Comparison of Statistical Testing and Predictive Analysis Methods for Feature Selection in Zero-inflated Microbiome Data

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

Xiaoying Wang

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

Background: Recent advances in next-generation sequencing (NGS) technology enable researchers to collect a large volume of microbiome data. Microbiome data consist of operational taxonomic unit (OTU) count data characterized by zero-inflation, over-dispersion, and grouping structure among the sample. Currently, statistical testing methods based on generalized linear mixed effect models (GLMM) are commonly performed to identify OTUs that are associated with a phenotype such as human diseases or plant traits. There are a number of limitations for statistical testing methods including these two: (1) the validity of p-value/q-value depends sensitively on the correctness of models, and (2) the statistical significance does not necessarily imply predictivity. Statistic testing methods depend on model correctness and attempt to select "marginally relevant" features, not the most predictive ones. Predictive analysis using methods such as LASSO is an alternative approach for feature selection. To the best of our knowledge, this approach has not been used widely for analyzing microbiome data.

Methodology: We use four synthetic datasets simulated from zero-inflated negative binomial distribution and a real human gut microbiome data to compare the feature selection performance of LASSO with the likelihood ratio test methods applied to GLMMs. We also investigate the performance of cross-validation in estimating the out-of-sample predictivity of selected features in zero-inflated data.

Results: Our studies with synthetic datasets show that the feature selection performance of LASSO is remarkably excellent in zero-inflated data and is comparable with the likelihood ratio test applied to the true data generating model. The feature selection performance of LASSO is better when the distributions of counts are more differentiated by the phenotype, which is a categorical variable in our synthetic datasets.

In addition, we performed LOOCV on the train set and out-of-sample prediction on the test set. The performance of the cross-validatory (CV) predictive measures are very close to the out-of-sample predictivity measures. This indicates that LOOCV predictive metrics provide honest measures of the predictivity of the features selected by LASSO. Therefore, the CV predictive measures are good guidance for choosing cutoffs (shrinkage parameter λ) in selecting features with LASSO. By contrast, when wrong models are fitted to a dataset, the differences between the q-values and the actual false discovery rates are huge; hence, their q-values are tremendously misleading for selecting features. Our comparison of LASSO and statistical testing methods (likelihood ratio test in our analysis) in the real dataset shows that small q-values do not necessarily imply high predictivity of the selected OTUs. However, the

researchers often use q-values to find the predictors. That is why we need to look at q-values carefully.

Conclusions: Statistical testing methods perform greatly in zero-inflated datasets on both synthetic and real data. However, a serious model checking should be conducted before we use q-values to choose features. Predictive analysis with LASSO is recommended to supplement q-values for selecting features and for measuring the predictivity of selected features.

Acknowledgements

My deepest gratitude goes first and foremost to Professor Longhai Li, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of the writing of this thesis. Without his consistent and illuminating instruction, this thesis could not have been done. Moreover, I am deeply indebted to my committee member, Professor Ebrahim Samei and Professor Juxin Liu for their valuable comments. I am also grateful to the Department of Mathematics and Statistics, all faculty, students and staff. Their lectures, seminars and support benefit me a lot.

Second, I would like to express my heartfelt gratitude to all Professors, postdocs and graduate students in P2IRC theme 3.3. My special thanks also go to Professor Cindy Feng, Dr. Jinhong Shi and Dr. Yan Yan, who have instructed and helped me a lot in the past two years.

Last my thanks would go to my parents for their understanding and support as they always do. Let me finish the study with confidence and courage.

Thanks for everyone and everything I have met.

Dedicated to my family.

Contents

List of Tables

List of Figures

List of Abbreviations

GLMM	Generalized Linear Mixed Model
ZINB	Zero-Inflated Negative Binomial
ZMP	Zero-Modified Poisson
NB	Negative Binomial
LASSO	Least Absolute Shrinkage and Selection Operator
LRT	Likelihood Ratio Test
FDR	False Discovery Rate
ROC	Receiver Operating Characteristic
FPR	False Positive Rate
TPR	True Positive Rate
LOOCV	Leave-One-Out Cross-Validation
ER	Error Rate
AMLP	the Average of Minus Log Predictive Probabilities

1. Introduction

1.1 Background and motivation

The microbiome comprises all of the genetic material within a microbiota (the entire collection of microorganisms in a specific niche, such as the human gut). This can also be referred to as the metagenome of the microbiota [1]. The far-reaching effects of the microbiome on human diseases and many other biological phenotypes have only recently discovered [2]. Bacteria in the body and on the surface have a significant impact on the development of health and disease states. For example, microbial changes are shown to be associated with Parkinson's disease [3]. The abundance of a bacterium species or genus is quantified by operational taxonomic unit (OTU) counts using genetic sequence similarity, produced via targeted amplification and sequencing of the 16S rRNA gene [4].

Microbiome OTU count data often have excessive zeros than what is expected in Poisson or NB models. There are three different types of zeros in Microbiome Data, which are outlier zeros, structural zeros and sampling zeros [5]. One source of the zero microbiota abundance is that only a few major bacterial taxa of the microbiota are shared across samples and the rest are detected only in a small percentage of the samples. The zero counts may also be observed when the counts are present with a low frequency but not observed because of sampling variation (sampling zeros). When OTU counts are non-zero, it is often observed that they are highly skewed to the right, often called over-dispersion. There are other statistical issues due to the study designs commonly used in microbiome studies. Microbiome studies usually collect samples with complicated grouping structure, for example, plants from the same plot, and individuals from the same family. The grouping structure in the sample causes correlation among the samples and thus further complicate the analysis and interpretation of microbiome count data. Ignoring the correlation among samples can result in biased inference and misleading results [6]. Generalized linear mixed effects models are often adopted to account for the grouping structure by treating the group identities as random factors [6, 7].

Sample/random variables represent sample collection identifiers in the hierarchical study design, such as family structure, repeated measures from multiple body sites or time points.

There is increasing interest among scientists to find the association between the abundances of a subset of OTUs and host factors, for example, health disorder, and plant traits; see [8–10]. For example, [11] reports various relationships between the gut microbiome and cancer, inflammatory bowel disease (IBD), and obesity. We will call this goal of data analysis by feature selection, that is, to select features related to a response variable. For microbiome data, the features are OTU variables, and the response is a phenotype variable such as disease indicator or plant traits. Currently, researchers fit each OTU variable with generalized mixed effect models given the phenotype variable and other factors and then apply a statistical testing method to each OTU to test whether the OTU is differentiated by the phenotype variable [6, 7]. Features selection can be achieved by thresholding the q-values returned by the statistical testing procedure. However, statistical testing methods have a number of limitations. First, the validity of q-values relies on the correctness of assumed models, which may not hold for real datasets. Second, q-values and p-values only measure statistical significance but not practical significance. Small q-values do not necessarily imply strong predictivity [12]. However, researchers still try to find the predictors with q-values method. For example, many SNPs selected by genome-wide association studies are not good predictors [12–14]. Third, the joint effects of features on a phenotype cannot be measured in statistical testing methods that look at each feature individually. Many phenotypes are believed to be related to multiple features [15–17].

1.2 Methodology

Feature selection can also be achieved by predictive analysis with statistical learning methods. In this thesis, we consider LASSO multinomial logistic regression [18]. LASSO applies the L_1 penalty to the coefficients of the logistic regression model of a phenotype variable given predictor variables derived from OTU measurements. The OTU variables with non-zero coefficients after the shrinkage will be selected. There also has been a tentative exploration of LASSO for microbiome data [19]. However, the study of LASSO for microbiome data is

still very limited. The reason is probably that many researchers think that LASSO logistic regression may have difficulty in handling the zero-inflation in microbiome data. This motivated us to investigate the performance of LASSO logistic regression in microbiome data. We conducted an empirical study using synthetic datasets and a real dataset to investigate the feature selection performance of statistical significance testing and LASSO methods. Four synthetic datasets were generated using zero-inflated negative binomial models (ZINB) with varying signal magnitudes on counts and zeros. We chose to generate datasets from ZINB models because many researchers in microbiome studies have adopted such models; see [6, 7]. We applied the likelihood ratio test (LRT) and LASSO logistic regression to select OTUs that are related to a fixed factor (a phenotype). The feature selection performance of LRT and LASSO are compared with the actual false discovery rate (FDR), which can be calculated in synthetic data with the true indicators of whether an OTU is associated with the phenotype. We also assessed the agreement of actual false discovery rates and the q-values which are estimates of FDR without using the true association indicators. For LASSO, we also assessed the agreement of cross-validatory predictive measures (error rate and average minus log probabilities) and the actual out-of-sample predictive measures which are obtained with the test sample. Out-of-sample forecast is the process of formally evaluating the predictive capabilities of the models developed using test data to see how effective the algorithms are in reproducing data.

1.3 Summary of results

Our studies with synthetic datasets show that the feature selection performance of LASSO is remarkably excellent in zero-inflated data, and is comparable with the likelihood ratio test applied to the true data generating model. The feature selection performance of LASSO is better when the distributions of counts are more differentiated with the phenotype. Besides, the cross-validatory (CV) predictive measures provide honest measures of the predictivity of features selected by LASSO. LOOCV can choose the best λ in selecting the most predictive features in microbiome data. Therefore, the CV predictive measures are proper guidance for choosing cutoffs (shrinkage parameter λ) in selecting features. It is reasonable that LASSO

performs slightly worse than likelihood ratio test applied to the true data generating model because LASSO model is not the true model for the dataset and LASSO will lose power even when the model is true for a dataset due to the shrinkage for controlling overfitting. By contrast, when wrong models are fitted to a dataset, the differences between the q-values and the actual false discovery rates are huge; hence, q-values are tremendously misleading for choosing cutoffs in selecting features. Our comparison of LASSO and statistical testing methods in the real dataset shows that small q-values do not necessarily imply high predictivity of selected OTUs.

1.4 Organization of the thesis

In Chapter 2, we introduce the feature selection methods by applying the likelihood ratio test to generalized linear mixed effect models (GLMMs), and by fitting LASSO logistic regression models. We also describe the metrics for comparing feature selection and prediction, including the false discovery rate (FDR), error rate and average minus log probabilities (AMLP). In Chapter 3, we describe how to simulate four datasets using zero-inflated negative binomial (ZINB) with varying signal magnitudes on counts and zeros and a real human gut microbiome data. In Chapter 4, we compare the feature selection performance of LASSO and LRT. We also investigate the performance of cross-validation in estimating the out-of-sample predictive accuracy of LASSO. The thesis is concluded in Chapter 5.

2. Methods

2.1 Feature selection in microbiome data

A typical OTU dataset contains measurements of abundance for OTUs, the total reads, a number of fixed factors and random factors for each sample, as shown by Table 2.1:

Table 2.1: A general form of microbiome Data

	OTU_1	OTU_2	$\cdots$	OTU_m	Total reads	Fixed factors	Random factors
Sample 1	$Y_1^{(1)}$	$Y_1^{(2)}$	$\cdots$	$Y_1^{(m)}$	T_1	$X_1^{(1)}, \cdots, X_1^{(s)}$	$Z_1^{(1)}, \cdots, X_1^{(t)}$
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
Sample n	$Y_n^{(1)}$	$Y_n^{(2)}$	$\cdots$	$Y_n^{(m)}$	T_n	$X_n^{(1)}, \cdots, X_n^{(s)}$	$Z_n^{(1)}, \cdots, Z_n^{(t)}$

We will describe the mathematical details of the notations in the Table 2.1:

- $Y_i^{(j)}$, $i = 1, 2, ..., n, j = 1, 2, ..., m$, is the count number of the OTU j in sample i. This number can be the abundance of taxa grouped at different levels such as species, genus, and family etc.

- $T_i, i = 1, 2, ..., n$ is the total sequence reads for sample i. If we all measured OTUs are included in our model, $T_i = \sum_{j=1}^{m} Y_i^{(j)}$. However, T_i may be smaller than $\sum_{j=1}^{m} Y_i^{(j)}$ if some OTUs are omitted for measurement quality control reason.

- $X_i^{(j)}$, $i = 1, 2, ..., n, j = 1, 2, ..., s$, represents the j^{th} fixed factor associated for the i^{th} sample. s is the total number of fixed factors considered in the mixed effect model. They could be host or clinical factors. When the jth fixed factor is categorical variable with k classes, $X_i^{(j)}$ is a row vector of $k-1$ binary variables indicating the class of sample i.

- $Z_i^{(j)}$, $i = 1, 2, ..., n, j = 1, 2, ..., t$, represents the j^{th} random factor associated for the i^{th} sample. t is the number of random factors considered in the mixed effect model. They are used to account for the correlation between samples since microbiota from the same

group of samples are more similar than the ones from different groups. Random factors are all categorical. Similar to $X_i^{(j)}$, the $Z_i^{(j)}$ is a row vector of binary indicator variables to represent the group identity of sample i in the jth random factor.

An important goal of microbiome studies is to identify/select a subset of OTUs that are associated/related with a fixed factor, say $X^{(1)}$, which represents a phenotype variable of interest in real microbiome data, for example, a variable indicating disease status or plant traits. Currently, researchers fit each OTU variable $Y^{(j)}$ with generalized linear mixed effect models (GLMM) [6] given all fixed and random factors, and then apply a statistical testing method to test whether the proportion of the jth OTU among all OTUs, often defined as $Y_i^{(j)}/T_i$, is differentiated (ie, associated) with $X^{(1)}$ [6, 7]. Selection of OTUs can be achieved by thresholding the q-values returned by the statistical testing procedure (likelihood ratio test). We will describe how to apply likelihood ratio test to GLMMs in Section 2.2 for selecting OTUs. Statistical learning is another important alternative approach to selecting OTUs. In LASSO, the $X^{(1)}$ is treated as a response/output variable, and the predictor/input variables are OTU variables $Y^{(1)}, \ldots, Y^{(m)}$ after certain transformation, and other fixed and random factors. Selection of OTUs can be achieved by looking at the coefficients associated with OTU variables. We will describe how to apply LASSO multinomial logistic regression for microbiome data in Section 2.3.2. The performances of these two distinct methods for feature selection will be compared with four synthetic data and a real gut microbiome dataset in Section 4.

2.2 Statistical testing methods

2.2.1 GLMMs for zero-inflated data

Generalized linear mixed model (GLMM) is a flexible modelling framework that can take into account both fixed effects and random effects into the modelling of a response variable. Generalized linear mixed models (GLMMs) are an extension of generalized linear models (GLMs) [20] and linear mixed models (LMMs) [21]. GLMMs are extended to include both fixed and random effects and use non-normal distribution to model a response. In this section,

we will describe three GLMMs which are often used to model count data with zero-inflation and over-dispersion. We will apply GLMMs to model each OTU variable $Y_i^{(j)}$ individually, conditional on fixed and random factors as shown in Table 2.1. Although different OTUs are theoretically dependent due to that they are summed to equal to T_i, we adopt this independent modelling for simplicity by considering that the correlations between OTUs are very small because the proportion of each OTU is very small when m is large. For simplicity in notations, we will omit the OTU index j in Section 2.2.

Negative binomial mixed effect models (NBMM):

For simplicity, we only describe one OTU here without superscript j. In NBMM, we will use negative binomial distribution to model Y_i given fixed and random factors. The PMF of Y_i is given by:

$$f^{\mathrm{NB}}(y_i;\mu_i,\theta)=\frac{\Gamma(y_i+\theta)}{\Gamma(\theta)\Gamma(y_i+1)}\left(\frac{\theta}{\theta+\mu_i}\right)^{\theta}\left(\frac{\mu_i}{\theta+\mu_i}\right)^{y_i}, \tag{2.1}$$

where y_i takes values in $\{0,1,\ldots\}$, $\mu_i>0$ is the mean of y_i, and $\theta>0$ is the inverse dispersion parameter. In NBMMs, the mean μ_i is linked to fixed factors and random factors as follows:

$$\log(\mu_i)=\log(T_i)+X_i\beta+Z_ib \tag{2.2}$$

where $X_i=(X_i^{(1)},\ldots,X_i^{(s)})$ is a vector for representing all fixed factors, and similarly $Z_i=(Z_i^{(1)},\ldots,Z_i^{(t)})$ is a vector of binary dummy variables for representing all random factors. The total reads T_i exhibit big differences across samples. Thus, the log of total reads (T_i) is also considered as a random factor and added to the link function with fixed coefficient 1. Such a variable is often called an offset variable. In other words, the equation (2.2) links the log of μ_i/T_i, i.e., the proportion of the abundance of an OTU among all OTUs, to fixed and random factors.

The NB distribution has heavier tails than the Poisson distribution. When $\theta\to\infty$, the NB distribution converges to Poisson distribution. Compared to other distributions such as Poisson or normal, the advantage of using the NB distribution with small parameter θ, such as 1 or 2, is that the heavier tails of NB can reduce the influence of extraordinarily large (over-dispersed) counts y_i in estimating the parameters in μ_i. Other characteristics of microbiome

data are the presence of many zeros. To address the zero-inflation in y_i, zero-inflated or zero-modified models have been adopted for modelling y_i.

Zero-modified poisson mixed effect model (ZMP):

In zero-modified models [22], also called hurdle models, the probability that $y_i = 0$ is modified be value ϕ_i different from what is postulated by a standard distribution such as the Poisson or NB distributions. The modified probability ϕ_i is often linked to fixed and random factors using logistic regression model. The non-zero values in y_i's, often called count values, are separately modelled by a zero-truncated distribution, for example, zero-truncated Poisson or NB distribution. Such distributions are often called zero-modified Poisson (ZMP) or zero-modified NB (ZMNB) models. In particular, the PMF of ZMP for y_i is written as:

$$f^{\text{ZMP}}(y_i) = \begin{cases} \phi_i & \text{for } y_i = 0 \\ (1-\phi_i)\dfrac{f^{\text{Pois}}(y_i;\mu_i)}{1-f^{\text{Pois}}(0;\mu_i)} & \text{for } y_i = 1, 2, \ldots \end{cases} \tag{2.3}$$

where $f^{\text{Pois}}(y_i;\mu_i)$ denotes the PMF of the Poisson distribution with mean μ_i. The ϕ_i and μ_i are often linked to fixed and random factors as follows:

$$\log\left(\frac{\phi_i}{1-\phi_i}\right) = W_i\gamma + V_i r \tag{2.4}$$

$$\log(\mu_i) = \log(T_i) + X_i\beta + Z_i b \tag{2.5}$$

where X_i, W_i, V_i, Z_i are covariates constructed from fixed and random factors. Note that the covariates used in ϕ_i and μ_i are not necessarily the same in theory.

Zero-inflated negative binomial mixed effect models (ZINB)

Another way to model zero-inflated count data is to model y_i as a mixture distribution of 0 and a standard distribution such as Poisson or NB distributions [23], instead of using zero-truncated distribution for non-zero y_i. They are often understood as that the y_i is generated in two steps. The first step is to generate a binary indicator from a logistic regression distribution. In the second step, if the indicator in the first step is zero, then the y_i is zero; otherwise, y_i is generated from a standard distribution such as NB. In particular, the PMF

of a zero-inflated negative binomial (ZINB) model for y_i can be written as:

$$f^{\text{ZINB}}(y_i) = \begin{cases} \phi_i + (1-\phi_i) f^{\text{NB}}(0; \mu_i, \theta), & \text{for } y_i = 0 \\ (1-\phi_i) f^{\text{NB}}(y_i; \mu_i, \theta), & \text{for } y_i = 1, 2, \ldots \end{cases} \tag{2.6}$$

where $0 < \phi_i < 1$ is the probability of an excess zero response. $f^{\text{NB}}(y_i; \mu_i, \theta)$ is the negative binomial distribution given by (2.1). The parameters μ_i and ϕ_i in ZINB are often linked to fixed and random factors using the same way as described in equations (2.4) and (2.5).

