorem 5.1). Note that *convex_min* is a function of M and ϵ. For elastic-net with α fixed, we need the smallest candidate parameter $\lambda_{min} \geqslant$ *convex_min*$/(1-\alpha)$.

In Figure 5.1, we analyze the sensitivity of our method. For different sparsity parameters α and different privacy budgets ϵ, which determine *convex_min* given a fixed M, we show how often, out of 100 simulations each, our algorithm recovered the interaction term (leftmost bar in red), the main effects scaled by a factor of $1/2$ to account for the two main effects (middle bar in green) and all effects, i.e. the interaction effect and the two main effects (rightmost bar in blue). As the privacy budget ϵ increases, the amount of noise added to the regression problem decreases, and hence the frequency of selecting the correct effects in the regression analysis increases. The plots also show that as the sparsity parameter α increases, the frequency of selecting the correct terms decreases.

In Figure 5.2 we analyze the specificity of our method. For different sparsity parameters α and different strong convexity conditions *convex_min*, we show how often, out of 100 simulations each, our algorithm did not include any additional effects in the selected model. As α increases, the selected model becomes sparser and the algorithm is hence less likely to wrongly include additional effects. We also observe that as *convex_min* decreases, the specificity increases. This can be explained by how we choose the candidate parameters λ, namely as multiples of the smallest allowed value for λ, which is *convex_min*$/(1-\alpha)$. When λ is small, the effect of the penalty terms diminishes, and we are essentially performing a regular logistic regression, which does not produce sparse models.

In Figure 5.3, we plotted the results of non-private penalized logistic regression with elastic-net penalty to contrast Figure 5.1 and Figure 5.2. The results of the non-private penalized logistic regression is indirectly related to ϵ because the choice of the smallest regularization parameter λ is bounded below by *convex_min*$/(1-\alpha)$ and *convex_min* is a function of ϵ. We can observe from Figure 5.3 that when the regularization parameter λ is large (i.e., *convex_min* $\geqslant 1.58$), the regression analysis screens out all effects. Hence, the sensitivity is 0 and the specificity is 1. When λ is small (i.e., *convex_min* $\leqslant 0.18$), the amount of regularization also becomes marginal, and we begin to see that the sensitivity increases but the specificity decreases. Figure 5.3 shows that we can identify the correct model when $\alpha = 0.1$ and *convex_min* $= 0.18$. In contrast, when we use the same α and *convex_min* for differentially private regressions, Figure 5.1 shows that we can obtain a good sensitivity result, but Figure 5.2 shows that the specificity result for this choice is poor.

Figure 5.1: Sensitivity analysis for different sparsity parameters α, privacy budgets ϵ, and strong convexity conditions *convex_min* when the top 5 SNPs are used for the analysis: the red (leftmost) bar shows how often, out of 100 simulations each, the algorithm recovered the interaction term, the green (middle) bar corresponds to the main effects scaled by a factor of $1/2$ and the rightmost (blue) bar corresponds to all effects, i.e. 2 main effects and 1 interaction effect.

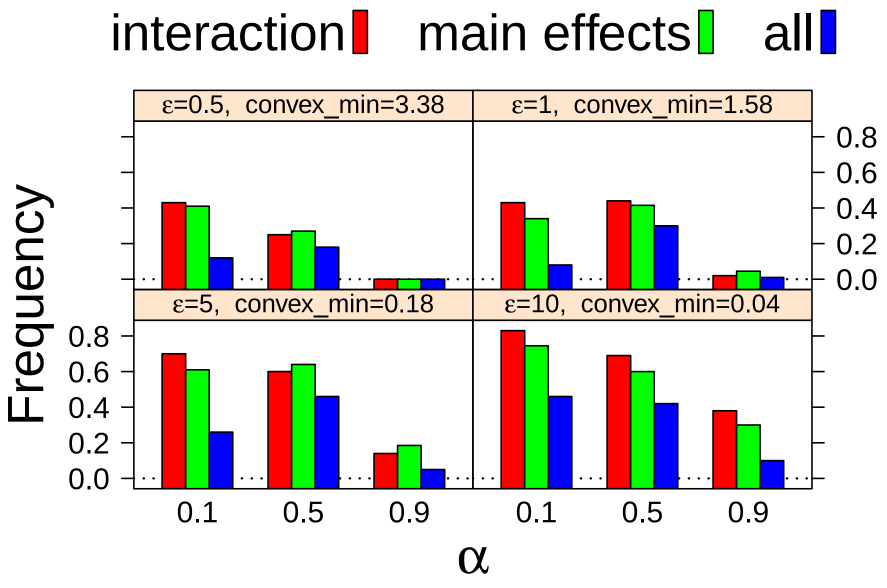


Figure 5.2: Specificity analysis for different sparsity parameters α and strong convexity conditions *convex_min*: the plot shows how often, out of 100 simulations each, our algorithm did not include any additional effects in the selected model.

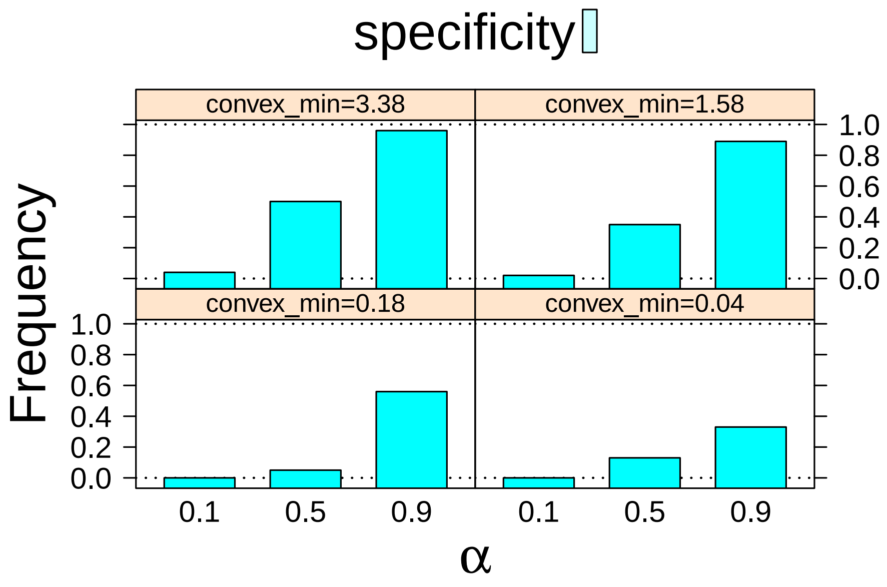
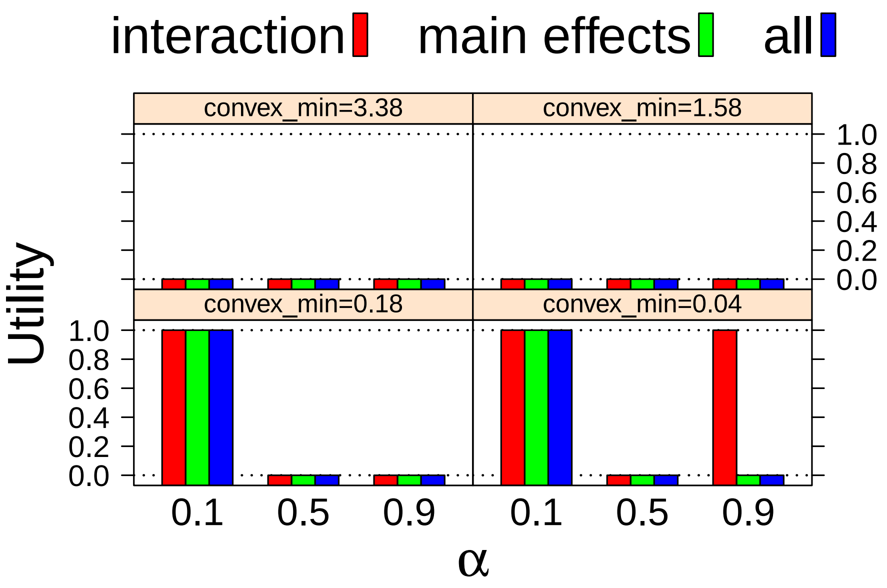
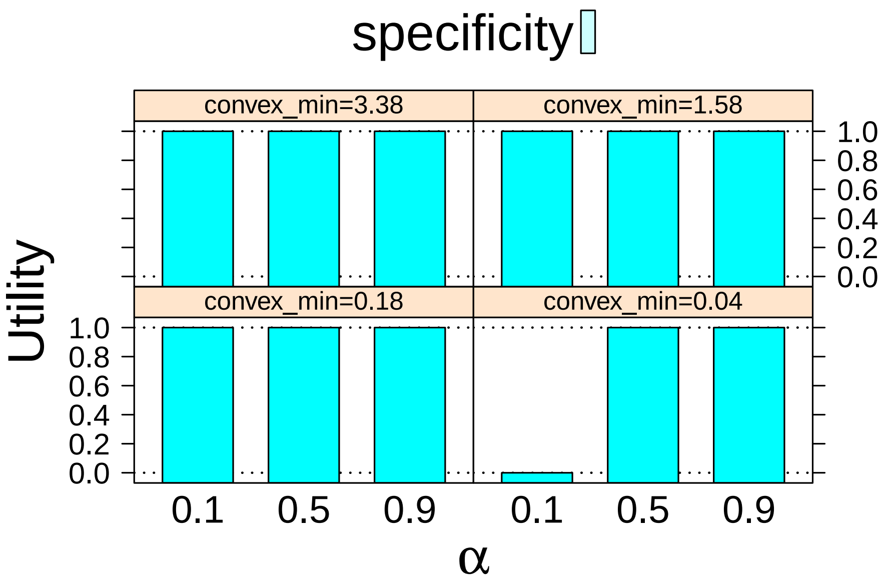


Figure 5.3: Results of non-private logistic regression with elastic-net penalty. Figure 5.4a and Figure 5.4b would be compared with Figure 5.1 and Figure 5.2, respectively.

(a) Sensitivity



(b) Specificity



Part IV

CONCLUSIONS

6

CONCLUSIONS AND FUTURE WORK

6.1 The Homer et al. [14] attack

In Section 2.2, we provide an in-depth analysis of the Homer et al. [14] attack and show that the effectiveness of the attack is highly dependent on the validity of several key assumptions. By analyzing the results of applying the Homer et al. [14] attack to a real human GWAS dataset, we show that these assumptions do not hold in a real human GWAS. We therefore conclude that the attack has limited applicability.

The key assumptions of the Homer et al. [14] attack are (1) that the individual of interest, Y, the case group, F, and the control group, G, are sampled from the same underlying population, and (2) that all the SNPs analyzed are in linkage equilibrium. In our application of Homer et al. [14] to a real human GWAS dataset, we use the case data of the GWAS as F and the control data of the GWAS as G, and show that F and G are not from the same underlying population. Therefore, the first assumption does not hold in a real human GWAS setting.

We show that the second assumption does not hold either when a large number—typically hundreds of thousands—of SNPs are released in a GWAS. While we can reduce the effect of linkage disequilibrium by releasing a smaller subset of randomly selected SNPs, we show that the power of the attack will also be reduced as a result. Future analysis of the Homer et al. [14] attack can look at the effectiveness of the attack when only the most relevant SNPs are released.

As the number of subjects in recent GWAS's increase to tens of thousands, the apparent limitations of the Homer et al. [14] attack and similar attacks are in fact amplified. Intuitively, the reason that the Homer et al. [14] attack is able to resolve an individual from a pool of individuals using only aggregate statistics (e.g., minor allele frequencies) is that the individual's data is still to some extent represented in the aggregate statistics. Thus, when the aggregate statistics are pooled from more subjects, we expect the statistical power of these attacks to decrease. Future work should explore the effectiveness of the Homer et al. [14] attack given different sample sizes.

6.2 Privacy preserving release of the most relevant SNPs

A number of authors have argued that it is possible to use aggregate data to compromise the privacy of individual-level information collected in GWAS databases. In Chapter 4, we have used the concept of differential privacy and built on the approach in Uhler et al. [29] to propose new methods of releasing aggregate GWAS data without compromising an individual's privacy.

A key component of the differential privacy approach involves the sensitivity of a released statistic when we replace an observation. In Section 4.3, we have obtained sensitivity results for the χ^2 statistic based on a 2×3 genotype table and the χ^2 statistic based on a 2×2 allelic table, with the genotype table and the allelic table assumed to have positive margins, and arbitrary number of cases and controls. Furthermore, we have shown that sensitivity can be reduced in situations where data for the cases (or the controls) are known to the attacker. Nevertheless, we have also shown that the reduction in sensitivity is insignificant in typical GWAS settings, in which the number of cases is large.

The differentially private method that uses Hamming distance score as score function has been shown to have the best performance among all methods examined in the chapter when the privacy budget is small. However, calculating Hamming distance score is not a computationally feasible task, and studies [17, 37] that used the differentially private method substituted Hamming distance score with a proxy score. In Section 4.4, we device an accurate and computationally efficient method to calculate Hamming distance score. The graphical interpretation of χ^2 statistic proves instrumental in our discovery of the method.

Finally, we have shown that a risk-utility analysis of the differentially private methods allows us to understand the trade-off between privacy budget and statistical utility, and therefore helps us decide on the appropriate level of privacy guarantee for the released data.

6.2.1 *Future work*

- The χ^2 statistic of a 2×3 genotype table and the χ^2 statistic of a 2×2 allelic table are both frequently used in GWAS. We have shown in Section 4.4 how to calculate the Hamming distance score based on the χ^2 statistic of a 2×2 allelic table. Future work explore how to calculate the Hamming distance score based on the χ^2 statistic of a 2×3 genotype table. The graphical interpretation of the χ^2 statistic based on a 2×2 allelic table proves instrumental in finding the corresponding Hamming distance score. If we can extend the graphical interpretation to the χ^2 statistic of a 2×3 genotype table, we may very well hold the key to finding a method for calculating the Hamming distance score based on the χ^2 statistic of a 2×3 genotype table.
- The Homer et al. [14] attack uses minor allele frequencies (MAFs) to infer whether an individual belongs to the case group, the control group, or neither group in a GWAS. To evaluate whether the differentially private methods developed in Chapter

4 can be used to foil the Homer et al. [14] attack, we can perform one of the following experiments:

1. Given the MAFs of the cases and the controls, and assuming that the Hardy-Weinberg equilibrium holds, we can construct a genotype table or an allelic table for each SNP. We can then calculate the χ^2 statistic of each SNP and use one of the differentially private methods to choose the most relevant SNPs. After a set of most relevant SNPs have been chosen differentially privately, we can then perturb the MAFs of the selected set of SNPs using the Laplace mechanism. For sensitivity results of MAF, see Uhler et al. [29].

2. As was shown in Uhler et al. [29], the sensitivity of MAF can be large. On the other hand, the sensitivity of χ^2 statistic is smaller when it is compared to typical values of χ^2 statistic for significant SNPs. Suppose that the data for the controls are known, and assume that the Hardy-Weinberg equilibrium holds, then the χ^2 statistic based on the 2×2 allelic table corresponds to at most two values of MAF. To see this, suppose that the MAF for the cases is f_{case} and the MAF for the controls, which is assumed to be known, is $f_{control}$. Let R be the total number of cases, S be the total number of controls, and $N = R + S$. Then the χ^2 statistic based on a 2×2 allelic table can be written as

$$\chi^2(f_{case}) = \frac{2N\,(f_{case}S - f_{control}R)^2}{RS(f_{case} + f_{control})2N(2 - f_{case} - f_{control})}.$$

   This result is also discussed in Section 4.4.4.

   The experiment will proceed as follows. Given the MAFs of the cases and the controls, and assuming that the Hardy-Weinberg equilibrium holds, we can construct a genotype table or an allelic table for each SNP. We can then calculate the χ^2 statistic of each SNP and use one of the differentially private methods to choose the most relevant SNPs. After a set of most relevant SNPs have been chosen differentially privately, we can then perturb the χ^2 statistics of the selected set of SNPs using the Laplace mechanism. Sensitivity results of χ^2 statistic have been derived in Section 4.3. Then, following the previous analysis, we release the smallest MAF corresponding to each perturbed χ^2 statistic.

6.3 Privacy preserving release of regression coefficients

Various papers have argued that it is possible to use aggregate genomic data to compromise the privacy of individual-level information collected in GWAS databases. In Chapter 5, we respond to these attacks by proposing a new method to release regression coefficients from association studies that satisfy differential privacy and hence come with privacy guarantees against arbitrary external information.

By extending the approaches in Kifer et al. [18] and Chaudhuri and Vinterbo [5], we have developed a differentially private method that not only solves regression problems with any convex penalty functions, but also handles the selection of regularization parameters by cross-validation. We have also provided the exact form of the random noise used in the objective function perturbation mechanism and showed that the perturbation noise can be efficiently sampled.

By combining the differentially private methods for releasing the most relevant SNPs and the method for solving penalized regression problems, we have developed an end-to-end procedure for analyzing epistasis in a GWAS.

As a special case of a regression problem, we focused on penalized logistic regression with an elastic-net penalty function, a method widely used to perform GWAS analyses and identify disease-causing genes. Our simulation results in have shown that our method is applicable to GWAS data sets and enables us to perform data analysis that preserves privacy and utility. The risk-utility analysis of the tradeoff between privacy (ϵ) and utility (correctly identifying the causative SNPs) helps us decide on the appropriate level of privacy guarantee for the released data. Future work should evaluate the performance of the two-step differentially private procedure using a real GWAS dataset.

6.4 Policy discussion

Current research on differentially private algorithms provides tools to share genetic data while at the same time control the level of privacy protection for genetic study participants. However, how to wield these tools properly in practice remains an open question. The main challenge of using differential privacy is balancing how much privacy and data utility to preserve. Intrinsic to differential privacy is a tuning parameter that controls the level of privacy protection. The tuning parameter correlates with data utility in different ways depending on the nature of the data, the algorithm, and how data utility is defined. For example, we define data utility in Section 4.5.3 as the proportion of the most relevant SNPs recovered after perturbation, whereas Johnson and Shmatikov [17] defined data utility as the difference between the sum of the logarithm of the p-values of the outputted SNPs and that of the original top SNPs. As a result of the difference in the definition of data utility, we observe different relationships between the privacy parameter and data utility.

Understanding the limits of the privacy tuning parameter and how to choose it in a sensible way is one of the key hurdles for making differentially private algorithms useful in practice in GWAS settings and others. There have been promising suggestions on how to solve this problem. One of the suggestions is that we should find the relationship between the privacy parameter ϵ and the payout to genetic study participants for giving up ϵ amount of privacy guarantee, in addition to the relationship between the privacy parameter and data utility. Perhaps then we can better understand what the appropriate level of privacy protection is. Another suggestion is that we correlate the privacy parameter with the probability of foiling a certain statistical attack on genetic databases.

