FEATURE SCREENING OF ULTRAHIGH DIMENSIONAL FEATURE SPACES WITH APPLICATIONS IN INTERACTION SCREENING

by

Randall D. Reese

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Statistics

Approved:

Dr. D. Richard Cutler Major Professor	Dr. Rose Qingyang Hu Committee Member
Dr. John R. Stevens Committee Member	Dr. Jia Zhao Committee Member
Dr. Mark McLellan Dean of Graduate Studies	

UTAH STATE UNIVERSITY
Logan, Utah

2018

ABSTRACT

Feature Screening of Ultrahigh Dimensional Feature Spaces with Applications in Interaction Screening

by

Randall D. Reese, Doctor of Philosophy

Utah State University, 2018

Major Professor: Dr. D. Richard Cutler
Department: Mathematics and Statistics

Feature screening for ultrahigh dimensional feature spaces plays a critical role in the analysis of data sets whose predictors exponentially exceed the number of observations. Such data sets are becoming increasingly prevalent in areas such as bioinformatics, medical imaging, computer vision, and social network analysis. We propose three new methods for feature screening ultrahigh dimensional data. The first method proposed (TC-SIS) is specifically meant for the screening of data with categorical response and predictors. We next introduce a new interaction screening procedure (JCIS) based on joint cumulants. JCIS is not inhibited by the usual limitations of weak heredity inherent in most extant interaction screening methods. Our final method (GenCorr) utilizes the generalized correlation matrix to feature screen data sets with multivariate response. GenCorr is the only multivariate screening method in existence which allows for feature screening both marginal, as well as interactive, effects. Each of these three methods is shown to have impressive finite sample performance via a series of simulations and real data analyses. All methods are also established as having the strong sure screening property.

(189 pages)

PUBLIC ABSTRACT

Feature Screening of Ultrahigh Dimensional Feature Spaces with Applications in Interaction Screening

Randall D. Reese

Data for which the number of predictors exponentially exceeds the number of observations is becoming increasingly prevalent in fields such as bioinformatics, medical imaging, computer vision, and social network analysis. One of the leading questions statisticians must answer when confronted with such "big data" is how to reduce a set of exponentially many predictors down to a set of a mere few predictors which have a truly causative effect on the response being modelled. This process is often referred to as *feature screening*. In this work we propose three new methods for feature screening. The first method we propose (TC-SIS) is specifically intended for use with data having both categorical response and predictors. The second method we propose (JCIS) is meant for feature screening for interactions between predictors. JCIS is rare among interaction screening methods in that it does not require first finding a set of causative main effects before screening for interactive effects. Our final method (GenCorr) is intended for use with data having a multivariate response. GenCorr is the only method for multivariate screening which can screen for both causative main effects and causative interactions. Each of these aforementioned methods will be shown to possess both theoretical robustness as well as empirical agility.

CONTENTS

Page

ABSTRACT . . . iii

PUBLIC ABSTRACT . . . iv

LIST OF TABLES . . . vii

LIST OF FIGURES . . . ix

1 Overview . . . 1
- 1.1 Generalities . . . 1
- 1.2 Categorical Feature Screening . . . 2
- 1.3 Interaction Feature Screening . . . 3
- 1.4 Feature Screening with Multivariate Response . . . 3
- 1.5 Concluding Remarks of Chapter 5 . . . 4
- 1.6 Miscellanea . . . 4

2 Screening Categorical Response and Predictors . . . 5
- 2.1 Introduction . . . 5
- 2.2 Strong Independence Screening Using Trend Correlation . . . 8
  - 2.2.1 Some Preliminaries . . . 8
  - 2.2.2 An Independence Ranking and Screening Procedure . . . 9
  - 2.2.3 Theoretical Properties . . . 11
- 2.3 Numerical Studies . . . 13
  - 2.3.1 Example 1 . . . 14
  - 2.3.2 Example 2 . . . 15
  - 2.3.3 Example 3 . . . 17
  - 2.3.4 Example 4 . . . 19
  - 2.3.5 Example 5 . . . 22
- 2.4 Discussion . . . 24
- 2.5 Appendices . . . 25
  - 2.5.1 Some Prefacing Lemmas . . . 25
  - 2.5.2 Proofs of Theorem 2.2.1 and 2.2.2 . . . 31

3 Interaction Screening Using Joint Cumulants . . . 35
- 3.1 Introduction . . . 35
- 3.2 Preliminaries . . . 42
- 3.3 Interaction Screening using Joint Cumulants . . . 43
  - 3.3.1 A Newly Proposed Interaction Feature Screening Method . . . 43
  - 3.3.2 Theoretical Background . . . 43
  - 3.3.3 Theoretical properties . . . 45
  - 3.3.4 Corollaries . . . 47

3.4 Simulations and Empirical Data Analysis . . . 47
3.4.1 Simulation 1 . . . 48
3.4.2 Simulation 2 . . . 48
3.4.3 Simulation 3 . . . 51
3.4.4 Simulation 4 . . . 52
3.4.5 Final Comments on Simulation Results . . . 53
3.4.6 Real Data Analysis: Epistasis Detection . . . 53
3.5 Concluding Remarks . . . 59
3.6 Proofs of Theoretical Results . . . 60
3.6.1 Prefacing Lemmas and a Definition . . . 60
3.6.2 Proofs of Theorems 3.3.1 and 3.3.2 . . . 62

4 Marginal and Interactive Feature Screening for Ultrahigh Dimensional Data with Multivariate Response . . . 69
4.1 Introduction . . . 69
4.2 Feature Screening and Generalized Correlation . . . 71
4.2.1 Marginal Feature Screening Via Generalized Correlation . . . 71
4.2.2 Extensions to Interaction Feature Screening . . . 74
4.2.3 Theoretical properties . . . 77
4.3 Simulations and Empirical Data Analysis . . . 79
4.3.1 Simulation 1 . . . 80
4.3.2 Simulation 2 . . . 85
4.3.3 Simulation 3 . . . 87
4.3.4 Simulation 4 . . . 91
4.3.5 Simulation 5 . . . 93
4.3.6 Conclusion on Simulations . . . 100
4.3.7 Real Data Analysis . . . 100
4.4 Discussion . . . 106
4.5 Proofs of Theorems 4.2.1 and 4.2.2 . . . 106
4.5.1 Prefacing Lemmas . . . 107
4.5.2 Proofs of Main Results . . . 113

5 Final Comments and Discussion . . . 118
5.1 Summary . . . 118
5.2 Future Work . . . 119

REFERENCES . . . 121

APPENDICES . . . 128
A R Code . . . 129
A.1 R Code for Chapter 2 . . . 129
A.2 R Code for Chapter 3 . . . 135
A.3 R Code for Chapter 4 . . . 144
B Using the Center for High Performance Computing . . . 176
B.1 SLURM Submission Examples . . . 177

CURRICULUM VITAE . . . 180

LIST OF TABLES

Table Page

2.1 Values of π_{mj} for Example 1 . . . 15

2.2 Example 1: Proportion of Replications Where X_j is in the Top d Causative Covariates . . . 16

2.3 Cutoff Values . . . 17

2.4 Example 2: Proportion of Replications Where X_j is in the Top d Causative Covariates . . . 18

2.5 Coefficients for L . . . 19

2.6 Example 3: Proportion of Replications Where X_j is in the Top d Causative Covariates . . . 20

2.7 First Ten Components of $\boldsymbol{\beta}$ in Example 4 . . . 21

2.8 Example 4: Proportion of Replications Where X_j is in the Top d Causative Covariates . . . 21

2.9 Mean Minimum Model Sizes for Examples 1 through 4 . . . 22

2.10 Empirical results of real data analysis. . . . 24

3.1 Mean and Median Ranking of Interaction Between X_1 and X_2 in Simulation 1 48

3.2 θ_{kj} Values for Simulation 2 . . . 49

3.3 Mean and Median Ranking of Causative Interactions in Simulation 2 . . . 50

3.4 Percentage of Replicates Finding (X_{j_1}, X_{j_2}) to be Important in Simulation 3 . 51

3.5 Percentage of Replicates Finding (X_{j_1}, X_{j_2}) to be Important in Simulation 4 . 52

3.6 MDR Accuracy . . . 57

4.1 Simulation 1 Mean Ranks . . . 84

4.2 Simulation 1 Median Ranks . . . 85

4.3 Simulation 2 Mean Ranks . . . 86

4.4 Simulation 2 Median Ranks 86

4.5 Simulation 3 Mean Ranks 90

4.6 Simulation 3 Median Ranks 91

4.7 Simulation 4 Mean Ranks 92

4.8 Simulation 4 Median Ranks 92

4.9 Simulation 5 Selection Proportions 99

4.10 Simulation 5 Quantiles 99

4.11 Observed Phenotypic Traits in NMRI Mice 101

4.12 Results of Real Data Analysis 105

LIST OF FIGURES

Figure Page

3.1 SNP$_{j_1}$ position versus $\widehat{R}_{j_1,j_2}$ value. 55

3.2 High-low risk bar plots broken down by genotype for the selected two-locus model. 57

3.3 High-low risk bar plots broken down by genotype for the selected three-locus model. Here "S—" indicates the selected SNP ending in the given three digits. 58

CHAPTER 1

Overview

1.1 Generalities

A standard practice in statistics is the prediction of an outcome based on a set of observed predictors. In classical regression and classification settings, one must operate on the explicit assumption of the number of predictors under consideration being less than the number of observations. Throughout the remainder of this work, we will use p to represent the number of predictors and n to represent the number of observations. Data sets for which $p < n$ will be referred to as *low dimensional* data sets. While low dimensional data sets are commonly used for purposes of pedagogy and allow for the implementation of traditional methods, areas such as bioinformatics, medical imaging, signal processing, and social network analysis (to name a mere few) present statisticians and researchers with data sets where low dimensionality is a rarity. When p exceeds n, many established methods for statistical modelling become unstable and break down. (In the simple case of ordinary least squares regression, for example, we are confronted with the issue of singular matrices when constructing the hat matrix). As such, there has arisen a demand for statistical methods that can confront the mathematical and computational issues inherent in the analysis of feature spaces for which $p > n$.

Of particular interest to us in this work will be data sets where p is significantly larger than n (i.e. $p \gg n$). In order to further facilitate our discussion of data sets having many more predictors than observations, we introduce two relevant terms [24]. When there is some constant $\xi > 0$ such that $p = \mathcal{O}(n^{\xi})$, then we will refer to such data as being *high dimensional*. (The notation $\mathcal{O}(\cdot)$ refers to Bachmann-Landau big-O notation). Similarly, when there is some constant $\xi > 0$ such that $\log(p) = \mathcal{O}(n^{\xi})$, then we will refer to such data as being *ultrahigh dimensional*. All methods presented herein are designed for use with

ultrahigh dimensional data. We will present real data analyses for each method using data exhibiting ultrahigh dimensionality, as well as multiple empirical simulations employing high and ultrahigh dimensional data.

Based on the principle of sparsity, nearly all extant methods for handling high and ultrahigh dimensional data attempt to determine a small (relative to n) set of predictors which have a truly causative effect on the outcome or response. [See 77, for an excellent overview of the topic of sparsity]. Many early approaches to scaling the feature space were based on the concept of penalized regression. This includes such well known methods as lasso [76], adaptive lasso [95], elastic net [96], and smoothly clipped absolute deviation (SCAD) [22]. While applicable to high dimensional feature spaces, penalized regression approaches become intractable when confronted with ultrahigh dimensional data. In their monumental paper, Fan and Lv [24] present the foundational framework to ultrahigh dimensional feature screening. Uniquely applicable to scaling ultrahigh dimensional feature spaces, the concepts of feature screening have become the prevailing approach for almost all contemporary dimension reduction techniques.

In this work we consider three main topics for analysing ultrahigh dimensional data. Each of these methods is an extension in some way of the original philosophies of feature screening initially presented in [24]. We first will consider a new method for feature screening data with both categorical predictors and categorical response. This is followed by an examination of feature screening in the context of interaction screening. Our concluding topic will be that of feature screening when the response is multivariate. Literature pertinent to these topics severally will be considered within the individual chapters.

1.2 Categorical Feature Screening

With recent advances in technology, ultrahigh dimensional data sets in fields like bioinformatics, genetics, and social network analysis are growing exponentially both in terms of size as well as in regards to prevalence. Frequently, these data sets have both categorical response and categorical covariates, yet extant feature screening literature rarely considers such data types. In Chapter 2, we propose a new screening procedure rooted in the Cochran-

Armitage trend test. Our method is specifically applicable for data where both the response and predictors are categorical. We will also establish that the strong sure screening property holds for this method, a valuable metric for measuring the effectiveness of current feature screening approaches. The empirical viability of our categorical screening procedure will be borne out via a series of simulations and a real data analysis. This chapter is derived from a forthcoming paper co-written with Xiaotian Dai and Guifang Fu.

1.3 Interaction Feature Screening

While a multiplicity of methods exist for screening ultrahigh dimensional data for marginal effects, much fewer methods exist for performing interaction screening in such a context. Moreover, among the extant interaction screening methods, almost no consideration is given to models for which the marginal effects are weak, but the interactive effects are strong. Chapter 3 will seek to ameliorate these issues by presenting a novel interaction feature screening approach that is not dependent on first screening for marginal effects. Once again, we will demonstrate the theoretical stability of our method in regards to strong sure screening, as well as show the numerical feasibility of our method in terms of finite sample performance. This chapter is derived from a forthcoming paper co-written with Xiaotian Dai and Guifang Fu.

1.4 Feature Screening with Multivariate Response

In Chapter 4 we will introduce a new method for feature screening ultrahigh dimensional data when the response variable is multivariate. By allowing for the response to be multivariate, our approach now permits us to examine the complex traits of a multifaceted response that may only be evident when considered as a collective whole and not merely as individual pieces. This newly proposed method for feature screening multivariate response data will allow for both marginal effects screening as well as interaction screening. We will establish the strong sure screening property for this approach. A collection of simulations, along with a real data analysis, will demonstrate the finite sample performance of the newly proposed method.

1.5 Concluding Remarks of Chapter 5

The main body of this work is closed by concluding comments given in Chapter 5. Therein we will review what the previous chapters have demonstrated as well as suggest areas for future work.

1.6 Miscellanea

All simulations and real data analyses were carried out in the R programming language. Relevant code for each associated chapter is presented in Appendix A. Many of the simulations as well as both of the real data analyses in Chapters 3 and 4 were done using the cluster computing capabilities at the Center for High Performance Computing (CHPC) centered at The University of Utah. This work would not have been possible without their assistance. Willis Barton [6] was instrumental in the initial process of accessing the CHPC framework. Appendix B demonstrates a short example of how to use the CHPC resources. I would also like to expressly thank my committee (Richard Cutler, Rose Hu, John Stevens, and Jia Zhao) for their willingness to meet with me as needed and for their input to this work and my graduate experience. The curriculum vitae found at the end of this work contains contact information whereat I can be reached for the foreseeable future. This dissertation was typeset using LaTeX.

CHAPTER 2

Screening Categorical Response and Predictors

2.1 Introduction

Ultrahigh dimensional data with dichotomous response and polytomous features has become increasingly prevalent in various fields. For example, applications using such data exist in genome-wide association studies (GWAS), medical imaging, finance, text mining, and classifying text documents by keywords, among others [35, 43]. Many existing approaches are feasible for classification or analyses involving polytomous features and dichotomous response with high dimension, including random forests [10, 54], k-nearest neighbors [39], and support vector machines [46, 78]. However, these methods may lose power or become increasingly unstable, and hence intractable, as the dimension of feature space becomes ultrahigh. Therefore, there has arisen an accompanying need to develop statistical methods satisfying computational expediency, statistical accuracy, and algorithmic stability for the rapidly growing assortment of ultrahigh dimensional data.

Given n samples for each of p predictors in question, the term “high dimensional" will be used to mean $p = \mathcal{O}(n^{\xi})$ for some $\xi > 0$, similarly we will use the term “ultrahigh dimensional" to mean $\log(p) = \mathcal{O}(n^{\xi})$ for some $\xi > 0$. [23] and [21] reviewed the challenges encountered in ultrahigh dimensional big data, in which the fundamental challenge comes from the accumulation of aggregate error rates due to a preponderance of noise features. For example, as discussed in [19], when using a discriminant analysis rule such as LDA or QDA, the population mean vectors are estimated from the observed sample. When the dimensionality is ultrahigh, although each individual component of the population mean vectors can be estimated with sufficient accuracy, the overall misclassification rate can become substantial due to the aggregated estimation errors.

One approach that has garnered considerable attention in recent ultrahigh dimensional

modeling literature is that of *feature screening*. Feature screening filters out a substantial amount of noise features and keeps a small set of influential variables which represent the majority of the extractable information in the data and accommodate the sparsity principle as it relates to ultrahigh dimensional data. In their seminal paper, Fan and Lv [24] established the underpinnings for what they termed sure independent feature screening (SIS) and introduced the conceptual framework of the bulk of feature screening literature that would come thereafter. However, they assumed not only a strict linear model but also normally distributed continuous predictors and response.

Almost all approaches stemming from [24] were constructed under the premise of both the predictors and the response being continuous. See e.g. [4, 18, 20, 93]. Even though these aforementioned methods at times relaxed the model specification assumptions [see an overview in 55], most SIS-based procedures still tacitly assume that the predictor variables and the response are continuous. Notably, this implicit presupposition of continuity of the predictors can be limiting in certain instances in that the given method has not been specifically calibrated in an empirical setting for categorical predictors.