2.2.2 Likelihood ratio test

In this thesis, we applied likelihood ratio test (LRT) to test whether a fixed factor of interest, say $X^{(1)}$, is associated with each OUT variable $y_i^{(j)}$ for $j = 1, \ldots, m$ based on a GLMM model as described in Section 2.2.1. A vector of p-values $p^{(j)}, j = 1, \ldots, m$ will be returned from these tests, and will be converted into q-values $q^{(j)}, j = 1, \ldots, m$ to reflect false discovery rate, which will be introduced in Section 2.2.3. The OTUs with small false discovery rates (q-values) will be selected. In this section, we will briefly describe how to apply LRT to GLMM. For more detailed discussion of LRT for testing fixed effects of GLMMs can be found from [24] and the references therein. We will omit OTU index j for simplicity. The fixed factor of interest is denoted by $X^{(1)}$, and all other factors are collectively denoted by $X^{(2)}$. In LRT, we test the following two model assumptions:

$$\begin{aligned} H_0 : y_i \sim\ & f(y_i | X_i^{(2)}), \text{in words, } X_i^{(1)} \text{ has no effect on } y_i \\ H_1 : y_i \sim\ & f(y_i | X_i^{(1)}, X_i^{(2)}), \text{in words, } X_i^{(1)} \text{ has effect on } y_i \end{aligned} \tag{2.7}$$

Let L_0 and L_1 represent the maximized likelihoods under model H_0 and H_1 respectively. The log likelihood ratio statistic is defined as

$$T_n = 2(\log L_1 - \log L_0). \tag{2.8}$$

If the model H_1 is the true model for y_i, T_n is expected to be a large value. Therefore, larger T_n favours H_1 against H_0. A p-value can be calculated by computing $P(T_n > T_n^{(\text{obs})}) | H_0)$

where $T_n^{(\text{obs})}$ is the observed value of T_n given the dataset. By Wilk's theorem [25], the sampling distribution of T_n under H_0 is asymptotically a chi-square distribution with degree freedom α equal to the difference of the numbers of parameters in H_0 and H_1. Suppose the number of levels of $X_i^{(1)}$ is K, the degree freedom α is equal to $K-1$ in NBMM, and α is equal to $2(K-1)$ in ZMP and ZINB if $X_i^{(1)}$ is used to model both μ_i and ϕ_i.

2.2.3 False discovery rate and q-value

Due to the large number of OTUs, there is the multiple testing problem. Therefore, we need to convert p-values into false discovery rate to better understand the chance of false positive. Suppose we know the true relationship between OTUs and a phenotype, represented by a relevancy indicator $S^{(j)}$: $S^{(j)} = 1$ if the jth OTU is related to the fixed factor $X_i^{(1)}$ (called alternative hypothesis) and $S^{(j)} = 0$ if the jth OTU is unrelated to $X_i^{(1)}$ (called null hypothesis). Given p-values obtained with a statistical test method, we make feature selection by thresholding $p^{(j)}$ with a cutoff t:

$$\hat{S}^{(j)} = \begin{cases} 1 & \text{if } p^{(j)} \leq t, \\ 0 & \text{if } p^{(j)} > t. \end{cases} \tag{2.9}$$

The binary vector, $(\hat{S}^{(1)}, \ldots, \hat{S}^{(m)})$, is a result of feature selection based on $p^{(j)}$ at cutoff t. The goodness of $(\hat{S}^{(1)}, \ldots, \hat{S}^{(m)})$ can be summarized with four numbers: the number of true positives $TP(t) = \sum_{j=1}^{m} I(S^{(j)} = 1, \hat{S}^{(j)} = 1)$, the number of false positives $FP(t) = \sum_{j=1}^{m} I(S^{(j)} = 0, \hat{S}^{(j)} = 1)$, the number of true negatives $TN(t) = \sum_{j=1}^{m} I(S^{(j)} = 0, \hat{S}^{(j)} = 0)$, the number of false negatives $FN(t) = \sum_{j=1}^{m} I(S^{(j)} = 1, \hat{S}^{(j)} = 0))$, where $I(\cdot)$ is the indicator function. We also denote the total number of selected features (rejections of null hypotheses) by $R(t) = FP(t) + TP(t)$. In terms of p-values, the $FP(t)$ and $R(t)$ are written as follows:

$$\begin{aligned} FP(t) &= \#\{p^{(j)} \leq t \text{ and } S^{(j)} = 0; j = 1, ..., m\}, \\ R(t) &= \#\{p^{(j)} \leq t; j = 1, ..., m\}. \end{aligned} \tag{2.10}$$

Let m_0 be the number of OTUs that are unrelated to $X_i^{(1)}$, and $m_1 = m - m_0$ be the number of truly related OTUs. These numbers can be displayed with a confusion matrix (Table 2.2):

Table 2.2: A feature selection confusion matrix

		Predicted result		
		Class 1	Class 0	Total
True condition	Class 0	FP	TN	m_0
	Class 1	TP	FN	m_1
	Total	R	$m-R$	m

The actual false discovery rate (FDR) of $(\hat{S}^{(1)}, \ldots, \hat{S}^{(m)})$ is the proportion of false positives among the selected features:

$$\text{actual FDR}(t) = \frac{\text{Number of false positive features}}{\text{Number of selected features}} = \frac{FP(t)}{R(t)} \tag{2.11}$$

The $FP(t)$, $R(t)$, and actual FDR(t) in (2.11) are random variables. There are many definitions of a theoretical FDR. [26] defines the positive FDR for independent tests as follows:

$$\text{FDR}(t) = P(S^{(j)} = 0 | p^{(j)} \leq t) \approx \frac{E[FP(t)]}{E[R(t)]} \tag{2.12}$$

In practice, we do not know the true relevancy indictor $S^{(j)}$. We need to estimate FDR(t) with only observed p-values $p^{(j)}$ for $j = 1, \ldots, m$. The observed $R(t)$ is a reasonable estimate of $E[R(t)]$, which is the number of observed p-values $\leq t$. Suppose we know the proportion of unrelated features (null hypotheses) $\pi_0 = m_0/m$. Based on the theory that p-values are uniformly distributed under null hypothesis ($S^{(j)}$=0), we can obtain that $P(\hat{S}^{(j)} = 1 | S^{(j)} = 0) = P(p^{(j)} \leq t | S^{(j)} = 0) = t$. Then, we can see that $E[FP(t)] = m \cdot \pi_0 \cdot t$. A conservative estimate of π_0 is 1. This will give an upper bound of FDR(t) as given by [27]. Alternatively, [28] provides a more accurate estimate by finding a cutoff ζ such that the probability density of p-values are nearly uniform on the right of ζ; the value of this density is expected to be close to π_0. This argument leads to an estimate of π_0 as follows:

$$\hat{\pi}_0(\zeta) = \frac{\#\{p^{(j)} \geq \zeta; j = 1, ..., m\}/m}{1 - \zeta}. \tag{2.13}$$

Given an estimate $\hat{\pi}_0$ for π_0, FDR(t) is estimated by

$$\widehat{\text{FDR}}(t) = \frac{m \cdot \hat{\pi}_0 \cdot t}{R(t)}. \tag{2.14}$$

The q-value for a p-value $p^{(j)}$ is the FDR if we use $p^{(j)}$ as the cutoff t in feature selection, ie, features with p-values $\leq p^{(j)}$ are selected. To ensure theoretical monotonicity, $q(p^{(j)})$ is defined as the minimum of FDR(t) for $t \geq p^{(j)}$:

$$q(p^{(j)}) = \min_{t \geq p^{(j)}} \widehat{\text{FDR}}(t). \tag{2.15}$$

If we order all the p-values, and denote the jth p-value by $p^{[j]}$, an approximate for the q-value is $\hat{q}(p^{[j]}) = m \cdot \hat{\pi}_0 \cdot p^{[j]}/j$, which may not be monotone with $p^{[j]}$, but is easy to calculate and typically close to $q(p^{[j]})$ (2.15) when m is large; see a discussion of FDR by [29].

When the model for y_i is correctly specified, the q-values are good estimates of the actual FDRs, as explained above. In such cases, the q-value is useful guidance for determining the cutoff t in feature selection. However, in practice, the correctness of a model often lacks serious verification. Our empirical studies with synthetic datasets will show that there are huge gaps between q-values and actual FDRs when a wrong model (the model describing the data distribution wrongly) is used. This problem is particularly crucial in microbiome data analysis because OTU counts are difficult to model due to the clustering, skewness, and zero-inflation. The basic counts of OTU categories are relative values, which can not be regarded as absolute values. They depend upon the sampling depth corresponding to each sample.

2.3 Predictive analysis with LASSO

2.3.1 Transformation of OTU counts

Statistical learning is another important alternative approach to selecting OTUs. In statistical learning methods, the phenotype variable $X_i^{(1)}$ is treated as a response/output variable, and the predictor/input variables are OTU variables $Y_i^{(1)}, \ldots, Y_i^{(m)}$ with certain transformation, and other fixed and random factors. In microbiome data, it is often believed that the phenotype affects the composition of OTUs, rather than the raw counts of OTUs which are also affected by the total read T_i. As such, a reasonable transformation for OTU counts is the variance-stability transformation of the proportion of the counts of the jth OTU among

those of all OTUs:

$$\tilde{Y}_i^{(j)} = \arcsin\left(\sqrt{Y_i^{(j)}/T_i}\right), \text{ for } j = 1, \ldots, m$$

Other transformations can be investigated too. For example, if we believe that only the presence or not of certain OTUs is related to the phenotype, we transform $Y_i^{(j)}$ by $\tilde{Y}_i^{(j)} = I(Y_i^{(j)} > 0)$, although some information would be lost by dichotomizing the raw data. This binary transformation is useful for eliminating the adverse effect of the over-dispersion in OTU counts. The choices of transformations are guided by the resulting predictive accuracy in the test sample or using cross-validation.

To be consistent with conventional notations for statistical learning models, we will denote the phenotype variable $X_i^{(1)}$ by y_i. The y_i is a categorical variable taking values in $\{1, \ldots, K\}$, where K is the number of levels of $X_i^{(1)}$; this is different from that $X_i^{(1)}$ is represented by a $K-1$ binary indicator variables used as a fixed factor of GLMMs described in Section 2.2.1. The predictor variables are collectively denoted by x_i, which includes $\tilde{Y}_i^{(j)}$ and all other fixed and random factors (represented by binary indicator variables). The x_i will be a column vector in this section. The first value in x_i is "1" for including an intercept. After we fit a statistical learning model for y_i given x_i, selection of OTUs can be achieved by looking at the coefficients associated with the transformed OTU variables $\tilde{Y}_i^{(j)}$.

2.3.2 LASSO multinomial logistic regression

The least absolute shrinkage and selection operator (LASSO) [18] adds the L_1 of the regression coefficients as a penalty term to the log likelihood function to achieve shrinkage of regression coefficients for avoiding overfitting and for achieving feature selection. Suppose the response variable y_i has K levels, that is, y_i takes values in $\{1, 2, ..., K\}$. The multinomial logistic regression links the probability of $y_i = k$ to x_i using the soft-max function as follows:

$$P(y_i = k|x_i, \beta_1, \ldots, \beta_K) = \frac{\exp(\beta_k^T x_i)}{\sum_{k=1}^K \exp(\beta_k^T x_i)}, \text{ for } k = 1, \ldots, K, \tag{2.16}$$

where β_k is the collection of all regression coefficients related to $y_i = k$, which is a column vector of the same length of x_i. We will denote all these regression coefficients collectively

by β. Given observations $\{(y_i, x_i), i = 1, \ldots, n\}$, and a tuning parameter λ, the LASSO-penalized negative log likelihood function is

$$l_{\text{LASSO}}(\beta) = -\sum_{i=1}^{n} \log(P(y_i|x_i, \beta_0, \beta_1, \ldots, \beta_K)) + \lambda \sum_{k=1}^{K} ||\beta_k||_1, \tag{2.17}$$

where $||\beta_k||_1$ is the L_1 norm of β_k, equal to the sum of absolute values in β_k (except the intercept). The LASSO estimate of β is the minimizer of the LASSO-penalized negative log likelihood function:

$$\hat{\beta}_{\text{LASSO}} = \underset{\beta}{\text{argmin}}\, l_{\text{LASSO}}(\beta). \tag{2.18}$$

L_1 can shrink the coefficients associated with less important predictor variables into exactly zeros. The OTU variables with non-zero coefficients will be selected. We fit LASSO multinomial logistic regression (MLR) using an R package called glmnet [30]. This package can fit LASSO MLR efficiently for a series of different λ values. For each value of λ, we can obtain a binary feature selection vector as follows:

$$\hat{S}^{(j)}_{\text{LASSO}}(\lambda) = \begin{cases} 0, & \text{if } \beta_k^{(j)} = 0 \text{ for all } k \in \{1, \ldots, K\}, \\ 1, & \text{otherwise.} \end{cases} \tag{2.19}$$

where $\beta_k^{(j)}$ represents the coefficient associated with the jth OTU for modelling $P(y_i = k)$.

The degree of coefficient shrinkage, correspondingly the size of selected features, is controlled by the tuning parameter λ. Larger λ enforces greater shrinkage to the coefficients β, resulting in selecting fewer predictors in x_i. Therefore, the role of λ is similar to that of the cutoff t for thresholding p-values in statistical testing methods. As discussed in Section 2.2.3, one method for choosing t in statistical methods is to look at the false discovery rates or q-values. In a predictive analysis, a straightforward method for choosing λ is to look at the predictive performance in test samples, which is measured by a certain metric, for example, the error rate in predicting y_i. Cross-validation is a procedure to split the dataset into artificial training and test sets to obtain out-of-sample predictive metrics. We use the leave-one-out cross-validation (LOOCV). We fit the LASSO MLR model by holding the ith sample out. Then, we can use the fitted model to make predictions of y_i given x_i by comput-

ing the predictive probability using logistic regression model (2.16); let $\hat{P}_i^{\lambda,-i}(k|x_i)$ denote this CV predictive probability of $y_i = k$, for $k = 1, \ldots, K$. The goodness of $\hat{P}_i^{\lambda,-i}(k|x_i)$ for $i = 1, \ldots, n$ can be assessed with the actually observed $\{y_i; i = 1, \ldots, n\}$. We will discuss two predictivity metrics in Section 2.3.3.

2.3.3 Predictive metrics

Let $\hat{P}_i(k|x_i)$ be a predictive probability of $y_i = k$ for $k = 1, \ldots K$. We can assess the goodness of $\hat{P}(y_i|x_i)$ with actually observed $\{y_i; i = 1, \ldots, n\}$. The first metric is **error rate**. We predict y_i by the k with the highest probability: $\hat{y}_i = \text{argmax}_k \hat{P}_i(k|x_i)$. The error rate is defined as the proportion of wrongly predicted cases:

$$ER = \frac{1}{n} \sum_{i=1}^{n} I(\hat{y}_i \neq y_i). \tag{2.20}$$

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

$$AMLP = -\frac{1}{n} \sum_{i=1}^{n} \log(\hat{P}_i(y_i|x_i)). \tag{2.21}$$

$AMLP$ is more sensitive than ER because it 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 of $AMLP$ and ER should be interpreted relatively not absolutely. Let f_k be the observed frequency of $y_i = k$ for $k = 1, \ldots, K$. Without including any predictor in x_i (call the null model), the naive predictive probabilities are $\hat{P}_i^{(0)}(k) = f_k$ for $k = 1, \ldots, K$. The actual CV predictive probabilities will estimate f_k with the y_i removed, but the frequency without considering y_i is very close to f_k. Based on these naive predictive probabilities, the point prediction is that $\hat{y}_i^{(0)} = \text{argmax}_k f_k$. for $i = 1, \ldots, n$. That is, we will predict all the y_i's by the mode of $\{y_i; i = 1, \ldots, n\}$. The ER with $\hat{P}_i^{(0)}(k)$ is

$$ER^{(0)} = 1 - \max\{f_k; k = 1, \ldots, K\}, \tag{2.22}$$

which we will call the baseline error rate. A dataset with more unbalanced distribution in y_i is obviously easier to predict. For example, if $\max_k f_k = 0.9$, the baseline error rate is 0.1. For such an easy dataset, an error rate of 0.1 means that the predictivity of x_i for y_i is low. By contrast, if the maximum frequency is 0.6, then the baseline error rate is 0.4. For such a difficult dataset, an error rate of 0.1 indicates that y_i can be fairly well predicted by x_i. Similar to R^2 used in linear regression, a relative predictivity metric based on ER is defined as the percentage of the reduction of ER from $ER^{(0)}$:

$$R^2_{\text{ER}} = \frac{ER^{(0)} - ER}{ER^{(0)}}. \tag{2.23}$$

Similarly, plugging $\hat{P}_i^{(0)}(k) = f_k$ for all samples into (2.21), we will see that nf_k of terms in the summation of (2.21) are equal to $\log(f_k)$. Then, we can obtain the baseline $AMLP^{(0)}$:

$$AMLP^{(0)} = -\sum_{k=1}^{K} f_k \log(f_k). \tag{2.24}$$

Note that $AMLP^{(0)}$ is the entropy of $\{f_k; k = 1, \ldots, K\}$, a quantity for measuring the uncertainty of f_k's. A relative predictivity metric based on $AMLP$ is defined as the percentage of the reduction of $AMLP$ from $AMLP^{(0)}$:

$$R^2_{\text{AMLP}} = \frac{AMLP^{(0)} - AMLP}{AMLP^{(0)}}. \tag{2.25}$$

3. Data

3.1 Synthetic datasets from ZINB

We generated four synthetic datasets with $n = 5000$ samples, $m = 3000$ OTUs, and 5 categorical factors from ZINB models with different signal level settings. In ZINB models, the response variable is modelled as a mixture of structural zeros and an NB distribution. The process for generating one dataset can be described as follows:

Step 1: Generate 5 factors, denoted by $X_i^{(f)}$, for $f = 1, \cdots, 5$, which are used as covariates for generating OTU counts. Each of the five factors is generated uniformly from integers $1, \cdots, l_f$, where l_f represents the number of levels of $X^{(f)}$. In particular, the factor $X^{(1)}$ has 10 levels, which will be used as a phenotype variable. The total read T_i is generated from Poisson distribution with mean $\lambda = 30$.

Step 2: A signal vector $(S^{(1)}, \ldots, S^{(m)})$ is generated from Bernoulli distribution with parameter $p = 0.03$. $S^{(j)} = 0$ indicates that the jth OTU is related to the factor $X^{(1)}$, otherwise unrelated.

Step 3: For $i = 1, ..., n$ and $j = 1, ..., m$, generate one OTU count $Y_i^{(j)}$ for the ith sample and jth OTU as follows.

Step 3.1: Generate a binary variable $H_i^{(j)}$ to indicate that $Y_i^{(j)}$ is a structural zero or not. The probability of $H_i^{(j)} = 0$, denoted by $\phi_i^{(j)}$, is linked to the five factors using logistic link function:

$$\log\left(\frac{\phi_i^{(j)}}{1-\phi_i^{(j)}}\right) = \gamma_0 + X_i^{(1)}\gamma_j^{(1)} + X_i^{(2)}\gamma_j^{(2)} + X_i^{(3)}\gamma_j^{(3)} + X_i^{(4)}\gamma_j^{(4)} + X_i^{(5)}\gamma_j^{(5)}$$

where, γ_0 is an intercept; the coefficient vector $\gamma_j^{(f)}$ for the factor $X^{(f)}$, for $f = 2, \cdots, 5$, are generated from a normal distribution $\mathrm{N}(0, 2^2)$; the

coefficient vector $\gamma_j^{(1)}$ for the phenotype factor $X^{(1)}$ is generated as follows:

$$\gamma_j^{(1)} \sim \begin{cases} 0, & \text{if } S^{(j)} = 0 \\ N(0, \sigma_{zero}^2) & \text{if } S^{(j)} = 1 \end{cases}$$

Step 3.2: If $H_i^{(j)} = 0$, $Y_i^{(j)} = 0$; otherwise, $Y_i^{(j)}$ is generated from a NB distribution with $\theta = 2$ and mean $\mu_i^{(j)}$, which is linked to the five factors as follows:

$$\log(\mu_i^{(j)}) = \log(T_i) + \beta_0 + X_i^{(1)}\beta_j^{(1)} + X_i^{(2)}\beta_j^{(2)} + X_i^{(3)}\beta_j^{(3)} + X_i^{(4)}\beta_j^{(4)} + X_i^{(5)}\beta_j^{(5)}$$

where, β_0 is an intercept; the coefficient vector $\beta_j^{(f)}$ for the factor $X^{(f)}$, for $f = 2, \cdots, 5$, are generated from a normal distribution $\mathrm{N}(0, 0.2^2)$; the vector $\beta_j^{(1)}$ for the phenotype factor $X^{(1)}$ is generated as follows:

$$\beta_j^{(1)} \sim \begin{cases} 0, & \text{if } S^{(j)} = 0 \\ N(0, \sigma_{count}^2) & \text{if } S^{(j)} = 1 \end{cases}$$

The parameter σ_{zero} controls the degree that the status of the jth OTU being zero or not is affected by the phenotype factor $X^{(1)}$. The parameter σ_{count} controls the degree that the magnitude of the count of the jth OTU is affected by the phenotype factor $X^{(1)}$. We will refer to σ_{zero} and σ_{count} respectively as **zero signal level** and **count signal level**. These two parameters are varied at two different levels, resulting in four combinations of zero and count signal levels. Their values are displayed in Table 3.1. Other parameter settings for the four datasets are the same.