It is important to understand the two seemingly conflicting properties of differential privacy: because of the composition property of differential privacy (e.g., McSherry [23]), the privacy guarantee afforded by ϵ-differential privacy will endure as long as subsequent analyses use only the perturbed data; in the event that another instance of differentially private output is released using the original data, the privacy guarantee for the original data no longer holds. In other words, each time a differential private output is released using the original data, some privacy guarantee is consumed. When the privacy guarantee budget has been used up, nothing based on the original data should be released any more. This is particular relevant to NIH and other data curating agencies. Because access to a particular dataset may have been granted to multiple entities, simply requiring each entity to release differential private results does not suffice to protect the dataset. These agencies will have to be mindful of the collective privacy budget consumption.

Part V

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Erin Austin, Wei Pan, and Xiaotong Shen. "Penalized regression and risk prediction in genome-wide association studies." In: *Statistical Analysis and Data Mining* 6.4 (Aug. 2013).

[2] Raghav Bhaskar, Srivatsan Laxman, Adam Smith, and Abhradeep Thakurta. "Discovering frequent patterns in sensitive data." In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '10*. New York, New York, USA: ACM Press, 2010, p. 503.

[3] Rosemary Braun, William Rowe, Carl Schaefer, Jinghui Zhang, and Kenneth Buetow. "Needles in the haystack: identifying individuals present in pooled genomic data." In: *PLoS Genetics* 5.10 (2009), e1000668.

[4] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. "Differentially private empirical risk minimization." In: *JMLR* 12.7 (Mar. 2011), pp. 1069–1109.

[5] Kamalika Chaudhuri and SA Vinterbo. "A stability-based validation procedure for differentially private machine learning." In: *Advances in Neural Information Processing Systems* (2013), pp. 1–19.

[6] Seoae Cho, Haseong Kim, Sohee Oh, Kyunga Kim, and Taesung Park. "Elastic-net regularization approaches for genome-wide association studies of rheumatoid arthritis." In: *BMC Proceedings* 3.Suppl 7 (2009), S25.

[7] David Clayton. "On inferring presence of an individual in a mixture: a Bayesian approach." In: *Biostatistics* 11.4 (2010), pp. 661–73.

[8] Jennifer Couzin. “Whole-genome data not anonymous, challenging assumptions.” In: *Science* 321.5894 (2008), p. 1278.

[9] Bernie Devlin and Kathryn Roeder. “Genomic control for association studies.” In: *Biometrics* 55.4 (1999), pp. 997–1004.

[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. “Calibrating noise to sensitivity in private data analysis.” In: *Theory of Cryptography* (2006), pp. 1–20.

[11] E. Gómez, M.a. Gomez-Viilegas, and J.M. Marín. “A multivariate generalization of the power exponential family of distributions.” In: *Communications in Statistics - Theory and Methods* 27.3 (Jan. 1998), pp. 589–600.

[12] Melissa Gymrek, Amy L. McGuire, David Golan, Eran Halperin, and Yaniv Erlich. “Identifying personal genomes by surname inference.” In: *Science* 339.6117 (2013), pp. 321–4.

[13] William G Hill and A Robertson. “Linkage disequilibrium in finite populations.” In: *Theoretical and Applied Genetics* 38.6 (June 1968), pp. 226–31.

[14] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. “Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays.” In: *PLoS Genetics* 4.8 (2008), e1000167.

[15] Hae Kyung Im, Eric R Gamazon, Dan L Nicolae, and Nancy J Cox. “On sharing quantitative trait GWAS results in an era of multiple-omics data and the limits of genomic privacy.” In: *American Journal of Human Genetics* 90.4 (2012), pp. 591–8.

[16] Kevin B Jacobs, Meredith Yeager, Sholom Wacholder, David Craig, Peter Kraft, David J Hunter, Justin Paschal, Teri A Manolio, Margaret Tucker, Robert N Hoover, Gilles D Thomas, Stephen J Chanock, and Nilanjan Chatterjee. “A new statistic and

its power to infer membership in a genome-wide association study using genotype frequencies." In: *Nature Genetics* 41.11 (2009), pp. 1253–7.

[17] Aaron Johnson and Vitaly Shmatikov. "Privacy-preserving data exploration in genome-wide association studies." In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2013, pp. 1079–1087.

[18] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. "Private convex empirical risk minimization and high-dimensional regression." In: *Proceedings of Journal of Machine Learning Research - Proceedings Track*. Vol. 23. 2012, pp. 25.1–25.40.

[19] B Laurent and P Massart. "Adaptive estimation of a quadratic functional by model selection." In: *Annals of Statistics* 28.5 (2000), pp. 1302–1338.

[20] Thomas Lumley and Kenneth Rice. "Potential for revealing individual-level information in genome-wide association studies." In: *JAMA* 303.7 (Feb. 2010), pp. 659–60.

[21] AS Malaspinas and Caroline Uhler. "Detecting epistasis via Markov bases." In: *Journal of Algebraic Statistics* 2.1 (2010), pp. 36–53.

[22] Nicholas Masca, Paul R Burton, and Nuala A Sheehan. "Participant identification in genetic association studies: improved methods and practical implications." In: *International Journal of Epidemiology* 40.6 (2011), pp. 1629–42.

[23] Frank McSherry. "Privacy integrated queries." In: *Proceedings of the 35th SIGMOD International Conference on Management of Data - SIGMOD '09*. New York, New York, USA: ACM Press, 2009, p. 19.

[24] Frank McSherry and Kunal Talwar. "Mechanism design via differential privacy." In: *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)* (Oct. 2007), pp. 94–103.

[25] Mee Young Park and Trevor Hastie. "Penalized logistic regression for detecting gene interactions." In: *Biostatistics* 9.1 (2008), pp. 30–50.

[26] Joshua Sampson and Hongyu Zhao. “Identifying individuals in a complex mixture of DNA with unknown ancestry.” In: *Statistical Applications in Genetics and Molecular Biology* 8.1 (Jan. 2009), Article 37.

[27] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. “Genomic privacy and limits of individual detection in a pool.” In: *Nature Genetics* 41.9 (2009), pp. 965–7.

[28] Aleksandra Slavkovic and Fei Yu. “Genomic privacy: risk and protection methods.” In: *CHANCE* 28.2 (2015), Submitted.

[29] Caroline Uhler, Aleksandra B. Slavkovic, and Stephen E. Fienberg. “Privacy-preserving data sharing for genome-wide association studies.” In: *Journal of Privacy and Confidentiality* 5.1 (2013), pp. 137–166.

[30] Peter M Visscher and William G Hill. “The limits of individual identification from sample allele frequencies: theory and statistical analysis.” In: *PLoS Genetics* 5.10 (2009), e1000628.

[31] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. “Learning your identity and disease from research papers.” In: *Proceedings of the 16th ACM Conference on Computer and Communications Security - CCS '09*. New York, New York, USA: ACM Press, 2009, p. 534.

[32] Larry Wasserman and Shuheng Zhou. “A statistical framework for differential privacy.” In: *Journal of the American Statistical Association* 105.489 (Mar. 2010), pp. 375–389.

[33] Wellcome Trust Case Control Consortium. “Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls.” In: *Nature* 447.7145 (2007), pp. 661–78.

[34] Danielle Welter, Jacqueline MacArthur, Joannella Morales, Tony Burdett, Peggy Hall, Heather Junkins, Alan Klemm, Paul Flicek, Teri Manolio, Lucia Hindorff, and

Helen Parkinson. "The NHGRI GWAS Catalog, a curated resource of SNP-trait associations." In: *Nucleic Acids Research* 42.Database issue (Jan. 2014), pp. D1001–6.

[35] Xiaoquan Wen and Matthew Stephens. "Using linear predictors to impute allele frequencies from summary or pooled genotype data." In: *Annals of Applied Statistics* 4.3 (2010), pp. 1158–1182.

[36] Fred A Wright, Hanwen Huang, Xiaojun Guan, Kevin Gamiel, Clark Jeffries, William T Barry, Fernando Pardo-Manuel de Villena, Patrick F Sullivan, Kirk C Wilhelmsen, and Fei Zou. "Simulating association studies: a data-based resampling method for candidate regions or whole genome scans." In: *Bioinformatics* 23.19 (Oct. 2007), pp. 2581–8.

[37] Fei Yu, Stephen E. Fienberg, Aleksandra B Slavković, and Caroline Uhler. "Scalable privacy-preserving data sharing methodology for genome-wide association studies." In: *Journal of Biomedical Informatics* 50C (Feb. 2014), pp. 133–141.

[38] Fei Yu and Zhanglong Ji. "Scalable privacy-preserving data sharing methodology for genome-wide association studies: an application to iDASH healthcare privacy protection challenge." In: *BMC Medical Informatics and Decision Making* 14 Suppl 1 (Dec. 2014), S3.

[39] Fei Yu, Michal Rybar, Caroline Uhler, and Stephen E. Fienberg. "Differentially-private logistic regression for detecting multiple-SNP association in GWAS databases." In: *Privacy in Statistical Databases*. Ed. by Josep Domingo-Ferrer. Vol. 8744. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 170–184.

[40] Elias A Zerhouni and Elizabeth G Nabel. "Protecting aggregate genomic data." In: *Science* 322.5898 (2008), p. 44.

[41] Gang Zheng, Jungnam Joo, and Yaning Yang. "Pearson's test, trend test, and MAX are all trend tests with different types of scores." In: *Annals of Human Genetics* 73.2 (Mar. 2009), pp. 133–40.

[42] Xiaoyong Zhou, Bo Peng, Y Li, and Yangyi Chen. “To release or not to release: evaluating information leaks in aggregate human-genome data.” In: *ESORICS*. Springer, 2011, pp. 607–627.

Part VI

APPENDICES

A

PROOFS

A.1 Privacy preserving release of the most relevant SNPs

A.1.1 *Algorithms*

A.1.1.1 *Proof of Theorem 4.2*

Following the notation in McSherry and Talwar [24], we define the random variable of sampling a single SNP, ε_q^ϵ, by

$$\begin{aligned}\Pr(\varepsilon_q^\epsilon(D) = i) &\propto \exp\left(\frac{\epsilon q(D,i)}{2\Delta_q}\right)\mu(i) \\ &\propto \exp\left(\frac{\epsilon q(D,i)}{2s}\right)\end{aligned}$$

where $q(D,i)$ is the score for SNP$_i$, s is the sensitivity for the scoring function $q(D,i)$, and $\mu(i) = 1/M$ is constant. We also define

$$q_B(D,i) = \begin{cases} \text{score of the SNP}_i & \text{if } i \notin B \\ -\infty & \text{if } i \in B \end{cases}.$$

where B is a set of SNPs and q_B denotes the scoring function given that the SNPs in B have been sampled and thus have 0 sampling probability in subsequent sampling steps. Note that

$$\begin{aligned}
\Pr(\varepsilon_{q_B}^{\epsilon}(D) = i, i \notin B) &= \frac{\exp\left(\frac{\epsilon q_B(D,i)}{2s}\right)}{\sum_{j \notin B} \exp\left(\frac{\epsilon q_B(D,j)}{2s}\right)} \\
&\leqslant \frac{\exp\left(\frac{\epsilon[q_B(D',r)+s]}{2s}\right)}{\sum_{j \notin B} \exp\left(\frac{\epsilon[q_B(D',r)-s]}{2s}\right)} \\
&= e^{\epsilon} \; \Pr(\varepsilon_{q_B}^{\epsilon}(D') = i, i \notin B).
\end{aligned}$$

Let σ denote a permutation of $\mathcal{S}$.

$$\begin{aligned}
\Pr(\text{sampling } \mathcal{S}|D) &= \sum_{\sigma \in \sigma(S)} \Pr(\varepsilon_q^{\epsilon/(K)}(D) = \sigma(1)) \prod_{i=2}^{M} \Pr(\varepsilon_{q_{\{\sigma(j), j<i\}}}^{\epsilon/(K)}(D) = \sigma(i)) \\
&\leqslant \sum_{\sigma \in \sigma(S)} \left\{ e^{\epsilon/(K)} \; \Pr(\varepsilon_q^{\epsilon/(K)}(D') = \sigma(1)) \right\} \\
&\qquad \prod_{i=2}^{K} \left\{ e^{\epsilon/(K)} \; \Pr(\varepsilon_{q_{\{\sigma(j), j<i\}}}^{\epsilon/(K)}(D') = \sigma(i)) \right\} \\
&= e^{\epsilon} \Pr(\text{sampling } \mathcal{S}|D').
\end{aligned}$$

$\square$

A.1.2 *Sensitivity of χ^2 statistics*

A.1.2.1 *Proof of Theorem 4.6*

The Pearson χ^2-statistic can be written as

$$\begin{aligned}
Y = \;& \frac{\left(r_0 - \frac{n_0 R}{N}\right)^2}{\frac{n_0 R}{N}} + \frac{\left(r_1 - \frac{n_1 R}{N}\right)^2}{\frac{n_1 R}{N}} + \frac{\left(r_2 - \frac{n_2 R}{N}\right)^2}{\frac{n_2 R}{N}} \\
&+ \frac{\left(s_0 - \frac{n_0 S}{N}\right)^2}{\frac{n_0 S}{N}} + \frac{\left(s_1 - \frac{n_1 S}{N}\right)^2}{\frac{n_1 S}{N}} + \frac{\left(s_2 - \frac{n_2 S}{N}\right)^2}{\frac{n_2 S}{N}} \\
= \;& (r_0 N - n_0 R)^2 \left(\frac{1}{n_0 RN} + \frac{1}{n_0 SN}\right) \\
&+ (r_1 N - n_1 R)^2 \left(\frac{1}{n_1 RN} + \frac{1}{n_1 SN}\right) \\
&+ (r_2 N - n_2 R)^2 \left(\frac{1}{n_2 RN} + \frac{1}{n_2 SN}\right) \\
= \;& \frac{(r_0 N - n_0 R)^2}{n_0 RS} + \frac{(r_1 N - n_1 R)^2}{n_1 RS} + \frac{(r_2 N - n_2 R)^2}{n_2 RS} \\
= \;& \frac{r_0^2 N^2}{n_0 RS} - \frac{2 r_0 N}{S} + \frac{n_0 R}{S} \\
&+ \frac{r_1^2 N^2}{n_1 RS} - \frac{2 r_1 N}{S} + \frac{n_1 R}{S} \\
&+ \frac{r_2^2 N^2}{n_2 RS} - \frac{2 r_2 N}{S} + \frac{n_2 R}{S} \\
= \;& \frac{N^2}{RS}\left(\frac{r_0^2}{n_0} + \frac{r_1^2}{n_1} + \frac{r_2^2}{n_2}\right) - N\frac{R}{S} && \text{(A.1a)} \\
= \;& \frac{N^2}{RS}\left(\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right) - N\frac{S}{R}. && \text{(A.1b)}
\end{aligned}$$

We denote a contingency table and its column sums by $v = (r_0, r_1, r_2, s_0, s_1, s_2, n_0, n_1, n_2)$. Let $v' = v + u$, with v' and v differing by Hamming distance 1. Finding the sensitivity of Y boils down to finding v and u that maximize $\mid Y(v) - Y(v+u) \mid$.

Suppose $r_0 > 0$ and consider $u = (-1, 1, 0, 0, 0, 0, -1, 1, 0)$. As a consequence of (A.1b) we find that

$$\begin{aligned} Y(v) - Y(v+u) &= \left[\frac{N^2}{RS}\left(\frac{s_0^2}{n_0} + \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right) - N\frac{S}{R}\right] \\ &\quad - \left[\frac{N^2}{RS}\left(\frac{s_0^2}{n_0 - 1} + \frac{s_1^2}{n_1 + 1} + \frac{s_2^2}{n_2}\right) - N\frac{S}{R}\right] \\ &= \frac{N^2}{RS}\left[\frac{s_1^2}{n_1(n_1+1)} - \frac{s_0^2}{n_0(n_0-1)}\right]. \end{aligned}$$

Because $r_0 > 0$, we get that $n_0 = r_0 + s_0 \geqslant 1 + s_0$, and

$$0 \leqslant \frac{s_0^2}{n_0(n_0-1)} \leqslant \frac{s_0}{n_0} \leqslant \frac{s_0}{1+s_0} \leqslant \frac{s_{\max}}{1+s_{\max}}.$$

Similarly,

$$0 \leqslant \frac{s_1^2}{n_1(n_1+1)} \leqslant \frac{s_1}{n_1+1} \leqslant \frac{s_1}{1+s_1} \leqslant \frac{s_{\max}}{1+s_{\max}}.$$

Therefore,

$$\left|\frac{s_1^2}{n_1(n_1+1)} - \frac{s_0^2}{n_0(n_0-1)}\right| \leqslant \max\left\{\frac{s_1^2}{n_1(n_1+1)}, \frac{s_0^2}{n_0(n_0-1)}\right\} \leqslant \frac{s_{\max}}{1+s_{\max}}.$$

A similar analysis for all possible directions u and scenarios in which $r_1 > 0$ or $r_2 > 0$ reveals that the sensitivity of Y is bounded above by $\frac{N^2}{RS}\frac{s_{\max}}{1+s_{\max}}$. □

A.1.2.2 *Proof of Collorary 4.7*

For a change that occurs in the cases, we first treat s_0, s_1, and s_2 as fixed, and get the result in Theorem 4.6. By taking $(r_0, r_1, r_2, s_0, s_1, s_2) = (r_0, 1, r_2, 0, S, 0)$, $r_0 \geqslant r_2 > 0$ and changing the table in the direction of $u = (1, -1, 0, 0, 0, 0)$, we attain the upper bound $\frac{N^2}{RS}\left(1 - \frac{1}{S+1}\right)$. The same analysis for a change that occurs in the controls shows that the maximum change of the χ^2-statistic (i.e., Y in Appendix A.1.2.1) is $\frac{N^2}{RS}\left(1 - \frac{1}{R+1}\right)$. □

A.1.2.3 *Proof of Theorem 4.8*

From the definition of $W(d)$, we know that $W(d) \geqslant C$ for all d. For $t > C$,

$$\begin{aligned} \frac{\mathbb{P}(W(d) = t)}{\mathbb{P}(W(d') = t)} = \frac{\mathbb{P}(Z(d) = t)}{\mathbb{P}(Z(d') = t)} &\leqslant \exp\left(\Big|\,|t - h_C(d')| - |t - h_C(d)|\,\Big|\epsilon/s\right) \\ &\leqslant \exp\left(|h_C(d) - h_C(d')|\,\epsilon/s\right) \\ &\leqslant \exp(\epsilon). \end{aligned}$$

For $t = C$,

$$\frac{\mathbb{P}(W(d) = C)}{\mathbb{P}(W(d') = C)} = \frac{\mathbb{P}(Z(d) \leqslant C)}{\mathbb{P}(Z(d') \leqslant C)} = \frac{\frac{1}{2}\exp\left(\frac{C - h_C(d)}{s/\epsilon}\right)}{\frac{1}{2}\exp\left(\frac{C - h_C(d')}{s/\epsilon}\right)}$$
$$\leqslant \exp\left(\left|h_C(d') - h_C(d)\right| \epsilon/s\right)$$
$$\leqslant \exp(\epsilon).$$

□

A.1.2.4 *Proof of Theorem 4.9*

We denote a contingency table and its column sums by $v = (r_0, r_1, r_2, s_0, s_1, s_2, n_0, n_1, n_2)$. With the number of cases, R, and the number of controls, S, fixed, we can simply write $v^s = (s_1, s_2, n_1, n_2)$ or $v^r = (r_1, r_2, n_1, n_2)$. Then the allelic test statistic can be written as

$$Y_A(v^s) = \frac{2N^3}{RS}\frac{\{(s_1 + 2s_2) - \frac{S}{N}(n_1 + 2n_2)\}^2}{2N(n_1 + 2n_2) - (n_1 + 2n_2)^2},$$
$$\text{or} \quad Y_A(v^r) = \frac{2N^3}{RS}\frac{\{(r_1 + 2r_2) - \frac{R}{N}(n_1 + 2n_2)\}^2}{2N(n_1 + 2n_2) - (n_1 + 2n_2)^2}.$$

Let $v' = v + u$, with v' and v differing by Hamming distance 1. Finding the sensitivity of Y_A boils down to finding v and v' that maximize $| Y_A(v) - Y_A(v') |$. This is equivalent to maximizing $| Y_A(v^s) - Y_A(v^s + u^s) |$ and $| Y_A(v^r) - Y_A(v^r + u^r) |$, with u^s and u^r defined as follows:

when $r_0 > 0$,

$$u_1^s = (0, 0, 1, 0) \qquad \text{(Case 0} \to \text{Case 1)}$$
$$u_2^s = (0, 0, 0, 1) \qquad \text{(Case 0} \to \text{Case 2)}$$

when $s_0 > 0$,

$$u_1^r = (0, 0, 1, 0) \qquad \text{(Control 0} \to \text{Control 1)}$$
$$u_2^r = (0, 0, 0, 1) \qquad \text{(Control 0} \to \text{Control 2).}$$

In other words, when $r_0 > 0$, we search for tables that maximize $| \nabla Y_A(v^s) \cdot u_1^s |_{r_0>0}$ or $| \nabla Y_A(v^s) \cdot u_2^s |_{r_0>0}$; when $s_0 > 0$, we search for tables that maximize $| \nabla Y_A(v^r) \cdot u_1^r |_{s_0>0}$ and $| \nabla Y_A(v^r) \cdot u_2^r |_{s_0>0}$.

