Feature Selection Bias in Assessing the Predictivity of SNPs for Alzheimer's Disease

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By

Mei Dong

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Mathematics and Statistics
142 Mcclean Hall, 106 Wiggins Road
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5E6
Canada

OR

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

Abstract

In the context of identifying related SNPs for a phenotype of interest (e.g., a disease status), we consider the problem of assessing the predictivity of SNPs that are selected by performing genome-wide association studies. Internal cross-validation (ICV) is a wrong but often used method for this assessment. With ICV, a subset of SNPs are pre-selected based on all samples then cross-validation (CV) is applied to assess the predictivity of the pre-selected SNPs. The predictivity estimate of the selected SNPs given by ICV is upwardly biased. This is often called the feature selection bias problem. The cause of this bias is that the feature selection procedure, which is a part of training procedure, is not external to the test samples in ICV. A correct method, called external cross-validation (ECV), is to re-select features based on only the training samples in each fold of CV such that the feature selection is external to test samples. The feature selection bias of ICV has been discussed by a few articles in the context of cancer diagnosis with microarray data. However, this problem has not received sufficient attention in the literature, especially in the context of predicting with SNP data. Many articles in the literature use ICV or do not state explicitly that their feature selection is external to test samples. In this thesis, we use an example of predicting late-onset Alzheimer's disease (LOAD) from selected SNPs to demonstrate that ICV could lead to severe false discovery. We use a real SNP dataset related to LOAD and two synthetic datasets (simulated response with real SNPs) for this demonstration. For the prediction, we compare the performances of three regularized logistic regression methods: LASSO, elastic-net, and a fully Bayesian hyper-LASSO method. For the LOAD dataset, we see that, except for APOE, no other SNPs can improve the prediction of LOAD using ECV method; however, the predictivity estimate of selected SNPs given by ICV can reach an R^2 as high as 80%. For the synthetic datasets, we obtain the similar results as in the real dataset; additionally we see that the predictivity estimate of selected SNPs obtained with ICV can be even higher than the oracle predictivity of the truly related SNPs used to generate the response. In this study, we also find that the hyper-LASSO method can achieve better predictive performance than the LASSO and elastic-net. We recommend that ICV should not be used to measure the predictivity of selected SNPs and this statement should be made clear in research articles.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Longhai Li, for his invaluable guidance and continuous support throughout the whole master training period. It is my great honour to work with him. His immense knowledge, constructive comments, and warm encouragements helped me a lot during difficult times. This thesis could not have been written without his invaluable advice and patient guidance.

I would like to thank my co-supervisor, Dr. Lloyd Balbuena, for his assistance in the topic of GWAS and financial support. I benefit a lot from his meticulous comments on my thesis. I appreciate my committee member, Dr. Cindy Feng, not only for the knowledge that I have learnt from her course but also for her support and insightful suggestion in my thesis. I would like to thank the Department of Mathematics and Statistics for the funding provided through my Master study.

I am indebted to all my friends and colleagues for their love and support for me for my life. Special thanks go to my roommate Huiyao Kuang and my friend Nathan Yang, who shared the knowledge of Linux with me patiently.

Last but not least, I am thankful to my family, my parents Jiayang Dong and Liyin Zhang, my brother Lu Dong, and my boyfriend Yiran Wang, who have loved me and supported me all the time.

Contents

List of Tables

List of Figures

List of Abbreviations

AD	Alzheimer's Disease
AMLP	Average Minus Log Probability
APOE $\varepsilon 4$	$\varepsilon 4$ alleles of apolipoprotein E
AUC	Area Under the Curve
CV	Cross-Validation
ECV	External Cross-Validation
FDR	False Discovery Rate
Fdr	Tail-Based False Discovery Rate
GWAS	Genome-Wide Association Study
LOOCV	Leave-One-Out Cross-Validation
LOAD	Late-Onset Alzheimer's Disease
LRT	Likelihood Ratio Test
ICV	Internal Cross-Validation
pFDR	positive FDR
ROC	Receiver operating characteristic
SNPs	Single Nucleotide Polymorphisms

1. Introduction

Current genotyping technologies have increased the capacity of genome-wide association studies (GWAS) for identifying genetic variants (single-nucleotide polymorphisms, or SNPs) that affect phenotypes or traits of interest. The number of SNPs, p, is thousands of times larger than the number of individuals n. In this thesis, we sometimes use a generic term "feature" to call SNP. Typically, GWAS calculates the p-value for a single SNP based on a statistical test and then identifies SNPs with the smallest p-values [1, 2]. To deal with the multiple comparison problem, some researchers have recommended converting p-values into false discovery rates [3, 4]. However, features that are highly statistically significant are not necessarily good predictors of a disease [5, 6]. Some recent studies have shown that the power of a predictive model is not increased when adding more significant variables from classical approaches based on significance test [7–9]. Thus, a different approach is needed. Using appropriate statistical learning methods to conduct predictive analysis in GWAS has important implications. Predictive analysis can help diagnose human diseases, facilitate the discovery of biological mechanisms for a phenotype [10], and assess the predictivity by metrics such as error rate and area under the Receiver operating characteristic (ROC) curve (AUC).

Two methods are often used by researchers to assess the predictivity of selected SNPs when applying cross-validation (CV): internal cross-validation (ICV) and external cross-validation (ECV). In ICV, a subset of SNPs is pre-selected based on all samples, and then CV is applied to assess the predictivity of the pre-selected SNPs. The test data in each fold of ICV is not external to the feature selection procedure. This practice inflates the predictive accuracy because the information from test samples has been used in feature selection. The bias caused by using ICV is also called feature selection bias. ECV, on the other hand, requires us to re-select SNPs based only on the training data in each fold of CV. Hence, all model building steps (i.e., feature selection, choosing tuning parameters, and model training) are implemented in each fold of training data such that no information from the test data is used in "training" stage. ECV means that we will use external test samples to assess

the predictivity of the selected subset. There are two possible reasons why ICV is used: it reduces computational costs [11], and it gives a stable subset of selected features [12].

The feature selection bias was first observed by Ambroise et al. [13] in the context of microarray classification. They used leave-one-out cross-validation (LOOCV) for colon cancer dataset and leukemia dataset. They found that the biases are above 15% and around 5% respectively for each dataset, which is not negligible. Two approaches, ECV and bootstrap, were recommended as countermeasures for the selection bias. Krawczuk et al. [14] investigated the feature selection bias of ICV in an empirical study of 4 feature selection methods applied to 7 microarray datasets. They found positive selection biases in all the cases, though the biases vary from case to case. However, there are some different conclusions. Singhi and Kiu [15] concluded that the ICV is not inappropriate for classification using a Bayesian perspective for synthetic datasets and real text datasets.

The feature selection bias of ICV has not received sufficient attention, especially with respect to genomic studies. There are many papers that used all samples for feature selection or did not state clearly that the test samples are external to feature selection procedure. For example, Derringer et al. [16] used all samples to select SNPs that are significantly associated with the sensation seeking and used a selected subset to build a model, in an attempt to show that a system-level approach can identify novel SNPs. They did not perform CV because of the small number of samples. Briones and Dinu [17] reported that they achieved an error rate of 9.8% using random forest based on APOE and GAB2 and a subset of pre-selected 199 SNPs in a SNP dataset related to Alzheimer's disease (AD). They used 10-fold CV but they did not mention that they repeated the feature selection procedure for each fold of the CV; that is, they probably used ICV.

In this thesis, we will demonstrate the feature selection bias of ICV in GWAS data using an example of late-onset Alzheimer's disease (LOAD). We used a real dataset related to AD from Mayo Clinic, and two synthetic datasets with synthetic phenotype variables generated from the real SNPs for this demonstration. The synthetic phenotype was generated by simple logistic regression models with coefficients varying in size, but all based on ten fixed SNPs randomly selected from the real dataset. Synthetic dataset1 has small coefficients representing weak signals which are generated from $N(0, 0.1^2)$. Synthetic dataset2 has large

coefficients representing strong signals which are generated from $N(0, 2^2)$. The coefficients of the rest SNP were set equal to 0, which means those SNPs were treated as noises. We conducted typical GWAS for each SNP by performing likelihood ratio test (LRT) comparing two logistic regression models, one with $\varepsilon 4$ alleles of apolipoprotein E (APOE $\varepsilon 4$) dosages (a known predictor of Alzheimer's) only, and the other with APOE $\varepsilon 4$ dosages and a single SNP. Then we converted the p-values into the tail-based false discovery rate (Fdr). Different subsets were selected from ICV and each fold of ECV based on the ordering of the Fdr. Given different selected subsets of features, we fitted the regularized logistic regression with different penalties, including L_1 LASSO penalty [18] and the combination of L_1 and L_2 elastic-net penalty [19]. Furthermore, we used the fully Bayesian hyper-LASSO with a t-prior [20], which had a heavier tail, to achieve a more sparse model. Regularized regression shrinks many coefficients to zero, which introduces bias but reduces the variance of predicted values, thus improving overall predictivity. Three predictive metrics, error rate [21], average minus log probability (AMLP) [20] and AUC [22], were used to assess the predictivity of selected subsets.

For the real dataset, if we use the ECV method, we find that no other SNPs can improve the prediction of LOAD using only the APOE $\varepsilon 4$ dosages, which has an error rate of 0.3. However, a subset of SNPs pre-selected based on all samples can give an error rate of 0.07, which decreases by 83% from the error rate (0.3) of using only APOE $\varepsilon 4$ dosages. The AUC of a subset of 2^{12} pre-selected SNPs from ICV can reach 0.98. For the synthetic datasets, the results in the dataset where only weak signals exist are similar to the real data application. Moreover, the predictivity estimates of pre-selected SNPs based on all samples can be even better than the oracle predictivity of the truly related SNPs that are used to generate the phenotype. When there are strong signals in the dataset, the top SNPs selected using ECV can improve the predictive performance of the models. We find that the Fdr is very informative to detect the false discovery. We also find that hyper-LASSO has a better performance than LASSO and elastic-net. As more noises are added to the model, hyper-LASSO is more stable to maintain a good performance than LASSO and elastic-net.

The remaining of this thesis will be organized as follows. In Chapter 2, we will describe the methodologies that will be used in this thesis. We will describe conducting typical GWAS

with LRT and describe the Fdr. We will discuss the three penalized logistic regressions and the algorithms to find the estimations of the coefficients. We will also compare the different procedures between ICV and ECV. Then we will describe the predictive metrics that will be used to assess the predictivity of selected SNPs from ICV and ECV. In Chapter 3, we will depict what Alzheimer's disease is and recent knowledge about Alzheimer's disease. We will then introduce the real dataset and two synthetic datasets. In Chapter 4, we present the results of the real dataset and the synthetic datasets and use the empirical study to explain why ECV can avoid the feature selection bias. Finally, in Chapter 5, we conclude this thesis by summarizing our findings and discussing advancements for the future. In Appendices, we present the R codes for producing the analysis in this thesis.

2. Methodology

2.1 GWAS and False Discovery Rate

In this section, we introduce how a typical GWAS uses single SNP analysis to select features. We also describe the use of the false discovery rate to correct for multiple comparison problems.

2.1.1 GWAS with Likelihood Ratio Test

GWAS is a way for scientists to identify genetic variants associated with risks of a disease or a particular trait. This method scans the whole genome for genetic polymorphisms, typically SNPs, that occur more frequently in cases than in controls. Once such SNPs are identified, people can use them to understand how genes contribute to the disease and develop better prevention and treatment strategies. In the past decade, thousands of SNPs have been identified to have a strong statistical association with many common diseases (for instance, type 2 diabetes and AD) through single SNP analysis. Single SNP analysis tests each SNP individually for the association of phenotype using statistical significance test (e.g., χ^2 test, Fisher's exact test).

A case-control GWAS measures a sample of n individuals and p genotyped SNPs, where $n \ll p$. This type of data is also called high-dimensional data. We denote the binary indicator for phenotype of individual i by y_i. Typically, y_i is coded as 0 for controls and 1 for cases. SNPs have three categories, 0, 1, and 2, which correspond to the number of minor alleles of the genotype. We can fit a logistic regression model to test the statistical significance of each SNP. Let $x_i^{(j)}$ denote the row indicator vector to represent SNP_j for individual i, For example, we take the category of SNP which equals to 0 as reference category, then $x_i^{(j)} = (0,0)$ for the SNP_j for individual i equals 0, $x_i^{(j)} = (1,0)$ for 1 and $x_i^{(j)} = (0,1)$ for 2. Thus $\boldsymbol{x}^{(j)} = (x_1^{(j)}, x_2^{(j)}, ..., x_n^{(j)})$ is the vector of SNP_j for all individuals. Let β_j denote the vector of regression coefficients associated with SNP_j and β_0 denote the intercept. The

logistic regression of SNP_j can be written as:

$$Pr(y_i = 1) = \frac{e^{\beta_0 + x_i^{(j)}\beta_j}}{1 + e^{\beta_0 + x_i^{(j)}\beta_j}}. \tag{2.1}$$

We can estimate β_j by minimizing the negative log likelihood:

$$l(\beta_0, \beta_j) = -\frac{1}{n}\sum_{i=1}^{n}(y_i(\beta_0 + x_i^{(j)}\beta_j) - \log(1 + e^{(\beta_0 + x_i^{(j)}\beta_j)}). \tag{2.2}$$

Since the APOE $\varepsilon 4$ has been known to scientists to have an effect on Alzheimer's disease [23], we consider testing the significance of SNP_j conditional on APOE $\varepsilon 4$ to look for novel genetic variants. Likelihood ratio test (LRT) is a statistical test used for comparing the goodness of fit between a null model and an alternative model. Let $x_i^{(APOE)}$ denote the vector of APOE $\varepsilon 4$ allele for individual i. We fit two logistic regression models to test the statistical significance of SNP_j, denoted by f. In LRT, the model assumptions are as follows:

$$\begin{aligned} H_0 &: y_i \sim f(y_i | x_i^{(APOE)}), \\ H_1 &: y_i \sim f(y_i | x_i^{(APOE)}, x_i^{(j)}), \end{aligned} \tag{2.3}$$

Denote the maximized likelihoods of the null model and the alternative model by $L_0^{(j)}$ and $L_1^{(j)}$, respectively. Then the log likelihood ratio is defined as

$$\Lambda^{(j)} = -2(\log L_0^{(j)} - \log L_1^{(j)}). \tag{2.4}$$

This statistic asymptotically follows a χ^2 distribution with $C - 1$ degree of freedom [24], where C is the number of variants in SNP_j. P-values are assigned by measuring the area of the χ^2 distribution to the right the test statistic $\Lambda^{(j)}$.

2.1.2 False Discovery Rate

When testing a single hypothesis test, we choose a rejection threshold, α_0, to control the false positive (Type 1 error). If we have m multiple simultaneous tests, family-wise error rate (FWER) is the probability of at least one Type 1 error in the m multiple tests, which is given by

$$\text{FWER} = 1 - (1 - \alpha_0)^m. \tag{2.5}$$

FWER will increase as m increases and will approach 1 if m approaches to infinite. For instance, we have 10,000 single hypotheses, and each of them has a rejection threshold with 0.05. Then we will have FWER $= 1-(1-0.05)^{10000} \approx 1$. Therefore, traditional approaches try to set stricter cutoffs to avoid underestimating the chance of false discovery. The conventional approach is the Bonferroni method. Bonferroni method corrects the probability of false positives by setting the cutoff of each test to be α_0/m to guarantee that FWER $\leq \alpha_0$. This method uses a very stringent criterion, which will increase the false negative (Type 2 error) rate, that is, making the power of discovering true positives small.

Table 2.1: Possible outcomes from m hypothesis tests.

	Accept null	**Reject null**	**Total**
Null true	U	V	m_0
Alternative true	T	S	m_1
Total	W	R	m

Table 2.1 displays all the possible outcomes when testing m null hypotheses. Benjamini and Hochberg [3] proposed the false discovery rate (FDR) as a measure of test error in multiple hypotheses. FDR is defined as

$$\text{FDR} = E\Big[\frac{V}{R}\Big|R > 0\Big] Pr(R > 0). \tag{2.6}$$

FDR offers a less strict control over false positive than FWER, but this FDR guarantees that the right side of equation 2.6 is less than a desired significance level α_0. Storey [25] introduced the positive FDR (pFDR) based on the FDR, which is defined as

$$\text{pFDR} = E\Big[\frac{V}{R}\Big|R > 0\Big]. \tag{2.7}$$

The difference between FDR and pFDR is that pFDR deals with the problem of $R = 0$, which means there is no genetic variant related to a trait. pFDR is better defined than the FDR because there are cases where $Pr(R = 0) > 0$. Storey [26] also presents a Bayesian interpretation of pFDR. Both FDR and pFDR are tail-based FDR (Fdr), which is based on the tail areas of test statistics, including p-values, correlations, z-scores or t-scores.

Consider the distribution function of a two-component mixture model of the observed p-values,

$$F(p) = \pi_0 F_0(p) + (1 - \pi_0) F_A(p) = \pi_0 + (1 - \pi_0) F_A(p), \tag{2.8}$$

where π_0 is the prior probability of null hypothesis, F_0 is the null cumulative density of p-values, which is the uniform distribution $U(0, 1)$ and corresponds to the "uninteresting" p-values, whereas F_A is an unspecified alternative cumulative density for the "interesting" p-values. Suppose we have m p-values $p_1, p_2, ..., p_m$ from m hypothesis tests, the $\text{Fdr}(p_i)$ is the Fdr of ith feature, which is defined as:

$$\text{Fdr}(p_i) = Pr('\text{uninteresting}'|p \leq p_i) = \pi_0 \frac{F_0(p_i)}{F(p_i)}. \tag{2.9}$$

Hence, estimating Fdr involves identifying the alternative model F_A and finding suitable estimates for the prior probability of null hypothesis π_0. We used the R package `fdrtool` to convert the p-values into Fdr. This package uses the "modified" Grenander estimator obtained as estimator of $F(p)$ and uses truncated maximum-likelihood approach to estimate π_0 [27].

2.2 Regularized Logistic Regression

In this section, we introduce three regularized logistic regression models and methods to estimate the coefficients.

2.2.1 LASSO

We consider three regularized logistic regression methods to fit a selected subset of SNPs. Suppose we have a selected subset of SNPs with size k, denoted by $\{s_1, s_2, ..., s_k\}$. $\boldsymbol{x_i} = (x_i^{(s_1)}, x_i^{(s_2)}, ..., x_i^{(s_k)})^T$ is the vector of selected SNPs for individual i, and β_0 is the intercept and $\beta = \{\beta_{s_1}^T, \beta_{s_2}^T, ..., \beta_{s_k}^T\}^T$ is the parameter vector. The simple logistic regression of the selected SNPs is:

$$Pr(y_i = 1) = \frac{e^{\beta_0 + \boldsymbol{x_i}^T \beta}}{1 + e^{\beta_0 + \boldsymbol{x_i}^T \beta}} \tag{2.10}$$

We can estimate β by minimizing the negative log-likelihood:

$$l(\beta_0, \beta) = -\frac{1}{n}\sum_{i=1}^{n}(y_i(\beta_0 + \boldsymbol{x}_i^T\beta) - \log(1 + e^{(\beta_0 + \boldsymbol{x}_i^T\beta)})). \tag{2.11}$$

To obtain sparse solutions and enhance the predictive performance, we add L_1 LASSO (Least absolute shrinkage and selection operator) penalty. The LASSO estimator is obtained from the penalized minus log-likelihood:

$$\hat{\beta}_{LASSO}(\lambda_1) = \underset{\beta_0,\beta}{\operatorname{argmin}}\ l(\beta_0, \beta) + \lambda_1||\beta||_1, \tag{2.12}$$

where $||\beta||_1 = \sum_{j=1}^{p}|\beta_j|$, p is total number of dummy variables of selected SNPs, and λ_1 is tuning parameter. Note that intercept is not included in the penalty term. LASSO penalty corresponds to a Laplace prior in Bayesian inference. Hence, it will get a subset of important features with non-zero coefficients and shrink the rest to zero. Increasing λ_1 will shrink more coefficients to zero by adding heavier penalty. Because this optimization problem is convex, it can be solved efficiently for large data. There are several algorithms for calculating the LASSO estimator, among which coordinate descent performs the best [28]. Coordinate descent optimizes each parameter separately while holding all others fixed. We will describe the algorithm in detail in Section 2.2.2.

2.2.2 Elastic-net

Elastic-net [19], a combination of L_1 and L_2 penalties, solves the convex problem:

$$\hat{\beta}_{EN}(\alpha, \lambda_2) = \underset{\beta_0,\beta}{\operatorname{argmin}}\ l(\beta_0, \beta) + \lambda_2[(1-\alpha)||\beta||_1 + 1/2\alpha||\beta||_2], \tag{2.13}$$

where $||\beta||_2 = \sum_{j=1}^{p}\beta_j^2$, $\alpha \in (0, 1)$ controls the weight between L_1 penalty and L_2 penalty. λ_2 is tuning parameter. When $\alpha = 0$, elastic-net reduces to LASSO penalty, and when $\alpha = 1$, elastic net reduces to ridge penalty. Ridge penalty can shrink the coefficients of correlated predictors towards each other. However, it will not shrink the coefficients to be exactly 0. LASSO tends to select one feature and ignores the rest when there are several features correlated. Elastic-net mixes the characteristics of LASSO and ridge regression. It can effectively shrink the coefficients of non-informative feature to 0 and automatically control the group of correlated features.