Table 3.1: Signal level settings for four synthetic datasets.

Dataset ID	σ_{count}	σ_{zero}	Description
1	1	1	large count signal, large zero signal
2	1	0.0001	large count signal, small zero signal
3	0.0001	1	small count signal, large zero signal
4	0.0001	0.0001	small count signal, small zero signal

3.2 A human gut microbiome dataset

We applied the methods described in this thesis to a dataset reported by [10]. The study cohort contained $1,561$ subjects from 2008 to 2014. Fecal samples were collected with 16S ribosomal DNA sequencing. Their fecal were collected using the stool commode provided by Fisher Scientific. They took an aliquot of stool with a polypropylene specimen collection container. Stool samples were kept frozen until bacterial DNA was extracted. The fecal bacterial DNA was extracted using the QIAamp DNA Stool Mini kit according to the protocol with minor modifications, including physically disrupting the bacterial cell wall with $0.1-mm$ glass beads. Then, non-chimeric sequences were grouped into OTUs at a minimum depth of $30,000$ reads/sample with a sequence identity threshold of 97%. They removed the samples with fewer than $30,000$ reads and OTUs with a prevalence of $< 5\%$ as quality control. The final data consists of $n = 1,098$ subjects.

Firmicutes (relative abundance of $64.4 \pm 13.9\%$), Bacteroidetes ($26.7 \pm 14.8\%$), and Actinobacteria ($5.0 \pm 5.0\%$) were the three major bacterial phyla. Blautia, Coprococcus, Ruminococcus, Bacteroides, Dorea, Roseburia, Faecalibacterium, Streptococcus, and Oscillospira are found across subjects in the identified 127 genera. The remaining 118 genera were observed irregularly in all subjects.

The study cohort ($n = 1,098$) consisted of first-degree relatives of Crohn's disease patients with asymptomatic self-described white. The average age of the subjects was 20.1 ± 7.8 years (mean $\pm$ s.d.; range of $6 - 35$ years). 54.6% were female. We treat the "ethnicity" and "sex" as fixed effects, "age" as random effects because there are too many levels. We are interested in selecting OTU variables that can predict the ethnicity and quantifying the predictivity of the selected OTUs. The purpose of this analysis is to illustrate the difference between statistical testing and LASSO-MLR methods in selecting features.

4. Results

4.1 Results of analyzing synthetic datasets

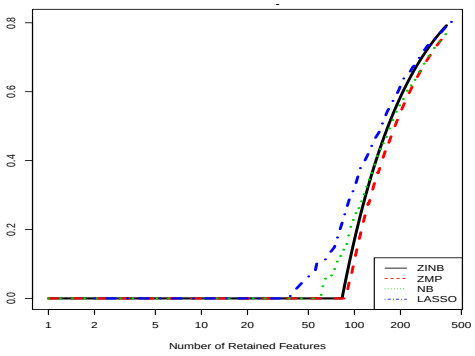
In this section, we will investigate the LRT methods applied to three GLMMs (ZINB, ZMP, NB) and LASSO-MLR methods, on the four synthetic datasets generated from a ZINB model, as described in Section 3.1. We split each dataset into a training dataset with $n_{train} = 300$ subjects and a test dataset with the remaining $n_{test} = 4700$ subjects. Only the training dataset is used to train the GLMMs and LASSO-MLR models. The test dataset is used to check the predictive performance of LASSO-MLR.

In fitting the GLMMs, we treat $X^{(1)}$, $X^{(2)}$ as fixed factors, and the remaining three factors as random factors. We fit the GLMMs using the R package `glmmTMB` [31]. In LRT tests, we use $X^{(1)}$ as a phenotype variable, that is, to test whether $X^{(1)}$ has an effect on each OTU. A vector of $m = 3000$ p-values are calculated for each model and each dataset. We employ the package `glmnet` [32] to fit LASSO-MLR models, which use $X^{(1)}$ as a response variable, and use the remaining four factors and all 3000 transformed OTU variables as predictors. For the dataset 1 and 2, we transform the OTUs with $\arcsin(\sqrt{Y_{ij}/T_i})$. For the dataset 3 and 4, we transform the OTUs with $I(Y_i^{(j)} > 0)$ since the magnitudes of the OTU counts are nearly unrelated to the phenotype $X^{(1)}$. For real data analysis, the choice of such a transformation is guided by cross-validatory or out-of-sample predictive metrics.

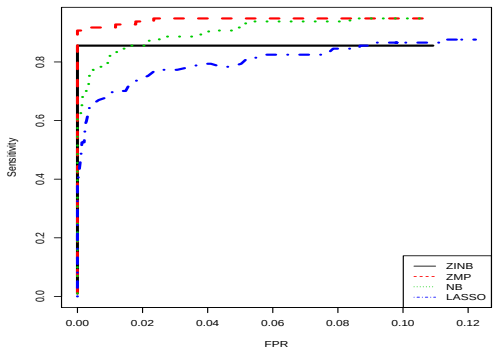
For feature selection with LRT p-values, we use all the means between two adjacent ordered p-values as cutoffs. For each cutoff t, we can obtain a feature selection vector $\{\hat{S}^{(j)}; j = 1, \ldots, m\}$ by selecting features with p-values $\leq t$. For the synthetic datasets, the true relevancy indicators $\{S^{(j)}; j = 1, \ldots, m\}$ are known. Therefore, we can compute $FP(t)$, $TP(t)$, $R(t)$, and the actual FDR(t) as described in Section 2.2.3. The left panel of Figure 4.1 shows the actual FDR(t) against $R(t)$ for each of the three GLMMs; the right panel of Figure 4.1 shows the ROC curves, ie, $TP(t)/(m - m_0)$ against $FP(t)/m_0$. We fit LASSO-MLR models using `glmnet` for 100 values of λ chosen in a reasonable range. For each

Figure 4.1: Comparison of the actual FDRs of LRT and ROC curves between statistical testing and LASSO under four different zero and count signal situations.

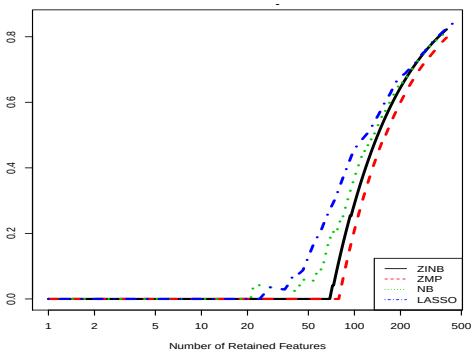
(a) Dataset 1, Actual FDR



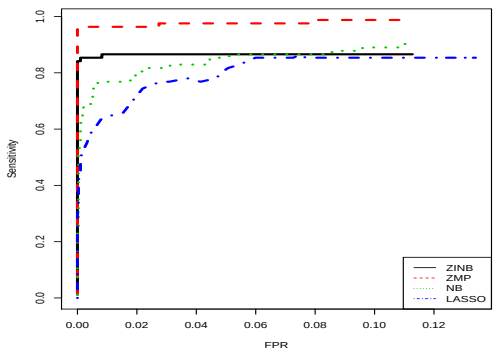
(b) Dataset 1, ROC



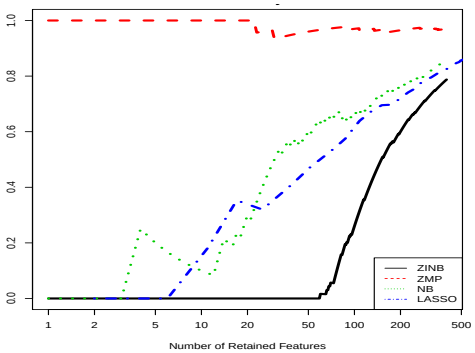
(c) Dataset 2, Actual FDR



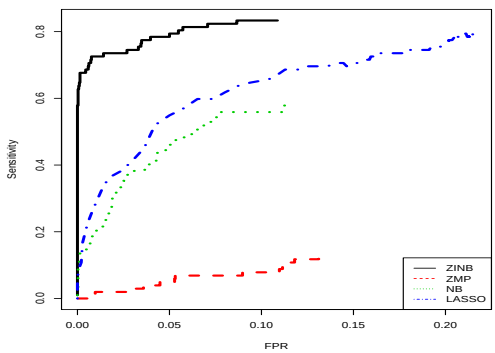
(d) Dataset 2, ROC



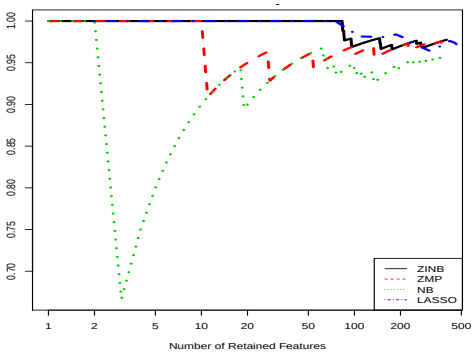
(e) Dataset 3, Actual FDR



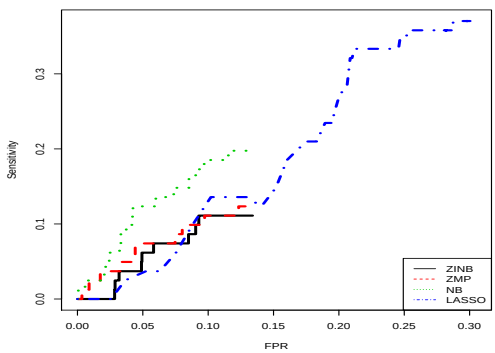
(f) Dataset 3, ROC



(g) Dataset 4, Actual FDR



(h) Dataset 4, ROC



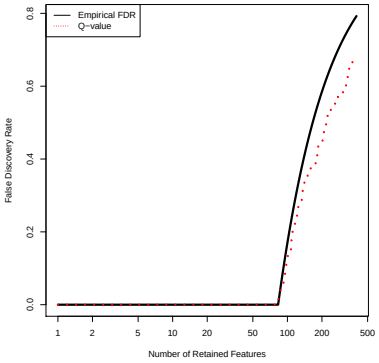
value of λ, we can use the estimates of the regression coefficients $\hat{\beta}_{\text{LASSO}}$ to select features as described in Section 2.3.2, that is, the features with at least one non-zero coefficient are retained. For each λ, we can compute the number of false positives $FP(\lambda)$, the number of true positives $TP(\lambda)$, the number of retained features $R(\lambda)$, and the actual FDR(λ) using the same method described in Section 2.2.3. The plots of FDR(λ) against $R(\lambda)$ and the ROC curves of $TP(\lambda)/(m-m_0)$ against $FP(\lambda)/m_0$ are shown in Figure 4.1 to compare with those of the LRT methods. From Figure 4.1, we see that, for dataset 1 and 2, LASSO-MLR has comparable FDRs and areas under the ROC curves with the statistical testing methods; all of the four methods perform similarly; the reason that the LRT tests applied to ZMP appear a little bit better the LRT tests applied to the true model ZINB is that the fittings of ZINB for a few OTUs that are truly related to $X^{(1)}$ failed to converge. Therefore, The number of not converging OTUs that are truly related to $X^{(1)}$ in ZINB is less than ZMP model. Hence, these features were not counted as being selected. For dataset 3, the LRT tests applied to ZINB are better than the other three methods. For dataset 4, all of the four methods do not work well because both of the count and zero signal levels are very small. In summary, the feature selection performances of the LASSO-MLR methods are comparable to those of the statistical testing methods in the synthetic datasets with pretty large count signal levels.

The feature selection performance shown in Figure 4.1 only indicates the goodness of the orderings of the features in light of their relationships with the phenotype variable. In practice, we also require a trustable method to determine the threshold (the cutoff t for p-values and the shrinkage parameter λ for LASSO) for retaining a feature subset. For statistical testing methods, the choice of t is often guided by the q-values, and for LASSO the choice of λ is guided by the predictive metrics, for example choosing the λ giving the best predictive metrics. We will discuss these two issues.

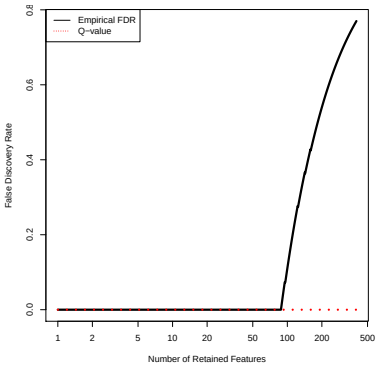
Calculating q-value is a method for estimating actual FDRs given only p-values. The methods for computing q-values are reviewed in Section 2.2.3. We use the R function `p.adjust` from the package `stats` to convert p-values into q-values. `p.adjust` has a few options for computing q-values. We chose the conservative method called "BH" [27]. As discussed in Section 2.2.3, the BH method estimates the proportion of unrelated OTUs by 1. Therefore, the BH method is expected to slightly over-estimate the true FDRs. Figure

Figure 4.2: Comparison of actual FDRs and q-values for the likelihood ratio tests.

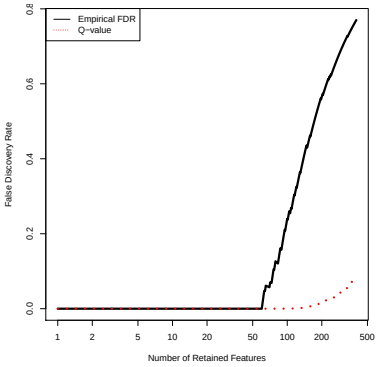
(a) Dataset 1, ZINB



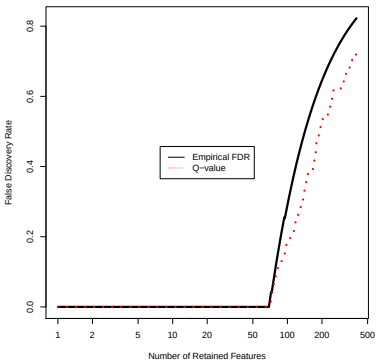
(b) Dataset 1, ZMP



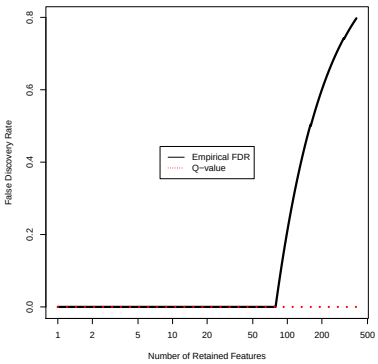
(c) Dataset 1, NB



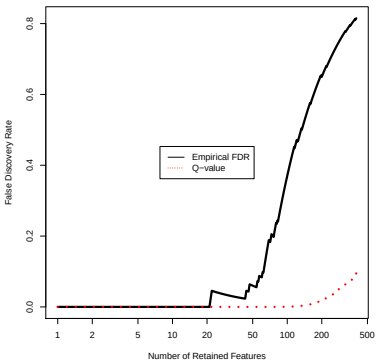
(d) Dataset 2, ZINB



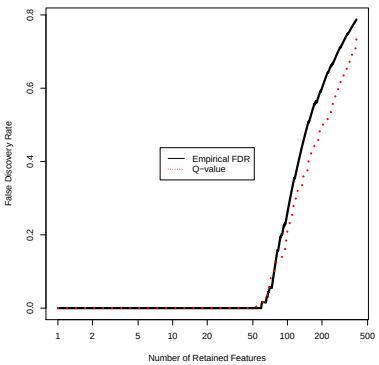
(e) Dataset 2, ZMP



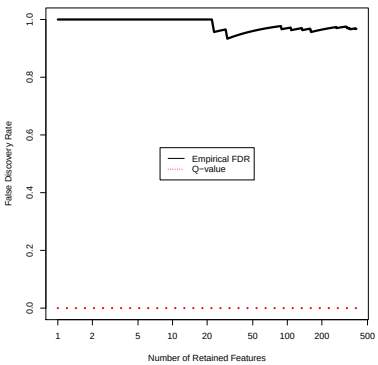
(f) Dataset 2, NB



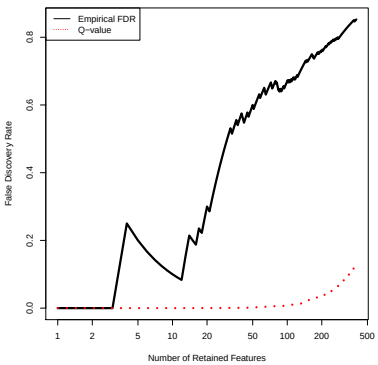
(g) Dataset 3, ZINB



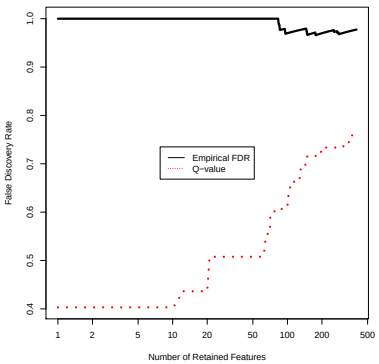
(h) Dataset 3, ZMP



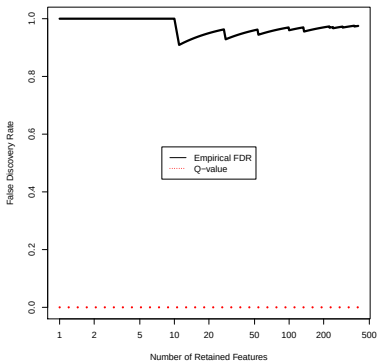
(i) Dataset 3, NB



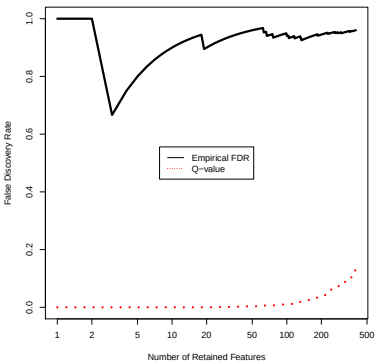
(j) Dataset 4, ZINB



(k) Dataset 4, ZMP



(l) Dataset 4, NB



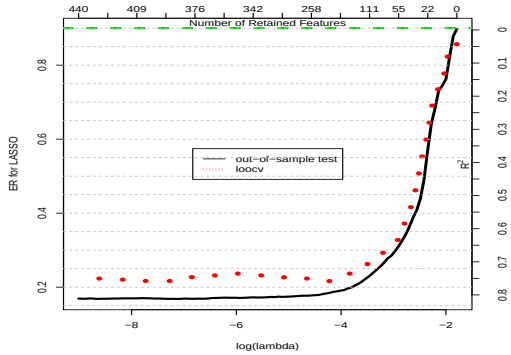
4.2 displays the actual FDRs and q-values against the number of selected features in four datasets. We can see that when the true model (ZINB) is used, the q-values are very close to the actual FDRs. However, when the wrong models (ZMP, NB) are fitted to the datasets, there are huge gaps between the actual FDRs and q-values, and typically the q-values are greatly smaller than the actual FDRs. In particular, for dataset 4 in which the relationship between OTUs and the phenotype is very weak, the q-values based on ZMP and NB models are nearly 0 whereas the actual FDRs are nearly 1. These results show that when wrong models are chosen, the resulting q-values are tremendously misleading for choosing the p-value cutoff and the actual FDRs are greatly underestimated.