Herein we will further explore three existing methods that admit both categorical response and predictors, as well as propose a new approach to screening categorical predictors when the feature space is ultrahigh dimensional. First among the extant methods is the maximum marginal likelihood estimator based approach (MMLE-SIS) of [26]. In the case of a binary response Y, they consider the maximum marginal likelihood of each covariate individually by examining the magnitude of the marginal regression coefficient, β_j^M, for each covariate, X_j, given by

$$\beta_j^M = \arg\min_{\beta_j} \mathbb{E}\left(\ell(\beta_j X_j, Y)\right),$$

where $\ell(\cdot)$ represents the traditional log-likelihood function. One essential limitation of MMLE-SIS in practice is that while it allows for categorical predictors, it has never been specifically verified to produce reliable numeric results under this setting.

Another current method admitting categorical data is the distance correlation based screening method (DC-SIS) presented in [53]. Using the concept of distance correlation

first presented in [75], DC-SIS measures the association between a covariate, X_j, and the response, Y. As was the case with MMLE-SIS, although DC-SIS allows for categorical response and predictors, the method has never been explicitly examined numerically for use with categorical data.

The final method for feature screening categorical data that we will consider is the Pearson's chi-squared test based screening method (PC-SIS) found in [43]. This method determines the association between the response and each predictor individually by treating the levels of the response as forming the columns of a two-way table and the levels of a given predictor as forming the rows of said table. A fundamental question they consider is that of classifying Internet advertisements based on the presence or absence of given keywords. A large collection of keywords acts as the set of predictors, with each predictor being binary (0 for absence of the keyword, 1 for presence). The response takes on K many levels, with each level representing a category of Internet advertisement. From this, they apply the Pearson chi-squared test to the associated $2 \times K$ table and use the resulting chi-squared test statistic as a marginal utility to screen the feature space. (Alternatively, when each predictor is allowed to differ from other predictors in its number of levels, they obtain the associated p-values from the chi-squared test and rank features based on decreasing p-value).

Motivated by data with a dichotomous response and ultrahigh dimensional polytomous features, we propose a new feature screening method based on the Cochran-Armitage trend test or trend correlation (we call our method TC-SIS for short). We compare TC-SIS with the aforementioned three most relevant approaches (MMLE-SIS, DC-SIS, PC-SIS) and demonstrate that TC-SIS outperforms these in terms of accuracy and speed for the motivated scenarios through finite sample simulation studies. Additionally, we prove that TC-SIS satisfies the strong screening consistency property under a set of reasonable conditions, which is a much stricter criterion than the sure screening property achieved by many other feature screening approaches. We demonstrate that the finite sample performance of TC-SIS is quite promising under various simulation studies as well as present a motivating real data analysis using TC-SIS.

The rest of this article is organized as follows. In Section 2.2 we describe our proposed TC-SIS method for feature screening in detail and establish its strong screening consistency property. In Section 2.3 we examine the finite sample performance of TC-SIS via multiple Monte Carlo simulations and a real data analysis example. We finally conclude with a brief discussion in Section 2.4. The Appendices (Section 2.5) are devoted to all technical proofs.

2.2 Strong Independence Screening Using Trend Correlation

2.2.1 Some Preliminaries

[16] and [3] advocate using the trend test for measuring association between two categorical variables. Specifically, let $u_1 \leq u_2 \leq \ldots \leq u_I$ be the numeric score assigned to each level of the rows and $v_1 \leq v_2 \leq \ldots \leq v_J$ be the numeric score assigned to each level of the columns, where I and J are the number of levels/categories of each row and column, respectively. For a sequence of n samples, let the sample mean score of the row denoted by $\bar{u}$ and sample mean score of the column denoted as $\bar{v}$. The trend correlation between the row and column variables can be defined as follows [1]:

$$\hat{\varrho} = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}(u_i - \bar{u})(v_j - \bar{v})\hat{p}_{ij}}{\sqrt{\left(\sum_{i=1}^{I}(u_i - \bar{u})^2\hat{p}_{i+}\right)\left(\sum_{j=1}^{J}(v_j - \bar{v})^2\hat{p}_{+j}\right)}},$$

where $\hat{p}_{i+}$, $\hat{p}_{+j}$, and $\hat{p}_{ij}$ are the respective frequency proportions of each row, column, and individual cell of the contingency table.

We are motivated to utilize $\hat{\varrho}$ in a feature screening procedure because $\hat{\varrho}$ is a categorical version of the Pearson correlation coefficient [1] and possesses the rather salient property of being equal to zero if the two involved variables are independent.

2.2.2 An Independence Ranking and Screening Procedure

Here we propose an independence screening procedure based on the following modification of the correlation between two categorical variables. Let Y be the response variable and let X_j (with $j = 1, \ldots, p$) be the j^{th} predictor. Specifically, let $v_k^{(j)}$ be the numeric score assigned to each level k of each polytomous predictor X_j, where $k = 1, \ldots, K_j$. Here p is the total number of predictors and K_j is the number of levels for X_j, for which we allow for a various number of levels for different predictors. Let $m = 0, 1$ be the encoding of the dichotomous response Y. The value ϱ_j can then be simply defined as

$$\varrho_j = \frac{\left| \sum\limits_{\substack{1 \le k \le K_j \\ 0 \le m \le 1}} (v_k^{(j)} - \mathbb{E}X_j)(m - \mathbb{E}Y) p_{km}^{(j)} \right|}{\sqrt{\left(\sum\limits_{k=1}^{K_j} (v_k^{(j)} - \mathbb{E}X_j)^2 p_k^{(j)} \right) \left(\sum\limits_{m=0}^{1} (m - \mathbb{E}Y)^2 p_m \right)}},$$

where

$$p_{km}^{(j)} = \mathbb{P}(X_j = k, Y = m), \quad p_k^{(j)} = \mathbb{P}(X_j = k), \quad p_m = \mathbb{P}(Y = m).$$

For a sequence of n samples, we will denote the sample mean score of X_j by $\bar{v}^{(j)}$ and sample mean of Y as $\bar{Y}$. We can then define the trend correlation between X_j and Y as in [1]

$$\hat{\varrho}_j = \frac{\left| \sum\limits_{\substack{1 \le k \le K_j \\ 0 \le m \le 1}} (v_k^{(j)} - \bar{v}^{(j)})(m - \bar{Y}) \hat{p}_{km}^{(j)} \right|}{\sqrt{\left(\sum\limits_{k=1}^{K_j} (v_k^{(j)} - \bar{v}^{(j)})^2 \hat{p}_k^{(j)} \right) \left(\sum\limits_{m=0}^{1} (m - \bar{Y})^2 \hat{p}_m \right)}},$$

where $\hat{p}_{km}^{(j)}$, $\hat{p}_k^{(j)}$, and $\hat{p}_m$ represent the sample estimates (by the relevant sample proportion) of the associated probabilities $p_{km}^{(j)}$, $p_k^{(j)}$, and p_m. Note that the denominator of $\hat{\varrho}_j$ consists of (biased) sample estimators for the standard deviations of X_j and Y. (However, the bias of these estimators disappears asymptotically). Both of these estimators are consistent estimators of their respective standard deviations. Consistency is easy to prove using Chebychev's

inequality and routine algebra. For completeness, this will be shown in Section 2.5.

When the predictors are ordinal, we can interpret $\hat{\varrho}_j$ as estimating the linear *trend* between X_j and Y, e.g., an increase in the observed level of X_j tends to be associated with decreasing or increasing levels of Y. Taking ordinality into account is desirable and will improve the interpretive power of our conclusions [1]. Therefore, it is suggested that the ordering of and the distance between the $v_k^{(j)}$ scores conform to those of the categorical levels. This possibility to examine trend between the response and covariates is, however, only one example of a robust number of settings that our method is capable of handling.

The sparsity property of ultrahigh dimensional feature spaces informs us that only a small number of the predictors are relevant to the response. Define $\mathcal{S}_F = \{1, 2, \ldots, p\}$ as the set of indices of the full model. This set contains the index of every predictor in the feature space. Let $\mathcal{S} \subseteq \{1, 2, 3, \ldots, p\}$ denote the set of indices of an arbitrary model under consideration, which is a subset of all possible predictors. Let $X_{(\mathcal{S})}$ be the corresponding model or set of predictors whose indicies are contained in $\mathcal{S}$.

We use $\hat{\varrho}_j$ as a marginal utility to rank the importance of each X_j according to its associations with the response, where higher $\hat{\varrho}_j$ values correspond to stronger association. Note that $\hat{\varrho}_j$ is non-negative because the absolute values are used in the numerator. As a result of the TC-SIS feature screening procedure, the estimated indices of the model are given by

$$\widehat{\mathcal{S}} = \{j : \hat{\varrho}_j > c, for \;\; 1 \leq j \leq p\},$$

where c is a pre-specified threshold value.

Let $\mathcal{D}\left(Y \mid X_{(\mathcal{S})}\right)$ denote the conditional distribution of Y given $X_{(\mathcal{S})}$. We will consider a model $\mathcal{S}$ to be sufficient if

$$\mathcal{D}\left(Y \mid X_{(\mathcal{S}_F)}\right) = \mathcal{D}\left(Y \mid X_{(\mathcal{S})}\right).$$

The full model $\mathcal{S}_F$ is trivially sufficient. We are ultimately only interested in finding the smallest (cardinality-wise) sufficient model. We will call the smallest sufficient model the

true model. Our aim in feature screening is to determine an estimated model which contains the true model and is moreover the *smallest* such model to contain the true model. The next subsection will outline the specifics of our proposed screening approach for estimating the true model. As a matter of further notation, we will denote the true model by $\mathcal{S}_T$ and the estimated model output from TC-SIS by $\widehat{\mathcal{S}}$.

2.2.3 Theoretical Properties

In this section the theoretical properties of the proposed independence screening procedure TC-SIS will be studied. We first define two weak conditions to facilitate the technical proofs:

(C1) *Bounds on the standard deviations.* Assume that there exists a positive constant $\sigma_{\min}$ such that for all j,

$$\min(\sigma_j, \sigma_Y) \geq \sigma_{min}$$

This excludes unusual or unreasonable features that are constant and hence have a standard deviation of zero. It should further be noted that an upper bound on σ_j and σ_Y can also be obtained, by use of Popoviciu's inequality on variances [see 67]:

$$\max(\sigma_j, \sigma_Y) \leq \sigma_{max}, \quad \sigma_{\max} = \max\left\{\frac{1}{2}, \sqrt{\frac{1}{4}\left(\max(v_k^{(j)}) - \min(v_k^{(j)})\right)}\right\}$$

where the first term in the maximum selection is a bound on the standard deviation of Y and the second term is given by Popoviciu's inequality on variances.

(C2) *Marginal Components.* Assume that $\varrho_j = 0$ for any $j \notin \mathcal{S}_T$. Define

$$\omega_{km}^{(j)} = \left|(v_k^{(j)} - \mathbb{E}(X_j))(m - \mathbb{E}(Y))p_{km}^{(j)}\right|,$$

and assume that $\omega_{km}^{(j)}$ is of the same sign for all levels k and m. Without loss of generality it can be assumed that $\omega_{km}^{(j)} \geq 0$. Assume also that there exists a positive

constant $\omega_{\min}$ such that

$$\min_{j \in \mathcal{S}_T} \left(\max_{\substack{1 \le k \le K_j \\ 0 \le m \le 1}} \left\{ \omega_{km}^{(j)} \right\} \right) > \omega_{\min} > 0$$

This places a lower bound on the minimum value (indexing by j) of the maximum values of the $\omega_{km}^{(j)}$ (indexing by k and m). Note that (C2) requires that for each truly influential feature (i.e. $j \in \mathcal{S}_T$), there exists at least one level of the response Y and one level of the feature X_j that are marginally correlated (i.e. $\omega_{km}^{(j)} > \omega_{\min}$). This is a natural assumption to make for the truly influential features and should be quite easy to satisfy in a wide variety of reasonable situations.

When these two conditions are satisfied, we can establish the following theorems that support the *strong screening property* for the TC-SIS procedure.

Theorem 2.2.1. *(Sure Screening Property). Under condition (C1) and removing from (C2) only the assumption that $\varrho_j = 0$ for any $j \notin \mathcal{S}_T$, there exists a positive constant $c > 0$ such that*

$$\mathbb{P}(\mathcal{S}_T \subseteq \widehat{\mathcal{S}}) \to 1 \text{ as } n \to \infty.$$

(However, $\mathbb{P}(\widehat{\mathcal{S}} \subseteq \mathcal{S}_T)$ may not converge to one as n approaches infinity).

Theorem 2.2.2. *(Strong Screening Consistency). Given conditions (C1) and (C2), there exists a positive constant $c > 0$ such that*

$$\mathbb{P}(\widehat{\mathcal{S}} = \mathcal{S}_T) \to 1 \text{ as } n \to \infty.$$

The property of strong screening consistency is much harder to achieve than the (weak) sure screening property that the majority of feature screening approaches achieve because it not only guarantees that the true model is contained in the selected subset, but also ensures that the selected subset equals the true model asymptotically. The proofs of these two theorems are presented in Section 2.5.

In addition to the aforementioned two theorems, we also draw two corollaries from the proofs of Theorems 2.2.1 and 2.2.2. These results are not themselves about sure screening, but they are nevertheless important conclusions on the underlying mechanics of our method.

Corollary 2.2.3. *There exists a value* $\varrho_{\min}$ *such that for any* $j \in \mathcal{S}_T$, *we have* $\varrho_j > \varrho_{\min}$.

This will be shown in Step 1 of the proofs of Theorems 2.2.1 and 2.2.2.

Corollary 2.2.4. *The estimator* $\hat{\varrho}_j$ *converges uniformly in probability to* ϱ_j. *In other words,*

$$\mathbb{P}\left(\max_{1\leq j\leq p} |\hat{\varrho}_j - \varrho_j| > \varepsilon\right) \to 0 \qquad as\ n \to \infty$$

for any $\varepsilon > 0$.

This will be shown from the end of Step 2 in the proofs of Theorems 2.2.1 and 2.2.2.

2.3 Numerical Studies

In this section we assess the performance of TC-SIS by four empirical Monte Carlo simulation studies under different designs. Additionally, we also perform an analysis on an empirical data set examining polycystic ovary syndrome (PCOS) to further illustrate the proposed screening procedure.

For each of the four simulations, we fix the sample size, n, to be 200 and set the number of predictors, p, to be 5,000 (Example 1-3) or 1,000 (Example 4). We replicate each simulation 500 times, and assess the performance of the screening procedures through the following two criteria given by [53]:

- Mean Minimum Model Size: The average of the minimum number of predictors that are required by the given screening procedure to acquire all truly influential predictors across all simulation replicates. The closer to the true model size $|S_T|$ the estimated model size $|\widehat{S}|$ is, the better the screening procedure is determined to be.

- Individual Acquisition Proportions: The proportion of all replicates in which a given individual predictor in $\mathcal{S}_T$ is correctly obtained by the screening method in question under the model sizes $d = 10, 15, 20$. This requires that the screening score of each truly influential individual predictor should rank within the 10 highest, the 15 highest, and the 20 highest among all p predictors. The closer to one that this proportion is, the better the screening procedure is determined to be.

We will use the same simulated data and same assessment criteria to compare in detail the finite sample performance of our proposed TC-SIS approach with three other relevant methods: maximum marginal likelihood estimator based screening (MMLE-SIS) [26]; distance correlation based screening (DC-SIS) [53]; and Pearson's Chi-squared test based screening (PC-SIS) [43]. We will in some cases borrow elements of simulations previously demonstrated in these aforementioned publications.

2.3.1 Example 1

For this example, each sample, Y_i, of the response will be generated by a Bernoulli process with $\mathbb{P}(Y = 1) = p_y$, where $p_y \sim \text{unif}(0.05, 0.95)$ is chosen anew for each replicate of the simulation. We design the first ten predictors to be truly influential to the response Y, i.e., $S_T = \{1, \ldots, 10\}$. Inspired by Example 1 of [43], we generate these first ten predictors as

$$\{X_{ij} \mid Y_i = m\} \sim \text{Binomial}(2, \pi_{mj})$$

with the values of π_{mj} being given by Table 2.1. This means that each causative X_j will take on values of 0, 1, or 2 (representative of three ordinal levels, with $0 \prec 1 \prec 2$). For any $j \notin S_T$, we define for each j individually the value $\pi_j \sim \text{Unif}(0.05, 0.95)$ and let $X_j \sim \text{Binomial}(2, \pi_j)$. The value of π_j is chosen anew with each replicate of the simulation. This means that these non-causative predictors will have no association with Y.