To obtain the estimation of $\hat{\beta}_{EN}$, let us first look at equation (2.11). Since equation (2.11) is a non-linear function, it is not possible to find a closed-form expression for $\hat{\beta}_{EN}$. The algorithm for estimating $\hat{\beta}_{EN}$ is known as the iteratively reweighted least squares (IRLS) algorithm, which implements a Taylor expansion to produce quadratic approximations to the loss function. IRLS is equivalent to Newton's method. For current estimates of parameters $(\tilde{\beta}_0^{(t)}, \tilde{\beta}^{(t)})$, the quadratic approximation to the log-likelihood (Taylor expansion) can be expressed:

$$l_Q(\beta_0, \beta) = -\frac{1}{2n}\sum_{i=1}^{n} w_i(z_i - \beta_0 - \boldsymbol{x}_i^T\beta)^2 + C(\tilde{\beta}_0^{(t)}, \tilde{\beta}^{(t)}), \tag{2.14}$$

where

$$z_i = \tilde{\beta}_0^{(t)} + \boldsymbol{x}_i^T\tilde{\beta}^{(t)} - \frac{1}{w_i}(\tilde{p}^{(t)}(\boldsymbol{x}_i) - y_i) \tag{2.15}$$

$$w_i = \tilde{p}^{(t)}(\boldsymbol{x}_i)(1 - \tilde{p}^{(t)}(\boldsymbol{x}_i)) \tag{2.16}$$

$$\tilde{p}^{(t)}(\boldsymbol{x}_i) = \frac{\exp(\tilde{\beta}_0^{(t)} + \boldsymbol{x}_i^T\tilde{\beta}^{(t)})}{1 + \exp(\tilde{\beta}_0^{(t)} + \boldsymbol{x}_i^T\tilde{\beta}^{(t)})}, \tag{2.17}$$

$C(\tilde{\beta}_0^{(t)}, \tilde{\beta}^{(t)})$ is a constant.

Similar to simple logistic regression, IRLS is used to estimate penalized logistic regression. Friedman et al. [28] developed a fast algorithm to solve penalized weighted least-squares problem

$$R(\beta_0, \beta) = \underset{\beta_0,\beta}{\text{argmin}}\ \{l_Q(\beta_0, \beta) + \lambda_2[(1-\alpha)||\beta||_1 + 1/2\alpha||\beta||_2]\}. \tag{2.18}$$

using coordinate descent. For a fixed $\lambda_2^{(r)}$ as the value in rth step of looping λ_2, we compute the gradient at $\beta_j = \tilde{\beta}^{(t)}$, which only exists if $\tilde{\beta}_j{}^{(t)} \neq 0$. If $\tilde{\beta}_j^{(t)} > 0$, then

$$\frac{\partial R}{\partial \beta_j}\Big|_{\beta=\tilde{\beta}^{(t)}} = -\frac{1}{n}\sum_{i=1}^{n} w_i(z_i - \tilde{\beta}_0^{(t)} - \boldsymbol{x}_i^T\tilde{\beta}^{(t)}) + \lambda_2^{(r)}(1-\alpha)\beta_j + \lambda_2^{(r)}\alpha. \tag{2.19}$$

The coordinate-wise update has the form:

$$\tilde{\beta}_j^{(t)} \leftarrow \frac{S(\frac{1}{n}\sum_{i=1}^{n} w_i x_i^{(j)}(y_i - \tilde{y}_i^{(j)}), \lambda_2^{(r)}\alpha)}{1 + \lambda_2^{(r)}(1-\alpha)}, \tag{2.20}$$

where $\tilde{y}_i^{(j)} = \tilde{\beta}_0^{(t)} + \sum_{l\neq j} x_{il}\tilde{\beta}_l^{(t)}$ is the fitted value excluding the contribution from x_{ij}, $y_i - \tilde{y}_i^{(j)}$ is the partial residual given $\beta_j^{(t)}$, and $S(a, b) = \text{sgn}(a)(|a|-b)+ = \text{sgn}(a)\max(|a|-b, 0)$ is the soft-thresholding operator.

In brief, $\hat{\beta}_{EN}(\lambda_2)$ is estimated from the nested loop:

- An outer Newton loop of decreasing value of λ_2.
- A middle loop for updating the quadratic approximation l_Q using current parameters $(\tilde{\beta}_0^{(t)}, \tilde{\beta}^{(t)})$ given the estimate for λ_2 at the rth step, $\lambda_2^{(r)}$.
- An inner loop for running the coordinate descent algorithm on the penalized weighted least-squares problem (2.18).

LASSO is a special case of elastic-net with $\alpha = 0$. The coordinate-wise update for LASSO has the form:

$$\tilde{\beta}_j^{(t)} \leftarrow \frac{S(\frac{1}{n}\sum_{i=1}^n w_i x_i^{(j)}(y_i - \tilde{y}_i^{(j)}), \lambda_1^{(r)})}{1 + \lambda_1^{(r)}}, \tag{2.21}$$

where $\lambda_1^{(r)}$ is the estimate for λ_1 in the rth step of the outer loop.

We use the R package `glmnet` for fitting LASSO and elastic-net. This package provides an option for choosing tuning parameters λ_1 and λ_2. We fit the regularization path with 500 values for both λ_1 and λ_2. They are selected by 10-fold cross-validation of training data with the minimum mean cross-validated error. Instead of gridding every possible value of α in elastic-net, we grid three values of α, 0.3, 0.5, and 0.7, to save the computation time.

LASSO and elastic-net are powerful methods for high-dimensional data learning problems. However, both LASSO and elastic-net have some drawbacks. For instance, they over-shrink the large coefficients while shrinking small coefficients, resulting in the bias in estimating the large coefficients. Another drawback of LASSO is that it can only select at most n features when $p > n$.

2.2.3 Fully Bayesian hyper-LASSO

It is important to get a very sparse model but also to preserve the large coefficients of important SNPs for genome-wide data. Hyper-LASSO penalty, a non-convex penalty, which is also known as global local penalty, has been widely recognized for its ability to shrink the coefficients of unrelated features (noise) more aggressively to 0 than LASSO while retaining the significantly large coefficients (signal). Hyper-LASSO can also automatically divide a group of correlated features into different posterior local modes [20]. Consider hyper-LASSO

with t-prior with α degrees of freedom and scale $\sqrt{w}$. The hierarchical Bayesian logistic regression model can be described as follows:

$$Pr(y_i = 1|\boldsymbol{x}_{i,1:p}, \beta_0, \beta_{1:p}) = \frac{e^{\beta_0 + \boldsymbol{x}_i^T \beta}}{1 + e^{\beta_0 + \boldsymbol{x}_i^T \beta}}, \tag{2.22}$$

$$\beta_j|\sigma_j^2 \sim N(0, \sigma_j^2), \text{ for } j = 0, 1, ..., p, \tag{2.23}$$

$$\sigma_j^2 \sim IG(a/2, wa/2), \text{ for } j = 1, 2, ..., p, \tag{2.24}$$

where σ_j^2 indicates the importance of the jth SNP dummy variable. With σ_j^2 marginalized with respect to Inverse-Gamma prior, equations (2.23) and (2.24) assign β_j a multivariate t prior with a degrees of freedom and scale $\sqrt{w}$. The full posterior can be written as:

$$P(\beta_{0:p}, \sigma_{1:p}^2|\boldsymbol{D}) \propto L(\beta_{0:p}) \times P(\beta_{0:p}|\sigma_{0:p}^2) \times P(\sigma_{1:p}^2|a/2, aw/2), \tag{2.25}$$

where $\boldsymbol{D}$ represents the data $y_i, \boldsymbol{x}_{i,1:p}$; α and σ_0^2 are fixed values; L is the likelihood function: $L(\beta_{0:p}) = \prod_{i=1}^n P(y_i|x_{i,1:p}, \beta_{0:p})$; the last two parts are the PDFs of the priors specified by equations (2.23) and (2.24). The full posterior in equation (2.25) is sampled by sampling the conditional distributions of $\sigma_{1:p}^2$ and $\beta_{0:p}$ given each other alternately for a number of iterations. Gibbs sampling is used to sample the priors, which involves alternating the following two steps:

Step1: Given $\sigma_{1:p}^2$ fixed, update $\beta_{0:p}$ jointly with an HMC transformation that leaves invariant the following distribution:

$$P(\beta_{0:p}|\sigma_{0:p}^2, \boldsymbol{D}) \propto L(\beta_{0:p}) \times P(\beta_{0:p}|\sigma_{0:p}^2). \tag{2.26}$$

Step2: Given the value of $\beta_{1:p}$ from Step1, update $\sigma_{1:p}^2$ by sampling from

$$\sigma_j^2|\beta_j \sim IG(\sigma_j^2|\frac{a+1}{2}, \frac{aw + \beta_j^2}{2}), \text{ for } j = 1, ..., p. \tag{2.27}$$

In this thesis, we use a R package `HTLR` (`https://math.usask.ca/~longhai/`) to perform hyper-LASSO with t-prior. In order to try different degrees of freedom for different datasets, we choose the degree of freedom a to be 0.5, 1, and 1.5. The smaller value of a represents the heavier tail. We set $\log(w) = -10$ as recommended by Li and Yao [20].

2.3 Internal and External Cross-Validation

In this section, we introduce the general idea of cross-validation and two types of cross-validation that are often used. Then we will explain why ECV can avoid feature selection bias.

2.3.1 Cross-Validation

Cross-validation (CV) is a widely used method to assess machine learning models when we have a small dataset. CV reserves part of a dataset for validation or testing and uses the remaining part to train the model. This process iterates without overlapping of the reserved samples. Two most frequently used CV methods are Leave-one-out CV (LOOCV) and K-fold CV. LOOCV means reserving only one data point from the available dataset and training the model by the rest of the data. This process iterates for each data point. K-fold CV means that splitting the samples into k folds with approximately equal size, using the $K-1$ folds as training data and the left 1 fold as test data.

2.3.2 Internal Cross-Validation

ICV means test samples are internal to feature selection when implementing CV. A typical procedure of ICV is:

- Select a subset of feature based all samples;
- Split the samples into K non-overlapping folds with roughly equal size at random;
- For each $k = 1, 2, ..., K$, implement cross-validation to choose tuning parameters and build models based on the samples except those in fold k and estimate the predictivity in fold k.

The problem of this procedure is that the features are chosen on the bias of all samples. Leaving the test samples out after the features have been selected does not correctly mimic the application of the model to a completely independent test set, since the features "have already seen" the left out samples.

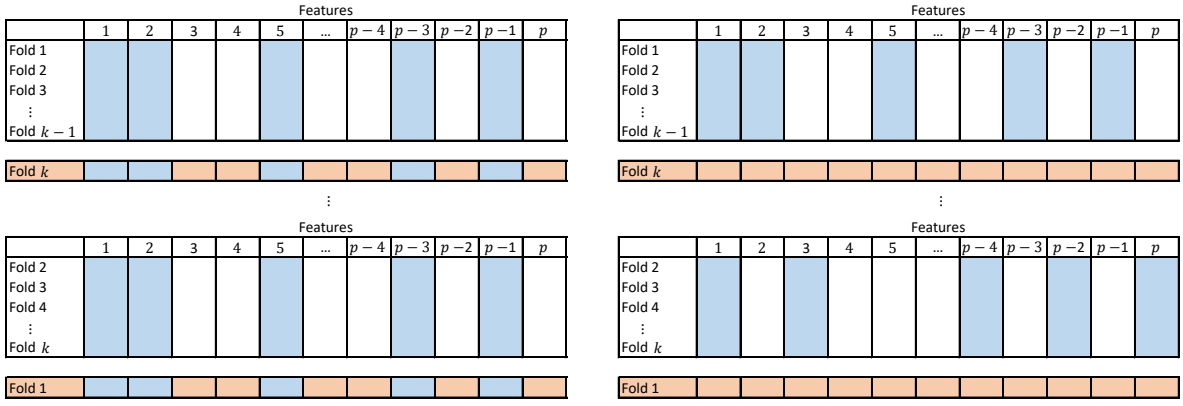
2.3.3 External Cross-Validation

To correct for the feature selection bias in ICV, test samples must be external to feature selection process. The procedure to carry out ECV is described as follows:

- Split the samples into K non-overlapping folds with roughly equal size at random;
- For each $k = 1, 2, ..., K$, choose tuning parameters, and train the model using selected subsets of features based on the samples except those in fold k.
- Use the model to predict for samples in fold k.

The difference in procedure between ICV and ECV is using different samples to select features. While CV is implemented to choose tuning parameters and train the models equally for ICV and ECV.

Figure 2.1: A figure of ICV and ECV. Blue shadow represents the selected features. Orange shadow represents the test set.



(a) Internal Cross-validation **(b)** External Cross-validation

Figure 2.1 shows the difference between ICV and ECV. ICV uses all samples to select features, thus resulting in the same subset of features being used in training dataset of CV. Because we look at the test samples before training the model, the selected features will have correlations with test samples. It is not surprising that the test error rate will underestimate the true error rate. When we have a set of independent samples, the predictive performance of the best model will be worse in the independent dataset than in the test set. ECV, on

the other hand, will select different subsets of features in training dataset in each fold of CV. The test set is held out and only used when testing the predictivity of the selected subset of features.

2.4 Predictive Metrics

In this section, we describe three metrics to rate the predictive performance of the models.

2.4.1 Error Rate and AMLP

Let $\hat{P}_i(y_i|x_i)$ be the predictive probability function for y_i. The point prediction for y_i is $\hat{y}_i = \text{argmax}\,\hat{P}_i(y_i|\boldsymbol{x}_i)$, where $\boldsymbol{x_i} = (x_i^{(s_1)}, x_i^{(s_2)}, ..., x_i^{(s_k)})^T$ is the vector of selected SNPs for individual i. We assess the goodness of $\hat{P}_i(y_i|\boldsymbol{x}_i)$ with the observed y_i. The first metric is **error rate**. The error rate is defined as the proportion of misclassification:

$$\text{ER} = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i \neq y_i). \tag{2.28}$$

The second metric is the average of minus log predictive probabilities (**AMLP**) at the observed y_i:

$$\text{AMLP} = -\frac{1}{n}\sum_{i=1}^{n} \log(\hat{P}_i(y_i|x_i)). \tag{2.29}$$

AMLP is more sensitive than ER because AMLP measures not only the correctness of a point estimate $\hat{y}_i$ but also the degree of correctness expressed by $\hat{P}_i(y_i|x_i)$. Both error rate and AMLP should be interpreted relative to the baseline, which is solely based on the frequency of y_i without using any predictor. Denote the frequency of $y_i = 1$ by $f_1 = \frac{1}{n}\sum_{i=1}^{n} y_i$ and the frequency of $y_i = 0$ by $f_0 = 1 - f_1$. $ER^{(0)} = \min\{f_1, f_0\}$ is the baseline error rate. And $AMLP^{(0)} = -[f_1 \log(f_1) + f_0 \log(f_0)]$ is the baseline AMLP. Suppose we have unbalanced data with 80% controls and 20% cases, the baseline error rate is 20% and the baseline AMLP is $-[0.8\log(0.8) + 0.2\log(0.2)] = 0.217$. Therefore, a model with 20% error rate or 0.217 AMLP is not a good model compared to the baseline. To assess the predictive performance of the model, we should compare the result to the baseline error rate and baseline AMLP. Therefore, we define R^2_{ER} by

$$R^2_{ER} = \frac{ER^{(0)} - ER}{ER^{(0)}}. \tag{2.30}$$

Similarly, we define R^2_{AMLP} by

$$R^2_{AMLP} = \frac{AMLP^{(0)} - AMLP}{AMLP^{(0)}} \tag{2.31}$$

2.4.2 AUC

The third metric is **AUC** (Area Under the Curve), which measures the area under the ROC curve. AUC represents a trade-off between sensitivity (true positive rate) and specificity (false positive rate). They are defined as:

$$\text{sensitivity} = \frac{\#\text{ of true positives}}{\#\text{ of true positives} + \#\text{ of false negatives}}, \tag{2.32}$$

$$\text{specificity} = \frac{\#\text{ of true negatives}}{\#\text{ of true negatives} + \#\text{ of false positives}}. \tag{2.33}$$

Unlike error rate or AMLP that requires a decision threshold (usually take 0.5) to discriminate, AUC is independent of the decision threshold. It measures each possible performance as the decision threshold is varied. For every cutoff point $c \in [0, 1]$, y_i is predicted by $\hat{y}_i = I(\hat{P}_i(y_i|x_i) \geq c)$, where $I(\cdot)$ is the indicator function which is equal to 1 if the condition in bracket is true, 0 otherwise. Sensitivity and specificity are determined by comparing $\hat{y}_i$ and the true label y_i. The ROC curve is plotted based on the sensitivity and specificity of each cutoff point. AUC is the area under the ROC curve. The baseline AUC is 0.5, which is the AUC of the random guess (or the random outcome of a coin toss). An AUC equal to 1 represents that the prediction is perfect.

For the three predictive metrics mentioned above, the averaged error rate, AMLP and AUC among 10 folds of test data are reported as a final classification performance for each model.

3. Data

3.1 Alzheimer's Disease (AD)

Alzheimer's disease (AD) is a chronic neurodegenerative disease associated with age. AD is the most common cause of dementia and it affects more than 50 million people worldwide. There are two types of AD, early-onset AD and late-onset AD (LOAD). Early-onset AD occurs before a person's age reaches the mid-60s and it represents less than 10 percent of all the people with AD. It is well known that three mutations: amyloid β protein precursor, presenilin 1, and presenilin 2 cause the early-onset familial form of AD with autosomal dominant inheritance. These three genes together account for 60% to 70% of early-onset AD but only 1% for all Alzheimer's. LOAD is the most common type of AD, with symptoms becoming apparent after the mid-60s. So far, the cause of LOAD has not been completely understood. There is still no effective treatment or preventive measure for LOAD. The only well-established susceptibility allele for LOAD is the APOE gene on chromosome 19. There are three kinds of APOE genes. APOE $\varepsilon 2$ is rare and may be protected against the disease. APOE $\varepsilon 3$ is believed to play a neutral role in the disease. APOE $\varepsilon 4$ increases the risk for developing AD and is also related to the early-onset form of AD.

In the past 15 years, many GWASs have been conducted to identify novel genetic loci for LOAD [1, 2, 29, 30]. APOE shows strong evidence for association in those studies. However, the associations of all other new risk alleles are much less strong than APOE. The GWAS Catalog lists around 140 genetic variants and risk alleles obtained from many AD GWAS. The AlzGene database also lists over ten "Top Results" that satisfy all the criteria obtained from different Meta-Analysis GWAS. Some of the genetic variants are replicated in different GWAS.

3.2 An Whole-Genome Sequence Data Related to AD

The LOAD GWAS data that we used was collected and originally analyzed by Carrasquillo et al. [1]. This dataset was downloaded from the link: https://www.synapse.org/#!Synapse:syn5591675. Genotypes were collected from Mayo Clinic LOAD GWAS using Illumina Human-Hap 3000 BeadChips, which includes a file in PLINK format with 313504 SNPs from 22 autosomes, chromosome X, and an independent file containing the dosages of APOE $\varepsilon 4$ alleles. The data consists of 2099 subjects with an age at diagnosis of 60-80 years, of which 844 cases had LOAD and 1255 controls had no LOAD. For quality control, we used PLINK to eliminate SNPs with as missing rate greater than 5%, SNPs with minor allele frequency less than 0.01, and SNPs and samples with a call rate less than 90%. Missing data were imputed with the mode of the SNP. After pre-processing, 309549 SNPs and 2099 participants were included in the analysis.

SNPs from 22 autosomes were coded as 0, 1, and 2 representing the number of minor allele copies. For SNPs from chromosome X, females have three categories, while males only have two categories, 0 and 1. To make use of SNPs from chromosome X, we included sex as a covariate to generate a five-category variable. Hence, all of the SNPs were included as vectors of dummy variables in the logistic regression models.

3.3 Synthetic Datasets

We examined the feature selection bias of ICV and the performance of various models in two different datasets. We generated two synthetic datasets based on real data. We chose 10 SNPs in chromosome 19 and APOE $\varepsilon 4$ from the real data as truly related genetic factors (true signals). We assigned them two different sets of synthetic coefficients. One set of coefficients with large values represents the scenario in which there are strong signals in a dataset while the other set with small values represents the scenario with only weak signals in a dataset. The intercept and coefficients of APOE $\varepsilon 4$ in the two datasets mimicked the coefficients calculated from the real data analysis. The coefficients of the ten SNPs were generated from a normal distribution with mean zero and different standard deviations. The

values of synthetic coefficients are displayed in Table 3.1. The coefficients of the remaining SNPs were set to 0, which were treated as noise. The phenotype y_i was generated from a logistic regression model given the set of coefficients.

Table 3.1: Synthetic parameters of selected SNPs.

genetic loci	β in dataset1	β in dataset2	genetic loci	β in dataset1	β in dataset2
apoe-1	1.00	1.00	rs8106922-2	-0.18	1.84
apoe-2	2.00	2.00	rs405509-1	-0.05	3.44
rs2075650-1	-0.05	-2.45	rs405509-2	0.08	1.65
rs2075650-2	-0.04	0.35	rs8039031-1	-0.06	0.77
rs157580-1	0.05	-1.18	rs8039031-2	-0.03	-3.29
rs157580-2	-0.14	-3.53	rs7318037-1	0.03	1.31
rs439401-1	-0.13	2.19	rs7318037-2	0.09	-0.02
rs439401-2	-0.00	-1.35	rs1420566-1	-0.07	0.73
rs6859-1	-0.03	-1.77	rs1420566-2	0.11	-0.67
rs6859-2	-0.12	-1.80	rs10402271-1	-0.25	-2.98
rs8106922-1	-0.02	2.43	rs10402271-2	-0.04	-3.34

The process for generating each dataset can be described as follows:

- Select APOE $\varepsilon 4$ and 10 SNPs, denoted by $x_i^{(f)}$, for $f = 0, 1, ..., 10$, which are used as covariates for generating y_i.
- Fix the value of intercept $\beta_0 = -1$ and coefficient of APOE $\varepsilon 4$ $\beta^{(0)} = (1, 2)$. Generate two coefficient vectors β for $x_i^{(f)}$, for $f = 1, ..., 10$, from normal distribution $N(0, \sigma^2)$, where $\sigma = 0.1$ for dataset1 and $\sigma = 2$ for dataset2.
- For $i = 1, 2, ..., 2099$, the phenotype is generated from a Bernoulli distribution:

$$y_i \sim \text{Bernoulli}(p_i), \tag{3.1}$$

where

$$p_i = Pr(y_i = 1 | x_i^{(0)}, x_i^{(1)}, ...x_i^{(10)}) = \frac{1}{1 + \exp(-(\beta_0 + \sum_{f=0}^{10} x_i^{(f)} \beta^{(f)}))}, \tag{3.2}$$

The synthetic datasets consist of the same number of SNPs and samples as the real data. Dataset1 has 869 cases with $y_i = 1$ and 1230 controls with $y_i = 0$. Dataset2 has 1052 cases with $y_i = 1$ and 1047 controls with $y_i = 0$.