Let's first consider the case $r_0 > 0$. We have $\mid \nabla Y_A(\nu^s) \cdot u_1^s \mid = \left| \frac{\partial}{\partial n_1} Y_A(\nu^s) \right|$ and $\mid \nabla Y_A(\nu^s) \cdot u_2^s \mid = \left| \frac{\partial}{\partial n_2} Y_A(\nu^s) \right|$. Denote by

$$\begin{aligned} \alpha &= \frac{2N^3}{S(N-S)}, \\ C &= (s_1 + 2s_2) - \frac{S}{N}(n_1 + 2n_2), \\ D &= 2N(n_1 + 2n_2) - (n_1 + 2n_2)^2 = (n_1 + 2n_2)(2n_0 + n_1), \end{aligned}$$

then

$$\begin{aligned} Y_A &= \alpha \, \frac{C^2}{D}, \\ \frac{\partial}{\partial n_1} Y_A(\nu^s) &= \alpha \, \frac{-2\left[(N - n_1 - 2n_2)C^2 + \frac{S}{N}DC\right]}{D^2} = \alpha \, \frac{-2}{D^2}\left[(n_0 - n_2)C^2 + \frac{S}{N}DC\right], \\ \frac{\partial}{\partial n_2} Y_A(\nu^s) &= 2\, \frac{\partial}{\partial n_1} Y_A(\nu^s). \end{aligned}$$

Therefore, tables that maximize $\mid \nabla Y_A(\nu^s) \cdot u_1^s \mid_{r_0 > 0}$ also maximize $\mid \nabla Y_A(\nu^s) \cdot u_2^s \mid_{r_0>0}$. Furthermore, for the same table ν^s, the change of $Y_A(\nu^s)$ in the direction of u_2^s is no less than that in the direction of u_1^s.

Fixing n_1 and n_2, $\left| \frac{\partial}{\partial n_1} Y_A(\nu^s) \right|$ depends only on s_1 and s_2. So maximizing $\left| \frac{\partial}{\partial n_1} Y_A(\nu^s) \right|$ is equivalent to maximizing the absolute value of

$$\begin{aligned} f(s_1, s_2) &:= (n_0 - n_2)C^2 + \frac{S}{N}DC \\ &= \frac{S}{N}DC \, \mathbb{I}_{n_0 = n_2} + (n_0 - n_2)\left\{\left[C + \frac{SD}{2N(n_0 - n_2)}\right]^2 - \left[\frac{SD}{2N(n_0 - n_2)}\right]^2\right\} \mathbb{I}_{n_0 \neq n_2} \\ &= \frac{S}{N}DC \, \mathbb{I}_{n_0 = n_2} + (n_0 - n_2)\left\{[g(s_1, s_2)]^2 - \left[\frac{SD}{2N(n_0 - n_2)}\right]^2\right\} \mathbb{I}_{n_0 \neq n_2}, \end{aligned}$$

where $g(s_1, s_2) = C + \frac{SD}{2N(n_0 - n_2)} = (s_1 + 2s_2) + \frac{S(n_1 + 2n_2)^2}{2N(n_0 - n_2)}$. Note that the term D does not depend on s_1 or s_2. There are three scenarios:

(i) when $n_0 = n_2$, $|f(s_1, s_2)| = \frac{S}{N}D|(s_1 + 2s_2) - \frac{S}{N}(n_1 + 2n_2)|$ is maximized when $s_1 + 2s_2$ is minimized or maximized;

(ii) when $n_0 > n_2$, $|f(s_1, s_2)|$ is maximized when $|g(s_1, s_2)|$ is maximized or minimized, which occurs when $s_1 + 2s_2$ is minimized or maximized;

(iii) when $n_0 < n_2$, $|f(s_1, s_2)|$ is maximized when $|g(s_1, s_2)|$ is maximized or minimized as well. Because

$$\begin{aligned} g(s_1, s_2) &= (s_1 + 2s_2) - \frac{S(n_1 + 2n_2)^2}{2N(n_2 - n_0)} \\ &= (s_1 + 2s_2) - \frac{S(N + n_2 - n_0)^2}{2N(n_2 - n_0)} \\ &= (s_1 + 2s_2) - S\left[1 + \frac{N^2 + (n_2 - n_0)^2}{2N(n_2 - n_0)}\right] \\ &\leqslant (s_1 + 2s_2) - 2S, \qquad \text{because } N^2 + (n_2 - n_0)^2 \geqslant 2N(n_2 - n_0) \\ &\leqslant 0, \end{aligned}$$

$|g(s_1, s_2)|$ is maximized when $(s_1 + 2s_2)$ is minimized, and it is minimized when $(s_1 + 2s_2)$ is maximized.

The preceding analysis shows that for any given n_1 and n_2, $\left|\frac{\partial}{\partial n_1} Y_A(v^s)\right|$ is maximized when $(s_1 + 2s_2)$ is maximized or minimized; in other words, to maximize $\left|\frac{\partial}{\partial n_1} Y_A(v^s)\right|$, we only need to consider tables for which $(s_1, s_2) = (0, 0)$ or $(s_1, s_2) = (n_1, n_2)$.

Given $(s_1, s_2) = (0, 0)$, we have $C = -\frac{S}{N}(n_1 + 2n_2)$, and

$$\begin{aligned} \frac{\partial}{\partial n_1} Y_A(v^s) \Big/ (-2\alpha) &= (n_0 - n_2)\frac{C^2}{D^2} + \frac{SC}{ND} \\ &= (n_0 - n_2)\frac{\left[\frac{S}{N}(n_1 + 2n_2)\right]^2}{[(n_1 + 2n_2)(2n_0 + n_1)]^2} + \frac{S}{N}\frac{-\frac{S}{N}(n_1 + 2n_2)}{(n_1 + 2n_2)(2n_0 + n_1)} \\ &= -\frac{S^2}{N(2n_0 + n_1)^2}. \end{aligned}$$

So $\left|\frac{\partial}{\partial n_1} Y_A(v^s)\right|$ is maximized when $2n_0 + n_1$ is minimized. Because $(r_0 > 0, s_1 = s_2 = 0) \implies (r_0 \geqslant 1, r_1 = n_1, r_2 = n_2, s_0 = S) \implies (n_0 \geqslant S + 1, n_1 \geqslant 1)$, the minimum occurs at $v^s = (0, 0, 1, R - 2)$, i.e.,

$$\begin{cases} r_0 = 1, & r_1 = 1, \quad r_2 = R - 2, \\ s_0 = S, & s_1 = 0, \quad s_2 = 0, \\ n_0 = S + 1, & n_1 = 1, \quad n_2 = R - 2. \end{cases}$$

Given $(s_1, s_2) = (n_1, n_2)$, we have $C = \frac{R}{N}(n_1 + 2n_2)$, and

$$\begin{aligned}\frac{\partial}{\partial n_1} Y_A(\nu^s) \Big/ (-2\alpha) &= (n_0 - n_2)\frac{\left[\frac{R}{N}(n_1+2n_2)\right]^2}{[(n_1+2n_2)(2n_0+n_1)]^2} + \frac{S}{N}\frac{\frac{R}{N}(n_1+2n_2)}{(n_1+2n_2)(2n_0+n_1)} \\ &= \frac{R(S+n_0-n_2)}{N(2n_0+n_1)^2} \\ &= -\frac{1}{N}\left[\left(\frac{R}{2n_0+n_1} - \frac{1}{2}\right)^2 - \frac{1}{4}\right].\end{aligned}$$

Because $(s_1 = n_1, s_2 = n_2) \implies (r_1 = r_2 = 0) \implies (r_0 = R) \implies (n_0 \geqslant R)$, we have $0 < \frac{R}{2n_0+n_1} < 1/2 \implies 0 < \left(\frac{R}{2n_0+n_1} - \frac{1}{2}\right)^2 < 1/4$. So $\left|\frac{\partial}{\partial n_1} Y_A(\nu^s)\right|$ is maximized when $\left(\frac{R}{2n_0+n_1} - \frac{1}{2}\right)^2$ is minimized, which is achieved when $2n_0 + n_1$ is minimized, which occurs at $\nu^s = (1, S-1, 1, S-1)$, i.e.,

$$\begin{cases} r_0 = R, \quad r_1 = 0, \quad r_2 = 0, \\ s_0 = 0, \quad s_1 = 1, \quad s_2 = S-1, \\ n_0 = R, \quad n_1 = 1, \quad n_2 = S-1. \end{cases}$$

To summarize, when $r_0 > 0$, for any table ν^s, the change of $Y_A(\nu^s)$ in the direction of u_2^s is no less than that in the direction of u_1^s. The maximum change of Y_A in the direction of $u_2 = (-1, 0, 1, 0, 0, 0, -1, 0, 1) \equiv u_2^s$ occurs

$$\begin{aligned}\text{at } \nu_1^* &= (1, 1, R-2, S, 0, 0, S+1, 1, R-2), \\ \text{with } \Delta_1 &= |Y_A(\nu_1^*) - Y_A(\nu_1^* + u_2)| \\ &= \frac{2N^3}{RS}\left(\frac{S}{N}\right)^2 \left|\frac{2R-3}{2N-(2R-3)} - \frac{2R-1}{2N-(2R-1)}\right| \\ &= \frac{8N^2 S}{R(2S+3)(2S+1)}, \\ \text{or at } \nu_2^* &= (R, 0, 0, 0, 1, S-1, R, 1, S-1), \\ \text{with } \Delta_2 &= |Y_A(\nu_2^*) - Y_A(\nu_2^* + u_2)| \\ &= \frac{2N^3}{RS}\left\{\left(\frac{R}{N}\right)^2 \frac{2S-1}{2R+1} - \frac{\left[\frac{R}{N}(2S+1) - 2\right]^2}{(2S+1)(2R-1)}\right\} \\ &= \frac{8N^2[R^2(2S-1) - S]}{RS(2S+1)(2R+1)(2R-1)}.\end{aligned}$$

The same analysis for $s_0 > 0$ reveals that

$$|\nabla Y\_A(v^r) \cdot u\_4| = 2\,|\nabla Y\_A(v^r) \cdot u\_3| = 2\left|\frac{\partial}{\partial n_1} Y\_A(v^r)\right|,$$

and the maximum change of Y_A in the direction of $u_4 = (0, 0, 0, -1, 0, 1, -1, 0, 1) \equiv u_2^r$ occurs

$$\begin{aligned}
\text{at } v_3^* &= (R, 0, 0, 1, 1, S-2, R+1, 1, S-2),\\
\text{with } \Delta_3 &= |Y_A(v_3^*) - Y_A(v_3^* + u_4)| = \frac{8N^2R}{S(2R+3)(2R+1)},\\
\text{or at } v_4^* &= (0, 1, R-1, S, 0, 0, S, 1, R-1),\\
\text{with } \Delta_4 &= |Y_A(v_4^*) - Y_A(v_4^* + u_4)| = \frac{8N^2[S^2(2R-1) - R]}{RS(2R+1)(2S+1)(2S-1)}.
\end{aligned}$$

□

A.1.3 *Hamming distance score*

A.1.3.1 *Proof of Lemma 4.10*

$$\frac{\partial}{\partial x} Y_A = \frac{2N}{RS} \frac{1}{(x+n_{10})^2(2N-x-n_{10})^2}\, C(x),$$

where

$$\begin{aligned}
C(x) &= 2S(xS - n_{10}R)(x+n_{10})(2N-x-n_{10}) - (xS-n_{10}R)^2\,[(2N-x-n_{10}) - (x+n_{10})]\\
&= (xS - n_{10}R)(2N-x-n_{10})\Big[S(x+n_{10}) - (xS - n_{10}R)\Big]\\
&\quad + (xS - n_{10}R)(x+n_{10})\Big[S(2N-x-n_{10}) + (xS - n_{10}R)\Big]\\
&= (xS - n_{10}R)\Big[(2N-x-n_{10})n_{10}N + (x+n_{10})(2S-n_{10})N\Big].
\end{aligned}$$

Because $2N - x - n_{10} > 0$ and $2S - n_{10} \geqslant 0$, therefore $\frac{\partial}{\partial x} Y_A > 0$ when $x > n_{10}R/S$, and $\frac{\partial}{\partial x} Y_A < 0$ when $x < n_{10}R/S$. □

A.1.3.2 *Proof of Proposition 4.11*

We will only prove the first part of Proposition 4.11 and show that when a significant genotype table exists the first part indeed yields the shortest Hamming distance; the second part of Proposition 4.11 is designed to handle the extreme cases and does not need a proof.

Denote the number of changes made to the insignificant genotype table D until it becomes a significant table D' in each possible direction by $\nu_E, \nu_{SE}, \nu_S, \nu_W, \nu_{NW}, \nu_N$, where the subscripts indicate the direction of change described in Figure 4.2; that is,

$$\begin{aligned}
E &: (r_0 \to r_0 + 1, \quad r_1 \to r_1) \\
SE &: (r_0 \to r_0 + 1, \quad r_1 \to r_1 - 1) \\
S &: (r_0 \to r_0, \qquad\quad r_1 \to r_1 - 1) \\
W &: (r_0 \to r_0 - 1, \quad r_1 \to r_1) \\
NW &: (r_0 \to r_0 - 1, \quad r_1 \to r_1 + 1) \\
N &: (r_0 \to r_0, \qquad\quad r_1 \to r_1 + 1).
\end{aligned}$$

Let x_L and x_R denote the values of the χ^2 statistic represented by the left and the right black lines, respectively, in Figure 4.3. Let x_0 denote the value of x resulting from D. When we move the table D to the shaded area to the left of the black lines, we will immediately stop moving D when it becomes a table D' that resides on the line $2r_0 + r_1 = \lfloor x_L \rfloor$. Similarly for the shaded area to the right of the black lines, we immediately stop moving D when it becomes a table D'' that resides on the line $2r_0 + r_1 = \lceil x_R \rceil$. Observe that the number of dotted lines $2r_0 + r_1 = x$, which represent discreet values of x, between D and the line on which D' reside is $x - \lfloor x_L \rfloor - 1$, and that between D and the line on which D'' reside is $\lceil x_R \rceil - x - 1$. Also observe that moving D in the direction of E, SE, S, W, NW, or N results in a change of the value of $x = 2r_0 + r_1$ in the direction of 2, 1, -1, -2, -1, or 1, respectively.