For LASSO-MLR, the choice of λ is guided by predictive metrics. In practice, we can use LOOCV with only the training dataset to estimate the predictive metrics, which is described in Section 2.3.2. For assessing the performance of LOOCV, we compute the actual predictive metrics using the test dataset. Figure 4.3 displays the AMLPs, and the ERs obtained with LOOCV and the test dataset. We see a close matching of the LOOCV and actual out-of-sample predictive metrics. This indicates that LOOCV predictive metrics provide honest measures of the predictivity of the features selected by LASSO. Therefore, the LOOCV predictive metrics are good guidance for choosing cutoffs (shrinkage parameter λ) in selecting features.

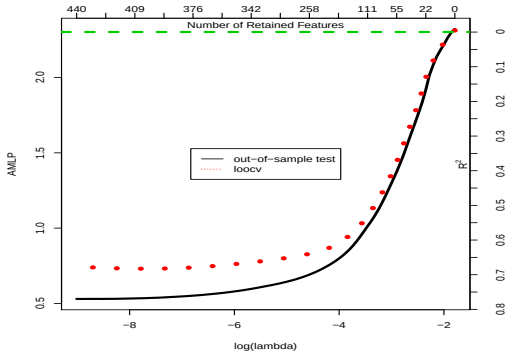
In addition to guide the choice of λ, the predictive metrics are also good indicators of the difficulty level in predicting the phenotype with selected OTUs. To demonstrate this, we show the baseline ERs and AMLPs using green lines in the plots of Figure 4.3, and display the R^2, the percentage of the reduction of ER or AMLP from those in the null model with no predictor, on the right y-axes. The R^2 is a good indicator of the predictivity of selected features for the phenotype. For example, the maximum R^2 values using the optimal λ for dataset 1 and 2 are larger than those for dataset 3 and 4. Such relative predictive metrics indicate that the phenotype in dataset 1 and 2 is more predictable than the phenotype in dataset 3 and 4. In particular, the R^2 for dataset 4 is near zero, which indicates that there are very weak relationships between the OTUs and the phenotype. This is the true situation in our data generating process; see Section 3.1.

Figure 4.3: The LOOCV and the out-of-sample predictive metrics of LASSO.

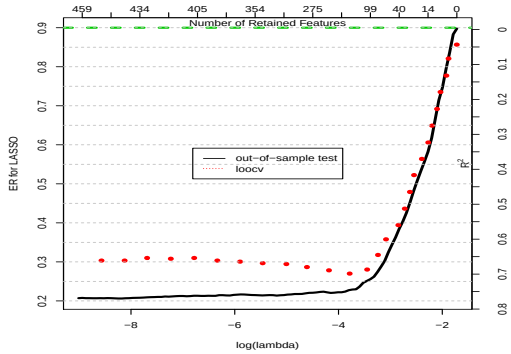
(a) Dataset 1, ER



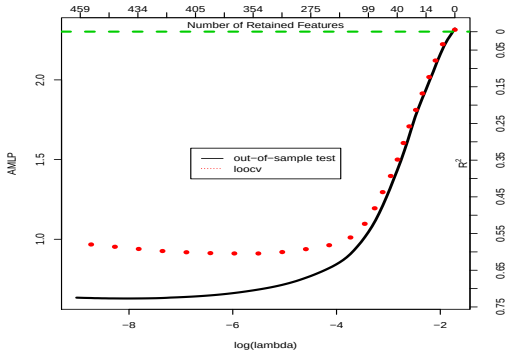
(b) Dataset 1, AMLP



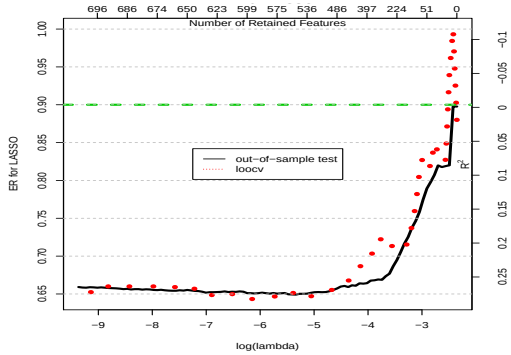
(c) Dataset 2, ER



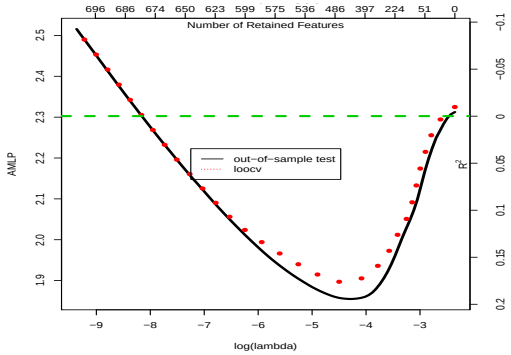
(d) Dataset 2, AMLP



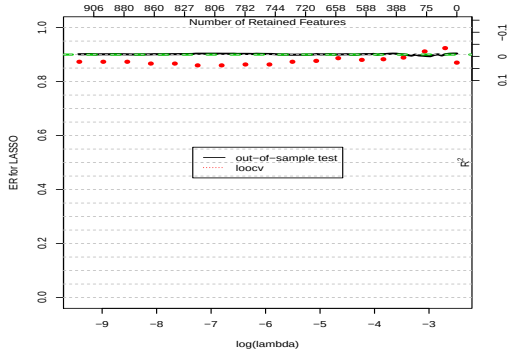
(e) Dataset 3, ER



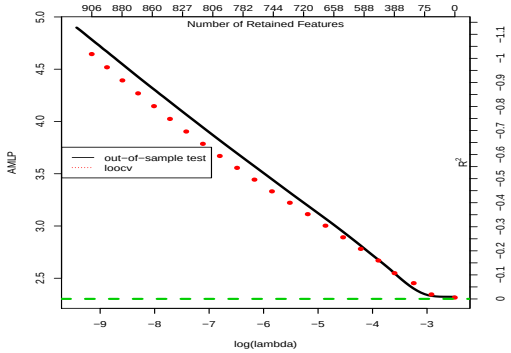
(f) Dataset 3, AMLP



(g) Dataset 4, ER



(h) Dataset 4, AMLP



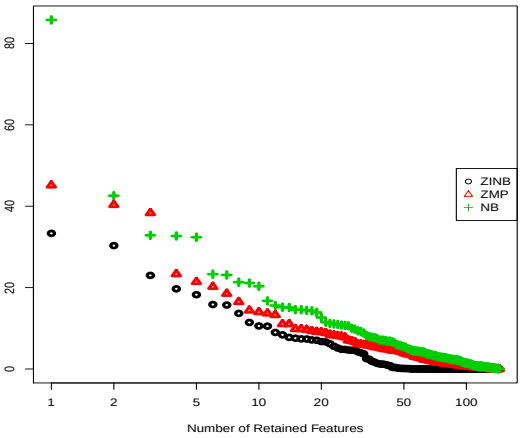
4.2 Results of analyzing the gut microbiome data

In this section, we apply the LRT testing methods based on three GLMMs and the LASSO-MLR method to the human gut microbiome dataset released by [10]. The description of the dataset is summarized in Section 3.2. The sample size n is 1098. We chose to select OTUs that can predict the variable ethnicity. The ethnicity has five levels: JewAsh, JewOth, JewSep, JewUnk and White, with frequencies being $0.096, 0.015, 0.042, 0.011$ and 0.836.

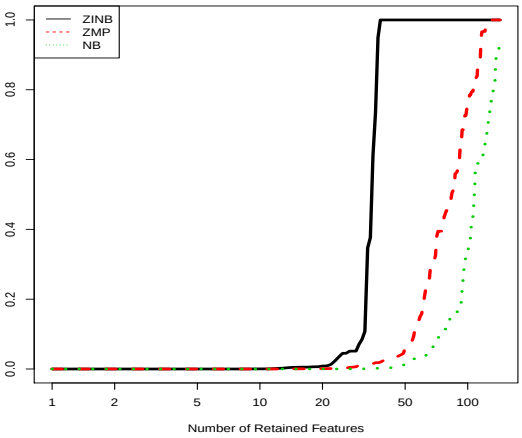
There are a total of $m = 144$ OTUs at genus level. We fit three GLMM models (ZINB, ZMP, and NB) to each OTU (feature) by considering three covariates, "ethnicity", "sex" and "age". We conduct the likelihood ratio test to each OTU to see whether it is affected by the ethnicity. The q-values of these three models are shown in Figure 4.4b. We see that the FDR for the top 20 OTUs is nearly 0. However, the small FDR only indicates that they are related to the ethnicity with a great chance, but does not indicate the strength of the association. The small FDR results from the small p-values, which are probably due to the large sample size ($n = 1098$), rather than due to the sharp difference of the proportions of these OTUs across the five ethnic groups. The predictive analysis looks at how much the ethnicity can be predicted by the OTUs, hence, can quantify the degree of the differentiation of the proportions of the top OTUs across the five ethnic groups. We conducted this analysis by applying LASSO-MLR to this dataset using LOOCV. Figures 4.4c and 4.4d shows the error rates and AMLPs for 100 values of λ. The best predictive metrics are reached by a subset of about 120 (out of 144) OTUs. However, the optimal predictive metrics are very close to the baseline error rates and AMLPs as shown by the green lines. The R^2 for the optimal error rates and AMLPs are about 20.99% and 16.37% respectively, which indicates that the predictivity of the 120 OTUs for the ethnicity is pretty low. In summary, the proportions of the top 120 OTUs are indeed differentiated across the five ethnic groups, but the differences are very small. This example shows that the selected features with small q-values may not be highly predictive to a phenotype, or maybe there are other factors we have to measure.

Figure 4.4: Results of analyzing a gut microbiome data.

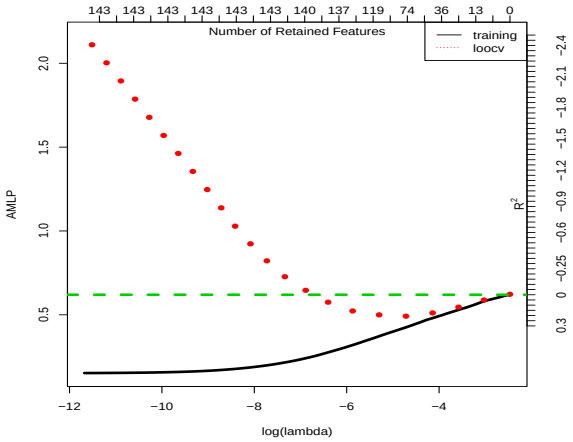
(a) -log(p-values) of three GLMMs



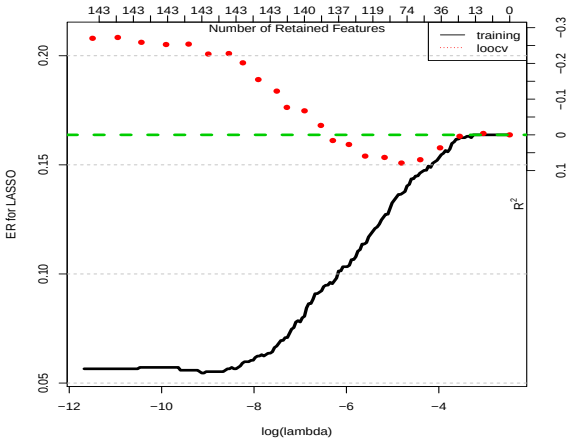
(b) q-values of three GLMMs



(c) AMLP for LASSO



(d) ER for LASSO



5. Conclusions

In this thesis, we conducted empirical studies using synthetic datasets and a real dataset to investigate the feature selection performance of statistical testing and LASSO methods for zero-inflated microbiome data. Our studies with synthetic datasets show that LASSO is comparable with the likelihood ratio test applied to the true data generating model. It is reasonable that LASSO performs slightly worse than the likelihood ratio test applied to the true data generating model since the multinomial logistic regression model is not the true model for such datasets. However, the performance of LASSO is still remarkable especially when the signals on counts are large enough. The cross-validatory predictive metrics for LASSO provide honest measures of the predictability of the response variable by the features. Therefore, they are useful for choosing reasonable cutoffs in feature selection and are useful to indicate the predictivity of the selected features. On the other hand, for statistical testing methods, when the model is not correctly specified, the q-values (estimated FDR) greatly underestimate the actual false discovery rate. In such cases, the q-values are tremendously misleading for choosing reasonable cutoffs in selecting features. Furthermore, our studies of these two methods in a real dataset show that small q-values do not necessarily imply high predictivity of selected OTUs. In conclusion, if the model is correctly specified, statistical testing methods perform well for selecting features in microbiome data, otherwise incorrect conclusions is likely to be drawn. Therefore, a serious model checking is required when using statistical testing methods. Unfortunately, model checking is rarely conducted in real data analysis. Predictive analysis with LASSO is recommended to supplement statistical testing methods for selecting features and for measuring the predictivity of the selected features.

Bibliography

[1] Samantha A Whiteside, Hassan Razvi, Sumit Dave, Gregor Reid, and Jeremy P Burton. The microbiome of the urinary tract—a role beyond infection. *Nature Reviews Urology*, 12:81, jan 2015. URL https://doi.org/10.1038/nrurol.2014.361http://10.0.4.14/nrurol.2014.361.

[2] Rob Knight, Chris Callewaert, Clarisse Marotz, Embriette R. Hyde, Justine W. Debelius, Daniel McDonald, and Mitchell L. Sogin. The microbiome and human biology. *Annual Review of Genomics and Human Genetics*, 18(1):65–86, 2017.

[3] Timothy R Sampson, Justine W Debelius, Taren Thron, Stefan Janssen, Gauri G Shastri, Zehra Esra Ilhan, Collin Challis, Catherine E Schretter, Sandra Rocha, Viviana Gradinaru, et al. Gut microbiota regulate motor deficits and neuroinflammation in a model of parkinson's disease. *Cell*, 167(6):1469–1480, 2016.

[4] Cecilia Noecker, Colin P McNally, Alexander Eng, and Elhanan Borenstein. High-resolution characterization of the human microbiome. *Translational research : the journal of laboratory and clinical medicine*, 179:7–23, jan 2017. ISSN 1878-1810. doi: 10.1016/j.trsl.2016.07.012. URL https://www.ncbi.nlm.nih.gov/pubmed/27513210https://www.ncbi.nlm.nih.gov/pmc/PMC5164958/.

[5] Abhishek Kaul, Siddhartha Mandal, Ori Davidov, and Shyamal D Peddada. Analysis of microbiome data in the presence of excess zeros. *Frontiers in microbiology*, 8:2114, 2017.

[6] Xinyan Zhang, Himel Mallick, Zaixiang Tang, Lei Zhang, Xiangqin Cui, Andrew K Benson, and Nengjun Yi. Negative binomial mixed models for analyzing microbiome count data. *BMC Bioinformatics*, 18:1–10, 2017. ISSN 1471-2105. doi: 10.1186/s12859-016-1441-7. URL http://dx.doi.org/10.1186/s12859-016-1441-7.

[7] Lizhen Xu, Andrew D. Paterson, Williams Turpin, and Wei Xu. Assessment and Selection of Competing Models for Zero-Inflated Microbiome Data. *PLOS ONE*, 10(7): e0129606, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0129606.

[8] W. Duncan Wadsworth, Raffaele Argiento, Michele Guindani, Jessica Galloway-Pena, Samuel A. Shelburne, and Marina Vannucci. An integrative Bayesian Dirichlet-multinomial regression model for the analysis of taxonomic abundances in microbiome data. *BMC Bioinformatics*, 18:94, February 2017. ISSN 1471-2105. doi: 10.1186/s12859-017-1516-0.

[9] Keith A. Martinez, Joseph C. Devlin, Corey R. Lacher, Yue Yin, Yi Cai, Jincheng Wang, and Maria G. Dominguez-Bello. Increased weight gain by C-section: Functional significance of the primordial microbiome. *Science Advances*, 3(10):eaao1874, October 2017. ISSN 2375-2548. doi: 10.1126/sciadv.aao1874.

[10] Williams Turpin, Osvaldo Espin-Garcia, Wei Xu, Mark S Silverberg, David Kevans, Michelle I Smith, David S Guttman, Anne Griffiths, Remo Panaccione, Anthony Otley, Lizhen Xu, Konstantin Shestopaloff, Gabriel Moreno-Hagelsieb, GEM Project Research Consortium, Andrew D Paterson, and Kenneth Croitoru. Association of host genome with intestinal microbial composition in a large healthy cohort. *Nature Genetics*, 48: 1413, October 2016.

[11] Curtis Huttenhower, Dirk Gevers, Rob Knight, Sahar Abubucker, Jonathan H Badger, Asif T Chinwalla, Heather H Creasy, Ashlee M Earl, Michael G FitzGerald, Robert S Fulton, et al. Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402):207, 2012.

[12] Adeline Lo, Herman Chernoff, Tian Zheng, and Shaw-Hwa Lo. Framework for making better predictions by directly estimating variables' predictivity. *PNAS*, page 201616647, November 2016. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1616647113.

[13] Adeline Lo, Herman Chernoff, Tian Zheng, and Shaw-Hwa Lo. Why significant variables aren't automatically good predictors. *Proceedings of the National Academy of Sciences*, 112(45):13892–13897, 2015.

[14] Klas Gränsbo, Peter Almgren, Marketa Sjögren, J. G. 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.

[15] International Schizophrenia Consortium. Common polygenic variation contributes to risk of schizophrenia and bipolar disorder. *Nature*, 460(7256):748, 2009.

[16] Paul DP Pharoah, Antonis Antoniou, Martin Bobrow, Ron L. Zimmern, Douglas F. Easton, and Bruce AJ Ponder. Polygenic susceptibility to breast cancer and implications for prevention. *Nature genetics*, 31(1):33, 2002.

[17] Sekar Kathiresan, Cristen J. Willer, Gina M. Peloso, Serkalem Demissie, Kiran Musunuru, Eric E. Schadt, Lee Kaplan, Derrick Bennett, Yun Li, and Toshiko Tanaka. Common variants at 30 loci contribute to polygenic dyslipidemia. *Nature genetics*, 41 (1):56, 2009.

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

[19] Pixu Shi, Anru Zhang, Hongzhe Li, et al. Regression analysis for microbiome compositional data. *The Annals of Applied Statistics*, 10(2):1019–1040, 2016.

[20] Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

[21] Nan M Laird and James H Ware. Random-effects models for longitudinal data. *Biometrics*, 38:963–974, 1982.

[22] John Mullahy. Specification and testing of some modified count data models. *Journal of Econometrics*, 33(3):341–365, 1986. ISSN 03044076. doi: 10.1016/0304-4076(86)90002-3.

[23] Kelvin K W Yau, Kui Wang, and Andy H Lee. Zero-inflated negative binomial mixed regression modeling of over-dispersed count data with extra zeros. *Biometrical Journal*, 45(4):437–452, 2003. ISSN 03233847. doi: 10.1002/bimj.200390024.

[24] Benjamin M. Bolker, Mollie E. Brooks, Connie J. Clark, Shane W. Geange, John R. Poulsen, M. Henry H. Stevens, and Jada-Simone S. White. Generalized linear mixed models: A practical guide for ecology and evolution. *Trends in Ecology & Evolution*, 24 (3):127–135, March 2009. ISSN 0169-5347. doi: 10.1016/j.tree.2008.10.008.

[25] S. S. Wilks. The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Ann. Math. Statist.*, 9(1):60–62, 03 1938. doi: 10.1214/aoms/1177732360. URL https://doi.org/10.1214/aoms/1177732360.

[26] J D Storey. The positive false discovery rate: A Bayesian interpretation and the q-value. *Ann. of Stat.*, pages 2013–2035, 2003.

[27] Y Benjamini and Y Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *JRSSB*, pages 289–300, 1995.

[28] John D. Storey and Robert Tibshirani. Statistical significance for genomewide studies. *PNAS*, 100(16):9440–9445, May 2003. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1530509100.