The results of Example 1 are summarized in Tables 2.2 (acquisition proportions) and 2.9 (mean minimum model sizes). For this simulation, TC-SIS results in the smallest aver-

Table 2.1: Values of π_{mj} for Example 1

	π_{m1}	π_{m2}	π_{m3}	π_{m4}	π_{m5}	π_{m6}	π_{m7}	π_{m8}	π_{m9}	$\pi_{m,10}$
$Y = 0$	0.3	0.4	0.6	0.7	0.2	0.4	0.3	0.8	0.4	0.2
$Y = 1$	0.6	0.1	0.1	0.4	0.8	0.7	0.9	0.2	0.7	0.6

age model size (54.674) required to contain the true model, an entire ten features less than the next closest method (DC-SIS obtains a mean minimum model size of 64.990). PC-SIS and MMLE-SIS required, respectively, mean minimum model sizes nearly double or triple that required by TC-SIS. These results establish the capability of TC-SIS to obtain excellent results when the predictors come from the binomial distribution. This is significant for applications in genetics as it models the encoding of genotypes for homozygous recessive ($X_j = 0$), heterozygous ($X_j = 1$), and homozygous dominant ($X_j = 2$) alleles, with the distribution of these genotypes being determined by the binomial distribution. Another interesting consideration to make is the relative stability of our method in the face of a potentially large unbalance in the number of positive ($Y = 1$) responses observed in the sample data. It is also important to note that for each model size thresholds ($d = 10, 15$ and 20), TC-SIS obtains the largest proportion of replicates for which each predictor individually is found in the top d-many predictors. In the case of TC-SIS versus MMLE-SIS, the proportions are at times nearly four fold more favorable towards our method.

2.3.2 Example 2

Inspired by the concept of discretization of a continuous random variable as found in Example 3 of [43], we connect the influential predictors with the response via an indirect way. Similar to Example 1 above, we generate the response Y_i from a Bernoulli process with $\mathbb{P}(Y_i = 1) = p_y$, where $p_y \sim \text{unif}(0.05, 0.95)$ is again chosen anew for each replicate of the simulation. Given $Y_i = m$, we generate a latent variable Z_{ij} independently distributed as $N(Y_i, 1)$ for the first ten truly influential predictors $j \in S_T$. The first ten influential predictors X_{ij} are then discretized from Z_{ij} based on the cutoffs $(\kappa_{Lj}, \kappa_{Uj})$ listed in Table

Table 2.2: Example 1: Proportion of Replications Where X_j is in the Top d Causative Covariates

$d = 10$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.864	0.980	0.988	0.832	1.000	0.998	0.844	0.974	0.836	0.858
MMLE-SIS	0.210	0.680	0.690	0.250	0.786	0.816	0.194	0.646	0.182	0.218
DC-SIS	0.850	0.978	0.982	0.818	0.998	0.998	0.804	0.968	0.824	0.834
PC-SIS	0.808	0.956	0.962	0.800	0.992	0.996	0.778	0.954	0.760	0.802
$d = 15$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.916	0.988	0.994	0.922	1.000	0.998	0.912	0.982	0.904	0.908
MMLE-SIS	0.384	0.746	0.756	0.404	0.822	0.844	0.354	0.742	0.320	0.400
DC-SIS	0.900	0.984	0.990	0.898	0.998	0.998	0.894	0.984	0.888	0.894
PC-SIS	0.862	0.974	0.980	0.864	0.994	0.998	0.860	0.966	0.854	0.862
$d = 20$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.920	0.988	0.994	0.930	1.000	0.998	0.926	0.990	0.930	0.930
MMLE-SIS	0.508	0.796	0.800	0.490	0.854	0.864	0.452	0.796	0.440	0.490
DC-SIS	0.916	0.988	0.992	0.920	0.998	0.998	0.922	0.986	0.908	0.914
PC-SIS	0.890	0.980	0.986	0.886	0.994	0.998	0.872	0.972	0.886	0.882

2.3 as:

$$X_{ij} = \begin{cases} 0 & \text{if } Z_{ij} < \kappa_{Lj}, \\ 1 & \text{if } \kappa_{Lj} \leq Z_{ij} \leq \kappa_{Uj}, \\ 2 & \text{if } \kappa_{Uj} < Z_{ij}. \end{cases}$$

These cutoffs in Table 2.3 are set to establish weaker correlations between the response Y and each of the influential predictor to increase the difficulty of determining the true model. However, the values used in this example are heuristic and can be flexibly changed. It should be noted that this method of generating the causative predictors results in an overall trend association between these first ten covariates (individually) and Y: namely, lower values of X_j are associated with $Y = 0$ and higher values of X_j are associated with $Y = 1$. Moreover, although these causative predictors are constructed to have positive linear correlation with

Y, covariates constructed to have negative correlation with Y yield identical results. This is due to the fact that we only care about the magnitude of $\hat{\varrho}_j$. For any $j \notin S_T$, we generated corresponding non-influential predictors using the same design as was used in Example 1.

Table 2.3: Cutoff Values

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
κ_{Lj}	0	0	0.2	0	-0.213	0.25	0	0.1	-0.2	0.213
κ_{Uj}	0.75	1	0.8	0.9	1.213	1	1	1	1.2	0.787

The results of Example 2 are summarized in Tables 2.4 (acquisition proportions) and 2.9 (mean minimum model sizes). As was the case with Example 1, our method here results in the smallest average model size required to contain the true model (mean minimum model size of 112.627). Since this simulation was designed to specifically take advantage of the forte of our method in testing for linear trend, these results should not be surprising. Because PC-SIS lacks the specific ability to employ the more substantive statistical power that is inherent in a trend based method like TC-SIS, the larger mean minimum model size obtained by PC-SIS is to be expected. Of especial note here, MMLE-SIS fails on average to produce an estimated model smaller than the sample size of $n = 200$. TC-SIS and DC-SIS are comparable in their acquisition of the true model for the usual threshold values of $d = 10, 15$, and 20. This leads us to the overall conclusion (based on the results for mean minimum model size) that TC-SIS does a better job than DC-SIS at avoiding ballooning models in the worst case outcomes.

2.3.3 Example 3

In this simulation we generate the sample data based on an explicit logistic regression model, a specific strength of MMLE-SIS. First generate each predictor X_j $(1 \leq j \leq p)$ by uniformly sampling the set $\{0, 1, 2\}$ with equal probability. We then connect a binary

response Y with the first five predictors by letting

$$L_i = \sum_{j=1}^{5}[I(X_{ij} = 0) \times \theta_{X_j=0} + I(X_{ij} = 1) \times \theta_{X_j=1} + I(X_{ij} = 2) \times \theta_{X_j=2}],$$

and setting

$$P(Y_i = 1) = \frac{1}{1+\exp(-L_i)}.$$

The coefficients $\theta_{X_j=k}$ are as given in Table 2.5. The responses are the generated by sampling the Bernoulli distribution with $\mathbb{P}(Y_i = 1)$ as given above.

The results of Example 3 are summarized in Tables 2.6 (acquisition proportions) and 2.9 (mean minimum model sizes). For this example, we once again obtain a smaller required mean model size than DC-SIS and PC-SIS (mean minimum model sizes of 46.470 and 93.270

Table 2.4: Example 2: Proportion of Replications Where X_j is in the Top d Causative Covariates

$d = 10$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.828	0.822	0.816	0.866	0.884	0.820	0.892	0.850	0.796	0.862
MMLE-SIS	0.028	0.026	0.038	0.038	0.424	0.016	0.302	0.054	0.014	0.124
DC-SIS	0.832	0.832	0.830	0.868	0.860	0.824	0.880	0.860	0.818	0.868
PC-SIS	0.754	0.766	0.744	0.774	0.846	0.752	0.844	0.776	0.738	0.808
$d = 15$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.876	0.876	0.882	0.904	0.924	0.874	0.920	0.906	0.878	0.900
MMLE-SIS	0.072	0.060	0.066	0.078	0.554	0.054	0.388	0.098	0.032	0.204
DC-SIS	0.876	0.884	0.886	0.904	0.886	0.880	0.880	0.910	0.878	0.906
PC-SIS	0.820	0.806	0.816	0.842	0.876	0.806	0.876	0.830	0.820	0.858
$d = 20$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.888	0.892	0.898	0.918	0.940	0.902	0.936	0.920	0.898	0.920
MMLE-SIS	0.110	0.098	0.106	0.152	0.612	0.086	0.468	0.166	0.064	0.270
DC-SIS	0.898	0.900	0.908	0.918	0.918	0.904	0.930	0.918	0.896	0.930
PC-SIS	0.844	0.830	0.850	0.862	0.880	0.824	0.888	0.862	0.838	0.876

Table 2.5: Coefficients for L

	θ_{X_1}	θ_{X_2}	θ_{X_3}	θ_{X_4}	θ_{X_5}
$X_j = 0$	0	-5	2	-6	1
$X_j = 1$	3	-3	4	-4	3
$X_j = 2$	5	-1	6	-2	5

for DC-SIS and PC-SIS respectively, as compared to a mean minimum model size of 41.976 for TC-SIS). A specific comment on the results of Example 3 for MMLE-SIS is in order. As was previously discussed, one strength of MMLE-SIS is screening data in a logistic regression setting. Indeed, MMLE-SIS recoups its earlier collapses and matches (by about four hundredths of an average minimum model size) our method nearly perfectly. Similar results are obtained for the proportion of replications that capture the true predictors within model sizes of the $d = 10$, 15, and 20. (Although, note that TC-SIS does capture four of the five causative predictors within the top ten predictors at least as often as MMLE-SIS does, while exceeding it in three cases). Moreover, it should ultimately be noted that since MMLE-SIS requires solving an optimization problem to produce its screening statistics, our newly proposed method is significantly faster in computational run time. Thus, when run time is an issue, we suggest the use of our method over MMLE-SIS, even in a logistic regression setting. As TC-SIS performs admirably under the settings of Example 3, producing results abreast with that of MMLE-SIS, the increased run time for MMLE-SIS is hard to justify.

2.3.4 Example 4

Although our method is not originally designed or emphasized for use on continuous data, this simulation presents a comparison of trend correlation versus DC-SIS when the covariates are multi-normally distributed. The motivation for this simulation is the statement by [53] that when the covariates are normally distributed, DC-SIS is “equivalent" (although not equal, see Theorem 7 of [75]) to the SIS method of [24]. However, [75] and [74] further elucidate the fact that the response must also be normally distributed for DC-SIS to be equivalent to SIS. Our aim here is to see how DC-SIS performs when the covariates are

multi-normally distributed, yet the response is not necessarily normally distributed. We do this in order to demonstrate a proof-of-concept for a general application of trend correlation to feature screening outside the assumptions of a normally distributed response being predicted by pair-wise independent normally distributed predictors.

The data for this simulation is generated as follows. Let $\mathbf{X_i}$ be a vector of length 1000 ($p = 1000$), where $\mathbf{X_i} \sim \text{MVN}(\mathbf{0}, \Sigma)$ is sampled for i from 1 to 200. Here the covariance matrix $\Sigma = [\sigma_{j_1 j_2}]$ is given by $\sigma_{j_1 j_2} = 0.2^{|j_1 - j_2|}$. We now generate the response, Y, using only the first ten predictors. Specifically, we let $Y_i = \mathbf{X_i}\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is a vector whose first ten entries are defined in Table 2.7, and whose remaining entries are all zero. Note that, because the first ten covariates are not independent, this construction of Y does not guarantee that Y itself is normally distributed.

The results of Example 4 are summarized in Tables 2.8 (acquisition proportions) and

Table 2.6: Example 3: Proportion of Replications Where X_j is in the Top d Causative Covariates

$d = 10$					
	X_1	X_2	X_3	X_4	X_5
TC-SIS	1.000	0.834	0.804	0.810	0.816
MMLE-SIS	1.000	0.822	0.798	0.808	0.824
DC-SIS	1.000	0.832	0.808	0.806	0.814
PC-SIS	1.000	0.742	0.708	0.710	0.740
$d = 15$					
	X_1	X_2	X_3	X_4	X_5
TC-SIS	1.000	0.860	0.858	0.842	0.862
MMLE-SIS	1.000	0.856	0.868	0.842	0.870
DC-SIS	1.000	0.860	0.850	0.838	0.866
PC-SIS	1.000	0.794	0.758	0.778	0.790
$d = 20$					
	X_1	X_2	X_3	X_4	X_5
TC-SIS	1.000	0.878	0.886	0.874	0.894
MMLE-SIS	1.000	0.882	0.892	0.872	0.890
DC-SIS	1.000	0.880	0.880	0.864	0.890
PC-SIS	1.000	0.832	0.806	0.800	0.828

Table 2.7: First Ten Components of $\boldsymbol{\beta}$ in Example 4

β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9	β_{10}
5	-5	5.5	-6	6	4	4.5	-5.5	5	-4

2.9 (mean minimum model sizes). For this example, we obtain the largest gap (of the four simulations considered) in mean minimum model size between our method and DC-SIS (95.610 for TC-SIS versus 142.084 for DC-SIS, a difference of nearly 50 predictors). Furthermore, for each of the cutoff values $d = 10$, 15, and 20, TC-SIS captures a higher proportion of the causative predictors than DC-SIS. (Excluding the few cases where the two methods both acquire a given predictor 100% of the time). This suggests that, under the conditions prescribed by Example 4, the use of trend correlation in screening continuous, but not necessarily normally distributed, data may prove superior to extant methods such as DC-SIS.

Table 2.8: Example 4: Proportion of Replications Where X_j is in the Top d Causative Covariates

$d = 10$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.874	0.498	0.804	0.746	0.998	1.000	0.928	0.702	0.580	0.534
DC-SIS	0.832	0.444	0.762	0.678	0.992	1.000	0.914	0.648	0.534	0.468
$d = 15$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.934	0.622	0.874	0.840	1.000	1.000	0.964	0.806	0.710	0.660
DC-SIS	0.900	0.550	0.838	0.768	0.998	1.000	0.950	0.742	0.640	0.564
$d = 20$										
	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
TC-SIS	0.954	0.678	0.910	0.894	1.000	1.000	0.978	0.868	0.762	0.728
DC-SIS	0.926	0.620	0.874	0.828	1.000	1.000	0.964	0.796	0.712	0.650

Table 2.9: Mean Minimum Model Sizes for Examples 1 through 4

	TC-SIS	MMLE-SIS	DC-SIS	PC-SIS
Example 1	54.674	150.340	64.990	93.018
Example 2	112.627	508.672	125.258	171.829
Example 3	41.976	41.934	46.470	93.270
Example 4	95.610	-	142.084	-

2.3.5 Example 5

We apply the proposed TC-SIS screening procedure to a clinical dataset pertaining to polycystic ovary syndrome (PCOS) affection status (dbGaP Study Access: phs000368.v1.p1). This data consists of 4,099 observations (1,043 cases and 3,056 controls) of each of 731,442 SNPs. The goal of this analysis is to identify the most influential susceptibility loci that affect PCOS status. The response for this data is PCOS affection status and the predictors are the encoded SNP genotype values. Compared to the sample size ($n = 4,099$), the number of parameters ($p = 731,442$) is ultrahigh dimensional.

Although the proposed TC-SIS approach is very powerful at filtering out noise and recognizing the truly important predictors among over half a million candidates, it may neglect some important predictors that are individually uncorrelated yet jointly correlated with the response. Furthermore it may rank highly some unimportant predictors that are spuriously correlated with the response due to their strong collinearity with other influential predictors [92]. To overcome these shortcomings, we use an iterative procedure on top of the proposed TC-SIS (call it TC-ISIS) to address these weaknesses. The main difference between TC-SIS and TC-ISIS is that TC-SIS selects the final subset in a single step, while TC-ISIS builds upon the selection procedure gradually, with multiple steps. This iterative analysis is modeled after that of [92]. Specifically, using the iterative screening approach outlined therein for their iterative real data analysis, but replacing their use of DC-ISIS with our TC-ISIS process, we first iterate over the values $p_1 = 5, 6, \ldots, [n/\log(n)] = 493$ to determine a value for p_1. The optimal value for p_1 is that which minimizes the mean squared prediction error (MSPE) for logistic regression in the first iterative step, again, as outlined

by [92]. When determining the optimal value for p_1, we used 75% of the observed data for training and 25% for testing. It was found that $p_1 = 191$ and $p_2 = [n/\log(n)] - p_1 = 302$ as initial values minimized the MSPE in our case.

After screening the real data set using the iterative application of our proposed method, we obtain a relatively small set of SNPs with positive screening scores scores (450 such SNPs). Using 10-fold cross validation in the R package `glmnet`, we then post screen our selected set of SNPs via a variety of penalized regression methods to further reduce the final model size. We use three such techniques: lasso [76], adaptive-lasso [95], and elastic net (with $\alpha = 0.09$; see below for the use of α) [96]. Each of these three methods employs penalized logistic regression of the negative binomial log-likelihood, which is as follows:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \left\{ -\left[\frac{1}{N}\sum_{i=1}^{N} y_i(x_i^T\boldsymbol{\beta}) - \log(1+e^{x_i^T\boldsymbol{\beta}})\right] + \lambda\left[\frac{(1-\alpha)}{2}\|\boldsymbol{\beta}\|_2^2 + \alpha\|\boldsymbol{\beta}\|_1\right]\right\}. \tag{2.1}$$

The aggressiveness of the penalty is controlled by a parameter λ. The parameter λ is chosen using a cross-validated coordinate descent approach, where the objective is minimizing the predicted misclassification rate. This process is handled internally in the `glmnet` package in R [30]. When $\alpha = 1$ in (2.1), we have the lasso penalty function. To perform adaptive lasso, we first fit weights for each component of $\boldsymbol{\beta}$ using ridge regression ($\alpha = 0$). Our elastic net model is tuned in a manner similar to the original paper by [96]: We first pick a grid of values for α. For simplicity we used $\alpha_k = \left\{\frac{k}{100}\right\}$ for $k = 1, 2, 3, \ldots 99$. (When $\alpha = 1$, this is lasso, which is examined separately above). Then, for each α_k, we fit a model for our selected parameters using elastic net. As with lasso and adaptive lasso, the other tuning parameter, λ, is selected by tenfold cross validation. The chosen λ is the one giving the smallest 10-fold cross validated misclassification error. Here, our tuning procedures found $\alpha = 0.09$ to be the α for which misclassification error was minimized. As a measure for goodness-of-fit for our final model, we evaluate two selection criteria: Akaike's Information Criterion (AIC) and Misclassification Rate (MR).