Let's first find the shortest Hamming distance from the table D, represented by the point (r_0, r_1), to the shaded area to the left of the black lines. Finding the shortest Hamming distance is equivalent to solving the following optimization problem:

$$\begin{aligned}
\text{minimize:}\quad & \nu_E + \nu_{SE} + \nu_S + \nu_W + \nu_{NW} + \nu_N \\
\text{subject to:}\quad & -2\nu_E - \nu_{SE} + \nu_S + 2\nu_W + \nu_{NW} - \nu_N \geqslant (2r_0 + r_1) - \lfloor x_L \rfloor \\
& -2\nu_E - \nu_{SE} + \nu_S + 2(\nu_W - 1) + \nu_{NW} - \nu_N < (2r_0 + r_1) - \lfloor x_L \rfloor \\
& \nu_W + \nu_{NW} - \nu_E - \nu_{SE} \leqslant r_0 - r_0^{\min} \\
& \nu_S + \nu_{SE} - \nu_N - \nu_{NW} \leqslant r_1 - r_1^{\min} \\
& \nu_E, \nu_{SE}, \nu_S, \nu_W, \nu_{NW}, \nu_N \geqslant 0,
\end{aligned}$$

where $r_0^{\min}$ and $r_1^{\min}$ are the smallest possible values for r_0 and r_1, respectively. Because of the requirement that the margins of the genotype table to be positive, $r_0^{\min}$ and $r_1^{\min}$ can be greater than 0. The first constraint ensures that D crosses or ends up on the black line on the left. The second constraint ensures that D does not end up too far from the black line; i.e., moving D to the east by 1 step will prevent D from crossing the black line. The third and fourth constraint ensure that D stays inside the grid and does not move past the $r_0^{\min} = 0$ and $r_1^{\min} = 0$ lines, respectively. Let's rewrite the optimization problem as the following:

$$\begin{aligned}
\text{minimize:} \quad & v_E + v_{SE} + v_S + v_W + v_{NW} + v_N \\
\text{subject to:} \quad & 2v_E + v_{SE} - v_S - 2v_W - v_{NW} + v_N + (2r_0 + r_1) - \lfloor x_L \rfloor \leqslant 0 \\
& 2v_E + v_{SE} - v_S - 2(v_W - 1) - v_{NW} + v_N + (2r_0 + r_1) - \lfloor x_L \rfloor - 1 \leqslant 0 \\
& -v_E - v_{SE} + v_W + v_{NW} - r_0 + r_0^{\min} \leqslant 0 \\
& v_{SE} + v_S - v_{NW} - v_N - r_1 + r_1^{\min} \leqslant 0 \\
& -v_E \leqslant 0 \\
& -v_{SE} \leqslant 0 \\
& -v_S \leqslant 0 \\
& -v_W \leqslant 0 \\
& -v_{NW} \leqslant 0 \\
& -v_N \leqslant 0
\end{aligned}$$

Let's assign $u_i \geqslant 0, i \in \{1,2,3,4,E,SE,S,W,NW,N\}$ to each inequality constraint. Then the *KKT* conditions are

$$\begin{cases} -1 = 2u_1 + 2u_2 - u_3 - u_E \\ -1 = u_1 + u_2 - u_3 + u_4 - u_{SE} \\ -1 = -u_1 - u_2 + u_4 - u_S \\ -1 = -2u_1 - 2u_2 + u_3 - u_W \\ -1 = -u_1 - u_2 + u_3 - u_4 - u_{NW} \\ -1 = u_1 + u_2 - u_4 - u_N \\ \quad 0 \geqslant 2v_E + v_{SE} - v_S - 2v_W - v_{NW} + v_N + (2r_0 + r_1) - \lfloor x_L \rfloor \\ \quad 0 \geqslant 2v_E + v_{SE} - v_S - 2(v_W - 1) - v_{NW} + v_N + (2r_0 + r_1) - \lfloor x_L \rfloor - 1 \\ \quad 0 \geqslant -v_E - v_{SE} + v_W + v_{NW} - r_0 + r_0^{\min} \\ \quad 0 \geqslant v_{SE} + v_S - v_{NW} - v_N - r_1 + r_1^{\min} \\ \quad 0 = u_1\{2v_E + v_{SE} - v_S - 2v_W - v_{NW} + v_N + (2r_0 + r_1) - \lfloor x_L \rfloor\} \\ \quad 0 = u_2\{2v_E + v_{SE} - v_S - 2(v_W - 1) - v_{NW} + v_N + (2r_0 + r_1) - \lfloor x_L \rfloor - 1\} \\ \quad 0 = u_3\left(-v_E - v_{SE} + v_W + v_{NW} - r_0 + r_0^{\min}\right) \\ \quad 0 = u_4\left(v_{SE} + v_S - v_{NW} - v_N - r_1 + r_1^{\min}\right) \\ \quad 0 = u_E v_E = u_{SE} v_{SE} = u_S v_S = u_W v_W = u_{NW} v_{NW} = u_N v_N \\ \quad 0 \leqslant u_i, i \in \{1,2,4,E,SE,S,W,NW,N\} \end{cases}$$

Because the objective function is concave and the inequality constraints are convex, the *KKT* conditions are sufficient for optimality. The following points satisfy the *KKT* conditions, and hence they are solutions to the optimization problem:

(i) When $2\left(r_0 - r_0^{\min}\right) \geqslant (2r_0 + r_1) - \lfloor x_L \rfloor$ and $(2r_0 + r_1) - \lfloor x_L \rfloor$ is even:

$$(v_E, v_{SE}, v_S, v_W, v_{NW}, v_N) = (0, 0, 0, \frac{(2r_0 + r_1) - \lfloor x_L \rfloor}{2}, 0, 0)$$
$$(u_1, u_2, u_3, u_4) = (\frac{1}{2}, 0, 0, 0)$$
$$(u_E, u_{SE}, u_S, u_W, u_{NW}, u_N) = (2, \frac{3}{2}, \frac{1}{2}, 0, \frac{1}{2}, \frac{3}{2})$$

(ii) When $2\left(r_0 - r_0^{\min}\right) \geqslant (2r_0 + r_1) - \lfloor x_L \rfloor$ and $(2r_0 + r_1) - \lfloor x_L \rfloor$ is odd:

$$(\nu_E, \nu_{SE}, \nu_S, \nu_W, \nu_{NW}, \nu_N) = (0, 0, 0, \left\lceil \frac{(2r_0 + r_1) - \lfloor x_L \rfloor}{2} \right\rceil, 0, 0)$$
$$(u_1, u_2, u_3, u_4) = (0, \frac{1}{2}, 0, 0)$$
$$(u_E, u_{SE}, u_S, u_W, u_{NW}, u_N) = (2, \frac{3}{2}, \frac{1}{2}, 0, \frac{1}{2}, \frac{3}{2})$$

(iii) When $2\left(r_0 - r_0^{\min}\right) < (2r_0 + r_1) - \lfloor x_L \rfloor$:

$$\nu_W = r_0 - r_0^{\min}$$
$$\nu_S = (2r_0 + r_1) - \lfloor x_L \rfloor - 2\nu_W$$
$$(\nu_E, \nu_{SE}, \nu_{NW}, \nu_N) = (0, 0, 0, 0)$$
$$(u_1, u_2, u_3, u_4) = (1, 0, 1, 0)$$
$$(u_E, u_{SE}, u_S, u_W, u_{NW}, u_N) = (2, 1, 0, 0, 1, 2)$$

That is, the optimal solution can be found by either

(1) increasing ν_W until a solution is found, if $2\left(r_0 - r_0^{\min}\right) > (2r_0 + r_1) - \lfloor x_L \rfloor$, or

(2) increasing ν_W until $\nu_W = r_0$ then decreasing ν_S until a solution is found, if $2\left(r_0 - r_0^{\min}\right) < (2r_0 + r_1) - \lfloor x_L \rfloor$.

Similarly, we can find the shortest Hamming distance from the table D, represented by the point (r_0, r_1), to the shaded area to the right of the black linesby solving the following optimization problem:

$$\begin{aligned}
\text{minimize:}\quad & \nu_E + \nu_{SE} + \nu_S + \nu_W + \nu_{NW} + \nu_N \\
\text{subject to:}\quad & 2\nu_E + \nu_{SE} - \nu_S - 2\nu_W - \nu_{NW} + \nu_N \geqslant +\lceil x_R \rceil - (2r_0 + r_1) \\
& 2(\nu_E - 1) + \nu_{SE} - \nu_S - 2\nu_W - \nu_{NW} + \nu_N < \lceil x_R \rceil - (2r_0 + r_1) \\
& \frac{(r_1 + \nu_N + \nu_{NW} - \nu_S - \nu_{SE}) - r_1^{\max}}{(r_0 + \nu_E + \nu_{SE} - \nu_W - \nu_{NW}) - r_0^{\min}} \leqslant \frac{r_1^{\min} - r_1^{\max}}{r_0^{\max} - r_0^{\min}} \\
& \nu_E, \nu_{SE}, \nu_S, \nu_W, \nu_{NW}, \nu_N \geqslant 0,
\end{aligned}$$

Where $r_i^{\max}$ and $r_i^{\min}$ are, respectively, the maximum and minimum values r_i can take. The first constraint ensures that D crosses or ends up on the black line on the right. The second constraint ensures that D does not end up too far from the black line; i.e., moving D to the west by 1 step will prevent D from crossing the black line. The third constraint ensures that D stays inside the grid and does not move past the line $\frac{y-r_1^{\max}}{x-r_0^{\min}} = \frac{r_1^{\min}-r_1^{\max}}{r_0^{\max}-r_0^{\min}}$, which in Figure 4.3 is the top-right boundary formed by connecting the right most dots for each r_1. Once again, the *KKT* conditions are sufficient for optimality. Let's assign $u_i \geqslant 0, i \in \{1,2,3,E,SE,S,W,NW,N\}$ to each inequality constraint, then the following points satisfy the *KKT* conditions:

(i) When $\frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right) \geqslant \frac{(2r_0+r_1)-\lceil x_R\rceil}{2}$ and $(2r_0+r_1)-\lceil x_R\rceil$ is even:

$$(v_E,v_{SE},v_S,v_W,v_{NW},v_N) = (\frac{(2r_0+r_1)-\lceil x_R\rceil}{2},0,0,0,0,0)$$
$$(u_1,u_2,u_3) = (\frac{1}{2},0,0)$$
$$(u_E,u_{SE},u_S,u_W,u_{NW},u_N) = (0,\frac{1}{2},\frac{3}{2},2,\frac{3}{2},\frac{1}{2})$$

(ii) When $\frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right) \geqslant \frac{(2r_0+r_1)-\lceil x_R\rceil}{2}$ and $(2r_0+r_1)-\lceil x_R\rceil$ is odd:

$$(v_E,v_{SE},v_S,v_W,v_{NW},v_N) = (\left\lceil\frac{(2r_0+r_1)-\lceil x_R\rceil}{2}\right\rceil,0,0,0,0,0)$$
$$(u_1,u_2,u_3) = (0,\frac{1}{2},0)$$
$$(u_E,u_{SE},u_S,u_W,u_{NW},u_N) = (0,\frac{1}{2},\frac{3}{2},2,\frac{3}{2},\frac{1}{2})$$

(iii) When $\frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right)<\frac{(2r_0+r_1)-\lceil x_R\rceil}{2}$:

$$\begin{aligned}
\nu_E &= \frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right)\\
\nu_{SE} &= (2r_0+r_1)-\lceil x_R\rceil-2\nu_E\\
(\nu_S,\nu_W,\nu_{NW},\nu_N) &= (0,0,0,0)\\
u_1 &= \frac{1}{2}+\frac{\left(r_1^{\max}-r_1^{\min}\right)/2}{2\left(r_0^{\max}-r_0^{\min}\right)-\left(r_1^{\max}-r_1^{\min}\right)}\\
u_2 &= 0\\
u_3 &= \frac{1}{2\left(r_0^{\max}-r_0^{\min}\right)-\left(r_1^{\max}-r_1^{\min}\right)}\\
(u_E,u_{SE},u_S,u_W,u_{NW},u_N) &= (0,0,1,2,2,1)
\end{aligned}$$

That is, the optimal solution can be found by either

(1) increasing ν_E until a solution is found, if $\frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right)\geqslant\frac{(2r_0+r_1)-\lceil x_R\rceil}{2}$, or

(2) increasing ν_E until $\nu_E=\frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right)$, then decreasing ν_{SE} until a solution is found, if $\frac{r_0^{\max}-r_0^{\min}}{r_1^{\max}-r_1^{\min}}\left(r_1^{\max}-r_1\right)-\left(r_0-r_0^{\min}\right)<\frac{(2r_0+r_1)-\lceil x_R\rceil}{2}$.

□

A.2 Objective function perturbation

A.2.1 *Proof of Theorem 5.1*

Lemma A.1. *Let* G, g_1, *and* g_2 *be vector-valued continuous functions. Suppose that* G *is* λ-*strongly convex,* g_1 *is convex and* γ_1-*Lipschitz, and* g_2 *is convex and* γ_2-*Lipschitz. If* $f_1=\arg\min_f(G+g_1)(f)$ *and* $f_2=\arg\min_f(G+g_2)(f)$, *then*

$$\|f_1-f_2\|_2\leqslant(\gamma_1+\gamma_2)/\lambda.$$

Proof of Lemma A.1. $G + g_1$ and $G + g_2$ are λ-strongly convex because G is λ-strongly convex and g_1 and g_2 are convex. Then for $j, k, w \in \{1, 2\}$, $j \neq k$,

$$(G + g_w)(f_j) \geqslant (G + g_w)(f_k) + \partial(G + g_w)(f_k)^T(f_j - f_k) + \frac{\lambda}{2}\|f_j - f_k\|^2$$

where $\partial(G + g_w)$ denotes the subgradient. We know that $0 \in \partial(G + g_w)(f_w)$ because f_w minimizes $G + g_w$. Hence,

$$\begin{aligned}(G + g_2)(f_1) &\geqslant (G + g_2)(f_2) + \frac{\lambda}{2}\|f_1 - f_2\|_2^2,\\ (G + g_1)(f_2) &\geqslant (G + g_1)(f_1) + \frac{\lambda}{2}\|f_1 - f_2\|_2^2.\end{aligned}$$

By summing these two inequalities we obtain

$$(G + g_2)(f_1) + (G + g_1)(f_2) \geqslant (G + g_2)(f_2) + (G + g_1)(f_1) + \lambda\|f_1 - f_2\|_2^2$$

and hence

$$[g_2(f_1) - g_2(f_2)] + [g_1(f_2) - g_2(f_1)] \geqslant \lambda\,\|f_1 - f_2\|_2^2\,.$$

The fact that g_w is γ_w-Lipschitz implies that

$$\Big|g_2(f_1) - g_2(f_2)\Big| + \Big|g_1(f_2) - g_2(f_1)\Big| \leqslant (\gamma_1 + \gamma_2)\,\|f_1 - f_2\|_2$$

and hence

$$\begin{aligned}\lambda\|f_1 - f_2\|_2^2 &\leqslant [g_2(f_1) - g_2(f_2)] + [g_1(f_2) - g_2(f_1)]\\ &\leqslant \Big|g_2(f_1) - g_2(f_2)\Big| + \Big|g_1(f_2) - g_2(f_1)\Big| \leqslant (\gamma_1 + \gamma_2)\|f_1 - f_2\|_2.\end{aligned}$$

Therefore

$$\|f_1 - f_2\|_2 \leqslant (\gamma_1 + \gamma_2)/\lambda.$$

□

Proof of Theorem 5.1. For notational convenience we assume that $c_{\min} \geqslant c^*$ so that

$$L(\theta; T) = \frac{1}{n}\sum_{d \in T} l(\theta; d) + \lambda^T r(\theta) + \frac{\phi}{\epsilon n} b^T\theta.$$

If $c_{\min} < c^*$, we can extend r to include $r_{t+1}(\theta) = \frac{\max\{0, c^* - c_{\min}\}}{2}\|\theta\|_2^2$ and extend each $\lambda \in \Lambda$ such that $\lambda_{t+1} = 1$. Denote $\theta^*(T) = \arg\min_\theta L(\theta; T)$. First, we show that $|q(\theta^*(T), V) -$

$q(\theta^*(T'), V)| \leqslant \beta_1/n$ for training sets T and T' that differ only by one record. Here, Let $d \in T \backslash T'$ and $d' \in T' \backslash T$. Because T and T' differ by only one record, d and d' are sets with only one element. Let

$$G(\theta; T, T') = \frac{1}{n} \sum_{d \in T \cap T'} l(\theta; d) + \lambda^T r(\theta) + \frac{\phi}{\epsilon n} b^T \theta,$$
$$g_1(\theta; T, T') = \frac{1}{n} l(\theta; d) \qquad \text{and} \qquad g_2(\theta; T, T') = \frac{1}{n} l(\theta; d').$$

Then G is $c_{\min}$-strongly convex, and g_1 and g_2 are convex and γ/n-Lipschitz. By Lemma A.1, $\|\theta^*(T) - \theta^*(T')\|_2 \leqslant \frac{2\gamma}{n c_{\min}}$. Since h is ψ-Lipschitz we obtain for any validation set V,

$$|q(\theta^*(T), V) - q(\theta^*(T'), V)| \leqslant \frac{2\gamma\psi}{n c_{\min}}.$$

Secondly, we show that for all $\lambda \in \Lambda$ and for all validation sets V and V' that differ in a single record, $|q(\theta^*(T), V) - q(\theta^*(T'), V')| \leqslant \beta_2/m$. Since h is non-negative,

$$|q(\theta^*(T), V) - q(\theta^*(T'), V')| \leqslant h_{\max}/m,$$

where $h_{\max} = \sup_d h(\theta^*(T); d)$. By definition, $h_{\max} \leqslant h^*$. Moreover, because h is ψ-Lipschitz, $h_{\max} \leqslant \psi \|\theta^*(T)\|_2$. So $h_{\max} \leqslant \min\{h^*, \psi \|\theta^*(T)\|_2\}$. Now let E be the event that $\|b\|_2 \leqslant \xi$. Provided that E holds, we have

$$|b^T\theta_1 - b^T\theta_2| \leqslant \|b\|_2 \|\theta_1 - \theta_2\|_2 \leqslant \xi \|\theta_1 - \theta_2\|_2 .$$

Let

$$G(\theta) = \lambda^T r(\theta),$$
$$g_1(\theta; T) = \frac{1}{n} \sum_{d \in T} l(\theta; d) + \frac{\phi}{\epsilon n} b^T \theta,$$
$$g_2(\theta) = 0.$$

Then G is $c_{\min}$-strongly convex, g_1 is $\left(\gamma + \frac{\phi\xi}{\epsilon n}\right)$-Lipschitz, and g_2 is 0-Lipschitz. Since $G + g_2$ is minimized when $\theta = 0$, we obtain by invoking Lemma A.1 that

$$\|\theta^*(T)\|_2 = \|\theta^*(T) - 0\|_2 \leqslant \frac{1}{c_{\min}} \left(\gamma + \frac{\phi\xi}{\epsilon n}\right).$$

Therefore,

$$|q(\theta^*(T), V) - q(\theta^*(T), V')| \leqslant \frac{1}{m} \min\left\{h^*, \frac{\psi}{c_{\min}}\left(\gamma + \frac{\phi\xi}{\epsilon n}\right)\right\}.$$

□

A.2.2 *Proof of Theorem 5.2*

Lemma A.2. *If* A *is of full rank and* E *has rank at most 2, then*

$$\frac{\det(A+E) - \det(A)}{\det(A)} = \lambda_1(A^{-1}E) + \lambda_2(A^{-1}E) + \lambda_1(A^{-1}E)\lambda_2(A^{-1}E),$$

where $\lambda_j(Z)$ *denotes the* j*-th eigenvalue of matrix* Z*.*

Proof of Lemma A.2. See Lemma 10 in Chaudhuri et al. [4]. □

Proof of Theorem 5.2. Similar to the proof by Chaudhuri et al. [4], we show that if r is infinitely differentiable, then Algorithm 5 is ϵ-differentially private. It then follows from the successive approximation method by Kifer et al. [18] that Algorithm 5 is still ϵ-differentially private even if r is convex but not necessarily differentiable.

Let g denote the probability density function of the algorithm's output θ^*. Our goal is to show that

$$e^{-\epsilon} \leqslant \frac{g(\theta|D)}{g(\theta|D')} \leqslant e^{\epsilon}.$$

Suppose that the Hessian of r is continuous. Because $0 = \nabla L(\theta; D)$, we have

$$\begin{aligned} T_D(\theta) := b &= -\frac{\epsilon}{\phi}\left[\sum_{d\in D} \nabla l(\theta; d) + n\nabla r(\theta)\right] \\ \nabla T_D(\theta) &= -\frac{\epsilon}{\phi}\left[\sum_{d\in D} \nabla^2 l(\theta; d) + n\nabla^2 r(\theta)\right]. \end{aligned}$$

T_D is injective because $L(\theta; D)$ is strongly convex. Also, T_D is continuously differentiable. Therefore,

$$\frac{g(\theta|D)}{g(\theta|D')} = \frac{f(T_D(\theta))}{f(T_{D'}(\theta))} \frac{|\det(\nabla T_D)(\theta)|}{|\det(\nabla T_{D'})(\theta)|},$$

where f is the density function of b.