[29] Yaling Yin, Christine E. Soteros, and Miķelis G. Bickis. A clarifying comparison of methods for controlling the false discovery rate. *Journal of Statistical Planning and Inference*, 139(7):2126–2137, July 2009. ISSN 0378-3758. doi: 10.1016/j.jspi.2008.10.010.

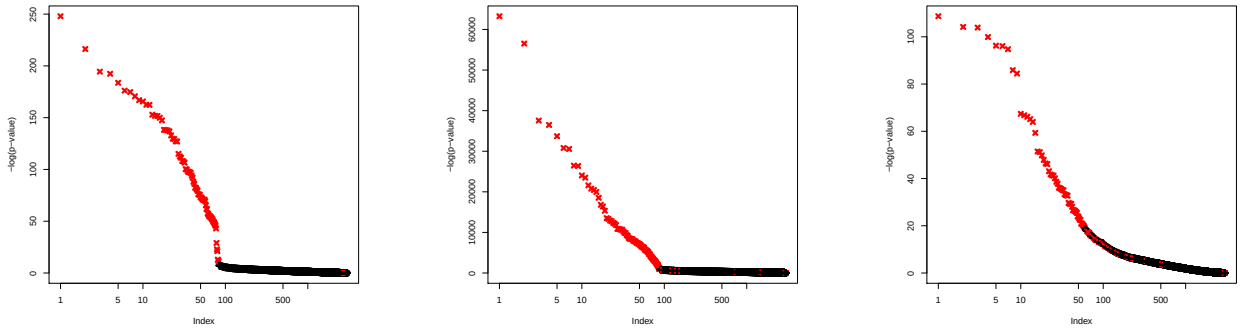
[30] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1–22, 2010. URL https://www.ncbi.nlm.nih.gov/pubmed/20808728.

[31] Arni Magnusson, Hans Skaug, Anders Nielsen, Casper Berg, Kasper Kristensen, Martin Maechler, Koen van Bentham, Ben Bolker, and Mollie Brooks. glmmTMB: Generalized Linear Mixed Models using Template Model Builder, October 2017.

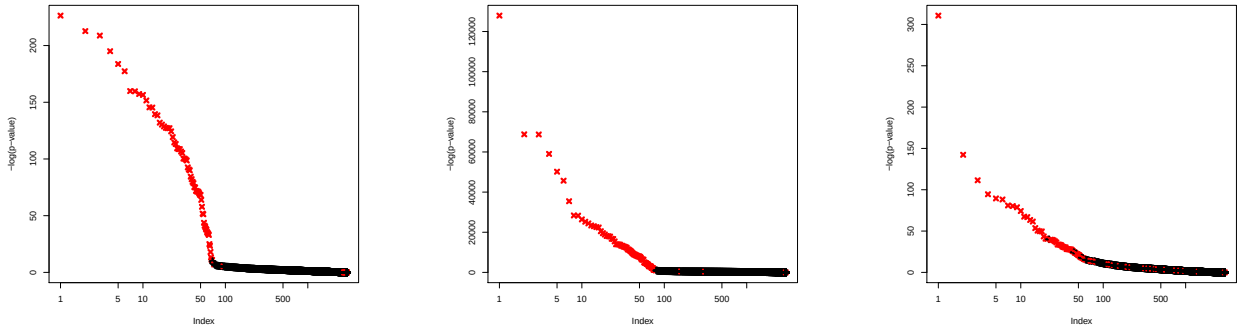
[32] Jerome Friedman, Trevor Hastie, Noah Simon, Junyang Qian, and Rob Tibshirani. Glmnet: Lasso and Elastic-Net Regularized Generalized Linear Models, September 2017.

Appendix A

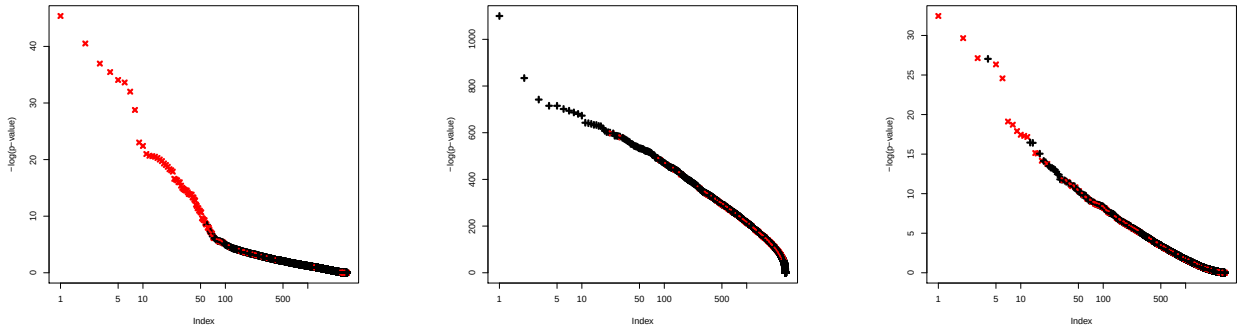
Supplementary Figures from Analyzing Synthetic Datasets



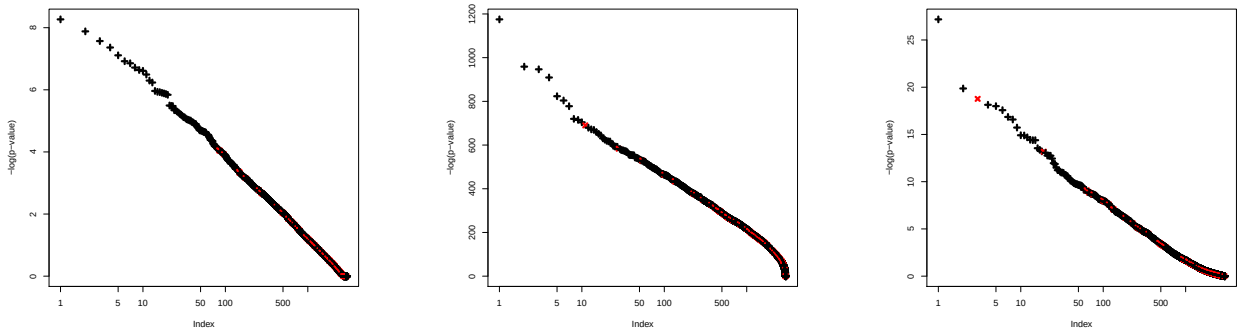
(a) Dataset 1, ZINB **(b)** Dataset 1, ZMP **(c)** Dataset 1, NB



(d) Dataset 2, ZINB **(e)** Dataset 2, ZMP **(f)** Dataset 2, NB

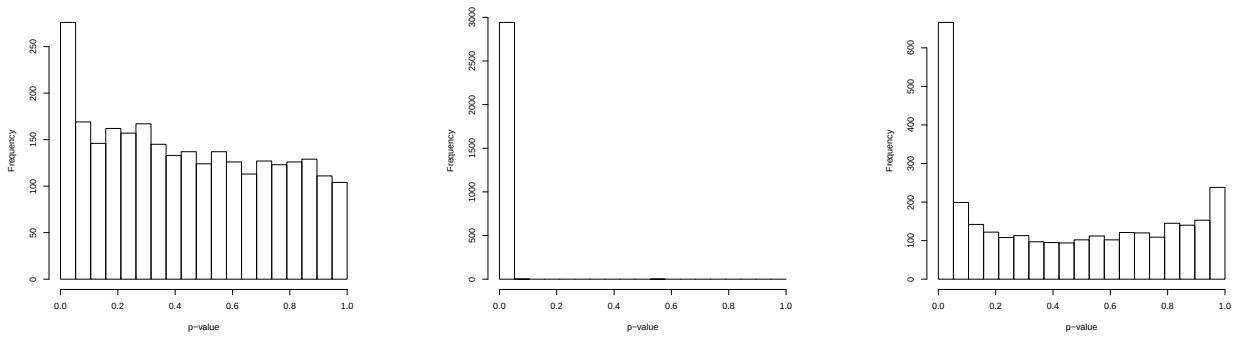


(g) Dataset 3, ZINB **(h)** Dataset 3, ZMP **(i)** Dataset 3, NB

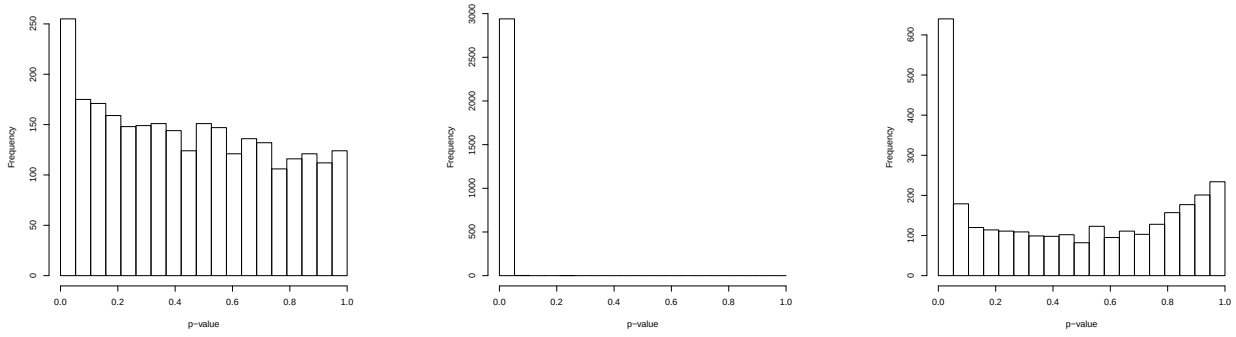


(j) Dataset 4, ZINB **(k)** Dataset 4, ZMP **(l)** Dataset 4, NB

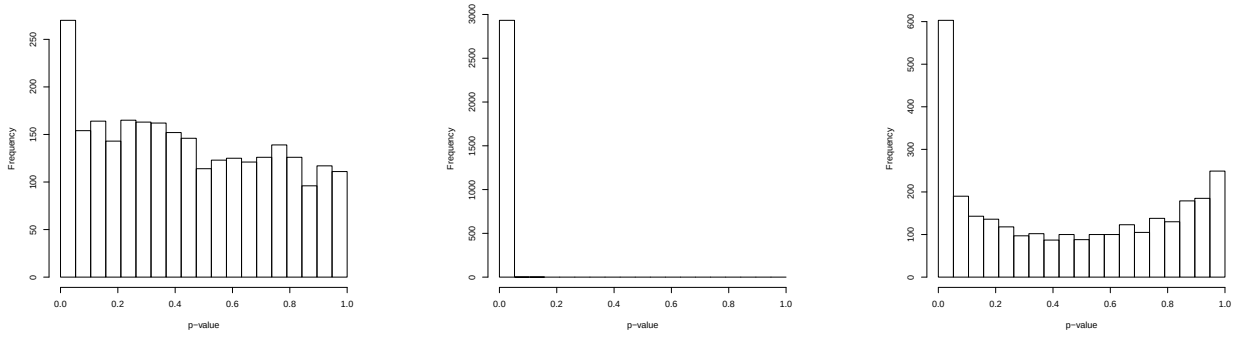
Figure A.1: Ordered log p-values from LRT applied to GLMMs. Rows from top to bottom are results for the four simulation datasets with large count and zero, large count and small zero, small count and large zero, and small count and zero signals, respectively.



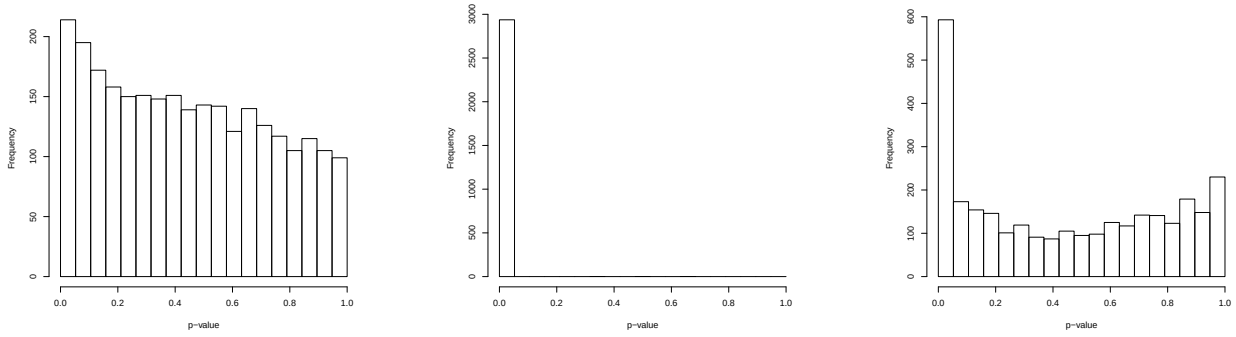
(a) Dataset 1, ZINB **(b)** Dataset 1, ZMP **(c)** Dataset 1, NB



(d) Dataset 2, ZINB **(e)** Dataset 2, ZMP **(f)** Dataset 2, NB

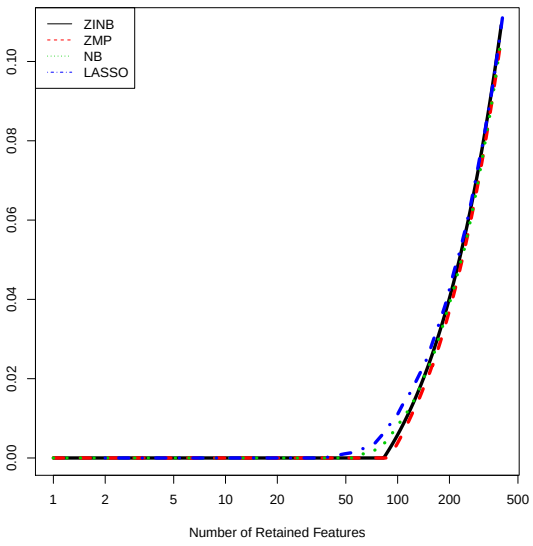


(g) Dataset 3, ZINB **(h)** Dataset 3, ZMP **(i)** Dataset 3, NB

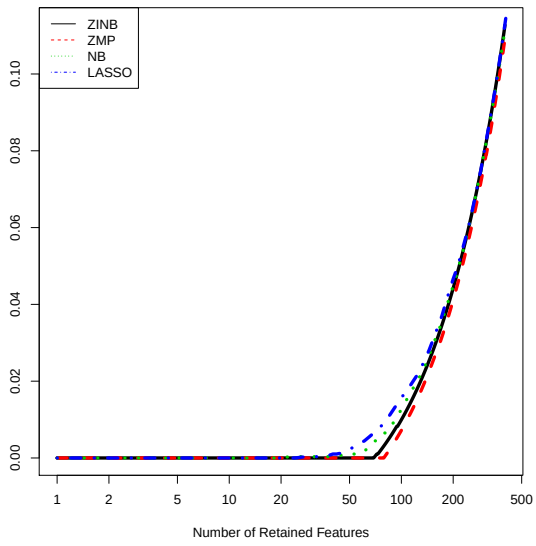


(j) Dataset 4, ZINB **(k)** Dataset 4, ZMP **(l)** Dataset 4, NB

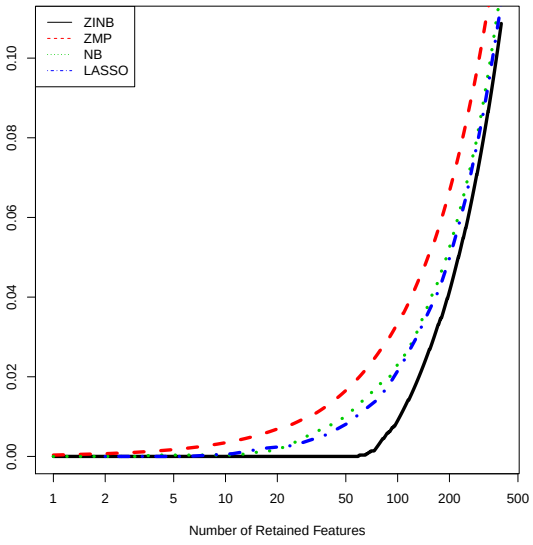
Figure A.2: Histogram of p-value from LR Test. Rows from top to bottom are results for the four simulation datasets with large count and zero, large count and small zero, small count and large zero, and small count and zero signals, respectively.



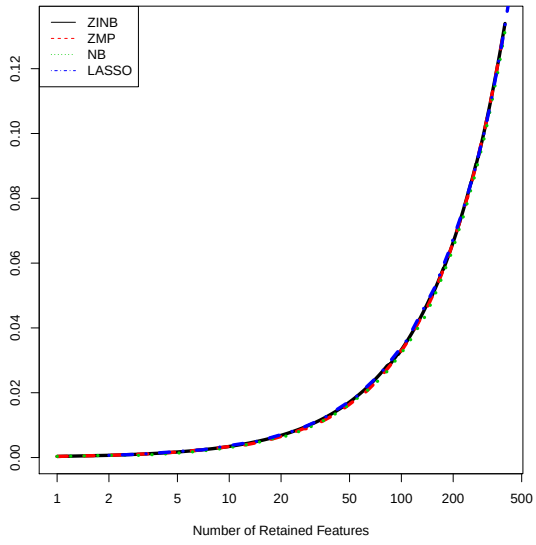
(a) Dataset1, FPR



(b) Dataset2, FPR

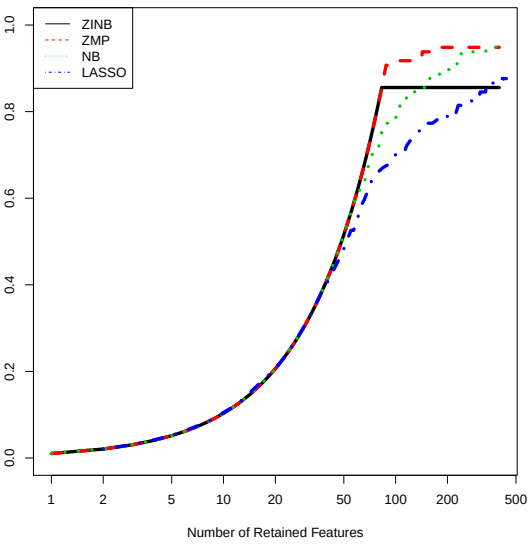


(c) Dataset3, FPR

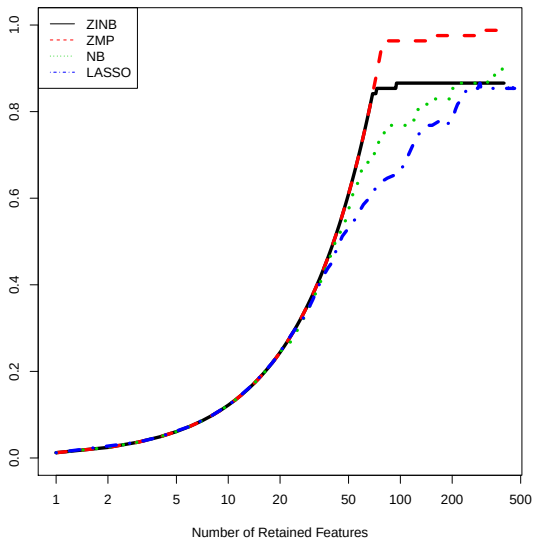


(d) Dataset4, FPR

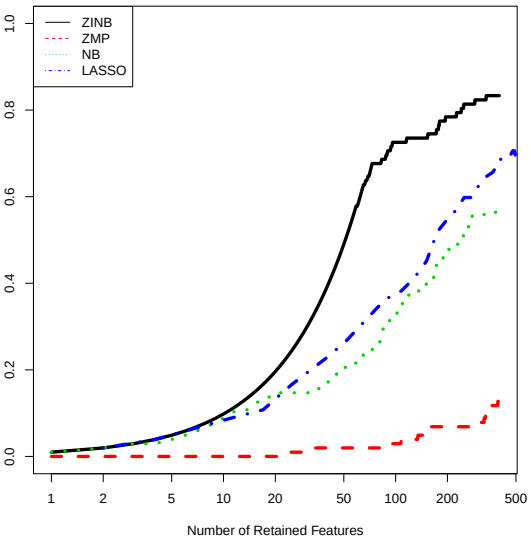
Figure A.3: Comparison of FPR curves between statistical testing and LASSO under four different zero and count signal situations.