The results of the three models and two selection criteria are tabulated in Table 2.10. One can see that TC-ISIS+Adaptive Lasso yields the sparsest model size of 56 final predictors, as well as the smallest misclassification rate and the smallest AIC. The estimated

coefficients of these 56 parameters can be obtained from the `glmnet` package.

Table 2.10: Empirical results of real data analysis.

Post Screening Method	Model size	AIC	MR
Lasso	71	-1735.27	21.59%
Adaptive Lasso	56	-1755.10	21.08%
Elastic Net ($\alpha = 0.09$)	91	-1691.66	21.15%

2.4 Discussion

In this chapter we proposed a new feature screening procedure using trend correlation. The finite performance of the proposed TC-SIS screening procedure was illustrated via performing four simulations comparing our method alongside three other relevant extant approaches for screening categorical data. We also vetted the performance of our method in a real data analysis of an ultrahigh dimensional data set. We furthermore established the strong screening consistency for this procedure when the number of predictors diverges exponentially vis-à-vis the sample size. Strong screening consistency is much harder to achieve compared to the sure screening property, as it guarantees that not only the selected model *contains* the true model, but also that the selected model *equals* the true model asymptotically.

The emphasis of the proposed TC-SIS procedure is to detect the most influential predictors from ultrahigh dimensional data having polytomous features and dichotomous response, which represents a possibly useful application to genome-wide association studies for human disease. In addition to performing well under the focused upon categorical context, we also illustrated that TC-SIS has the potential to perform well under more general settings for continuous data. This acted as a proof-of-concept for a general application of TC-SIS.

Compared to the general association yielded by methods such as the Pearson chi-squared based PC-SIS approach, TC-SIS may be able to explore more information related to the *trend* between the response and the predictors, e.g., larger predictor values tends to be

associated with larger (or conversely, smaller) response values in certain practices. TC-SIS assumes milder conditions than other approaches in that it neither requires any regression function nor assumes any specific distribution for the covariates or the response, like unto the spherical distribution assumptions of many other feature screening approaches. Based on these differences, TC-SIS serves as a unified alternative to existing model-based sure screening procedures.

The proposed TC-SIS method can be easily extended to a categorical response having greater than two levels if needed, however we only considered binary Y because this allows for some simplification of our notation and proofs. It has been noted that the choice of a cutoff c is of importance in some feature screening literature. Several methods have been proposed to determine such a cutoff, e.g. [93], [91], [43], and [48]. One may adopt their ideas for the proposed TC-SIS approach, but we have opted not to pursue this further as it is beyond the perview of this work.

2.5 Appendices

In this section we present in full the proofs for Theorems 2.2.1 and 2.2.2 given at Subsection 2.2.3 of the main text. Before proceeding into the proofs, we will establish a pair of lemmas that will be used repeatedly in the proofs.

2.5.1 Some Prefacing Lemmas

In reference to the numerator of the definition of $\hat{\varrho}_j$ given in Subsection 2.2.2 of the main text, we define

$$\hat{\tau}_j = \left| \sum_{k=1}^{K_j} \sum_{m=0}^{1} (v_k^{(j)} - \bar{v}^{(j)})(m - \bar{Y}) \hat{p}_{km}^{(j)} \right|$$

as an estimator of the covariance between X_j and Y (i.e., $\mathrm{Cov}(X_j, Y)$). Furthermore, in accordance with the denominator of the definition of $\hat{\varrho}_j$ given in Subsection 2.2.2 of the main text, we define

$$\hat{\sigma}_j = \sqrt{\sum_{k=1}^{K_j} (v_k^{(j)} - \bar{v}^{(j)})^2 \hat{p}_k^{(j)}}$$

as an estimator of the standard deviation of X_j (i.e., σ_j). Similarly, we define

$$\hat{\sigma}_Y = \sqrt{\sum_{m=0}^{1} (m - \bar{Y})^2 \hat{p}_m}$$

as an estimator of the standard deviation of Y (i.e., σ_Y).

Lemma 2.5.1. *$\hat{\tau}_j$ is a consistent estimator of $|Cov(X_j, Y)|$.*

Proof. We begin by first showing that $\hat{\tau}_j$ is equal to the following estimator for $|\text{Cov}(X_j, Y)|$:

$$\left| \frac{1}{n} \sum_{i=1}^{n} (X_{ij} - \bar{X}_j)(Y_i - \bar{Y}) \right|, \tag{2.2}$$

where $\bar{X}_j = \frac{1}{n} \sum X_{ij}$ and $\bar{Y}$ is defined the same as before. Specific to the focus of this article, we know that $Y_i \in \{0, 1\}$ and $X_{ij} \in \{v_1^{(j)}, v_2^{(j)}, \ldots, v_{K_j}^{(j)}\}$, and that $\bar{X}_j = \bar{v}^{(j)}$. Let n_{km} denote the number of observations satisfying $X_{ij} = k$ and $Y_i = m$, meaning we can write $\hat{p}_{km}^{(j)} = \frac{n_{km}}{n}$. We can now rewrite Statement (2.2) as follows:

$$\left|\frac{1}{n}\sum_{i=1}^{n}(X_{ij}-\bar{X}_j)(Y_i-\bar{Y})\right|$$

$$=\left|\frac{1}{n}\sum_{i=1}^{n}(X_{ij}-\bar{X}_j)(1-\bar{Y})-\frac{1}{n}\sum_{i=1}^{n}(X_{ij}-\bar{X}_j)(\bar{Y})\right|$$

$$=\left|\frac{1}{n}\left((v_1^{(j)}-\bar{v}^{(j)})(1-\bar{Y})n_{11}+\cdots+(v_{K_j}^{(j)}-\bar{v}^{(j)})(1-\bar{Y})n_{K_j1}\right)\right.$$
$$\left.-\frac{1}{n}\left((v_1^{(j)}-\bar{v}^{(j)})(\bar{Y})n_{10}+\cdots+(v_{K_j}^{(j)}-\bar{v}^{(j)})(\bar{Y})n_{K_j0}\right)\right|$$

$$=\left|\frac{1}{n}\sum_{k=1}^{K_j}\sum_{m=0}^{1}(v_k^{(j)}-\bar{v}^{(j)})(m-\bar{Y})n_{km}\right|$$

$$=\left|\sum_{k=1}^{K_j}\sum_{m=0}^{1}(v_k^{(j)}-\bar{v}^{(j)})(m-\bar{Y})\hat{p}_{km}^{(j)}\right|$$

$$=\hat{\tau}_j.$$

As convenient, we will use the form (2.2) when discussing $\hat{\tau}_j$.

Secondly, we now show that $\hat{\tau}_j$ is a consistent estimator of $|\mathrm{Cov}(X_j,Y)|$. Expanding Statement (2.2) yields

$$\hat{\tau}_j=\left|\frac{1}{n}\sum X_{ij}Y_i-\frac{1}{n}\sum\bar{X}_jY_i-\frac{1}{n}\sum X_{ij}\bar{Y}+\frac{1}{n}\sum\bar{X}_j\bar{Y}\right|. \tag{2.3}$$

By applying the weak law of large numbers [15] we have

$$\frac{1}{n}\sum X_{ij}Y_i\xrightarrow{P}\mathbb{E}(X_jY),$$

$$\frac{1}{n}\sum \bar{X}_j Y_i \xrightarrow{P} \mathbb{E}(X_j)\mathbb{E}(Y),$$

$$\frac{1}{n}\sum X_{ij} \bar{Y} \xrightarrow{P} \mathbb{E}(X_j)\mathbb{E}(Y),$$

$$\frac{1}{n}\sum \bar{X}_j \bar{Y} \xrightarrow{P} \mathbb{E}(X_j)\mathbb{E}(Y),$$

with all convergence being in probability. Hence we have by the Mann-Wald Theorem [11, 58, 72] (also known as the Continuous Mapping Theorem),

$$\begin{aligned} \hat{\tau}_j &\xrightarrow{P} |\mathbb{E}(X_j Y) - 2\mathbb{E}(X_j)\mathbb{E}(Y) + \mathbb{E}(X_j)\mathbb{E}(Y)| \\ &= |\mathbb{E}(X_j Y) - \mathbb{E}(X_j)\mathbb{E}(Y)| \\ &= |\mathrm{Cov}(X_j, Y)|\,. \end{aligned}$$

(Note of course that the absolute value function is a continuous function). Therefore, Lemma 2.5.1 establishes that $\hat{\tau}_j$ is a consistent estimator of $|\mathrm{Cov}(X_j, Y)|$. □

Lemma 2.5.2. *It can be shown that $\hat{\sigma}_j$ is a consistent estimator of σ_j and $\hat{\sigma}_Y$ is a consistent estimator of σ_Y.*

Proof. We begin by noting that, similar unto what was done in rewriting $\hat{\tau}_j$ in Lemma 2.5.1, we can rewrite $\hat{\sigma}_j$ as follows:

$$\begin{aligned}
\hat{\sigma}_j^2 &= \sum_{k=1}^{K_j} (v_k^{(j)} - \bar{v}^{(j)})^2 \hat{p}_k^{(j)} \\
&= \frac{1}{n} \sum_{k=1}^{K_j} (v_k^{(j)} - \bar{v}^{(j)})^2 n_k \\
&= \frac{1}{n} \left((v_1^{(j)} - \bar{X}_j)^2 n_1 + \cdots + (v_{K_j}^{(j)} - \bar{X}_j)^2 n_{K_j} \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} (X_{ij} - \bar{X}_j)^2,
\end{aligned} \tag{2.4}$$

where n_k is the number of times X_j takes on the value v_k. Similarly we can obtain for $\hat{\sigma}_Y^2$

$$\hat{\sigma}_Y^2 = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \bar{Y})^2. \tag{2.5}$$

Equations (2.4) and (2.5) can both be written using a general random variable W as follows. Let $W_1, W_2, \ldots, W_n$ be realizations of a bounded random variable W. Define

$$S^2 = \frac{1}{n} \sum_{i=1}^{n} (W_i - \bar{W})^2,$$

where $\bar{W} = \frac{1}{n} \sum W_i$. Let σ^2 denote the variance of W. Define the unbiased sample variance as

$$\hat{\sigma}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (W_i - \bar{W})^2.$$

We then have

$$S^2 = \frac{n-1}{n}\hat{\sigma}^2.$$

It follows that

$$\mathbb{E}(S^2) = \frac{(n-1)}{n}\sigma^2, \quad \mathrm{Var}(S^2) = \left(\frac{n-1}{n}\right)^2 \mathrm{Var}(\hat{\sigma}^2).$$

We can easily show that $\hat{\sigma}^2$ is a consistent estimator of σ^2. This is done as follows. [11, See also Example 5.5.3 of] It can be established that

$$\mathrm{Var}(\hat{\sigma}^2) = \frac{1}{n}\left(\mu_4 - \frac{n-3}{n-1}\mu_2^2\right), \tag{2.6}$$

where $\mu_\ell = \frac{1}{n}\sum(W_i - \mathbb{E}W)^\ell$ (with $\ell = 2$ or $\ell = 4$) [12]. Employing Chebychev's inequality for any $\varepsilon > 0$, we get the following:

$$\mathbb{P}\left(|\hat{\sigma}^2 - \sigma^2| \geq \varepsilon\right) \leq \frac{\mathrm{Var}(\hat{\sigma}^2)}{\varepsilon^2}. \tag{2.7}$$

Statements (2.6) and (2.7) yield that

$$\lim_{n\to\infty} \mathbb{P}\left(|\hat{\sigma}^2 - \sigma^2| \geq \varepsilon\right) = 0$$

since $|\mu_\ell| < \infty,\ \ell = 2, 4$, if W is bounded. (It is easy to see the finiteness of the moments here: Any bounded function integrable on a compact domain also has integrable powers over that domain. The random variable W is in fact such a function. See also Corollary 5.11 in [50]) This means that $\hat{\sigma}^2$ is indeed a consistent estimator of σ^2. It is easily observable that the numeric sequence $\left\{\frac{n-1}{n}\right\}_{n=1}^{\infty}$ converges to one. By the Mann-Wald theorem once again, we have that $S^2 = \frac{n-1}{n}\hat{\sigma}^2$ converges in probability to $1 \cdot \sigma^2 = \sigma^2$. This confirms that S^2 is a consistent estimator of σ^2. Therefore, the Mann-Wald theorem guarantees that S is a consistent estimator of $\sigma = \sqrt{\mathrm{Var}(W)}$. This establishes that $\hat{\sigma}_j$ and $\hat{\sigma}_Y$ are consistent estimators of σ_j and σ_Y. □

Lemma 2.5.3. *$\hat{\varrho}_j$ is a consistent estimator of ϱ_j.*

Proof. The definition of $\hat{\varrho}_j$ given in Subsection 2.2.2 of the main text can be naturally rewritten as

$$\hat{\varrho}_j = \frac{\hat{\tau}_j}{\hat{\sigma}_j \hat{\sigma}_Y}.$$

We will employ the Mann-Wald theorem three times. This theorem asserts that continuous functions on $\mathbb{R}^\ell$ preserve convergence in probability. Thus if Z_1 is a consistent estimator of α_1 and Z_2 is a consistent estimator of α_2, then for any continuous function f on $\mathbb{R}^2$, we have $f(Z_1, Z_2) \xrightarrow{P} f(\alpha_1, \alpha_2)$. This implies that $f(Z_1, Z_2)$ is a consistent estimator of $f(\alpha_1, \alpha_2)$.

Define the function $f(\hat{\sigma}_j, \hat{\sigma}_Y) = \frac{1}{\hat{\sigma}_j \hat{\sigma}_Y}$ on $\mathbb{R}^2_+ = (0, \infty)^2$ (All positive real-valued 2-vectors). This function is well defined and continuous on its entire domain. (Note that, in line with condition (C1), we can assume without loss of generality that $\hat{\sigma}_j$ and $\hat{\sigma}_Y$ are both positive). This implies by the Mann-Wald theorem that in fact $\frac{1}{\hat{\sigma}_j \hat{\sigma}_Y}$ is a consistent estimator for $\frac{1}{\sigma_j \sigma_Y}$. Then, under the obvious assumption that multiplication of real numbers is a continuous function, we obtain

$$\hat{\varrho}_j = \hat{\tau}_j \frac{1}{\hat{\sigma}_j \hat{\sigma}_Y}$$

is a consistent estimator of ϱ_j. □

2.5.2 Proofs of Theorem 2.2.1 and 2.2.2

Proof. The proof of these two theorems is accomplished in three steps:

- Step 1: We show that there exists a positive value $\varrho_{\min} > 0$ such that $\varrho_j > \varrho_{\min}$ holds for any $j \in \mathcal{S}_T$ (this is also Corollary 2.2.3). Recall that we defined

$$\omega_{km}^{(j)} = \left| (v_k^{(j)} - \mathbb{E}(X_j))(m - \mathbb{E}(Y)) p_{km}^{(j)} \right|$$

in condition (C2) given in Subsection 2.2.3 of the main text.

It follows that for $j \in \mathcal{S}_T$,

$$\begin{aligned} \varrho_j = \frac{\sum\limits_{\substack{1\le k\le K_j \\ 0\le m\le 1}} \omega_{km}^{(j)}}{\sigma_j \sigma_Y} &\geq \frac{1}{\sigma_{\max}^2} \sum_{\substack{1\le k\le K_j \\ 0\le m\le 1}} \omega_{km}^{(j)} \quad \text{by (C1)}, \\ &\geq \frac{1}{\sigma_{\max}^2} \max_{\substack{1\le k\le K_j \\ 0\le m\le 1}} \omega_{km}^{(j)} \\ &\geq \frac{\omega_{\min}}{\sigma_{\max}^2} \quad \text{by (C2)}, \\ &> 0. \end{aligned}$$

Define $\varrho_{\min} = \omega_{\min}/(2\sigma_{\max}^2)$. Then $\varrho_j > \varrho_{\min} > 0$ for all $j \in \mathcal{S}_T$. This establishes a positive lower bound on ϱ_j for all $j \in \mathcal{S}_T$, and hence completing Step 1 and also Corollary 2.2.3 of Subsection 2.2.3.

- Step 2: We now show that $\hat{\varrho}_j$ is a uniformly consistent estimator of ϱ_j for each $1 \leq j \leq p$ (this is also Corollary 2.2.4). From Lemma 2.5.3, we know that $\hat{\varrho}_j$ is consistent estimator of ϱ_j. This implies that for any $1 \leq j \leq p$ and any $\varepsilon > 0$, we have

$$\mathbb{P}(|\hat{\varrho}_j - \varrho_j| > \varepsilon) \to 0 \quad \text{as } n \to \infty.$$

Let $J = \text{argmax}_{1\le j\le p} |\hat{\varrho}_j - \varrho_j|$. Then, since $J \in \{1, 2, \ldots, p\}$ itself, we indeed know that

$$\mathbb{P}(|\hat{\varrho}_J - \varrho_J| > \varepsilon) \to 0 \quad \text{as } n \to \infty$$

for any $\varepsilon > 0$. In other words, we have that

$$\mathbb{P}\left(\max_{1\le j\le p} |\hat{\varrho}_j - \varrho_j| > \varepsilon\right) \to 0 \quad \text{as } n \to \infty$$

for any $\varepsilon > 0$. This shows that $\hat{\varrho}_j$ is a *uniformly* consistent estimator of ϱ_j, thus completing Step 2 and also Corollary 2.2.4 of Subsection 2.2.3.