4. Results and Discussions

4.1 Results on the AD SNP Data

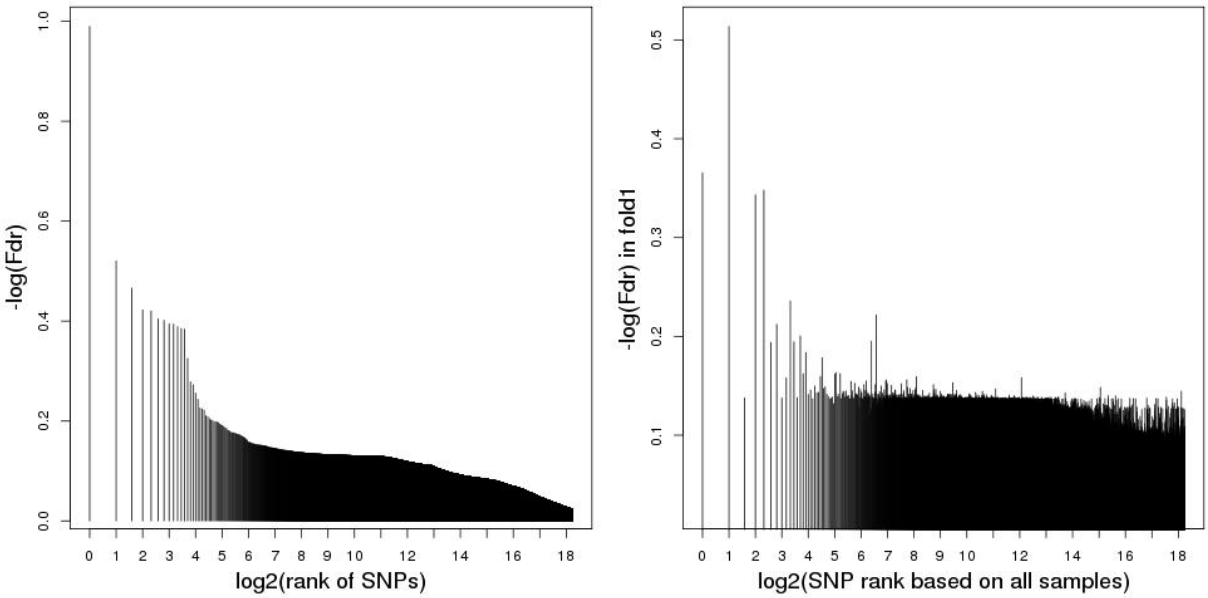
In this section, we demonstrated the feature selection bias of ICV in GWAS data using the real Alzheimer's disease dataset described in Section 3.2. We applied predictive analysis to this real dataset to measure the predictive estimates of selected subsets of SNPs. The sample size n of this dataset is 2099, in which 844 participants have LOAD and 1255 participants have no LOAD. We applied 10-fold CV by splitting the dataset into 10 folds of approximately equal size. When using ECV method, we re-selected subsets of SNPs based on the training samples in each fold of CV. When using ICV, we pre-selected subsets of SNPs based on all samples and then applied CV to train the model.

We first implemented GWAS by conducting LRT for each SNP conditional on APOE $\varepsilon 4$ to compute a p-value for each SNP. Then we converted the p-values into the tail-based false discovery rate using the R package `fdrtool` [27] in order to calculate the chance of false discovery. We then ordered SNPs by $-\log_{10}(\text{Fdr})$ values to generate an ordered list of SNPs. To compare the ordered list of SNPs based on all samples with the ordered list of SNPs based on the training samples in one fold in ECV, we show the ordered Fdrs based on all samples and the Fdrs of fold 1 in ECV given the SNP ranking based on all samples in Figure 4.1. Figure 4.1a shows the values of $-\log(\text{Fdr})$ of ordered SNPs based on all samples. Figure 4.1b shows the values of $-\log(\text{Fdr})$ of SNPs calculated based on the training samples of fold 1 in ECV corresponding to the ordered SNPs by using all samples. It can be seen that the largest $-\log_{10}(\text{Fdr})$ in Figure 4.1a is around 1, whereas the largest $-\log_{10}(\text{Fdr})$ in Figure 4.1b is around 0.52, corresponding to SNP rs1279795. The Fdr values are very large; hence the chance that all the SNPs are not associated with the phenotype is high.

Figure 4.1b shows that the ordering of SNPs based on all samples is different from the ordering of SNPs based on training samples in fold 1 of ECV. This indicates that the subsets of SNPs selected based on all samples are different from the subsets of SNPs re-selected based

Figure 4.1: The Fdrs based on all samples (left) and the Fdrs based on the training samples in fold 1 of ECV given the SNP ranking based on all samples (right).

(a) All samples **(b)** Fold 1 in ECV, re-ordered



on the training samples of each fold in ECV. Table 4.1 shows the top 10 SNPs selected based on all samples and selected based on the training samples in fold 1. It can be seen that 6 of the top 10 SNPs were selected commonly by the two methods. However, SNPs rs6648176 and rs6671507 have ranks of 3rd and 8th respectively based on all samples whereas they have ranks of 378th and 375th based on the training samples of fold 1. These two SNPs rank high when using all samples because these SNPs are highly correlated to the phenotype in test data in fold 1. On the other hand, SNPs rs17103033 and rs13237949 have ranks of 6th and 9th based on the training samples of fold 1 whereas they have ranks of 95th and 84th respectively based on all samples, but the correlations between these two SNPs and the phenotype in the training data in fold 1 are low. SNPs rs17103033 and rs13237949 may have high correlations with phenotype in the training data in fold 1 but have low correlations with the phenotype in the test data in fold 1.

Given a list of ordered SNPs, we selected a sequence of subsets that contains the top l SNPs and fitted the three regularized logistic regression models using training data. We set $l = 2^s$, for $s = 0, 1, ..., 12$. We only report the best model for elastic-net and hyper-LASSO.

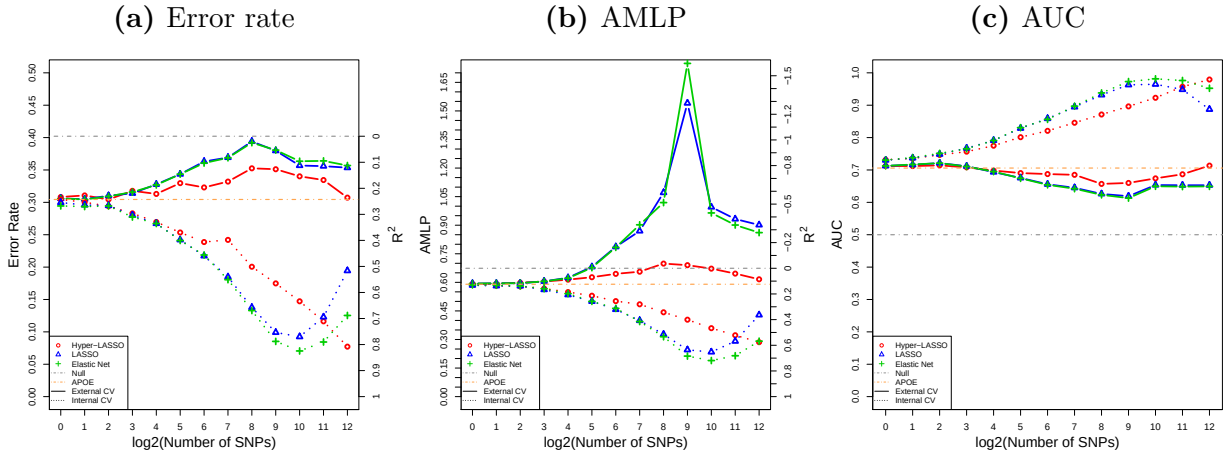
Table 4.1: Top 10 SNPs selected based on all samples and based on the training samples in fold 1 of ECV. The number in the bracket indicates the rank of SNP ordered using the other method to select SNPs.

SNP rank	Based on all samples	Based on fold 1 in ECV
1	rs1279795 (2)	rs8039031 (2)
2	rs8039031 (1)	rs1279795 (1)
3	rs6649176 (378)	rs7318037 (5)
4	rs1552820 (4)	rs1552820 (4)
5	rs7318037 (3)	rs17103033 (10)
6	rs9788079 (11)	rs10753514 (95)
7	rs5915434 (7)	rs5915434 (7)
8	rs6671507 (375)	rs10519111 (13)
9	rs4435421 (21)	rs13237949 (84)
10	rs17103033 (5)	rs1552828 (11)

The best α in elastic-net are 0.3 and 0.7 for ICV and ECV, respectively. The best degrees of freedom a in hyper-LASSO are 1.5 and 1 for ICV and ECV, respectively. These models can be used to quantify the predictivity of the top l SNPs selected subsets together with APOE $\varepsilon 4$ dosages predicting Alzheimer's disease status. We also fitted a simple logistic regression model with APOE $\varepsilon 4$ dosages as the only predictor. We fitted a logistic regression model with only APOE $\varepsilon 4$ dosages as the predictor. The baseline error rate of this dataset is 0.4, which is by randomly predicting that $y_i = 0$ for $i = 1, 2, ..., 2099$. Figure 4.2 shows the predictivity of the selected subsets of SNPs based on all samples versus the predictivity of the selected subsets of SNPs based the training samples in each fold of ECV. We can see that the model with only APOE $\varepsilon 4$ dosages can decrease the baseline error rate from 0.4 to 0.31, achieving an R^2_{ER} of 23%. When conducting ECV, the optimal predictivity of SNPs selected using the training samples is very close to the model with APOE $\varepsilon 4$ dosages as the only predictor; adding more SNPs into our model does not improve the predictive performance based on APOE $\varepsilon 4$ only. On the other hand, when training the model based on the subsets of SNPs pre-selected using all samples, the optimal model can reach an error rate of 0.07, attained by using a subset of 2^{10} SNPs. It can be seen that the error rate, AMLP, and AUC

in ICV and ECV are nearly identical when the size of the subset is smaller than 8 SNPs. This is because of overlapping SNPs between ICV and ECV as shown in Table 4.1 for the top 10 SNPs. However, the ICV's estimates of the predictivity of SNPs pre-selected based on all samples start to become significantly stronger than the predictivity of selected SNPs measured with ECV when the size of the subset is larger than 2 SNPs.

Figure 4.2: The plots of predictivity of selected SNPs averaged over 10-fold CV for the real dataset.



From Figure 4.2, it can also be seen that the predictive performance of hyper-LASSO is better than both LASSO and elastic-net in this dataset. When more SNPs are added into the model, hyper-LASSO is more stable in maintaining the performance. While the error rates of those three methods are similar, the AMLP of hyper-LASSO is much smaller than those of LASSO and elastic-net when the subset size of SNPs is large. The high AMLP of LASSO and elastic-net when the size of the subset is large results from a few very small $\hat{P}_i(y_i|x_i)$. Overall, LASSO performs a little better than elastic-net for this dataset.

4.2 Results on the Synthetic Datasets

We have repeated the predictive analysis method used for Alzheimer's disease on the two synthetic datasets described in Section 3.3. The purpose of these simulation studies is to understand the cause of feature selection bias of ICV with datasets in which we know the

true relationship between the response and the SNPs. The simulation studies also serve to demonstrate the effectiveness of the predictive analysis method that we used for analyzing Alzheimer's disease data.

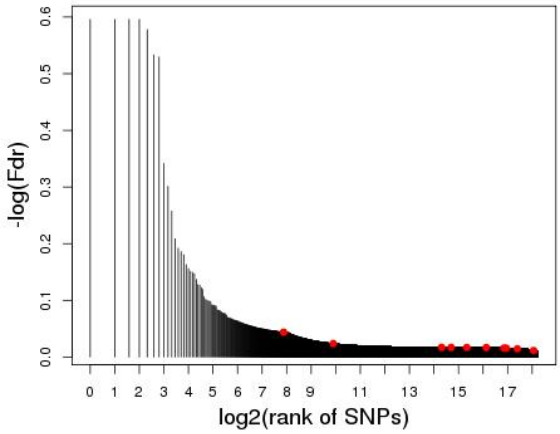
The synthetic datasets were generated by choosing ten SNPs from the real dataset and generating the coefficients for the chosen SNPs from normal distributions. The phenotype was generated with logistic regression models. Dataset1 has weak signals with coefficients generated from $N(0, 0.1^2)$. Dataset2 has strong signals with coefficients generated from $N(0, 2^2)$. The coefficients of the rest SNPs were set to be 0. The way to split the synthetic datasets into ten folds was the same as the real dataset, which means for any SNPs data of a participant in fold k in the real dataset, the SNPs data of the participant is in fold k in the synthetic dataset.

We replicated the method that was used for the real data. We first tested the statistical significance of each SNP conditional on APOE $\varepsilon 4$ by applying the LRT to compute a p-value. Then we converted the p-values into the tail-based false discovery rate. Figure 4.3 shows the ordered Fdrs using all samples and using the training samples in fold 1 of ECV for the two synthetic datasets. Figures 4.3a and 4.3b show the ordered $-\log_{10}(\text{Fdr})$ calculated based on all samples for dataset1 and dataset2, respectively. Figures 4.3c and 4.3d show the ordered $-\log_{10}(\text{Fdr})$ calculated based on the training samples in fold 1 of ECV for dataset1 and dataset2. It can be seen that when the signals in the dataset are weak (dataset1), the Fdrs are very large. The rankings of the true signals (SNPs truly related to the response) are very low based on all samples or the training samples in fold 1 of ECV. In contrast, when the signals in the dataset are strong (dataset2), the Fdrs are very small, as shown in Figures 4.3b and 4.3d. The true signals are ranked very highly.

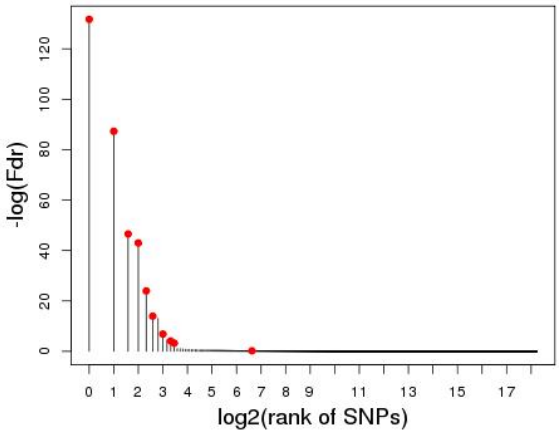
To highlight the difference in the feature selection based on all samples and based only on the training samples in fold 1 of ECV, we re-ordered SNPs by the ordering of SNPs based on all samples. Figure 4.3e and 4.3f show the Fdrs of these re-ordered SNPs given the SNP ranking based on all samples. Clearly, the ordering of SNPs based on all samples is different from the ordering of SNPs selected with the training samples in fold 1 of ECV, especially for weak signals. In dataset1, we can see that SNPs selected using the two methods have 5 overlapping SNPs among the top 10 SNPs from Table 4.2. However, the ranks of these

Figure 4.3: The Fdrs based on all samples and the Fdrs of fold 1 in ECV given the SNP ranking based on all samples for dataset1 (left) and dataset2 (right). The red dots indicate true signals.

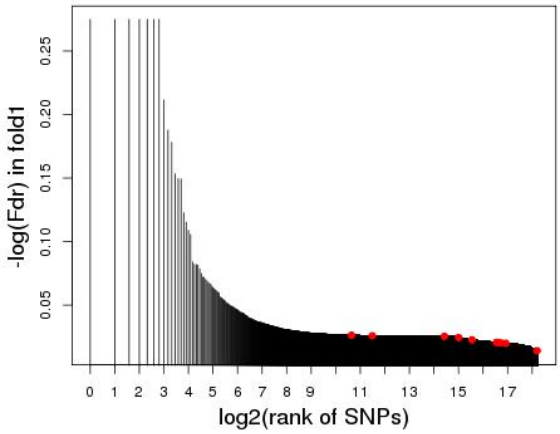
(a) Dataset1, all samples



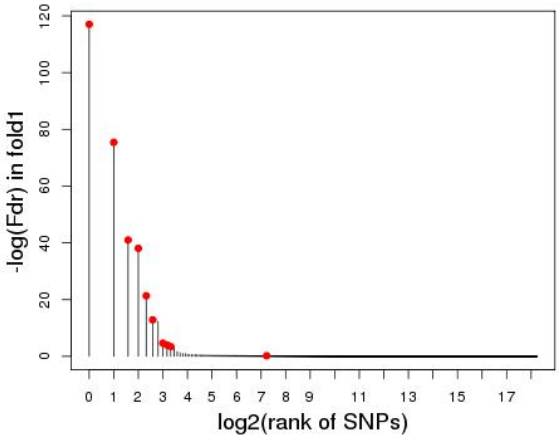
(b) Dataset2, all samples



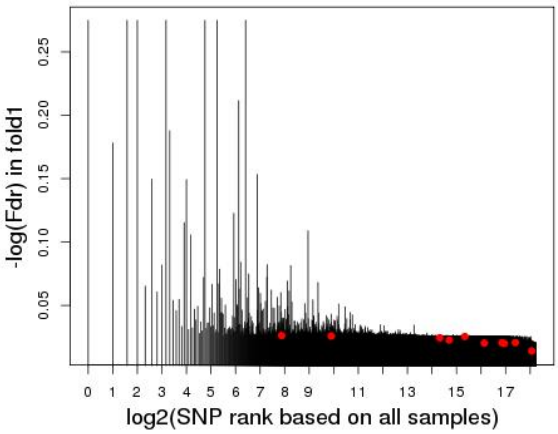
(c) Dataset1, fold 1 in ECV



(d) Dataset2, fold 1 in ECV



(e) Dataset1, fold 1 in ECV, re-ordered



(f) Dataset2, fold 1 in ECV, re-ordered

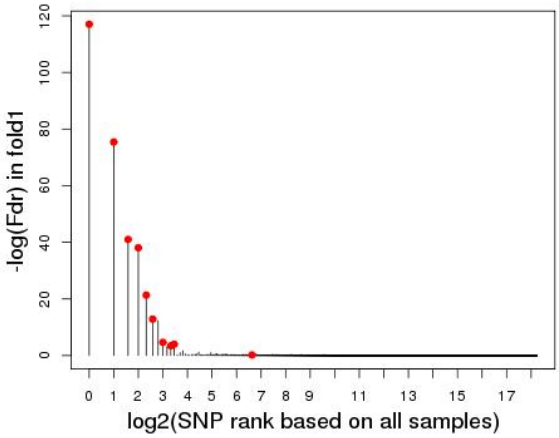


Table 4.2: Top 10 SNPs selected based on all samples and based on the training samples in fold 1 of ECV for dataset1 and dataset2. SNPs with * represent the true signals. The number in the bracket indicates the rank of the SNP ordered using the other method to select SNPs.

SNP rank	Dataset1		Dataset2	
	Based on all samples	Based on fold 1	Based on all samples	Based on fold 1
1	rs1808380 (7)	rs13190617 (4)	rs405509* (1)	rs405509* (1)
2	rs2280201 (10)	rs1321981 (3)	rs8106922* (2)	rs8106922* (2)
3	rs1321981 (2)	rs1010196 (27)	rs157580* (3)	rs157580* (3)
4	rs13190617 (1)	rs10519980 (85)	rs439401* (4)	rs439401* (4)
5	rs11664142 (31)	rs10008892 (38)	rs2075650* (5)	rs2075650* (5)
6	rs2425483 (12)	rs2255994 (9)	rs10402271* (6)	rs10402271* (6)
7	rs2894111 (36)	rs1808380 (1)	rs460527 (7)	rs460527 (7)
8	rs2868574 (20)	rs11084445 (70)	rs8039031* (8)	rs8039031* (8)
9	rs2255994 (6)	rs1432679 (10)	rs2597504 (11)	rs7318037* (11)
10	rs1432679 (9)	rs2280201 (2)	rs1420566* (10)	rs1420566* (10)

overlapped SNPs differ. For example, rs1808380 ranks first based on all samples but it ranks 7th based on the training samples in fold 1. Moreover, SNPs rs11664142 ranks 5th based on all samples but ranks 31st based on the training samples in fold 1 and rs2894111 ranks 7th based on all samples but ranks 36th based on the training samples in fold 1. Those two SNPs have high ranks when selecting SNPs using all samples but low rank when re-selecting SNPs using the training samples of each fold in ECV because they are only weakly associated with the test data. In dataset2, it is noteworthy that the ordered lists of the top 10 SNPs based on all samples and training samples in fold 1 are very similar. The overlapping is at 90% , with the exception of the SNP ranked 9th for both two methods (rs2597504 and rs7318037). The ranks of the true signals are very close to the top SNPs selected by using the two methods in dataset2. On the other hand, we can still see from Figure 4.3f that as the ordered list gets longer, the ranking of SNPs based on all samples becomes less similar to the ranking of SNPs based on the training samples in fold 1. This is because more SNPs correlated to the phenotype in test data will be included in the ordered list by ICV. The models built using

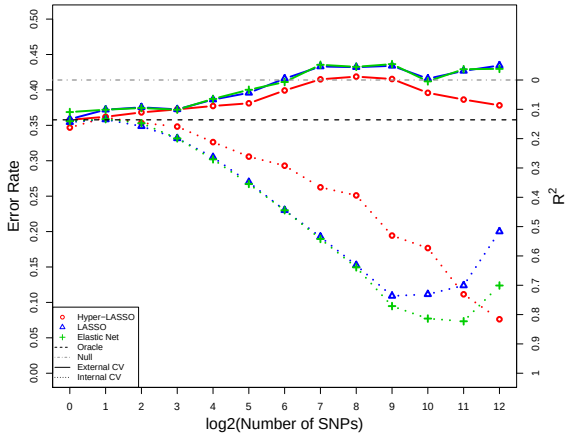
those SNPs will then underestimate the true error rate in test data.