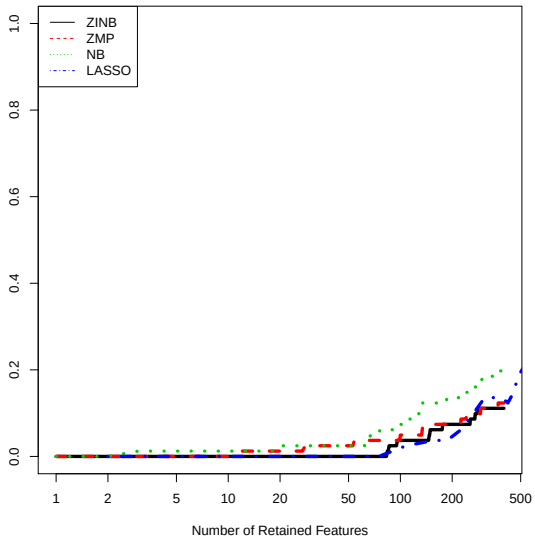
(a) Dataset1, Sensitivity



(b) Dataset2, Sensitivity



(c) Dataset3, Sensitivity



(d) Dataset4, Sensitivity

Figure A.4: Comparison of sensitivity curves between statistical testing and LASSO under four different zero and count signal situations.

Appendix B

R Code

B.1 R Code for Generating A Dataset from ZINB

```
## parameter settings for simulation
n <- 5000
otu.num <- 3000
#count signal (high low)
level.count=c(1, 0.0001)
#zero signal (high low)
level.zero=c(1, 0.0001)

###########################################

sig_factor1 = level.count[1]
sig_factor1.zero = level.zero[1]
home <- "/home/xiw378/canola/lassoglmm/sim1"
###########################################
library (MASS)

l1=10
l2=2
l3=2
l4=10
l5=2
meanT <- 30

sig_factor2=0.2
sig_factor3=0.2
sig_factor4=0.2
sig_factor5=0.2
betafactor0=1

sig_factor2.zero=2
sig_factor3.zero=2
sig_factor4.zero=2
sig_factor5.zero=2
betafactor0s=-0.2
```

```
theta.nb=2
signals <- rbinom(otu.num, size=1, p = 0.03)
indicator.signals <- which(signals==1)

## simulated functions for independent variables and one otu
simulate_x<-function(n,l1,l2,l3,l4,l5, meanT)
{
  factorA<-as.factor(sample(c(1:l1),n,replace=TRUE))
  factorB<-as.factor(sample(c(1:l2),n,replace=TRUE))
  factorC<-as.factor(sample(c(1:l3),n,replace=TRUE))
  factorD<-as.factor(sample(c(1:l4),n,replace=TRUE))
  factorE<-as.factor(sample(c(1:l5),n,replace=TRUE))
  #meanT is the lambda
  logn<-log(rpois(n, meanT))
  data.frame(logn,factorA,factorB,factorC,factorD,factorE)
}

simulate_one_otu.zinb<-function(sig_factor1, sig_factor2, sig_factor3,
                                sig_factor4, sig_factor5, betafactor0,
                                sig_factor1.zero, sig_factor2.zero,
                                sig_factor3.zero, sig_factor4.zero,
                                sig_factor5.zero, betafactor0s, theta.nb,
                                signals)
{
  # negative binomial
  if(signals==0) sig_factor1 = sig_factor1.zero = 0

  betafactor1<-rnorm(l1,sd=sig_factor1)
  betafactor2<-rnorm(l2,sd=sig_factor2)
  betafactor3<-rnorm(l3,sd=sig_factor3)
  betafactor4<-rnorm(l4,sd=sig_factor4)
  betafactor5<-rnorm(l5,sd=sig_factor5)

  mu.nb <-exp(independent.variables[,"logn"]+betafactor0
              +betafactor1[independent.variables[,"factorA"]]
              +betafactor2[independent.variables[,"factorB"]]
              +betafactor3[independent.variables[,"factorC"]]
              +betafactor4[independent.variables[,"factorD"]]
              +betafactor5[independent.variables[,"factorE"]])

  # bernoulli
  betafactor1s<-rnorm(l1,sd=sig_factor1.zero)
  betafactor2s<-rnorm(l2,sd=sig_factor2.zero)
  betafactor3s<-rnorm(l3,sd=sig_factor3.zero)
  betafactor4s<-rnorm(l4,sd=sig_factor4.zero)
```

```
  betafactor5s<-rnorm(l5,sd=sig_factor5.zero)

  mu.b <- exp(betafactor0s+betafactor1s[independent.variables[,"factorA"]]
              + betafactor2s[independent.variables[,"factorB"]]
              + betafactor3s[independent.variables[,"factorC"]]
              + betafactor4s[independent.variables[,"factorD"]]
              +betafactor5s[independent.variables[,"factorE"]])

  pzero <- mu.b / (1 + mu.b) # binomial probabilities

  manyzero <- TRUE
  while (manyzero == TRUE){
    y.zinb <- (runif(n) > pzero)
    is.counts <- which(y.zinb ==1 )
    manyzero <- FALSE

    # Simulate the negative binomial data
    y.nb <- rnegbin(n = length(is.counts),
                    mu = mu.nb[is.counts],
                    theta = theta.nb)
    y.zinb[is.counts] <- y.nb

  }
  y.zinb
}

## simulate fixed and random variables
independent.variables <- data.frame(matrix (0, n, 6))
independent.variables <- simulate_x(n,l1,l2,l3,l4,l5,meanT)

## simulate otus
otus.zinb <- data.frame(matrix (0, n, otu.num))

for(i in 1:otu.num)
{

  otus.zinb[,i] <- simulate_one_otu.zinb(sig_factor1, sig_factor2,
                                         sig_factor3, sig_factor4,
                                         sig_factor5, betafactor0,
                                         sig_factor1.zero, sig_factor2.zero,
                                         sig_factor3.zero, sig_factor4.zero,
                                         sig_factor5.zero, betafactor0s,
                                         theta.nb,  signals[i])

}
```

```
## calculate the Total Reads and genarate one dataset
TotalRead <- rowSums(otus.zinb)
logTR <- log(TotalRead)
dataset <- data.frame(independent.variables, otus.zinb, TotalRead, logTR)

index.factors <- t(as.matrix(sapply(colnames(independent.variables[-1]),
                   grep,colnames(dataset))))
index.otus <- match(colnames(otus.zinb),colnames(dataset))

one_dataset <- list(dataset = dataset, n = n , otu.num = otu.num,
                    l1 = l1, l2 = l2, l3 = l3, l4 = l4, l5 = l5,
                    meanT = meanT, sig_factor1 = sig_factor1,
                    sig_factor2 = sig_factor2, sig_factor3 = sig_factor3,
                    sig_factor4 = sig_factor4, sig_factor5 = sig_factor5,
                    betafactor0 = betafactor0,
                    sig_factor1.zero = sig_factor1.zero,
                    sig_factor2.zero = sig_factor2.zero,
                    sig_factor3.zero = sig_factor3.zero,
                    sig_factor4.zero = sig_factor4.zero,
                    sig_factor5.zero = sig_factor5.zero,
                    betafactor0s = betafactor0s, theta.nb = theta.nb,
                    signals = signals,
                    indicator.signals = indicator.signals,
                    index.factors = index.factors, index.otus = index.otus,
                    sig_factor1.zero)

save(one_dataset, file=paste0(home, '/', 'one_dataset.Rdata'))
```

B.2 R Code for Fitting ZINB, ZMP, NB and LASSO

```
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)

method <- "zinb"
logpdir <- paste0(home, '/', method, "out", '/', "logp")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(logpdir,recursive = TRUE)
dir.create(plotsdir,recursive = TRUE)
```

```
library(glmmTMB)
source(paste0(sourcedir, '/','lrtest.R'))

if(!exists("ifold"))
{
  ifold <- 1
}
j <- ifold

load(paste0(home, '/', 'one_dataset.Rdata'))

ntr <- 500
data.glmm <-one_dataset$dataset[1:ntr,]
data.glmm$y <- data.glmm[, paste0("X", j)]

## Zero-inflated negative binomial model
zinb <- glmmTMB( y ~ factorA+factorB+factorC
                 +(1|factorD)+(1|factorE), data=data.glmm,
                 zi= ~ factorA+factorB+factorC
                 +(1|factorD)+(1|factorE),
                 family = nbinom2)

zinb0 <- glmmTMB( y ~ factorB+factorC
                      +(1|factorD)+(1|factorE), data=data.glmm,
                  zi= ~ factorB+factorC+(1|factorD)+(1|factorE),
                  family = nbinom2)

#log p_value  for each otu j
log.p.zinb <- lrtest.zi.model(zinb0,zinb)$log.pvalue
## output for log.pvalue
file.remove(paste0(logpdir, '/', 'log.p.zinb', j , '.txt'))
cat (log.p.zinb, file = paste0(logpdir, '/', 'log.p.zinb', j , '.txt'))
#####################################################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)

method <- "zmp"
logpdir <- paste0(home, '/', method, "out", '/', "logp")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(logpdir,recursive = TRUE)
dir.create(plotsdir,recursive = TRUE)

library(glmmTMB)
```

```
source(paste0(sourcedir, '/','lrtest.R'))
######################
if(!exists("ifold"))
{
  ifold <- 1
}
j <- ifold

load(paste0(home, '/', 'one_dataset.Rdata'))

ntr <- 500
data.glmm <-one_dataset$dataset[1:ntr,]
data.glmm$y <- data.glmm[, paste0("X", j)]

#glm
#logistic
hurdle.l<-glmmTMB((y>0)~factorA+factorB+factorC
                        +(1|factorD)+(1|factorE),data=data.glmm,
                        ziformula =~0,family=binomial())
hurdle.l0<-glmmTMB((y>0)~factorB+factorC
                         +(1|factorD)+(1|factorE),data=data.glmm,
                         ziformula =~0,family=binomial())

#poisson
hurdle.poisson <-  glm(y ~ factorA+factorB+factorC+factorD
                           +factorE+offset(logn),
                       family  = "poisson",
                       data = subset(data.glmm,y>0))

hurdle.poisson0 <-  glm(y ~ factorB+factorC+factorD+factorE
                            +offset(logn),
                        family  = "poisson",
                        data = subset(data.glmm,y>0))

log.p.zmp <- lrtest.hurdle(hurdle.l0, hurdle.poisson0,
                           hurdle.l, hurdle.poisson)$log.pvalue

## output
file.remove(paste0(logpdir, '/', 'log.p.zmp', j , '.txt'))
cat (log.p.zmp, file = paste0(logpdir, '/', 'log.p.zmp', j , '.txt'))
######################################################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)
```

```
method <- "nb"
logpdir <- paste0(home, '/', method, "out", '/', "logp")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(logpdir,recursive = TRUE)
dir.create(plotsdir,recursive = TRUE)


library(glmmTMB)
source(paste0(sourcedir, '/','lrtest.R'))

if(!exists("ifold"))
{
  ifold <- 1
}
j <- ifold

load(paste0(home, '/', 'one_dataset.Rdata'))

ntr <- 500
data.glmm <-one_dataset$dataset[1:ntr,]
data.glmm$y <- data.glmm[, paste0("X", j)]


nb <- glmmTMB( y ~ factorA+factorB+factorC
                    +(1|factorD)+(1|factorE), data=data.glmm,
               family = nbinom2)

nb0 <- glmmTMB( y ~ factorB+factorC+(1|factorD)+(1|factorE), data=data.glmm,
               family = nbinom2)

log.p.nb <- lrtest.zi.model(nb0,nb)$log.pvalue

## output for log.pvalue
file.remove(paste0(logpdir, '/', 'log.p.nb', j , '.txt'))
cat (log.p.nb, file = paste0(logpdir, '/', 'log.p.nb', j , '.txt'))
#####################################################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)

method <- "lasso"
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(plotsdir,recursive = TRUE)
```

```
library(glmnet)
#load the dataset
load(paste0(home, '/', 'one_dataset.Rdata'))

source(paste0(sourcedir, '/','fun for lasso.R'))
source(paste0(sourcedir, '/','dataprocess_for_lasso.R'))

#model.matrix for lasso
X <- model.matrix(factorA~., data.lasso)[,-1]
y <- data.lasso$factorA

#num of samples for the train model and the prediction
ntr <- 300
X_tr <- X[1:ntr,]
y_tr <- y[1:ntr]

## fit lasso with the lambda and  predictions
lasso.tr <- glmnet (x = X_tr, y= y_tr, nlambda=200, lambda.min.ratio= 1E-8,
                    family = "multinomial")
probs_pred_lasso.tr <- predict(lasso.tr, newx = X[(ntr+1):one_dataset$n,],
                               type = "response")

#save the lambda for loocv.lasso
file.remove(paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))
cat (lasso.tr$lambda, file =
     paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))

#evaluate the performance for train model
tab_sel.lasso.tr <- summary.fsel.lasso
                    (lasso.tr, one_dataset$signals, cutoff=0)
eval.sel.lasso.tr <- as.data.frame
                    (t(sapply (tab_sel.lasso.tr, eval.tab.sel)))
eval.sel.lasso.tr$cutoff <- lasso.tr$lambda

#plots for results
pdf(file=paste0(plotsdir, '/', 'Number-of-Returned-Features.pdf'))
plot (log(lasso.tr$lambda), eval.sel.lasso.tr$nsel,
      xlim = range(log(lasso.tr$lambda)),
      xlab="log(lambda)", ylab="Number of Retained Features",
      main = "Number of Returned Features")
dev.off()

pdf(file=paste0(plotsdir, '/', 'FDR for LASSO.pdf'))
plot (eval.sel.lasso.tr$nsel,eval.sel.lasso.tr$fdr, log = "x",
```

```
      xlab="number of returned features", ylab="fdr",
      main = "FDR for LASSO")
grid(ny = 10, nx =20)
dev.off()

pdf(file=paste0(plotsdir, '/', 'Sensitivity for LASSO.pdf'))
par (mar = c(4,4,2,4))
plot (eval.sel.lasso.tr$nsel, eval.sel.lasso.tr$sensitivity,
      xlab="number of returned features", ylab="sensitivity",
      main="Sensitivity for LASSO")
par (new = TRUE)
plot (eval.sel.lasso.tr$nsel, eval.sel.lasso.tr$TP,
        axes = F, xlab = "", ylab = "")
axis(side = 4);mtext(side = 4, line = 3, text = "True Positive")
grid(ny = 10, nx =20)
dev.off()

pdf(file=paste0(plotsdir, '/', 'FPR for LASSO.pdf'))
par (mar = c(4,4,2,4))
plot (eval.sel.lasso.tr$nsel, 1-eval.sel.lasso.tr$specificity,
      xlab="number of returned features", ylab="FPR", main="FPR for LASSO")
par (new = TRUE)
plot (eval.sel.lasso.tr$nsel, eval.sel.lasso.tr$FP,
      axes = F, xlab = "", ylab = "")
axis(side = 4);mtext(side = 4, line = 3, text = "False Positive")
grid(ny = 10, nx =20)
dev.off()

pdf(file=paste0(plotsdir, '/', 'ROC for LASSO.pdf'))
plot (1- eval.sel.lasso.tr$specificity, eval.sel.lasso.tr$sensitivity,
      type = "b", xlab="FPR", ylab="sensitivity",
      main="ROC for LASSO") ## roc
grid(ny = 10, nx =20)
dev.off()
```

B.3 R Code for feature Selection and LOOCV for LASSO

```
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)
```

```
##
plotscomdir <- paste0(home, '/', "comparisons")
dir.create(plotscomdir,recursive = TRUE)

method <- c("zinb","zmp","nb", "lasso")
label<- c("ZINB", "ZMP", "NB", "LASSO")

load(paste0(home, '/', 'one_dataset.Rdata'))
source(paste0(sourcedir, '/', 'generators', '/', 'fun for lasso.R'))
source(paste0(sourcedir, '/', 'evaluation', '/', 'evalp.R'))

#get all the log(p)
log.p <- matrix(NA, one_dataset$otu.num, length(method)-1)

for(i in 1 : (length(method)-1))
{
  logpdir <- paste0(home, '/', method[i], "out", '/', "logp")
  log.p[,i] <- extract.p(logpdir, method[i], one_dataset$otu.num)
}

#compute the confusion matrix for all the glmm models
model.eval <- as.list(1:length(method))

for(i in 1 : (length(method)-1))
{
  plotsdir <- paste0(home, '/', method[i],"out", '/', "plots")
  model.eval[[i]] <- evaluate.one.model
                     (plotsdir, method[i], label[i], log.p[,i],
                      one_dataset$signals, num.cutoff = 400)
}

#add to the lasso results
model.eval[[length(method)]] <- eval.sel.lasso.tr
names(model.eval) <- paste(method)

#comparison
plotscomdir <- paste0(home, '/', "comparisons")
plots.4methods <- compare.4methods(plotscomdir, model.eval,
                  method[1], method[2], method[3], method[4])

sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)

method <- "lasso"
```

```
probs_preddir <- paste0(home, '/', method,"out", '/', "probs_pred")

dir.create(probs_preddir,recursive = TRUE)
library(glmnet)
if(!exists("ifold"))
{
  ifold <- 1
}

k <- ifold

#load the dataset and the lambda
load(paste0(home, '/', 'one_dataset.Rdata'))
lambda <- scan(paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))
#####################################################
source(paste0(sourcedir, '/','dataprocess_for_lasso.R'))
#model.matrix for lasso
X <- model.matrix(factorA~., data.lasso)[,-1]
y <- data.lasso$factorA
#num of samples for the train model and the prediction
ntr <- 300
X_tr <- X[1:ntr,]
y_tr <- y[1:ntr]
# leave one out cross validation and predictions
probs_pred_lasso <- data.frame (matrix(0, one_dataset$l1, length(lambda)))
# fit the model
lasso.cvfit <- glmnet (x = X_tr[-k,], y=y_tr[-k],  lambda=lambda,
                       family = "multinomial")
# make predictions
probs_pred_lasso <- predict(lasso.cvfit, newx = t(X_tr[k,]),
                            type = "response")[1,,]

## output
file.remove(paste0(probs_preddir, '/', 'probs_pred_lasso', k, '.txt'))
cat (probs_pred_lasso, file = paste0(probs_preddir, '/',
                                     'probs_pred_lasso', k, '.txt'))

sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/sim1"
setwd(home)

method <- "lasso"
probs_preddir <- paste0(home, '/', method,"out", '/', "probs_pred")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")
```

```
source(paste0(sourcedir, '/','compred.r'))

#load the lambda
lambda <- scan(paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))
probs_pred_lasso.cvfit <- array(0, dim =
                                  c(ntr, one_dataset$l1, length(lambda)))

for(k in 1:ntr)
{

  prob0 <- scan(paste0(probs_preddir, '/', 'probs_pred_lasso', k, '.txt'))
  prob <-  matrix (prob0, one_dataset$l1, length(lambda))

  probs_pred_lasso.cvfit[k,,] <- prob
}

#calculate the ER & AMLP
amlp.tr = er.tr = amlp.cvfit = er.cvfit = rep(0, length(lambda))

for(l in 1:length(lambda))
{
  amlp.tr[l] <- evaluate_pred(probs_pred_lasso.tr[,,l],
                              y[(ntr+1):one_dataset$n])$amlp
  er.tr[l] <- evaluate_pred(probs_pred_lasso.tr[,,l],
                            y[(ntr+1):one_dataset$n])$er

  amlp.cvfit[l] <- evaluate_pred(probs_pred_lasso.cvfit[,,l], y_tr)$amlp
  er.cvfit[l] <- evaluate_pred(probs_pred_lasso.cvfit[,,l], y_tr)$er
}
loglambda <- log(lambda)
#plots for results of ER & AMLP
pdf(file=paste0(plotsdir, '/', 'ER-for-LASSO.pdf'))
par(mar = c(4, 4, 2.5, 1))
plot (loglambda, er.tr, lwd=4, type = "l", lty = 1,
      ylim = range(er.tr, er.cvfit),
      xlab="log(lambda)", ylab="ER for LASSO")
points (loglambda, er.cvfit, type = "l", lty = 3, col = 2, lwd=8)
title("ER for LASSO", line = 1.5)
#double axis
sequence <- seq(1,length(lambda), by=8)
xtick2 <- loglambda[sequence]
xlabel2 <- eval.sel.lasso.tr$nsel[sequence]
axis(side = 3, padj = 1,
     at = xtick2 , label = xlabel2 )
mtext("Number of Retained Features", side = 3, line = -1)
```

```
legend("center", c("out-of-sample test", "loocv"), col = c(1, 2),
       text.col = "black", lty = c(1, 3))
abline(h = (one_dataset$l1-1)/one_dataset$l1, col = 3, lty = 2, lwd=4)
abline(h = seq(0, 1, by=0.05), col = "grey", lty = 2)
dev.off()

##AMLP
pdf(file=paste0(plotsdir, '/', 'AMLP-for-LASSO.pdf'))
par(mar = c(4, 4, 2.5, 1))
plot (loglambda, amlp.tr, lwd=4, type = "l", lty = 1,
      ylim = range(amlp.tr, amlp.cvfit),
      xlab="log(lambda)", ylab="AMLP")
points (loglambda, amlp.cvfit, type = "l", lty = 3, col = 2, lwd=8)
title("AMLP for LASSO", line = 1.5)
#double axis
sequence <- seq(1,length(lambda), by=8)
xtick2 <- loglambda[sequence]
xlabel2 <- eval.sel.lasso.tr$nsel[sequence]
axis(side = 3, padj = 1,
     at = xtick2 , label = xlabel2 )
mtext("Number of Retained Features", side = 3, line = -1.2)

legend("center", c("out-of-sample test", "loocv"), col = c(1, 2),
       text.col = "black", lty = c(1, 3))
abline(h = log(one_dataset$l1), col = 3, lty = 2, lwd=4)
#abline(h = seq(0, 1, by=0.05), col = "black", lty = 2)
dev.off()
```