We first consider $\frac{|\det(\nabla T_D)(\theta)|}{|\det(\nabla T_{D'})(\theta)|}$. Let

$$A = -\frac{\phi}{\epsilon}\nabla T_{D'}, \qquad \text{and} \qquad E = \nabla^2 l(\theta; D\backslash D') - \nabla^2 l(\theta; D'\backslash D).$$

Because l is convex and r is strongly convex, $\nabla T_D(\theta)$ is positive definite. Hence, A has full rank. Also, E has rank at most 2 because $\nabla^2 l(\theta; d)$ is a rank 1 matrix by assumption. By Lemma A.2,

$$\begin{aligned}\frac{|\det(\nabla T_D(\theta))|}{|\det(\nabla T_{D'}(\theta))|} &= \left|\frac{\det(A+E)}{\det(A)}\right| \\ &\leqslant 1 + s_1(A^{-1}E) + s_2(A^{-1}E) + s_1(A^{-1}E)s_2(A^{-1}E),\end{aligned}$$

where $s_i(M)$ denotes the ith largest singular value of M. Because r is c^*-strongly convex, the smallest eigenvalue of A is at least nc^*. So $s_i(A^{-1}E) \leqslant \frac{s_i(E)}{nc^*}$. Because $\|\nabla l(\theta; d)\|_j \leqslant \kappa$ for $j \in \{1, 2\}$, applying the triangle inequality to the nuclear norm yields

$$s_1(E) + s_2(E) \leqslant \left\|\nabla^2 l(\theta; D\backslash D')\right\|_1 + \left\|\nabla^2 l(\theta; D'\backslash D)\right\|_1 \leqslant 2c.$$

Therefore, $s_1(A^{-1}E)\, s_2(A^{-1}E) \leqslant \left(\frac{c}{nc^*}\right)^2$, and

$$\frac{|\det(\nabla T_D)(\theta)|}{|\det(\nabla T_{D'})(\theta)|} = \frac{|\det(A+E)|}{|\det(A)|} \leqslant \left(1 + \frac{c}{nc^*}\right)^2.$$

Now we consider $\frac{f(T_D(\theta))}{f(T_{D'}(\theta))}$. Since

$$\begin{aligned}\|T_D(\theta) - T_{D'}(\theta)\|_j &= \left(\frac{\epsilon}{\phi}\right)\left\|\nabla l(\theta; D\backslash D') - \nabla l(\theta; D'\backslash D)\right\|_j \\ &\leqslant \left(\frac{\epsilon}{\phi}\right)\left(\left\|\nabla l(\theta; D\backslash D')\right\|_j + \left\|\nabla l(\theta; D'\backslash D)\right\|_j\right) \leqslant \frac{2\kappa\epsilon}{\phi},\end{aligned}$$

we obtain

$$\frac{f(T_D(\theta))}{f(T_{D'}(\theta))} = \frac{\exp\left(-\frac{\|T_D(\theta)\|_j}{2}\right)}{\exp\left(-\frac{\|T_{D'}(\theta)\|_j}{2}\right)} = \exp\left(\frac{\|T_{D'}(\theta)\|_j - \|T_D(\theta)\|_j}{2}\right) \leqslant \exp\left(\frac{\kappa\epsilon}{\phi}\right),$$

and therefore,

$$\frac{f(T_D(\theta))}{f(T_{D'}(\theta))}\frac{|\det(\nabla T_D)(\theta)|}{|\det(\nabla T_{D'})(\theta)|} \leqslant \exp\left(\frac{\kappa\epsilon}{\phi} + 2\log\left(1 + \frac{c}{nc^*}\right)\right) \leqslant e^{\epsilon}.$$

□

A.2.3 *Proof of Proposition 5.3*

The distribution of X is a special case of an s-dimensional *power exponential distribution* as defined by Gómez et al. [11], namely $X \sim \mathrm{PE}_s(\mu, \Sigma, \beta)$ with $\mu = (0, \dots, 0)^T$, $\Sigma = \mathrm{Id}_s$ and $\beta = \frac{1}{2}$. Gómez et al. [11] proved that if $T \sim \mathrm{PE}_s(\mu, \Sigma, \beta)$, then T has the same distribution as

$$\mu + Y A^T Z,$$

where Z is a random vector with uniform distribution on the unit sphere in $\mathbb{R}^s$, Y is an absolutely continuous non-negative random variable, independent from Z, whose density function is

$$g(y) = \frac{s}{\Gamma\left(1 + \frac{s}{2\beta}\right) 2^{\frac{s}{2\beta}}} y^{s-1} \exp\left(-\frac{1}{2} y^{2\beta}\right) I_{(0,\infty)}(y),$$

and $A \in \mathbb{R}^{s \times s}$ is a square matrix such that $A^T A = \Sigma$.

Note that for $\beta = \frac{1}{2}$, the distribution of Y boils down to a χ^2-distribution with $2s$ degrees of freedom. In addition, if $W \sim \mathcal{N}(0, \mathrm{Id}_s)$, then $W/|W|$ is uniformly distributed on the unit s-sphere. Finally, since $\Sigma = \mathrm{Id}_s$ we get that $A = \mathrm{Id}$. $\square$

B

CODE

B.1 Releasing the most significant SNPs

This section contains code for Yu et al. [37], which compares the differentially private methods for releasing the most significant SNPs. The code can also be accessed online at `https://github.com/fy/compare_dp_mechanisms`.

The example case and control genotype data were generated by HAP-SAMPLE[1]. For more details, see Malaspinas and Uhler [21].

Program B.1: Main program that calls other scripts.

```python
# Compare all differentially private mechanisms. The results are at the end of the page.

import sys, os, time
import subprocess, shlex
import matplotlib.pyplot as plt     # matplotlib - plots
from IPython.utils import io

class Dummy(dict):
    pass

CASE_FILE = '../example/case_genotypes.dat'
CONTROL_FILE = '../example/anticase_genotypes.dat'
SNP_TABLE_FILE = '../table.tmp'
JS_DISTANCE_FILE = '../js_distance.tmp'
CHISQUARE_FILE = '../chisquare.tmp'
```

1 http://www.hapsample.org/

```

# Convert raw files to genotype tables.

subprocess.check_call("python raw_to_geno_table.py {case_file} {control_file} {outfile}".format(case_file=CA

# Set common parameters.

PARAMS = Dummy()
PARAMS.NN_case = 1000  # number of cases
PARAMS.NN_control = 1000  # number of controls
PARAMS.MM_vec = np.array([1, 2, 3, 10])
PARAMS.epsilon_vec = np.arange(1, 1522, 300)
PARAMS.NN_perturb = 20
PARAMS.sig_level_vec = np.array([0.1, 0.05])

perturb_result_dict = {}  # store the perturbation results

# ## Johnson & Shmatikov method

# Count the number of SNPs.

snp_num = 0
with open(SNP_TABLE_FILE, 'r') as infile:
    for line in infile:
        snp_num += 1

# Do perturbation and collect results.

start_time = time.time()
sensitivity = 1

perturb_result = {}
for sig_level in PARAMS.sig_level_vec:
    print("sig_level={}".format(sig_level))
    pval = sig_level / snp_num
    subprocess.check_call(shlex.split("python write_JS_distance.py -p {pval} {infile} {outfile}".format(inf
    perturb_result[sig_level] = {}
    for MM in PARAMS.MM_vec:
        print("\tMM={}".format(MM))
        perturb_result[sig_level][MM] = {}
        for epsilon in PARAMS.epsilon_vec:
            print("\t\tepsilon={}".format(epsilon))
            perturbation = []
            for ii in xrange(PARAMS.NN_perturb):
                proc = subprocess.Popen(shlex.split('python get_JS_results.py {k} {e} {infile}'.format(k=MM
                perturbation.append(proc.communicate()[0].split())
            perturb_result[sig_level][MM][epsilon] = perturbation
```

```

perturb_result_dict['JS'] = perturb_result
print('Time spent: {} minutes.\n'.format(round((time.time() - start_time) / 60, 2)))

# ## Exponential and Laplace Mechanism

# Write file of χ²-statistics,

subprocess.check_call(shlex.split("python write_chisquare.py {infile} {outfile}".format(infile=SNP_TABLE_FILE, outfile=C

# Exponential mechanism

start_time = time.time()
## perturb
perturb_result_no_sigLevel = {}
for MM in PARAMS.MM_vec:
    print("\tMM={}".format(MM))
    perturb_result_no_sigLevel[MM] = {}
    for epsilon in PARAMS.epsilon_vec:
        print("\t\tepsilon={}".format(epsilon))
        perturbation = []
        for ii in xrange(PARAMS.NN_perturb):
            proc = subprocess.Popen(shlex.split('python get_expo_results.py {k} {e} {n_case} {n_control} {infile}'.forma
            perturbation.append(proc.communicate()[0].split())
        perturb_result_no_sigLevel[MM][epsilon] = perturbation

perturb_result = {}
for sig_level in PARAMS.sig_level_vec:
    perturb_result[sig_level] = perturb_result_no_sigLevel

perturb_result_dict['Exponential'] = perturb_result
print('Time spent: {} minutes.\n'.format(round((time.time() - start_time) / 60, 2)))

# Laplace Mechanism.

start_time = time.time()
## perturb
perturb_result_no_sigLevel = {}
for MM in PARAMS.MM_vec:
    print("\tMM={}".format(MM))
    perturb_result_no_sigLevel[MM] = {}
    for epsilon in PARAMS.epsilon_vec:
        print("\t\tepsilon={}".format(epsilon))
        perturbation = []
        for ii in xrange(PARAMS.NN_perturb):
            proc = subprocess.Popen(shlex.split('python get_laplace_results.py {k} {e} {n_case} {n_control} {infile}'.fo
            perturbation.append(proc.communicate()[0].split())
        perturb_result_no_sigLevel[MM][epsilon] = perturbation

perturb_result = {}
for sig_level in PARAMS.sig_level_vec:
    perturb_result[sig_level] = perturb_result_no_sigLevel

perturb_result_dict['Laplace'] = perturb_result
print('Time spent: {} minutes.\n'.format(round((time.time() - start_time) / 60, 2)))

# # Analysis

# ## Get the average number of SNPs recovered

# Get the χ^2-statistics.

name_score_tuples = []
with open(CHISQUARE_FILE, 'r') as infile:
    # skip header line
    garbage = infile.readline()
    for line in infile:
        name, score = line.split()
        name_score_tuples.append((name, float(score)))

name_score_dict = dict(name_score_tuples)
snp_scores = np.array([ss for name, ss in name_score_tuples])

perturb_result_utility = {}
for method in perturb_result_dict:
    perturb_result_utility[method] = {}
    for sig_level in perturb_result_dict[method]:
        perturb_result_utility[method][sig_level] = {}
        for MM in perturb_result_dict[method][sig_level]:
            perturb_result_utility[method][sig_level][MM] = {}
            for epsilon in perturb_result_dict[method][sig_level][MM]:
                M_highest_score = np.sort(snp_scores)[::-1][MM-1]
                perturbed_snp_scores = [np.array(map(name_score_dict.get, vec))
                                        for vec in perturb_result_dict[method][sig_level][MM][epsilon]]
                snps_recovered = map(lambda vec: np.sum(vec >= M_highest_score), perturbed_snp_scores)
                perturb_result_utility[method][sig_level][MM][epsilon] = 1. * np.mean(snps_recovered) / MM

# ## Make some plots

## make plots bigger
import matplotlib
matplotlib.rcParams['savefig.dpi'] = 1.5 * matplotlib.rcParams['savefig.dpi']

MY_COLORS = ["#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7"]
```

```
import matplotlib.lines as lines
MARKERS = lines.Line2D.filled_markers
LINE_STYLES = ['-', '--',  ':', '-.', '_' ]

# ### χ^2-statistics sorted in descending order

fig, ax = plt.subplots(figsize=(7,7))
ax.scatter(np.arange(len(snp_scores)), np.sort(snp_scores)[::-1], s=10, marker='x')
ax.set_ylabel('$\chi^2$-statistic')
ax.set_xlabel('Rank')

# ### Performance comparison of the DP mechanisms

fig, ax_array = plt.subplots(len(PARAMS.MM_vec), len(PARAMS.sig_level_vec),
                       sharex='col', sharey='row', figsize=(7,7))
plt.subplots_adjust(left=0.1, right=0.9, top=0.9, bottom=0.1)
for ii, MM in enumerate(PARAMS.MM_vec):
    for jj, sig_level in enumerate(PARAMS.sig_level_vec):
        ax = ax_array[ii, jj]
        ax.set_xlim([0, np.max(PARAMS.epsilon_vec) + 5])
        for kk, method in enumerate(perturb_result_utility):
            xx = PARAMS.epsilon_vec
            yy = map(perturb_result_utility[method][sig_level][MM].get, xx)
            ax.plot(xx, yy, color=MY_COLORS[kk % len(MY_COLORS)], label=method,
                    linestyle=LINE_STYLES[kk], linewidth=2.0, marker=MARKERS[kk],
                    markersize=4.5, )
        if ii == 0 and jj == 0:
            ax.legend(bbox_to_anchor=(0., 1.02, 1., .102), loc=3, ncol=3,
                     prop={'size':8}, mode="tight", borderaxespad=0.)
        if ii == 0:  # set column title on the first row
            ## set title
            ax.text(0.5, 1.25,
                r'$p$-value=$\frac{%s}{%s}$' % (str(sig_level), str(snp_num)),
                horizontalalignment='center',
                fontsize="large",
                transform=ax.transAxes)
        if jj == len(PARAMS.sig_level_vec) - 1:  # set row title on the last column
            ax.text(1.15, 0.5, 'M={}'.format(MM),
                    horizontalalignment='center',
                    verticalalignment='center',
                    rotation=0,
                    transform=ax.transAxes)
        if ii == len(PARAMS.MM_vec) - 1 and jj == 0:
            ax.set_ylabel('Utility')
            ax.set_xlabel('Privacy budget ($\epsilon$)')
```

Program B.2: `raw_to_geno_table.py`

```python
import sys, os, time
import argparse
from collections import Counter
import imp
import numpy as np

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

# Parse arguments.

parser = argparse.ArgumentParser()
parser.add_argument("case_file", help="raw case genotype data file")
parser.add_argument("control_file", help="raw control genotype data file")
parser.add_argument("outfile", help="output file")
args = parser.parse_args()

# Process the raw data.

utility_functions = import_anywhere('utility_functions', [SCRIPT_DIR])
from utility_functions import check_table_valid

## read case data
case_data = {}
with open(args.case_file, 'r') as infile:
    for line in infile:
        fields = line.split()
        snp_name = fields[1]
        case_data[snp_name] = Counter(fields[4:])

## read control data
control_data = {}
with open(args.control_file, 'r') as infile:
    for line in infile:
        fields = line.split()
        snp_name = fields[1]
        control_data[snp_name] = Counter(fields[4:])

## combine case and control data and write as table
with open(args.outfile, 'w') as outfile:
    headers = "name, case_0, case_1, case_2, ctrl_0, ctrl_1, ctrl_2".split(', ')
    line_template = '\t'.join(['{}'] * len(headers)) + '\n'
    outfile.write(line_template.format(*headers))
    for snp_name in case_data.keys():
        ## check whether the input table is a 2x3 contingency table with positive margins first
        input_table = np.array([[int(case_data[snp_name]['0']),
                                 int(case_data[snp_name]['1']),
                                 int(case_data[snp_name]['2'])],
                                [int(control_data[snp_name]['0'],),
                                 int(control_data[snp_name]['1'],),
                                 int(control_data[snp_name]['2'],)],])
        if not check_table_valid(input_table):
            continue
        ## the input table is valid. proceed.
        outfile.write(line_template.format(*[snp_name,
                                             case_data[snp_name]['0'],
                                             case_data[snp_name]['1'],
                                             case_data[snp_name]['2'],
                                             control_data[snp_name]['0'],
                                             control_data[snp_name]['1'],
                                             control_data[snp_name]['2'],]))
```

Program B.3: `write_JS_distance.py`

```
# Write the JS distance to a file. Each line will have the following fields:
#
# * name: SNP name
# * distance: JS distance

import sys, os, time
from collections import deque, Counter
import argparse
import numpy as np
import scipy as sp
import imp

parser = argparse.ArgumentParser(description="write JS distance")
parser.add_argument("infile", help="input genotype table file")
parser.add_argument("outfile", help="JS distance")
parser.add_argument("-p", help="threshold p-value", default=0.9, type=float)
args = parser.parse_args()
```

```python

if not os.path.isfile(args.infile):
    sys.exit("The follwoing file does not exist: {}".format(args.infile))

# Utility functions.

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

# Import some functions.

get_distance_to_significance = import_anywhere('get_distance_to_significance', [SCRIPT_DIR])
from get_distance_to_significance import greedy_distance_to_significance_flip
get_distance_to_significance.DEBUG = False

# Set p-value.

pval = args.p

# Write JS distance.

utility_functions = import_anywhere('utility_functions', [SCRIPT_DIR])
from utility_functions import check_table_valid

start_time = time.time()

with open(args.infile, 'r') as infile, open(args.outfile, 'w') as outfile:
    outfile.write("{}\t{}\n".format(*['name', 'distance']))
    headers = infile.readline().split()
    for line in infile:
        dd = dict(zip(headers, line.split()))
        input_table = np.array([[int(dd['case_0']),
                                 int(dd['case_1']),
                                 int(dd['case_2'])],
                                [int(dd['ctrl_0']),
                                 int(dd['ctrl_1']),
                                 int(dd['ctrl_2'])],])
        if not check_table_valid(input_table):
            continue
        print dd['name']
        dist = greedy_distance_to_significance_flip(input_table, pval)
        outfile.write('{}\t{}\n'.format(*[dd['name'], dist]))

print('Time spent: {} minutes.\n'.format(round((time.time() - start_time) / 60, 2)))
```

Program B.4: get_JS_results.py

```
# Get results using the JS method.
import sys, os, time
from collections import deque, Counter
import argparse
import imp
import numpy as np

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

# Utility functions.

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

# Parse command line arguments.

parser = argparse.ArgumentParser(description="Get results based on the JS method.")
parser.add_argument("k", metavar="NUM_SNP", help="number of SNPs to output", type=int)
parser.add_argument("e", metavar="EPSILON", help="privacy budget epsilon", type=float)
parser.add_argument("infile", help="input file of JS distances")
parser.add_argument("-s", help="sensitivity of the scoring function", default=1, type=int)
args = parser.parse_args()

if not os.path.isfile(args.infile):
    sys.exit("The follwoing file does not exist: {}".format(args.infile))

# Setup data.

js_dist_tuples = []
```

```
with open(args.infile, 'r') as infile:
    # skip header line
    garbage = infile.readline()
    for line in infile:
        name, distance = line.strip().split()
        js_dist_tuples.append((name, int(distance)))

indexed_snp_name_dict = dict(enumerate([name for name, dd in js_dist_tuples]))
snp_scores =  np.array([-dd if dd >= 0 else -dd - 1 for name, dd in js_dist_tuples])

# Perform JS algorithm.

loc_sig = import_anywhere('loc_sig', [SCRIPT_DIR])
from loc_sig import loc_sig

results_indices = loc_sig(args.e, args.k, args.s, snp_scores)
results_names = map(indexed_snp_name_dict.get, results_indices)

for nn in results_names:
    print nn
```