Given a list of ordered SNPs, we selected a sequence of subsets containing the top l SNPs and fitted the three regularized logistic regression models using training data, which is similar to the procedure of the real data analysis. The training data was used to train the three penalized logistic regression models and test data was used to assess the predictive performance of each model. In ICV, we pre-selected a list of SNPs using all samples and then applied 10-fold CV to this list of SNPs to measure their predictivity. In ECV, the list of selected SNPs was produced using only the training samples in each fold of CV. Hence, different lists of SNPs were used in each training process. The α with the best performance is that of elastic-net which was 0.7 for both dataset1 and dataset2. The degree of freedom a with the best performance was that of hyper-LASSO which was 1 and 1.5 for dataset1 and dataset2, respectively. In order to better interpret the predictive performance, we also measured the predictive performance of two special prediction cases. One is called the null case, in which no SNP is used as a predictor in the model. The other is called the oracle case, in which we fitted a logistic regression using the ten truly related SNPs and APOE $\varepsilon 4$ dosages.

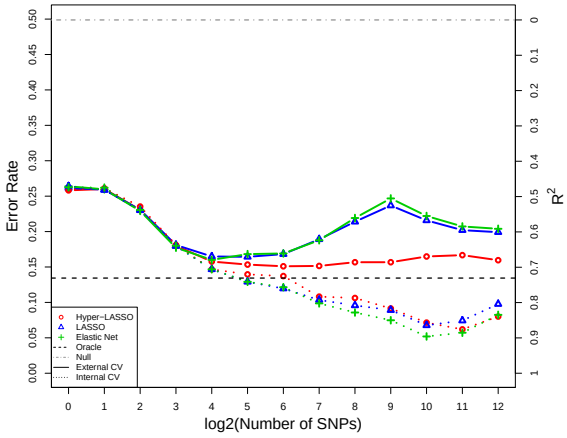
The predictivity against k SNPs on the top list is showed in Figure 4.4. We first examined dataset1, in which the signals are very small. From Figure 4.3, we can see that there are only two true signals among the top 2^{12} SNPs selected based on all samples or only the training samples in fold 1 and their rankings are very low. All the rest of the top 2^{12} SNPs are not related to the response. In this scenario, we expect that selecting more SNPs and adding them into models does not improve the performance of predicting the response. The error rate of the oracle case is 0.357, corresponding to an R^2_{ER} of 13%. Using the ECV method, the predictive performances of the three models are all worse than the oracle case. The smallest error rate is 0.357, which is the same as the oracle case, and this was achieved by hyper-LASSO with a subset of 1 SNP. What's more, the predictivity of SNPs selected based on all samples is too high. The error rate and AMLP shown in Figures 4.4a and 4.4c are below the line corresponding to the oracle case even when a small number of top SNPs were used to make prediction. The best R^2_{ER} was attained by pre-selected SNPs based on all samples is 0.80 using the LASSO with 2^{11} SNPs. The AUC in Figure 4.4e even reached

Figure 4.4: The plots of predictivity of selected SNPs averaged over 10-fold CV for synthetic datasets.

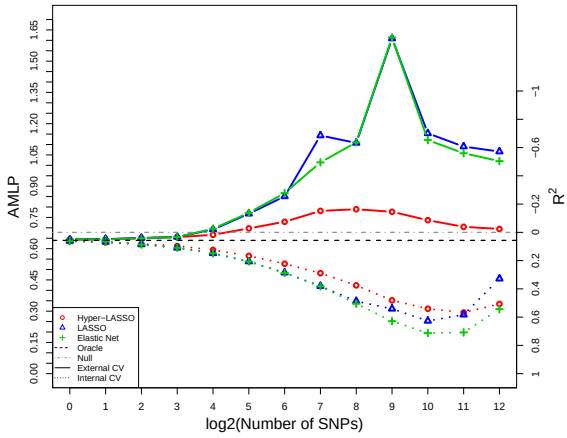
(a) Error Rate for Dataset1



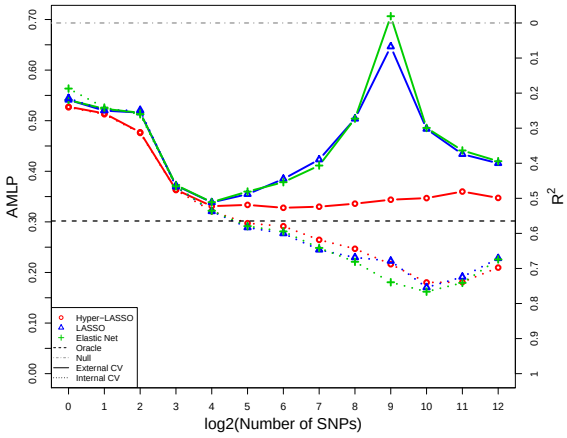
(b) Error Rate for Dataset2



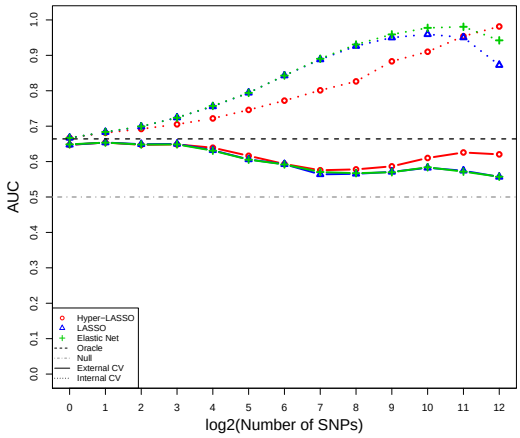
(c) AMLP for Dataset1



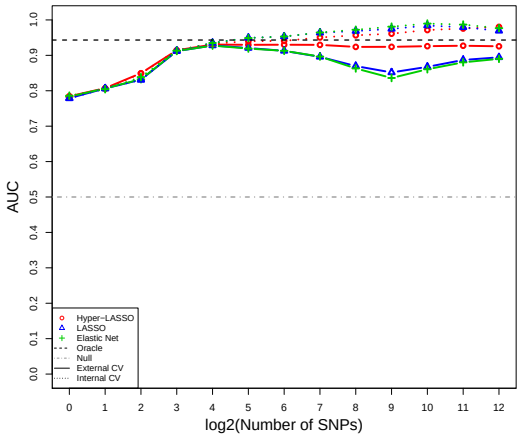
(d) AMLP for Dataset2



(e) AUC for Dataset1



(f) AUC for Dataset2



0.98 for hyper-LASSO with 2^{12} SNPs. These results show that the bias is severe in the ICV predictivity estimates in dataset1. The results of dataset1 are very similar to the result of the real data.

Now we turn to look at the comparison of ICV and ECV in dataset 2, in which strong signals exist. The error rate of the oracle case is 0.137. It can be seen from Figures 4.4b, 4.4d and 4.4f that the true signals can improve the predictive performance of models. As the subset size increases, more true signals are included in the models, the error rates are decreasing for both SNPs selected using ICV method and ECV method. When using ECV method, the lowest error rate of 0.15 is achieved by the hyper-LASSO with a subset of 2^7 SNPs, in which almost all ten true signals are selected using the training data of each fold. The predictive performance of 10-fold cross validation may still have some bias. All the three models achieved an R^2_{ER} of approximately 70% and an R^2_{AMLP} of 52% based on 2^4 SNPs. The largest AUC among the three models was 0.93, attained when the subset contained 2^5 SNPs, which is very close to the oracle case AUC of 0.94. When using ICV method, the predictivity of SNPs selected based on all samples becomes even stronger than the predictivity of the truly related SNPs (the oracle case). The smallest error rate is 0.05 when using a subset of 2^{10} SNPs. Although the biases in ICV's predictivity estimates are less severe than those in dataset1, the highest R^2_{ER} is 90%, which is significantly higher than the R^2_{ER} of the oracle prediction. Because the ordered lists of first 2^4 SNPs using ICV and ECV are almost the same, it is not surprising that the predictivity estimates given by ICV and ECV are similar when the subset size $l \leq 2^4$.

Comparing the performance of the three penalized logistic regression methods, hyper-LASSO outperforms LASSO and elastic-net for both dataset1 and dataset2. It can be seen that hyper-LASSO is more stable than LASSO and elastic-net when the size of the subset of SNPs increases. The predictive performances of the three models are similar when the subset of predictors is small. However, when the size of the subset increases, hyper-LASSO tends to select fewer features and retain the large coefficients of important features. On the other hand, LASSO and elastic-net tend to select many non-zero coefficients. Figure 4.4d also shows that the AMLP of hyper-LASSO is lower than both LASSO and elastic-net at every number of l. When the error rate is the same, the predictive probability of hyper-LASSO

at the true label is higher than LASSO and elastic-net. The performances of LASSO and elastic-net do not differ by much.

5. Conclusion and Future Work

In this thesis, we conducted empirical studies using a real dataset and two synthetic datasets to investigate the feature selection bias caused by using ICV and the predictivity of selected SNPs from GWAS of Alzheimer's disease using three penalized logistic regression methods. This thesis reinforces that there might be huge bias in the predictivity estimate given by ICV, especially when there is no strong signal in the dataset. For the real dataset, identified SNPs by GWAS using ECV do not help with predicting the Alzheimer's disease status except APOE $\varepsilon 4$, which has been known by scientist to increase the risk of Alzheimer's disease. When using ICV, the predictivity estimate of pre-selected SNPs based on all samples can reach an R^2 of 80%. For the synthetic datasets, the result in the dataset where only weak signals exist is similar to the real data application. Moreover, the predictivity estimate of pre-selected SNPs is even better than the predictivity of the chosen ten SNPs that used to generate the phenotype. On the other hand, when there are strong signals in the dataset, the selected top SNPs using ECV can improve the predictive performance of the models. We found that hyper-LASSO has better performance than LASSO and elastic net. As more noises added to the model, hyper-LASSO is more stable to maintain the good performance than LASSO and elastic net. In a nutshell, ICV should not be used to measure the predictivity of selected SNPs and should be stated clearly.

Although single-SNP analysis works well in identifying large signals, it fails to detect a group of correlated SNPs. Different SNPs may interact with other SNPs that form a complex network. Single-SNP analysis is not able to distinguish SNPs interactions. Some Bayesian methods using MCMC algorithm have been developed to select a group of correlated SNPs [31]. In the future we can apply those methods to select a group of correlated SNPs. Another future work is to apply survival analysis with penalty to build the predictive model of Alzheimer's disease. Fail to detect Alzheimer's disease at a specific time does not mean we will not get Alzheimer's disease in the future. The dataset with specific event time should be used to construct the survival analysis.

References

[1] Minerva M. Carrasquillo, Fanggeng Zou, V. Shane Pankratz, Samantha L. Wilcox, Li Ma, Louise P. Walker, Samuel G. Younkin, Curtis S. Younkin, Linda H. Younkin, Gina D. Bisceglio, et al. Genetic variation in pcdh11x is associated with susceptibility to late-onset alzheimer's disease. *Nature genetics*, 41(2):192–198, 2009.

[2] Allen D. Roses, Michael William Lutz, Heather Amrine-Madsen, Ann M. Saunders, Donna Crenshaw, Scott S. Sundseth, Matt Huentelman, Kathleen Anne Welsh-Bohmer, and Eric Reiman. A tomm40 variable-length polymorphism predicts the age of late-onset alzheimer's disease. *The pharmacogenomics journal*, 10(5):375, 2010.

[3] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[4] Lynn M. Bekris, Franziska Lutz, and Chang-En Yu. Functional analysis of apoe locus genetic variation implicates regional enhancers in the regulation of both tomm40 and apoe. *Journal of human genetics*, 57(1):18, 2012.

[5] Adeline Lo, Herman Chernoff, Tian Zheng, and Shaw-Hwa Lo. Why significant variables arent automatically good predictors. *Proceedings of the National Academy of Sciences of the United States of America*, 112(45):13892–13897, November 2015.

[6] Klas Gränsbo, Peter Almgren, Marketa Sjögren, JG Smith, Gunnar Engström, Bo Hedblad, and Olle Melander. Chromosome 9p21 genetic variation explains 13% of cardiovascular disease incidence but does not improve risk prediction. *Journal of internal medicine*, 274(3):233–240, 2013.

[7] David G. Clayton. Prediction and Interaction in Complex Disease Genetics: Experience in Type 1 Diabetes. *PLoS Genetics*, 5(7):e1000540, July 2009.

[8] Johanna Jakobsdottir, Michael B. Gorin, Yvette P. Conley, Robert E. Ferrell, and Daniel E. Weeks. Interpretation of genetic association studies: markers with replicated highly significant odds ratios may be poor classifiers. *PLoS genetics*, 5(2):e1000337, 2009.

[9] A. Cecile J.W. Janssens and Cornelia M. van Duijn. Genome-based prediction of common diseases: advances and prospects. *Human molecular genetics*, 17(R2):R166–R173, 2008.

[10] Sanghee Kang, Bo Ram Kim, Myoung-Hee Kang, Dae-Young Kim, Dae-Hee Lee, Sang Cheul Oh, Byung Wook Min, and Jun Won Um. Anti-metastatic effect of metformin via repression of interleukin 6-induced epithelial–mesenchymal transition in human colon cancer cells. *PloS one*, 13(10):e0205449, 2018.

[11] Michael Lecocke and Kenneth Hess. An Empirical Study of Univariate and Genetic Algorithm-Based Feature Selection in Binary Classification with Microarray Data. *Cancer Informatics*, 2:313–327, February 2007.

[12] Charles Kooperberg, Michael LeBlanc, and Valerie Obenchain. Risk prediction using genome-wide association studies. *Genetic Epidemiology*, 34(7):643–652, 2010.

[13] Christophe Ambroise and Geoffrey J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences*, 99(10):6562–6566, 2002.

[14] Jerzy Krawczuk and Tomasz ukaszuk. The feature selection bias problem in relation to high-dimensional gene data. *Artificial Intelligence in Medicine*, 66:63–71, January 2016.

[15] Surendra K. Singhi and Huan Liu. Feature subset selection bias for classification learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 849–856. ACM, 2006.

[16] Jaime Derringer, Robert F. Krueger, Danielle M. Dick, Scott Saccone, Richard A. Grucza, Arpana Agrawal, Peng Lin, Laura Almasy, Howard J. Edenberg, Tatiana Foroud, John I. Nurnberger, Victor M. Hesselbrock, John R. Kramer, Samuel Kuperman, Bernice Porjesz, Marc A. Schuckit, Laura J. Bierut, and Gene Environment Association Studies (GENEVA) Consortium. Predicting Sensation Seeking From Dopamine Genes: A Candidate-System Approach. *Psychological Science*, 21(9):1282–1290, September 2010.

[17] Natalia Briones and Valentin Dinu. Data mining of high density genomic variant data for prediction of Alzheimer's disease risk. *BMC Medical Genetics*, 13:7, January 2012.

[18] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[19] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, April 2005.

[20] Longhai Li and Weixin Yao. Fully Bayesian Logistic Regression with Hyper-Lasso Priors for High-dimensional Feature Selection. *Journal of Statistical Computation and Simulation*, 88(14):2827–2851, September 2018.

[21] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[22] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[23] Jungsu Kim, Jacob M. Basak, and David M. Holtzman. The Role of Apolipoprotein E in Alzheimers Disease. *Neuron*, 63(3):287–303, August 2009.

[24] Samuel S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, 1938.

[25] John D. Storey. A direct approach to false discovery rates. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):479–498, August 2002.

[26] John D. Storey. The positive false discovery rate: a bayesian interpretation and the q-value. *The Annals of Statistics*, 31(6):2013–2035, 2003.

[27] Korbinian Strimmer. fdrtool: a versatile r package for estimating local and tail area-based false discovery rates. *Bioinformatics*, 24(12):1461–1462, 2008.

[28] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of statistical software*, 33(1):1–22, 2010.

[29] Sudha Seshadri, Annette L. Fitzpatrick, M. Arfan Ikram, Anita L. DeStefano, Vilmundur Gudnason, Merce Boada, Joshua C. Bis, Albert V. Smith, Minerva M. Carrasquillo, Jean Charles Lambert, et al. Genome-wide analysis of genetic loci associated with alzheimer disease. *Jama*, 303(18):1832–1840, 2010.

[30] Denise Harold, Richard Abraham, Paul Hollingworth, Rebecca Sims, Amy Gerrish, Marian L. Hamshere, Jaspreet Singh Pahwa, Valentina Moskvina, Kimberley Dowzell, Amy Williams, et al. Genome-wide association study identifies variants at clu and picalm associated with alzheimer's disease. *Nature genetics*, 41(10):1088–1093, 2009.

[31] Yongtao Guan and Matthew Stephens. Bayesian variable selection regression for genome-wide association studies and other large-scale problems. *The Annals of Applied Statistics*, 5(3):1780–1815, September 2011.

Appendix

R Code

A.1 Utility Functions

```
## General functions
log_sum_exp <- function (lx) {mlx <- max (lx); log(sum(exp(lx - mlx))) + mlx}
log_sum_exp_row <- function (mlx) {apply (mlx, 1, log_sum_exp)}
log_sum_exp_col <- function (mlx) {apply (mlx, 2, log_sum_exp)}

## find mode
Mode <- function(x,na.rm) {
  xtab <- table(x)
  xmode <- names(which(xtab == max(xtab)))
  return(xmode[1])
}

## likelihood ratio test
LRtest <- function(model_null,model_full) {
  loglike_null <- logLik(model_null)
  loglike_full <- logLik(model_full)

  df0 <- attr(loglike_null, "df")
  df1 <- attr(loglike_full, "df")
  df <- df1 - df0
  SS <- as.numeric(2*(loglike_full - loglike_null))
  list (SS = SS, df=df, pvalue = pchisq (SS, df, lower.tail = FALSE, log.p = F),
        log.pvalue=pchisq (SS, df, lower.tail = FALSE, log.p = T))
}

checkna_col <- function (x) sum(is.na(x) | x=="-9")

## check na
checkna <- function(imputedata2) {
  num.na <- apply(imputedata2,2, checkna_col)
  na.names <- colnames (imputedata2)[which (num.na >0)]

  if (length (na.names)>0){
    cat ("Some Variables Contain NA\n")
  } else {
    cat("Your Data Has NO NA\n")
  }
  na.names
}

library(glmnet)
glmnet_fit <- function(X_tr, Y_tr, X_ts, Y_ts, a){
  ## read information about data
  n <- nrow (X_tr) ## numbers of obs
```

```
  ## find number of observations in each group
  nos_g <- as.vector(tapply(rep(1,n), INDEX = Y_tr, sum))

  if (any(nos_g < 2)) stop ("Less than 2 cases in some group")

  ## choosing the best lambda
  cvfit <- cv.glmnet(x = X_tr, y = Y_tr, alpha = a, nlambda = 500, family = "binomial",
                     type.measure = "class")
  lambda <- cvfit$lambda[which.min(cvfit$cvm)]
  cat("The best lambda chosen by CV:", lambda, "\n")
  ## fit model with the best lambda
  fit <- glmnet (x = X_tr, y= Y_tr, alpha = a, nlambda = 500, family = "binomial")

  betas <- coef(fit, s = lambda)

  ## predicting for new cases
  if (is.null (X_ts)) {
    return (betas)
  }
  else {

    probs_pred_1 <- predict(fit, newx = X_ts, s =lambda, type="response")
    probs_pred_0 <- 1- probs_pred_1
    pred_matrix <- cbind(probs_pred_0, probs_pred_1)
    eval <- evaluate_pred(pred_matrix, Y_ts+1, showplot = F)
    list(eval = eval, predictor = class_pred, betas = betas)
  }
}
```