B.4 R Code for Some General Functions in Use

```
##lrtest for zmp
lrtest.hurdle <- function (M0_C, M0_Z, M1_C,M1_Z)
{
  loglike0_C <- logLik(M0_C)
  loglike0_Z <- logLik(M0_Z)
  loglike1_C <- logLik(M1_C)
  loglike1_Z <- logLik(M1_Z)

  loglike0 <- loglike0_C + loglike0_Z
  loglike1 <- loglike1_C + loglike1_Z
```

```
  df0 <- attr(loglike0_C, "df") + attr(loglike0_Z, "df")
  df1 <- attr(loglike1_C, "df") + attr(loglike1_Z, "df")
  df <- df1 - df0
  SS <- as.numeric(2*(loglike1 - loglike0))
  list (SS = SS, df=df, log.pvalue=
                        pchisq (SS, df, lower.tail = FALSE, log.p = TRUE))

}

##lrtest for zi model
lrtest.zi.model <- function(M0,M1)
{
  loglike0 <- logLik(M0)
  loglike1 <- logLik(M1)
  df0 <- attr(loglike0, "df")
  df1 <- attr(loglike1, "df")
  df <- df1 - df0
  SS <- as.numeric(2*(loglike1 - loglike0))
  list (SS = SS, df=df, log.pvalue=
                        pchisq (SS, df, lower.tail = FALSE, log.p = TRUE))
}

## transformations
trans <- function (x) asin (sqrt(x))
#trans <- function (x) asin (2*x - 1)
#trans <- function (x) asin (x)
a <- 0
otu_trans_ra <- trans((one_dataset$dataset[,one_dataset$index.otus]+a)
                      /(one_dataset$dataset$TotalRead))

data.lasso <- data.frame(one_dataset$dataset[,one_dataset$index.factors],
                         otu_trans_ra)
########################################################
library ("glmnet")
library ("rda")
########################################################
summary.fsel.lasso <- function(lassofit, signals, cutoff)
{
  betas <- coef(lassofit)
  nlambda <- length(lassofit$lambda)

  num.features <- length(signals)
  num.allvars <- dim(betas[[1]])[[1]]
  num.fixedvars <- num.allvars - num.features
```

```
  G <- length (betas)

  mbetas <- array(0, dim = c(G, num.allvars, nlambda),
                  dimnames = list(names(betas),
                                  rownames(betas[[G]]),
                                  colnames(betas[[G]])
                  )
  )
  for (g in 1:G)
  {
    mbetas[g,,] <- as.matrix(betas[[g]])
  }

  SD.beta <- apply (mbetas, c(2,3), sd )

  sel <- (SD.beta > cutoff)*1

  tab_sel <- apply (sel[-(1:num.fixedvars), ], 2, table01, signals)

  tab_sel
}
#########################################################
# tab_sel1 <- apply (sel[-(1:num.fixedvars), ], 2, table, truesignal)
#########################################################
table01 <- function (sel, signals)
{
  table01 <- data.frame(matrix (0, 2,2))
  rownames(table01) <- paste0("sel", c("0", "1"))
  colnames(table01) <- paste0("true", c("0", "1"))

  table01[1,1] <- sum ((sel==0) * (signals==0))
  table01[1,2] <- sum ((sel==0) * (signals==1))
  table01[2,1] <- sum ((sel==1) * (signals==0))
  table01[2,2] <- sum ((sel==1) * (signals==1))
  table01

}
#########################################################
eval.tab.sel <- function (tab_sel)
{
  nsel <- sum (tab_sel[2,])
  TP <- tab_sel[2,2]
  FP <- tab_sel[2,1]
  fdr <- tab_sel[2,1]/ nsel
  sensitivity <- tab_sel[2,2]/ sum (tab_sel[,2])
```

```
  specificity <- tab_sel[1,1] / sum (tab_sel[,1])
  c(fdr = fdr, sensitivity = sensitivity,
    specificity=specificity, nsel = nsel,
    TP=TP, FP=FP)
}

# function to calculate the retained feature for lasso
summary.nsel.lasso <- function(lassofit, otu.num, cutoff)
{
  betas <- coef(lassofit)
  nlambda <- length(lassofit$lambda)

  num.features <- otu.num
  num.allvars <- dim(betas[[1]])[[1]]
  num.fixedvars <- num.allvars - num.features

  G <- length (betas)

  mbetas <- array(0, dim = c(G, num.allvars, nlambda),
                  dimnames = list(names(betas),
                                  rownames(betas[[G]]),
                                  colnames(betas[[G]])
                  )
  )
  for (g in 1:G)
  {
    mbetas[g,,] <- as.matrix(betas[[g]])
  }

  SD.beta <- apply (mbetas, c(2,3), sd )

  sel <- (SD.beta > cutoff)*1

  nsel <- apply (sel[-(1:num.fixedvars), ], 2, sum)
  #nsel <- apply (sel[-(1:num.fixedvars), ], 2, function(x) sum (x == 1))
  nsel
}

#########################################################
#function to extract log p_value
extract.p <- function(logpdir, method, otu.num)
{
  #extract the log p_values
  log.p0 = rep(NA, otu.num)
```

```
  for(i in 1:otu.num)
  {
    fn.log.p <- paste0(logpdir, '/', 'log.p.',  method, i, '.txt')

    if (file.exists(fn.log.p))
    {
      log.p0[i] <- scan(fn.log.p)
    }
    else
    {
      cat (fn.log.p, "does not exsit\n")
    }
  }
  log.p0
}
#########################################################
#function to feature selection with each cutoff
fs.log.p <- function(log.p0,cutoff)
{
  if(log.p0 <= cutoff) {log.p.fs=1;} else {log.p.fs=0;}
}
#########################################################
#function to process log p_value, evaluate the model and draw the plots
evaluate.one.model <- function(plotsdir, method, label,
                               log.p0, signals, num.cutoff)
{

  #save the num of NA's
  num.na <- sum(is.na(log.p0))

  ##use 1 instead all the NA's
  log.p0[which(is.na(log.p0) == TRUE)] <- 0

  #use all the means for each two adjacent p_values as cutoffs
  order.log.p0 <- order(log.p0)
  log.p0.mean <- zoo::rollmean(log.p0[order.log.p0], 2)
  cutoff <- log.p0.mean[1:num.cutoff]

  #calculate all the confusion matrixs for each cutoff
  tab_sel.model = as.list (1:length(cutoff))

  for(i in 1:length (cutoff))
  {
    fs.model <- unlist(lapply(log.p0, fs.log.p, cutoff[i]))
    tab_sel.model[[i]] <- table01(fs.model,signals)
```

```
}

#evaluate the confusion matrixs for the model
eval.sel.model <- as.data.frame(t(sapply (tab_sel.model, eval.tab.sel)))

##calculate the estimated fdr for the model
estimated.fdr <- p.adjust(exp(log.p0[order(log.p0)]),
                          method = "fdr")[1:length(cutoff)]

eval.sel.model$estimated.fdr <- estimated.fdr
eval.sel.model$cutoff <- cutoff
eval.sel.model$num.na <- num.na

# plots for p_value
pdf(file=paste0(plotsdir, '/', 'log(p) for ', method,'.pdf'))
plot(-log.p0[order.log.p0], col = (signals + 1)[order.log.p0], lwd=4,
     pch = c(3,4)[(signals + 1)[order.log.p0]],
     log = "x", ylab="-log(pvalue)")
dev.off()

pdf(file=paste0(plotsdir, '/', 'p_value for ', method,'.pdf'))
hist (exp(log.p0[log.p0<0]), breaks = seq(0,1, length=20),
      xlab="pvalue", main="Histogram of pvalue")
dev.off()

##plot of true fdr and estimated fdr
pdf(file=paste0(plotsdir, '/', 'true vs estimated fdr for ',
                method,'.pdf'))
par(mar = c(4, 4, 2, 1))
plot(eval.sel.model$nsel, eval.sel.model$fdr, log = "x",
     type = "l", lty = 1, col = 1, lwd=4,
     xlim = range(na.omit(c(eval.sel.model$nsel))),
     ylim = range(na.omit(c(eval.sel.model$fdr,
     eval.sel.model$estimated.fdr))),
     xlab="Number of Retained Features", ylab="False Discovery Rate",
     main = paste0('False Discovery Rate for  ', label))
points(eval.sel.model$nsel, eval.sel.model$estimated.fdr,
       type = "l", lty = 3, col = 2, lwd=4)

legend("topleft", c("Empirical FDR", "Q-value"), col = c(1, 2),
       text.col = "black", lty = c(1, 3))

dev.off()

#feature selection
```

```
pdf(file=paste0(plotsdir, '/', 'feature selection for ', method,'.pdf'))
plot(eval.sel.model$cutoff,eval.sel.model$nsel,
     xlab="cutoff", ylab="number of returned features",
     main = paste0('feature selection for  ', label))
grid(ny = 10, nx =20)
dev.off()

#FDR
pdf(file=paste0(plotsdir, '/', 'FDR for ', method,'.pdf'))
plot(eval.sel.model$nsel,eval.sel.model$fdr, log = "x",
     type = "l", lty = 1, col = 1,
     xlab="Number of Retained Features", ylab="FDR",
     main = paste0('FDR for  ', label))
grid(ny = 10, nx =20)
dev.off()

#Sensitivity
pdf(file=paste0(plotsdir, '/', 'Sensitivity for ', method,'.pdf'))
par (mar = c(4,4,2,4))
plot (eval.sel.model$nsel,eval.sel.model$sensitivity,
      xlab="Number of Retained Features", ylab="Sensitivity",
      main = paste0('Sensitivity for  ', label))
par (new = TRUE)
plot (eval.sel.model$nsel, eval.sel.model$TP,
      axes = F, xlab = "", ylab = "")
axis(side = 4);mtext(side = 4, line = 3, text = "True Positive")
grid(ny = 10, nx =20)
dev.off()

#FPR
pdf(file=paste0(plotsdir, '/', 'FPR for ', method,'.pdf'))
par (mar = c(4,4,2,1))
plot (eval.sel.model$nsel,1-eval.sel.model$specificity,
      xlab="Number of Retained Features", ylab="FPR",
      main = paste0('FPR for  ', label))
par (new = TRUE)
plot (eval.sel.model$nsel, eval.sel.model$FP,
      axes = F, xlab = "", ylab = "")
axis(side = 4);mtext(side = 4, line = 3, text = "False Positive")
grid(ny = 10, nx =20)
dev.off()

#ROC
pdf(file=paste0(plotsdir, '/', 'ROC for ', method,'.pdf'))
plot (1- eval.sel.model$specificity,eval.sel.model$sensitivity,
```

```
        type = "b",
        xlab="FPR", ylab="Sensitivity", main = paste0('ROC for  ', label))
  grid(ny = 10, nx =20)
  dev.off()

  eval.sel.model
}
#######################################################
#function to compare all the methods
compare.4methods <- function(plotscomdir, eval.one.model,
                             method1, method2, method3, method4)
{
  #FDR
  pdf(file=paste0(plotscomdir, '/', 'FDR for ',
                  'ZINB, ZMP, NB and LASSO', '.pdf'))

  par(mar = c(4, 4, 2, 1))
  #make the plot
  plot(eval.one.model[[method1]]$nsel,eval.one.model[[method1]]$fdr, lwd=4,
       type = "l",  col = 1, lty = 1,  log = "x",
       xlab="Number of Retained Features", ylab="",
       ylim =range(na.omit(c(eval.one.model[[method1]]$fdr,
       eval.one.model[[method2]]$fdr,
       eval.one.model[[method3]]$fdr, eval.one.model[[method4]]$fdr))),
       main=paste0('False Discovery Rate'))

  points(eval.one.model[[method2]]$nsel,eval.one.model[[method2]]$fdr,
         lwd=4, type="l", col=2, lty = 2)

  points(eval.one.model[[method3]]$nsel,eval.one.model[[method3]]$fdr,
         lwd=4, type="l", col=3, lty = 3)

  points(eval.one.model[[method4]]$nsel,eval.one.model[[method4]]$fdr,
         lwd=4, type="l", col=4, lty = 4)


  legend("bottomright", toupper(c(method1, method2, method3, method4)),
         col = c(1,2,3,4), text.col = "black", lty = c(1,2,3,4))
  dev.off()

  #Sensitivity
  pdf(file=paste0(plotscomdir, '/', 'Sensitivity for ',
                  'ZINB, ZMP, NB and LASSO', '.pdf'))

  par(mar = c(4, 4, 2, 1))
```

```
#make the plot
plot(eval.one.model[[method1]]$nsel,eval.one.model[[method1]]$sensitivity,
     lwd=4, type = "l",  lty = 1, col = 1, log = "x",
     xlab="Number of Retained Features", ylab="",
     ylim = range(0,1),
     main=paste0('Sensitivity'))

points(eval.one.model[[method2]]$nsel,
       eval.one.model[[method2]]$sensitivity,
       lwd=4, type="l", col=2, lty = 2)

points(eval.one.model[[method3]]$nsel,
       eval.one.model[[method3]]$sensitivity,
       lwd=4, type="l", col=3, lty = 3)

points(eval.one.model[[method4]]$nsel,
       eval.one.model[[method4]]$sensitivity,
       lwd=4, type="l", col=4, lty = 4)


legend("topleft", toupper(c(method1, method2, method3, method4)),
       col = c(1,2,3,4), text.col = "black", lty = c(1,2,3,4))
dev.off()

#FPR
pdf(file=paste0(plotscomdir, '/', 'FPR for ', 'ZINB, ZMP, NB and LASSO',
                '.pdf'))

par(mar = c(4, 4, 2, 1))
#make the plot
plot(eval.one.model[[method1]]$nsel,
     1-eval.one.model[[method1]]$specificity,
     lwd=4, type = "l",  lty = 1, col = 1, log = "x",
     xlab="Number of Retained Features", ylab="",
     main=paste0('FPR'))

points(eval.one.model[[method2]]$nsel,
       1-eval.one.model[[method2]]$specificity,
       lwd=4, type="l", col=2, lty = 2)

points(eval.one.model[[method3]]$nsel,
       1-eval.one.model[[method3]]$specificity,
       lwd=4, type="l", col=3, lty = 3)

points(eval.one.model[[method4]]$nsel,
```

```
       1-eval.one.model[[method4]]$specificity,
       lwd=4, type="l", col=4, lty = 4)


legend("topleft", toupper(c(method1, method2, method3, method4)),
       col = c(1,2,3,4),
       text.col = "black", lty = c(1,2,3,4))
dev.off()

#ROC
pdf(file=paste0(plotscomdir, '/', 'ROC for ',
    'ZINB, ZMP, NB and LASSO', '.pdf'))

par(mar = c(4, 4, 2, 1))
xlim =range(na.omit(c(1-eval.one.model[[method1]]$specificity,
                      1-eval.one.model[[method2]]$specificity,
                      1-eval.one.model[[method3]]$specificity,
                      1-eval.one.model[[method4]]$specificity)))
ylim =range(na.omit(c(eval.one.model[[method1]]$sensitivity,
                      eval.one.model[[method2]]$sensitivity,
                      eval.one.model[[method3]]$sensitivity,
                      eval.one.model[[method4]]$sensitivity)))
print (ylim)
#make the plot
plot(1- eval.one.model[[method1]]$specificity,
     eval.one.model[[method1]]$sensitivity, lwd=4,
     type = "l",  lty = 1, col = 1,
     xlab="FPR", ylab="Sensitivity",
     xlim = xlim,
     ylim = ylim,
     main=paste0('ROC'))

points(1- eval.one.model[[method2]]$specificity,
       eval.one.model[[method2]]$sensitivity, lwd=4,
       type="l", col=2, lty = 2)

points(1- eval.one.model[[method3]]$specificity,
       eval.one.model[[method3]]$sensitivity, lwd=4,
       type="l", col=3, lty = 3)

points(1- eval.one.model[[method4]]$specificity,
       eval.one.model[[method4]]$sensitivity, lwd=4,
       type="l", col=4, lty = 4)
```

```
  legend("bottomright", toupper(c(method1, method2, method3, method4)),
         col = c(1,2,3,4),
         text.col = "black", lty = c(1,2,3,4))
         # , cex=0.8, border="black",
         # bty = "n",  inset = c(0, 0), adj=0.15, text.width=0.15,
         x.intersp=2,
         # lwd=4, seg.len=7)
  dev.off()
}

#######################################################
evaluate_pred <- function (probs_pred, y, caseid = names (y),
                           showplot = FALSE)
{
    if (is.factor(y)) y <- as.numeric (y)
    if (is.null (caseid)) caseid <- 1:length (y)

    C <- ncol (probs_pred)
    values_pred <- apply (probs_pred, 1, which.max)

    table_eval <- data.frame (caseid, y, probs_pred, 1 * (values_pred != y))
    colnames (table_eval) <- c("Case ID", "True Label",
                               paste ("Pred. Prob", 1:C), "Wrong?")

    eval_tab_pred (table_eval, showplot = showplot)
}

eval_tab_pred <- function (table_eval, showplot = TRUE, ...)
{
   if (is.character (table_eval))
   {
     table_eval <- as.matrix (read.table (table_eval))
   }

   C <- ncol (table_eval) - 3
   colnames (table_eval) <- c("Case ID", "True Label",
                              paste ("Pred. Prob", 1:C), "Wrong?")

   probs_pred <- table_eval [, 2+(1:C)]
   y <- table_eval[,2]
   probs_at_truelabels <- probs_attrue_bplr (probs_pred, y)
   which.wrong <- which (table_eval[,C+3] == 1)
   n <- nrow (table_eval)

   amlp <- - mean (log (probs_at_truelabels))
```

```
   no_errors <- sum (table_eval[, C+3])
   er <- no_errors/n

   yl <- y; if (C == 2) yl[y==2] <- 3

   plotargs <- list (...)
   if (is.null (plotargs$ylab))
       plotargs$ylab <- "Predictive Probability at True Label"
   if (is.null (plotargs$xlab)) plotargs$xlab <- "Case Index"
   if (is.null (plotargs$ylim)) plotargs$ylim <- c(0,1)
   if (is.null (plotargs$pch)) plotargs$pch <- yl

   if (showplot)
   {
       plotargs$x <- probs_at_truelabels
       do.call (plot, plotargs)

       if (C == 2) abline (h = 0.5)
       abline (h = 0.1,lty = 2)

       title (main = sprintf ("AMLP = %5.3f, Error Rate = %4.2f%% (%d/%d)",
                              amlp, er*100, no_errors, n),
              cex = 0.8, line = 0.5)

       if (no_errors > 0) {
        text (which.wrong, probs_at_truelabels[which.wrong],
              labels = which.wrong,
                  srt = 90, adj = - 0.4, cex = 0.9, col = "red")

       }
   }
   list (probs_at_truelabels = probs_at_truelabels, table_eval = table_eval,
         amlp = amlp, er = er, which.wrong = which.wrong )
}
```