Program B.5: write_chisquare.py

```
# Write the χ²-statistics to a file. Each line will have the following fields:
#
# * name: SNP name
# * score: χ²-statistics

import sys, os, time
from collections import deque, Counter
import argparse
import numpy as np
import scipy as sp
import imp

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

parser = argparse.ArgumentParser(description="write chisquare distance")
parser.add_argument("infile", help="input genotype table file")
parser.add_argument("outfile", help="chisquare statistics")
args = parser.parse_args()

if not os.path.isfile(args.infile):
    sys.exit("The follwoing file does not exist: {}".format(args.infile))

```

```python
# Utility functions.

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

class Dummy(dict):
    pass

# Write χ²-statistics

utility_functions = import_anywhere('utility_functions', [SCRIPT_DIR])
from utility_functions import check_table_valid, chisq_stat

start_time = time.time()

with open(args.infile, 'r') as infile, open(args.outfile, 'w') as outfile:
    outfile.write("{}\t{}\n".format(*['name', 'chisquare']))
    headers = infile.readline().split()
    for line in infile:
        dd = dict(zip(headers, line.split()))
        input_table = np.array([[int(dd['case_0']),
                                 int(dd['case_1']),
                                 int(dd['case_2'])],
                                [int(dd['ctrl_0']),
                                 int(dd['ctrl_1']),
                                 int(dd['ctrl_2'])],])
        if not check_table_valid(input_table):
            continue
        outfile.write('{}\t{}\n'.format(*[dd['name'], chisq_stat(input_table)]))

print('Time spent: {} minutes.\n'.format(round((time.time() - start_time) / 60, 2)))
```

Program B.6: get_expo_results.py

```python
# Get results using the exponential mechanism.

import sys, os, time
from collections import deque
```

```python
import argparse
import numpy as np
import imp

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

# Utility functions.

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

utility_functions = import_anywhere('utility_functions', [SCRIPT_DIR])
from utility_functions import get_chisq_sensitivity

# Parse command line arguments.

parser = argparse.ArgumentParser(description="Get results based on the JS method.")
parser.add_argument("k", metavar="NUM_SNP", help="number of SNPs to output", type=int)
parser.add_argument("e", metavar="EPSILON", help="privacy budget epsilon", type=float)
parser.add_argument("n_case", help="number of cases", type=int)
parser.add_argument("n_control", help="number of controls", type=int)
parser.add_argument("infile", help="input file of chisquare statistics")
parser.add_argument("-s", help="sensitivity of the scoring function", default=1, type=int)
args = parser.parse_args()

if not os.path.isfile(args.infile):
    sys.exit("The follwoing file does not exist: {}".format(args.infile))

# Setup data.

name_score_tuples = []
with open(args.infile, 'r') as infile:
    # skip header line
    garbage = infile.readline()
    for line in infile:
        name, score = line.strip().split()
        name_score_tuples.append((name, float(score)))

indexed_snp_name_dict = dict(enumerate([name for name, ss in name_score_tuples]))
snp_scores =  np.array([ss for name, ss in name_score_tuples])
```

```

# Perform exponential algorithm.

loc_sig = import_anywhere('loc_sig', [SCRIPT_DIR])
from loc_sig import loc_sig

sensitivity = get_chisq_sensitivity(args.n_case, args.n_control)
results_indices = loc_sig(args.e, args.k, sensitivity, snp_scores)
results_names = map(indexed_snp_name_dict.get, results_indices)

for nn in results_names:
    print nn
```

Program B.7: `get_laplace_results.py`

```
# Get results using the exponential mechanism.

import sys, os, time
from collections import deque
import argparse
import numpy as np
import imp

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

# Utility functions.

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

utility_functions = import_anywhere('utility_functions', [SCRIPT_DIR])
from utility_functions import get_chisq_sensitivity

# Parse command line arguments.

parser = argparse.ArgumentParser(description="Get results based on the JS method.")
parser.add_argument("k", metavar="NUM_SNP", help="number of SNPs to output", type=int)
parser.add_argument("e", metavar="EPSILON", help="privacy budget epsilon", type=float)
parser.add_argument("n_case", help="number of cases", type=int)
parser.add_argument("n_control", help="number of controls", type=int)
parser.add_argument("infile", help="input file of chisquare statistics")
parser.add_argument("-s", help="sensitivity of the scoring function", default=1, type=int)
args = parser.parse_args()

if not os.path.isfile(args.infile):
    sys.exit("The follwoing file does not exist: {}".format(args.infile))

# Setup data.

name_score_tuples = []
with open(args.infile, 'r') as infile:
    # skip header line
    garbage = infile.readline()
    for line in infile:
        name, score = line.strip().split()
        name_score_tuples.append((name, float(score)))

indexed_snp_name_dict = dict(enumerate([name for name, ss in name_score_tuples]))
snp_scores =  np.array([ss for name, ss in name_score_tuples])

# Perform Laplace mechanism.

sensitivity = get_chisq_sensitivity(args.n_case, args.n_control)
scale = sensitivity * 2.0 * args.k / args.e
scores_perturbed = snp_scores + np.random.laplace(scale=scale,
                                                  size=len(snp_scores))
results_indices = np.argsort(scores_perturbed)[::-1][:args.k]
results_names = map(indexed_snp_name_dict.get, results_indices)

for nn in results_names:
    print nn
```

Program B.8: `get_distance_to_significance.py`

```
# This function will calculate the distance to flipping the significance of a SNP.
#
# Reference: Johnson & Shmatikov (2013)

# Assumptions:
#
# * The MAF of the controls are fixed.

import numpy as np
import scipy.stats as stats
```

```python
import os
import imp

SCRIPT_DIR = os.path.dirname(os.path.realpath(__file__))

def import_anywhere(module_name, paths):
    """import methods from any folder"""
    try:
        f, filename, desc = imp.find_module(module_name, paths)
        return imp.load_module(module_name, f, filename, desc)
    finally:
        # Since we may exit via an exception, close fp explicitly.
        if f:
            f.close()

# <headingcell level=3>

# Some constants and utility functions

DEBUG = True

utility_functions = import_anywhere('utility_functions', [SCRIPT_DIR])
from utility_functions import chisq_stat, chisq_gradient

## direction vectors for the case row and the genotype total of a 2x3 genotype table: (r_0, r_1, n_0, n_1)
DIRECTION_VECTORS = np.array([
    (0, 0, -1, 1), (0, 0, -1, 0), (0, 0, 1, -1), (0, 0, 1, 0),
    (0, 0, 0, -1), (0, 0, 0, 1),
    (-1, 1, -1, 1), (-1, 0, -1, 0), (1, -1, 1, -1), (1, 0, 1, 0),
    (0, -1, 0, -1), (0, 1, 0, 1),
])

def augment_direction_vector(uu):
    """Augment the direction vectors so that they are compatible with 2x3 input
    tables.
    Args:
        uu: direction vector for the cases
    """
    uu_for_r = np.array([uu[0], uu[1], -(uu[0] + uu[1])])
    ## check if uu will change the first row
    if np.any(uu_for_r != 0):
        uu_for_s = np.zeros(3)
    else:
        uu_for_s = np.array([uu[2], uu[3], -(uu[2] + uu[3])])
    return np.array([uu_for_r, uu_for_s])

def move_is_legal(input_table, uu):
    """Chcek whether moving the input table in the direction of uu is legal.
    Args:
        input_table: A 2x3 numpy matrix.
        uu: A direction vector compatible with the input table.
    Returns:
        True/False.
    """
    new_table = input_table + uu
    ## check negative cells
    if np.any(new_table < 0):
        return False
    ## check zero margins
    colsum = np.array(map(np.sum, new_table.T))
    if np.any(colsum == 0):
        return False
    return True

def greedy_distance_to_significance_flip(input_table, threshold_pval):
    """Calculate the distance to flip the significance.
    Distance is defined as the Hamming distance in the space of all databases.
    Args:
        input_table: A 2x3 numpy matrix.
        threshold_pval: Threshold p-value.
    Returns:
        The Hamming distance. Return "None" if no such distance can be found.
    At any point, say v, if the absolute value of the directional
    derivative is maximized in the direction u, then move v in the direction
    of u.
    """
    def find_best_legal_move(input_table, sig_direction):
        """Find the best legal move conditioning on the direction of the change
            in significance.
        """
        ## determine which moves are legal
        legal_moves = np.array([move_is_legal(input_table, augment_direction_vector(ee))
                                for ee in DIRECTION_VECTORS])
        if not np.any(legal_moves):
            ## return if not legal move is possible
            return None
        legal_move_indices = np.arange(len(DIRECTION_VECTORS))[legal_moves]
        ## get directional_derivatives
        gradient = chisq_gradient(input_table)
        directional_derivatives = np.array([
            np.dot(gradient, ee) for ee in np.array(DIRECTION_VECTORS)[legal_moves]])
        assert np.max(directional_derivatives * sig_direction) >= 0,\
        "The direction of the significance flip is %d " % (sig_direction) +\
        "but all of the directional derivatives are of the opposite sign." +\
            "The directional derivatives are: {}".format(str(directional_derivatives))
        if sig_direction > 0:
            best_move_idx = legal_move_indices[np.argmax(directional_derivatives)]
        else:
            best_move_idx = legal_move_indices[np.argmin(directional_derivatives)]
        if gradient == 0:
            print input_table
        return DIRECTION_VECTORS[best_move_idx]
    ## make a copy of the input table
    input_table = input_table.copy()
    dist = 0
    threshold_chisq = stats.chi2.isf(threshold_pval, 2)
    ## If sig_direction > 0, make table significant.
    ## If sig_direction < 0, make table insignificant.
    sig_direction = 1 if chisq_stat(input_table) < threshold_chisq else -1
    curr_table = input_table.copy()
    while 1:
        ## find best direction vector
        uu = find_best_legal_move(curr_table, sig_direction)
        if uu is None:
            break
        uu = augment_direction_vector(uu)
        curr_table += uu
        if DEBUG:
            print("chi-sqaure stat = {}, curr_table={}, uu={}".format(
                  chisq_stat(curr_table),
                  str(curr_table.tolist()),
                  str(uu.tolist())))
        dist += 1
        if sig_direction * (chisq_stat(curr_table) - threshold_chisq) >= 0:
            ## Significance has flipped!
            return dist * sig_direction
    return None

def main():
    print("Begin to debug.")
    ## check chisq_stat()
    tt1 = np.array([[16, 17, 18],
                    [10, 20, 5]])
    assert round(chisq_stat(tt1), 3) == 6.214, "Error with chisq_stat()."
    ## check chisq_gradient()
    grad1 = chisq_gradient(tt1)
    assert map(round, grad1, [3] * len(grad1)) == [-0.334, -0.646, 0.234, 0.401],\
        "Error with chisq_gradient()."
    tt2 = np.array([[0, 17, 18],
                    [10, 20, 5]])
    grad2 = chisq_gradient(tt2)
    assert map(round, grad2, [3] * len(grad2)) == [-1.565, -0.646, 0.612, 0.401],\
        "Error with chisq_gradient()."
    ## check move_is_legal()
    assert move_is_legal(tt1, augment_direction_vector((1, -1, 1, -1))),\
        "Error with move_is_legal()."
    assert not move_is_legal(tt2, augment_direction_vector((-1, 0, -1, 0))),\
        "Error with move_is_legal()."
    print("Finish debugging.")

if __name__ == "__main__":
    main()
```

Program B.9: `loc_sig.py`

```
# This function implements Johnson & Shmatikov (2013)'s LocSig function.

import numpy as np

def loc_sig(epsilon, kk, sensitivity, all_scores):
    """Return the index of the top kk SNPs.

    Algorithm:
    (1) Weight the sampling probability of each indexed element by e^score.
    (2) Sample an index using the sampling probabilities.
    (3) Store the index and set its sampling probability to be 0.
    (4) Repeat (1) to (3) until kk unique indices have been obtained.
    Args:
        epsilon: Privacy budget.
        kk: The top k SNPs to output.
        sensitivity: sensitivity of the scoring function.
        all_scores: The scores to significance (+) or insignificance (-).
    Returns:
        A list indices of the top kk SNPs.
    """
    def get_sampling_weights(exponent_vec):
        max_exponent = np.max(exponent_vec)
        sampling_weights = np.array([0 if ss is None else np.exp(ss - max_exponent + 50)
                                     for ss in exponent_vec])
        sampling_weights = sampling_weights / np.sum(sampling_weights)
        return sampling_weights

    ## get the exponents used to calculate the sampling weights
    exponent_vec = [None if ss is None else 1. * ss * (1. * epsilon / kk) / (2 * sensitivity)
                    for ss in all_scores]
    sampling_weights = get_sampling_weights(exponent_vec)
    loc_vec = []
    for ii in xrange(kk):
        loc = np.random.choice(np.arange(len(sampling_weights)),
                               p=sampling_weights)
        loc_vec.append(loc)
        exponent_vec[loc] = None
        sampling_weights = get_sampling_weights(exponent_vec)
    return loc_vec
```

Program B.10: `utility_functions.py`

```
# Utility functions.

import numpy as np

def get_chisq_sensitivity(NN_case, NN_control):
    """sensitivity for the chi-square statistic based on 2x3 genotype tables"""
    NN = NN_case + NN_control  # total number of subjects
    CC_max = max(NN_case, NN_control)
    CC_min = min(NN_case, NN_control)
    sensitivity = 1. * NN**2 / (CC_min * (CC_max + 1))  # sensitivity of chisq
    return sensitivity

def get_allelic_test_sensitivity(NN_case, NN_control):
    """sensitivity for the chi-square statistic based on 2x2 allelic tables derived from 2x3 genotype tables"""
    def sensitivity_type_1(SS, RR):
        NN = SS + RR
        return 1.0 * 8 * NN**2 * SS / \
                (RR * (2 * SS + 3) * (2 * SS + 1))

    def sensitivity_type_2(SS, RR):
        NN = SS + RR
        return 1.0 * 4 * NN**2 * ((2 * RR**2 - 1) * (2 * SS - 1) - 1) / \
                (SS * RR * (2 * RR + 1) * (2 * RR - 1) * (2 * SS + 1))

    return np.max([sensitivity_type_1(NN_case, NN_control),
                   sensitivity_type_1(NN_control, NN_case),
                   sensitivity_type_2(NN_case, NN_control),
                   sensitivity_type_2(NN_control, NN_case)])

def check_table_valid(input_table):
    """Make sure that the margins (row sums and column sums ) are all positive.
    Args:
        input_table: A 2x3 numpy matrix.
    """
    ## check zero margins
    rowsum = np.array(map(np.sum, input_table))
    colsum = np.array(map(np.sum, input_table.T))
    if np.any(rowsum == 0) or np.any(colsum == 0):
        return False
    else:
        return True

def chisq_stat(input_table):
    """Calculate the Pearson’s chi-square staitsitc.
    Args:
        input_table: A 2x3 numpy matrix.
    Returns:
        A tuple (chisquare_statistics, degree_of_freedom).
    """
    input_table = input_table.astype(float)
    rowsum = np.array(map(np.sum, input_table))
    colsum = np.array(map(np.sum, input_table.T))
    expected = np.outer(rowsum, colsum) / np.sum(rowsum)
    # df = (len([1 for rr in rowsum if rr > 0]) - 1) * \
    #      (len([1 for cc in colsum if cc > 0]) - 1)
    chisq = np.sum(np.array(input_table[expected > 0] -
                            expected[expected > 0]) ** 2 /
                   expected[expected > 0])
    # return (chisq, df)
    return chisq

def chisq_gradient(input_table):
    """Return the changable part of the gradient of the chi-square staitsitc.
    Args:
        input_table: A 2x3 numpy matrix.
    Returns:
        A four-element tuple consisting of the partial derivatives based on the
        parametrization the chi-square statistic by (r0, r1, n0, n1). The
        full parametrization would be
        (r0, r1, r2, s0, s1, s2, n0, n1, n2), where ri + si = ni. The returned
        value will be scaled down by N^2 / (R * S).
    """
    input_table = input_table.astype(float)
    colsum = np.array(map(np.sum, input_table.T))
    ## divide each cell by colsum
    fraction_table = input_table / colsum
    dy_dr0, dy_dr1 = [2 * fraction_table[0, ii] - 2 * fraction_table[0, 2] for
                      ii in [0, 1]]
    dy_dn0, dy_dn1 = [-fraction_table[0, ii] ** 2 + fraction_table[0, 2] ** 2 for
                      ii in [0, 1]]
    return (dy_dr0, dy_dr1, dy_dn0, dy_dn1)
```

B.2 Releasing coefficients of penalized logistic regressions

This section contains code for Yu et al. [39], which releases coefficients of elastic-net penalized logistic regression, with regularization parameter selection. The code can also be accessed online at `https://github.com/fy/dp_penalized_logistic_regression`.

The example case and control genotype data were generated by HAP-SAMPLE[2]. For more details, see Malaspinas and Uhler [21].

