2019-01-01

On The Performance Of Variable Selection And Classification Via Rank-Based Classifier

Md Showaib Rahman None Sarker
University of Texas at El Paso

ON THE PERFORMANCE OF VARIABLE SELECTION AND CLASSIFICATION VIA RANK-BASED CLASSIFIER

MD SHOWAIB RAHMAN SARKER

Master's Program in Mathematical Sciences

APPROVED:

Sangjin Kim, Ph.D., Chair

Amy Wagler, Ph.D.

Lin Li, Ph.D.

Xiaogang Su, Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

to my

MOTHER and SISTER

with love

ON THE PERFORMANCE OF VARIABLE SELECTION AND CLASSIFICATION VIA RANK-BASED CLASSIFIER

by

MD SHOWAIB RAHMAN SARKER

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mathematical Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

August 2019

Acknowledgements

With profound warmth, I would like to express my deep-felt gratitude to my advisor Dr. Sangjin Kim of the Department of Mathematical Sciences at the University of Texas at El Paso, who made me a part to UTEP family. His clarity of thoughts, insights and simplicity in exposition will remain inspirational to me. Beyond introducing me to beautiful ideas and teaching me techniques to make them precise, he has shown an exquisite sense for how to nurture me towards becoming an independent researcher, by daydreaming with me about what might be true, by giving me the space to slowly grope towards realizing those daydreams, and by offering the crucial insights when I need guidance. His critical comments, timely feedback, stimulating suggestions, and commitment to my writing made this thesis what it is. His relentless support and instructions made possible to publish this research work as a journal paper.

I also wish to extend my gratitude to my committee members, Dr. Amy Wagler, Associate chairman of the Mathematical Science Department, Dr. Xiaogang Su of the Mathematical Science Department, and Dr. Lin Li of the Physics Department,all of the University of Texas at El Paso.Their suggestions, comments and additional guidance were invaluable to the completion of this work.

I wish to thank Dr. Michael Pokojovy, Dr. Naijun Sha, Dr. Ori Rosen, all of the Department of Mathematical Sciences at the University of Texas at El Paso and staffs of Mathematical Science Department of The University of Texas at El Paso for all their hard work and dedication, providing me the means to complete my degree and prepare for a career as a Statistician.

Through all my endeavors, mathematical and otherwise, I am sustained by the love and trust of my family. In my mom, I found my intellectual role model and mentor. She taught me the joy of explaining an idea clearly, to ask for whats missing in my current point of view, and to have faith in my intuition but test its suggestions thoroughly. Beyond the intellectual, I admire and seek to emulate her devotion to the quirky joke and the thoughtful gesture. Through her deep engagement with my life, I am lucky never to have felt unloved or unimportant. Finally, I thank my father, who is my original teacher, interlocutor, and collaborator. My parent's unconditional support and love have allowed me to thrive and grow into an adult and Statistician.

NOTE: This thesis was submitted to my Supervising Committee on the June 1, 2019.

Abstract

In high-dimensional gene expression data analysis, the accuracy and reliability of cancer classification and selection of important genes play a very crucial role. To identify these important genes and predict future outcomes (tumor vs. non-tumor), various methods have been proposed in the literature. But only few of them take into account correlation patterns and grouping effects among the genes. In this article, we propose a rank-based modification of the popular penalized logistic regression procedure based on a combination of ℓ_1 and ℓ_2 penalties capable of handling possible correlation among genes in different groups. While the ℓ_1 penalty maintains sparsity, the ℓ_2 penalty induces smoothness based on the information from the Laplacian matrix, which represents the correlation pattern among genes. We combined logistic regression with the BH-FDR (Benjamini and Hochberg false discovery rate) screening procedure and a newly developed rank-based selection method to come up with an optimal model retaining the important genes. Through simulation studies and real-world application to high-dimensional colon cancer gene expression data, we demonstrated that the proposed rank-based method outperforms such currently popular methods as lasso, adaptive lasso and elastic net when applied both to gene selection and classification.

keywords: gene-expression data; ℓ_2 ridge; ℓ_1 lasso; adapative lasso; elastic net; BH-FDR; Laplacian matrix

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Microarray Gene Expression Data

Microarray technology has become one of the indispensable tools that many biologists use to monitor genome wide expression levels of genes in a given organism. A microarray is typically a glass slide on to which DNA molecules are fixed in an orderly manner at specific locations called spots (or features). A microarray may contain thousands of spots and each spot may contain a few million copies of identical DNA molecules that uniquely correspond to a gene. The DNA in a spot may either be genomic DNA or short stretch of oligo-nucleotide strands that correspond to a gene. The spots are printed on to the glass slide by a robot or are synthesized by the process of photolithography. Microarrays may be used to measure gene expression in many ways, but one of the most popular applications is to compare expression of a set of genes from a cell maintained in a particular condition (condition A) to the same set of genes from a reference cell maintained under normal conditions (condition B). First, RNA is extracted from the cells. Next, RNA molecules in the extract are reverse transcribed into cDNA by using an enzyme reverse transcriptase and nucleotides labelled with different fluorescent dyes. For example, cDNA from cells grown in condition A may be labelled with a red dye and from cells grown in condition B with a green dye. Once the samples have been differentially labelled, they are allowed to hybridize onto the same glass slide. At this point, any cDNA sequence in the sample will hybridize to specific spots on the glass slide containing its complementary sequence. The amount of cDNA bound to a spot will be directly proportional to the initial number of RNA molecules present for that gene in both samples. Following the hybridization step, the spots

in the hybridized microarray are excited by a laser and scanned at suitable wavelengths to detect the red and green dyes. The amount of fluorescence emitted upon excitation corresponds to the amount of bound nucleic acid. For instance, if cDNA from condition A for a particular gene was in greater abundance than that from condition B, one would find the spot to be red. If it was the other way, the spot would be green. If the gene was expressed to the same extent in both conditions, one would find the spot to be yellow, and if the gene was not expressed in both conditions, the spot would be black. Thus, what is seen at the end of the experimental stage is an image of the microarray, in which each spot that corresponds to a gene has an associated fluorescence value representing the relative expression level of that gene. Tens of thousands of genes can be analyzed simultaneously with this microarray [Houwelingen1(2006)]. This technology is widely used to distinguish between normal (tumor) and (non-tumor) cancerous tissue samples and support cancer diagnosis.

1.2 Curse of High-Dimensionality and Correlation

Identifying the genes related to cancer and building high-performance prediction models of maximal accuracy (tumor vs. non-tumor) based on gene expression levels are among central problems in genomic research [Lofti(2014), Algamal(2015), Li(2008)].Typically, in high-dimensional gene expression data analysis, the number of genes is significantly larger than the sample size, i.e., $m \gg n$. This is called curse of high-dimensionality in high-dimensional gene expression data.

Hence, it is particularly challenging to overcome the curse of high-dimensionality by identifying those genes that are relevant to cancer disease and put forth prediction models. The main problem associated with high-dimensional data ($m \gg n$) is that of overfitting or overparametrization which leads to poor generalizability from training to test data. Researchers are applying different types of regularization methods to handle the curse of high-dimensionality ($m >> n$) in regression frameworks. These include Lasso(Tibshirani,1996),

Adaptive Lasso(Zou,2006), Fused Lasso (Tibshirani,Saunders,2005), SCAD(Fan,Li,2001), Elastic net(Zou,Hastie,2005) etc.

But only a few methods take into consideration the correlation pattern and grouping effects among genes.Motivated by network-constrained regularization method (Li,Li,2008,10) and penalized logistic regression for case-control study (Sun,Wang,2012) we proposed a rank-based penalized logistic regression model to handle the correlation among genes in different groups.

1.3 Outline of the thesis

The remaining parts of the thesis are organized in this manner. Chapter 2 provides a literature review on screening, variable selection, and classification via different regularization methods in details. In Chapter 3, we describe variable screening procedure with adjusted p-values and regularization procedure for grouped and correlated predictors and present the computational algorithm. Further, we state the ranking criteria of four models and summarize the result of ranking procedure. In Chapter 4, we compare the proposed procedure with existing cutting edge regularization methods on simulation studies. In Chapter 5, we apply four penalized logistic regression methods to the high dimensional gene expression data of colon cancer carcinoma to evaluation and comparison of the performance. Finally, we present a brief discussion of results and future research direction in Chapter 6.

Chapter 2

Literature Review

In this chapter we provide a brief discussion to the different existing regularization techniques of variable selection and prediction. Then we present a literature review on regularization methods which incorporate the information from the Laplacian matrix. Limitations of those methods in high-dimensional settings are also addressed.

2.1 Variable Selection and Prediction via Existing Regularization Methods

During the last decade, the motivation for applying feature selection (FS) techniques in bioinformatics has shifted from being an illustrative example to becoming a real prerequisite for model building. In particular, the high dimensional nature of many modelling tasks in bioinformatics, going from sequence analysis over microarray analysis to spectral analyses and literature mining has given rise to a wealth of feature selection techniques being presented in the field [Sayes(2007)].

Therefore, various researchers apply different types of regularization methods to overcome this "curse of dimensionality" in regression and other statistical and machine learning frameworks. These regularization approaches include, for example, the ℓ_1-penalty or lasso [Tibshirani(1996)], which performs continuous shrinkage and feature selection simultaneously; smoothly clipped ℓ_1-penalty or SCAD [Li(2001)], which is symmetric, non-concave and has singularities at the origin to produce sparse solutions; fussed lasso [Tibshirani(2005)], which imposes the ℓ_1-penalty on the absolute difference of regression

coefficients in order to enforce some smoothness of coefficients; or the adaptive lasso [Zou(2006)], etc. Unfortunately, ℓ_1-regularization sometimes perform inconsistently when used for variable selection [Zou(2006)]. In some situations, it introduces a major bias in estimated parameters in the logistic regression [Meinshausen(2009), Huang(2016)]. In contrast, the elastic net regularization procedure [Zou(2005)] as a combination of ℓ_1- and ℓ_2-penalties can successfully handle the highly correlated variables which are grouped together. Among the procedures mentioned above, elastic net and fussed lasso penalized methods are appropriate for gene expression data analysis. Unfortunately, when some prior knowledge needs to be utilized, e.g., when studying complex diseases such as cancer, those methods are not appropriate [Li(2008)].

2.2 Regularization with Laplacian Matrix

Of particular interest are gene-regulatory pathways that provide regulatory relationships between genes or gene products. These pathways are often interconnected and form a network, which can be represented as graphs, where the vertices of the graphs are genes or gene products and the edges of the graphs indicate some regulatory relationship between the genes. This kind of a priori information is a useful supplement to the standard numerical data coming from an experiment. Incorporating the information from these graphs into an analysis of the numerical data is a non-trivial task that is generating increasing interest.Several statistical methods have been developed to utilize the pathways or network information, including the hidden Markov-random field approaches to utilize the network structures in identifying the differentially expressed genes (Wei and Li, 2007, 2008; Wei and Pan, 2008). Rahnenfhrer et al. (2004) demonstrated that the sensitivity of detecting relevant pathways can be improved by integrating information about pathway topology. However, none of these methods were developed in the framework of regression analysis.

To account for a regulatory relationship between the genes and a priori knowledge about these genes,network-constrained regularization [Li(2008)] is known to perform very well by

incorporating a Laplacian matrix into the ℓ_2-penalty from the enet procedure. This Laplacian matrix represents a graph-structure of genes which are linked with each other.To select significant genes in high-dimensional gene expression data for classification, the graph-constrained regularization method is extended to logistic regression model [Sun(2012)]. Such a procedure can select subgroups of correlated features in the network, thus enjoying global smoothness over the network. This procedure, which includes the elastic net regulation procedure as a special case, is similar in spirit to the fused-lasso [Tibshirani(2005)]. It induces smoothed coefficient profiles, which can result in more interpretable identification of genes and subnetworks that are related to the responses in the context of known biology. However, it is different from fused-lasso in that this procedure does not requires that the neighboring genes to have the same coefficients and the network-structure is explicitly modeled using the Laplacian matrix of the graph.Using penalized logistic regression methods [Sun(2012), Sun(2012)] and graph-constrained procedures [Li(2008), Sun(2012)], we would build rank-based logistic regression method with variable screening procedure to improve the power of detecting most promising variables as well as classification capability.

Chapter 3

Methodology

3.1 Adjusted p-Values: Benjamini and Hochberg False Discovery Rate (BH-FDR)

Multiple hypohtesis testing methods have been playing an important role in selecting most promising features while controlling type I error in high-dimensional settings. One of the most popular methods is BH-FDR [Reiner(2003), Benjamin(1995)] which is concerned with the expected proportion of incorrect number of rejections among a total number of rejections. The formula is mathematically expressed as $E\left(\frac{V}{R}\middle| R > 0\right)$, where V is the number of false positives and R is the total number of rejections. In this paper, the FDR method is used both for the purpose of prelimary variable screening both in the simulation studies and real data analysis to be presented later. The procedure of the method is as follows:

(1) Let $p_1, p_2, \ldots, p_m$ be the p-values of m hypothesis tests and sort them with the increasing oder: $p_{(1)}, p_{(2)}, \ldots, p_{(m)}$.

(2) Let $\hat{i} = \max\{i \,|\, p_{(i)} \leq \frac{iq}{m}, i = 1, \ldots, m\}$ for a given threshold q. If $\hat{i} > 1$, then reject the null hypotheses associated with $p_{(1)}, p_{(2)}, \ldots, p_{(i)}$. Otherwise, no hypotheses are rejected.

3.1.1 Example

Thrombolysis with recombinant tissue-type plasminogen activator (rt-PA) and anisoylated plasminogen streptokinase activator (APSAC) in myocardial infarction has been proved to reduce mortality. Neuhaus et al. (1992) investigated the effects of a new front-loaded administration of rt-PA versus those obtained with a standard regimen of APSAC, in a randomized multicentre trial in 421 patients with acute myocardial infarction. Four families of hypotheses can be identified in the study:

(1) base-line comparisons (11 hypotheses), where the problem is of showing equivalence;

(2) patency of infarct-related artery (eight hypotheses);

(3) reocclusion rates of patent infarct-related artery (six hypotheses);

(4) cardiac and other events after the start of thrombolitic treatment (15 hypotheses).

In this last family FDR control may be desired: we do not wish to conclude that the front-loaded treatment is better if it is merely equivalent to the previous treatment in all respects. In the paper, however, there is no attention to the problem of multiplicity (the only exception being the division of the end points into primary and secondary). The individual p-values are reported as they are, with no word of warning regarding their interpretation. The authors conclude that 'Compared to APSAC treatment, despite more early reocclusions, the clinical course with rt-PA treatment is more favorable with fewer bleeding complications and a substantially lower in-hospital mortality rate, presumably due to improved early patency of the infarct- related artery'.

The statement about the mortality is based on a p-value of 0.0095.Consider now the fourth family, which contains the comparison of mortality and 14 other comparisons.

The ordered $p_{(i)}s$ for the 15 comparisons made are 0.0001, 0.0004, 0.0019, 0.0095, 0.0201, 0.0278, 0.0298, 0.0344, 0.0459, 0.3240, 0.4262, 0.5719, 0.6528, 0.7590, 1.000.

Controlling the FWER at 0.05, the Bonferroni approach, using $0.05/15 = 0.0033$, rejects the three hypotheses corresponding to the smallest p-values. These hy- potheses correspond

to reduced allergic reaction, and to two different aspects of bleeding; they do not include the comparison of mortality. Using Hochberg's procedure leaves us with the same three hypotheses rejected. Thus the statement about a significant reduction in mortality is unjustified from the classical point of view.

Using the FDR controlling procedure with $q^*= 0.05$, we now compare sequen- tially each $p_{(i)}$ with $(0.05 * i)/15$, starting with $p_{(15)}$. The first p-value to satisfy the constraint is $p_{(4)}$as,

$p_{(4)} = 0.0095 \leq \frac{4}{15} * 0.05 = 0.013.$

Thus we reject the four hypotheses having p-values which are less than or equal to 0.013. We may support now with appropriate confidence the statements about mortality decrease, of which we did not have sufficiently strong evidence before.

3.2 Regularized Logistic Regression

3.2.1 The Lasso

Suppose that we have data $(X^i; y_i), i = 1, 2, ...N$ where $X^i = (x_{i1}, \cdots, x_{im})$ are the predictor variables and yi are the responses. As in the usual regression setup, we assume that either that the observations are independent or that the $y_i)s$ are conditionally independent given the $x_{ij}s$. We assume that the $x_{ij}$are standardized so that $\sum_i \frac{x_{ij}}{N} = 0, \sum_i \frac{x_{ij}^2}{N} = 1$.

The parameter vector $\boldsymbol{\eta} = (\boldsymbol{\beta}_0, \boldsymbol{\beta})$ comprised of an intercept β_0 and m "slopes", $\beta_1, \ldots, \beta_m$. The lasso estimate $(\hat{\boldsymbol{\beta}}_{\mathbf{0}}, \hat{\boldsymbol{\beta}})$ is defined by

$$(\hat{\boldsymbol{\beta}}_{\mathbf{0}}, \hat{\boldsymbol{\beta}}) = argmin \sum_{i=1} N(y_i - \alpha - \sum_j \boldsymbol{\beta_j} x_{ij})^2$$
$$\text{subject to} \sum_j |\boldsymbol{\beta_j}| \leq t \tag{3.1}$$

Here $t \geq 0$ is a tuning parameter. Now for all t, the solution for $\boldsymbol{\beta}_0$ is $\hat{\boldsymbol{\beta}}_{\mathbf{0}} = \bar{y}$. We can assume without loss of generality that $\bar{y} = 0$ and hence omit $\boldsymbol{\beta}_0$. Computation of the

solution to (3.1) is a quadratic programming problem with linear inequality constraints. The parameter $t \geq 0$ controls the amount of shrinkage that is applied to the estimates.

3.2.2 The Adaptive Lasso

The lasso cannot be an oracle procedure. However, the asymptotic setup is somewhat unfair, because it forces the coefficients to be equally penalized in the ℓ_1 penalty.We can certainly assign different weights to different coefficients. Let us consider the weighted lasso,

$$\arg\min_{\beta} ||y_i - \alpha - \sum_j \boldsymbol{\beta_j} x_{ij}||^2 + \lambda \sum_j w_j |\boldsymbol{\beta_j}|,$$

where w is a known weights vector. If the weights are data-dependent and cleverly chosen, then the weighted lasso can have the oracle properties. The new methodology is called the adaptive lasso. We now define the adaptive lasso. Suppose that $\hat{\boldsymbol{\beta}}$ is a rootnconsistent estimator to $\boldsymbol{\beta^*}$; for example, we can use $\hat{\boldsymbol{\beta}}(ols)$. Pick a $\gamma \geq 0$, and define the weight vector $\hat{w} = \frac{1}{|\hat{\boldsymbol{\beta}}|^\gamma}$. The adaptive lasso estimates $\hat{\boldsymbol{\beta}}^{*(n)}$ are given by

$$\hat{\boldsymbol{\beta}}^{*(n)} = \arg\min_{\beta} ||y_i - \alpha - \sum_j \boldsymbol{\beta_j} x_{ij}||^2 + \lambda_n \sum_j \hat{w}_j |\boldsymbol{\beta_j}| \tag{3.2}$$

Similarly, let $\{{A_n}^* = j : \hat{\boldsymbol{\beta}}_j^{*(n)} \neq 0\}$.