A.2 R Code for Feature Selection and Model Fitting using ICV with real Dataset

```
############### calculating P-value conditional on APOE for chromosome 1 to 22 ###############
source("/home/med826/Mayo/utility.r")
if (!exists("irep"))  irep <- 1
## file path
SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", irep)
cov_file <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"
pvalue_file <- "/home/med826/Mayo/ICV/pvalue/"

# get the single SNP
SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
SNPname <- substr(colnames(SNP_single_rawdata)[7], 1, nchar(colnames(SNP_single_rawdata)[7])-2)
SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]

# phenotype
covdata <- readRDS(cov_file)

## imputation with mode
n <- ncol(SNP_single)
SNP_single_imputeddata <- SNP_single[, n]
```

```
SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)

ssnpdata <- cbind.data.frame(pheno=covdata$Dx, apoe=covdata$APOE4_dosage..0.1.2.,
                             SNP = SNP_single_imputeddata)

Y_name <- "pheno"
X_cov <- "apoe"
X_name <- "SNP"

## calculate p value for Internal CV
if (nlevels(ssnpdata$SNP) < 2) {
  saveRDS(c(SNPname = SNPname, pvalue = 0), file = paste0(pvalue_file, sprintf("pvalue%d.rds", irep)))
} else {
  glm_null <- glm(formula = paste0(Y_name, "~", X_cov), data = ssnpdata, family = "binomial")
  glm_full <- glm(formula = paste0(Y_name, "~", X_cov, "+", X_name), data = ssnpdata,
                  family = "binomial")

  lrtest <- LRtest(glm_null, glm_full)
  pvalue <- lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue =  pvalue), file = paste0(pvalue_file,
                                                                sprintf("pvalue%d.rds", irep)))
}

################ calculate p-value conditional on APOE for X chromosome ################
source("/home/med826/Mayo/utility.r")

if (!exists("irep"))  irep <- 1
irep <- irep+300768

## file path
SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", irep)
cov_file <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"
pvalue_file <- "/home/med826/Mayo/ICV/pvalue/"

# get the single SNP
SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
SNPname <- substr(colnames(SNP_single_rawdata)[7], 1, nchar(colnames(SNP_single_rawdata)[7])-2)
SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]

# phenotype
covdata <- readRDS(cov_file)
covaritedata <- covdata[order(covdata$IID),]

## imputation with mode
n <- ncol(SNP_single_rawdata)

snp_male <- SNP_single[which(SNP_single$SEX==1),7]
snp_male_imputed <- snp_male
snp_male_imputed[is.na(snp_male)] <- Mode(snp_male,na.rm = T)
snp_male_imputed[snp_male_imputed==0] <- 11
snp_male_imputed[snp_male_imputed==2] <- 12

snp_female <- SNP_single[which(SNP_single$SEX==2),7]
snp_female_imputed <- snp_female
```

```
snp_female_imputed[is.na(snp_female)] <- Mode(snp_female,na.rm = T)
snp_female_imputed[snp_female_imputed==0] <- 21
snp_female_imputed[snp_female_imputed==1] <- 22
snp_female_imputed[snp_female_imputed==2] <- 23

SNP_single[SNP_single$SEX==1,7] <- snp_male_imputed
SNP_single[SNP_single$SEX==2,7] <- snp_female_imputed

ssnpdata <- cbind.data.frame(pheno=covdata$Dx, sex=covdata$Sex, apoe=covdata$APOE4_dosage..0.1.2.,
                             SNP = factor(SNP_single[,7]))

Y_name <- "pheno"
X_cov <- "apoe"
X_name <- "SNP"

## calculate p value for Internal CV
if (nlevels(ssnpdata$SNP) < 2) {
  saveRDS(c(SNPname = SNPname, pvalue = 0), file = paste0(pvalue_file,
                                                          sprintf("pvalue%d.rds", irep)))
} else {
  glm_null <- glm(formula = paste0(Y_name, "~", X_cov), data = ssnpdata, family = "binomial")
  glm_full <- glm(formula = paste0(Y_name, "~", X_cov, "+", X_name), data = ssnpdata,
                                    family = "binomial")

  lrtest <- LRtest(glm_null, glm_full)
  pvalue <- lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue =  pvalue), file = paste0(pvalue_file,
                                                                sprintf("pvalue%d.rds", irep)))
}

################ convert p-values into Fdr and select top features ################
library("fdrtool")

Pvalue_path <- "/home/med826/Mayo/ICV/pvalue"
datapath <- "/home/med826/Mayo/ICV/dataset/"
evalpath <- "/home/med826/Mayo/ICV/evaluation/"

Pvalue_list <- list.files(path = Pvalue_path, full.names = T)

Pvalue_all <- do.call('rbind', lapply(Pvalue_list, readRDS))
pvalue <- as.numeric(Pvalue_all[,2])
ordered.name <- Pvalue_all[order(pvalue, decreasing = F),1]

## plot of all qvalues
fdr <- fdrtool(pvalue, statistic = "pvalue", plot = F, cutoff.method = "fndr")
#lfdr in fdrtool is the local fdr
#qval in fdrtool is the tail area-based fdr
qvalue <- fdr$qval
mlog.qvalue <- -log (qvalue, base = 10)
mll.sorted <- sort(mlog.qvalue, decreasing = T)
xaxis <- log2(1:309549)
jpeg(paste0(evalpath, "fdr_ICV.jpg"), width = 480, height = 480)
par(mar = c(4,4,0.5,0.5))
plot(xaxis, mll.sorted, type = "h", xlab="", ylab="", xaxp=c(0,19,19))
```

```
#plot(mlog.qvalue, type = "h", xlab="", ylab="")
title(xlab = "log2(rank of SNPs)", ylab = "-log(Fdr)", cex.lab = 1.5, line = 2.5)
dev.off()

top5000 <- cbind(snpname = ordered.name[1:5000], qvalue = sort(qvalue, decreasing = F)[1:5000])
saveRDS(top5000, file = paste0(datapath, "top5000SNPs_ICV.rds"))

############### Select top 5000 features ###############
source("/home/med826/Mayo/utility.r")

datapath <- "/home/med826/Mayo/ICV/dataset/"
top5000_file <- paste0(datapath, "top5000SNPs_ICV.rds")
covdata_file <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"
allsnplist <- "/home/med826/Mayo/SNPdata/alldata_snplist.rds"

top5000SNP <- readRDS(top5000_file)
## infodata contain fold, phenotype, apoe
covdata <- readRDS(covdata_file)

## extract the snp names
names_snp <- top5000SNP[,1]
snpname_list <- as.character(readRDS(allsnplist))

## NULL matrix
SNP <- matrix(0, 2099, 5000)
colnames(SNP) <- names_snp
for (i in 1:5000){
  index_snp <- which(snpname_list %in% names_snp[i])
  SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", index_snp)
  SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
  SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]
  n <- ncol(SNP_single_rawdata)
  if (index_snp > 300767){
    snp_male <- SNP_single[which(SNP_single$SEX==1),n]
    snp_male_imputed <- snp_male
    snp_male_imputed[is.na(snp_male)] <- Mode(snp_male,na.rm = T)
    snp_male_imputed[snp_male_imputed==0] <- 11
    snp_male_imputed[snp_male_imputed==2] <- 12

    snp_female <- SNP_single[which(SNP_single$SEX==2),7]
    snp_female_imputed <- snp_female
    snp_female_imputed[is.na(snp_female)] <- Mode(snp_female,na.rm = T)
    snp_female_imputed[snp_female_imputed==0] <- 21
    snp_female_imputed[snp_female_imputed==1] <- 22
    snp_female_imputed[snp_female_imputed==2] <- 23

    SNP_single[SNP_single$SEX==1,n] <- snp_male_imputed
    SNP_single[SNP_single$SEX==2,n] <- snp_female_imputed
    SNP[,i] <- SNP_single[,n]
  } else{
    SNP_single_imputeddata <- SNP_single[, n]
    SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)
    SNP[,i] <- SNP_single_imputeddata
  }
```

```
}

SNPdata <- cbind.data.frame(SNP, apoe=covdata$APOE4_dosage..0.1.2.,                                phenotyp
saveRDS(SNPdata, file = paste0(datapath, "SNPdata5000_ICV.rds"))

################ Model fitting ###############
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

datapath <- "/home/med826/Mayo/ICV/dataset/"
errorpath <- "/home/med826/Mayo/ICV/errorrate/"
evalpath <- "/home/med826/Mayo/ICV/evaluation/"
predpath <- "/home/med826/Mayo/ICV/prediction/"
amlppath <- "/home/med826/Mayo/ICV/amlp/"
predmatpath <- "/home/med826/Mayo/ICV/predmat/"

top5000_file <- paste0(datapath, "top5000SNPs_ICV.rds")
SNPdata_file <- paste0(datapath, "SNPdata5000_ICV.rds")

top5000SNP <- readRDS(top5000_file)
## extract the snp names
names_snp <- top5000SNP[,1]

## get the SNP file
SNPdata <- readRDS(SNPdata_file)

training <- SNPdata[SNPdata$fold != iloc,]
testing <- SNPdata[SNPdata$fold == iloc,]

## nsnp_set <- c(1, 2, 4, 8, ..., 1024, 2048, 4096)
nsnp_set <- c(0, 13)
for ( i in 1:13){
  nsnp_set[i] <- 2^(i-1)
}

Y_name <- "phenotype"
X_cov <- "apoe"
X_chosen <- names_snp[1:nsnp_set[irep]]

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_chosen), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training)[, -1]
Y_tr <- as.numeric(training[, Y_name])-1
X_ts <- model.matrix(fit_formula, data = testing)[, -1]
Y_ts <- as.numeric(testing[, Y_name])-1

## htlr
htlr.pred <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.htlr <- matrix(0, nrow=length(Y_ts),ncol=3)
er.htlr <- vector()
amlp.htlr <- vector()
alpha1 <- c(0.5, 1, 1.5)
```

```
for (j in 1:3){
  htlr.fit <- htlr_fit (
    y_tr = Y_tr, X_tr = X_tr, X_ts = X_ts, stdzx = F, ## data
    pty = "t", alpha = alpha1[j], s = -10,   ## alpha = df and s= log (w)
    iters_h = 1000, iters_rmc = 1000, thin = 10, ## mcmc iteration settings,
    leap_L_h = 5, leap_L = 50, leap_step = 0.3, hmc_sgmcut = 0.3, ## hmc settings
    initial_state = "lasso", silence = !interactive())  ## initial state settings
  htlr.pred[,2*j-1] <- htlr.fit$probs_pred[,1]
  htlr.pred[,2*j] <- htlr.fit$probs_pred[,2]
  htlr.predeval <- evaluate_pred(htlr.fit$probs_pred, Y_ts+1, showplot=F)
  er.htlr[j] <- htlr.predeval$er
  pred.htlr[,j] <- ifelse(htlr.pred[,1] > 0.5, 0, 1)
  amlp.htlr[j] <- htlr.predeval$amlp
}

## lasso prediction
lasso.fit <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, 1)
las.pred <- lasso.fit$eval[["table_eval"]]
predmat.las <- las.pred[, 3:4]
er.las <- lasso.fit$eval[["er"]]
pred.las <- as.numeric(lasso.fit$predictor)
amlp.las <- lasso.fit$eval[["amlp"]]

## elastic net
alpha2 <- c(0.3,0.5,0.7)
predmat.ela <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.ela <- matrix(0, nrow=length(Y_ts),ncol=3)
er.ela <- vector()
amlp.ela <- vector()
for (i in 1:3){
  elastic.net <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, alpha2[i])
  ela.pred <- elastic.net$eval[["table_eval"]]
  predmat.ela[,2*i-1] <- unlist(ela.pred[, 3])
  predmat.ela[,2*i] <- unlist(ela.pred[, 4])
  er.ela[i] <- elastic.net$eval[["er"]]
  pred.ela[,i] <- as.numeric(elastic.net$predictor)
  amlp.ela[i] <- elastic.net$eval[["amlp"]]
}

saveRDS(cbind(pred.htlr, pred.las, pred.ela),
        file = paste0(predpath, sprintf("predictor_ICV_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(er.htlr), er.las,  t(er.ela)),
        file = paste0(errorpath, sprintf("errorrate_ICV_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(amlp.htlr), amlp.las, t(amlp.ela)),
        file = paste0(amlppath, sprintf("amlp_ICV_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(htlr.pred, predmat.las, predmat.ela),
        file = paste0(predmatpath, sprintf("predmat_ICV_fold%d_nsnp%d.rds",iloc, nsnp_set[irep])))
```

A.3 R Code for Feature Selection and Model Fitting using ECV with Real Dataset

```
############### calculating P-value conditional on APOE for chromosome 1 to 22 ###############
```

```
source("/home/med826/Mayo/utility.r")

if (!exists("irep"))  irep <- 1
# file path
SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", irep)
covariate_file <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"

# get the single SNP
SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
SNPname <- substr(colnames(SNP_single_rawdata)[7], 1, nchar(colnames(SNP_single_rawdata)[7])-2)
SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]

# imputation with mode
n <- ncol(SNP_single)
SNP_single_imputeddata <- SNP_single[, n]
SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)

# load covariate file
covariate <- readRDS(covariate_file)
SNPdata <- cbind.data.frame(fold = covariate$tenfolds, pheno = covariate$Dx,
                            apoe = covariate$APOE4_dosage..0.1.2., SNP = SNP_single_imputeddata)

Y_name <- "pheno"
X_cov <- "apoe"
X_name <- "SNP"

kfolds=10
for (ifold in 1:kfolds) {
  pvalue_rds_10folds <- sprintf("/home/med826/Mayo/ECV/pvalue_apoe/pvalue_fold%d_SNP%d.rds",ifold,irep)
  training <- SNPdata[SNPdata$fold != ifold, ]
  # check mutation
  if (nlevels(training$SNP) < 2) {
    saveRDS(c(SNPname = SNPname, pvalue = 1), file = pvalue_rds_10folds)
  }
  else {
    glm_null <- glm(formula = paste0(Y_name, "~", X_cov), data = training, family = "binomial")
    glm_full <- glm(formula = paste0(Y_name, "~", X_cov, "+", X_name), data = training,
                    family = "binomial")
    lrtest <- LRtest(glm_null, glm_full)
    pvalue <- lrtest$pvalue
    saveRDS(c(SNPname = SNPname, pvalue = pvalue), file = pvalue_rds_10folds)
  }
}

################ calculate p-value conditional on APOE for X chromosome ################
source("/home/med826/Mayo/utility.r")

if (!exists("irep"))  irep <- 1
irep <- irep+300768

SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", irep)
cov_file <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"
pvalue_file <- "/home/med826/Mayo/ICV/pvalue/"
```

```
# get the single SNP
SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
SNPname <- substr(colnames(SNP_single_rawdata)[7], 1, nchar(colnames(SNP_single_rawdata)[7])-2)
SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]

# phenotype
covdata <- readRDS(cov_file)
covaritedata <- covdata[order(covdata$IID),]

# imputation with mode and add sex for covariate
snp_male <- SNP_single[which(SNP_single$SEX==1),7]
snp_male_imputed <- snp_male
snp_male_imputed[is.na(snp_male)] <- Mode(snp_male,na.rm = T)
snp_male_imputed[snp_male_imputed==0] <- 11
snp_male_imputed[snp_male_imputed==2] <- 12

snp_female <- SNP_single[which(SNP_single$SEX==2),7]
snp_female_imputed <- snp_female
snp_female_imputed[is.na(snp_female)] <- Mode(snp_female,na.rm = T)
snp_female_imputed[snp_female_imputed==0] <- 21
snp_female_imputed[snp_female_imputed==1] <- 22
snp_female_imputed[snp_female_imputed==2] <- 23

SNP_single[SNP_single$SEX==1,7] <- snp_male_imputed
SNP_single[SNP_single$SEX==2,7] <- snp_female_imputed

SNPdata <- cbind.data.frame(fold = covdata$tenfolds, pheno=covdata$Dx, sex=covdata$Sex,
                            apoe=covdata$APOE4_dosage..0.1.2., SNP = factor(SNP_single[,7]))

Y_name <- "pheno"
X_cov <- "apoe"
X_name <- "SNP"

kfolds=10
for (ifold in 1:kfolds) {
  pvalue_rds_10folds <- sprintf("/home/med826/Mayo/ECV/pvalue_apoe/pvalue_fold%d_SNP%d.rds",ifold,irep)
  training <- SNPdata[SNPdata$fold != ifold, ]

  if (nlevels(training$SNP) < 2) {
    saveRDS(c(SNPname = SNPname, pvalue = 1), file = pvalue_rds_10folds)
  }
  else {
    glm_null <- glm(formula = paste0(Y_name, "~", X_cov), data = training, family = "binomial")
    glm_full <- glm(formula = paste0(Y_name, "~", X_cov, "+", X_name), data = training,
                    family = "binomial")
    lrtest <- LRtest(glm_null, glm_full)
    pvalue <- lrtest$pvalue
    saveRDS(c(SNPname = SNPname, pvalue = pvalue), file = pvalue_rds_10folds)
  }
}

################ convert p-values into Fdr ################
if (!exists("irep")) irep <- 1
```

```
library("fdrtool")
Pvalue_path <- "/home/med826/Mayo/ECV/pvalue_apoe/"
evalpath <- "/home/med826/Mayo/ECV/evaluation/"
datapath <- "/home/med826/Mayo/ECV/dataset/"

Pvalue_list <- list.files(path = Pvalue_path, pattern = sprintf("^pvalue_fold%d_.*\\.rds$", irep),
                          full.names = T)

Pvalue_all <- do.call('rbind', lapply(Pvalue_list, readRDS))
pvalue <- as.numeric(Pvalue_all[,2])
ordered.name <- Pvalue_all[order(pvalue, decreasing = F),1]

fdr <- fdrtool(pvalue,statistic = "pvalue", plot = F, cutoff.method = "fndr")
#lfdr in fdrtool is the local fdr
#qval in fdrtool is the tail area-based fdr
qvalue <- fdr$qval
mlog.qvalue <- -log (qvalue, base = 10)
mll.sorted <- sort(mlog.qvalue, decreasing = T)
xaxis <- log2(1:309549)
jpeg(paste0(evalpath, sprintf("qvalue_fold%d.jpg", irep)), width = 480, height = 480)
par (mar = c(4,4,0.5,0.5))
plot(xaxis, mll.sorted, type = "h", xlab="", ylab="", xaxp=c(0,19,19))
title(xlab = "log2(rank of SNPs)", ylab = sprintf("-log(q-values) in fold%d", irep),
      cex.lab = 1.5, line = 2.5)
dev.off()

top5000 <- cbind(snpname = ordered.name[1:5000], qvalue = sort(qvalue, decreasing = F)[1:5000])
saveRDS(top5000, file = paste0(datapath, sprintf("top5000SNPs_fold%d.rds",irep)))

################ Re-order SNP given the the rank of SNPs based on all samples################
if (!exists("irep")) irep <- 1

library("fdrtool")
Pvalue_ICV <- "/home/med826/Mayo/ICV/pvalue"
Pvalue_ECV <- "/home/med826/Mayo/ECV/pvalue_apoe"
evalpath <- "/home/med826/Mayo/ECV/evaluation/"

Pvalue_ICV_list <- list.files(path = Pvalue_ICV, full.names = T)
Pvalue_ECV_list <- list.files(path = Pvalue_ECV, pattern = sprintf("^pvalue_fold%d_.*\\.rds$", irep),
                              full.names = T)

ICVlist <- do.call('rbind', lapply(Pvalue_ICV_list, readRDS))
ECVlist <- do.call('rbind', lapply(Pvalue_ECV_list, readRDS))

pvalue_icv <- as.numeric(ICVlist[,2])
ordered.name <- ICVlist[order(pvalue_icv, decreasing = F),1]
ordered.ICVlist <- as.numeric(ICVlist[order(pvalue_icv, decreasing = F),2])

fdr_icv <- fdrtool(ordered.ICVlist, statistic = "pvalue", plot = F, cutoff.method = "fndr")
qvalue_icv <- fdr_icv$qval
Qvalue_ICV_list <- cbind.data.frame(name=ordered.name, qvalue_icv=qvalue_icv)

fdr_ecv <- fdrtool(as.numeric(ECVlist[,2]), statistic = "pvalue", plot = F, cutoff.method = "fndr")
qvalue_ecv <-fdr_ecv$qval
```

```
Qvalue_ECV_list <- cbind.data.frame(name=ECVlist[,1], qvalue_ecv=qvalue_ecv)

qvalue_merged <- join(Qvalue_ICV_list, Qvalue_ECV_list)

xaxis <- log2(1:309549)
jpeg(paste0(evalpath, sprintf("fdr_compare_fold%d.jpg",irep)), width = 480, height = 480)
par (mar = c(4,4,0.5,0.5))
plot(xaxis, -log(qvalue_merged$qvalue_ecv,base = 10), type = "h", xlab = "",
     ylab = "", xaxp=c(0,19,19))
title (xlab = "log2(SNP rank based on all samples)",
       ylab = sprintf("-log(Fdr) in fold%d", irep), cex.lab=1.5, line =2.5)
dev.off()

############### model fitting ###############
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

datapath <- "/home/med826/Mayo/ECV/dataset/"
errorpath <- "/home/med826/Mayo/ECV/errorrate/"
evalpath <- "/home/med826/Mayo/ECV/evaluation/"
predpath <- "/home/med826/Mayo/ECV/prediction/"
amlppath <- "/home/med826/Mayo/ECV/amlp/"
predmatpath <- "/home/med826/Mayo/ECV/predmat/"

top5000_file <- paste0(datapath, sprintf("top5000SNPs_fold%d.rds", iloc))
SNPdata_file <- paste0(datapath, sprintf("SNPdata5000_fold%d.rds", iloc))

top5000SNP <- readRDS(top5000_file)
## extract the snp names
names_snp <- top5000SNP[,1]

## get the SNP file
SNPdata <- readRDS(SNPdata_file)

training <- SNPdata[SNPdata$fold != iloc,]
testing <- SNPdata[SNPdata$fold == iloc,]

## nsnp_set <- c(1, 2, 4, 8, ..., 1024, 2048, 4096)
nsnp_set <- c(0, 13)
for ( i in 1:13){
  nsnp_set[i] <- 2^(i-1)
}
nsnp_set

Y_name <- "phenotype"
X_cov <- "apoe"
X_chosen <- names_snp[1:nsnp_set[irep]]

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_chosen), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training, xlev = 3)[, -1]
Y_tr <- as.numeric(training[, Y_name])-1
```

```
X_ts <- model.matrix(fit_formula, data = testing, xlev = 3)[, -1]
Y_ts <- as.numeric(testing[, Y_name])-1

## htlr
htlr.pred <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.htlr <- matrix(0, nrow=length(Y_ts),ncol=3)
er.htlr <- vector()
amlp.htlr <- vector()
alpha1 <- c(0.5, 1, 1.5)
for (j in 1:3){
  htlr.fit <- htlr_fit (
    y_tr = Y_tr, X_tr = X_tr, X_ts = X_ts, stdzx = F, ## data
    pty = "t", alpha = alpha1[j], s = -10,   ## alpha = df and s= log (w)
    iters_h = 1000, iters_rmc = 1000, thin = 10, ## mcmc iteration settings,
    leap_L_h = 5, leap_L = 50, leap_step = 0.3, hmc_sgmcut = 0.3, ## hmc settings
    initial_state = "lasso", silence = !interactive())  ## initial state settings
  htlr.pred[,2*j-1] <- htlr.fit$probs_pred[,1]
  htlr.pred[,2*j] <- htlr.fit$probs_pred[,2]
  htlr.predeval <- evaluate_pred(htlr.fit$probs_pred, Y_ts+1, showplot=F)
  er.htlr[j] <- htlr.predeval$er
  pred.htlr[,j] <- ifelse(htlr.pred[,1] > 0.5, 0, 1)
  amlp.htlr[j] <- htlr.predeval$amlp
}


## lasso prediction
lasso.fit <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, 1)
las.pred <- lasso.fit$eval[["table_eval"]]
predmat.las <- las.pred[, 3:4]
er.las <- lasso.fit$eval[["er"]]
pred.las <- as.numeric(lasso.fit$predictor)
amlp.las <- lasso.fit$eval[["amlp"]]

## elastic net
alpha2 <- c(0.3,0.5,0.7)
predmat.ela <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.ela <- matrix(0, nrow=length(Y_ts),ncol=3)
er.ela <- vector()
amlp.ela <- vector()
for (i in 1:3){
  elastic.net <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, alpha2[i])
  ela.pred <- elastic.net$eval[["table_eval"]]
  predmat.ela[,2*i-1] <- unlist(ela.pred[, 3])
  predmat.ela[,2*i] <- unlist(ela.pred[, 4])
  er.ela[i] <- elastic.net$eval[["er"]]
  pred.ela[,i] <- as.numeric(elastic.net$predictor)
  amlp.ela[i] <- elastic.net$eval[["amlp"]]
}

saveRDS(cbind(pred.htlr, pred.las, pred.ela),
        file = paste0(predpath, sprintf("predictor_ECV_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(er.htlr), er.las,  t(er.ela)),
        file = paste0(errorpath, sprintf("errorrate_ECV_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(amlp.htlr), amlp.las, t(amlp.ela)),
```

```
        file = paste0(amlppath, sprintf("amlp_ECV_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(htlr.pred, predmat.las, predmat.ela),
        file = paste0(predmatpath, sprintf("predmat_ECV_fold%d_nsnp%d.rds",iloc, nsnp_set[irep])))

################ Model Evaluation ################
cvtype <- c("ECV", "ICV")

erpath <- paste0("/home/med826/Mayo/", cvtype, "/errorrate/")
amlppath <- paste0("/home/med826/Mayo/",cvtype,"/amlp/")
evalpath <- "/home/med826/Mayo/ECV/evaluation/"
covpath <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"

nsnp_set <- c(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096)
nset <- length(nsnp_set)

locations <- paste0("fold",1:10)

## calculate the frequency of large coefficient Phenotype and small coefficient Phenotype
phenodata <- readRDS(covpath)$Dx
prob_null <-prop.table(table(phenodata))
er_null <- min(prob_null)
amlp_null <- - sum(prob_null*log(prob_null))

## generate the array of error rate, which contians error rates of hltr, lasso, elastic
## then calculate the mean error rate of all folds
errates <- array(0, dim = c(nset, 14, 11))
er_apoe <- vector()
for (iloc in 1:length(locations)){
  er_ecv_list <- paste0(erpath[1], "errorrate_ECV_", locations[iloc],
                        "_nsnp", nsnp_set, ".rds")
  er_icv_list <- paste0(erpath[2], "errorrate_ICV_", locations[iloc],
                        "_nsnp", nsnp_set, ".rds")
  errates[, 1:7, iloc] <- do.call('rbind', lapply(er_ecv_list, function(x) readRDS(x)[1:7]))
  errates[, 8:14, iloc] <- do.call('rbind', lapply(er_icv_list, function(x) readRDS(x)[1:7]))
  er_apoe[iloc] <- readRDS(sprintf("/home/med826/Mayo/ICV/errorrate/errorrate_apoe_fold%d.rds", iloc))
}

for (i in 1:14){
  errates[,i,11] <- apply(errates[,i,1:10],1,mean)
}
errates_apoe <- mean(er_apoe)

pdf(paste0(evalpath, "errorrate_combine.pdf"), width = 7, height = 7)
par (mar = c(4,4,0.5,4))
matplot(0:12, errates[,c(3,4,7,9,11,12),11],
        ylim = c(0, 0.5),
        type = "b",
        col = rep(c(2,4,3),2),
        lwd = rep(3,6),
        pch = rep(c(1:3),2),
        cex = 1, lty = c(rep(1,3), rep(3,3)),
        xaxp=c(0,13,13),
        yaxp=c(0,0.5,10),
        xlab = "",
```

```
        ylab = "")
abline(h=er_null, lty=4, lwd=2, col="grey58")
abline(h=errates_apoe,lty=4, lwd=2, col="tan1")
axis(side = 4, at=seq(0,er_null,by=er_null/10), labels=seq(1,0,by=-0.1))
mtext(side=4, line=3, expression(R^2), cex=1.5)
title (xlab = "log2(Number of SNPs)",
       ylab = "Error Rate",cex.lab=1.5, line =2.5)
legend("bottomleft", cex=0.8,
       legend = c("Hyper-LASSO", "LASSO", "Elastic Net",
                  "Null", "APOE","External CV", "Internal CV"),
       col = c(2,4,3, "grey58","tan1", 1,1),
       pch = c(1:3,NA,NA, NA,NA),
       lty = c(NA,NA,NA,4,4,1,3))
dev.off()

amlp <- array(0, dim = c(nset, 14, 11))
amlp_apoe <- vector()
for (iloc in 1:length(locations)){
  amlp_ecv_list <- paste0(amlppath[1], "amlp_ECV_",  locations[iloc],
                          "_nsnp", nsnp_set, ".rds")
  amlp_icv_list <- paste0(amlppath[2], "amlp_ICV_",  locations[iloc],
                          "_nsnp", nsnp_set, ".rds")
  amlp[, 1:7, iloc] <- do.call('rbind', lapply(amlp_ecv_list, function(x) readRDS(x)[1:7]))
  amlp[, 8:14, iloc] <- do.call('rbind', lapply(amlp_icv_list, function(x) readRDS(x)[1:7]))
  amlp_apoe[iloc] <- readRDS(sprintf("/home/med826/Mayo/ICV/amlp/amlp_apoe_fold%d.rds", iloc))
}

for (i in 1:14){
  amlp[,i,11] <- apply(amlp[,i,1:10],1,mean)
}
mamlp_apoe <- mean(amlp_apoe)

pdf(paste0(evalpath, "amlp_combine.pdf"), width = 7, height = 7)
par (mar = c(4,4,0.5,4))
matplot(0:12, amlp[,c(3,4,7,9,11,12),11],
        ylim = c(0,1.7),
        type = "b",
        col = c(rep(c(2,4,3),2),1,7),
        lwd = rep(3,6),
        pch = rep(c(1:3),2),
        cex = 1, lty = c(rep(1,3), rep(3,3)),
        xaxp=c(0,13,13),
        yaxp=c(0,1.7,34),
        xlab = "",
        ylab = "")
abline(h=amlp_null, lty=4, lwd=2, col = "grey58")
abline(h=mamlp_apoe,lty=4, lwd=2, col="tan1")
axis(side = 4, at=seq(0,amlp_null*2.5,by=amlp_null/10), labels=seq(1,-1.5,by=-0.1))
mtext(side=4, line=3, expression(R^2),cex=1.5)
title (xlab = "log2(Number of SNPs)",
       ylab = "AMLP", cex.lab = 1.5, line =2.5)
legend("bottomleft", cex=0.8,
       legend = c("Hyper-LASSO", "LASSO", "Elastic Net",
                  "Null","APOE", "External CV", "Internal CV"),
```

```
       col = c(2,4,3, "grey58","tan1", 1,1),
       pch = c(1:3,NA,NA, NA,NA),
       lty = c(NA,NA,NA,4,4,1,3))
dev.off()
```