B.5 R Code for Real Data analysis

```
#########################clean the data#######################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/realdata/ethnicity"
setwd(home)
```

```
method <- "lasso"
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(plotsdir,recursive = TRUE)

library(glmnet)

#load the dataset and replace blank with "NA's"
realdata=read.csv(file="/home/xiw378/canola/data/OTU_NG_Ethnicity.csv",
                  header = T, stringsAsFactors = F,
                  na.strings=c("","NA"))

#Convert to Factors
realdata$Age <- factor(realdata$Age, levels=sort(unique(realdata$Age)))
realdata$Sex <- factor(realdata$Sex, levels=sort(unique(realdata$Sex)))
realdata$Ethnicity <- factor(realdata$Ethnicity,
                             levels=sort(unique(realdata$Ethnicity)))

#delete the replicated samples(if any);
#find and delete all the samples with "NA's"
summary(realdata$Age)
summary(realdata$Sex)
summary(realdata$Ethnicity)

realdata=unique(realdata)

realdata[which(is.na(realdata), arr.ind = T), 146:148]
realdata<-na.omit(realdata)

##delete the samples with only one obsvervation in ethnicity in excel
##and save as "OTU_NG_Ethnicity.csv

n <- dim(realdata)[1]

independent.variables <- data.frame (realdata[,146:148])

# extract OTU data
otus = realdata[, 2:(ncol(realdata)-3)]
otu.num <- dim(otus)[2]

TotalRead = rowSums(otus)
logTR <- log(TotalRead)
dataset <- data.frame(independent.variables, otus, TotalRead, logTR)

index.factors <- t(as.matrix(sapply(colnames(independent.variables),
```

```
                    grep,colnames(dataset))))
index.otus <- match(colnames(otus),colnames(dataset))

one_dataset <- list(dataset = dataset, n = n, otu.num = otu.num,
                    l1 = length(unique(realdata$Ethnicity)),

                    index.factors = index.factors, index.otus = index.otus)

save(one_dataset, file=paste0(home, '/', 'one_dataset.Rdata'))

#########################fit the Lasso######################
load(paste0(home, '/', 'one_dataset.Rdata'))

source(paste0(sourcedir, '/','fun for lasso.R'))
source(paste0(sourcedir, '/','dataprocess_for_lasso.R'))

#model.matrix for lasso
X <- model.matrix(Ethnicity~., data.lasso)[,-1]
y <- data.lasso$Ethnicity

#num of samples for the train model and the prediction
ntr <- one_dataset$n
X_tr <- X[1:ntr,]
y_tr <- y[1:ntr]

## fit lasso with the lambda and  predictions
lasso.tr <- glmnet (x = X_tr, y= y_tr, nlambda=200,
                    family = "multinomial")
probs_pred_lasso.tr <- predict(lasso.tr, newx = X[1:one_dataset$n,],
                               type = "response")

#evaluate the performance for train model
lasso.nsel <- summary.nsel.lasso(lasso.tr, one_dataset$otu.num, cutoff=0)

#save the nsel for lasso.tr
file.remove(paste0(home, '/', method, 'out', '/', 'lasso.nsel', '.txt'))
cat (lasso.nsel, file = paste0(home, '/', method, 'out', '/',
     'lasso.nsel', '.txt'))

#save the lambda for loocv.lasso
file.remove(paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))
cat (lasso.tr$lambda, file = paste0(home, '/', method, 'out', '/',
     'lambda', '.txt'))

plot(lasso.tr, xvar="lambda", label=TRUE)
```

```
plot(lasso.tr, xvar="norm", label=TRUE)
plot(lasso.tr, xvar="dev", label=TRUE)

#save the 'probs_pred_lasso.tr' for plotting
file.remove(paste0(home, '/', method, 'out', '/',
            'probs_pred_lasso.tr', '.Rdata'))
save(probs_pred_lasso.tr, file = paste0(home, '/', method, 'out', '/',
     'probs_pred_lasso.tr', '.Rdata'))

########################Loocv for Lasso######################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/realdata/ethnicity"
setwd(home)

method <- "lasso"
probs_preddir <- paste0(home, '/', method,"out", '/', "probs_pred")

dir.create(probs_preddir,recursive = TRUE)

library(glmnet)

# irep <- 1 # this line will be added by qsubR with 1 taking 1,2,...
if (!exists("irep")) irep <- 1

#load the dataset and the lambda
load(paste0(home, '/', 'one_dataset.Rdata'))
lambda <- scan(paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))

source(paste0(sourcedir, '/','dataprocess_for_lasso.R'))
#model.matrix for lasso
X <- model.matrix(Ethnicity~., data.lasso)[,-1]
y <- data.lasso$Ethnicity

#num of samples for the train model and the prediction
ntr <- one_dataset$n
X_tr <- X[1:ntr,]
y_tr <- y[1:ntr]

# leave one out cross validation and predictions
probs_pred_lasso <- data.frame (matrix(0, one_dataset$l1, length(lambda)))

# fit the model
lasso.cvfit <- glmnet (x = X_tr[-irep,], y=y_tr[-irep],  lambda=lambda,
                       family = "multinomial")
```

```
# make predictions
probs_pred_lasso <- predict(lasso.cvfit, newx = t(X_tr[irep,]),
                            type = "response")[1,,]

## output
file.remove(paste0(probs_preddir, '/', 'probs_pred_lasso', irep, '.txt'))
cat (probs_pred_lasso, file = paste0(probs_preddir, '/', 'probs_pred_lasso',
     irep, '.txt'))

########################Compute the ER and AMLP##############
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/realdata/ethnicity"
setwd(home)

method <- "lasso"
probs_preddir <- paste0(home, '/', method,"out", '/', "probs_pred")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

source(paste0(sourcedir, '/','compred.r'))

#load the dataset
load(paste0(home, '/', 'one_dataset.Rdata'))

source(paste0(sourcedir, '/','fun for lasso.R'))
source(paste0(sourcedir, '/','dataprocess_for_lasso.R'))

#model.matrix for lasso
X <- model.matrix(Ethnicity~., data.lasso)[,-1]
y <- data.lasso$Ethnicity

#num of samples for the train model and the prediction
ntr <- one_dataset$n
X_tr <- X[1:ntr,]
y_tr <- y[1:ntr]

#load 'probs_pred_lasso.tr' , the lambda and the nsel for plotting
load (paste0(home, '/', method, 'out', '/', 'probs_pred_lasso.tr',
      '.Rdata'))
lambda <- scan(paste0(home, '/', method, 'out', '/', 'lambda', '.txt'))
lasso.nsel <- scan(paste0(home, '/', method, 'out', '/', 'lasso.nsel',
                  '.txt'))
probs_pred_lasso.cvfit <-  array(0, dim = c(ntr, one_dataset$l1,
                                length(lambda)))

for(k in 1:ntr)
```

```
{

  fn <- paste0(probs_preddir, '/', 'probs_pred_lasso', k, '.txt')

  if (file.exists(fn))
  {
    prob0 <- scan(paste0(probs_preddir, '/', 'probs_pred_lasso', k, '.txt'))
    prob <-  matrix (prob0, one_dataset$l1, length(lambda))
    probs_pred_lasso.cvfit[k,,] <- prob
  }
  else
  {
    cat (fn, "does not exsit\n")
  }
}

#calculate the ER & AMLP
amlp.tr = er.tr = amlp.cvfit = er.cvfit = rep(0, length(lambda))

for(l in 1:length(lambda))
{
  amlp.tr[l] <- evaluate_pred(probs_pred_lasso.tr[,,l],
                              y[1:one_dataset$n])$amlp
  er.tr[l] <- evaluate_pred(probs_pred_lasso.tr[,,l], y[1:one_dataset$n])$er

  amlp.cvfit[l] <- evaluate_pred(probs_pred_lasso.cvfit[,,l], y_tr)$amlp
  er.cvfit[l] <- evaluate_pred(probs_pred_lasso.cvfit[,,l], y_tr)$er
}

loglambda <- log(lambda)

## calculate expected amlp (entropy)

freq.eth <- table(data.lasso$Ethnicity)/nrow (data.lasso)
ex.amlp.enth <- -sum(freq.eth*log(freq.eth)); ex.amlp.enth

## expected error rate is the minority frequency
## (as we can always predict with mode)

ex.er.enth <- 1-max(freq.eth)


#plots for results of ER & AMLP
pdf(file=paste0(plotsdir, '/', 'ER-for-LASSO.pdf'))
par(mar = c(4, 4, 2.5, 1))
```

```
plot (loglambda, er.tr, lwd=4, type = "l", lty = 1,
      ylim = range(er.tr, er.cvfit),
      xlab="log(lambda)", ylab="ER for LASSO")
points (loglambda, er.cvfit, type = "l", lty = 3, col = 2, lwd=8)
title("ER for LASSO", line = 1.5)

#double axis
sequence <- seq(1,length(lambda), by=8)
xtick2 <- loglambda[sequence]
xlabel2 <- lasso.nsel[sequence]
axis(side = 3, padj = 1,
     at = xtick2 , label = xlabel2 )
mtext("Number of Retained Features", side = 3, line = -1)

legend("right", c("training", "loocv"), col = c(1, 2),
       text.col = "black", lty = c(1, 3))
abline(h = ex.er.enth, col = 3, lty = 2, lwd=4)
abline(h = seq(0, 1, by=0.05), col = "grey", lty = 2)
dev.off()

##AMLP
pdf(file=paste0(plotsdir, '/', 'AMLP-for-LASSO.pdf'))
par(mar = c(4, 4, 2.5, 1))
plot (loglambda, amlp.tr, lwd=4, type = "l", lty = 1,
      ylim = range(amlp.tr, amlp.cvfit),
      xlab="log(lambda)", ylab="AMLP")
points (loglambda, amlp.cvfit, type = "l", lty = 3, col = 2, lwd=8)
title("AMLP for LASSO", line = 1.5)

#double axis
sequence <- seq(1,length(lambda), by=8)
xtick2 <- loglambda[sequence]
xlabel2 <- lasso.nsel[sequence]
axis(side = 3, padj = 1,
     at = xtick2 , label = xlabel2 )
mtext("Number of Retained Features", side = 3, line = -1.2)

legend("right", c("training ", "loocv"), col = c(1, 2),
       text.col = "black", lty = c(1, 3))
abline(h = ex.amlp.enth, col = 3, lty = 2, lwd=4)
#abline(h = seq(0, 1, by=0.05), col = "black", lty = 2)
dev.off()

#########################Fit ZINB#################################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
```

```
home <- "/home/xiw378/canola/lassoglmm/realdata/ethnicity"
setwd(home)

method <- "zinb"
logpdir <- paste0(home, '/', method, "out", '/', "logp")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(logpdir,recursive = TRUE)
dir.create(plotsdir,recursive = TRUE)


library(glmmTMB)
source(paste0(sourcedir, '/','lrtest.R'))

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

load(paste0(home, '/', 'one_dataset.Rdata'))

ntr <- one_dataset$n
data.glmm <-one_dataset$dataset[1:ntr,]

data.glmm$y <- data.glmm[, one_dataset$index.otus[irep]]

## Zero-inflated negative binomial model
zinb <- glmmTMB( y ~ Ethnicity+Sex+(1|Age), data=data.glmm,
                 zi= ~ Ethnicity+Sex+(1|Age),
                 family = nbinom2)

zinb0 <- glmmTMB( y ~ Sex+(1|Age), data=data.glmm,
                  zi= ~ Sex+(1|Age),
                  family = nbinom2)


#log p_value  for each otu j
log.p.zinb <- lrtest.zi.model(zinb0,zinb)$log.pvalue

## output for log.pvalue
file.remove(paste0(logpdir, '/', 'log.p.zinb', irep , '.txt'))
cat (log.p.zinb, file = paste0(logpdir, '/', 'log.p.zinb', irep , '.txt'))

#########################Fit ZMP################################
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm//realdata/ethnicity"
setwd(home)
```

```
method <- "zmp"
logpdir <- paste0(home, '/', method, "out", '/', "logp")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(logpdir,recursive = TRUE)
dir.create(plotsdir,recursive = TRUE)


library(glmmTMB)
source(paste0(sourcedir, '/','lrtest.R'))

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

load(paste0(home, '/', 'one_dataset.Rdata'))

ntr <- one_dataset$n
data.glmm <-one_dataset$dataset[1:ntr,]

data.glmm$y <- data.glmm[, one_dataset$index.otus[irep]]

#glm
#logistic
hurdle.l<-glmmTMB((y>0)~Ethnicity+Sex+(1|Age),data=data.glmm,
                  ziformula =~0,family=binomial())
hurdle.l0<-glmmTMB((y>0)~Sex+(1|Age),data=data.glmm,
                   ziformula =~0,family=binomial())

#poisson
hurdle.poisson <-  glm(y ~ Ethnicity+Sex+Age+offset(logTR),
                       family  = "poisson",
                       data = subset(data.glmm,y>0))

hurdle.poisson0 <-  glm(y ~ Ethnicity+Sex+Age+offset(logTR),
                        family  = "poisson",
                        data = subset(data.glmm,y>0))

log.p.zmp <- lrtest.hurdle(hurdle.l0, hurdle.poisson0, hurdle.l,
                           hurdle.poisson)$log.pvalue

## output
file.remove(paste0(logpdir, '/', 'log.p.zmp', irep , '.txt'))
cat (log.p.zmp, file = paste0(logpdir, '/', 'log.p.zmp', irep , '.txt'))

########################Fit NB#################################
```

```
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode/generators"
home <- "/home/xiw378/canola/lassoglmm/realdata/ethnicity"
setwd(home)

method <- "nb"
logpdir <- paste0(home, '/', method, "out", '/', "logp")
plotsdir <- paste0(home, '/', method,"out", '/', "plots")

dir.create(logpdir,recursive = TRUE)
dir.create(plotsdir,recursive = TRUE)


library(glmmTMB)
source(paste0(sourcedir, '/','lrtest.R'))

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

load(paste0(home, '/', 'one_dataset.Rdata'))

ntr <- one_dataset$n
data.glmm <-one_dataset$dataset[1:ntr,]

data.glmm$y <- data.glmm[, one_dataset$index.otus[irep]]


nb <- glmmTMB( y ~ Ethnicity+Sex+(1|Age), data=data.glmm,
               family = nbinom2)

nb0 <- glmmTMB( y ~ Sex+(1|Age), data=data.glmm,
                family = nbinom2)

log.p.nb <- lrtest.zi.model(nb0,nb)$log.pvalue

## output for log.pvalue
file.remove(paste0(logpdir, '/', 'log.p.nb', irep , '.txt'))
cat (log.p.nb, file = paste0(logpdir, '/', 'log.p.nb', irep , '.txt'))

########################Plot the p_value and q_value#########
sourcedir <- "/home/xiw378/canola/lassoglmm/Rcode"
home <- "/home/xiw378/canola/lassoglmm/realdata/ethnicity"
setwd(home)
##
plotscomdir <- paste0(home, '/', "comparisons")
dir.create(plotscomdir,recursive = TRUE)
```

```
method <- c("zinb","zmp","nb", "lasso")
label<- c("ZINB", "ZMP", "NB", "LASSO")

load(paste0(home, '/', 'one_dataset.Rdata'))
source(paste0(sourcedir, '/', 'generators', '/', 'fun for lasso.R'))
source(paste0(sourcedir, '/', 'evaluation', '/', 'evalp.R'))

#get all the log(p)
log.p <- matrix(NA, one_dataset$otu.num, length(method)-1)
order.log.p <- matrix(NA, one_dataset$otu.num, length(method)-1)
num.cutoff = 143
model.eval <- as.list(1:length(method))


for(i in 1 : (length(method)-1))
{
  logpdir <- paste0(home, '/', method[i], "out", '/', "logp")
  log.p[,i] <- extract.p(logpdir, method[i], one_dataset$otu.num)

  #save the num of NA's
  num.na <- sum(is.na(log.p[,i]))

  ##use 1 instead all the NA's
  log.p[,i][which(is.na(log.p[,i]) == TRUE)] <- 0

  #use all the means for each two adjacent p_values as cutoffs
  order.log.p0 <- order(log.p[,i])
  order.log.p[,i] <- log.p[,i][order.log.p0]

  log.p0.mean <- zoo::rollmean(order.log.p[,i], 2)
  cutoff <- log.p0.mean[1:num.cutoff]

  ##calculate the estimated fdr for the model
  estimated.fdr <- p.adjust(exp(order.log.p[,i]),
                            method = "fdr")[1:length(cutoff)]

  eval.sel.model <- list(estimated.fdr= NA, cutoff= NA, num.na= NA)

  eval.sel.model$estimated.fdr <- estimated.fdr
  eval.sel.model$cutoff <- cutoff
  eval.sel.model$num.na <- num.na

  model.eval[[i]] <- eval.sel.model

  plotsdir <- paste0(home, '/', method[i],"out", '/', "plots")
```

```
  #Histogram of p-value
  pdf(file=paste0(plotsdir, '/', 'p_value-for-', method[i],'.pdf'))
  hist (exp(log.p[,i][log.p[,i]<0]), breaks = seq(0,1, length=20),
        xlab="p-value",
        main= paste0('Histogram of p-value for ', label[i]))
  dev.off()

}


#comparison
plotscomdir <- paste0(home, '/', "comparisons")

#log(p-value)
  pdf(file=paste0(plotscomdir, '/', '-log(p-value)-for-',
      'ZINB-ZMP-and-NB', '.pdf'))

  par(mar = c(4, 4, 2, 1))
  #make the plot
  plot(-order.log.p[,1],
       lwd=4,
       type = "p",  col = 1, pch = 1,  log = "x",
       xlab="Number of Retained Features",
       ylab="",
       ylim = range(na.omit(c(-order.log.p[,1],-order.log.p[,2],
                              -order.log.p[,3]))),
       main = paste0('-log(p-value) for ZINB, ZMP and NB')
       )

  points(-order.log.p[,2], lwd=4,
         type="p", col=2, pch = 2)

  points(-order.log.p[,3], lwd=4,
         type="p", col=3, pch = 3)

  legend("right", toupper(c(method[1], method[2], method[3])),
         col = c(1,2,3), text.col = "black", pch = c(1,2,3))
  dev.off()


##q-value#################################################
  pdf(file=paste0(plotscomdir, '/', 'q-value-for-',
      'ZINB-ZMP-and-NB', '.pdf'))

  par(mar = c(4, 4, 2, 1))
```

```
#make the plot
plot(model.eval[[1]]$estimated.fdr,
     lwd=4,
     type = "l",  col = 1, lty = 1,  log = "x",
     xlab="Number of Retained Features", ylab="",
     ylim = range(na.omit(c(model.eval[[1]]$estimated.fdr,
                            model.eval[[2]]$estimated.fdr,
                            model.eval[[3]]$estimated.fdr))),
     main = paste0('q-value for ZINB, ZMP and NB')
)

points(model.eval[[2]]$estimated.fdr, lwd=4,
       type="l", col=2, lty = 2)

points(model.eval[[3]]$estimated.fdr, lwd=4,
       type="l", col=3, lty = 3)

legend("topleft", toupper(c(method[1], method[2], method[3])),
       col = c(1,2,3), text.col = "black", lty = c(1,2,3))
dev.off()
```