It is worth emphasizing that (3.2)is a convex optimization problem, and thus it does not suffer from the multiple local minimal issue, and its global minimizer can be efficiently solved.This is very different from concave oracle penalties. The adaptive lasso is essentially an ℓ_1 penalization method. We can use the current efficient algorithms for solving the lasso to compute the adaptive lasso estimates.

3.2.3 The Elastic net

After a location and scale transformation from same data, we can assume that the response is centred and the predictors are standardized,

$$\sum_{i=1}^{n} y_i = 0, \sum_{i=1}^{n} x_{ij} = 0, \text{and} \sum_{i=1}^{n} x_{ij}^2 = 1, \text{j=1,2,.....,m.} \tag{3.3}$$

For any fixed non-negative λ_1 and λ_2, we define the naive elastic net criterion

$$L(\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \boldsymbol{\beta}) = |y - X\boldsymbol{\beta}|^2 + \lambda_2|\boldsymbol{\beta}|^2 + \lambda_1|\boldsymbol{\beta}|_1, \tag{3.4}$$

where

$$|\boldsymbol{\beta}|^2 = \sum_{j=1}^{m} \boldsymbol{\beta}_j^2,$$

$$|\boldsymbol{\beta}|_1 = \sum_{j=1}^{m} |\boldsymbol{\beta}_j|,$$

The naive elastic net estimator $\hat{\boldsymbol{\beta}}$ is the minimizer of equation (3.4):

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} \{L(\boldsymbol{\lambda_1}, \boldsymbol{\lambda_2}, \boldsymbol{\beta})\} \tag{3.5}$$

This procedure can be viewed as a penalized least squares method. Let $\alpha = \frac{\lambda_2}{(\lambda_1+\lambda_2)}$; then solving $\hat{\boldsymbol{\beta}}$ in equation(3.4) is equivalent to the optimization problem

$$\hat{\boldsymbol{\beta}} = \arg\min_{\beta} |y - X\boldsymbol{\beta}|^2, \text{subject to } (1-\alpha)|\boldsymbol{\beta}|_1 + \alpha|\boldsymbol{\beta}|^2 \leq t \text{ for some t} \tag{3.6}$$

We call the function $(1-\alpha)|\boldsymbol{\beta}|_1 + \alpha|\boldsymbol{\beta}|^2$ the elastic net penalty, which is a convex combination of the lasso and ridge penalty. When $\alpha = 1$, the naive elastic net becomes simple ridge regression. we consider only $\alpha < 1$. For all $\alpha \epsilon [0, 1)$, the elastic net penalty function is singular (without first derivative) at 0 and it is strictly convex for all $\alpha > 0$, thus having the characteristics of both the lasso and ridge regression. Note that the lasso penalty ($\alpha = 0$) is convex but not strictly convex.

3.2.4 The Graph-Constrained Regularized Logistic Regression

In the following, we present the regularized logistic regression model used in this paper (cf. [Sun(2012)]). Since this model is an integral part of our computational algorithm to be outlined in the section to follow, presenting the formula with all appropriate notations is necessary for our purposes.

Let the $n \times (m+1)$ matrix

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1j} & \cdots & x_{1m} \\ 1 & x_{21} & x_{22} & \cdots & x_{2j} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{i1} & x_{i2} & \cdots & x_{ij} & \cdots & x_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nj} & \cdots & x_{nm} \end{pmatrix}$$

denote the design matrix, where n is the sample size and m is the total number of predictor variables. Without loss of generality, we assume the data are standardized with respect to each variable. This step is also performed by the `pclogit` `R`-package used in the present paper. Define the parameter vector $\boldsymbol{\eta} = (\boldsymbol{\beta}_0, \boldsymbol{\beta})$ comprised of an intercept β_0 and m "slopes", $\beta_1, \ldots, \beta_m$. The objective function then is written as

$$f(\boldsymbol{\eta}) = -L(\boldsymbol{\eta}) + p(\boldsymbol{\beta}) \tag{3.7}$$

with the log-likelihood function

$$L(\boldsymbol{\eta}) = \frac{1}{n} \sum_{i=1}^{n} \left[y_i \log \pi(\boldsymbol{x}_i) + (1 - y_i) \log(1 - \pi(\boldsymbol{x}_i)) \right]$$

and resulting probabilities

$$\pi(\boldsymbol{x}_i) = \frac{\exp(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta})}{1 + \exp(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta})}.$$

Here, $p(\boldsymbol{\beta})$ is the penalty function and the response variable y_i takes the value 1 for cases and 0 for controls. The i-th individual is deemed case or control based on the probability π_i. Following [Li(2008)], statistical dependence among the m explanatory variables can

be modeled by a graph, which, in turn, can be described by its m-dimensional Laplacian matrix $\boldsymbol{L} = \big(L(u,v) \,|\, u, v \text{ vertices}\big)$ with the entries

$$L(u,v) = \begin{cases} 1, & \text{if } u = v \text{ and } d_u \neq 0, \\ -(d_u d_v)^{-\frac{1}{2}}, & \text{if } u \text{ and } v \text{ are adjacent}, \\ 0, & \text{otherwise.} \end{cases}$$

Here, d_v is the degree of a vertex v, i.e., the number of edges through this vertex. If there is no link in v (i.e., v is isolated), then $d_v = 0$. The martix L is symmetric, positive semi-definite and has 0 as the smallest eigenvalue and 2 as the largest eigenvalue. In the following, we will write $u \sim v$ to refer to adjecent vertices. The penalty term in equation (3.7) can is defined as

$$p(\boldsymbol{\beta}) = \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \boldsymbol{\beta}^T \boldsymbol{L} \boldsymbol{\beta} = \lambda_1 \sum_{j=1}^{m} |\beta_j| + \lambda_2 \sum_{u=1}^{m} \sum_{u \sim v} \Big(\frac{\beta_u}{\sqrt{d_u}} - \frac{\beta_v}{\sqrt{d_v}} \Big)^2. \tag{3.8}$$

Here, λ_1 and λ_2 are tuning parameters meant to control the sparsity and smoothness, $\|\boldsymbol{\beta}\|_1$ is the ℓ_1-norm and $\sum_{u \sim v}(\dots)$ denotes the summation over all adjacent vertex pairs. When $\lambda_2 = 0$, the penalty reduces to that of lasso [Tibshirani(1996)], and if $\boldsymbol{L}$ is replaced by the $m \times m$-identity matrix $\boldsymbol{I}$, the penalty corresponds to that of an elastic net [Zou(2005)]. If $\lambda_1 = 0$ and $\boldsymbol{L} = \boldsymbol{I}$, we arrive at ridge regression. In Equation (3.8), the penalty consists of ℓ_1- and ℓ_2-components. The ℓ_2-penalty is a degree-scaled difference of coefficients between linked predictors. According to [Li(2008)], the predictor variables with more connections have larger coefficients. That is why small change of expression in the variables can lead to large change in response. Thus, this imposes sparsity and smoothness as well as correlation and grouping effects among variables. In case-control DNA methylation data analysis, ring networks and fully connected networks (cf. Figure 3.1) are typically used to describe correlation pattern of CpG sites within genes [Sun(2012)]. The Laplacian matrix is sparse and tri-diagonal (except for two corner elements) for ring networks and has all non-zero elements for fully connected networks. Those variables with more links produce strong grouping effects and are more likely to be selected in both networks [Sun(2012)].

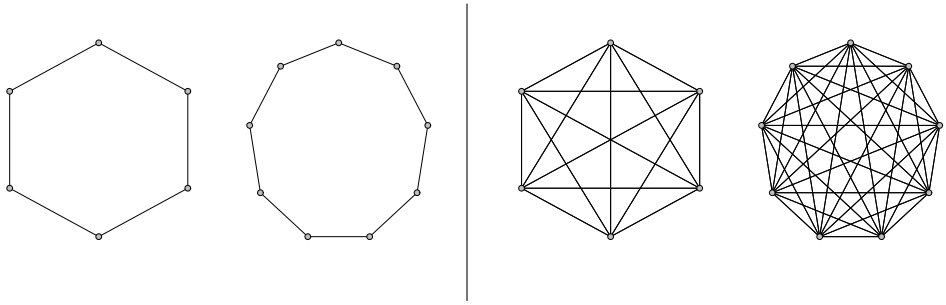

Figure 3.1. The ring network (left) and F.con network (right) are shown for the case there are two genes consisting of 6 and 9 CpG sites, respectively.

3.3 Computational Algorithm

Li & Li (2010) [Li(2010)] developed an algorithm for graph-constrained regularization motivated by a coordinate descent algorithm from [Friedman(2007)] for solving the unconstrained minimization problem for the objective in Equation (3.7). The algorithm implementation from the `pclogit` R-package [Sun(2012), Sun(2012)] replaced the identity matrix by Laplacian matrix in the elastic net algorithm from the `glmnet` R-package [Friedman(2010)]. According to Equation (3.7), the objective function is

$$f(\boldsymbol{\eta}) = -L(\boldsymbol{\eta}) + p(\boldsymbol{\beta}),$$

where

$$p(\boldsymbol{\beta}) = \lambda\alpha \sum_{i=1}^{m} |\beta_i| + \frac{1}{2}\lambda(1-\alpha) \sum_{u=1}^{m} \sum_{u\sim v} \Big(\frac{\beta_u}{\sqrt{d_u}} - \frac{\beta_v}{\sqrt{d_v}} \Big)^2 \tag{3.9}$$

with $\lambda = \lambda_1 + 2\lambda_2$ and $\alpha = \frac{\lambda_1}{\lambda_1 + 2\lambda_2}$ for some $\lambda_1, \lambda_2 > 0$.

Following [Friedman(2010)], we perform a second-order Taylor expansion of $L(\cdot)$ around the current estimate $(\beta_0^*, \boldsymbol{\beta}^*)$ to approximate the objective $L(\cdot)$ in Equation (3.7) via

$$f^*(x) = -\frac{1}{2n} \sum_{i=1}^{n} q_i (t_i - \beta_0 - x_i^T \boldsymbol{\beta})^2 + p(\boldsymbol{\beta}),$$

where

$$
\begin{aligned}
t_i &= \beta_0^* + \boldsymbol{x}_i^T \boldsymbol{\beta}^* + q_i^{-1}\big(y_i - \pi^*(\boldsymbol{x}_i)\big), \\
q_i &= \pi^*(\boldsymbol{x}_i)\big(1 - \pi^*(\boldsymbol{x}_i)\big), \\
\pi^*(\boldsymbol{x}_i) &= 1 - \big(1 + \exp(\beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}^*)\big)^{-1}.
\end{aligned}
$$

Now, if all other estimates for all $v = u$ are fixed, $\beta_u = \beta_u^*$ can be computed. To update the estimate from β_u^*, we have to set the gradient of $f^*(\cdot)$ equal zero (strictly speaking, zero has to be included in the subgradient of $f^*(\cdot)$) and then solve for β_u to obtain

$$
\beta_u^* = \frac{s(\frac{1}{n}\sum_{i=1}^n q_i \boldsymbol{x}_{iu}(t_i - t_i^{(\tilde{u})}) + \lambda(1-\alpha)g(u), \lambda\alpha)}{\frac{1}{n}\sum_{i=1}^n q_i \boldsymbol{x}_{iu}^2 + \lambda(1-\alpha)},
$$

where

$$
\begin{aligned}
t_i^{(\tilde{u})} &= \beta_0^* + \sum_{j \neq u} x_{ij}\beta_j^*, \\
g(u) &= \sum_{u \sim v} \frac{\beta_v^*}{\sqrt{d_u d_v}}
\end{aligned}
\tag{3.10}
$$

and $s(z, r)$ denotes the "soft threshholding" operator given by

$$
s(z,r) = \operatorname{sign}(z)(|z| - r)_+ = \begin{cases} z - r, & \text{if } z > 0 \text{ and } r < |z|, \\ z + r, & \text{if } z < 0 \text{ and } r < |z|, \\ 0, & \text{otherwise.} \end{cases}
$$

If the u-th predictor has no links to other predictors, then $g(u)$ in Equation (3.10) becomes zero, while Equation (3.9) takes the form

$$
p(\boldsymbol{\beta}) = \lambda\alpha \sum_{i=1}^m |\beta_i| + \frac{1}{2}\lambda(1-\alpha)\sum_{u=1}^m \beta_u^2.
$$

Thus, the regularization reduces to that of the elastic net (enet) procedure. In general, when the linkage is nontrivial, the term $\lambda(1-\alpha)g(u)$ is added to the elastic net to get the desired grouping effect.

3.4 Adaptive Link-Constrained Regularization

When there is a link between two predictors but their regression coefficients have different signs, the coefficients cannot be expected to be smooth [Li(2010)]—even locally. To resolve this problem, we first need to estimate the sign of the coefficients and then refit the model with estimated signs. When the number of predictor variables is smaller than that of sample points, ordinary least squares are performed, while ridge estimates are computed, otherwise. We have to modify the Laplacian matrix in the penalty function:

$$L^*(u,v) = \begin{cases} 1, & \text{if } u = v \text{ and } d_u \neq 0, \\ -s_u s_v (d_u d_v)^{-\frac{1}{2}}, & \text{if } u \text{ and } v \text{ are adjacent}, \\ 0, & \text{otherwise} \end{cases}$$

and then update the $g(\cdot)$-function in Equation (3.10) via

$$g^*(u) = \sum_{u \sim v} \frac{s_u s_v \beta_v^*}{\sqrt{d_u d_v}}.$$

3.5 Accuracy, Sensitivity, Specificity and Area under the Receiver Operating Curve (AUROC)

We evaluated four metrics of binary classification for each of lasso, adaptive lasso, elastic net and the proposed rank based logistic regression methods to compare the performance. These metrics are accuracy, sensitivity, specificity and AUROC.

Based on the notations in Table 3.1, we define

$$\text{Accuracy} = \frac{a+b}{a+b+c+d}, \qquad \text{Specificity} = \frac{d}{n-m}, \qquad \text{Sensitivity} = \frac{a}{m}$$

as well as

$$\text{TPR (true positive rate)} = \frac{a}{k}, \qquad \text{FPR (false positive rate)} = \frac{b}{n-m}.$$

The last metric AUROC is related to the probability that the classifier under consideration will rank a randomly selected positive case higher than a randomly selected negative case [Fawcett(2006)]. The values of all these fours metrics—accuracy, sensitivity, specificity and AUROC—range from 0 to 1. The value of 1 represents a perfect model whereas the value of 0.5 corresponds to "coin tossing". The class prediction for each individual in binary classification is made based on a continuous random variable z. Given a threshold k as a tuning parameter, an individual is classified as "positive" if $z > k$ and "negative", otherwise. The random variable z follows a probability density $f_1(z)$ if the individual belongs to "positives" and $f_0(z)$, otherwise. So, the true positive and true negative rates are given by

$$\mathrm{TPR}(k) = \int_k^\infty f_1(z)\mathrm{d}z \quad \text{and} \quad \mathrm{FPR}(k) = \int_k^\infty f_0(z)\mathrm{d}z, \text{ respectively.}$$

Now, the AUROC statistic can be expressed as

$$A = \int_0^1 \mathrm{TPR}\big(\mathrm{FPR}^{-1}(z)\big)\mathrm{d}z = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{1}\{k' > k\} f_1(k') f_0(k)\mathrm{d}k'\mathrm{d}k = P(z_1 > z_0),$$

where z_1 and z_0 are the values of positive or negative instances, respectively.

Table 3.1. Confusion table: a is the number of true positives, b the number of false positives, c the number of false negatives and d the number of true negatives.

Predicted Condition	**True Condition**		
	Positive	Negative	Total
Positive	a	b	k
Nnegative	c	d	$n-k$
Total	m	$n-m$	$n = a+b+c+d$

3.6 Ranking and Best Model Selection

3.6.1 The Normalized Laplacian Matrix from Correalation Structure among Variables

The penalty function in Equation (3.9) has two tuning parameters, namely, $\alpha \in [0, 1]$ and $\lambda > 0$. The "limiting" cases $\alpha = 0$ and $\alpha = 1$ correspond to ridge and lasso regression, respectively. For a fixed value of α, the model selects more variables for smaller λ's and fewer variables for larger λ's. Theoretically, the result continuosly depends on α and should not significantly change under small perturbations of the latter [Sun(2012), Sun(2012)]. Empirically, however, we discovered that the results produced by `pclogit` significantly vary with α. In `pclogit`, the Laplacian matrix determines the group effects of predictors and is calculated from adjacency matrix via

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A},$$

where $\boldsymbol{D}$ is the degree matrix and $\boldsymbol{A}$ is the adjacency matrix. The degree-scaled difference of predictors in Equation (3.9) is computed from the normalized Laplacian matrix

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}.$$

We computed the adjacency matrix by using the information from the correlation matrix obtaining

$$A(u, v) = \begin{cases} 1, & \text{if } u \neq v \text{ and } |\text{cor}(u, v)| \geq \epsilon, \\ 0, & \text{if } u = v \text{ or } |\text{cor}(u, v)| < \epsilon. \end{cases}$$

Here, $\epsilon \in (0, 1)$ is a specific cut-off value for correlation. So, ϵ is another tuning parameter in our model which needs to be optimally selected.

3.6.2 Best Tuning parameter

In summary, to find an optimal combination of parameters α and ϵ, we make the combination of tuning parameter α and ϵ, where the total number of combinations is given

by

$$C = K \times L$$

with K and L being the number of ϵ and α values, respectively. We compared the performance for each of different combinations with T resamplings. The (negative) measure of performance for each combination is the misclassification or error rate. The pair (α, ϵ) producing the smallest misclassification rate is declared optimal and used in the next step.