A.4 R Code for Generating Simulation Data

```
source("/home/med826/Mayo/utility.r")

datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
allsnplist <- "/home/med826/Mayo/SNPdata/alldata_snplist.rds"
SNPs <- "/home/med826/Mayo-origin/10folds_cv/fold1/snp_step1_top2000.rds"
covariate_file <- "/home/med826/Mayo/covariatedata/covariate_processed.rds"
singleSNP_file <- "/home/med826/Mayo/SNPdata/singleSNP/"

# extract 10 SNPs with largest variance
snpname_list <- as.character(readRDS(allsnplist))
SNPs_file <- readRDS(SNPs)
SNPnames <- colnames(SNPs_file)[14:23]
#saveRDS(SNPnames, file = paste0(filepath, "sign_snps.rds"))
snp_index <- which(snpname_list %in% SNPnames)
chosenSNP <- snpname_list[snp_index]


SNP <- matrix(0, 2099, 10)
colnames(SNP) <- SNPnames

for (i in 1:10){
  index_snp <- which(snpname_list %in% SNPnames[i])
  SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", index_snp)
  SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
  SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]
  n <- ncol(SNP_single_rawdata)

  SNP_single_imputeddata <- SNP_single[, n]
  SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)
  SNP[,i] <- SNP_single_imputeddata
}

covariate <- readRDS(covariate_file)
apoe <- covariate$APOE4_dosage..0.1.2.
apoesnp <- cbind.data.frame(apoe, SNP)
saveRDS(apoesnp, file = paste0(datapath,"sign_SNPdata.rds"))

# generate categorical matrix
formula <- as.formula(paste0("~", paste(c("apoe", SNPnames), collapse = "+")))
mm <- model.matrix(formula, data = apoesnp); dim(mm)

## intercept and coeficient of APOE dosage is based on real analysis
intercept <- -1
coef_apoe <- c(1, 2)
```

```
## generate 10 folds
fold <- rep(1:10, length=2099)

## generate small coefficient
coef_small <- c(intercept, coef_apoe, rnorm(20, mean = 0, sd = 0.1)); coef_small
names(coef_small) <- colnames(mm)
linear_small <-mm %*% coef_small
prob_small <- 1/(1+exp(-linear_small))
Phenotype_small <- rbinom(2099, 1, prob_small)
table(Phenotype_small,fold)
saveRDS(coef_small, file = paste0(datapath, "small_truecoef.rds"))
saveRDS(cbind.data.frame(fold, Phenotype_small,apoe), file = paste0(datapath, "sc_data.rds"))

## generate large coefficient
## assign first 5 SNPs as significant and the rest not
coef_large <- c(intercept, coef_apoe, rnorm(20, mean = 0, sd = 2)); coef_large
names(coef_large) <- colnames(mm)
linear_large <- mm %*% coef_large
prob_large <- 1/(1+exp(-linear_large))
Phenotype_large <- rbinom(2099, 1, prob_large)
table(Phenotype_large, fold)
saveRDS(coef_large, file = paste0(datapath, "large_truecoef.rds"))
saveRDS(cbind.data.frame(fold, Phenotype_large, apoe), file = paste0(datapath, "lc_data.rds"))

############### Oracle Case for dataset1 ###############
## oracle case
if (!exists("irep")) irep <- 1
iloc <- irep

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

phenopath <- "/home/med826/Mayo_simulate/10Sign/dataset/sc_data.rds"
trueSNPpath <- "/home/med826/Mayo_simulate/10Sign/dataset/sign_SNPdata.rds"
errorpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
predpath <- "/home/med826/Mayo_simulate/10Sign/prediction/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
coefpath <- "/home/med826/Mayo_simulate/10Sign/coef/"
predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"

phenodata <- readRDS(phenopath)
SNPdata <- readRDS(trueSNPpath)

alldata <- cbind.data.frame(phenodata, SNPdata)

training <- alldata[which(alldata$fold != iloc), ]
testing <- alldata[which(alldata$fold == iloc), ]

X_names <- colnames(SNPdata)
X_cov <- "apoe"
Y_name <- "Phenotype_small"

## feature selection and model training
```

```
fit_formula <- as.formula(paste0(Y_name,"~", paste(c(X_cov, X_names), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training)[, -1]
Y_tr <- training[, Y_name]
X_ts <- model.matrix(fit_formula, data = testing)[, -1]
Y_ts <- testing[, Y_name]

## simple logistic regression
glm.fit <- glm(fit_formula, data = training, family = "binomial")
coef.glm <- coef(glm.fit)
glm.prob <- predict(glm.fit, newdata = testing, type = "response")
pred.glm <- ifelse(glm.prob > 0.5, 1, 0)
eval.glm <- evaluate_pred(cbind(1-glm.prob, glm.prob), Y_ts+1, showplot = F)
er.glm <- eval.glm$er
amlp.glm <- eval.glm$amlp
predmat.glm <- cbind(1-glm.prob, glm.prob)

saveRDS(pred.glm, file = paste0(predpath, sprintf("predictor_oracle_sc_fold%d.rds", iloc)))
saveRDS(er.glm, file = paste0(errorpath, sprintf("errorrate_oracle_sc_fold%d.rds", iloc)))
saveRDS(amlp.glm, file = paste0(amlppath, sprintf("amlp_oracle_sc_fold%d.rds", iloc)))
saveRDS(predmat.glm, file = paste0(predmatpath, sprintf("predmat_oracle_sc_fold%d.rds",iloc)))

################ Oracle Case for dataset2 ################
if (!exists("irep")) irep <- 1
iloc <- irep

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

phenopath <- "/home/med826/Mayo_simulate/10Sign/dataset/lc_data.rds"
trueSNPpath <- "/home/med826/Mayo_simulate/10Sign/dataset/sign_SNPdata.rds"
errorpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
predpath <- "/home/med826/Mayo_simulate/10Sign/prediction/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
coefpath <- "/home/med826/Mayo_simulate/10Sign/coef/"
predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"

phenodata <- readRDS(phenopath)
SNPdata <- readRDS(trueSNPpath)

alldata <- cbind.data.frame(phenodata, SNPdata)
training <- alldata[which(alldata$fold != iloc), ]
testing <- alldata[which(alldata$fold == iloc), ]

X_names <- colnames(SNPdata)
X_cov <- "apoe"
Y_name <- "Phenotype_large"

# feature selection and model training
fit_formula <- as.formula(paste0(Y_name,"~", paste(c(X_cov, X_names), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training)[, -1]
Y_tr <- training[, Y_name]
X_ts <- model.matrix(fit_formula, data = testing)[, -1]
Y_ts <- testing[, Y_name]
```

```
## simple logistic regression
glm.fit <- glm(fit_formula, data = training, family = "binomial")
coef.glm <- coef(glm.fit)
glm.prob <- predict(glm.fit, newdata = testing, type = "response")
pred.glm <- ifelse(glm.prob > 0.5, 1, 0)
eval.glm <- evaluate_pred(cbind(1-glm.prob, glm.prob), Y_ts+1, showplot = F)
er.glm <- eval.glm$er
amlp.glm <- eval.glm$amlp
predmat.glm <- cbind(1-glm.prob, glm.prob)

saveRDS(pred.glm, file = paste0(predpath, sprintf("predictor_oracle_lc_fold%d.rds", iloc)))
saveRDS(er.glm, file = paste0(errorpath, sprintf("errorrate_oracle_lc_fold%d.rds", iloc)))
saveRDS(amlp.glm, file = paste0(amlppath, sprintf("amlp_oracle_lc_fold%d.rds", iloc)))
saveRDS(predmat.glm, file = paste0(predmatpath, sprintf("predmat_oracle_lc_fold%d.rds",iloc)))
```