- Step 3: We show that there exists a positive constant $c > 0$ such that

$$\mathbb{P}(\mathcal{S}_T \subseteq \widehat{\mathcal{S}}) \to 1 \text{ as } n \longrightarrow \infty.$$

Note that this is Theorem 2.2.1 of Subsection 2.2.3.

Let $c = (2/3)\varrho_{\min}$. Suppose by way of contradiction that this c is insufficient to be able to claim $\mathcal{S}_T \subseteq \widehat{\mathcal{S}}$. This would mean that there exists some $j^* \in \mathcal{S}_T$, yet $j^* \notin \widehat{\mathcal{S}}$. It then follows that we must have (by the definition of $\widehat{\mathcal{S}}$)

$$\hat{\varrho}_{j^*} \leq (2/3)\varrho_{\min},$$

while at the same time having (by the conclusion of Corollary 2.2.3)

$$\varrho_{j^*} > \varrho_{\min} > (2/3)\varrho_{\min}.$$

From this we can conclude that $|\hat{\varrho}_{j^*} - \varrho_{j^*}| > (1/3)\varrho_{\min}$, which implies that

$$\max_{1 \leq j \leq p} |\hat{\varrho}_j - \varrho_j| > (1/3)\varrho_{\min}$$

as well.

However, we know by the uniform consistency of $\hat{\varrho}_j$ that by letting $\varepsilon = 1/3\varrho_{\min}$, we have

$$\mathbb{P}(\mathcal{S}_T \not\subseteq \widehat{\mathcal{S}}) \leq \mathbb{P}\left(\max_{1 \leq j \leq p} |\hat{\varrho}_j - \varrho_j| > (1/3)\varrho_{\min}\right) \to 0 \quad \text{as } n \to \infty.$$

This is a contradiction to the assumption of non containment above. This yields

$$\mathbb{P}(\mathcal{S}_T \subseteq \widehat{\mathcal{S}}) \to 1 \quad \text{as } n \to \infty,$$

proving Theorem 2.2.1.

- Step 4: We finish by showing that there exists a positive constant $c > 0$ such that

$$\mathbb{P}(\widehat{\mathcal{S}} \subseteq \mathcal{S}_T) \to 1 \text{ as } n \to \infty.$$

Suppose again by way of contradiction that $\widehat{\mathcal{S}} \not\subseteq \mathcal{S}_T$. Then there is some $j^* \in \widehat{\mathcal{S}}$, yet $j^* \notin \mathcal{S}_T$. This means that (by the definition of $\widehat{\mathcal{S}}$)

$$\hat{\varrho}_{j^*} > (2/3)\varrho_{\min},$$

while at the same time (by condition (C2)) having

$$\varrho_{j^*} = 0.$$

It now follows that

$$|\hat{\varrho}_{j^*} - \varrho_{j^*}| > (2/3)\varrho_{\min},$$

which implies that $\max_{1 \le j \le p} \ |\hat{\varrho}_j - \varrho_j| > (2/3)\varrho_{\min}$ as well.

Set $\varepsilon = (2/3)\varrho_{\min}$. By uniform consistency we have

$$\mathbb{P}(\widehat{\mathcal{S}} \not\subseteq \mathcal{S}_T) \le \mathbb{P}\left(\max_{1 \le j \le p} |\hat{\varrho}_j - \varrho_j| > (2/3)\varrho_{\min}\right) \to 0 \quad \text{as } n \to \infty.$$

This contradicts our finding that $\max_{1 \le j \le p} \ |\hat{\varrho}_j - \varrho_j| > (2/3)\varrho_{\min}$. Hence it in fact follows that

$$\mathbb{P}(\widehat{\mathcal{S}} \subseteq \mathcal{S}_T) \to 1 \quad \text{as } n \to \infty.$$

Combining Step 3 and Step 4 together, we conclude that for $c = (2/3)\varrho_{\min}$, we have $\mathbb{P}(\mathcal{S}_T = \widehat{\mathcal{S}}) \to 1$ as $n \to \infty$, completing the proof of Theorem 2.2.2.

□

CHAPTER 3

Interaction Screening Using Joint Cumulants

3.1 Introduction

Ultrahigh dimensional data in fields such as bioinformatics, medical imaging, finance, and the social sciences has become increasingly commonplace. With the yet to cease rapid advances in data collection techniques and computing power, there has arisen an accompanying desire to more comprehensively analyze said data. However, a significant challenge in dealing with ultrahigh dimensional data comes in the fact that classical methods often become intractable or unreliable when confronted with such dimensionality. Here, and throughout this chapter, we will use the term *high dimensional* to refer to the case when $p = \mathcal{O}(n^{\xi})$ for some constant $\xi > 0$ and we will used the term *ultrahigh dimensional* to refer to the case when $\log(p) = \mathcal{O}(n^{\xi})$ for some constant $\xi > 0$, where p is the number of predictors and n is the number of observations of each of those predictors.

When the feature space is ultrahigh dimensional, [24] introduces us to the concept of sure independence screening (SIS). Many methods possessing the sure screening property of [24] have been developed. Several of these strengthen the original statements of (weak) sure screening and establish that the newly proposed method in question has the *strong* sure screening property. See for example [53], [26], and [43]. However, while there exists (including and beyond those previously mentioned above) an abundance of feature screening methods for marginal effects in ultrahigh dimensional feature spaces, [see for example 4, 19, 20, 47, 82, 85, 93, with an overview given in 55], less consideration has been given to determining *interactions* between features. Nevertheless, because the relationship between predictors and response is often more complex than can be captured by main effects alone, developing techniques for determining interactive effects between predictors is vital. It is common to find two predictors that are simultaneously associated with the response and

whose effects on the response may not be additive. In particular, such interaction between variables has been recognized as playing a pivotal role in contributing to the variation of the response [28, 38]. Procedures addressing interaction screening will allow us to better ascertain the interplay between covariates when modelling a response.

A significant challenge in the topic of interaction screening, especially when the feature space is high or ultrahigh dimensional, is that of computational feasibility [38]. For a general discussion on the computational challenges of ultrahigh dimensional data, see for example [23] and [21]. Early interaction selection literature such as [86], [87], [90], and [13] use an approach that is sometimes referred to as *joint analysis*. These methods examine all marginal and interaction effects in a single global search. Such joint analysis approaches are feasible when the feature space is not high dimensional [38]. However, in many modern interaction screening settings (in which the feature space is likely *not* medium or low dimensional), the most prevalent barrier comes from the growing number of interactions that must be considered. When the feature space is high or ultrahigh dimensional, and p is also large generally, not only are we faced with the theoretical difficulties of having $p \gg n$, but we also must handle a number of interactions on the order of $\mathcal{O}(p^2)$ [29, 37]. An exhaustive search of all possible interactions requires examining $\binom{p}{2}$ potential interactions for association with the response, something that may not always be feasible to the same degree that the joint analysis approaches were for medium and low dimensional data [28, 38]. For even a relatively small value of $p = 10,000$, an exhaustive search requires looking at nearly 50 million potential interactions. When p becomes even larger, the number of interactions to be considered can easily number in the trillions. (For example, $p = 45,000$ would yield over a trillion potential interactions to consider).

A second obstacle to the development of interaction feature screening approaches comes in the fact that many classic approaches designed for detecting interaction may lose power or become increasingly unstable as the number of potential interactions grows [28]. Another barrier in the development of interaction feature screening approaches is related to the excessive reliance on an *a priori* heredity structure. Specifically, *weak heredity* requires

that the interaction between X_{j_1} and X_{j_2} is considered only if at least one of the main effects, X_{j_1} or X_{j_2}, is individually associated with the response. Similarly, *strong heredity* requires that the interaction between X_{j_1} and X_{j_2} is considered only if *both* main effects are influential. Many currently existing approaches rely on determining marginal effects first and then examining only a subset of interactions having at least one influential main effect. [See e.g. 37, 38, 52, each of which will be considered below.]. While these approaches significantly reduce the number of interactions that are to be examined, they neglect a scenario where there exists an interaction with strong effect, but for which both corresponding main effects are very weak [60], which is found to be important in many practical applications [5, 17, 29, 61, 79]. As such, the ideal interaction screening method will not *a priori* assume that a model must possess certain strong marginal features if it contains an associated interaction.

We now examine a number of extant interaction screening methods individually, while noting the challenges and drawbacks of each.

In [69], the authors examine the use of random forests [10] in screening time-to-event data for interactions. In brief, they take the cumulative hazard function as the response and fit a variant of random forests called random survival forests. The variable importance measures from this random forest model are then used in selecting important main effects. The variable inclusion frequencies of each pairwise combination of these selected main effects are then calculated. Given a predetermined cutoff d, the top d most frequent pairs are taken as the potential interactions for the final model. The final model is determined by a fitting a CoxBoost model [8]. This approach works well when the feature space is medium or high dimensional, but as the feature space becomes ultrahigh dimensional, random forests may become unstable or intractable computationally due to the required computational intensity and memory demands [43, 70, 88, 94].

[28] proposes an approach to feature screening in the context of classification. They base their method off of an adaptive selection of sparse linear or quadratic discriminant analysis (LDA & QDA). Although presented as a generally applicable interaction screening algorithm for high dimensional data, their empirical results are couched in the realm of data

with $p \leq 500$. (They present a real data analysis where $p = 231$ and $n = 77$, for example). As such, while the data sets they use are indeed "high dimensional," in that $p > n$, they never address the computational challenges of applying their approach to much larger absolute data size (data where say $p > 10,000$). (For instance, [19] specifically addresses issues of LDA and QDA when applied to high dimensional data with large p. This alone makes a discussion on the computational feasibility of feature screening methods derived from LDA and QDA entirely relevant).

[37] presents a two-stage interaction screening process called *regularization algorithm under marginality principle* (RAMP) which employs lasso [76]. Their approach to interaction screening is rather straight forward: *(First stage)* Given a set of predictors, RAMP first applies lasso to this set of predictors to determine as set of main effects, $\widehat{M}$. *(Second stage)* Then, using only those predictors found in $\widehat{M}$, apply lasso to both the marginal and the order-2 terms among all elements of $\widehat{M}$. The output of lasso after this second stage will be taken as the final model. It should be noted, however, that two important challenges arise with RAMP. First, because lasso (and penalized regression in general) is not applicable to ultrahigh dimensional feature spaces [among many, see 24], RAMP cannot be applied in any setting with ultrahigh dimensional data. Furthermore, RAMP relies on strong heredity, which, as has been well established previously, significantly limits the applicability of the method.

One of the most common applications of interaction feature screening for high and ultrahigh dimensional data is in the field of bioinformatics and genetics. [See e.g. 29, 52, 65, 79, 83]. Interactions between single-nucleotide-polymorphisms (SNPs) or interactions between genes is of particular interest. (Such genetic interaction is broadly referred to as *epistasis*). [52] offers one method for detecting epistasis in ultrahigh dimensional genetic data sets. They use the standard SIS procedure of [24] to first determine a set of main effects. Then using this set of main effects, they use weak heredity to examine all pairwise interactions containing at least one causative main effect. Thus, their method relies on principles of model heredity, which has been shown to be subject to the drawbacks already

addressed above.

Seeking to overcome the limitations of directly assuming model heredity, [79] suggest an interaction feature screening method for ultrahigh dimensional data that does not rely on first screening for marginal effects. Nevertheless, their method is limited in the types of covariates it admits and assumes an interactive structure that may be too rigid for broader application. In detail, the method they present requires that each predictor is categorical with three levels (i.e. encoding for homozygous recessive, heterozygous, and homozygous dominant allele pairs). This encoding is used to then create nine dummy variables, representative of each of the nine possible pairwise genotype combinations between SNPs. This in turn inflates the feature space to nine times its original size. Moreover, as their approach is directly dependent on SIS [24], a strict linear model between response and predictor must be assumed.

[44] also uses SIS [24] to screen for interactions in genetic data. Briefly, they first screen for a set of main effects using SIS. This is then followed by applying SIS to find the pseudomarkers most associated with the residuals of regressing the main effects onto the response. A multilocus model is finally used to determine important interactive effects. It should here be noted that their approach is not directly reliant on model heredity, however, it is nevertheless subject to the rigid model limitations of SIS (viz. a strict linear model). Furthermore, the authors seem rather hesitant to claim applicability of their method to ultrahigh dimensional data. In their words, the possible number of predictors examined "depends on the number of individuals in the data set. Generally, the maximum number is assumed to be 10 times the number of individuals [41], but in our experience, an even smaller proportion may be optimal [45]."[44] Hence, in feature spaces for which $p/n > 10$, this method may not be fully applicable.

The two-stage grouped-SIS (TS-GSIS) procedure of [29] also attempts to partially address the issues of model heredity in the context of epistasis. They use a grouped sure independence screening (GSIS) method from [63] to determine gene-gene interactions, and then, based on those gene-gene interactions, they apply lasso [76] to select individual SNP-

SNP interactions. The GSIS method [63] is briefly described as follows: First group SNPs by pre-determined gene groups. Then fit a stepwise forward selection regression model one gene group at a time by using a subset of SNPs in said group. For each regression model selected by forward selection, let the quotient of the residual sum of squares and the degrees of freedom act as a screening utility score. This procedure is done for each gene group. The two stages of TS-GSIS are comprised of first applying GSIS to screen for gene-level (not SNP-level) main effects and then repeating the GSIS process on all pairs of genes for which at least one gene in the pair was determined to be marginally associated with the response. These two main stages are followed by a final step whereat lasso is applied to the sets of SNPs (from stage one) and SNP-SNP interactions (from stage two) derived from the genes found in the two main stages of the algorithm. Hence, while this final step potentially allows for admission of interactions between SNPs not found to be marginally important, TS-GSIS does not fully escape the constraints of weak heredity, as it yet enforces weak heredity at the gene level.

Herein we will propose a new and novel method for interaction screening of ultrahigh dimensional data that seeks to address many of the issues with existing interaction screening methods as outlined above. Unlike many of the aforementioned extant processes for interaction screening, our method will not rely on the presence of marginal effects (model heredity) in order to determine interactions between covariates. Moreover, our method is computationally more cost effective than current methods, making the application of our proposed techniques to larger and larger data distinctively feasible.

We will compare our newly proposed method empirically with two existing methods for interaction feature screening: The iterative forward selection method (iFORM) of [38] and the generalization of the Pearson Chi-squared-based (PC-SIS) method given by [43]. These two methods most closely resemble our to-be-proposed method in that they admit ultrahigh dimensional data and possess certain salient theoretical properties.

The iFORM uses an adaptation of forward-selection based on Bayesian information criterion (BIC). This is done by performing a standard stepwise forward selection [see 49]

and admitting one predictor or interaction at a time. Model fitness is determined by BIC. Their approach relies on the assumption of strong heredity, as they only admit interactions for which *both* main effects are also present. [33] presents an implementation of iFORM in the setting of epistasis.

The mathematical generalization of the PC-SIS method given by [43] will be referred to as Generalized Pearson Correlation (GPC). GPC can briefly be outlined as follows: Given a set of predictors, estimate by use of the associated sample proportions the probability of each pair of predictors being jointly associated with the response. Subtract from this the product of the estimated probabilities of each predictor from the aforementioned pair being marginally associated with the response. This quantity is then squared and divided by the product of the estimated marginal probabilities previously found. This final quantity acts as an interaction screening utility. Note that this is indeed a generalization of the Pearson Chi-squared-based test of PC-SIS: Instead of looking at joint association between just one predictor, we now examine the joint association of two predictors. While GPC does not directly require selecting marginal effects before proceeding with screening for interactive effects, their method can become computationally intractable (by their own admission) when applied to any exhaustive search of all possible interactions. (Under empirical observation, our newly proposed method ran about six to seven times faster than GPC when applied to the same data sets on the same machine). As such, GPC has never been numerically tested under a setting where marginal features have not first been selected. The empirical results of Section 3.4 will furthermore show that GPC produces rather unfavorable results when main effects are not first determined. Our newly proposed method overcomes many of the less than desirable issues of both iFORM and GPC.

The remainder of this chapter is outlined as follows: In Section 3.2, we introduce some general notation for the models we will be discussing. Section 3.3 will detail our proposed interaction screening method, including statements on the theoretical properties for said method. This is followed by several numeric simulations in Section 3.4. Among many applications of interaction screening, perhaps the most prevalent is in genome-wide

association studies (GWAS) for determining interactive (epistatic) effects between single nucleotide polymorphisms (SNPs). We too will demonstrate such an application in a real data analysis, also found in Section 3.4. In Section 3.5 we provide concluding remarks on our method and our findings. Finally, Section 3.6 contains the proofs of the theorems presented in Section 3.3.