3.6.3 Ranking and Best Model

The sparse coefficient matrix with dimensions $m \times$ nlam (nlam = number of λ's) is used in `pclogit` (cf. [Sun(2012), Sun(2012)]). By default, nlam = 100. We extracted all predictors with non-zero coefficients for each of λ values. Then we built 100 logistic regression models. Given estimated parameter values $\boldsymbol{\beta}$, we have the estimated class probability for a predictor vector $\boldsymbol{x}$ at each of λ values.

$$\pi(\boldsymbol{x}) = \frac{\exp(\boldsymbol{x}^T\boldsymbol{\beta})}{1 + \exp(\boldsymbol{x}^T\boldsymbol{\beta})}$$

Using the "naïve" Bayesian approach, we infer $y = 1$ if $\pi \geq 0.5$ and $y = 0$, otherwise. The values of accuracy, sensitivity, specificity and AUROC statistics are computed for each of 100 models and ranked in an increasing order by their values. Note that AUROC method does not use a fixed cut-off value, e.g., $c = 0.5$, but rather describes the overall performance with all possible cut-off values in the decision rule. Let R_{ij}, $i = 1, 2, 3, 4$, $j = 1, 2, \ldots, 100$, comprise the ranking matrix $\boldsymbol{R}$. The first row, i.e., $i = 1$, displays the ranking of models with respect to their accuracy. Similarly, $i = 2$ ranks the models with respect to their sensitivity, $i = 3$, in terms of specificity and $i = 4$ by AUROC. Suppose, $R_{1,5} > R_{1,8}$. Then in the 1st row (i.e., in terms of accuracy), model 5 outperforms model 8. We calculate the column means $(\bar{R}_{\cdot j})$ of the $\boldsymbol{R}$ matrix. The column with the highest overall mean value of accuracy, sensitivity, specificity and AUROC will be chosen as the resulting optimal model. Note that there is a one-to-one correspondence between columns and the 100 competing models. In (the unlikely) case of two or more columns producing

the same mean, the column with a smaller index j is selected since the model represented by such column is more parsimonious. Formally, suppose p and q, $p > q$, are two column indices in the $\boldsymbol{R}$ matrix. If $\bar{R}_{.p} = \bar{R}_{.q} = \max_r \bar{R}_{.r}$, the q-th column will be selected and the associated model becomes our proposed rank-based penalized logistic regression model.

Chapter 4

Analysis of Simulated Data

4.1 Simulation from Multivariate Normal Distribution

In this chapter, we conducted extensive simulation studies to compare the performance in terms of accuracy, sensitivity, specificity and AUROC as well as the power of detecting true important variables by the proposed method with the performance of such three prominent regularized logistic regression methods as lasso, adaptive lasso and elastic net. We decided to focus on these (meanwhile) classical methods due to their popularity both in the literature and applications. Some of their very recently developed comptetitors such as [Lee et al. (2016)] (R-package `SelectiveInference`) and [Cilluffo et al. (2019)] (R-package `islasso`) are currently gaining attention from the community and will be used as benchmarks in our future research.

Continuing with the description of our simulation study, all predictors $\boldsymbol{x}$ were generated from a multivariate normal distribution with the following probability density function

$$f(\boldsymbol{x}) = \Big(\frac{1}{2\pi}\Big)^{\frac{m}{2}} \frac{1}{\sqrt{\det(\boldsymbol{\Sigma})}} \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big)$$

with an m-dimensional mean vector $\boldsymbol{\mu}$ and an $(m \times m)$-dimensional covariance matrix $\boldsymbol{\Sigma}$. Writing out the covariance matrix

$$\boldsymbol{\Sigma} = (\sigma_{ij}) = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \cdots & \sigma_{1m} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \cdots & \sigma_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{m1} & \sigma_{m2} & \sigma_{m3} & \cdots & \sigma_{mm} \end{pmatrix} \quad \text{with} \quad \sigma_{ii} = \sigma_i^2,$$

the correlation matrix $\boldsymbol{M}$ can be expressed as

$$\boldsymbol{M} = (\rho_{ij}) = \begin{pmatrix} \rho_{11} & \rho_{12} & \rho_{13} & \cdots & \rho_{1m} \\ \rho_{21} & \rho_{22} & \rho_{23} & \cdots & \rho_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho_{m1} & \rho_{m2} & \rho_{m3} & \cdots & \rho_{mm} \end{pmatrix} \quad \text{with} \quad \rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}^2 \sigma_{jj}^2}}.$$

The binary response variable is generated using Bernoulli distribution with individual probability (π) defined as

$$\pi(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{x}^T \boldsymbol{\beta})},$$

$\boldsymbol{x}$ is the matrix of true important variables and $\boldsymbol{\beta}$ is the associated preassigned regression coefficients.

4.2 Three Different Scenarios of Simulation Study

In this section, we present the details of the three different simulation scenarios considered.

4.2.1 Scenario 1

Under scenario 1, each of the simulated datasets has 200 observations and 1000 predictors. Here, for all $\boldsymbol{x}$ vectors, we let $\boldsymbol{\mu} = \mathbf{0}$ and $\mathbf{Var}(x_j) = 0.3$. Pairwise correlation of $\rho = 0.4$ was applied to the first eight variables, while the remaining 992 variables were left uncorrelated. The $\boldsymbol{\beta}$-vector was chosen as

$$\boldsymbol{\beta} = \big(\underbrace{2, 2, 2, 2, 2}_{\text{5 entries}}, \underbrace{3, 3, 3}_{\text{3 entries}}, \underbrace{0, 0, 0, \ldots, 0}_{\text{992 entries}}\big).$$

Each of the datasets was split into training and test sets with equal proportions.

4.2.2 Scenario 2

The datasets under scenario 2 also have 200 observations and 1000 predictors. Again, $\boldsymbol{\mu} = \mathbf{0}$ and $\mathbf{Var}(x_j) = 0.3$. Now, the first five variables were assumed to have a correlation of $\rho = 0.4$. The remaining 995 variables were independent. The $\boldsymbol{\beta}$-vector was selected as

$$\boldsymbol{\beta} = \big(\underbrace{2.0, 2.0, 2.0, 2.7, 2.0, 2.0, 2.5, 2.7, -2.8, 3.0, 2.6, 3.0, 3.0, 3.0, 3.0}_{\text{15 entries}}, \underbrace{0, 0, 0, \ldots, 0}_{\text{985 entries}}\big).$$

Each of the datasets was split into training and test sets with equal proportions.

4.2.3 Scenario 3

Under the last scenario 3, each of the datasets has 150 observations and 1000 predictors.We let $\boldsymbol{\mu} = \mathbf{0}$ and $\mathbf{Var}(x_j) = 0.4$. The first five variables were assigned into a correlation value of $\rho = 0.3$, while the variables with indices from 11 to 30 were chosen to have the correlation value of $\rho = 0.6$. Outside of these two blocks, the variables were assumed uncorrelated. The $\boldsymbol{\beta}$-vector was chosen

$$\boldsymbol{\beta} = \big(\underbrace{2.0, 2.0, 2.0, 2.0, 2.0, 2.5, -2.6, 2.7, 3.0, -2.9, 2.0, 2.0, 2.0, 2.0, 2.0}_{\text{15 entries}},$$
$$\underbrace{2.5, -2.0, 2.7, 3.0, -2.5, 2.0, 2.0, 2.0, 2.0, 2.0, 2.5, -2.0, 2.7, 3.0, -2.5}_{\text{15 entries}}, \underbrace{0, 0, 0, \ldots, 0}_{\text{970 entries}}\big).$$

The dataset was split into training and test sets with ratio of 70 to 30.

4.3 Algorithm for Simulation Study

We compared the proposed rank-based penalized logistic regression method with lasso, adaptive lasso and elastic net methods from the glmnet `R`-package [Zou(2005)]. The above-protocoled Algorithm summarizes the procedure to calculate the average value of accuracy, sensitivity, specificity and AUROC based on a given number of iterations for each of the three simulation scenarios.

Algorithm 1 Calculation of overall mean and standard deviation on simulation studies

Step 1: Generate the data on each of the three simulation scenarios.

Step 2: Split the data into training and test sets randomly with the ratio of 70 to 30.

Step 3: Screen the variables using BH-FDR based on the training dataset.

Step 4: Plug the screened variables to each of the four methods.

Step 5: Calculate the values of Accuracy, Sensitivity, Specificity and AUROC for each of the methods.

Step 6: Repeats Step 1–5 to achieve a given number of replications.

Step 7: Calculate the means and standard deviations for each of the methods.

4.4 Comparison of the Performance of Four Different Methods

In Table 4.1, we compare the estimated mean and standard deviation of accuracy, sensitivity, specificity and AUROC values based on 200 iterations under correlation structure of $\rho = 0.4$ in the simulation of scenario 1. The proposed rank-based penalized method shows the highest accuracy of 0.963 with the standard deviation of 0.02, sensitivity of 0.961 with standard deviation of 0.03, specificity of 0.965 with standard deviation of 0.03. In addition, it yields the same AUROC of 0.995 with standard deviation of 0.01 as elastic net and adaptive lasso.

Table 4.1. Comparison of the performance among the four methods over 200 replications under simulation scenario 1. The values in parentheseses are the standard deviations.

Method	**Accuracy**	**Sensitivity**	**Specificity**	**AUROC**
rank-based	0.963 (0.02)	0.961 (0.03)	0.965 (0.03)	0.995 (0.01)
lasso	0.953 (0.03)	0.952 (0.04)	0.955 (0.03)	0.993 (0.01)
alasso	0.957 (0.03)	0.955 (0.04)	0.960 (0.03)	0.995 (0.01)
enet	0.961 (0.02)	0.959 (0.04)	0.962 (0.03)	0.995 (0.01)

In Table 4.2, we compare estimated mean and standard deviation of accuracy, sensitivity, specificity and AUROC values using 200 iterations under correlation structure of $\rho = 0.4$ in the simulation of scenario 2. The proposed rank-based method also shows highest accuracy of 0.831 with standard deviation 0.04, sensitivity of 0.833 with standard deviation of 0.06, specificity of 0.829 with standard deviation of 0.05. In addition, the proposed method produces AUROC of 0.913 with standard deviation of 0.03. This is the second highest value which is slightly lower than the AUROC value of the elastic net.

Table 4.2. Comparison of the performance among the four methods over 200 replications under simulation scenario 2. The values in parentheseses are the standard deviations.

Method	**Accuracy**	**Sensitivity**	**Specificity**	**AUROC**
rank-based	0.831 (0.04)	0.833 (0.03)	0.829 (0.05)	0.913 (0.03)
lasso	0.826 (0.05)	0.827 (0.07)	0.825 (0.09)	0.910 (0.04)
alasso	0.815 (0.04)	0.814 (0.07)	0.815 (0.07)	0.902 (0.04)
enet	0.826 (0.04)	0.828 (0.07)	0.825 (0.07)	0.915 (0.03)

In Table 4.3, we compare estimated mean and standard deviation of accuracy, sensitivity, specificity and AUROC values with 150 iterations under correlation structure of $\rho = 0.3$ and $\rho = 0.6$ in simulation of scenario 3. The proposed method shows highest accuracy of 0.916 with standard deviation of 0.04, sensitivity of 0.919 with standard deviation of 0.06, specificity of 0.912 with standard deviation of 0.06 and AUROC of 0.977 with standard deviation of 0.02.

Table 4.3. Comparison of the performance among the four methods over 150 replications under simulation scenario 3. The values in parentheseses are the standard deviations.

Method	**Accuracy**	**Sensitivity**	**Specificity**	**AUROC**
rank-based	0.916 (0.04)	0.919 (0.06)	0.912 (0.06)	0.977 (0.02)
lasso	0.888 (0.04)	0.898 (0.06)	0.880 (0.07)	0.963 (0.02)
alasso	0.866 (0.04)	0.877 (0.07)	0.855 (0.07)	0.949 (0.03)
enet	0.909 (0.04)	0.916 (0.06)	0.903 (0.06)	0.975 (0.02)

Furthermore, we compared the performance in terms of selecting the number of true important variables by each of the four methods under three different simulation scenarios. First, we performed multiple hypothesis testing with BH-FDR [Benjamin(1995)] to reduce the dimensionality of the data. After performing a screening step to retain the relevant

variables, we used them as input for the proposed rank-based penalized method with the regularization step outlined in Section 3.3. We illustrate the performance of variable selection with boxplots in Figures 4.1–4.3 for simulation scenarios 1, 2, and 3. Each figure displays two boxplots, which, in turn, depict the distribution of the number of variables selected (NVS) and the number of true important variables (NTIV) within the number of variables selected (NVS) with each of the four methods computed based on the given number of iterations in each of the three simulation scenarios.

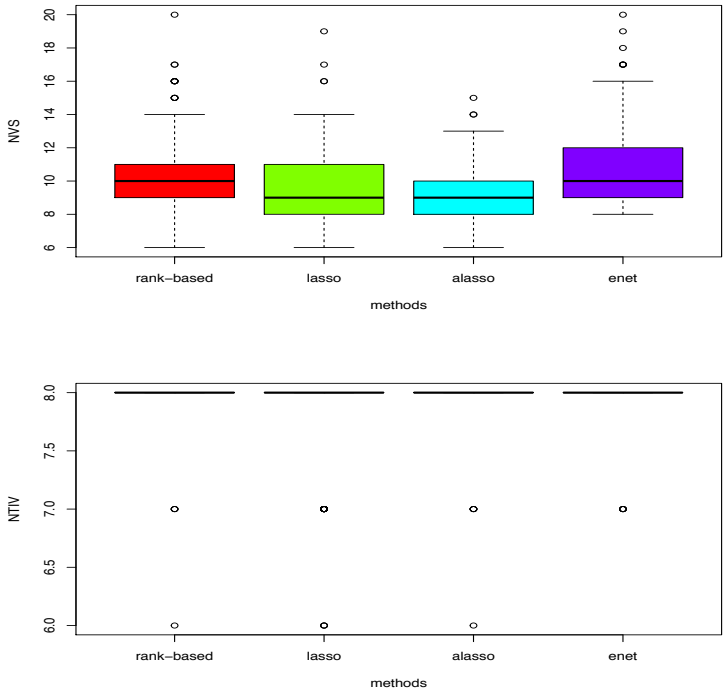

Figure 4.1. Boxplots of total number of variables (NVS) selected and the number of true important variables (NTIV) within the number of variables selected with four different models under scenario 1 based on 200 replications.

Figure 4.1 reports that the proposed rank-based method has a slightly higher median number of variables selected (displayed as a thick line in the upper boxplots) than lasso, adaptive lasso and elastic net under scenario 1. The lower boxplots show that all four

methods performed head-to-head for selection of true important variables under scenario 1 with 200 replications. Table 4.4 compares the mean and the standard deviation (in parentheseses) of the number of variables (NVS) selected and the number of true important variables (NTIV) in NVS for each of the four methods over 200 replications. The proposed rank-based method and elastic net performed head-to-head while slightly outperforming lasso and adaptive lasso.

Table 4.4. Estimated mean and standard deviation of number of variables (NVS) selected and the number of true important variables (NTIV) among NVS with four different models under simulated scenario 1 with 200 replications. The values in parentheseses are standard deviations.

Method	**NVS**	**NTIV**
rank-based	10.465 (2.24)	7.975 (0.19)
lasso	9.885 (1.98)	7.880 (0.37)
alasso	9.475 (1.47)	7.970 (0.20)
enet	10.805 (2.37)	7.975 (0.16)

Figure 4.2 suggests the proposed method has a marginally higher median number of variable selected compared to the other three methods in the upper boxplot. It is also clear that the proposed method has a slightly higher median number of true important variables in the lower boxplot on scenarios 2 computed with 200 replications. Table 4.5 confirms that the rank-based penalized method has the highest mean both for selecting the number of variables and important variables.

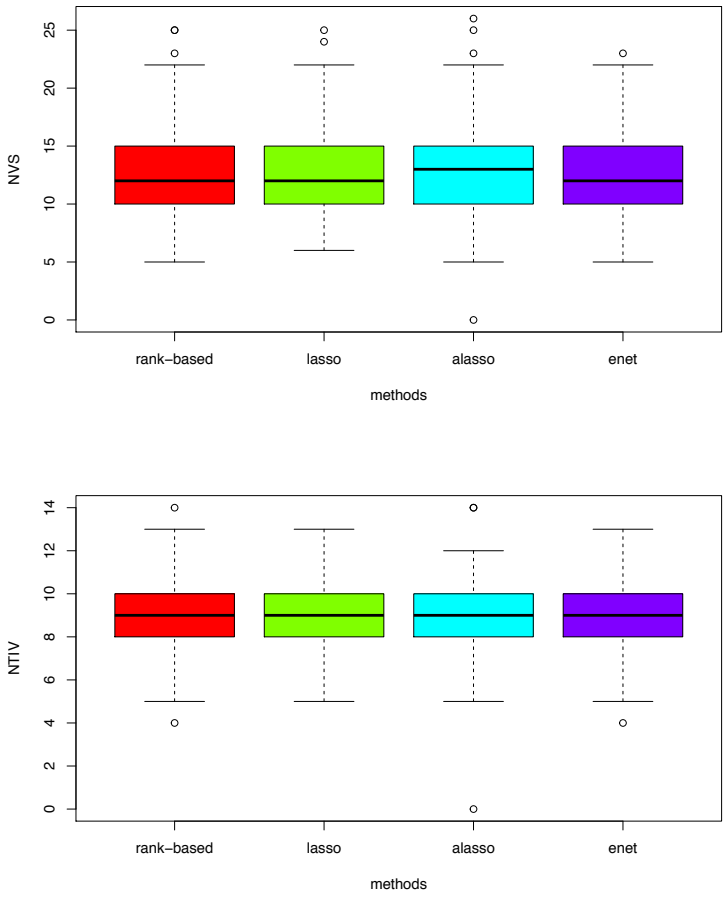

Figure 4.2. Boxplots of total number of variables (NVS) selected and the number of true important variables (NTIV) within the number of variables selected with four different models on scenario 2 based on 200 replications.

Table 4.5. Estimated mean and standard deviation of number of variables (NVS) selected and the number of true important variables (NTIV) among NVS in four different models under simulation scenario 2 with 200 replications. The values in parentheseses are standard deviations.

Method	**NVS**	**NTIV**
rank-based	13.675 (3.95)	9.345 (1.62)
lasso	12.750 (3.50)	8.905 (1.81)
alasso	11.965 (3.18)	8.720 (1.73)
enet	13.115 (3.92)	9.105 (1.73)

In Figure 4.3, the upper boxplot demonstrates that the proposed rank-based method has the highest median number of variables selected, elastic net has second highest median, lasso has third largest median and adaptive lasso has the smallest median under scenario 3 based on 150 replications. The lower boxplots also show that the proposed rank-based method has the highest median number of true important variables selected. However, adaptive lasso has a higher median number of true important variables than lasso unlike the upper boxplots. Thus, the proposed rank based-method clearly outperforms other three methods under high-correlation settings among variables.