A.5 R Code for Feature Selection and Model Fitting with Real Dataset

```
################ Calculate p-value for chromosome 1- 22 for dataset1 and dataset2 ################
source("/home/med826/Mayo/utility.r")

if (!exists("irep"))  irep <- 1

SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", irep)
lc_pheno_file <- "/home/med826/Mayo_simulate/10Sign/dataset/lc_data.rds"
sc_pheno_file <- "/home/med826/Mayo_simulate/10Sign/dataset/sc_data.rds"
pvalue_icv_file <- "/home/med826/Mayo_simulate/10Sign/ICV/pvalue/"
pvalue_ecv_file <- "/home/med826/Mayo_simulate/10Sign/ECV/pvalue/"


# get the single SNP
SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
SNPname <- substr(colnames(SNP_single_rawdata)[7], 1, nchar(colnames(SNP_single_rawdata)[7])-2)
SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]

# get fold and phenotype
lc_pheno <- readRDS(lc_pheno_file)
sc_pheno <- readRDS(sc_pheno_file)

## imputation with mode
n <- ncol(SNP_single)
SNP_single_imputeddata <- SNP_single[, n]
SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)

lcdata <- cbind.data.frame(lc_pheno, SNP = SNP_single_imputeddata)
scdata <- cbind.data.frame(sc_pheno, SNP = SNP_single_imputeddata)

Y_lc_name <- "Phenotype_large"
Y_sc_name <- "Phenotype_small"
X_cov <- "apoe"
X_name <- "SNP"
```

```
## calculate p value for Internal CV
## for both large and small coefficient
if (nlevels(lcdata$SNP) < 2) {
  saveRDS(c(SNPname = SNPname, pvalue = 1),
          file = paste0(pvalue_icv_file, sprintf("pvalue_lc%d.rds", irep)))
  saveRDS(c(SNPname = SNPname, pvalue = 1),
          file = paste0(pvalue_icv_file, sprintf("pvalue_sc%d.rds", irep)))
} else {
  glm_null_lc_icv <- glm(formula = paste0(Y_lc_name, "~", X_cov), data = lcdata, family = "binomial")
  glm_full_lc_icv <- glm(formula = paste0(Y_lc_name, "~", X_cov, "+", X_name), data = lcdata,
                         family = "binomial")

  lc_lrtest <- LRtest(glm_null_lc_icv, glm_full_lc_icv)
  lc_pvalue <- lc_lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue = lc_pvalue),
          file = paste0(pvalue_icv_file, sprintf("pvalue_lc%d.rds", irep)))

  glm_null_sc_icv <- glm(formula = paste0(Y_sc_name, "~", X_cov), data = scdata, family = "binomial")
  glm_full_sc_icv <- glm(formula = paste0(Y_sc_name, "~", X_cov, "+", X_name), data = scdata,
                         family = "binomial")

  sc_lrtest <- LRtest(glm_null_sc_icv, glm_full_sc_icv)
  sc_pvalue <- sc_lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue = sc_pvalue),
          file = paste0(pvalue_icv_file, sprintf("pvalue_sc%d.rds", irep)))
}

## pvalue for External CV
kfolds=10
for (ifold in 1:kfolds) {
  pvalue_lc <- sprintf("/home/med826/Mayo_simulate/10Sign/ECV/fold%d/pvalue/pvalue_lc%d.rds",
                       ifold,irep)
  pvalue_sc <- sprintf("/home/med826/Mayo_simulate/10Sign/ECV/fold%d/pvalue/pvalue_sc%d.rds",
                       ifold,irep)

  training_lc <- lcdata[lcdata$fold != ifold, ]
  training_sc <- scdata[scdata$fold != ifold, ]

if (nlevels(training_lc$SNP) < 2) {
  saveRDS(c(SNPname = SNPname, pvalue = 1), file = pvalue_lc)
  saveRDS(c(SNPname = SNPname, pvalue = 1), file = pvalue_sc)
}
else {
  glm_null_lc_ecv <- glm(formula = paste0(Y_lc_name, "~", X_cov), data = training_lc,
                         family = "binomial")
  glm_full_lc_ecv <- glm(formula = paste0(Y_lc_name, "~", X_cov, "+", X_name), data = training_lc,
                         family = "binomial")

  lc.lrtest <- LRtest(glm_null_lc_ecv, glm_full_lc_ecv)
  lc.pvalue <- lc.lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue = lc.pvalue), file = pvalue_lc)

  glm_null_sc_ecv <- glm(formula = paste0(Y_sc_name, "~", X_cov), data = training_sc,
```

```
                        family = "binomial")
  glm_full_sc_ecv <- glm(formula = paste0(Y_sc_name, "~", X_cov, "+", X_name), data = training_sc,
                        family = "binomial")

  sc.lrtest <- LRtest(glm_null_sc_ecv, glm_full_sc_ecv)
  sc.pvalue <- sc.lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue = sc.pvalue), file = pvalue_sc)
  }
}

################ Calculate p-value for chromosome X for dataset1 and dataset2 ################
source("/home/med826/Mayo/utility.r")

for (irep in 300769:309550){
SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", irep)
lc_pheno_file <- "/home/med826/Mayo_simulate/10Sign/dataset/lc_data.rds"
sc_pheno_file <- "/home/med826/Mayo_simulate/10Sign/dataset/sc_data.rds"
pvalue_icv_file <- "/home/med826/Mayo_simulate/10Sign/ICV/pvalue/"
pvalue_ecv_file <- "/home/med826/Mayo_simulate/10Sign/ECV/pvalue/"


# get the single SNP
SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
SNPname <- substr(colnames(SNP_single_rawdata)[7], 1, nchar(colnames(SNP_single_rawdata)[7])-2)
SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]

# get fold and phenotype
lc_pheno <- readRDS(lc_pheno_file)
sc_pheno <- readRDS(sc_pheno_file)

## imputation with mode
n <- ncol(SNP_single)

snp_male <- SNP_single[which(SNP_single$SEX==1),n]
snp_male_imputed <- snp_male
snp_male_imputed[is.na(snp_male)] <- Mode(snp_male,na.rm = T)
snp_male_imputed[snp_male_imputed==0] <- 11
snp_male_imputed[snp_male_imputed==2] <- 12

snp_female <- SNP_single[which(SNP_single$SEX==2),n]
snp_female_imputed <- snp_female
snp_female_imputed[is.na(snp_female)] <- Mode(snp_female,na.rm = T)
snp_female_imputed[snp_female_imputed==0] <- 21
snp_female_imputed[snp_female_imputed==1] <- 22
snp_female_imputed[snp_female_imputed==2] <- 23

SNP_single[SNP_single$SEX==1,n] <- snp_male_imputed
SNP_single[SNP_single$SEX==2,n] <- snp_female_imputed

lcdata <- cbind.data.frame(lc_pheno, SNP = SNP_single[,n])
scdata <- cbind.data.frame(sc_pheno, SNP = SNP_single[,n])

Y_lc_name <- "Phenotype_large"
Y_sc_name <- "Phenotype_small"
```

```
X_cov <- "apoe"
X_name <- "SNP"

## calculate p value for Internal CV
## for both large and small coefficient
if (nlevels(lcdata$SNP) < 2) {
  saveRDS(c(SNPname = SNPname, pvalue = 1),
          file = paste0(pvalue_icv_file, sprintf("pvalue_lc%d.rds", irep)))
  saveRDS(c(SNPname = SNPname, pvalue = 1),
          file = paste0(pvalue_icv_file, sprintf("pvalue_sc%d.rds", irep)))
} else {
  glm_null_lc_icv <- glm(formula = paste0(Y_lc_name, "~", X_cov), data = lcdata, family = "binomial")
  glm_full_lc_icv <- glm(formula = paste0(Y_lc_name, "~", X_cov, "+", X_name), data = lcdata,
                         family = "binomial")

  lc_lrtest <- LRtest(glm_null_lc_icv, glm_full_lc_icv)
  lc_pvalue <- lc_lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue = lc_pvalue),
          file = paste0(pvalue_icv_file, sprintf("pvalue_lc%d.rds", irep)))

  glm_null_sc_icv <- glm(formula = paste0(Y_sc_name, "~", X_cov), data = scdata, family = "binomial")
  glm_full_sc_icv <- glm(formula = paste0(Y_sc_name, "~", X_cov, "+", X_name), data = scdata,
                         family = "binomial")

  sc_lrtest <- LRtest(glm_null_sc_icv, glm_full_sc_icv)
  sc_pvalue <- sc_lrtest$pvalue
  saveRDS(c(SNPname = SNPname, pvalue = sc_pvalue),
          file = paste0(pvalue_icv_file, sprintf("pvalue_sc%d.rds", irep)))
}

## pvalue for External CV
kfolds=10
for (ifold in 1:kfolds) {
  pvalue_lc <- sprintf("/home/med826/Mayo_simulate/10Sign/ECV/fold%d/pvalue/pvalue_lc%d.rds",
                       ifold,irep)
  pvalue_sc <- sprintf("/home/med826/Mayo_simulate/10Sign/ECV/fold%d/pvalue/pvalue_sc%d.rds",
                       ifold,irep)

  training_lc <- lcdata[lcdata$fold != ifold, ]
  training_sc <- scdata[scdata$fold != ifold, ]

  if (nlevels(training_lc$SNP) < 2) {
    saveRDS(c(SNPname = SNPname, pvalue = 1), file = pvalue_lc)
    saveRDS(c(SNPname = SNPname, pvalue = 1), file = pvalue_sc)
  }
  else {
    glm_null_lc_ecv <- glm(formula = paste0(Y_lc_name, "~", X_cov), data = training_lc,
                           family = "binomial")
    glm_full_lc_ecv <- glm(formula = paste0(Y_lc_name, "~", X_cov, "+", X_name), data = training_lc,
                           family = "binomial")

    lc.lrtest <- LRtest(glm_null_lc_ecv, glm_full_lc_ecv)
    lc.pvalue <- lc.lrtest$pvalue
    saveRDS(c(SNPname = SNPname, pvalue = lc.pvalue), file = pvalue_lc)
```

```
    glm_null_sc_ecv <- glm(formula = paste0(Y_sc_name, "~", X_cov), data = training_sc,
                           family = "binomial")
    glm_full_sc_ecv <- glm(formula = paste0(Y_sc_name, "~", X_cov, "+", X_name), data = training_sc,
                           family = "binomial")

    sc.lrtest <- LRtest(glm_null_sc_ecv, glm_full_sc_ecv)
    sc.pvalue <- sc.lrtest$pvalue
    saveRDS(c(SNPname = SNPname, pvalue = sc.pvalue), file = pvalue_sc)
  }
}
}

############### Fdr for ICV ###############
if (!exists("irep")) irep <- 1

library("fdrtool")

Pvalue_path <- "/home/med826/Mayo_simulate/10Sign/ICV/pvalue"
SignifSNP <- "/home/med826/Mayo_simulate/10Sign/dataset/sign_snps.rds"
datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"

datatype <- c("lc", "sc")
datatype2 <<- c("large coefficient", "small coefficient")
Pvalue_list <- list.files(path = Pvalue_path,
                          pattern = sprintf("^pvalue_%s.*\\.rds$", datatype[irep]), full.names = T)
sign_snps <- readRDS(SignifSNP)

Pvalue_all <- do.call('rbind', lapply(Pvalue_list, readRDS))
pvalue <- as.numeric(Pvalue_all[,2])
ordered.name <- Pvalue_all[order(pvalue, decreasing = F),1]
index.sign <- which(ordered.name %in% sign_snps)

fdr <- fdrtool(pvalue,statistic = "pvalue", plot = F, cutoff.method = "fndr")
#lfdr in fdrtool is the local fdr
#qval in fdrtool is the tail area-based fdr
qvalue <- fdr$qval
mlog.qvalue <- -log (qvalue, base = 10)
mll.sorted <- sort(mlog.qvalue, decreasing = T)
jpeg(paste0(evalpath, sprintf("fdr_%s_ICV_2.jpg", datatype[irep])), width = 450, height = 350)
par (mar = c(4,4,0.5,0.5))
xaxis <- log2(1:309549)
plot(xaxis, mll.sorted, type = "h", xlab = "", ylab = "", xaxp=c(0,19,19))
points(log2(index.sign), mll.sorted[index.sign], col = "red", pch = 19, cex = 1)
title (xlab = "log2(rank of SNPs)",
       ylab = "-log(Fdr)", cex.lab=1.5, line =2.5)
dev.off()

top5000 <- cbind(snpname = ordered.name[1:5000], qvalue = sort(qvalue, decreasing = F)[1:5000])
saveRDS(top5000, file = paste0(datapath, sprintf("top5000SNPs_%s_ICV.rds", datatype[irep])))

############### Select 5000 for ICV ###############
if (!exists("irep")) irep <- 1
```

```
source("/home/med826/Mayo/utility.r")

datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"

datatype <- c("lc", "sc")

top5000_file <- paste0(datapath, sprintf("top5000SNPs_%s_ICV.rds", datatype[irep]))
infodata_file <- paste0(datapath, sprintf("%s_data.rds", datatype[irep]))
allsnplist <- "/home/med826/Mayo/SNPdata/alldata_snplist.rds"

top5000SNP <- readRDS(top5000_file)
## infodata contain fold, phenotype, apoe
infodata <- readRDS(infodata_file)

## extract the snp names
names_snp <- top5000SNP[,1]
snpname_list <- as.character(readRDS(allsnplist))

## NULL matrix
SNP <- matrix(0, 2099, 5000)
colnames(SNP) <- names_snp
for (i in 1:5000){
  index_snp <- which(snpname_list %in% names_snp[i])
  SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", index_snp)
  SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
  SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]
  n <- ncol(SNP_single_rawdata)
  if (index_snp > 300767){
    snp_male <- SNP_single[which(SNP_single$SEX==1),n]
    snp_male_imputed <- snp_male
    snp_male_imputed[is.na(snp_male)] <- Mode(snp_male,na.rm = T)
    snp_male_imputed[snp_male_imputed==0] <- 11
    snp_male_imputed[snp_male_imputed==2] <- 12

    snp_female <- SNP_single[which(SNP_single$SEX==2),7]
    snp_female_imputed <- snp_female
    snp_female_imputed[is.na(snp_female)] <- Mode(snp_female,na.rm = T)
    snp_female_imputed[snp_female_imputed==0] <- 21
    snp_female_imputed[snp_female_imputed==1] <- 22
    snp_female_imputed[snp_female_imputed==2] <- 23

    SNP_single[SNP_single$SEX==1,n] <- snp_male_imputed
    SNP_single[SNP_single$SEX==2,n] <- snp_female_imputed
    SNP[,i] <- SNP_single[,n]
  } else{
    SNP_single_imputeddata <- SNP_single[, n]
    SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)
    SNP[,i] <- SNP_single_imputeddata
  }
}
SNPdata <- cbind(SNP, infodata)

saveRDS(SNPdata, file = paste0(datapath, sprintf("SNPdata5000_%s_ICV.rds", datatype[irep])))
```

```
############### Fdr for ECV ###############
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

library("fdrtool")

Pvalue_path <- sprintf("/home/med826/Mayo_simulate/10Sign/ECV/fold%d/pvalue",iloc)
SignifSNP <- "/home/med826/Mayo_simulate/10Sign/dataset/sign_snps.rds"
datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"

datatype <- c("lc", "sc")
datatype2 <<- c("large coefficient", "small coefficient")
Pvalue_list <- list.files(path = Pvalue_path,
                          pattern = sprintf("^pvalue_%s.*\\.rds$", datatype[irep]), full.names = T)
sign_snps <- readRDS(SignifSNP)

Pvalue_all <- do.call('rbind', lapply(Pvalue_list, readRDS))
pvalue <- as.numeric(Pvalue_all[,2])
ordered.name <- Pvalue_all[order(pvalue, decreasing = F),1]
index.sign <- which(ordered.name %in% sign_snps)

fdr <- fdrtool(pvalue,statistic = "pvalue", plot = F, cutoff.method = "fndr")
#lfdr in fdrtool is the local fdr
#qval in fdrtool is the tail area-based fdr
qvalue <- fdr$qval
mlog.qvalue <- -log (qvalue, base = 10)
mll.sorted <- sort(mlog.qvalue, decreasing = T)
xaxis <- log2(1:309549)
jpeg(paste0(evalpath, sprintf("fdr_%s_ECV_fold%d_2.jpg", datatype[irep], iloc)),
            width = 450, height = 350)
par (mar = c(4,4,0.5,0.5))
plot(xaxis, mll.sorted, type = "h", xlab="", ylab="", xaxp=c(0,19,19))#,  main = title)
points(log2(index.sign), mll.sorted[index.sign], col = "red", pch = 19, cex = 1)
title (xlab = "log2(rank of SNPs)",
       ylab = sprintf("-log(Fdr) in fold%d", iloc), cex.lab=1.5, line =2.5)
dev.off()

top5000 <- cbind(snpname = ordered.name[1:5000], qvalue = sort(qvalue, decreasing = F)[1:5000])
saveRDS(top5000, file = paste0(datapath,
        sprintf("top5000SNPs_%s_ECV_fold%d.rds", datatype[irep], iloc)))

############### Fdr comparisons ###############
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

library("fdrtool")
library("plyr")

Pvalue_ICV <- "/home/med826/Mayo_simulate/10Sign/ICV/pvalue"
Pvalue_ECV <- sprintf("/home/med826/Mayo_simulate/10Sign/ECV/fold%d/pvalue",iloc)
SignifSNP <- "/home/med826/Mayo_simulate/10Sign/dataset/sign_snps.rds"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
```

```
datatype <- c("lc", "sc")
Pvalue_ICV_list <- list.files(path = Pvalue_ICV,
                  pattern = sprintf("^pvalue_%s.*\\.rds$", datatype[irep]), full.names = T)
Pvalue_ECV_list <- list.files(path = Pvalue_ECV,
                  pattern = sprintf("^pvalue_%s.*\\.rds$", datatype[irep]), full.names = T)

ICVlist <- do.call('rbind', lapply(Pvalue_ICV_list, readRDS))
ECVlist <- do.call('rbind', lapply(Pvalue_ECV_list, readRDS))

pvalue_icv <- as.numeric(ICVlist[,2])
ordered.name <- ICVlist[order(pvalue_icv, decreasing = F),1]
ordered.ICVlist <- as.numeric(ICVlist[order(pvalue_icv, decreasing = F),2])

sign_snps <- readRDS(SignifSNP)
index.sign <- which(ordered.name %in% sign_snps)

fdr_icv <- fdrtool(ordered.ICVlist, statistic = "pvalue", plot = F, cutoff.method = "fndr")
qvalue_icv <- fdr_icv$qval
Qvalue_ICV_list <- cbind.data.frame(name=ordered.name, qvalue_icv=qvalue_icv)

fdr_ecv <- fdrtool(as.numeric(ECVlist[,2]), statistic = "pvalue", plot = F, cutoff.method = "fndr")
qvalue_ecv <-fdr_ecv$qval
Qvalue_ECV_list <- cbind.data.frame(name=ECVlist[,1], qvalue_ecv=qvalue_ecv)

qvalue_merged <- join(Qvalue_ICV_list, Qvalue_ECV_list)

xaxis <- log2(1:309549)
jpeg(paste0(evalpath, sprintf("fdr_compare_%s_fold%d_2.jpg", datatype[irep], iloc)),
            width = 450, height = 350)
par (mar = c(4,4,0.5,0.5))
plot(xaxis, -log(qvalue_merged$qvalue_ecv,base = 10), type = "h", xlab = "", ylab = "",
     xaxp=c(0,19,19))
points(log2(index.sign), -log(qvalue_merged$qvalue_ecv, base = 10)[index.sign],
       col="red", pch=19, cex=1)
title (xlab = "log2(SNP rank based on all samples)",
       ylab = sprintf("-log(Fdr) in fold%d", iloc),
       cex.lab=1.5, line =2.5)
dev.off()


############### Select 5000 for ECV ###############
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")

datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"

datatype <- c("lc", "sc")

top5000_file <- paste0(datapath, sprintf("top5000SNPs_%s_ECV_fold%d.rds", datatype[irep], iloc))
infodata_file <- paste0(datapath, sprintf("%s_data.rds", datatype[irep]))
allsnplist <- "/home/med826/Mayo/SNPdata/alldata_snplist.rds"
```

```
top5000SNP <- readRDS(top5000_file)
## infodata contain fold, phenotype, apoe
infodata <- readRDS(infodata_file)

## extract the snp names
names_snp <- top5000SNP[,1]
snpname_list <- as.character(readRDS(allsnplist))

## generate NULL matrix
SNP <- matrix(0, 2099, 5000)
colnames(SNP) <- names_snp
for (i in 1:5000){
  index_snp <- which(snpname_list %in% names_snp[i])
  SNP_single_rawfile <- sprintf("/home/med826/Mayo/SNPdata/singleSNP/snp%d.raw", index_snp)
  SNP_single_rawdata <- read.table(SNP_single_rawfile, header = T)
  SNP_single <- SNP_single_rawdata[order(SNP_single_rawdata$IID),]
  n <- ncol(SNP_single_rawdata)
  if (index_snp > 300767){
    snp_male <- SNP_single[which(SNP_single$SEX==1),n]
    snp_male_imputed <- snp_male
    snp_male_imputed[is.na(snp_male)] <- Mode(snp_male,na.rm = T)
    snp_male_imputed[snp_male_imputed==0] <- 11
    snp_male_imputed[snp_male_imputed==2] <- 12

    snp_female <- SNP_single[which(SNP_single$SEX==2),7]
    snp_female_imputed <- snp_female
    snp_female_imputed[is.na(snp_female)] <- Mode(snp_female,na.rm = T)
    snp_female_imputed[snp_female_imputed==0] <- 21
    snp_female_imputed[snp_female_imputed==1] <- 22
    snp_female_imputed[snp_female_imputed==2] <- 23

    SNP_single[SNP_single$SEX==1,n] <- snp_male_imputed
    SNP_single[SNP_single$SEX==2,n] <- snp_female_imputed
    SNP[,i] <- SNP_single[,n]
  } else{
    SNP_single_imputeddata <- SNP_single[, n]
    SNP_single_imputeddata[is.na(SNP_single[, n])] = Mode(SNP_single[, n], na.rm = T)
    SNP[,i] <- SNP_single_imputeddata
  }
}

SNPdata <- cbind(SNP, infodata)

saveRDS(SNPdata, file = paste0(datapath, sprintf("SNPdata5000_%s_ECV_fold%d.rds",
        datatype[irep], iloc)))

################ Mdeol fit for ICV dataset1 ################
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")
```

```
datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
errorpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
predpath <- "/home/med826/Mayo_simulate/10Sign/prediction/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
coefpath <- "/home/med826/Mayo_simulate/10Sign/coef/"
predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"

top5000_file <- paste0(datapath, sprintf("top5000SNPs_sc_ICV.rds"))
SNPdata_file <- paste0(datapath, sprintf("SNPdata5000_sc_ICV.rds"))

top5000SNP <- readRDS(top5000_file)
## extract the snp names
names_snp <- top5000SNP[,1]

## get the SNP file
SNPdata <- readRDS(SNPdata_file)

training <- SNPdata[SNPdata$fold != iloc,]
testing <- SNPdata[SNPdata$fold == iloc,]


## nsnp_set <- c(1, 2, 4, 8, ..., 1024, 2048, 4096)
nsnp_set <- c(0, 13)
for ( i in 1:13){
  nsnp_set[i] <- 2^(i-1)
}
nsnp_set

Y_name <- "Phenotype_small"
X_cov <- "apoe"
X_chosen <- names_snp[1:nsnp_set[irep]]

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_chosen), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training)[, -1]
Y_tr <- training[, Y_name]
X_ts <- model.matrix(fit_formula, data = testing)[, -1]
Y_ts <- testing[, Y_name]


## htlr
htlr.pred <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.htlr <- matrix(0, nrow=length(Y_ts),ncol=3)
er.htlr <- vector()
amlp.htlr <- vector()
alpha1 <- c(0.5, 1, 1.5)
for (j in 1:3){
  htlr.fit <- htlr_fit (
    y_tr = Y_tr, X_tr = X_tr, X_ts = X_ts, stdzx = F, ## data
    pty = "t", alpha = alpha1[j], s = -10,   ## alpha = df and s= log (w)
    iters_h = 1000, iters_rmc = 1000, thin = 10, ## mcmc iteration settings,
    leap_L_h = 5, leap_L = 50, leap_step = 0.3, hmc_sgmcut = 0.3, ## hmc settings
    initial_state = "lasso", silence = !interactive())  ## initial state settings
  htlr.pred[,2*j-1] <- htlr.fit$probs_pred[,1]
```

```
  htlr.pred[,2*j] <- htlr.fit$probs_pred[,2]
  htlr.predeval <- evaluate_pred(htlr.fit$probs_pred, Y_ts+1, showplot=F)