Program B.11: `analyze_simulation_for_paper.R`

```
## This function analyzes simulation results generated by
## ‘simulation_for_paper.R‘ and makes plots.

###############################################################################
## functions
###############################################################################
get_design_matrix = function(MM, training_or_testing="testing") {
  ## get the design matrix of the pair-wise interaction regression model
  top_M_snp = valid_snps[order(valid_snp_chisq, decreasing=TRUE)[1:MM]]
  if (training_or_testing == "testing") {
    top_snp_data = testing_data[, match(top_M_snp, names(testing_data))]
  } else {
    top_snp_data = training_data[, match(top_M_snp, names(training_data))]
  }
  pairwise_interact_formula = sprintf("~(%s)^2",paste(names(top_snp_data),
                                                        collapse='+'))
  design_matrix = model.matrix(formula(pairwise_interact_formula),
                               top_snp_data)
  design_matrix
}

validate.f = function(XX, yy, beta) {
  ## this is the validation function
  NN = nrow(XX)
  ss = 0
  for (ii in 1:nrow(XX)) {
```

2 http://www.hapsample.org/

```
    ss = ss + yy[ii] * sum(XX[ii, ] * beta) - log(1 + exp(sum(XX[ii, ] *
                                                                 beta)))
  }
  return(1 / NN * ss)
}

get_kappa = function(MM, norm=2) {
  ## get upper bound of the the p-norm of the gradient, which has the form xx^T
  if (norm==2) {
    kappa = sqrt(8 * MM^2 - 4 * MM)
  }
  if (norm==1) {
    kappa = 2 * MM^2
  }
  kappa
}

get_convex_min = function(MM, NN, epsilon, norm=2) {
  ## get the minimum strong convexity parameter
  kappa = get_kappa(MM, norm)
  kappa^2 / (NN *  (exp(epsilon / 4) - 1))
}

################################################################################
## constants and experiment parameters
################################################################################
MM_vec = c(5) ## top M SNPs
epsilon_vec = c(0.5, 1, 5, 10)
alpha_vec = c(0.1, 0.5, 0.9)
# ignore coefficients if |coef| / |largest_coef| < coef_thresh_ratio
coef_thresh_ratio = 1e-2

true_snps = c("rs4111409", "rs2324591")
main_effect_set = true_snps
interaction_set = c(paste(main_effect_set, collapse=':'),
                    paste(main_effect_set[2:1], collapse=':'))
all_effect_set = c(main_effect_set, interaction_set)

################################################################################
## Load data
################################################################################
source('./analyze_hapsample.R')
NN = nrow(testing_data)

if (FALSE) {
  ## the "simulation_for_paper.R" script should have been run.
  source("./simulation_for_paper.R")
```

```
}

library(glmnet)

################################################################################
## handle noisy results
################################################################################
load("./noisy_result.RData")

delta = 0.1
nn_sim = 100 ## number of simulations
nn_lambda = 5 ## number of regularization parameters

##########################################
## obtain validation scores
##########################################
yy = as.numeric(testing_data$status) - 1

validation_score_dtf = c()
for (ii in 1:length(noisy_result)) {
  for (MM in MM_vec) {
    design_matrix = get_design_matrix(MM)
    for (epsilon in epsilon_vec) {
      for (alpha in alpha_vec) {
        result_by_lambda =
          noisy_result[[ii]][[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
        for (res in result_by_lambda) {
          lambda = res$params$lambda
          score = validate.f(design_matrix, yy, res$optim_result$estimate)
          validation_score_dtf = rbind(validation_score_dtf,
                                       c(iter=ii, MM=MM, epsilon=epsilon,
                                         alpha=alpha, lambda=lambda,
                                         score=score))
        }
      }
    }
  }
}
validation_score_dtf = as.data.frame(validation_score_dtf)

##########################################
## obtain betas and other parameters
##########################################
param_dtf = c()
for (MM in MM_vec) {
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      kappa = get_kappa(MM)
      result_by_lambda =
        noisy_result[[1]][[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
      phi = result_by_lambda[[1]]$params$phi
      noise_scale = result_by_lambda[[1]]$params$noise_scale
      lambda_convex_min = kappa^2 / (NN *  (exp(epsilon / 4) - 1))
      lambda_vec = sapply(result_by_lambda, function(ee) ee$params$lambda)
      names(lambda_vec) = paste("lam", c(1:length(lambda_vec)), sep='')
      beta1 = 2 * kappa * kappa / lambda_convex_min
      beta2 = min(1, kappa / lambda_convex_min * (kappa + noise_scale * delta))
      use_delta = ifelse(beta2 < 1, TRUE, FALSE)
      param_dtf = rbind(param_dtf,
                        c(MM=MM, epsilon=epsilon, alpha=alpha,
                          kappa=kappa, phi=phi, noise_scale=noise_scale,
                          lambda_convex_min=lambda_convex_min,
                          beta1=beta1, beta2=beta2, use_delta=use_delta,
                          lambda_vec))
    }
  }
}
param_dtf = as.data.frame(param_dtf)

############################################
## sample a lambda differentially privately
############################################
best_lambda_dtf = c()
for (ii in 1:nn_sim) {
  for (MM in MM_vec) {
    for (epsilon in epsilon_vec) {
      for (alpha in alpha_vec) {
        ## get beta
        param_idx = (param_dtf$MM == MM) &
          (param_dtf$epsilon == epsilon) &
          (param_dtf$alpha == alpha)
        beta1 = param_dtf[param_idx, "beta1"][1]
        beta2 = param_dtf[param_idx, "beta2"][1]
        beta = max(beta1 / nrow(training_data), beta2 / nrow(testing_data))
        ## get validation scores
        val_score_idx = (validation_score_dtf$iter == ii) &
          (validation_score_dtf$MM == MM) &
          (validation_score_dtf$epsilon == epsilon) &
          (validation_score_dtf$alpha == alpha)
        score_vec = validation_score_dtf[val_score_idx, "score"]
        lambda_vec = validation_score_dtf[val_score_idx, "lambda"]
        ## perturbed scores
        perturbed_score_vec = score_vec + 2 * beta * rexp(length(score_vec),
                                                          rate=epsilon)
        best_lambda = lambda_vec[which.max(perturbed_score_vec)]
        best_lambda_dtf = rbind(best_lambda_dtf,
          c(iter=ii, MM=MM, epsilon=epsilon, alpha=alpha,
            best_lambda=best_lambda))
      }
    }
  }
}
best_lambda_dtf = as.data.frame(best_lambda_dtf)


###########################################
## get model corresponding to the best lambda
###########################################
best_model_list = list()
for (ii in 1:length(noisy_result)) {
  best_model_list[[ii]] = list()
  for (MM in MM_vec) {
    best_model_list[[ii]][[paste(MM)]] = list()
    for (epsilon in epsilon_vec) {
      best_model_list[[ii]][[paste(MM)]][[paste(epsilon)]] = list()
      for (alpha in alpha_vec) {
        best_lambda_idx = (best_lambda_dtf$iter == ii) &
          (best_lambda_dtf$MM == MM) &
          (best_lambda_dtf$epsilon == epsilon) &
          (best_lambda_dtf$alpha == alpha)
        best_lambda = best_lambda_dtf[best_lambda_idx, 'best_lambda']
        best_model_list[[ii]][[paste(MM)]][[paste(epsilon)]][[paste(alpha)]] =
          noisy_result[[ii]][[paste(MM)]][[paste(epsilon)]][[paste(alpha)]][[paste(best_lambda)]]
      }
    }
  }
}

###########################################
## calculate utility (sensitivity)
###########################################
freq_dtf = c()
for (ii in 1:length(noisy_result)) {
  for (MM in MM_vec) {
    design_matrix = get_design_matrix(MM)
    for (epsilon in epsilon_vec) {
      for (alpha in alpha_vec) {
        best_model =
          best_model_list[[ii]][[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
        coef_vec = best_model$optim_result$estimate
        top_coef_idx = (abs(coef_vec) / max(abs(coef_vec)) > coef_thresh_ratio)
        # ignore intercept
        top_coef_name = colnames(design_matrix)[-1][top_coef_idx[-1]]
        recovered_interaction = sum(top_coef_name %in% interaction_set)
        recovered_main_effcts = sum(top_coef_name %in% main_effect_set)
        recovered_all = (recovered_interaction + recovered_main_effcts == 3)
        freq_dtf = rbind(freq_dtf,
                         c(iter=ii, MM=MM, epsilon=epsilon, alpha=alpha,
                           recovered_interaction=recovered_interaction,
                           recovered_main_effcts=recovered_main_effcts,
                           recovered_all=recovered_all))
      }
    }
  }
}
freq_dtf = as.data.frame(freq_dtf)

util_dtf = c()
for (MM in MM_vec) {
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      idx = (freq_dtf$MM == MM) &
        (freq_dtf$epsilon == epsilon) &
        (freq_dtf$alpha == alpha)
      sub_dtf = freq_dtf[idx, ]
      util_dtf = rbind(util_dtf,
                       c(MM=MM, epsilon=epsilon, alpha=alpha,
                         recovered_interaction=
                           mean(sub_dtf$recovered_interaction),
                         recovered_main_effcts=
                           mean(sub_dtf$recovered_main_effcts) / 2,
                         recovered_all=mean(sub_dtf$recovered_all)))
    }
  }
}
util_dtf = as.data.frame(util_dtf)
util_dtf$alpha = as.factor(util_dtf$alpha)
util_dtf$epsilon = factor(util_dtf$epsilon)
util_dtf$epsilon = factor(util_dtf$epsilon,
                          levels=levels(util_dtf$epsilon)[c(3, 4, 1, 2)])

##########################################
## calculate specificity
##########################################
specificity_freq_dtf = c()
for (ii in 1:length(noisy_result)) {
  for (MM in MM_vec) {
    design_matrix = get_design_matrix(MM)
    for (epsilon in epsilon_vec) {
      for (alpha in alpha_vec) {
        bets_model =
          best_model_list[[ii]][[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
        coef_vec = bets_model$optim_result$estimate
        top_coef_idx = (abs(coef_vec) / max(abs(coef_vec)) > coef_thresh_ratio)
        top_coef_name =
          colnames(design_matrix)[-1][top_coef_idx[-1]] # ignore intercept
        no_wrong_snp = all(top_coef_name %in% all_effect_set)

        specificity_freq_dtf = rbind(specificity_freq_dtf,
                                 c(iter=ii, MM=MM, epsilon=epsilon, alpha=alpha,
                                   no_wrong_snp=no_wrong_snp))
      }
    }
  }
}
specificity_freq_dtf = as.data.frame(specificity_freq_dtf)

specificity_util_dtf = c()
for (MM in MM_vec) {
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      idx = (specificity_freq_dtf$MM == MM) &
        (specificity_freq_dtf$epsilon == epsilon) &
        (specificity_freq_dtf$alpha == alpha)
      sub_dtf = specificity_freq_dtf[idx, ]
      specificity_util_dtf = rbind(specificity_util_dtf,
                                 c(MM=MM, epsilon=epsilon, alpha=alpha,
                                   specificity=mean(sub_dtf$no_wrong_snp)))
    }
  }
}
specificity_util_dtf = as.data.frame(specificity_util_dtf)
specificity_util_dtf$alpha = as.factor(specificity_util_dtf$alpha)
specificity_util_dtf$epsilon = factor(specificity_util_dtf$epsilon)
specificity_util_dtf$epsilon =
  factor(specificity_util_dtf$epsilon,
         levels=levels(specificity_util_dtf$epsilon)[c(3, 4, 1, 2)])

##########################################
## make some plots for experiment with noisy data
##########################################
library(ggplot2)
library(lattice)

pdf("RU_barchart.pdf", width=6, height=4)
```

```r

epsilon_labels = sapply(levels(util_dtf$epsilon), function(ee) {
  convex_min = get_convex_min(util_dtf$MM[1], NN, as.numeric(ee))
  as.expression(substitute(paste(epsilon, "=", ee, ',  convex_min=',
                                 convex_min,  sep=''),
                           list(ee=ee, convex_min=round(convex_min, 2))))
})

barchart(recovered_interaction + recovered_main_effcts +
           recovered_all ~ alpha | epsilon,
         data=util_dtf,
         scales = list(x = list("free", cex=1.2), y=list(cex=1.2)),
         par.settings=list(superpose.polygon=list(col=rainbow(3))),
         layout = c(round(length(epsilon_vec)/2),2),
         strip=strip.custom(factor.levels=epsilon_labels),
         horizontal = FALSE,
         xlab=list(expression(alpha), cex=2),
         ylab=list("Frequency", cex=2),
         auto.key = list(columns=3, cex=2, size = 1, between = 0.5,
                         points=FALSE, rectangles=TRUE,
                         text=c("interaction", "main effects", "all")),
         panel = function( x,y,...) {
           panel.abline( h=0, lty = "dotted", col = "black", lwd=1.5)
           panel.abline( h=1, lty = "dotted", col = "black", lwd=1.5)
           panel.barchart( x,y,...)})


epsilon_labels = sapply(levels(specificity_util_dtf$epsilon), function(ee) {
  convex_min = get_convex_min(specificity_util_dtf$MM[1], NN, as.numeric(ee))
  as.expression(substitute(paste('convex_min=', convex_min,  sep=''),
                           list(convex_min=round(convex_min, 2))))
})

barchart(specificity ~ alpha | epsilon,
         data=specificity_util_dtf,
         scales = list(x = list("free", cex=1.2), y=list(cex=1.2)),
         layout = c(round(length(epsilon_vec)/2),2),
         strip=strip.custom(factor.levels=epsilon_labels),
         horizontal = FALSE,
         xlab=list(expression(alpha), cex=2),
         ylab=list("Frequency", cex=2),
         auto.key = list(columns=1, cex=2, size = 1, between = 0.5,
                         points=FALSE, rectangles=TRUE,
                         text=c("specificity")),
         panel = function( x,y,...) {
           panel.abline( h=0, lty = "dotted", col = "black", lwd=1.5)
           panel.abline( h=1, lty = "dotted", col = "black", lwd=1.5)
```

```
          panel.barchart( x,y,...)})
dev.off()

###############################################################################
## get result with no noise
###############################################################################
load('./no_noise_result.RData')

## get validation scores
no_noise_validation_score_dtf = c()
for (MM in MM_vec) {
  design_matrix = get_design_matrix(MM, "training")
  yy = as.numeric(testing_data$status) - 1
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      result_by_lambda = n
      o_noise_result[[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
      for (res in result_by_lambda) {
        lambda = res$params$lambda
        score = validate.f(design_matrix, yy, res$optim_result$estimate)
        no_noise_validation_score_dtf =
          rbind(no_noise_validation_score_dtf,
                c(MM=MM, epsilon=epsilon, alpha=alpha,
                  lambda=lambda, score=score))
      }
    }
  }
}
no_noise_validation_score_dtf = as.data.frame(no_noise_validation_score_dtf,
                                              stringsAsFactors=FALSE)
no_noise_validation_score_dtf$score =
  as.numeric(no_noise_validation_score_dtf$score)

## get best no noise model

no_noise_best_model_list = list()
for (ii in 1:length(no_noise_result)) {
  no_noise_best_model_list[[ii]] = list()
  for (MM in MM_vec) {
    no_noise_best_model_list[[ii]][[paste(MM)]] = list()
    for (epsilon in epsilon_vec) {
      no_noise_best_model_list[[ii]][[paste(MM)]][[paste(epsilon)]] = list()
      for (alpha in alpha_vec) {
        val_score_idx = (no_noise_validation_score_dtf$MM == MM) &
          (no_noise_validation_score_dtf$epsilon == epsilon) &
          (no_noise_validation_score_dtf$alpha == alpha)
        lambda_vec = no_noise_validation_score_dtf[val_score_idx, "lambda"]
```

```r
        score_vec = no_noise_validation_score_dtf[val_score_idx, "score"]
        best_lambda = lambda_vec[which.max(score_vec)]
        no_noise_best_model_list[[paste(MM)]][[paste(epsilon)]][[paste(alpha)]] =
          no_noise_result[[paste(MM)]][[paste(epsilon)]][[paste(alpha)]][[paste(best_lambda)]]
      }
    }
  }
}

###########################################
## get utility (sensitivity): no noise
###########################################
no_noise_freq_dtf = c()
ii = -1
for (MM in MM_vec) {
  design_matrix = get_design_matrix(MM)
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      bets_model = no_noise_best_model_list[[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
      coef_vec = bets_model$optim_result$estimate
      top_coef_idx = (abs(coef_vec) / max(abs(coef_vec)) > coef_thresh_ratio)
      top_coef_name = colnames(design_matrix)[-1][top_coef_idx[-1]] # ignore intercept
      recovered_interaction = sum(top_coef_name %in% interaction_set)
      recovered_main_effcts = sum(top_coef_name %in% main_effect_set)
      recovered_all = (recovered_interaction + recovered_main_effcts == 3)
      no_noise_freq_dtf = rbind(no_noise_freq_dtf,
                         c(iter=ii, MM=MM, epsilon=epsilon, alpha=alpha,
                           recovered_interaction=recovered_interaction,
                           recovered_main_effcts=recovered_main_effcts,
                           recovered_all=recovered_all))
    }
  }
}
no_noise_freq_dtf = as.data.frame(no_noise_freq_dtf, stringsAsFactors=FALSE)
no_noise_freq_dtf$recovered_interaction =
  as.numeric(no_noise_freq_dtf$recovered_interaction)
no_noise_freq_dtf$recovered_main_effcts =
  as.numeric(no_noise_freq_dtf$recovered_main_effcts)
no_noise_freq_dtf$recovered_all = as.logical(no_noise_freq_dtf$recovered_all)

no_noise_util_dtf = c()
for (MM in MM_vec) {
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      idx = (no_noise_freq_dtf$MM == MM) &
        (no_noise_freq_dtf$epsilon == epsilon) &
        (no_noise_freq_dtf$alpha == alpha)
      sub_dtf = no_noise_freq_dtf[idx, ]
      no_noise_util_dtf =
        rbind(no_noise_util_dtf,
              c(MM=MM, epsilon=epsilon, alpha=alpha,
                recovered_interaction=mean(sub_dtf$recovered_interaction),
                recovered_main_effcts=mean(sub_dtf$recovered_main_effcts) / 2,
                recovered_all=mean(sub_dtf$recovered_all)))
    }
  }
}
no_noise_util_dtf = as.data.frame(no_noise_util_dtf, stringsAsFactors=FALSE)
no_noise_util_dtf$recovered_interaction =
  as.numeric(no_noise_util_dtf$recovered_interaction)
no_noise_util_dtf$recovered_main_effcts =
  as.numeric(no_noise_util_dtf$recovered_main_effcts)
no_noise_util_dtf$recovered_all = as.numeric(no_noise_util_dtf$recovered_all)
no_noise_util_dtf$alpha = as.factor(no_noise_util_dtf$alpha)
no_noise_util_dtf$epsilon = factor(no_noise_util_dtf$epsilon)
no_noise_util_dtf$epsilon =
  factor(no_noise_util_dtf$epsilon,
         levels=levels(no_noise_util_dtf$epsilon)[c(3,4,1,2)])

##########################################
## calculate specificity: no noise
##########################################
ii = - 1
no_noise_specificity_freq_dtf = c()
for (MM in MM_vec) {
  design_matrix = get_design_matrix(MM)
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      bets_model =
        no_noise_best_model_list[[paste(MM)]][[paste(epsilon)]][[paste(alpha)]]
      coef_vec = bets_model$optim_result$estimate
      top_coef_idx = (abs(coef_vec) / max(abs(coef_vec)) > coef_thresh_ratio)
      top_coef_name =
        colnames(design_matrix)[-1][top_coef_idx[-1]] # ignore intercept
      no_wrong_snp = all(top_coef_name %in% all_effect_set)

      no_noise_specificity_freq_dtf =
        rbind(no_noise_specificity_freq_dtf,
              c(iter=ii, MM=MM, epsilon=epsilon, alpha=alpha,
                no_wrong_snp=no_wrong_snp))
    }
  }
}
no_noise_specificity_freq_dtf =
```

```r
  as.data.frame(no_noise_specificity_freq_dtf, stringAsFactor=FALSE)
no_noise_specificity_freq_dtf$no_wrong_snp =
  as.logical(no_noise_specificity_freq_dtf$no_wrong_snp)

no_noise_specificity_util_dtf = c()
for (MM in MM_vec) {
  for (epsilon in epsilon_vec) {
    for (alpha in alpha_vec) {
      idx = (no_noise_specificity_freq_dtf$MM == MM) &
        (no_noise_specificity_freq_dtf$epsilon == epsilon) &
        (no_noise_specificity_freq_dtf$alpha == alpha)
      sub_dtf = no_noise_specificity_freq_dtf[idx, ]
      no_noise_specificity_util_dtf = rbind(no_noise_specificity_util_dtf,
                                    c(MM=MM, epsilon=epsilon, alpha=alpha,
                                      specificity=mean(sub_dtf$no_wrong_snp)))
    }
  }
}
no_noise_specificity_util_dtf =
  as.data.frame(no_noise_specificity_util_dtf, stringsAsFactors=FALSE)
no_noise_specificity_util_dtf$specificity =
  as.numeric(no_noise_specificity_util_dtf$specificity)
no_noise_specificity_util_dtf$alpha =
  as.factor(no_noise_specificity_util_dtf$alpha)
no_noise_specificity_util_dtf$epsilon =
  factor(no_noise_specificity_util_dtf$epsilon)
no_noise_specificity_util_dtf$epsilon =
  factor(no_noise_specificity_util_dtf$epsilon,
         levels=levels(no_noise_specificity_util_dtf$epsilon)[c(3,4,1,2)])

##########################################
## make some plots including no noise
##########################################
library(ggplot2)
library(lattice)

pdf("RU_barchart_with_no_noise.pdf", width=6, height=4)

epsilon_labels = sapply(levels(no_noise_util_dtf$epsilon), function(ee) {
  convex_min = get_convex_min(no_noise_util_dtf$MM[1], NN, as.numeric(ee))
  as.expression(substitute(paste('convex_min=', convex_min,  sep=''),
                          list(convex_min=round(convex_min, 2))))
})

barchart(recovered_interaction + recovered_main_effcts +
           recovered_all ~ alpha | epsilon,
         data=no_noise_util_dtf,
         scales = list(x = list("free", cex=1.5), y=list(cex=1.2)),
         par.settings=list(superpose.polygon=list(col=rainbow(3))),
         layout = c(ceiling(length(epsilon_vec)/2), 2),
         strip=strip.custom(factor.levels=epsilon_labels),
         horizontal = FALSE,
         xlab=list(expression(alpha), cex=2),
         ylab=list("Utility", cex=2),
         auto.key = list(columns=3, cex=2, size = 1, between = 0.5,
                         points=FALSE, rectangles=TRUE,
                         text=c("interaction", "main effects", "all")),
         panel = function( x,y,...) {
           panel.abline( h=0, lty = "dotted", col = "black", lwd=1.5)
           panel.abline( h=1, lty = "dotted", col = "black", lwd=1.5)
           panel.barchart( x,y,...)})

epsilon_labels = sapply(levels(no_noise_specificity_util_dtf$epsilon),
                        function(ee) {
  convex_min = get_convex_min(no_noise_specificity_util_dtf$MM[1], NN,
                              as.numeric(ee))
  as.expression(substitute(paste('convex_min=', convex_min,  sep=''),
                           list(convex_min=round(convex_min, 2))))
})
barchart(specificity ~ alpha | epsilon,
         data=no_noise_specificity_util_dtf,
         scales = list(x = list("free", cex=1.5), y=list(cex=1.2)),
         layout = c(ceiling(length(epsilon_vec)/2), 2),
         strip=strip.custom(factor.levels=epsilon_labels),
         horizontal = FALSE,
         xlab=list(expression(alpha), cex=2),
         ylab=list("Utility", cex=2),
         auto.key = list(columns=1, cex=2, size = 1, between = 0.5,
                         points=FALSE, rectangles=TRUE,
                         text=c("specificity")),
         panel = function( x,y,...) {
           panel.abline( h=0, lty = "dotted", col = "black", lwd=1.5)
           panel.abline( h=1, lty = "dotted", col = "black", lwd=1.5)
           panel.barchart( x,y,...)})
dev.off()
```