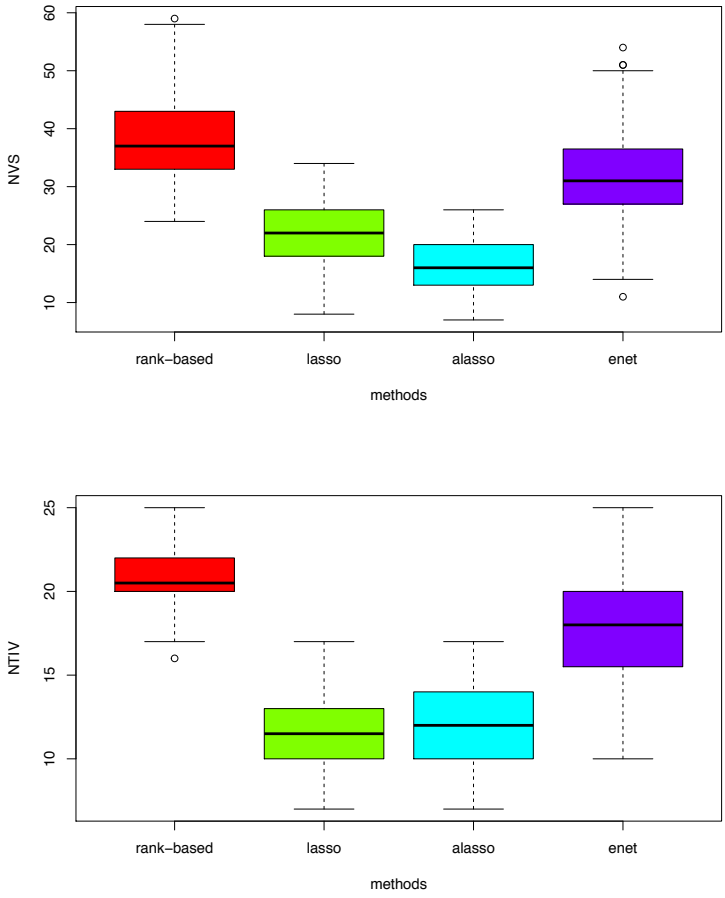

Figure 4.3. Boxplots of total number of variables (NVS) selected and the number of true important variables (NTIV) within the number of variables selected with four different models under scenario 3 based on 150 replications.

Table 4.6 summarizes the number of variables selected and true important variables selected across the four methods under the high-correlation setting among variables computed from 150 replications. The proposed rank-based method has the highest mean number of overall variables selected and true important variables selected.

Table 4.6. Estimated mean and standard deviation of number of variables (NVS) selected and the number of true important variables (NTIV) among NVS in four different models under simulation scenario 3 with 150 replications. The values in parentheseses are standard deviations.

Method	**NVS**	**NTIV**
rank-based	37.830 (7.14)	20.770 (1.88)
lasso	22.010 (4.96)	11.780 (2.21)
alasso	16.430 (4.39)	11.920 (2.28)
enet	32.270 (8.55)	17.600 (3.15)

Chapter 5

Real Data Analysis

5.1 Real Colon Cancer Gene Expression Data

We applied four logistic regression methods to select differentially expressed genes and assess their discrimination capability between colon cancer cases and healthy controls using high-dimensional gene expression data [ALON(1999)]. The colon cancer gene expression dataset is available at http://genomics-pubs.princeton.edu/oncology/affydata/index.html. It contains 2000 genes with the highest minimal intensity across 62 tissues. The data were measured on 40 colon tumor samples and 22 normal colon tissue samples. We split the data set into training and testing sets with proportions 70% and 30%, respectively. To detect significantly differentially expressed genes for high-dimensional colon cancer carcinoma and measure classification prediction, we adapted two step procedures of filtering and variable selection. First, we applied BH-FDR [Benjamin(1995)] to select most promising candidates of genes as a preprocessing step and then used the screened genes as input to the proposed rank-based method and three other popular methods – lasso, adaptive lasso and elastic net. The performance in terms of accuracy, sensitivity, specificity and AUROC as well as the selection probabilities for the four methods are reported in Tables 5.1 and 5.2, respectively.

5.2 Algorithm for Real Data Analysis

The Algorithm outlines above protocols the procedure of calculating the average values of accuracy, sensitivity, specificity and AUROC.

Algorithm 2 Calculation of mean with standard deviation on colon cancer data

Step 1: Split the data into training and test sets randomly with the ratio of 70 to 30.

Step 2: Screen genes with the BH-FDR method based on the training data.

Step 3: Plug the screened genes as the input to each of four methods.

Step 4: Calculate the values of Accuracy, Sensitivity, Specificity and AUROC across each of the methods on the test data.

Step 5: Repeat Steps 1 through 4 for 100 times.

Step 6: Calculate means and standard deviations for each of the methods.

5.3 Comparison of Performance of Four Different methods on Real Data Application

In Table 5.1, the performance of all four metrics are computed based on 100 iterations of resampled subsets of individuals. Here, the performance in terms of accuracy, sensitivity, specificity and AUROC as well as the selection probabilities for the four methods are reported.

Table 5.1. Estimated mean values and standard deviations for the four metrics across the four competing penalized logistic regression models computed from 100 resamplings. The values in parentheseses are standard deviations.

Colon Cancer Data Analysis Based on 100 Times Resmpling				
Method	**Accuracy**	**Sensitivity**	**Specificity**	**AUROC**
rank-based	0.853 (0.08)	0.860 (0.13)	0.840 (0.13)	0.917 (0.06)
lasso	0.801 (0.09)	0.911 (0.07)	0.637 (0.21)	0.897 (0.08)
adaptive lasso	0.804 (0.09)	0.869 (0.09)	0.719 (0.21)	0.877 (0.08)
elastic net	0.802 (0.09)	0.917 (0.07)	0.640 (0.22)	0.903 (0.07)

The average AUROC of 0.853 with standard deviation of 0.06 in the proposed rank-based method has the highest value compared to other three methods. Also the accuracy of 0.853 with standard deviation of 0.08 are optimal among the four methods. The values of sensitivity (0.860) and specificity (0.840) are also better than those of the other three methods. In summary, it is fair to conclude that the proposed rank-based method outperforms the other three popular penalized logistic regression methods.

5.4 Selection Probabilities of Important Genes

Table 5.2 shows top 5 ranked genes with highest selection probabilities for the proposed rank-based method, lasso, adaptive lasso and elastic net. An expressed sequence tag (EST) of Hsa.1660 associated with colon cancer carcinoma is found by all four methods. Hsa.36689 [Shevade(2003), Li(2002)] is shown and top ranked by the proposed method, lasso and elastic net. Hsa692 also appeared and is second ranked by the proposed method, lasso and elastic net. In addition, Hsa.37937 is shown and is third and second ranked by the proposed method and elastic net, respectively.

Table 5.2. List of top 5 ranked genes across rank-based, lasso, adaptive and elastic net. An extra asterix (*) sign is put next to a gene each time the gene is selected by one of four methods.

EST Name	Gene ID	Gene Description	Selection Probability
		Rank-Based	
***Hsa.36689	Z50753	H.sapiens mRNA for GCAP-II/uroguanylin precursor	1.00
***Hsa.692.2	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6	0.99
**Hsa.37937	R87126	Myosin heavy chain,nonmuscle(Gallus gallus)	0.97
****Hsa.1660	H55916	Peptidyl-prolyl cis-trans isomerase, mitrochondrial precursor(human)	0.91
Hsa.1832	R44887	nedd5 protein (Mus musculus)	0.90
		Lasso	
***Hsa.36689	Z50753	H.sapiens mRNA for GCAP-II/uroguanylin precursor	0.87
Hsa.692.2	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6	0.82
*****Hsa.1660	H55916	Peptidyl-prolyl cis-trans isomerase, mitrochondrial precursor(human)	0.66
Hsa.6814	H08393	Collagen alpha 2(XI) chain(Homo sapiens)	0.52
Hsa.8147	M63391	Human desmin gene, complete cds	0.50
		Adaptive Lasso	
Hsa.1454	M82919	H. gamma amino butyric acid(GABAA)receptor beta3 subunit mRNA,cds	0.83
Hsa.6814	H08393	Collagen alpha 2(XI) chain(Homo sapiens)	0.77
****Hsa.1660	H55916	Peptidyl-prolyl cis-trans isomerase, mitrochondrial precursor(human)	0.77
Hsa.14069	T67077	Sodium/Potasssium-transporting atpase gamma chain(Ovis aries)	0.69
Hsa.2456	U25138	Human MaxiK potassium channel beta subunit mRNA, complete cds	0.55
		Elastic Net	
***Hsa.36689	Z50753	H.sapiens mRNA for GCAP-II/uroguanylin precursor	0.98
**Hsa.37937	R87126	Myosin heavy chain,nonmuscle(Gallus gallus)	0.94
***Hsa.692.2	M76378	Human cysteine-rich protein (CRP) gene, exons 5 and 6	0.94
Hsa.8147	M63391	Human desmin gene, complete cds	0.91
****Hsa.1660	H55916	Peptidyl-prolyl cis-trans isomerase, mitrochondrial precursor(human)	0.84

Chapter 6

Discussion and Conclusions

6.1 Discussion

In this thesis, we proposed a new rank-based penalized logistic regression method to improve classification performance and the power of variable selection in high-dimensional data with strong correlation structure.

Our simulation studies demonstrated that the proposed method improves not only the performance of classification or class prediction but also the detection of true important variables under various correlation settings among features when compared to existing popular regularization methods such as lasso, adaptive lasso, and elastic net. As demonstrated by simulation studies, if the true important variables are not passed through the filtering method such as BH-FDR, their chance of being selected in the final model decreases signficantly, thus, leading to reduction in variable selection and classification performance. Therefore, effective filtering methods which are likely to retain as many most promising variables as possible are indispensable.

Applied to high-dimensional colon gene expression data, the proposed rank-based logistic regresson method with BH-FDR screening produced the highest average AUROC value of 0.917 with standard deviation of 0.06 and accuracy of 0.853 with standard deviation of 0.08 using 100 resampling steps. The proposed method produced a good balance between sensitivity and specificity in contrast to other methods. Elastic net demonstrated the second best peformance with an average AUROC value of 0.903 with standard deviation of 0.07. A probable reason is that elastic net accounts for group correlation effects. In addition, we compared top 5 ranked ESTs across the proposed method, lasso, adaptive lasso

and elastic net [Sun(2012)]. They had a common EST of Hsa.1660 associated to colon cancer data. We also found that Hsa.36689 was both deemed important and top ranked by the proposed method, lasso and elastic net. This also applied to Hsa.692, which was deemed important and second top ranked by the proposed method and lasso, whereas it was only third-ranked by the elastic net. Hsa.37937 was detected by both the proposed method and the elastic net. Hence, the four ESTs mentioned appear to be promising candidate biomarkers associated with colon cancer carcinoma. The function of the genes corresponding to ESTs is summarized in Table 5.2.

6.2 Conclusions

In this study the proposed rank-based classifier demonstrated the superiority in not only classification prediction but also the power of detecting true important variables when compared to lasso, adaptive lasso, and elastic net through the extensive simulation studies. Besides, in the application of high-dimensional colon cancer gene expression data, the proposed classifier showed the best performance in terms of accuracy and AUROC among the four classifiers considered in the paper. Our proposed method also outperforms the lasso, adaptive lasso, and elastic net in case of selection probabilities of important genes in real colon cancer gene expression data.

6.3 Future Work

As a future research, we would develop the methodology of variable selection and compare the performance with those of most recent competitors such as [Lee et al. (2016), Cilluffo et al. (2019), Frost et al. (2017), Boulesteix et al. (2017)], etc.

References

[Houwelingen1(2006)] Houwelingen, H.C.V.; Bruinsma, T.; Hart, A.A.M.; Veer, L.J.V.; Wessels, L.F.A. Cross-validated Cox regression on microarray gene expression data. *Stat. Med.* **2006**, *25*, 3201–3216.

[Lofti(2014)] Lofti, E.; Keshavarz, A. Gene expression microarray classification using PCA–BEL. *Comput. Biol. Med.* **2014**, *54*, 180–187.

[Algamal(2015)] Algamal, Z.Y.; Lee, M.H. Penalized logistic regression with the adaptive LASSO for gene selection in high-dimensional cancer classification. *Expert Syst. Appl.* **2015**, *42*, 9326–9332.

[Li(2008)] Li, C.; Li, H. Network-constrained regularization and variable selection for analysis of genomic data. *Bioinformatics* **2008**, *24*, 1175–1182.

[Sayes(2007)] Sayes, Y.; Inzka, I.; Larranaga, P. A review of feature selection techniques in bioinformatics. *Bioinformatics* **2007**, *23*, 2507–2517.

[Tibshirani(1996)] Tibshirani, R. Regression shrinkage and selection via the Lasso. *J. R. Stat. Soc. B* **1996**, *58*, 267–288.

[Li(2001)] Fan, J.; Li, R. Variable selection via nonconcave penalized likelihood and its oracle properties. *J. Am. Stat. Assoc.* **2001**, *96*, 1175–1182.

[Tibshirani(2005)] Tibshirani, R.; Saunders, M. Sparsity and smoothness via the fused lasso. *J. R. Stat. Soc. B* **2005**, *67*, 91–108.

[Zou(2006)] Zou, H. The adaptive Lasso and its oracle properties. *J. Am. Stat. Assoc.* **2006**, *101*, 1418–1429.

[Meinshausen(2009)] Meinshausen, N.; Yu, B. Lasso-type recovery of sparse representations for high-dimensional data. *Ann. Stat.* **2009**, *37*, 246–270.

[Huang(2016)] Huang, H.; Liu, X.Y.; Liang, Y. Feature selection and cancer classification via sparse logistic regression with the hybrid $L_{1/2+2}$ regularization. *PLoS ONE* **2009**, *11*, e0149675.

[Zou(2005)] Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. R. Stat. Soc. B* **2005**, *67*, 301–320.

[Sun(2012)] Sun, H.; Wang, S. Penalized logistic regression for high-dimensional DNA methylation data with case-control studies. *Bioinformatics* **2012**, *28*, 1368–1375.

[Sun(2012)] Sun, H.; Wang, S. Network-based regularization for matched case-control analysis of high-dimensional DNA methylation data. *Stat. Med.* **2012**, *32*, 2127–2139.

[Reiner(2003)] Reiner, H.; Yekutieli, D.; Benjamin, Y. Identifying differentially expressed genes using false discovery rate controlling procedures. *Bioinformatics* **2003**, *19*, 368–375.

[Benjamin(1995)] Benjamini, Y.; Hochberg, Y. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. R. Stat. Soc. B* **1995**, *57*, 289–300.

[Li(2010)] Li, C.; Li, H. Variable selection and regression analysis for graph-structured covariates with an application to genomics. *Ann. Appl. Stat.* **2010**, *4*, 1498–1516.

[Friedman(2007)] Friedman, J.; Hastie, T.; Hofling, H.; Tibshirani, R. Pathwise coordinate optimization. *Ann. Appl. Stat.* **2007**, *1*, 302–332.

[Friedman(2010)] Friedman, J.; Hastie, T.; Tibshirani, R. Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.* **2010**, *33*, 1–22.

[Fawcett(2006)] Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874.

[Lee et al. (2016)] Lee, J.D.; Sun, D.L.; Sun, Y.; Taylor, J.E. Exact post-selection inference, with application to the lasso. *Ann. Stat.* **2016**, *44*, 907–927.

[Cilluffo et al. (2019)] Cilluffo, G.; Sottile, G.; La Grutta, S.; Muggeo, V.M.R. The Induced Smoothed lasso: A practical framework for hypothesis testing in high dimensional regression. *Stat. Methods Med. Res.* **2019**, doi:10.1177/0962280219842890.

[ALON(1999)] Alon, U.; Barakai, N.; Notterman, D.A.; Gish, K.; Ybarra, S.; Mack, D.; Levine, A.J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA* **1999**, *96*, 6745–6750.

[Shevade(2003)] Ding, Y.; Wilkins, D. A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics* **2003**, *19*, 2246–2253.

[Li(2002)] Li, Y.; Campbell, C.; Tipping, M. Bayesian automatic relevance determination algorithms for classfifying gene expression data. *Bioinformatics* **2002**, *18*, 1332–1339.

[Frost et al. (2017)] Frost, H.R.; Amos, C.I. Gene set selection via LASSO penalized regression (SLPR). *Nucleic Acids Res.* **2017**, doi:10.1093/nar/gkx291.

[Boulesteix et al. (2017)] Boulesteix, A.L.; De, B.R.; Jiang, X.; Fuchs, M. IPF-LASSO: Integrative L_1-Penalized Regression with Penalty Factors for Prediction Based on Multi-Omics Data. *Comput. Math. Methods Med.* **2017**, doi:10.1155/2017/7691937.