  er.htlr[j] <- htlr.predeval$er
  pred.htlr[,j] <- ifelse(htlr.pred[,1] > 0.5, 0, 1)
  amlp.htlr[j] <- htlr.predeval$amlp
}

## lasso prediction
lasso.fit <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, 1)
las.pred <- lasso.fit$eval[["table_eval"]]
predmat.las <- las.pred[, 3:4]
er.las <- lasso.fit$eval[["er"]]
pred.las <- as.numeric(lasso.fit$predictor)
amlp.las <- lasso.fit$eval[["amlp"]]

## elastic net
alpha2 <- c(0.3,0.5,0.7)
predmat.ela <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.ela <- matrix(0, nrow=length(Y_ts),ncol=3)
er.ela <- vector()
amlp.ela <- vector()
for (i in 1:3){
  elastic.net <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, alpha2[i])
  ela.pred <- elastic.net$eval[["table_eval"]]
  predmat.ela[,2*i-1] <- unlist(ela.pred[, 3])
  predmat.ela[,2*i] <- unlist(ela.pred[, 4])
  er.ela[i] <- elastic.net$eval[["er"]]
  pred.ela[,i] <- as.numeric(elastic.net$predictor)
  amlp.ela[i] <- elastic.net$eval[["amlp"]]
}

saveRDS(cbind(pred.htlr, pred.las, ela = pred.ela),
        file = paste0(predpath, sprintf("predictor_ICV_sc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(er.htlr), er.las, elastic = t(er.ela)),
        file = paste0(errorpath, sprintf("errorrate_ICV_sc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(amlp.htlr), amlp.las, elastic = t(amlp.ela)),
        file = paste0(amlppath, sprintf("amlp_ICV_sc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(htlr.pred, predmat.las, predmat.ela),
        file = paste0(predmatpath, sprintf("predmat_ICV_sc_fold%d_nsnp%d.rds",iloc, nsnp_set[irep])))

################ Model fit for ICV dataset2 ################
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

phenopath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
errorpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
predpath <- "/home/med826/Mayo_simulate/10Sign/prediction/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
coefpath <- "/home/med826/Mayo_simulate/10Sign/coef/"
predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"
```

```
top5000_file <- paste0(phenopath, sprintf("top5000SNPs_lc_ICV.rds" ))
SNPdata_file <- paste0(phenopath, sprintf("SNPdata5000_lc_ICV.rds"))

top5000SNP <- readRDS(top5000_file)
## extract the snp names
names_snp <- top5000SNP[,1]

## get the SNP file
SNPdata <- readRDS(SNPdata_file)

training <- SNPdata[SNPdata$fold != iloc,]
testing <- SNPdata[SNPdata$fold == iloc,]

## nsnp_set <- c(1, 2, 4, 8, ..., 1024, 2048, 4096)
nsnp_set <- c(0, 13)
for ( i in 1:13){
  nsnp_set[i] <- 2^(i-1)
}
nsnp_set

Y_name <- "Phenotype_large"
X_cov <- "apoe"
X_chosen <- names_snp[1:nsnp_set[irep]]

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_chosen), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training)[, -1]
Y_tr <- training[, Y_name]
X_ts <- model.matrix(fit_formula, data = testing)[, -1]
Y_ts <- testing[, Y_name]

## htlr
htlr.pred <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.htlr <- matrix(0, nrow=length(Y_ts),ncol=3)
er.htlr <- vector()
amlp.htlr <- vector()
alpha1 <- c(0.5, 1, 1.5)
for (j in 1:3){
  htlr.fit <- htlr_fit (
    y_tr = Y_tr, X_tr = X_tr, X_ts = X_ts, stdzx = F, ## data
    pty = "t", alpha = alpha1[j], s = -10,   ## alpha = df and s= log (w)
    iters_h = 1000, iters_rmc = 1000, thin = 10, ## mcmc iteration settings,
    leap_L_h = 5, leap_L = 50, leap_step = 0.3, hmc_sgmcut = 0.3, ## hmc settings
    initial_state = "lasso", silence = !interactive())  ## initial state settings
  htlr.pred[,2*j-1] <- htlr.fit$probs_pred[,1]
  htlr.pred[,2*j] <- htlr.fit$probs_pred[,2]
  htlr.predeval <- evaluate_pred(htlr.fit$probs_pred, Y_ts+1, showplot=F)
  er.htlr[j] <- htlr.predeval$er
  pred.htlr[,j] <- ifelse(htlr.pred[,1] > 0.5, 0, 1)
  amlp.htlr[j] <- htlr.predeval$amlp
}

## lasso prediction
lasso.fit <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, 1)
```

```
las.pred <- lasso.fit$eval[["table_eval"]]
predmat.las <- las.pred[, 3:4]
er.las <- lasso.fit$eval[["er"]]
pred.las <- as.numeric(lasso.fit$predictor)
amlp.las <- lasso.fit$eval[["amlp"]]

## elastic net
alpha2 <- c(0.3,0.5,0.7)
predmat.ela <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.ela <- matrix(0, nrow=length(Y_ts),ncol=3)
er.ela <- vector()
amlp.ela <- vector()
for (i in 1:3){
  elastic.net <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, alpha2[i])
  ela.pred <- elastic.net$eval[["table_eval"]]
  predmat.ela[,2*i-1] <- unlist(ela.pred[, 3])
  predmat.ela[,2*i] <- unlist(ela.pred[, 4])
  er.ela[i] <- elastic.net$eval[["er"]]
  pred.ela[,i] <- as.numeric(elastic.net$predictor)
  amlp.ela[i] <- elastic.net$eval[["amlp"]]
}

saveRDS(cbind(pred.htlr, pred.las, pred.ela),
        file = paste0(predpath, sprintf("predictor_ICV_lc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(er.htlr), er.las, t(er.ela)),
        file = paste0(errorpath, sprintf("errorrate_ICV_lc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(amlp.htlr), amlp.las, t(amlp.ela)),
        file = paste0(amlppath, sprintf("amlp_ICV_lc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))

saveRDS(cbind(htlr.pred, predmat.las, predmat.ela),
        file = paste0(predmatpath, sprintf("predmat_ICV_lc_fold%d_nsnp%d.rds",iloc, nsnp_set[irep])))

################ Model fit ECV for dataset1 ################
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
errorpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
predpath <- "/home/med826/Mayo_simulate/10Sign/prediction/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
coefpath <- "/home/med826/Mayo_simulate/10Sign/coef/"
predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"

top5000_file <- paste0(datapath, sprintf("top5000SNPs_sc_ECV_fold%d.rds", iloc))
SNPdata_file <- paste0(datapath, sprintf("SNPdata5000_sc_ECV_fold%d.rds", iloc))

top5000SNP <- readRDS(top5000_file)
## extract the snp names
names_snp <- top5000SNP[,1]
```

```
## get the SNP file
SNPdata <- readRDS(SNPdata_file)

training <- SNPdata[SNPdata$fold != iloc,]
testing <- SNPdata[SNPdata$fold == iloc,]

## nsnp_set <- c(1, 2, 4, 8, ..., 1024, 2048, 4096)
nsnp_set <- c(0, 13)
for ( i in 1:13){
  nsnp_set[i] <- 2^(i-1)
}
nsnp_set

Y_name <- "Phenotype_small"
X_cov <- "apoe"
X_chosen <- names_snp[1:nsnp_set[irep]]

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_chosen), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training, xlev = 3)[, -1]
Y_tr <- training[, Y_name]
X_ts <- model.matrix(fit_formula, data = testing, xlev = 3)[, -1]
Y_ts <- testing[, Y_name]


## htlr
htlr.pred <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.htlr <- matrix(0, nrow=length(Y_ts),ncol=3)
er.htlr <- vector()
amlp.htlr <- vector()
alpha1 <- c(0.5, 1, 1.5)
for (j in 1:3){
  htlr.fit <- htlr_fit (
    y_tr = Y_tr, X_tr = X_tr, X_ts = X_ts, stdzx = F, ## data
    pty = "t", alpha = alpha1[j], s = -10,   ## alpha = df and s= log (w)
    iters_h = 1000, iters_rmc = 1000, thin = 10, ## mcmc iteration settings,
    leap_L_h = 5, leap_L = 50, leap_step = 0.3, hmc_sgmcut = 0.3, ## hmc settings
    initial_state = "lasso", silence = !interactive())  ## initial state settings
  htlr.pred[,2*j-1] <- htlr.fit$probs_pred[,1]
  htlr.pred[,2*j] <- htlr.fit$probs_pred[,2]
  htlr.predeval <- evaluate_pred(htlr.fit$probs_pred, Y_ts+1, showplot=F)
  er.htlr[j] <- htlr.predeval$er
  pred.htlr[,j] <- ifelse(htlr.pred[,1] > 0.5, 0, 1)
  amlp.htlr[j] <- htlr.predeval$amlp
}

## lasso prediction
lasso.fit <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, 1)
las.pred <- lasso.fit$eval[["table_eval"]]
predmat.las <- las.pred[, 3:4]
er.las <- lasso.fit$eval[["er"]]
pred.las <- as.numeric(lasso.fit$predictor)
amlp.las <- lasso.fit$eval[["amlp"]]

## elastic net
```

```
alpha2 <- c(0.3,0.5,0.7)
predmat.ela <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.ela <- matrix(0, nrow=length(Y_ts),ncol=3)
er.ela <- vector()
amlp.ela <- vector()
for (i in 1:3){
  elastic.net <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, alpha2[i])
  ela.pred <- elastic.net$eval[["table_eval"]]
  predmat.ela[,2*i-1] <- unlist(ela.pred[, 3])
  predmat.ela[,2*i] <- unlist(ela.pred[, 4])
  er.ela[i] <- elastic.net$eval[["er"]]
  pred.ela[,i] <- as.numeric(elastic.net$predictor)
  amlp.ela[i] <- elastic.net$eval[["amlp"]]
}

saveRDS(cbind(pred.htlr, pred.las, pred.ela),
        file = paste0(predpath, sprintf("predictor_ECV_sc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(er.htlr), er.las, t(er.ela)),
        file = paste0(errorpath, sprintf("errorrate_ECV_sc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(amlp.htlr), amlp.las, t(amlp.ela)),
        file = paste0(amlppath, sprintf("amlp_ECV_sc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(htlr.pred, predmat.las, predmat.ela),
        file = paste0(predmatpath, sprintf("predmat_ECV_sc_fold%d_nsnp%d.rds",iloc, nsnp_set[irep])))

################ Model fit ECV for dataset2
if (!exists("irep")) irep <- 1
if (!exists("iloc")) iloc <- 1

source("/home/med826/Mayo/utility.r")
library (HTLR, lib.loc = "/home/longhai/Rdev/HTLR_3.1-1")

datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"
errorpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
predpath <- "/home/med826/Mayo_simulate/10Sign/prediction/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
coefpath <- "/home/med826/Mayo_simulate/10Sign/coef/"
predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"

top5000_file <- paste0(datapath, sprintf("top5000SNPs_lc_ECV_fold%d.rds", iloc))
SNPdata_file <- paste0(datapath, sprintf("SNPdata5000_lc_ECV_fold%d.rds", iloc))

top5000SNP <- readRDS(top5000_file)
## extract the snp names
names_snp <- top5000SNP[,1]

## get the SNP file
SNPdata <- readRDS(SNPdata_file)

training <- SNPdata[SNPdata$fold != iloc,]
testing <- SNPdata[SNPdata$fold == iloc,]


## nsnp_set <- c(1, 2, 4, 8, ..., 1024, 2048, 4096)
```

```
nsnp_set <- c(0, 13)
for ( i in 1:13){
  nsnp_set[i] <- 2^(i-1)
}
nsnp_set

Y_name <- "Phenotype_large"
X_cov <- "apoe"
X_chosen <- names_snp[1:nsnp_set[irep]]

fit_formula <- as.formula(paste0(Y_name, "~", paste(c(X_cov, X_chosen), collapse = "+")))
X_tr <- model.matrix(fit_formula, data = training, xlev = 3)[, -1]
Y_tr <- training[, Y_name]
X_ts <- model.matrix(fit_formula, data = testing, xlev = 3)[, -1]
Y_ts <- testing[, Y_name]

## htlr
htlr.pred <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.htlr <- matrix(0, nrow=length(Y_ts),ncol=3)
er.htlr <- vector()
amlp.htlr <- vector()
alpha1 <- c(0.5, 1, 1.5)
for (j in 1:3){
  htlr.fit <- htlr_fit (
    y_tr = Y_tr, X_tr = X_tr, X_ts = X_ts, stdzx = F, ## data
    pty = "t", alpha = alpha1[j], s = -10,   ## alpha = df and s= log (w)
    iters_h = 1000, iters_rmc = 1000, thin = 10, ## mcmc iteration settings,
    leap_L_h = 5, leap_L = 50, leap_step = 0.3, hmc_sgmcut = 0.3, ## hmc settings
    initial_state = "lasso", silence = !interactive())  ## initial state settings
  htlr.pred[,2*j-1] <- htlr.fit$probs_pred[,1]
  htlr.pred[,2*j] <- htlr.fit$probs_pred[,2]
  htlr.predeval <- evaluate_pred(htlr.fit$probs_pred, Y_ts+1, showplot=F)
  er.htlr[j] <- htlr.predeval$er
  pred.htlr[,j] <- ifelse(htlr.pred[,1] > 0.5, 0, 1)
  amlp.htlr[j] <- htlr.predeval$amlp
}

## lasso prediction
lasso.fit <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, 1)
las.pred <- lasso.fit$eval[["table_eval"]]
predmat.las <- las.pred[, 3:4]
er.las <- lasso.fit$eval[["er"]]
pred.las <- as.numeric(lasso.fit$predictor)
amlp.las <- lasso.fit$eval[["amlp"]]

## elastic net
alpha2 <- c(0.3,0.5,0.7)
predmat.ela <- matrix(0, nrow=length(Y_ts), ncol=6)
pred.ela <- matrix(0, nrow=length(Y_ts),ncol=3)
er.ela <- vector()
amlp.ela <- vector()
for (i in 1:3){
  elastic.net <- glmnet_fit(X_tr, Y_tr, X_ts, Y_ts, alpha2[i])
  ela.pred <- elastic.net$eval[["table_eval"]]
```

```
  predmat.ela[,2*i-1] <- unlist(ela.pred[, 3])
  predmat.ela[,2*i] <- unlist(ela.pred[, 4])
  er.ela[i] <- elastic.net$eval[["er"]]
  pred.ela[,i] <- as.numeric(elastic.net$predictor)
  amlp.ela[i] <- elastic.net$eval[["amlp"]]
}

saveRDS(cbind(pred.htlr, lasso = pred.las, pred.ela),
          file = paste0(predpath, sprintf("predictor_ECV_lc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(er.htlr), lasso = er.las, t(er.ela)),
        file = paste0(errorpath, sprintf("errorrate_ECV_lc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(t(amlp.htlr), lasso = amlp.las, t(amlp.ela)),
        file = paste0(amlppath, sprintf("amlp_ECV_lc_fold%d_nsnp%d.rds", iloc, nsnp_set[irep])))
saveRDS(cbind(htlr.pred, predmat.las, predmat.ela),
        file = paste0(predmatpath, sprintf("predmat_ECV_lc_fold%d_nsnp%d.rds",iloc, nsnp_set[irep])))

############### Evaluation ###############
if (!exists("irep")) irep <- 1

erpath <- "/home/med826/Mayo_simulate/10Sign/errorrate/"
amlppath <- "/home/med826/Mayo_simulate/10Sign/amlp/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"

nsnp_set <- c(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096)
nset <- length(nsnp_set)

locations <- paste0("fold",1:10)
model_set <- c("lc","sc")
modeltype <- model_set[irep]

## calculate the frequency of large coefficient Phenotype and small coefficient Phenotype
phenodata <- readRDS(paste0(datapath, modeltype, "_data.rds"))
prob_null <-prop.table(table(phenodata[,2]))
er_null <- min(prob_null)
amlp_null <- - sum(prob_null*log(prob_null))

## generate the array of error rate, which contians error rates of hltr, lasso, elastic, glm
## then calculate the mean error rate of all folds
errates <- array(0, dim = c(nset, 14, 11))
error_oracle <- rep(0,11)

for (iloc in 1:length(locations)){
  er_ecv_list <- paste0(erpath, "errorrate_ECV_", modeltype, "_", locations[iloc],
                        "_nsnp", nsnp_set, ".rds")
  er_icv_list <- paste0(erpath, "errorrate_ICV_", modeltype, "_", locations[iloc],
                        "_nsnp", nsnp_set, ".rds")
  errates[, 1:7, iloc] <- do.call('rbind', lapply(er_ecv_list, function(x) readRDS(x)[1:7]))
  errates[, 8:14, iloc] <- do.call('rbind', lapply(er_icv_list, function(x) readRDS(x)[1:7]))
  er_oracle <- paste0(erpath, "errorrate_oracle_", modeltype, "_", locations[iloc], ".rds")
  error_oracle[iloc] <- readRDS(er_oracle)[1]
}

for (i in 1:14){
```

```
  errates[,i,11] <- apply(errates[,i,1:10],1,mean)
}
error_oracle[11] <- mean(error_oracle[1:10])

pdf(paste0(evalpath, modeltype, "_er.pdf"), width = 9, height = 7)
par (mar = c(4,4,0.5,4))
matplot(0:12, errates[,c(2,4,7,9,11,12),11],
        ylim = c(0, 0.5),
        type = "b",
        col = rep(c(2,4,3),2),
        lwd = rep(3,6),
        pch = rep(c(1:3),2),
        cex = 1, lty = c(rep(1,3), rep(3,3)),
        xaxp=c(0,13,13),
        yaxp=c(0,0.5,10),
        xlab = "",
        ylab = "")
abline(h=error_oracle[11], lty=2, lwd=2)
abline(h=er_null, lty=4, lwd=2, col="grey58")
axis(side = 4, at=seq(0,er_null,by=er_null/10), labels=seq(1,0,by=-0.1))
mtext(side=4, line=3, expression(R^2),cex=1.5)
title (xlab = "log2(Number of SNPs)",
       ylab = "Error Rate",cex.lab=1.5, line =2.5)
legend("bottomleft", cex=0.8,
       legend = c("Hyper-LASSO", "LASSO", "Elastic Net",
                  "Oracle", "Null", "External CV", "Internal CV"),
       col = c(2,4,3, 1, "grey58", 1,1),
       pch = c(1:3,NA,NA, NA,NA),
       lty = c(NA,NA,NA,2,4,1,3))
dev.off()

amlp <- array(0, dim = c(nset, 14, 11))
amlp_oracle <- rep(0, 11)
for (iloc in 1:length(locations)){
  amlp_ecv_list <- paste0(amlppath, "amlp_ECV_", modeltype, "_", locations[iloc],
                          "_nsnp", nsnp_set, ".rds")
  amlp_icv_list <- paste0(amlppath, "amlp_ICV_", modeltype, "_", locations[iloc],
                          "_nsnp", nsnp_set, ".rds")
  amlp[, 1:7, iloc] <- do.call('rbind', lapply(amlp_ecv_list, function(x) readRDS(x)[1:7]))
  amlp[, 8:14, iloc] <- do.call('rbind', lapply(amlp_icv_list, function(x) readRDS(x)[1:7]))
  amlp_oracle_loc <- paste0(amlppath, "amlp_oracle_", modeltype, "_", locations[iloc], ".rds")
  amlp_oracle[iloc] <- readRDS(amlp_oracle_loc)[1]
}

for (i in 1:14){
  amlp[,i,11] <- apply(amlp[,i,1:10],1,mean)
}
amlp_oracle[11] <- mean(amlp_oracle[1:10])

pdf(paste0(evalpath, modeltype, "_amlp.pdf"), width = 9, height = 7)
par (mar = c(4,4,0.5,4))
matplot(0:12, amlp[,c(2,4,7,8,11,12),11],
        ylim = cbind(c(0, 0.7), c(0,1.7))[,irep],
        type = "b",
```

```
        col = c(rep(c(2,4,3),2),1,7),
        lwd = rep(3,6),
        pch = rep(c(1:3),2),
        cex = 1, lty = c(rep(1,3), rep(3,3)),
        xaxp=c(0,13,13),
        yaxp=rbind(c(0,0.7,14), c(0,1.7,34))[irep,],
        xlab = "",
        ylab = "")
abline(h=amlp_oracle[11],lty=2, lwd=2)
abline(h=amlp_null, lty=4, lwd=2, col = "grey58")
axis(side = 4, at=seq(0,amlp_null*2,by=amlp_null/5), labels=seq(1,-1,by=-0.2))
mtext(side=4, line=3, expression(R^2),cex=1.5)
title (xlab = "log2(Number of SNPs)",
       ylab = "AMLP", cex.lab = 1.5, line =2.5)
legend("bottomleft", cex=0.8,
       legend = c("Hyper-LASSO", "LASSO", "Elastic Net",
                  "Oracle", "Null", "External CV", "Internal CV"),
       col = c(2,4,3, 1, "grey58", 1,1),
       pch = c(1:3,NA,NA, NA,NA),
       lty = c(NA,NA,NA,2,4,1,3))
dev.off()

################ AUC ################
if (!exists("irep")) irep <- 1

library('pROC')

predmatpath <- "/home/med826/Mayo_simulate/10Sign/predmat/"
evalpath <- "/home/med826/Mayo_simulate/10Sign/evaluation/"
datapath <- "/home/med826/Mayo_simulate/10Sign/dataset/"

nsnp_set <- c(1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096)
nset <- length(nsnp_set)

locations <- paste0("fold",1:10)

## get the information of training dataset
model_set <- c("lc","sc")
modeltype <- model_set[irep]
phenodata <- readRDS(paste0(datapath, modeltype, "_data.rds"))

prob_null <- prop.table(table(phenodata[,2]))
pred_null <- rep(max(prob_null),2099)
auc_null <- roc(phenodata[,2], pred_null)$auc

auc_mat <- array(0, dim = c(nset, 6, 11))
auc_oracle <- rep(0,11)

for (iset in 1:nset){
  for (iloc in 1:length(locations)){
    pheno <- as.integer(phenodata[which(phenodata$fold == iloc),2])
    predictor_ECV <- paste0(predmatpath, "predmat_ECV_",modeltype, "_",
                            sprintf("fold%d_nsnp%d.rds",iloc, nsnp_set[iset]))
    predictor_ICV <- paste0(predmatpath, "predmat_ICV_",modeltype, "_",
```

```
                              sprintf("fold%d_nsnp%d.rds",iloc, nsnp_set[iset]))

    pred_mat_ECV <- do.call('cbind', lapply(predictor_ECV, readRDS))
    pred_mat_ICV <- do.call('cbind', lapply(predictor_ICV, readRDS))


    pred_frame <- cbind.data.frame(pred_mat_ECV[,c(4,8,14)], pred_mat_ICV[,c(4,8,10)], pheno)
    colnames(pred_frame) <- c("htlr_ECV", "lasso_ECV", "elasticnet_ECV",
                              "htlr_ICV", "lasso_ICV", "elasticnet_ICV", "pheno")

    auc_mat[iset, 1, iloc] <- roc(pred_frame$pheno, pred_frame$htlr_ECV)$auc
    auc_mat[iset, 2, iloc] <- roc(pred_frame$pheno, pred_frame$lasso_ECV)$auc
    auc_mat[iset, 3, iloc] <- roc(pred_frame$pheno, pred_frame$elasticnet_ECV)$auc
    auc_mat[iset, 4, iloc] <- roc(pred_frame$pheno, pred_frame$htlr_ICV)$auc
    auc_mat[iset, 5, iloc] <- roc(pred_frame$pheno, pred_frame$lasso_ICV)$auc
    auc_mat[iset, 6, iloc] <- roc(pred_frame$pheno, pred_frame$elasticnet_ICV)$auc


    predictor_oracle <- paste0(predmatpath, "predmat_oracle_", modeltype, "_", locations[iloc], ".rds")
    pred_oracle <- readRDS(predictor_oracle)[,2]
    auc_oracle[iloc] <- roc(pred_frame$pheno, pred_oracle)$auc
   }
}

for (i in 1:6){
  auc_mat[,i,11] <- apply(auc_mat[,i,1:10],1,mean)
}
auc_oracle[11] <- mean(auc_oracle[1:10])

pdf(paste0(evalpath, modeltype, "_auc.pdf"), width = 9, height = 7)
par (mar = c(4,4,0.5,4))
matplot(0:12, auc_mat[,1:6,11],
        ylim = c(0, 1),
        type = "b",
        col = rep(c(2,4,3),2),
        lwd = rep(3,6),
        pch = rep(c(1:3),2),
        cex = 1, lty = c(rep(1,3), rep(3,3)),
        xaxp=c(0,13,13),
        yaxp=c(0,1,10),
        xlab = "",
        ylab = "")
abline(h=auc_oracle[11], lty=2, lwd=2)
abline(h=auc_null, lty=4, lwd=2, col="grey58")
title (xlab = "log2(Number of SNPs)",
       ylab = "AUC",cex.lab=1.5, line =2.5)
legend("bottomleft", cex=0.8,
       legend = c("Hyper-LASSO", "LASSO", "Elastic Net",
                  "Oracle", "Null", "External CV", "Internal CV"),
       col = c(2,4,3, 1, "grey58", 1,1),
       pch = c(1:3,NA,NA, NA,NA),
       lty = c(NA,NA,NA,2,4,1,3))
dev.off()
```