3.2 Preliminaries

When we need to refer to a general subset of the covariate pairs (X_{j_1}, X_{j_2}), we will use $\boldsymbol{X}_{\mathcal{S}}$, where

$$\mathcal{S} \subseteq \Big(\{1,2,3,\ldots,p\} \times \{1,2,3,\ldots,p\}\Big) \setminus \{(j,j) \mid j = 1,2,3\ldots,p\}$$

is the set of covariate pairs we wish to discuss. Here $\times$ is the Cartesian product. As a matter of notation, we will let $\mathcal{S}$ refer to the model consisting of the covariate pairs found in $\mathcal{S}$. Let

$$\mathcal{S}_F = \Big(\{1,2,3,\ldots,p\} \times \{1,2,3,\ldots,p\}\Big) \setminus \{(j,j) \mid j = 1,2,3\ldots,p\}$$

designate the full model, which contains all covariate pairs. Given some model $\mathcal{S}$, we will let $\mathcal{D}\left(Y_i \mid \boldsymbol{X}_{\mathcal{S}}\right)$ indicate the conditional distribution of Y_i given the covariates of $\boldsymbol{X}_{\mathcal{S}}$. A model $\mathcal{S}$ will be considered sufficient if

$$\mathcal{D}\left(Y_i \mid \boldsymbol{X}_{\mathcal{S}_F}\right) = \mathcal{D}\left(Y_i \mid \boldsymbol{X}_{\mathcal{S}}\right)$$

The full model $\mathcal{S}_F$ is of course trivially sufficient. We are ultimately interested only in finding the sufficient model with the fewest number of interaction pairs. We will call the smallest sufficient model (i.e. the sufficient model with the least number of pairs) the true model. Our aim overall is to determine an estimated model which contains the true model and is moreover the *smallest* such model to contain the true interaction features. The following section will outline the specifics of our proposed interaction screening approach for estimating the true model. As a matter of further notation, we will denote the true model

by $\mathcal{S}_T$ and an estimated model by $\widehat{\mathcal{S}}$.

3.3 Interaction Screening using Joint Cumulants

The general form for the linear correlation between X_j and Y is given by

$$\varrho_j = \frac{\text{cov}(X_j, Y)}{\sigma_j \sigma_Y},$$

where $\text{cov}(X_j, Y)$ is the covariance of X_j versus Y, σ_j is the standard deviation of X_j, and σ_Y is the standard deviation of Y. This can be extended, as follows, to a generalized form admitting three, not two, random variables as arguments.

3.3.1 A Newly Proposed Interaction Feature Screening Method

We propose a method for interaction screening of high and ultrahigh dimensional feature spaces. We call this new method JCIS, which stands for "Joint Cumulant Interaction Screening." The details of JCIS will be outlined below. Herein we will be comparing our method to the iFORM method of [38], as well as to the generalized PC-SIS method (which they leave unnamed, but we will call GPC) of [43] . Which method we compare JCIS to will depend on the data type of the response. GPC admits only categorical responses; iFORM admits only continuous responses. JCIS allows for either categorical or continuous responses, which in and of itself is salient.

3.3.2 Theoretical Background

The multivariate analogue of the covariance function is the r-way joint cumulant of the random variables $Z_1, Z_2, \ldots, Z_r$

$$\kappa_r(Z_1, Z_2, \ldots, Z_r) = \mathbb{E}\left(\prod_{i=1}^{r}(Z_i - \mathbb{E}Z_i)\right).$$

For general references on cumulants, see for example [42] and [62].

The three-way joint cumulant, written as $\kappa_3(\cdot,\cdot,\cdot)$, between three random variables Y, X_{j_1} and X_{j_2} is given as follows:

$$\begin{aligned}\kappa_3(Y, X_{j_1}, X_{j_2}) &= \mathbb{E}\left((Y - \mathbb{E}Y)(X_{j_1} - \mathbb{E}X_{j_1})(X_{j_2} - \mathbb{E}X_{j_2})\right) \\ &= \mathbb{E}(YX_{j_1}X_{j_2}) - \mathbb{E}(YX_{j_1})\mathbb{E}X_{j_2} - \mathbb{E}(YX_{j_2})\mathbb{E}X_{j_1} \\ &\quad - \mathbb{E}(X_{j_1}X_{j_2})\mathbb{E}Y + 2\mathbb{E}Y\mathbb{E}X_{j_1}\mathbb{E}X_{j_2}\end{aligned}$$

Notice that $\kappa_3(\cdot,\cdot,\cdot)$ is zero if any one of the variables is statistically independent from the other two. From this, we can write a generalized version of the Pearson correlation discussed above that allows for the screening of interaction between two covariates in relation to their effect on a response Y. Define R_{j_1,j_2} as follows:

$$R_{j_1,j_2} = \frac{|\kappa_3(Y, X_{j_1}, X_{j_2})|}{\sqrt{\kappa_2(X_{j_1}, X_{j_1})\kappa_2(X_{j_2}, X_{j_2})\kappa_2(Y, Y)}}.$$

The two-way cumulant $\kappa_2(\cdot,\cdot)$ is just the covariance between two random variables (and thus the variance when both arguments are equal). The three-way cumulant is the generalization of the covariance.

Pairs of covariates are then ordered based on their pairwise selection score R_{j_1,j_2}. We can estimate R_{j_1,j_2} using the following formula:

$$\hat{R}_{j_1,j_2} = \frac{\sqrt{n}\left|\sum_{i=1}^{n}(X_{ij_1} - \overline{X}_{j_1})(X_{ij_2} - \overline{X}_{j_2})(Y_i - \overline{Y})\right|}{\sqrt{\left(\sum_{i=1}^{n}(X_{ij_1} - \overline{X}_{j_1})^2\right)\left(\sum_{i=1}^{n}(X_{ij_2} - \overline{X}_{j_2})^2\right)\left(\sum_{i=1}^{n}(Y_i - \overline{Y})^2\right)}}.$$

Here X_{ij_i} refers to the ith observation of X_{j_i} and similarly Y_i refers to the ith observation of Y. Also, $\overline{X}_{j_i}$ and $\overline{Y}$ refer to the standard estimates of the respective means of X_{j_1}, X_{j_2} and Y. The interactions with the largest $\hat{R}_{j_1,j_2}$ can then be selected as contributing the most to the response Y.

We form the estimated model $\widehat{\mathcal{S}}$ by choosing some cutoff $c > 0$. Methods for choosing such a c are varied (see for example [31]) and will not be the focus of this paper. Define $\widehat{\mathcal{S}}$

as follows:

$$\widehat{\mathcal{S}} = \{j : 1 \leq j \leq p,\ \widehat{R}_{j_1,j_2} > c\}.$$

Designate the numerator of $\widehat{R}_{j_1,j_2}$ (sans the constant $\sqrt{n}$ and the absolute values) by $\hat{\tau}_{j_1,j_2}$. We will show that $\hat{\tau}_{j_1,j_2}$ is a consistent estimator of the numerator of R_{j_1,j_2}, $\kappa_3(X_{j_1}, X_{j_2}, Y)$. The denominator of $\widehat{R}_{j_1,j_2}$ consists of (biased) sample estimators for the standard deviations of X_{j_1}, X_{j_2}, and Y. (The bias of these estimators will disappear asymptotically, however). It is a routine proof (see Chapter 2) to show that these estimators of the standard deviations are consistent estimators of their respective standard deviations.

3.3.3 Theoretical properties

We establish four conditions that will aid us in determining further properties of JCIS:

(C1) *Lower bound on the standard deviations.* We assume that there exists a positive constant $\sigma_{\min}$ such that for all j,

$$\sigma_j > \sigma_{\min} \qquad \text{and} \qquad \sigma_Y > \sigma_{\min}$$

This excludes features that are constant and hence have a standard deviation of 0.

(C2) *Upper bound on the standard deviations.* We take as our second condition the assumption that

$$\sigma_j, \sigma_Y < \sigma_{\max} < \infty$$

for all $j = 1, 2, 3, \ldots, p$. This is a relatively lenient condition, and one that is easy to satisfy in a large variety of applications. When each of X_{j_1}, X_{j_2}, and Y are categorical and ordinal (with Y being binary and each covariate having, without loss of generality, K many levels), we can explicitly obtain a simultaneous upper bound on each $\sigma_{j_1}, \sigma_{j_2}$, and σ_Y by use of Popoviciu's inequality on variances (see [67]):

$$\text{Let} \quad \sigma_{\max} = \max\left\{\frac{1}{2},\ \sqrt{\frac{1}{4}(v_K - v_1)}\right\},$$

where the first term in the maximum selection is a bound on the standard deviation of Y and the second term is given by Popoviciu's inequality on variances. Here v_1 and v_K represent the lowest and the highest levels (by chosen encoding) of any X_j.

(C3) *Joint cumulant association.* Define the following function on a subset of $\mathbb{R}^3$:

$$\omega_{j_1,j_2}(k_1,k_2,m) = |(k_1 - \mathbb{E}X_{j_1})(k_2 - \mathbb{E}X_{j_2})(m - \mathbb{E}Y)\pi_{j_1,j_2,Y}(k_1,k_2,m)|,$$

where $\pi_{j_1,j_2,Y}$ is the joint probability density function for X_{j_1}, X_{j_2}, and Y. Assume that $\omega_{j_1,j_2}(k_1,k_2,m)$ is of the same sign for all (k_1,k_2,m). Without loss of generality, we will assume a positive sign in each instance. Taking X_{j_1} and X_{j_2} as having the same support $\Psi \subseteq \mathbb{R}$, we now assume there exists a positive constant $\omega_{\min}$ such that

$$\min_{(j_1,j_2)\in\mathcal{S}_T} \left(\sup_{\substack{k_1,k_2\in\Psi \\ m\in\mathbb{R}}} \{\omega_{j_1,j_2}(k_1,k_2,m)\} \right) > \omega_{\min} > 0.$$

This is an easy assumption to require the true features to satisfy and should be quite easy to achieve in a wide variety of reasonable situations.

(C4) *Existence.* Assume that $R_{j_1,j_2} = 0$ for any pair of indicies $(j_1,j_2) \notin \mathcal{S}_T$. It is also to be assumed that R_{j_1,j_2} exists for all X_{j_1} and X_{j_2} pairs. That is, $R_{j_1,j_2} < \infty$.

We can now state the following theorems:

Theorem 3.3.1. *(Strong Screening Consistency). Given conditions (C1), (C2), (C3) and (C4), there exists a positive constant $c > 0$ such that*

$$\mathbb{P}(\widehat{\mathcal{S}} = \mathcal{S}_T) \longrightarrow 1 \text{ as } n \longrightarrow \infty.$$

Theorem 3.3.2. *(Weak Screening Consistency). Given that conditions (C1), (C2), and (C3) still hold, while removing from (C4) only the assumption of $R_{j_1,j_2} = 0$ for all $(j_1,j_2) \notin$*

$\mathcal{S}_T$, there exists a positive constant $c > 0$ such that

$$\mathbb{P}(\widehat{\mathcal{S}} \supseteq \mathcal{S}_T) \longrightarrow 1 \text{ as } n \longrightarrow \infty.$$

(But $\mathbb{P}(\widehat{\mathcal{S}} \subseteq \mathcal{S}_T)$ may not converge to 1 as n approaches infinity).

The proofs of these two theorems are presented in Section 3.6.

3.3.4 Corollaries

We can draw several corollaries from the proofs of Theorems 3.3.1 and 3.3.2 (see Section 3.6). These results do not themselves directly deal with sure screening, but they nevertheless allow us to make observations pertaining to the underlying mechanics of JCIS.

Corollary 3.3.3. *In the initial step of the proofs of Theorems 3.3.1 and 3.3.2, it will be shown that there exists a value $R_{\min}$ such that for any pair $(j_1, j_2) \in \mathcal{S}_T$, we have $R_{j_1,j_2} > R_{\min}$.*

Corollary 3.3.4. *From the end of Step 2 in the proofs of Theorems 3.3.1 and 3.3.2, we will conclude that $\widehat{R}_{j_1,j_2}$ converges uniformly in probability to R_{j_1,j_2}. In other words,*

$$\mathbb{P}\left(\max_{(j_1,j_2)} |\widehat{R}_{j_1,j_2} - R_{j_1,j_2}| > \varepsilon\right) \to 0 \quad \text{as } n \to \infty$$

for any $\varepsilon > 0$.

3.4 Simulations and Empirical Data Analysis

We performed four simulations on artificially generated data to empirically validate our theoretical results. Each of these simulations, as well as the associated results, are summarized below. We also performed an analysis on an empirical data set relating to polycystic ovary syndrome (PCOS) from the NCBI databases.

3.4.1 Simulation 1

In this simulation, we will be observing 200 samples ($n = 200$) of 1000 covaraiates ($p = 1000$). Of these p-many covariates, only the interaction between X_1 and X_2 will be considered to have meaningful contribution to the outcome Y. We will run 100 replications and report the average (mean) ranking and the median ranking of the interaction between X_1 and X_2 as it relates to association with Y. The test data is the same for both JCIS and the GPC method of [43].

We generate all X_j randomly from the set $\{0, 1\}$, with each outcome being equally likely. We then let

$$Y = X_1 \times X_2.$$

This will mean that Y depends on only the interaction between X_1 and X_2. Note that we omit any main effects to exhibit the robustness of JCIS even in the absence of main effects on the response. The results for Simulation 1 are summarized in Table 3.1 below.

Table 3.1: Mean and Median Ranking of Interaction Between X_1 and X_2 in Simulation 1

	JCIS	GPC
Mean Rank of (X_1, X_2)	1	2104.5
Median Rank of (X_1, X_2)	1	1306

Note that GPC fails prodigiously to establish the importance of the interaction between X_1 and X_2 on the response Y. Both the average and the median rankings of (X_1, X_2) by GPC are much too large for GPC to be considered a reliable feature screening approach in this case. On the other hand, our JCIS method accurately ranks (X_1, X_2) as being the most important interaction in relation to the response in each of the 100 replicates.

3.4.2 Simulation 2

This simulation closely resembles the interaction simulation found in [43]. Here, we assume that Y only has two levels. (The original simulation assumes Y has four levels). We

also retain the assumption in the aforementioned simulation of [43] that each X_j is binary. It should be noted that, while GPC performs admirably in the original simulation of [43], that simulation is apparently dependent on first identifying a small set of relevant main effects, something that we do not do here. This demonstrates one marked benefit of JCIS over GPC: No predetermined set of predictors is required to obtain accurate results. This holds true whether important causative main effects exist or not. To be specifically clear, GPC does not explicitly require obtaining main effects first. This means that their method *in theory* works for cases where the marginal effects are weak and the interactive effects are strong. However, the results of this simulation here will show that the GPC algorithm is wildly inaccurate in this case.

We will be observing 200 samples ($n = 200$) of 1000 covariates ($p = 1000$). First, we generate a response vector Y, where $Y = 0$ or $Y = 1$ and $\mathbb{P}(Y_i = 1) = 0.75$. Next, we generate $X_{ij} \in \{0, 1\}$ for $j = 1, 3, 5, 7$ as follows:

- Conditional on $Y_i = k$, let $\mathbb{P}(X_{ij} = 1 | Y_i = k) = \theta_{kj}$, where θ_{kj} is given in Table 3.2.

Table 3.2: θ_{kj} Values for Simulation 2

	j			
θ_{kj}	1	3	5	7
$k = 0$	0.3	0.4	0.5	0.3
$k = 1$	0.95	0.9	0.9	0.95

- Given Y_i and $X_{i,2m-1}$ (for $m = 1, 2, 3, 4$), we generate $X_{i,2m} \in \{0, 1\}$ using the following probabilities:

$$\mathbb{P}(X_{i,2m} = 1 | Y_i = k, X_{i,2m-1} = 0) = 0.6 I(\theta_{k,2m-1} > 0.5) + 0.4 I(\theta_{k,2m-1} \leq 0.5);$$

$$\mathbb{P}(X_{i,2m} = 1 | Y_i = k, X_{i,2m-1} = 1) = 0.95 I(\theta_{k,2m-1} > 0.5) + 0.05 I(\theta_{k,2m-1} \leq 0.5),$$

where $I(\cdot)$ is the standard indicator function.

- For all remaining covariates (i.e. X_j for $j > 8$), randomly sample the set $\{0, 1\}$ with $\theta_{kj} = 0.5$.

Overall, the causative interactive effects will be (X_1, X_2), (X_3, X_4), (X_5, X_6), and (X_7, X_8). The test data is the same for both JCIS and GPC. We will run 100 replications and report the average (mean) ranking and the median ranking of each these interactions as they relate to association with Y. The results of Simulation 2 are given in 3.3 below.

Table 3.3: Mean and Median Ranking of Causative Interactions in Simulation 2

	JCIS	GPC
Mean Rank of (X_1, X_2)	2.01	7302.73
Median Rank of (X_1, X_2)	2	534
Mean Rank of (X_3, X_4)	3.53	2365.05
Median Rank of (X_3, X_4)	3	42.5
Mean Rank of (X_5, X_6)	4.65	936.65
Median Rank of (X_5, X_6)	4	16.5
Mean Rank of (X_7, X_8)	2.33	6563.83
Median Rank of (X_7, X_8)	2	1083.5

Since it is obviously impossible for every causative interaction to be consistently ranked as the absolute top interaction, any method placing each true interaction on average in the top four or so causative interactions can easily be said to be producing accurate results. However, as has been mentioned previously, we see here the unfortunate over-reliance of GPC on first establishing a small set of relevant main effects in order to produce a dependable set of causative interactions. The average ranking of each causative interaction by GPC does not lend to confidence in being able to select via GPC the true interactions with any degree of consistency. Although the median rank of each interaction by GPC is better (and even decent in the case of (X_5, X_6)) than the average respective rank by GPC, the reliability of the method is, on the whole, questionable.