Appendix

R Codes

```
---
title: "200_simulation1_Result"
output: word_document
---

#############################################################
#Simulating Data from the multivariate normal Distribution#
#scenario 1                                                 #
#############################################################
```{r,warning=FALSE}
set.seed(9512)
library(rockchalk)
library(MASS)
library(pclogit)
library(glmnet)
library(MESS)
library(verification)
mat.lmd.md=matrix(0,nrow=200,ncol=11)
mat.lmd.md.lasso=matrix(0,nrow=200,ncol=11)
mat.lmd.md.alasso=matrix(0,nrow=200,ncol=11)
mat.lmd.md.ellasso=matrix(0,nrow=200,ncol=11)
best.models<-vector("list", 200)
best.models.lasso<-vector("list", 200)
best.models.alasso<-vector("list", 200)
best.models.ellasso<-vector("list", 200)
Beg=Sys.time()
#200 times itteration#
for (l in 1:200)
{
mycor= matrix(data=0, nrow=1000, ncol=1000)
#####################################
#First eight variables has           #
#correlation among them, which is 0.4#
#####################################
for(j in 1:8){
for(i in 1:8){
mycor[i,j] = (.4)
}
```
```

```
}
diag(mycor)<-1
myCov <- lazyCov(Rho = mycor, Sd = rep(0.3,1000))
#variance is 0.3#
set.seed(l)
datm1<-mvrnorm(n=200, mu=rep(0,1000), Sigma =myCov)
colnames(datm1)<-paste('var',1:1000)
beta<-c(rep(2,5),rep(3,3),rep(0,992))

########################################
#first five variables has coefficeient 2 #
#then 3 are 3  and others are zero       #
########################################
beta<-as.vector(beta)

z<-(datm1%*%beta)
prob <- 1/(1 + exp(-z))
y<-ifelse(prob>=.5,1,0)
y<-as.vector(y)
datm<-cbind(datm1,y)
#we generated response variable by loistic regression model#
datm.1<-data.frame(datm)
#imporatant variables need to compare later#
imp.vrb<-colnames(datm.1[,1:8])
########################################################
#filtering the data by variable screening              #
#(Benjamini & Hochberg (1995) ("BH" or its alias "fdr")#
#by adjustedp-values)                                  #
########################################################
datm.1.s<-datm.1[,-c(length(colnames(datm.1)))]
y<-datm.1[,c(length(colnames(datm.1)))]
p = apply(datm.1.s,2,function(x)t.test(x[y==0],x[y==1])$p.value)
padj = p.adjust(p,method="BH")
sel.col<-as.vector(names(padj[(padj<0.25)]))
datm.f<-datm.1.s[,c(sel.col)]
datm.f.1<-cbind(datm.f,y)
#partitioning the data#

data.col<-datm.f.1
n1<-NROW(data.col)
set.seed(120)
id.test<-sample(1:2,size=n1,replace=TRUE,prob=c(1/2,1/2))
dat.training<-data.col[id.test==1,]
dat.test<-data.col[id.test==2,]
train<-dat.training[,-c(length(colnames(data.col)))]
```

```
y.train<-dat.training$y
test<-dat.test[,-c(length(colnames(data.col)))]
y.test<-dat.test$y

set.seed(1223)
V<-2
index.cv<-sample(1:V,size=NROW(train),replace=TRUE)
mcer=al.cv=cr.cv=cv.v=NULL
for(v in 1:V)
{
train.v<-dat.training[index.cv!=v,]
valid.v<-dat.training[index.cv==v,]
train.cv<-train.v[,-c(length(colnames(data.col)))]
valid.cv<-valid.v[,-c(length(colnames(data.col)))]
y.train<-train.v$y
yobs<-valid.v$y
adjm.train.cv<-cor((train.cv))
diag(adjm.train.cv)<-0
########################################
# Randomly assigned correlation        #
#values to compute the adjacency matrix #
########################################
cr<-c(0.1,0.2,0.3)
for(i in 1:length(cr))
{
adjm.train.cv[abs(adjm.train.cv)<(cr[i])]<-0
adjm.train.cv[abs(adjm.train.cv)>=(cr[i])]<-1
sg.train.cv<-sign(rnorm(ncol(train.cv)))
al<-c(0.4,0.5,0.6)
for(k in 1:length(al))
{
g.ring.pc.cv<-pclogit(train.cv,y.train,
alpha=(al[k]),nlam=100,group=adjm.train.cv,
sgnc=sg.train.c)
bta.cv<-g.ring.pc.cv$beta
bta.cv<-as.matrix(bta.cv)
N<-ncol(bta.cv)
accuracy=sensitivity=specificity=auc=NULL
for(j in 1:N)
{
i.nm.cv<-as.vector(which(bta.cv[,j]!=0))
bt.a.cv<-bta.cv[,j]
b.bta.cv<-bt.a.cv[i.nm.cv]
B.cv<-matrix(b.bta.cv,ncol=1)
d.t.cv<-(train.cv[,which(bta.cv[,j]!=0)])
```

```
dt.cv<-as.matrix(d.t.cv)
e.cv<-exp(dt.cv%*%B.cv)
p.cv<-(e.cv/(1+e.cv))
yhat.cv<-ifelse(p.cv>=0.5,1,0)
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
ac<-accuracy.own(y.train,yhat.cv)
accuracy=c(accuracy,ac)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1) else
return(0)}
sen<-sensitivity.own(y.train,yhat.cv)
sensitivity=c(sensitivity,sen)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
spe<-specificity.own(y.train,yhat.cv)
specificity=c(specificity,spe)
AUC<-roc.area(obs=y.train, pred=p.cv)$A
auc<-c(auc,AUC)
}
Rank.M<-matrix(c(as.numeric(factor(accuracy)),
as.numeric(factor(sensitivity)),
as.numeric(factor(specificity)),
as.numeric(factor(auc))), nrow=4,
ncol=N,byrow=TRUE)
Mean.col=NULL
for (L in 1:NCOL(Rank.M) )
{
A.column<-function(x)
{
a<-mean(Rank.M[,x])
}
mn<-A.column(L)
Mean.col=c(Mean.col,mn)
}
Mean.col<-as.vector(as.numeric(Mean.col))
O.Rank<-as.matrix(Mean.col)
m<-max(as.numeric(O.Rank[,1]))
```

```
optimal<-which(O.Rank[,1]==m)[1]
B.t<-as.matrix(bta.cv[which(bta.cv[,optimal]!=0),][,optimal])
dt.t<-as.matrix(valid.cv[,which(bta.cv[,optimal]!=0)])
e.t<-exp(dt.t%*%B.t)
p.t<-(e.t/(1+e.t))
yhat.test<-ifelse(p.t>=0.5,1,0)
cm<-table(actual=yobs, fitted=yhat.test)
er.cm<-1-sum(diag(cm))/sum(cm)
mcer<-c(mcer,er.cm)
al.cv<-c(al.cv,(al[k]))
cr.cv<-c(cr.cv,(cr[i]))
cv.v<-c(cv.v,v)
}
}
}

crs.mat<-matrix(c(mcer,al.cv,cr.cv,cv.v),ncol=4)
cl.min<-which.min(crs.mat[,1])
optim.alp<-crs.mat[cl.min,2]
optim.corr<-as.numeric(crs.mat[cl.min,3])
#####################################################
#Here to select best alpha value and best         #
#correlation for adjacency matrix we use          #
#resampling from the train data set. We           #
#compute missclafication error rate for           #
#selecting the best alpha and best correlation.   #
#The combination of alpha nad correlation give    #
#minimum missclassification error rate , those    #
#are our best values.                             #
#####################################################
adjm.train<-cor((train))
diag(adjm.train)<-0
adjm.train[abs(adjm.train)<optim.corr]<-0
adjm.train[abs(adjm.train)>=optim.corr]<-1

#######################################################
# We used optimum correlation value from            #
#validation data for computing the adjacency matrix.#
#Which will used to compute the laplacian matrix in #
#"pclogit" function                                 #
#######################################################
sg.train<-sign(rnorm(ncol(train)))
#We also need to consider the sign#

y.train<-dat.training$y
```

```
set.seed(257)
#################################################################
#We applied the training data for trained our "pclogit" model#
#################################################################
g.ring.pc<-pclogit(train,y.train,alpha=optim.alp,
nlam=100,group=adjm.train,
sgnc=sg.train)
bta<-g.ring.pc$beta
bta<-as.matrix(bta)
#########################################################
#Here we extract the sparse matrix of beta             #
#values with respect to all different                  #
#lamda values. The dimension will be (number of        #
#variables by number of lambda values). later          #
#on we build up number of different logistic           #
#regression model which is equal number of different   #
#lamda values. We only consider the variables for      #
#each model which has non  zero coefficient.           #
#########################################################
ac.lambda<-g.ring.pc$lambda
N<-ncol(bta)
accuracy=sensitivity=specificity=auc=NULL
for(j in 1:N)
{
i.nm<-as.vector(which(bta[,j]!=0))
bt.a<-bta[,j]
b.bta<-bt.a[i.nm]
B<-matrix(b.bta,ncol=1)
d.t<-(train[,which(bta[,j]!=0)])
dt<-as.matrix(d.t)
e<-exp(dt%*%B)
p<-(e/(1+e))
yhat<-ifelse(p>=0.5,1,0)
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
ac<-accuracy.own(y.train,yhat)
accuracy=c(accuracy,ac)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)}
sen<-sensitivity.own(y.train,yhat)
sensitivity=c(sensitivity,sen)
```

```
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
spe<-specificity.own(y.train,yhat)
specificity=c(specificity,spe)
AUC<-roc.area(obs=y.train, pred=p)$A
auc<-c(auc,AUC)
}
Rank.M<-matrix(c(as.numeric(factor(accuracy)),
as.numeric(factor(sensitivity)),as.numeric(factor(specificity)),
as.numeric(factor(auc))), nrow=4, ncol=N,byrow=TRUE)
#################################################################
#We computed the accuracy, sensitivity , specificity and     #
#AUC values of all different model And created a Matrix,     #
#we values. Later on we take the average of each colum and   #
#found which has the maximum average value is our best model.#
#In case of ties we considered the more parsimonious model.  #
#################################################################

#mean of all column#
Mean.col=NULL
for (i in 1:NCOL(Rank.M) )
{
A.column<-function(x)
{
a<-mean(Rank.M[,x])
}
mn<-A.column(i)
Mean.col=c(Mean.col,mn)
}
Mean.col<-as.vector(as.numeric(Mean.col))
O.Rank<-as.matrix(Mean.col)
m<-max(as.numeric(O.Rank[,1]))
optimal<-which(O.Rank[,1]==m)[1]
B.t<-as.matrix(bta[which(bta[,optimal]!=0),][,optimal])
dt.t<-as.matrix(test[,which(bta[,optimal]!=0)])
e.t<-exp(dt.t%*%B.t)
p.t<-(e.t/(1+e.t))
yhat.test<-ifelse(p.t>=0.5,1,0)
#########################################
#accuracy, sensitivity, specificity and AUC#
#value for best model on test data         #
```

```
##########################################
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
accuracy.test<-accuracy.own(y.test,yhat.test)

sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1) else
return(0)}
sensitivity.test<-sensitivity.own(y.test,yhat.test)

specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
specificity.test<-specificity.own(y.test,yhat.test)

AUC.test<-roc.area(obs=y.test, pred=p.t)$A
#(TP,TN,FP,FN)
mt<-table(pred=yhat.test,truth=y.test)
mt.v<-as.vector(mt)
tr.postv<-mt.v[4]
tr.ngtv<-mt.v[1]
flse.postv<-mt.v[2]
flse.ngtv<-mt.v[3]
#############################################################
#I applied our best model on test data set then computed    #
#the accuracy, sensitivity, specificity,AUC values          #
#True positive,  True negative, False positiveand           #
#False negative value                                       #
#############################################################
opt.modl<-optimal
lmda.opt<-ac.lambda[optimal]
op.nm<-as.vector(which(bta[,optimal]!=0))
op.bt.a<-bta[,optimal]
op.b.bta<-op.bt.a[op.nm]
lenth.opt.modl<-length(op.b.bta)
tp.var<-intersect(imp.vrb,names(op.b.bta))
nm.tp.var<-length(tp.var)
precsn.vrb<-(nm.tp.var/lenth.opt.modl)
#########################################################
```

```
# We can get the number of variables in each        #
#model and number of truley important variables     #
#selcted by  each model. Then we can compute the    #
#precision of selected truely important varibales   #
#for each model.matrix for number of  best models   #
#and lamda values and number of coeef each model    #
#####################################################

opt.lmd.mdl<-c(lenth.opt.modl,nm.tp.var,precsn.vrb,
tr.postv,tr.ngtv,flse.postv,flse.ngtv,accuracy.test,
sensitivity.test,specificity.test,AUC.test)
mat.lmd.md[l,1:11]=opt.lmd.mdl

#list for best models#
best.models[[l]]<-names(op.b.bta)

#LASSO#
X<-model.matrix(y~.-1,data=dat.training)
#(without intercept)#
y<-dat.training$y
X.test<-model.matrix(y~.-1,data=dat.test)
fit.lasso<-glmnet(x=X,y=y,family="binomial",alpha = 1)

#cross-validation for glmnet#
lambda<-fit.lasso$lambda
cv.lasso <- cv.glmnet(x=X, y=y, alpha =1,
lambda = lambda, nfolds=10)
lmbd0<-cv.lasso$lambda.min

#fit final model#
fit.lasso.final<- glmnet(x=X, y=y,
family="binomial", alpha = 1,lambda=lmbd0,
standardize = T, thresh = 1e-07, maxit=1000)

#prediction
pred.final<- predict(fit.lasso.final,
newx=X.test, s=lmbd0, type="response")
yhat.lasso<-ifelse(pred.final>=0.5,1,0)
#accuracy of lasso#
accuracy.lasso<- function(truth, predicted)
if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)
accuracy.lasso.f<-accuracy.lasso(y.test,yhat.lasso)
```

```
#sensitivity of lasso model#
sensitivity.lasso<- function(truth, predicted)
# 1 means positive (present)#
if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)
sensitivity.lasso.f<-sensitivity.lasso(y.test,yhat.lasso)
#specificity of lasso model
specificity.lasso<- function(truth, predicted)
if(sum(truth==0) > 0)
sum(predicted[truth==0]==0)/sum(truth==0)   else
return(0)
specificity.lasso.f<-specificity.lasso(y.test,yhat.lasso)

#AUC of lasso
Auc.lasso.f<-roc.area(obs=y.test, pred=pred.final)$A
#All value
All.val.lasso<-c(accuracy.lasso.f,sensitivity.lasso.f,
specificity.lasso.f,Auc.lasso.f)
#selected variables in all model
st.m.bt<-as.matrix(fit.lasso.final$beta)
sld.v<-as.vector(which(st.m.bt[,1]!=0))
fn.vr<-colnames(dat.training[,sld.v])
# selected variables by lasso
tp.var.l<-intersect(imp.vrb,fn.vr)
# selected truely improtant variables
nm.tp.var.l<-length(tp.var.l)
# number of selected truely improtant variables
fn.vr.l<-length(fn.vr)
# number of selected variables by lasso
prsn.v.l<-(nm.tp.var.l/fn.vr.l)
# variable selection prcision for each lasso model
#(TP,TN,FP,FN)
mt.l<-table(pred=yhat.lasso,truth=y.test)
mt.v.l<-as.vector(mt.l)
tr.postv.l<-mt.v.l[4]
tr.ngtv.l<-mt.v.l[1]
flse.postv.l<-mt.v.l[2]
flse.ngtv.l<-mt.v.l[3]
#matrix for number of  best models and lamda#
#values and number of coeef each model#