Program B.12: `simulation_for_paper.R`

```
## This function performs differentially private elastic-net penalized logistic
## regression multiple times and saves the results to an RData file. It also
## performs non-private penalized logistic regression and saves the results to
## an RData file.
```

```

################################################################################
## load data
################################################################################
source('./analyze_hapsample.R')

################################################################################
## functions
################################################################################
obj.f = function(beta, XX, yy, lambda, alpha, noise, noise_scale) {
  ## object function of penalized logistic regression
  ## (with intercept)
  NN = nrow(XX)
  ss = 0
  for (ii in 1:nrow(XX)) {
    ss = ss + yy[ii] * sum(XX[ii, ] * beta) - log(1 + exp(sum(XX[ii, ] *
                                                                beta)))
  }
  ## object function of elastic-net penalized logistic regression
  -1 / NN * ss + lambda / 2 * (1 - alpha) * sum(beta[-1]^2) + lambda * alpha *
    sum(sapply(beta[-1], abs)) + noise_scale * sum(noise[-1] * beta[-1])
}

get_design_matrix = function(MM) {
  ## get the design matrix of the pair-wise interaction regression model
  top_M_snp = valid_snps[order(valid_snp_chisq, decreasing=TRUE)[1:MM]]
  top_snp_data = training_data[, match(top_M_snp, names(training_data))]
  pairwise_interact_formula = sprintf("~(%s)^2",paste(names(top_snp_data),
                                                      collapse='+'))
  design_matrix = model.matrix(formula(pairwise_interact_formula),
                               top_snp_data)
  design_matrix = design_matrix[, -1] ## remove intercept
  design_matrix
}

get_kappa = function(MM, norm=2) {
  ## Get upper bound of the the p-norm of the gradient,
  ## and the gradient is of the form xx^T.
  if (norm==2) {
    kappa = sqrt(8 * MM^2 - 4 * MM)
  }
  if (norm==1) {
    kappa = 2 * MM^2
  }
  kappa
}

```

```r
rB2 = function(deg) {
  ## perturbation noise function (l2-norm)
  WW = rnorm(deg)
  YY = rchisq(1, 2*deg)
  YY * WW / sqrt(sum(WW^2))
}

################################################################################
## simulation parameters
################################################################################
NN = nrow(training_data)

nn_sim = 100 ## number of simulations
nn_lambda = 5 ## number of regularization parameters

MM_vec = c(5) ## top M SNPs
epsilon_vec = c(0.5, 1, 5, 10)
alpha_vec = c(0.1, 0.5, 0.9)

param_list = lapply(MM_vec, function(MM) {
  kappa = get_kappa(MM)
  phi = 2 * kappa
  res = lapply(epsilon_vec, function(epsilon) {
    lambda_convex_min = kappa^2 / (NN *  (exp(epsilon / 4) - 1))
    return(list(kappa=kappa,
                phi=phi,
                lambda_convex_min=lambda_convex_min,
                epsilon=epsilon,
                MM=MM))
  })
  names(res) = sapply(epsilon_vec, paste)
  return(res)
})
names(param_list) = sapply(MM_vec, paste)

param_dtf = c()
for (MM_res in param_list) {
  for (epsilon_res in MM_res) {
    param_dtf = rbind(param_dtf, c(kappa=epsilon_res$kappa,
                        phi=epsilon_res$phi,
                        lambda_convex_min=epsilon_res$lambda_convex_min,
                        epsilon=epsilon_res$epsilon,
                        MM=epsilon_res$MM))
  }
}
param_dtf= as.data.frame(param_dtf)
```

```r
################################################################################
## Run simulation on a separate dataset to obtain the **optimal regularization
## constants**. At the moment, use the validation dataset. In the future,
## should use a new one.
################################################################################
library(glmnet)

glmnet_results = lapply(MM_vec, function(MM) {
  ## only use top MM SNPs
  design_matrix = get_design_matrix(MM=MM)
  res = lapply(alpha_vec, function(alpha) {
    glmnet(design_matrix, training_data$status, family="binomial",
                    standardize=FALSE, alpha=alpha)
  })
  names(res) = sapply(alpha_vec, paste)
  return(res)
})
names(glmnet_results) = sapply(MM_vec, paste)

################################################################################
## Simulations.
################################################################################

###########################################
## helper functions
###########################################
sim_result = function(MM, epsilon, alpha, lambda, phi, noisy=TRUE) {
  ## aggregate the simulation results
  design_matrix = get_design_matrix(MM)
  nn_var = 1 + ncol(design_matrix)  # add intercept
  if (noisy) {
    noise = rB2(nn_var)
    noise_scale = phi / (epsilon * nrow(design_matrix))
  } else  {
    noise_scale = 0
    noise = rep(0, nn_var)
  }
  XX = cbind(1, design_matrix)
  yy = as.numeric(training_data$status) - 1

  optim_result = run_cvx(XX, yy, lambda, alpha, noise, noise_scale)

  list(optim_result=optim_result,
       params=list(alpha=alpha,
                   lambda=lambda,
                   MM=MM,
                   epsilon=epsilon,
                   phi=phi,
                   noise_scale=noise_scale))
}

get_lambda_vec = function(all_lambda, lambda_convex_min, alpha, nn_lambda) {
  ## first test glmnet's default lambda list
  rescaled_lambda = all_lambda * (1 - alpha) / 2
  valid_lambda_vec = all_lambda[rescaled_lambda > lambda_convex_min]
  while (TRUE) {
    nn_valid_lambda = length(valid_lambda_vec)
    if (nn_valid_lambda > nn_lambda) {
      mult = floor((nn_valid_lambda - 1) / nn_lambda)
      lambda_vec = sort(valid_lambda_vec)[1 + c(0:(nn_lambda - 1)) * mult]
      break
    }
    if (nn_valid_lambda > 0) {
      lambda_vec = c(valid_lambda_vec,
                     max(valid_lambda_vec) * c(2: (1 + nn_lambda -
                                                     nn_valid_lambda)))
      break
    }
    lambda_vec = lambda_convex_min * c(1:nn_lambda)
    break
  }
  lambda_vec
}

sim_wrapper = function(noisy=TRUE) {
  ## wraper function for the simulations
  res_MM_list = lapply(MM_vec, function(MM) {
    res_epsilon_list = lapply(epsilon_vec, function(epsilon) {
      lambda_convex_min =
        param_dtf[((param_dtf$MM==MM) & (param_dtf$epsilon==epsilon)),
                  "lambda_convex_min"]
      phi = param_dtf[param_dtf$MM==MM, "phi"][1]
      res_alpha_list = lapply(alpha_vec, function(alpha) {
        ## first test glmnet's default lambda list
        glmnet_fit = glmnet_results[[paste(MM)]][[paste(alpha)]]
        lambda_vec = get_lambda_vec(glmnet_fit$lambda, lambda_convex_min, alpha,
                                    nn_lambda)
        res = lapply(lambda_vec, function(lambda) {
          print(sprintf("Processed MM=%s, epsilon=%s, alpha=%s, lambda=%s", MM,
                        epsilon, alpha, lambda))
          sim_result(MM=MM, epsilon=epsilon, alpha=alpha, lambda=lambda,
                     phi=phi, noisy=noisy)
        })
        names(res) = sapply(lambda_vec, paste)
        return(res)
      })
      names(res_alpha_list) = sapply(alpha_vec, paste)
      return(res_alpha_list)
    })
    names(res_epsilon_list) = sapply(epsilon_vec, paste)
    return(res_epsilon_list)
  })
  names(res_MM_list) = sapply(MM_vec, paste)
  return(res_MM_list)
}

###########################################
## get the simulation results when noise is added
## run optimization using cvx in matlab
###########################################
ptm = proc.time()
print(paste("Number of nlm fits = ",
            length(MM_vec) * length(epsilon_vec) * length(alpha_vec) *
              nn_lambda * nn_sim))

if (TRUE) {
  require(R.matlab)

  Matlab$startServer()
  ## Create a MATLAB client object used to communicate with MATLAB
  matlab <- Matlab()
  ## Connect to the MATLAB server.
  isOpen <- open(matlab)
  ## Confirm that the MATLAB server is open, and running
  if (!isOpen) {
    throw("MATLAB server is not running: waited 30 seconds.")
  }

  source('./run_cvx.R')
  noisy_result = lapply(1:nn_sim, function(ee) {
    print(paste("Simulation iteration", ee))
    return(sim_wrapper(noisy=TRUE))
  })

  ## When done, close the MATLAB client, which will also shutdown
  ## the MATLAB server and the connection to it.
  close(matlab)
}
print(floor((proc.time() - ptm) / 60))

## save the results
```

```r
save(noisy_result, file="./noisy_result.RData")

###########################################
## get the simulation results when NO noise is added
###########################################
## when we require that the smallest candidate lambda to depend on epsilon

## use glmnet to find the estimates
sim_wrapper.glmnet = function() {
  res_MM_list = lapply(MM_vec, function(MM) {
    design_matrix = get_design_matrix(MM=MM)
    res_epsilon_list = lapply(epsilon_vec, function(epsilon) {
      lambda_convex_min =
        param_dtf[((param_dtf$MM==MM) & (param_dtf$epsilon==epsilon)),
                  "lambda_convex_min"]
      phi = param_dtf[param_dtf$MM==MM, "phi"][1]
      res_alpha_list = lapply(alpha_vec, function(alpha) {
        ## first test glmnet's default lambda list
        glmnet_fit = glmnet_results[[paste(MM)]][[paste(alpha)]]
        lambda_vec =
          get_lambda_vec(glmnet_fit$lambda, lambda_convex_min, alpha, nn_lambda)
        res = lapply(lambda_vec, function(lambda) {
          print(sprintf("Processed MM=%s, epsilon=%s, alpha=%s, lambda=%s", MM,
                        epsilon, alpha, lambda))
          fit = glmnet(design_matrix, training_data$status, family="binomial",
                       standardize=FALSE, alpha=alpha, lambda=lambda)
          list(optim_result=list(estimate=c(as.vector(fit$a0),
                                            as.vector(fit$beta))),
               params=list(alpha=alpha,
                           lambda=lambda,
                           MM=MM,
                           epsilon=epsilon,
                           phi='NA',
                           noise_scale=0))
        })
        names(res) = sapply(lambda_vec, paste)
        return(res)
      })
      names(res_alpha_list) = sapply(alpha_vec, paste)
      return(res_alpha_list)
    })
    names(res_epsilon_list) = sapply(epsilon_vec, paste)
    return(res_epsilon_list)
  })
  names(res_MM_list) = sapply(MM_vec, paste)
  return(res_MM_list)
}
```

```
##
ptm = proc.time()

print(paste("Number of nlm fits = ",
            length(MM_vec) * length(epsilon_vec) * length(alpha_vec) *
              nn_lambda))
no_noise_result = sim_wrapper.glmnet()
print(floor((proc.time() - ptm) / 60))
## save the results
save(no_noise_result, file="./no_noise_result.RData")
```

Program B.13: `analyze_hapsample.R`

```
## This is a helper function. It loads HapSample data into the workspace.

################################################################################
## read in data
################################################################################
case_path = "./data/case_genotypes_Nov09_interaction1_MAF025_1"
control_path = "./data/anticase_genotypes_Nov09_interaction1_MAF025_1"

case_data = read.table(case_path, header=FALSE, sep='\t', as.is=TRUE)
case_data = data.frame(case_data)
rownames(case_data) = case_data[, 2]
control_data = read.table(control_path, header=FALSE, sep='\t', as.is=TRUE)
control_data = data.frame(control_data)

## transpose the data then combine cases and controls into a single dataframe
case_nn = ncol(case_data)-4
snp_nn = nrow(case_data)

control_data_t = data.frame(t(control_data[,  -(1:4)]))
names(control_data_t) = control_data[, 2]
case_data_t = data.frame(t(case_data[,  -(1:4)]))
names(case_data_t) = case_data[, 2]

all_data = rbind(control_data_t,
                 case_data_t[, match(names(case_data_t),
                                     names(control_data_t))])
all_data = cbind(status=as.factor(rep(c("control", "case"),
                                      c(case_nn, case_nn))), all_data)

################################################################################
## calculate the chi-square statistics and p-values for each snp
################################################################################
```

```
chisq_test_results = list()
for (ii in 1:(ncol(all_data)-1)) {
  if (length(unique(all_data[, 1+ii])) == 1) {
    chisq_test_results[[ii]] = NULL
  } else {
    chisq_test_results[[ii]] = chisq.test(all_data[, 1+ii],
                                          all_data[, 1],
                                          correct=FALSE)
  }
}
chisq_stat = sapply(chisq_test_results, function(ee) {
  if (is.null(ee)) return(NA)
  if (ee$parameter[['df']] != 2) return(NA)
  return(ee$statistic)
})
chisq_stat = as.vector(chisq_stat)
chisq_pval = sapply(chisq_test_results, function(ee) {
  if (is.null(ee)) return(NA)
  if (ee$parameter[['df']] != 2) return(NA)
  return(ee[['p.value']])
})
chisq_pval = as.vector(chisq_pval)

################################################################################
## sample training and testing data
################################################################################
if (TRUE) {
  set.seed(101)
  training_indiv = sample(1:case_nn, case_nn / 2)
}
testing_indiv = c(1:case_nn)[-training_indiv]

training_data = all_data[c(training_indiv, case_nn + training_indiv), ]
testing_data = all_data[c(testing_indiv, case_nn + testing_indiv), ]

################################################################################
## get valid snps and valid chisquare statistics
################################################################################
valid_snps = names(all_data)[-1][!is.na(chisq_stat)]
valid_snp_chisq = chisq_stat[!is.na(chisq_stat)]
```

Program B.14: run_cvx.R

```
## This is a helper function. It is a wrapper for running CVX for MATLAB in R.

run_cvx <- function(XX, yy, lambda, alpha, noise, noise_scale){
  ## Save data to Matlab data file
  writeMat('cvx_data.mat', XX=XX, yy=yy, lambda=lambda, alpha_var=alpha,
           noise=noise, noise_scale=noise_scale);
  warnings()

  # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  # Use MATLAB server
  # - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  evaluate(matlab, "opt_elastic_net_for_R()")

  ## Load results from Matlab
  fit_beta = unlist(read.csv("cvx_results.csv", header=FALSE))

  return(list(estimate=fit_beta))
}
```

Program B.15: opt_elastic_net_for_R.m

```
function [] = opt_elastic_net_for_R()

% perfromes the optimization step
% output is vectors of parameters and minimized optimial value

% XX is a matrix of patients data
% yy is the response vector (+1 -1 for non-diseased and diseased patients)
% alpha_var controls the sparseness of the results
% lambda is the penalty calculated during the algorithm
% noise is a vector of noise values
% noise_scale is the multiplicative scale of the noise

load('cvx_data.mat');

MM = length(XX(1, :)) ; % number of features
NN = length(XX(:, 1)) ; % number of patients


% Solve optimization problem
cvx_begin
    variable bbeta(MM);
    minimize ((-yy(:)' * (XX * bbeta(:)) + sum_log(1 + exp( (XX * ...
        bbeta(:))’))) / NN  + lambda / 2 * (1 - alpha_var) * ...
        sum_square_abs(bbeta(2:MM, 1)) + lambda * alpha_var * ...
        sum(abs(bbeta(2:end, 1))) + noise_scale * dot(noise(2:MM), ...
        bbeta(2:MM, 1)));
cvx_end

csvwrite(’cvx_results.csv’, bbeta)

end
```