3.4.3 Simulation 3

Simulation 3 is similar in form to Simulation 1. However, we now will test the ability of JCIS to screen for interactions when the covariates are continuous. We will be observing 200 samples ($n = 200$) of 1000 covariates ($p = 1000$). Of these p-many covariates, only the interaction between X_1 and X_2 and the interaction between X_3 and X_4 will be considered to have meaningful contribution to the outcome Y.

We generate all X_j randomly based on repeated random samples of the normal distribution with mean 0 and standard deviation 2:

$$X_j \overset{\text{i.i.d.}}{\sim} N(\mu = 0, \sigma = 2).$$

We then let

$$Y = X_1 \times X_2 + X_3 \times X_4.$$

Note that this simulation will also be testing the ability of JCIS to correctly locate multiple two-way interactions having an effect on the response. We will report the percentage of replicates (out of 100) where the interactions (X_1, X_2) and (X_3, X_4) are individually within the top five interactions detected, as well as the percentage of time that *both* (X_1, X_2) and (X_3, X_4) are simultaneously within the top five interactions detected. The test data is the same for both JCIS and the iFORM method of [38]. The results of Simulation 3 are detailed in Table 3.4.

Table 3.4: Percentage of Replicates Finding (X_{j_1}, X_{j_2}) to be Important in Simulation 3

	JCIS	iFORM
(X_1, X_2)	100%	0%
(X_3, X_4)	100%	0%
(X_1, X_2) & (X_3, X_4)	100%	0%

Here an interaction is considered to be "important" if it is ranked in the top five most

relevant interactions by the screening method in question. These results show the remarkable difference between JCIS and iFORM in being able to determine interactive effects in the event that no main effects are prevalent in the data. This demonstrates one larger limitation of iFORM in that it requires the existence of main effects between covariates in order to find any meaningful interactive effects. This is especially important when one wants to screen for interactions in genetic data, where gene SNPs with weak marginal effects can have stronger interactive effects on the response. For further discussion on this, see e.g. [59] and [79].

3.4.4 Simulation 4

In this simulation we will test the ability of JCIS versus iFORM in successfully screening two interactive features in the presence of individual main effects among those covariates forming the interactive effects. This is done in order to show that even when strong marginal effects are present, JCIS can outperform iFORM in determining the true interactions. We examine 100 observations ($n = 100$) of 500 covariates ($p = 500$) over 100 replications. Let X follow the multivariate normal distribution with mean vector $\mathbf{0}$ and $\text{cov}(X_{j_1}, X_{j_2}) = 0.1^{|j_1 - j_2|}$ for $1 \leq j_1, j_2 \leq p$. Now define

$$Y = X_1 + X_3 + X_6 + X_{10} + 3(X_1 \times X_3) + 3(X_6 \times X_{10}).$$

We will apply JCIS and iFORM to screen for the true interactions (X_1, X_3) and (X_6, X_{10}). The results of Simulation 4 are given in Table 3.5.

Table 3.5: Percentage of Replicates Finding (X_{j_1}, X_{j_2}) to be Important in Simulation 4

	JCIS	iFORM
(X_1, X_3)	92%	21%
(X_6, X_{10})	92%	19%
(X_1, X_3) & (X_6, X_{10})	84%	9%

Here an interaction is considered to be "important" if it is ranked in the top five most

relevant interactions by the screening method in question. Note that even when marginal effects are included in the generation of the response, iFORM still struggles to accurately and consistently detect the true interactive effects. Note that JCIS, on the other hand, accurately detects at least one of the two true interactions in every replication, and detects *both* true interactions in 84 of the 100 replications.

3.4.5 Final Comments on Simulation Results

An overall issue that arises in interaction feature screening is the reliance of extant methods on a predetermined set of marginally important predictors. Simulations 1 and 3 demonstrate this shortcoming in even very simple cases. The first and third simulations lead us to believe that, in the absence of strong main effects, JCIS is a much superior method to GPC and iFORM. Simulations 2 and 4 add main effects to the simulation data model. However, even with the presence of main effects, both GPC and iFORM do not produce competent or reliable results. Again, as with the first and third simulations, JCIS performs admirably.

3.4.6 Real Data Analysis: Epistasis Detection

We apply a two-stage process to a real data set examining prevalence of polycystic ovary syndrome (PCOS) in females who self-identified as having Caucasian or European-ancestry. With the proper approvals, this PCOS dataset was downloaded from the database of genotypes and phenotypes (dbGaP) of the National Center for Biotechnology Information (NCBI) at the NIH (dbGaP Study Accession: phs000368.v1.p1). This data consists of 4099 (3055 controls, 1042 cases) observations of each of 731,442 SNPs. The response is PCOS affection status (0 = control, 1 = case) and the predictors are the encoded SNP geneotype values. Our specific aim is to identify SNPs which most strongly interact with one another in determining PCOS affection status.

Stage 1 analysis

In the first stage of our interaction feature screening, we apply JCIS to all pairwise

combinations of SNPs coming from the same chromosome. All 23 *homo sapien* chromosomes were used. As is common with such data sets, we removed all SNPs with less than a 95% call rate, as well as all SNPs with a minor allele frequency less than 10%. (See [2], [51], [66]). All of the analysis in this stage was performed on the cluster machine centered at the Center for High Performance Computing at the University of Utah. After recording a $\widehat{R}_{j_1,j_2}$ value for all possible within-chromosome pairs, we ordered the SNP pairs from largest to smallest $\widehat{R}_{j_1,j_2}$ value. To ensure that all important SNP pairs are selected after the first stage, we keep all SNPS associated with the $n = 4099$ largest values of $\widehat{R}_{j_1,j_2}$. We then proceed to Stage 2 of the real data analysis.

It should here be noted that while an exhaustive search among *all* pairs of SNPs (including between-chromosome pairings) can be done, preliminary results on all possible between-chromosome pairs of SNPs from chromosomes 11 through 23 indicated that approximately 300,000 within-chromosome SNP pairings (SNP pairs coming from the same chromosome) had a stronger interactive effect on PCOS status than even the top ranked between-chromosome SNP pair. Further examination as to why this is could be pursued at a later date.

Stage 2 analysis

We now turn our attention to a more in depth analysis using multifactor dimensionality reduction (MDR) on a small set of the SNPs comprising SNP pairs having the largest values of $\widehat{R}_{j_1,j_2}$. MDR is a model-free and nonparametric approach first introduced in [68] that can be used to identify high-order SNP-SNP interactions, even in the absence of independent main effects of the gene SNPs on the outcome. Ideally, we would like to select a set of SNPs associated with the n largest $\widehat{R}_{j_1,j_2}$ values. However, as discussed in much of the MDR literature (e.g. [68], [36], [84], [32]), the run time of MDR increases drastically as the number of SNPs under consideration grows. Thus, we must choose a relatively small set of SNPs to consider for analysis by MDR.

In order to select a reasonably small set of SNPs to consider (approximately 100 candidate SNPs), we can project onto the position of SNP_{j_1} the top 10,000 $\widehat{R}_{j_1,j_2}$ values, then look

for a cutoff value for which most of the SNPs lie below. We only plot the top 10,000 $\widehat{R}_{j_1,j_2}$ values due to computational limitations in plotting a larger set of values. Approximately 99% of the $\widehat{R}_{j_1,j_2}$ values are less than 0.1. This tells us that the vast majority of SNP pairs can be omitted as having little to no effect on PCOS status. The top 10,000 SNPs still easily provide a set of SNPs encompassing the overall patterns of the $\widehat{R}_{j_1,j_2}$ values. Figure 3.1 shows the top 10,000 $\widehat{R}_{j_1,j_2}$ values versus the associated position of SNP_{j_1}. We will look for a cutoff value for which (approximately) less than 100 $\widehat{R}_{j_1,j_2}$ values lie above. Based on

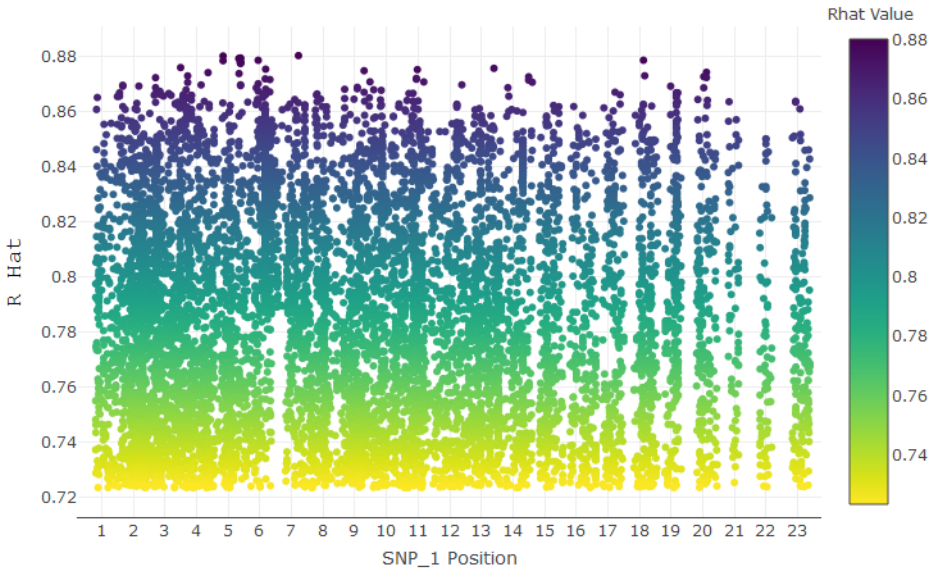

Fig. 3.1: SNP_{j_1} position versus $\widehat{R}_{j_1,j_2}$ value.

the plot in Figure 3.1, we select a cutoff of $\widehat{R}_{j_1,j_2} = 0.865$. This yields a computationally feasible set of 85 SNPs for our candidate set for MDR.

Because our case to control ratio is unbalance (i.e. not equal to 1), we will use balanced accuracy (BA) as the evaluation measure of our MDR results. The BA can be defined as follows:

$$BA = \frac{1}{2}\left(\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} + \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}\right),$$

where the true and false positives and negatives refer to the classification of a subject based

on the loci-genotype combinations selected by MDR. Note that BA is just the arithmetic mean of the specificity and the sensitivity. For further discussion on the use of the BA as the metric for our model evaluations in the presence of unbalanced case to control ratios see, for example, [81] and [84].

A further consideration that must be made in regards to the unbalanced ratio of case to control PCOS instances is that of choosing a threshold T at which to classify subjects as high or low risk for PCOS based on their genotype combination among the SNPs selected by MDR. While traditional approaches tacitly assume *a priori* balance of the case to control ratio (either by design or by over/undersampling of the case/control observations), and take $T = 1$ as the threshold, more robust implementations of MDR allow for an adjusted threshold T_{adj}, where T_{adj} is the case to control ratio. This use of the adjusted threshold is seen commonly in the MDR literature (e.g. [81] and [32]).

Using an implementation of MDR in Java from the researchers at www.epistasis.org (see also [64] and [34], both of which recommend this implementation), we obtained the following two, three, four, and five-loci results. Table 3.6 contains the 10-fold cross validated accuracies (BA-wise) for each model.

- Two-loci model: `rs1423304`, `rs1024216`.
- Three-loci model: `rs1002424`, `rs1423304`, `rs1024216`.
- Four-loci model: `rs1002424, rs1024216, rs657718, rs4745466`.
- Five-loci model: `rs1002424, rs1423304, rs1024216, rs657718, rs4745466`.

All models found via this MDR implementation employ the ensemble of BA to test model accuracy, 10-fold cross validation to prevent overfitting, and the adjusted threshold outlined by [81]. Higher order models can also be found, however, with greater balanced accuracy comes an exponentially increasing computational cost. The plots given in Figures 3.2 and 3.3 show the classification of geneotype combinations in the two- and three-loci models as either high or low risk for PCOS. Higher order models exceed the limitations of succinct plotting and are omitted.

Table 3.6: MDR Accuracy

k-way	CV Accuracy
2-way	53.50%
3-way	52.69%
4-way	51.97%
5-way	53.26%

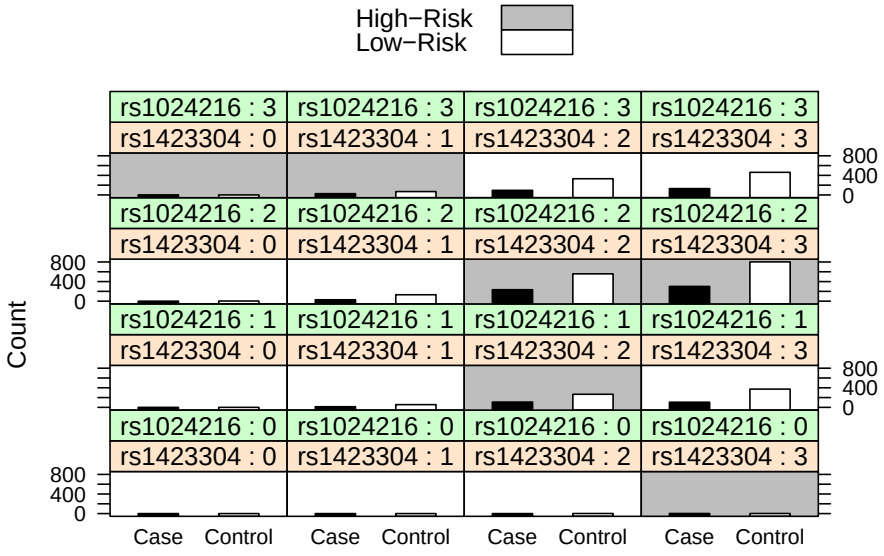

Fig. 3.2: High-low risk bar plots broken down by genotype for the selected two-locus model.

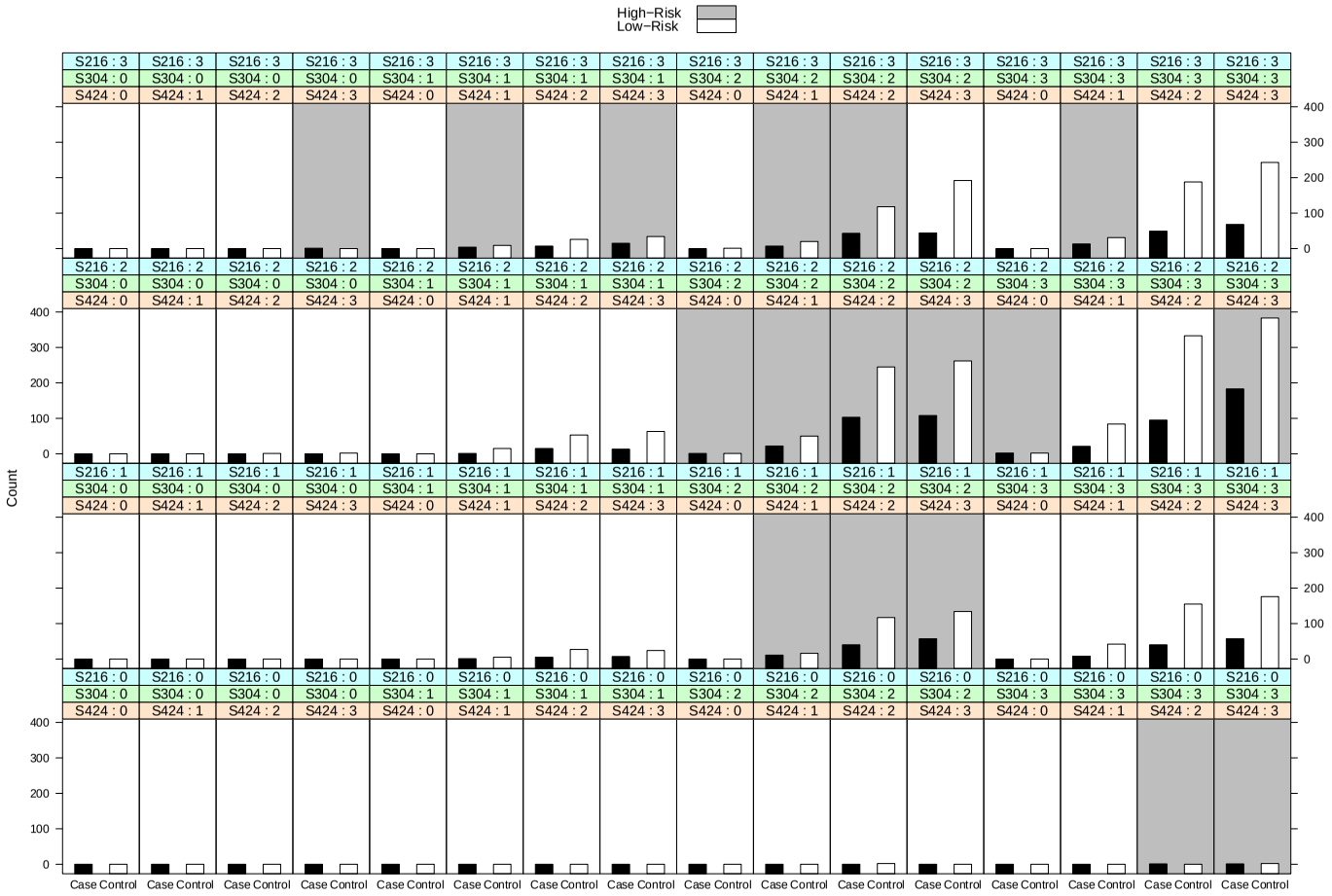

Fig. 3.3: High-low risk bar plots broken down by genotype for the selected three-locus model. Here "S—" indicates the selected SNP ending in the given three digits.