opt.lmd.mdl.l<-c(fn.vr.l,nm.tp.var.l,prsn.v.l,
tr.postv.l,tr.ngtv.l,flse.postv.l
,flse.ngtv.l,accuracy.lasso.f,
```

```
sensitivity.lasso.f,
specificity.lasso.f,Auc.lasso.f)
mat.lmd.md.lasso[l,1:11]=opt.lmd.mdl.l
best.models.lasso[[l]]<-fn.vr
#adaptive Lasso
wt <- adaptive.weights(x=X, y=y, weight.method="univariate")
cv.fit.adl<- cv.glmnet(x=X, y=y, family="binomial",
alpha=1, nlambda=100,
penalty.factor=as.numeric(wt$weights),
standardize=TRUE)
lam.al<-cv.fit.adl$lambda.min
fit.ad.l<- glmnet(x=X, y=y, family="binomial",
alpha = 1,lambda=lam.al, standardize = T,
penalty.factor=as.numeric(wt$weights),
thresh = 1e-07, maxit=1000)
pred.adl<- predict(fit.ad.l,
newx=X.test, s=lam.al, type="response")
yhat.adl<-ifelse(pred.adl>=0.5,1,0)
#accuracy of adaptive adaptive lasso
accuracy.alasso.f<-accuracy.lasso(y.test,yhat.adl)
#sensitivity of adaptive lasso model
sensitivity.alasso.f<-sensitivity.lasso(y.test,yhat.adl)
#specificity of adaptive lasso model
specificity.alasso.f<-specificity.lasso(y.test,yhat.adl)
#AUC of lasso
Auc.alasso.f<-roc.area(obs=y.test, pred=pred.adl)$A
#All value
All.val.alasso<-c(accuracy.alasso.f,
sensitivity.alasso.f,specificity.alasso.f,
Auc.alasso.f)
#selected variables in all model
st.m.abt<-as.matrix(fit.ad.l$beta)
sld.av<-as.vector(which(st.m.abt[,1]!=0))
fn.avr<-colnames(dat.training[,sld.av])
# selected variables by adative lasso
tp.var.al<-intersect(imp.vrb,fn.avr)
# selected truely improtant variables
nm.tp.var.al<-length(tp.var.al)
# number of selected truely improtant variables
fn.vr.al<-length(fn.avr)
# number of selected variables by adative lasso
prsn.v.al<-(nm.tp.var.al/fn.vr.al)
# variable selection prcision for each adative lasso model
#(TP,TN,FP,FN)
mt.al<-table(pred=yhat.adl,truth=y.test)
```

```
mt.v.al<-as.vector(mt.al)
tr.postv.al<-mt.v.al[4]
tr.ngtv.al<-mt.v.al[1]
flse.postv.al<-mt.v.al[2]
flse.ngtv.al<-mt.v.al[3]
#matrix for number of  best models and
#lamda values and number of coeef each model#
opt.lmd.mdl.al<-c(fn.vr.al,nm.tp.var.al,
prsn.v.al,tr.postv.al,tr.ngtv.al,
lse.postv.al
,flse.ngtv.al,accuracy.alasso.f,
sensitivity.alasso.f,specificity.alasso.f,Auc.alasso.f)
mat.lmd.md.alasso[l,1:11]=opt.lmd.mdl.al
best.models.alasso[[l]]<-fn.avr
#Elastic Net
a <- seq(0.1, 0.9, 0.05)
search <- foreach(i = a, .combine = rbind) %dopar% {
cv <- cv.glmnet(X, y, family = "binomial",
nfold = 10, type.measure = "deviance",
paralle = TRUE, alpha = i)
data.frame(cvm = cv$cvm[cv$lambda == cv$lambda.1se],
lambda.1se = cv$lambda.1se, alpha = i)
}
cv3 <- search[search$cvm == min(search$cvm), ]
fit.el.net<- glmnet(X, y, family = "binomial",
lambda = cv3$lambda.1se, alpha = cv3$alpha)
#prediction
pred.final<- predict(fit.el.net,
newx=X.test, s=cv3$lambda.1se,
alpha =cv3$alpha,
type="response")
yhat.el<-ifelse(pred.final>=0.5,1,0)
#accuracy
accuracy.el.f<-accuracy.lasso(y.test,yhat.el)
#sensitivity
sensitivity.el.f<-sensitivity.lasso(y.test,yhat.el)
#specificity
specificity.el.f<-specificity.lasso(y.test,yhat.el)
#Auc
Auc.el.f<-roc.area(obs=y.test, pred=pred.final)$A
All.val.el<-c(accuracy.el.f,sensitivity.el.f,
specificity.el.f,Auc.el.f)

#selected variables in all model
st.m.elbt<-as.matrix(fit.el.net$beta)
```

```
sld.elv<-as.vector(which(st.m.elbt[,1]!=0))
fn.elvr<-colnames(dat.training[,sld.elv])
# selected variables by elastic net
tp.var.ell<-intersect(imp.vrb,fn.elvr)
# selected truely improtant variables
nm.tp.var.ell<-length(tp.var.ell)
# number of selected truely improtant variables
fn.vr.ell<-length(fn.elvr)
# number of selected variables by elastic net
prsn.v.ell<-(nm.tp.var.ell/fn.vr.ell)
# variable selection prcision for each elastic net model
#(TP,TN,FP,FN)
mt.ell<-table(pred=yhat.el,truth=y.test)
mt.v.ell<-as.vector(mt.ell)
tr.postv.ell<-mt.v.ell[4]
tr.ngtv.ell<-mt.v.ell[1]
flse.postv.ell<-mt.v.ell[2]
flse.ngtv.ell<-mt.v.ell[3]
#matrix for number of  best models and lamda
#values and number of coeef each model.#
opt.lmd.mdl.ell<-c(fn.vr.ell,nm.tp.var.ell,prsn.v.ell,
tr.postv.ell,tr.ngtv.ell,flse.postv.ell
,flse.ngtv.ell,accuracy.el.f,
sensitivity.el.f,specificity.el.f,Auc.el.f)
mat.lmd.md.ellasso[l,1:11]=opt.lmd.mdl.ell
best.models.ellasso[[l]]<-fn.elvr
}
Sys.time()-Beg
colnames(mat.lmd.md)<-paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv","flse.postv",
"flse.ngtv","accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All<-Reduce(intersect,best.models)
colnames(mat.lmd.md.lasso)<-paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv","flse.postv","flse.ngtv",
"accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All.lasso<-Reduce(intersect,best.models.lasso)
colnames(mat.lmd.md.alasso)<-paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv","flse.postv","
flse.ngtv","accuracy","sensitivity",
"specificity","AUC"))
```

```
shared.Model.All.alasso<-Reduce(intersect,best.models.alasso)
colnames(mat.lmd.md.ellasso)<-paste(c("slctd.vr.f.m",
"true.imp.vr.m",
"prscn.var",
"tr.postv","tr.ngtv",
"flse.postv","flse.ngtv",
"accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All.ellasso<-Reduce(intersect,best.models.ellasso)
optim.corr
optim.alp
round(colMeans(mat.lmd.md),digits = 3)
round(colMeans(mat.lmd.md.lasso),digits = 3)
round(colMeans(mat.lmd.md.alasso),digits = 3)
round(colMeans(mat.lmd.md.ellasso),digits = 3)
```

```{r}
round(apply(mat.lmd.md,2,sd),digits = 2)
round(apply(mat.lmd.md.lasso,2,sd),digits = 2)
round(apply(mat.lmd.md.alasso,2,sd),digits = 2)
round(apply(mat.lmd.md.ellasso,2,sd),digits = 2)
```

```{r}
require(graphics)
par(mfrow=c(2,1))
methods<-as.factor(c(rep("ranked-based",200),
rep("lasso",200),rep("a_lasso",200),rep("enet",200)))
total.var.model<-c(mat.lmd.md[,1],
mat.lmd.md.lasso[,1],mat.lmd.md.alasso[,1],
mat.lmd.md.ellasso[,1])
true.imp.var.model<-c(mat.lmd.md[,2],
mat.lmd.md.lasso[,2],
mat.lmd.md.alasso[,2],
mat.lmd.md.ellasso[,2])
dat.sim1.f.var<-data.frame(methods,
total.var.model,
true.imp.var.model)
boxplot(total.var.model~methods,
col=rainbow(4),xlab = "methods",
ylab = "total number of variables in model",
data=dat.sim1.f.var)
boxplot(true.imp.var.model~methods,
```

```
col=rainbow(4),xlab = "methods",
ylab = "true important variables in model",
data=dat.sim1.f.var)
```

```
---
title: "200_times simulation_2"
output: word_document
---
```

```
#############################################################
#Simulating Data from the multivariate normal Distribution#
#scenario 2                                                 #
#############################################################
```{r,warning=FALSE}
set.seed(9516)
library(rockchalk)
library(MASS)
library(pclogit)
library(glmnet)
library(MESS)
library(verification)
mat.lmd.md=matrix(0,nrow=200,ncol=11)
mat.lmd.md.lasso=matrix(0,nrow=200,ncol=11)
mat.lmd.md.alasso=matrix(0,nrow=200,ncol=11)
mat.lmd.md.ellasso=matrix(0,nrow=200,ncol=11)
best.models<-vector("list", 200)
best.models.lasso<-vector("list", 200)
best.models.alasso<-vector("list", 200)
best.models.ellasso<-vector("list", 200)
Beg=Sys.time()
for (l in 1:200)
{
mycor= matrix(data=0, nrow=1000, ncol=1000)
#####################################
#First five variables has correlation#
#among them, which is 0.4           #
#####################################
for(j in 1:5){
for(i in 1:5){
mycor[i,j] = (.4)
}
}
```
```

```
diag(mycor)<-1
myCov <- lazyCov(Rho = mycor, Sd = rep(0.3,1000))
# variance is 0.3
set.seed(l)
datm1<-mvrnorm(n=200, mu=rep(0,1000), Sigma =myCov)
colnames(datm1)<-paste('var',1:1000)
beta<-c(c(2.0,2.0,2.0,2.7,2.0,2.0,2.5,2.7,
-2.8,3.0,2.6,3,3,3,3),rep(0,985))
#######################################
# first fiftheen variables            #
#are tue important, and others are zero#
#######################################
beta<-as.vector(beta)

z<-(datm1%*%beta)
prob<- 1/(1 + exp(-z))
y<-ifelse(prob>=.5,1,0)
y<-as.vector(y)
datm<-cbind(datm1,y)
datm.1<-data.frame(datm)
#imporatant variables need to compare later
imp.vrb<-colnames(datm.1[,1:15])
#########################################################
#filtering the data by variable screening              #
#(Benjamini & Hochberg (1995) ("BH" or its alias "fdr")#
#by adjustedp-values)                                   #
#########################################################

datm.1.s<-datm.1[,-c(length(colnames(datm.1)))]
y<-datm.1[,c(length(colnames(datm.1)))]
p = apply(datm.1.s,2,function(x)t.test(x[y==0],x[y==1])$p.value)
padj = p.adjust(p,method="BH")
sel.col<-as.vector(names(padj[(padj<0.3)]))
datm.f<-datm.1.s[,c(sel.col)]
datm.f.1<-cbind(datm.f,y)
#final important variables after adjusted p-value#
imp.vrb<-intersect(imp.vrb,colnames(datm.f))
#partitioning the data#

data.col<-datm.f.1
n1<-NROW(data.col)
set.seed(120)
  id.test<-sample(1:2,size=n1,replace=TRUE,prob=c(1/2,1/2))
dat.training<-data.col[id.test==1,]
dat.test<-data.col[id.test==2,]
```

```
train<-dat.training[,-c(length(colnames(data.col)))]
y.train<-dat.training$y
test<-dat.test[,-c(length(colnames(data.col)))]
y.test<-dat.test$y

set.seed(1223)
V<-2
index.cv<-sample(1:V,size=NROW(train),replace=TRUE)
mcer=al.cv=cr.cv=cv.v=NULL
for(v in 1:V)
{
train.v<-dat.training[index.cv!=v,]
valid.v<-dat.training[index.cv==v,]
train.cv<-train.v[,-c(length(colnames(data.col)))]
valid.cv<-valid.v[,-c(length(colnames(data.col)))]
y.train<-train.v$y
yobs<-valid.v$y
adjm.train.cv<-cor((train.cv))
diag(adjm.train.cv)<-0
# Randomly assigned correlation values
#to compute the adjacency matrix #
cr<-c(0.1,0.2,0.3,0.4)
for(i in 1:length(cr))
{
adjm.train.cv[abs(adjm.train.cv)<(cr[i])]<-0
adjm.train.cv[abs(adjm.train.cv)>=(cr[i])]<-1
sg.train.cv<-sign(rnorm(ncol(train.cv)))
al<-c(0.4,0.5,0.6,0.7)
for(k in 1:length(al))
{
g.ring.pc.cv<-pclogit(train.cv,
y.train,alpha=(al[k]),
group=adjm.train.cv
,sgnc=sg.train.cv)
bta.cv<-g.ring.pc.cv$beta
bta.cv<-as.matrix(bta.cv)
N<-ncol(bta.cv)
accuracy=sensitivity=specificity=auc=NULL
for(j in 1:N)
{
i.nm.cv<-as.vector(which(bta.cv[,j]!=0))
bt.a.cv<-bta.cv[,j]
b.bta.cv<-bt.a.cv[i.nm.cv]
B.cv<-matrix(b.bta.cv,ncol=1)
d.t.cv<-(train.cv[,which(bta.cv[,j]!=0)])
```

```
dt.cv<-as.matrix(d.t.cv)
e.cv<-exp(dt.cv%*%B.cv)
p.cv<-(e.cv/(1+e.cv))
yhat.cv<-ifelse(p.cv>=0.5,1,0)
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
ac<-accuracy.own(y.train,yhat.cv)
accuracy=c(accuracy,ac)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)}
sen<-sensitivity.own(y.train,yhat.cv)
sensitivity=c(sensitivity,sen)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
spe<-specificity.own(y.train,yhat.cv)
specificity=c(specificity,spe)
AUC<-roc.area(obs=y.train, pred=p.cv)$A
auc<-c(auc,AUC)
}
Rank.M<-matrix(c(as.numeric(factor(accuracy)),
as.numeric(factor(sensitivity)),
as.numeric(factor(specificity)),
as.numeric(factor(auc))),
nrow=4, ncol=N,byrow=TRUE)
Mean.col=NULL
for (L in 1:NCOL(Rank.M) )
{
A.column<-function(x)
{
a<-mean(Rank.M[,x])
}
mn<-A.column(L)
Mean.col=c(Mean.col,mn)
}
Mean.col<-as.vector(as.numeric(Mean.col))
O.Rank<-as.matrix(Mean.col)
m<-max(as.numeric(O.Rank[,1]))
```

```
optimal<-which(O.Rank[,1]==m)[1]
B.t<-as.matrix(bta.cv[which(bta.cv[,optimal]!=0),][,optimal])
dt.t<-as.matrix(valid.cv[,which(bta.cv[,optimal]!=0)])
e.t<-exp(dt.t%*%B.t)
p.t<-(e.t/(1+e.t))
yhat.test<-ifelse(p.t>=0.5,1,0)
cm<-table(actual=yobs, fitted=yhat.test)
er.cm<-1-sum(diag(cm))/sum(cm)
mcer<-c(mcer,er.cm)
al.cv<-c(al.cv,(al[k]))
cr.cv<-c(cr.cv,(cr[i]))
cv.v<-c(cv.v,v)
}
}
crs.mat<-matrix(c(mcer,al.cv,cr.cv,cv.v),ncol=4)
cl.min<-which.min(crs.mat[,1])
optim.alp<-crs.mat[cl.min,2]
optim.corr<-as.numeric(crs.mat[cl.min,3])
#####################################################
#Here to select best alpha value and best         #
#correlation for adjacency matrix we use          #
#resampling from the train data set. We           #
#compute missclafication error rate for           #
#selecting the best alpha and best correlation.   #
#The combination of alpha nad correlation give    #
#minimum missclassification error rate , those    #
#are our best values.                             #
#####################################################
adjm.train<-cor((train))
diag(adjm.train)<-0
adjm.train[abs(adjm.train)<optim.corr]<-0
adjm.train[abs(adjm.train)>=optim.corr]<-1
#####################################################
# We used optimum correlation value from           #
#validation data for computing the adjacency matrix.#
#Which will used to compute the laplacian matrix in #
#"pclogit" function                                 #
#####################################################
sg.train<-sign(rnorm(ncol(train)))
y.train<-dat.training$y
set.seed(257)
#####################################################################
#We applied the training data for trained our "pclogit" model#
#####################################################################
g.ring.pc<-pclogit(train,y.train,
```

```
alpha=optim.alp,group=adjm.train,
sgnc=sg.train)
bta<-g.ring.pc$beta
bta<-as.matrix(bta)
#########################################################
#Here we extract the sparse matrix of beta              #
#values with respect to all different                   #
#lamda values. The dimension will be (number of         #
#variables by number of lambda values). later           #
#on we build up number of different logistic            #
#regression model which is equal number of different    #
#lamda values. We only consider the variables for       #
#each model which has non  zero coefficient.            #
#########################################################
ac.lambda<-g.ring.pc$lambda
library(verification)
N<-ncol(bta)
accuracy=sensitivity=specificity=auc=NULL
for(j in 1:N)
{
i.nm<-as.vector(which(bta[,j]!=0))
bt.a<-bta[,j]
b.bta<-bt.a[i.nm]
B<-matrix(b.bta,ncol=1)
d.t<-(train[,which(bta[,j]!=0)])
dt<-as.matrix(d.t)
e<-exp(dt%*%B)
p<-(e/(1+e))
yhat<-ifelse(p>=0.5,1,0)
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
ac<-accuracy.own(y.train,yhat)
accuracy=c(accuracy,ac)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)}
sen<-sensitivity.own(y.train,yhat)
sensitivity=c(sensitivity,sen)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
```

```
}
}
spe<-specificity.own(y.train,yhat)
specificity=c(specificity,spe)
AUC<-roc.area(obs=y.train, pred=p)$A
auc<-c(auc,AUC)
}
Rank.M<-matrix(c(as.numeric(factor(accuracy)),
as.numeric(factor(sensitivity)),
as.numeric(factor(specificity)),
as.numeric(factor(auc))),
nrow=4, ncol=N,byrow=TRUE)
################################################################
#We computed teh accuracy, sensitivity , specificity and     #
#AUC values of all different model And created a Matrix,     #
#we values. Later on we take the average of each colum and   #
#found which has the maximum average value is our best model.#
#In case of ties we considered the more parsimonious model.  #
################################################################
Mean.col=NULL
for (i in 1:NCOL(Rank.M) )
{
A.column<-function(x)
{
a<-mean(Rank.M[,x])
}
mn<-A.column(i)
Mean.col=c(Mean.col,mn)
}
Mean.col<-as.vector(as.numeric(Mean.col))
O.Rank<-as.matrix(Mean.col)
m<-max(as.numeric(O.Rank[,1]))
optimal<-which(O.Rank[,1]==m)[1]
B.t<-as.matrix(bta[which(bta[,optimal]!=0),][,optimal])
dt.t<-as.matrix(test[,which(bta[,optimal]!=0)])
e.t<-exp(dt.t%*%B.t)
p.t<-(e.t/(1+e.t))
yhat.test<-ifelse(p.t>=0.5,1,0)
##########################################
#accuracy, sensitivity, specificity and AUC#
#value for best model on test data         #
##########################################
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
```

```
return(0)}
accuracy.test<-accuracy.own(y.test,yhat.test)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)}
sensitivity.test<-sensitivity.own(y.test,yhat.test)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
specificity.test<-specificity.own(y.test,yhat.test)

AUC.test<-roc.area(obs=y.test, pred=p.t)$A
#(TP,TN,FP,FN)
mt<-table(pred=yhat.test,truth=y.test)
mt.v<-as.vector(mt)
tr.postv<-mt.v[4]
tr.ngtv<-mt.v[1]
flse.postv<-mt.v[2]
flse.ngtv<-mt.v[3]
############################################################
#I applied our best model on test data set then computed #
#the accuracy, sensitivity, specificity,AUC values ,     #
#True positive,  True negative, False positiveand        #
#False negative value                                     #
############################################################
opt.modl<-optimal
lmda.opt<-ac.lambda[optimal]
#matrix for number of  best models and lamda values
op.nm<-as.vector(which(bta[,optimal]!=0))
op.bt.a<-bta[,optimal]
op.b.bta<-op.bt.a[op.nm]
lenth.opt.modl<-length(op.b.bta)
tp.var<-intersect(imp.vrb,names(op.b.bta))
nm.tp.var<-length(tp.var)
precsn.vrb<-(nm.tp.var/lenth.opt.modl)
#######################################################
# We can get the number of variables in each        #
#model and number of truley important variables     #
#selcted by  each model. Then we can compute the    #
#precision of selected truely important varibales   #
#for each model.matrix for number of  best models   #
```

```
#and lamda values and number of coeef each model    #
#####################################################
opt.lmd.mdl<-
c(lenth.opt.modl,nm.tp.var,
precsn.vrb,tr.postv,tr.ngtv,flse.postv
,flse.ngtv,accuracy.test,sensitivity.test,
specificity.test,AUC.test)
mat.lmd.md[l,1:11]=opt.lmd.mdl
#list for best models
best.models[[l]]<-names(op.b.bta)
#LASSO
X<-model.matrix(y~.-1,data=dat.training)
#(without intercept)
y<-dat.training$y
X.test<-model.matrix(y~.-1,data=dat.test)
fit.lasso<-glmnet(x=X,y=y,family="binomial",alpha = 1)
#cross-validation for glmnet
lambda<-fit.lasso$lambda
cv.lasso <- cv.glmnet(x=X, y=y,
alpha =1, lambda = lambda, nfolds=10)
lmbd0<-cv.lasso$lambda.min
#fit final model
fit.lasso.final<- glmnet(x=X,
y=y, family="binomial", alpha = 1,
lambda=lmbd0, standardize = T,
thresh = 1e-07, maxit=1000)
#prediction
pred.final<- predict(fit.lasso.final,
newx=X.test, s=lmbd0, type="response")
yhat.lasso<-ifelse(pred.final>=0.5,1,0)
#accuracy of lasso
accuracy.lasso<- function(truth, predicted)
if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)
accuracy.lasso.f<-accuracy.lasso(y.test,yhat.lasso)
#sensitivity of lasso model
sensitivity.lasso<- function(truth, predicted)
# 1 means positive (present)
if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)
sensitivity.lasso.f<-sensitivity.lasso(y.test,yhat.lasso)
#specificity of lasso model
specificity.lasso<- function(truth, predicted)
```

```
if(sum(truth==0) > 0)
sum(predicted[truth==0]==0)/sum(truth==0)   else
return(0)
specificity.lasso.f<-specificity.lasso(y.test,yhat.lasso)
#AUC of lasso
Auc.lasso.f<-roc.area(obs=y.test, pred=pred.final)$A
#All value
All.val.lasso<-c(accuracy.lasso.f,
sensitivity.lasso.f,specificity.lasso.f,
Auc.lasso.f)
#selected variables in all model
st.m.bt<-as.matrix(fit.lasso.final$beta)
sld.v<-as.vector(which(st.m.bt[,1]!=0))
fn.vr<-colnames(dat.training[,sld.v])
# selected variables by lasso
tp.var.l<-intersect(imp.vrb,fn.vr)
# selected truely improtant variables
nm.tp.var.l<-length(tp.var.l)
# number of selected truely improtant variables
fn.vr.l<-length(fn.vr)
# number of selected variables by lasso
prsn.v.l<-(nm.tp.var.l/fn.vr.l)
# variable selection prcision for each lasso model
#(TP,TN,FP,FN)
mt.l<-table(pred=yhat.lasso,truth=y.test)
mt.v.l<-as.vector(mt.l)
tr.postv.l<-mt.v.l[4]
tr.ngtv.l<-mt.v.l[1]
flse.postv.l<-mt.v.l[2]
flse.ngtv.l<-mt.v.l[3]
#matrix for number of  best models and
#lamda values and number of coeef each model
opt.lmd.mdl.l<-c(fn.vr.l,nm.tp.var.l,
prsn.v.l,tr.postv.l,
tr.ngtv.l,flse.postv.l
,flse.ngtv.l,accuracy.lasso.f,
sensitivity.lasso.f,specificity.lasso.f,
Auc.lasso.f)
mat.lmd.md.lasso[l,1:11]=opt.lmd.mdl.l
best.models.lasso[[l]]<-fn.vr

#adaptive Lasso

wt <- adaptive.weights(x=X, y=y, weight.method="univariate")
cv.fit.adl<- cv.glmnet(x=X,
```

```
y=y, family="binomial", alpha=1, nlambda=100,
penalty.factor=as.numeric(wt$weights),
standardize=TRUE)

lam.al<-cv.fit.adl$lambda.min
fit.ad.l<- glmnet(x=X, y=y,
family="binomial", alpha = 1,
lambda=lam.al,standardize =T,
penalty.factor=as.numeric(wt$weights),
thresh = 1e-07, maxit=1000)

pred.adl<- predict(fit.ad.l,
newx=X.test, s=lam.al,
type="response")
yhat.adl<-ifelse(pred.adl>=0.5,1,0)

#accuracy of adaptive adaptive lasso
accuracy.alasso.f<-accuracy.lasso(y.test,yhat.adl)
#sensitivity of adaptive lasso model
sensitivity.alasso.f<-sensitivity.lasso(y.test,yhat.adl)
#specificity of adaptive lasso model
specificity.alasso.f<-specificity.lasso(y.test,yhat.adl)
#AUC of lasso
Auc.alasso.f<-roc.area(obs=y.test, pred=pred.adl)$A
#All value
All.val.alasso<-c(accuracy.alasso.f,
sensitivity.alasso.f,specificity.alasso.f,Auc.alasso.f)
#selected variables in all model
st.m.abt<-as.matrix(fit.ad.l$beta)
sld.av<-as.vector(which(st.m.abt[,1]!=0))
fn.avr<-colnames(dat.training[,sld.av])
# selected variables by adative lasso
tp.var.al<-intersect(imp.vrb,fn.avr)
# selected truely improtant variables
nm.tp.var.al<-length(tp.var.al)
# number of selected truely improtant variables
fn.vr.al<-length(fn.avr)
# number of selected variables by adative lasso
prsn.v.al<-(nm.tp.var.al/fn.vr.al)
# variable selection prcision for each adative lasso model
#(TP,TN,FP,FN)
mt.al<-table(pred=yhat.adl,truth=y.test)
mt.v.al<-as.vector(mt.al)
tr.postv.al<-mt.v.al[4]
tr.ngtv.al<-mt.v.al[1]
```

```
flse.postv.al<-mt.v.al[2]
flse.ngtv.al<-mt.v.al[3]
#matrix for number of  best models and
#lamda values and number of coeef each model
opt.lmd.mdl.al<-c(fn.vr.al,nm.tp.var.al,
prsn.v.al,tr.postv.al,tr.ngtv.al,
flse.postv.al
,flse.ngtv.al,accuracy.alasso.f,
sensitivity.alasso.f,
specificity.alasso.f,Auc.alasso.f)
mat.lmd.md.alasso[l,1:11]=opt.lmd.mdl.al
best.models.alasso[[l]]<-fn.avr
#Elastic Net
a <- seq(0.1, 0.9, 0.05)
search <- foreach(i = a, .combine = rbind) %dopar% {
cv <- cv.glmnet(X, y, family = "binomial",
nfold = 10,
type.measure = "deviance",
paralle = TRUE, alpha = i)
data.frame(cvm = cv$cvm[cv$lambda == cv$lambda.1se],
lambda.1se = cv$lambda.1se, alpha = i)
}
cv3 <- search[search$cvm == min(search$cvm), ]
fit.el.net<- glmnet(X, y,
family = "binomial", lambda = cv3$lambda.1se,
alpha = cv3$alpha)
#prediction
pred.final<- predict(fit.el.net,
newx=X.test, s=cv3$lambda.1se,alpha =cv3$alpha,
type="response")
yhat.el<-ifelse(pred.final>=0.5,1,0)
#accuracy
accuracy.el.f<-accuracy.lasso(y.test,yhat.el)
#sensitivity
sensitivity.el.f<-sensitivity.lasso(y.test,yhat.el)
#specificity
specificity.el.f<-specificity.lasso(y.test,yhat.el)
#Auc
Auc.el.f<-roc.area(obs=y.test, pred=pred.final)$A
All.val.el<-c(accuracy.el.f,sensitivity.el.f,
specificity.el.f,Auc.el.f)

#selected variables in all model
st.m.elbt<-as.matrix(fit.el.net$beta)
sld.elv<-as.vector(which(st.m.elbt[,1]!=0))
```

```
fn.elvr<-colnames(dat.training[,sld.elv])
# selected variables by elastic net
tp.var.ell<-intersect(imp.vrb,fn.elvr)
# selected truely improtant variables
nm.tp.var.ell<-length(tp.var.ell)
# number of selected truely improtant variables
fn.vr.ell<-length(fn.elvr)
# number of selected variables by elastic net
prsn.v.ell<-(nm.tp.var.ell/fn.vr.ell)
# variable selection prcision for each elastic net model
#(TP,TN,FP,FN)
mt.ell<-table(pred=yhat.el,truth=y.test)
mt.v.ell<-as.vector(mt.ell)
tr.postv.ell<-mt.v.ell[4]
tr.ngtv.ell<-mt.v.ell[1]
flse.postv.ell<-mt.v.ell[2]
flse.ngtv.ell<-mt.v.ell[3]
#matrix for number of  best models and
#lamda values and number of coeef each model#
opt.lmd.mdl.ell<-c(fn.vr.ell,nm.tp.var.ell,
prsn.v.ell,tr.postv.ell,
tr.ngtv.ell,flse.postv.ell
,flse.ngtv.ell,accuracy.el.f,
sensitivity.el.f,specificity.el.f,
Auc.el.f)
mat.lmd.md.ellasso[l,1:11]=opt.lmd.mdl.ell
best.models.ellasso[[l]]<-fn.elvr
}
Sys.time()-Beg
colnames(mat.lmd.md)<-
paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv","
flse.postv","flse.ngtv",
"accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All<-Reduce(intersect,best.models)
colnames(mat.lmd.md.lasso)<-paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv","flse.postv",
"flse.ngtv","accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All.lasso<-Reduce(intersect,best.models.lasso)
colnames(mat.lmd.md.alasso)<-paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var","tr.postv",
```

```
"tr.ngtv","flse.postv","flse.ngtv",
"accuracy","sensitivity","specificity","AUC"))
shared.Model.All.alasso<-Reduce(intersect,best.models.alasso)
colnames(mat.lmd.md.ellasso)<-
paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv","flse.postv",
"flse.ngtv","accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All.ellasso<-Reduce(intersect,best.models.ellasso)
round(colMeans(mat.lmd.md),digits = 3)
round(colMeans(mat.lmd.md.lasso),digits = 3)
round(colMeans(mat.lmd.md.alasso),digits = 3)
round(colMeans(mat.lmd.md.ellasso),digits = 3)
```

```{r}
round(apply(mat.lmd.md,2,sd),digits = 2)
round(apply(mat.lmd.md.lasso,2,sd),digits = 2)
round(apply(mat.lmd.md.alasso,2,sd),digits = 2)
round(apply(mat.lmd.md.ellasso,2,sd),digits = 2)
```

```{r}
require(graphics)
par(mfrow=c(2,1))
methods<-as.factor(c(rep("ranked-based",200),
rep("lasso",200),
rep("a_lasso",200),
rep("enet",200)))
total.var.model<-c(mat.lmd.md[,1],
mat.lmd.md.lasso[,1],
mat.lmd.md.alasso[,1],
mat.lmd.md.ellasso[,1])
true.imp.var.model<-c(mat.lmd.md[,2],
mat.lmd.md.lasso[,2],
mat.lmd.md.alasso[,2],
mat.lmd.md.ellasso[,2])
dat.sim2.f.var<-data.frame(methods,
total.var.model,
true.imp.var.model)
boxplot(total.var.model~methods,
col=rainbow(4),xlab = "methods",
ylab = "total number of variables in model",
```

```
data=dat.sim2.f.var)
boxplot(true.imp.var.model~methods,
col=rainbow(4),xlab = "methods",
ylab = "true important variables in model",
data=dat.sim2.f.var)
```

```
---
title: "100_times_simulation_3"
output: word_document
---

#############################################################
#Simulating Data from the multivariate normal Distribution#
#scenario 3                                                 #
#############################################################
```{r,warning=FALSE}
set.seed(9510)
library(rockchalk)
library(MASS)
library(pclogit)
library(glmnet)
library(MESS)
library(verification)
mat.lmd.md=matrix(0,nrow=100,ncol=11)
mat.lmd.md.lasso=matrix(0,nrow=100,ncol=11)
mat.lmd.md.alasso=matrix(0,nrow=100,ncol=11)
mat.lmd.md.ellasso=matrix(0,nrow=100,ncol=11)
best.models<-vector("list", 100)
best.models.lasso<-vector("list", 100)
best.models.alasso<-vector("list", 100)
best.models.ellasso<-vector("list", 100)
Beg=Sys.time()
for (l in 1:100)
{
mycor= matrix(data=0, nrow=1000, ncol=1000)
#######################################
#First five variables has correlation #
#among them, which is 0.3 then the    #
#corrlation among 11 to 30 is 0.6     #
#######################################

for(j in 1:5){
for(i in 1:5){
mycor[i,j] = (.3)
```
```

```
}
}

for(j in 11:30){
#11 to 30 varibales has the correlation which is 0.6.#
for(i in 11:30){
mycor[i,j] = (.6)
}
}
diag(mycor)<-1
myCov <- lazyCov(Rho = mycor, Sd = rep(0.4,1000))
# variance is 0.4
set.seed(l)
datm1<-mvrnorm(n=150, mu=rep(0,1000), Sigma =myCov)
colnames(datm1)<-paste('var',1:1000)
beta<-c(c(2,2,2,2,2,2.5,-2.6,2.7,3,-2.9,
2,2,2,2,2,2.5,-2,2.7,3,-2.5,2,2,2,2,2,2.5,-2,
2.7,3,-2.5),rep(0,970))

##############################################
#first five variables has coefficeient 2,   #
#then five ares other values ,then 20 are 3,#
#and others are zero                        #
##############################################
beta<-as.vector(beta)
z<-(datm1%*%beta)
prob<- 1/(1 + exp(-z))
y<-ifelse(prob>=.5,1,0)
y<-as.vector(y)
datm<-cbind(datm1,y)
datm.1<-data.frame(datm)
#imporatant variables need to compare later#
imp.vrb<-colnames(datm.1[,1:30])
############################################################
#filtering the data by variable screening                  #
#(Benjamini & Hochberg (1995) ("BH" or its alias "fdr")#
#by adjustedp-values)                                      #
############################################################

datm.1.s<-datm.1[,-c(length(colnames(datm.1)))]
y<-datm.1[,c(length(colnames(datm.1)))]
p = apply(datm.1.s,2,function(x)t.test(x[y==0],x[y==1])$p.value)
padj = p.adjust(p,method="BH")
sel.col<-as.vector(names(padj[(padj<0.45)]))
datm.f<-datm.1.s[,c(sel.col)]
```

```
datm.f.1<-cbind(datm.f,y)
#final important variables after adjusted p-value#
imp.vrb<-intersect(imp.vrb,colnames(datm.f))
#partitioning the data#
data.col<-datm.f.1
n1<-NROW(data.col)
set.seed(120)
id.test<-sample(1:2,size=n1,replace=TRUE,prob=c(2/3,1/3))
dat.training<-data.col[id.test==1,]
dat.test<-data.col[id.test==2,]
train<-dat.training[,-c(length(colnames(data.col)))]
y.train<-dat.training$y
test<-dat.test[,-c(length(colnames(data.col)))]
y.test<-dat.test$y
set.seed(1223)
V<-2
index.cv<-sample(1:V,size=NROW(train),replace=TRUE)
mcer=al.cv=cr.cv=cv.v=NULL
for(v in 1:V)
{
train.v<-dat.training[index.cv!=v,]
valid.v<-dat.training[index.cv==v,]
train.cv<-train.v[,-c(length(colnames(data.col)))]
valid.cv<-valid.v[,-c(length(colnames(data.col)))]
y.train<-train.v$y
yobs<-valid.v$y
adjm.train.cv<-cor((train.cv))
diag(adjm.train.cv)<-0
########################################
#Randomly assigned correlation         #
#values to compute the adjacency matrix #
########################################
cr<-c(0.1,0.2,0.3,0.5,0.6)
for(i in 1:length(cr))
{
adjm.train.cv[abs(adjm.train.cv)<(cr[i])]<-0
adjm.train.cv[abs(adjm.train.cv)>=(cr[i])]<-1
sg.train.cv<-sign(rnorm(ncol(train.cv)))
al<-c(0.05,0.1,0.2,0.3)
for(k in 1:length(al))
{
g.ring.pc.cv<-pclogit(train.cv,y.train,
alpha=(al[k]),group=adjm.train.cv
,sgnc=sg.train.cv)
bta.cv<-g.ring.pc.cv$beta
```

```
bta.cv<-as.matrix(bta.cv)
N<-ncol(bta.cv)
accuracy=sensitivity=specificity=auc=NULL
for(j in 1:N)
{
i.nm.cv<-as.vector(which(bta.cv[,j]!=0))
bt.a.cv<-bta.cv[,j]
b.bta.cv<-bt.a.cv[i.nm.cv]
B.cv<-matrix(b.bta.cv,ncol=1)
d.t.cv<-(train.cv[,which(bta.cv[,j]!=0)])
dt.cv<-as.matrix(d.t.cv)
e.cv<-exp(dt.cv%*%B.cv)
p.cv<-(e.cv/(1+e.cv))
yhat.cv<-ifelse(p.cv>=0.5,1,0)
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
ac<-accuracy.own(y.train,yhat.cv)
accuracy=c(accuracy,ac)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1) else
return(0)}
sen<-sensitivity.own(y.train,yhat.cv)
sensitivity=c(sensitivity,sen)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
spe<-specificity.own(y.train,yhat.cv)
specificity=c(specificity,spe)
AUC<-roc.area(obs=y.train, pred=p.cv)$A
auc<-c(auc,AUC)
}
Rank.M<-matrix(c(as.numeric(factor(accuracy)),
as.numeric(factor(sensitivity)),
as.numeric(factor(specificity)),
as.numeric(factor(auc))), nrow=4,
ncol=N,byrow=TRUE)
Mean.col=NULL
for (L in 1:NCOL(Rank.M) )
{
```

```
A.column<-function(x)
{
a<-mean(Rank.M[,x])
}
mn<-A.column(L)
Mean.col=c(Mean.col,mn)
}
Mean.col<-as.vector(as.numeric(Mean.col))
O.Rank<-as.matrix(Mean.col)
m<-max(as.numeric(O.Rank[,1]))
optimal<-which(O.Rank[,1]==m)[1]
B.t<-as.matrix(bta.cv[which(bta.cv[,optimal]!=0),
][,optimal])
dt.t<-
as.matrix(valid.cv[,which(bta.cv[,optimal]!=0)])
e.t<-exp(dt.t%*%B.t)
p.t<-(e.t/(1+e.t))
yhat.test<-ifelse(p.t>=0.5,1,0)
cm<-table(actual=yobs, fitted=yhat.test)
er.cm<-1-sum(diag(cm))/sum(cm)
mcer<-c(mcer,er.cm)
al.cv<-c(al.cv,(al[k]))
cr.cv<-c(cr.cv,(cr[i]))
cv.v<-c(cv.v,v)
}
}
}
crs.mat<-matrix(c(mcer,al.cv,cr.cv,cv.v),ncol=4)
cl.min<-which.min(crs.mat[,1])
optim.alp<-crs.mat[cl.min,2]
optim.corr<-as.numeric(crs.mat[cl.min,3])
#####################################################
#Here to select best alpha value and best         #
#correlation for adjacency matrix we use          #
#resampling from the train data set. We           #
#compute missclafication error rate for           #
#selecting the best alpha and best correlation.   #
#The combination of alpha nad correlation give    #
#minimum missclassification error rate , those    #
#are our best values.                             #
#####################################################
adjm.train<-cor((train))
diag(adjm.train)<-0
adjm.train[abs(adjm.train)<optim.corr]<-0
adjm.train[abs(adjm.train)>=optim.corr]<-1
```

```
#######################################################
# We used optimum correlation value from            #
#validation data for computing the adjacency matrix.#
#Which will used to compute the laplacian matrix in #
#"pclogit" function                                  #
#######################################################
sg.train<-sign(rnorm(ncol(train)))
#We also need to consider the sign#
y.train<-dat.training$y
#################################################################
#We applied the training data for trained our "pclogit" model#
#################################################################
set.seed(257)
g.ring.pc<-pclogit(train,y.train,
alpha=optim.alp,group=adjm.train,
sgnc=sg.train)
bta<-g.ring.pc$beta
bta<-as.matrix(bta)
#########################################################
#Here we extract the sparse matrix of beta            #
#values with respect to all different                 #
#lamda values. The dimension will be (number of       #
#variables by number of lambda values). later         #
#on we build up number of different logistic          #
#regression model which is equal number of different  #
#lamda values. We only consider the variables for     #
#each model which has non  zero coefficient.          #
#########################################################
ac.lambda<-g.ring.pc$lambda
library(verification)
N<-ncol(bta)
accuracy=sensitivity=specificity=auc=NULL
for(j in 1:N)
{
i.nm<-as.vector(which(bta[,j]!=0))
bt.a<-bta[,j]
b.bta<-bt.a[i.nm]
B<-matrix(b.bta,ncol=1)
d.t<-(train[,which(bta[,j]!=0)])
dt<-as.matrix(d.t)
e<-exp(dt%*%B)
p<-(e/(1+e))
yhat<-ifelse(p>=0.5,1,0)
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
```

```
sum(truth==predicted)/length(truth) else
return(0)}
ac<-accuracy.own(y.train,yhat)
accuracy=c(accuracy,ac)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)}
sen<-sensitivity.own(y.train,yhat)
sensitivity=c(sensitivity,sen)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
spe<-specificity.own(y.train,yhat)
specificity=c(specificity,spe)
AUC<-roc.area(obs=y.train, pred=p)$A
auc<-c(auc,AUC)
}
Rank.M<-matrix(c(as.numeric(factor(accuracy)),
as.numeric(factor(sensitivity)),
as.numeric(factor(specificity)),
as.numeric(factor(auc))),
nrow=4, ncol=N,byrow=TRUE)
#################################################################
#We computed teh accuracy, sensitivity , specificity and     #
#AUC values of all different model And created a Matrix,     #
#we values. Later on we take the average of each colum and   #
#found which has the maximum average value is our best model.#
#In case of ties we considered the more parsimonious model.  #
#################################################################
Mean.col=NULL
for (i in 1:NCOL(Rank.M) )
{
A.column<-function(x)
{
a<-mean(Rank.M[,x])
}
mn<-A.column(i)
Mean.col=c(Mean.col,mn)
}
Mean.col<-as.vector(as.numeric(Mean.col))
O.Rank<-as.matrix(Mean.col)
```

```
m<-max(as.numeric(O.Rank[,1]))
optimal<-which(O.Rank[,1]==m)[1]
B.t<-as.matrix(bta[which(bta[,optimal]!=0),][,optimal])
dt.t<-as.matrix(test[,which(bta[,optimal]!=0)])
e.t<-exp(dt.t%*%B.t)
p.t<-(e.t/(1+e.t))
yhat.test<-ifelse(p.t>=0.5,1,0)

########################################
#accuracy, sensitivity, specificity and AUC#
#value for best model on test data         #
########################################
accuracy.own<- function(truth, predicted)
{if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)}
accuracy.test<-accuracy.own(y.test,yhat.test)
sensitivity.own <- function(truth, predicted)
{if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)}
sensitivity.test<-sensitivity.own(y.test,yhat.test)
specificity.own <- function(truth, predicted)
{ if(sum(truth==0) > 0)
{
return(sum(predicted[truth==0]==0)/sum(truth==0))
}
}
specificity.test<-specificity.own(y.test,yhat.test)
AUC.test<-roc.area(obs=y.test, pred=p.t)$A
#(TP,TN,FP,FN)
mt<-table(pred=yhat.test,truth=y.test)
mt.v<-as.vector(mt)
tr.postv<-mt.v[4]
tr.ngtv<-mt.v[1]
flse.postv<-mt.v[2]
flse.ngtv<-mt.v[3]
#############################################################
#I applied our best model on test data set then computed #
#the accuracy, sensitivity, specificity,AUC values ,     #
#True positive,  True negative, False positiveand        #
#False negative value                                     #
#############################################################
opt.modl<-optimal
lmda.opt<-ac.lambda[optimal]
```

```
#matrix for number of  best models and lamda values
op.nm<-as.vector(which(bta[,optimal]!=0))
op.bt.a<-bta[,optimal]
op.b.bta<-op.bt.a[op.nm]
lenth.opt.modl<-length(op.b.bta)
tp.var<-intersect(imp.vrb,names(op.b.bta))
nm.tp.var<-length(tp.var)
precsn.vrb<-(nm.tp.var/lenth.opt.modl)
#######################################################
# We can get the number of variables in each        #
#model and number of truley important variables     #
#selcted by  each model. Then we can compute the    #
#precision of selected truely important varibales   #
#for each model.matrix for number of  best models   #
#and lamda values and number of coeef each model    #
#######################################################
opt.lmd.mdl<-c(lenth.opt.modl,
nm.tp.var,precsn.vrb,tr.postv,
tr.ngtv,flse.postv,flse.ngtv,
accuracy.test,sensitivity.test,
specificity.test,AUC.test)
mat.lmd.md[l,1:11]=opt.lmd.mdl
#list for best models
best.models[[l]]<-names(op.b.bta)
#LASSO
library(glmnet)
X<-model.matrix(y~.-1,data=dat.training)
#(without intercept)
y<-dat.training$y
X.test<-model.matrix(y~.-1,data=dat.test)
fit.lasso<-glmnet(x=X,y=y,family="binomial",alpha = 1)
#cross-validation for glmnet
lambda<-fit.lasso$lambda
cv.lasso <- cv.glmnet(x=X, y=y,
alpha =1, lambda = lambda, nfolds=10)
lmbd0<-cv.lasso$lambda.min
#fit final model
fit.lasso.final<-glmnet(x=X,
y=y, family="binomial", alpha = 1,
lambda=lmbd0, standardize = T,
thresh = 1e-07, maxit=1000)
#prediction
pred.final<- predict(fit.lasso.final,
newx=X.test, s=lmbd0, type="response")
yhat.lasso<-ifelse(pred.final>=0.5,1,0)
```

```
#accuracy of lasso
accuracy.lasso<- function(truth, predicted)
if(length(truth) > 0)
sum(truth==predicted)/length(truth) else
return(0)
accuracy.lasso.f<-accuracy.lasso(y.test,yhat.lasso)
#sensitivity of lasso model
sensitivity.lasso<- function(truth, predicted)
# 1 means positive (present)
if(sum(truth==1) > 0)
sum(predicted[truth==1]==1)/sum(truth==1)   else
return(0)
sensitivity.lasso.f<-sensitivity.lasso(y.test,yhat.lasso)
#specificity of lasso model
specificity.lasso<- function(truth, predicted)
if(sum(truth==0) > 0)
sum(predicted[truth==0]==0)/sum(truth==0)   else
return(0)
specificity.lasso.f<-specificity.lasso(y.test,yhat.lasso)

#AUC of lasso
Auc.lasso.f<-roc.area(obs=y.test, pred=pred.final)$A
#All value
All.val.lasso<-c(accuracy.lasso.f,
sensitivity.lasso.f,specificity.lasso.f,
Auc.lasso.f)
#selected variables in all model
st.m.bt<-as.matrix(fit.lasso.final$beta)
sld.v<-as.vector(which(st.m.bt[,1]!=0))
fn.vr<-colnames(dat.training[,sld.v])
# selected variables by lasso
tp.var.l<-intersect(imp.vrb,fn.vr)
# selected truely improtant variables
nm.tp.var.l<-length(tp.var.l)
# number of selected truely improtant variables
fn.vr.l<-length(fn.vr)
# number of selected variables by lasso
prsn.v.l<-(nm.tp.var.l/fn.vr.l)
# variable selection prcision for each lasso model
#(TP,TN,FP,FN)
mt.l<-table(pred=yhat.lasso,truth=y.test)
mt.v.l<-as.vector(mt.l)
tr.postv.l<-mt.v.l[4]
tr.ngtv.l<-mt.v.l[1]
flse.postv.l<-mt.v.l[2]
```

```
flse.ngtv.l<-mt.v.l[3]
################################################
#matrix for number of  best models             #
#and lamda values and number of coeef each model#
################################################
opt.lmd.mdl.l<-c(fn.vr.l,nm.tp.var.l,
prsn.v.l,tr.postv.l,tr.ngtv.l,
flse.postv.l,flse.ngtv.l,accuracy.lasso.f,
sensitivity.lasso.f,specificity.lasso.f,Auc.lasso.f)
mat.lmd.md.lasso[l,1:11]=opt.lmd.mdl.l
best.models.lasso[[l]]<-fn.vr

#adaptive Lasso

library(MESS)
wt <- adaptive.weights(x=X,
y=y, weight.method="univariate")
cv.fit.adl<- cv.glmnet(x=X,
y=y, family="binomial",
alpha=1, nlambda=100,
penalty.factor=as.numeric(wt$weights),
standardize=TRUE)
lam.al<-cv.fit.adl$lambda.min
fit.ad.l<- glmnet(x=X, y=y,
family="binomial",
alpha = 1,lambda=lam.al,
standardize = T,
penalty.factor=as.numeric(wt$weights),
thresh = 1e-07, maxit=1000)
pred.adl<- predict(fit.ad.l,
newx=X.test, s=lam.al,
type="response")
yhat.adl<-ifelse(pred.adl>=0.5,1,0)
#accuracy of adaptive adaptive lasso
accuracy.alasso.f<-accuracy.lasso(y.test,yhat.adl)
#sensitivity of adaptive lasso model
sensitivity.alasso.f<-sensitivity.lasso(y.test,yhat.adl)
#specificity of adaptive lasso model
specificity.alasso.f<-specificity.lasso(y.test,yhat.adl)
#AUC of lasso
Auc.alasso.f<-roc.area(obs=y.test, pred=pred.adl)$A
#All value
All.val.alasso<-c(accuracy.alasso.f,
sensitivity.alasso.f,
specificity.alasso.f,Auc.alasso.f)
```

```
#selected variables in all model
st.m.abt<-as.matrix(fit.ad.l$beta)
sld.av<-as.vector(which(st.m.abt[,1]!=0))
fn.avr<-colnames(dat.training[,sld.av])
# selected variables by adative lasso
tp.var.al<-intersect(imp.vrb,fn.avr)
# selected truely improtant variables
nm.tp.var.al<-length(tp.var.al)
# number of selected truely improtant variables
fn.vr.al<-length(fn.avr)
# number of selected variables by adative lasso
prsn.v.al<-(nm.tp.var.al/fn.vr.al)
# variable selection prcision for each adative lasso model
#(TP,TN,FP,FN)
mt.al<-table(pred=yhat.adl,truth=y.test)
mt.v.al<-as.vector(mt.al)
tr.postv.al<-mt.v.al[4]
tr.ngtv.al<-mt.v.al[1]
flse.postv.al<-mt.v.al[2]
flse.ngtv.al<-mt.v.al[3]
##############################################
#matrix for number of  best models and      #
#lamda values and number of coeef each model#
##############################################
opt.lmd.mdl.al<-c(fn.vr.al,
nm.tp.var.al,prsn.v.al,
tr.postv.al,tr.ngtv.al,
flse.postv.al
,flse.ngtv.al,
accuracy.alasso.f,
sensitivity.alasso.f,
specificity.alasso.f,
Auc.alasso.f)
mat.lmd.md.alasso[l,1:11]=opt.lmd.mdl.al
best.models.alasso[[l]]<-fn.avr
#Elastic Net
a <- seq(0.1, 0.9, 0.05)
search <- foreach(i = a, .combine = rbind) %dopar% {
cv <- cv.glmnet(X, y,
family = "binomial",
nfold = 10, type.measure = "deviance",
paralle = TRUE, alpha = i)
data.frame(cvm = cv$cvm[cv$lambda == cv$lambda.1se],
lambda.1se = cv$lambda.1se, alpha = i)
}
```

```
cv3 <- search[search$cvm == min(search$cvm), ]
fit.el.net<- glmnet(X, y,
family = "binomial", lambda = cv3$lambda.1se,
alpha = cv3$alpha)
#prediction
pred.final<- predict(fit.el.net,
newx=X.test, s=cv3$lambda.1se,
alpha =cv3$alpha,type="response")
yhat.el<-ifelse(pred.final>=0.5,1,0)
#accuracy
accuracy.el.f<-accuracy.lasso(y.test,yhat.el)
#sensitivity
sensitivity.el.f<-sensitivity.lasso(y.test,yhat.el)
#specificity
specificity.el.f<-specificity.lasso(y.test,yhat.el)
#Auc
Auc.el.f<-roc.area(obs=y.test, pred=pred.final)$A
All.val.el<-c(accuracy.el.f,
sensitivity.el.f,
specificity.el.f,Auc.el.f)
#selected variables in all model
st.m.elbt<-as.matrix(fit.el.net$beta)
sld.elv<-as.vector(which(st.m.elbt[,1]!=0))
fn.elvr<-colnames(dat.training[,sld.elv])
# selected variables by elastic net
tp.var.ell<-intersect(imp.vrb,fn.elvr)
# selected truely improtant variables
nm.tp.var.ell<-length(tp.var.ell)
# number of selected truely improtant variables
fn.vr.ell<-length(fn.elvr)
# number of selected variables by elastic net
prsn.v.ell<-(nm.tp.var.ell/fn.vr.ell)
# variable selection prcision for each elastic net model
#(TP,TN,FP,FN)
mt.ell<-table(pred=yhat.el,truth=y.test)
mt.v.ell<-as.vector(mt.ell)
tr.postv.ell<-mt.v.ell[4]
tr.ngtv.ell<-mt.v.ell[1]
flse.postv.ell<-mt.v.ell[2]
flse.ngtv.ell<-mt.v.ell[3]
########################################
#matrix for number of  best            #
#models and lamda values and number of  #
#coeef each model                       #
########################################
```

```
opt.lmd.mdl.ell<-c(fn.vr.ell,
nm.tp.var.ell,prsn.v.ell,
tr.postv.ell,tr.ngtv.ell,flse.postv.ell
,flse.ngtv.ell,accuracy.el.f,
sensitivity.el.f,specificity.el.f,Auc.el.f)

mat.lmd.md.ellasso[l,1:11]=opt.lmd.mdl.ell
best.models.ellasso[[l]]<-fn.elvr
}
Sys.time()-Beg
colnames(mat.lmd.md)<-
paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv",
"flse.postv","flse.ngtv",
"accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All<-Reduce(intersect,best.models)
colnames(mat.lmd.md.lasso)<-
paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv",
"flse.postv","flse.ngtv",
"accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All.lasso<-Reduce(intersect,best.models.lasso)
colnames(mat.lmd.md.alasso)<-
paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv",
"flse.postv","flse.ngtv",
"accuracy","sensitivity",
"specificity","AUC"))
shared.Model.All.alasso<-Reduce(intersect,best.models.alasso)
colnames(mat.lmd.md.ellasso)<-
paste(c("slctd.vr.f.m",
"true.imp.vr.m","prscn.var",
"tr.postv","tr.ngtv",
"flse.postv",
"flse.ngtv",
"accuracy",
"sensitivity",
"specificity","AUC"))
shared.Model.All.ellasso<-Reduce(intersect,best.models.ellasso)
round(colMeans(mat.lmd.md),digits = 3)
```

```
round(colMeans(mat.lmd.md.lasso),digits = 3)
round(colMeans(mat.lmd.md.alasso),digits = 3)
round(colMeans(mat.lmd.md.ellasso),digits = 3)
```

```{r}
round(apply(mat.lmd.md,2,sd),digits = 2)
round(apply(mat.lmd.md.lasso,2,sd),digits = 2)
round(apply(mat.lmd.md.alasso,2,sd),digits = 2)
round(apply(mat.lmd.md.ellasso,2,sd),digits = 2)
```

```{r}
require(graphics)
par(mfrow=c(2,1))
methods<-as.factor(c(rep("ranked-based",100),
rep("lasso",100),rep("a_lasso",100),
rep("enet",100)))
total.var.model<-c(mat.lmd.md[,1],
mat.lmd.md.lasso[,1],mat.lmd.md.alasso[,1],
mat.lmd.md.ellasso[,1])
true.imp.var.model<-c(mat.lmd.md[,2],
mat.lmd.md.lasso[,2],
mat.lmd.md.alasso[,2],mat.lmd.md.ellasso[,2])
dat.sim3.f.var<-data.frame(methods,total.var.model,
true.imp.var.model)
boxplot(total.var.model~methods,
col=rainbow(4),
xlab = "methods",
ylab = "total number of variables in model",
data=dat.sim3.f.var)
boxplot(true.imp.var.model~methods,
col=rainbow(4),xlab = "methods",
ylab = "true important variables in model",
data=dat.sim3.f.var)
```

```
##############################################
#Box plot of total number of variables (NVS) #
#selected and the number of                  #
#true important variables (NTIV)             #
#within the number of variables selected     #
#with four different models under            #
#scenario 1 based on 200 replications        #
```

```
##############################################

require(graphics)
par(mfrow=c(2,1))
methods<-as.factor(c(rep("rank-based",200),
rep("lasso",200),rep("alasso",200),rep("enet",200)))
total.var.model<-c(mat.lmd.md[,1],
mat.lmd.md.lasso[,1],mat.lmd.md.alasso[,1],
mat.lmd.md.ellasso[,1])
true.imp.var.model<-c(mat.lmd.md[,2],
mat.lmd.md.lasso[,2],mat.lmd.md.alasso[,2],
mat.lmd.md.ellasso[,2])
dat.sim1.f.var<-data.frame(methods,
total.var.model,true.imp.var.model)
dat.sim1.f.var$methods=factor(dat.sim1.f.var$methods ,
levels=levels(dat.sim1.f.var$methods)[c(4,3,1,2)])
boxplot(dat.sim1.f.var$total.var.model~dat.sim1.f.var$methods,
col=rainbow(4),xlab = "methods",ylab = "NVS")
boxplot(dat.sim1.f.var$true.imp.var.model~dat.sim1.f.var$methods,
col=rainbow(4),xlab = "methods",ylab = "NTIV")

##############################################
#Box plot of total number of variables (NVS) #
#selected and the number of                  #
#true important variables (NTIV)             #
#within the number of variables selected     #
#with four different models under            #
#scenario 2 based on 200 replications        #
##############################################
par(mfrow=c(2,1))
methods<-as.factor(c(rep("rank-based",100),
rep("lasso",100),rep("alasso",100),rep("enet",100)))
total.var.model<-c(mat.lmd.md[,1],
mat.lmd.md.lasso[,1],mat.lmd.md.alasso[,1],
mat.lmd.md.ellasso[,1])
true.imp.var.model<-c(mat.lmd.md[,2],
mat.lmd.md.lasso[,2],mat.lmd.md.alasso[,2],
mat.lmd.md.ellasso[,2])
dat.sim2.f.var<-data.frame(methods,total.var.model,
true.imp.var.model)
dat.sim2.f.var$methods=factor(dat.sim2.f.var$methods,
levels=levels(dat.sim2.f.var$methods)[c(4,3,1,2)])
boxplot(dat.sim2.f.var$total.var.model~dat.sim2.f.var$methods,
col=rainbow(4),xlab = "methods",ylab = "NVS")
boxplot(dat.sim2.f.var$true.imp.var.model~dat.sim2.f.var$methods,
```

```
col=rainbow(4),xlab = "methods",ylab = "NTIV")

##############################################
#Box plot of total number of variables (NVS) #
#selected and the number of                  #
#true important variables (NTIV)             #
#within the number of variables selected     #
#with four different models under            #
#scenario 3 based on 150 replications        #
##############################################

require(graphics)
par(mfrow=c(2,1))
methods<-as.factor(c(rep("rank-based",100),
rep("lasso",100),rep("alasso",100),rep("enet",100)))
total.var.model<-c(mat.lmd.md[,1],
mat.lmd.md.lasso[,1],mat.lmd.md.alasso[,1],
mat.lmd.md.ellasso[,1])
true.imp.var.model<-c(mat.lmd.md[,2],
mat.lmd.md.lasso[,2],mat.lmd.md.alasso[,2],
mat.lmd.md.ellasso[,2])
dat.sim3.f.var<-data.frame(methods,
total.var.model,true.imp.var.model)
dat.sim3.f.var$methods=factor(dat.sim3.f.var$methods,
levels=levels(dat.sim3.f.var$methods)[c(4,3,1,2)])
boxplot(dat.sim3.f.var$total.var.model~dat.sim3.f.var$methods,
col=rainbow(4),xlab = "methods",ylab = "NVS")
boxplot(dat.sim3.f.var$true.imp.var.model~dat.sim3.f.var$methods,
col=rainbow(4),xlab = "methods",ylab = "NTIV")
```

Curriculum Vitae

Md Showaib Rahman Sarker was born on March 8^{th}, 1991. The second son of A.S.M. Zillur Rahman Sarker and Most. Akter Jahan, obtained a Bachelor of Science (in Statistics, Biostatistics and Informatics) from The University of Dhaka, one of the top ranked universities for science research in Bangladesh. He entered the University of Texas at El Paso in the fall of 2017. While pursuing his master's degree in Mathematical Sciences he worked as a Teaching Assistant. He is a member of Bangladesh Mathematician Club and American Statistical Association.

Email address: msarker@miners.utep.edu