3.5 Concluding Remarks

In this paper we have addressed the important issue of interaction screening in ultrahigh dimensional feature spaces. Although applications of interaction screening are wide spread, few extant methods exist for doing such. We have introduced a novel interaction screening method called JCIS (Joint Cumulant Interaction Screening) that is empirically accurate, theoretically sound, and computationally feasible.

One unrivaled advantage of JCIS when compared to existing interaction screening methods such as iFORM [38] and GPC [43] is the ability of JCIS to determine interactive effects among predictors even when no no strong marginal effects exist. Extant methods for feature interaction screening are deleteriously over-reliant on the existance of pronounced and explicit main effects from both an empirical and theoretical standpoint. The superiority of JCIS in this regard is born out repeatedly in the simulations of Section 3.4.

Our proposed method also has the strong sure screening consistency property, meaning that even as the number of covariates increases exponentially with respect to sample size, JCIS prevails in discovering the exact set of relevant features with probability approaching one. This property has become the benchmark theoretical property for feature screening methods. The proofs pertaining to strong sure screening of JCIS are presented in Section 3.6 found below.

Via a real data analysis on an empirical data set relating to polycystic ovary syndrome (PCOS) from the NCBI databases, we demonstrated the ability of our method to be applied to extremely large real life data sets such as those found in genetics. In terms of number of covariate pairs in question, as well as the number of observations considered, the real data set we examine is (by conservative estimates) about 2800 times larger than the data sets examined in similar papers (e.g. the inbred mouse microarray gene expression dataset found in [38]). In turn, this means that the computational considerations necessary for our PCOS data were, until now, unseen in the setting of interaction feature screening. As both data dimension and computational power continue to grow, we feel confident that JCIS will remain a salient approach for analyzing two-way interactions in ultrahigh (and beyond)

dimensional data sets.

3.6 Proofs of Theoretical Results

Here we present in full the proofs for Theorems 3.3.1 and 3.3.2. Before proceeding into the proofs, we will establish a lemma which employs the Continuous Mapping Theorem (see [58] and [11]).

3.6.1 Prefacing Lemmas and a Definition

The following lemmas will assist in the proof of our main theorems on strong sure screening.

Lemma 3.6.1. *Let $\hat{\sigma}_{j_1}$, $\hat{\sigma}_{j_2}$, and $\hat{\sigma}_Y$ be the estimators of σ_{j_1}, σ_{j_2}, and σ_Y used in the definition of $\widehat{R}_{j_1,j_2}$. Assume that $\hat{\sigma}_{j_1}$, $\hat{\sigma}_{j_2}$, $\hat{\sigma}_Y$, and $\hat{\tau}_{j_1,j_2}$ are all (individually speaking) consistent estimators of the respective values they are estimating (viz. σ_{j_1}, σ_{j_2}, σ_Y, and $\kappa_3(X_{j_1}, X_{j_2}, Y)$). We then have that*

$$\widehat{R}_{j_1,j_2} = \frac{\hat{\tau}_{j_1,j_2}}{\hat{\sigma}_{j_1}\hat{\sigma}_{j_2}\hat{\sigma}_Y}$$

is a consistent estimator of R_{j_1,j_2}.

Proof. The proof of this lemma follows easily from a direct application of the Continuous Mapping Theorem paired with a straight forward generalization of the very similar proof found in Chapter 2. □

Classical results

It is a classical result that

$$\hat{\sigma}_{j_1} = \frac{1}{n}\sum_{i=1}^{n}\left(X_{ij_1} - \bar{X}_{j_1}\right)^2$$

is a biased, yet consistent, estimator of the standard deviation of X_{j_1}. Similar statements can of course be made for $\hat{\sigma}_{j_2}$ and $\hat{\sigma}_Y$.

The following definition introduces some necessary concepts from the field of set topology. For readers interested in further background on this topic, we suggest the texts [9] and [57].

Definition 3.6.2. Unless otherwise noted, from hereon let D be a directed set and let $f : D \to \mathbb{R}$ be any well define function. Denote by $\mathbb{F}(D)$ the collection of all finite subsets of D. Define a function $h : \mathbb{F}(D) \to \mathbb{R}$ as follows:

$$h(A) := \sum_{a \in A} f(a),$$

with $A \in \mathbb{F}(D)$. Note that $\mathbb{F}(D)$ is partially ordered by set inclusion. Moreover, since for any A, B in $\mathbb{F}(D)$, we have $A \subseteq A \cup B$ and $B \subseteq A \cup B$, then $\mathbb{F}(D)$ is itself a directed set. (Obviously $A \cup B$ is in $\mathbb{F}(D)$ as the finite union of finite sets is also finite). Because $\mathbb{F}(D)$ is a directed set, then h is a topological net on $\mathbb{F}(D)$ into $\mathbb{R}$. The function f is defined to be *summable* if the net h converges in the usual (Moore-Smith) sense. The limit of h is the sum of f over D.

Lemma 3.6.3. *If a function $f : D \to \mathbb{R}$ is summable in the sense of Definition 3.6.2, then the set $\{d \in D : f(d) > 0\}$ is a countable subset of D.*

Proof. Assume that f is summable. Let

$$\sum_{d \in D} f(d) = M < \infty.$$

Define the following sets for each $n \in \mathbb{N}$:

$$S_n := \left\{ d \in D : f(d) > \frac{1}{n} \right\}.$$

We then have the following chain of inequalities for any $n \in \mathbb{N}$:

$$M \geq \sum_{d \in S_n} f(d) > \sum_{d \in S_n} \frac{1}{n} = \frac{|S_n|}{n},$$

where $|S_n|$ denotes the cardinality of S_n. Note that by necessity $|S_n| < \infty$, as otherwise we would have that $\sum_{d \in D} f(d) = M = \infty$ and f would not be summable. All told, we can say that $Mn \geq |S_n|$, implying that for every $n \in \mathbb{N}$, S_n is a finite set. Thus

$$\{d \in D : f(d) > 0\} = \bigcup_{n \in \mathbb{N}} S_n$$

is a countable union of finite sets, the result of which must necessarily be countable. □

Note that the same argument can be used *mutatis mutandis* to show that $\{d \in D : f(d) < 0\}$ is also countable. This allows us to now state a corollary.

Corollary 3.6.4. *Lemma 3.6.3 means that, when $f : D \to \mathbb{R}$ is summable, there exists a countable set $D' \subseteq D$ such that*

$$\sum_{d \in D} f(d) = \sum_{d' \in D'} f(d') < \infty.$$

Proof. Define the following two sets:

$$D_1 = \{d \in D : f(d) > 0\} \qquad D_2 = \{d \in D : f(d) < 0\}.$$

Begin by noting that we can apply Lemma 3.6.3 twice to get that the set $D' = \{d \in D : f(d) \neq 0\}$ is a countable subset. This definition of D' yields the desired result. □

We can now proceed into the proofs of our main theorems on sure screening.

3.6.2 Proofs of Theorems 3.3.1 and 3.3.2

The proof of these two theorems is accomplished in three steps:

1. We first show that a positive lower bound $R_{\min}$ exists for all R_{j_1,j_2} with $(j_1, j_2) \in \mathcal{S}_T$. In other words, we will show the following:

$$\text{There exists } \; R_{\min} > 0 \text{ such that } R_{j_1,j_2} > R_{\min} \; \text{ for all } \; (j_1, j_2) \in \mathcal{S}_T.$$

2. This is followed by our showing that $\widehat{R}_{j_1,j_2}$ is a uniformly consistent estimator of R_{j_1,j_2} for each $1 \leq j_1 < j_2 \leq p$. This will effectually consist of showing that $\hat{\tau}_{j_1,j_2}$ is a consistent estimator of $\kappa_3(X_{j_1}, X_{j_2}, Y)$, since the standard deviation estimators in the denominator of $\widehat{R}_{j_1,j_2}$ are already well established consistent estimators of the standard deviations of X_{j_1}, X_{j_2}, and Y.

3. We finally show that there exists a constant $c > 0$ such that

$$\mathbb{P}(\widehat{\mathcal{S}} = \mathcal{S}_T) \longrightarrow 1 \text{ as } n \longrightarrow \infty$$

Weak consistency is shown as a natural subcase of this, which will establish Theorem 3.3.2.

Step 1

We previously defined the following in Subsection 3.3.3:

$$\omega_{j_1,j_2}(k_1, k_2, m) \quad = \quad (k_1 - \mathbb{E}X_{j_1})(k_2 - \mathbb{E}X_{j_2})(m - \mathbb{E}Y)\pi_{j_1,j_2,Y}(k_1, k_2, m).$$

Taking any fixed X_{j_1} and X_{j_2}, let $f : \mathbb{R}^3 \to \mathbb{R}$ be defined by

$$f(k_1, k_2, m) = \frac{\omega_{j_1,j_2}(k_1, k_2, m)}{\sigma_{j_1}\sigma_{j_2}\sigma_Y}.$$

Clearly $\mathbb{R}^3$ is a directed set under the routine product direction. Furthermore, if we use f to define the net h as in Definition 3.6.2, then the limit of h is R_{j_1,j_2}. Since by Condition (C4) R_{j_1,j_2} is finite for all X_{j_1} and X_{j_2} pairs, then this in turn implies that f is a summable function. We have thus satisfied the conditions of Corollary 3.6.4, which means that there

is some countable set $D_{j_1,j_2} \subset \mathbb{R}^3$ such that

$$\begin{aligned} R_{j_1,j_2} &= \int_{D_{j_1,j_2}} f(k_1,k_2,m)\; dk_1 dk_2 dm \\ &= \int_{D_{j_1,j_2}} \frac{\omega_{j_1,j_2}(k_1,k_2,m)}{\sigma_{j_1}\sigma_{j_2}\sigma_Y}\; dk_1 dk_2 dm. \end{aligned}$$

Hence, for $(j_1, j_2) \in \mathcal{S}_T$,

$$\begin{aligned} R_{j_1,j_2} &= \int_{D_{j_1,j_2}} \frac{\omega_{j_1,j_2}(k_1,k_2,m)}{\sigma_{j_1}\sigma_{j_2}\sigma_Y}\; dk_1 dk_2 dm \\ &\geq \frac{1}{\sigma^3_{\max}} \int_{D_{j_1,j_2}} \omega_{j_1,j_2}(k_1,k_2,m)\; dk_1 dk_2 dm \quad \text{by (C2)}, \\ &\geq \frac{1}{\sigma^3_{\max}} \sup_{(k_1,k_2,m)\in D_{j_1,j_2}} \omega_{j_1,j_2}(k_1,k_2,m) \\ &\geq \frac{\omega_{\min}}{\sigma^3_{\max}} \quad \text{by (C3)}, \\ &> 0. \end{aligned}$$

Define $R_{\min} = \dfrac{\omega_{\min}}{2\sigma^3_{\max}}$. Then $R_{j_1,j_2} > R_{\min} > 0$ for all $(j_1, j_2) \in \mathcal{S}_{\mathcal{T}}$. This establishes a positive lower bound on R_{j_1,j_2} for all $(j_1, j_2) \in \mathcal{S}_{\mathcal{T}}$, completing Step 1. Corollary 3.3.3 is also established by this step.

Step 2

We now apply the weak law of large numbers to show that $\widehat{R}_{j_1,j_2}$ is a (uniformly) consistent estimator of R_{j_1,j_2}. This will consist of showing that $\hat{\tau}_{j_1,j_2}$ is a consistent estimator of $\kappa_3(X_{j_1}, X_{j_2}, Y)$, since the denominator of $\widehat{R}_{j_1,j_2}$ is comprised of the routine (and, importantly here, consistent) estimators of σ_{j_1}, σ_{j_2} and σ_Y. As it can be show using the Mann-Wald Theorem that the quotient of consistent estimators is itself a consistent estimator, our aforementioned work with $\hat{\tau}_{j_1,j_2}$ will suffice. This is reflected in the statement of Lemma 3.6.1.

By a slight rearrangement of the numerator in the definition of $\widehat{R}_{j_1,j_2}$, we can obtain

$$\hat{\tau}_{j_1,j_2} = \frac{1}{n}\sum_{i=1}^{n}(X_{ij_1} - \overline{X}_{j_1})(X_{ij_2} - \overline{X}_{j_2})(Y_i - \overline{Y}). \tag{3.1}$$

We now can explicitly expand the product of binomials in (3.1) to obtain

$$\begin{aligned}\hat{\tau}_{j_1,j_2} &= \frac{1}{n}\sum X_{ij_1}X_{ij_2}Y_i - \frac{1}{n}\sum \overline{X}_{j_1}X_{ij_2}Y_i - \frac{1}{n}\sum X_{ij_1}\overline{X}_{j_2}Y_i \\ &\quad -\frac{1}{n}\sum X_{ij_1}X_{ij_2}\overline{Y} + \frac{1}{n}\sum \overline{X}_{j_1}\overline{X}_{j_2}Y_i \\ &\quad +\frac{1}{n}\sum \overline{X}_{j_1}X_{ij_2}\overline{Y} + \frac{1}{n}\sum X_{ij_1}\overline{X}_{j_2}\overline{Y} - \frac{1}{n}\sum \overline{X}_{j_1}\overline{X}_{j_2}\overline{Y}\end{aligned}$$

By repeated applications (summand wise) of the weak law of large numbers to this above expression for $\hat{\tau}_{j_1,j_2}$, we obtain:

$$\frac{1}{n}\sum X_{ij_1}X_{ij_2}Y_i \xrightarrow{p} \mathbb{E}(X_{j_1}X_{j_2}Y)$$

$$\frac{1}{n}\sum \overline{X}_{j_1}X_{ij_2}Y_i \xrightarrow{p} \mathbb{E}(X_{j_1})\mathbb{E}(X_{j_2}Y)$$

$$\frac{1}{n}\sum X_{ij}\bar{Y} \xrightarrow{p} \mathbb{E}(X_j)\mathbb{E}(Y),$$

with all convergence being in probability. Similar conclusions can be reached for like terms.

Hence we have

$$\begin{aligned}
\hat{\tau}_{j_1,j_2} &= \frac{1}{n}\sum X_{ij_1}X_{ij_2}Y_i - \frac{1}{n}\sum \overline{X}_{j_1}X_{ij_2}Y_i - \frac{1}{n}\sum X_{ij_1}\overline{X}_{j_2}Y_i \\
&\quad -\frac{1}{n}\sum X_{ij_1}X_{ij_2}\overline{Y} + \frac{1}{n}\sum \overline{X}_{j_1}\overline{X}_{j_2}Y_i \\
&\quad +\frac{1}{n}\sum \overline{X}_{j_1}X_{ij_2}\overline{Y} + \frac{1}{n}\sum X_{ij_1}\overline{X}_{j_2}\overline{Y} - \frac{1}{n}\sum \overline{X}_{j_1}\overline{X}_{j_2}\overline{Y} \\
&\xrightarrow{p} \mathbb{E}(YX_{j_1}X_{j_2}) - \mathbb{E}(YX_{j_1})\mathbb{E}X_{j_2} - \mathbb{E}(YX_{j_2})\mathbb{E}X_{j_1} \\
&\quad - \mathbb{E}(X_{j_1}X_{j_2})\mathbb{E}Y + 2\mathbb{E}Y\mathbb{E}X_{j_1}\mathbb{E}X_{j_2} \\
&= \kappa_3(X_{j_1}, X_{j_2}, Y).
\end{aligned}$$

So indeed $\hat{\tau}_{j_1,j_2}$ is a consistent estimator of $\kappa_3(X_{j_1}, X_{j_2}, Y)$. Furthermore, this shows, by Lemma 3.6.1, that $\widehat{R}_{j_1,j_2}$ is a consistent estimator of R_{j_1,j_2}.

We will now show that such consistency is also uniform. Since $\widehat{R}_{j_1,j_2}$ is consistent as an estimator of R_{j_1,j_2}, we know that for any $1 \le j_1 < j_2 \le p$ and any $\varepsilon > 0$,

$$\mathbb{P}(|\widehat{R}_{j_1,j_2} - R_{j_1,j_2}| > \varepsilon) \to 0 \quad \text{as } n \to \infty.$$

Let

$$(J_1, J_2) = \text{argmax}_{1\le j_1<j_2\le p} |\widehat{R}_{j_1,j_2} - R_{j_1,j_2}|.$$

Then, since $(J_1, J_2) \in \{1, 2, \ldots, p\} \times \{1, 2, \ldots, p\}$, we know that

$$\mathbb{P}(|\widehat{R}_{J_1,J_2} - R_{J_1,J_2}| > \varepsilon) \to 0 \quad \text{as } n \to \infty$$

for any $\varepsilon > 0$. In other words, we have that

$$\mathbb{P}\left(\max_{1\leq j_1<j_2\leq p} |\widehat{R}_{j_1,j_2} - R_{j_1,j_2}| > \varepsilon\right) \to 0 \quad \text{as } n \to \infty$$

for any $\varepsilon > 0$. This shows that $\widehat{R}_{j_1,j_2}$ is a *uniformly* consistent estimator of R_{j_1,j_2}, completing Step 2. This also establishes Corollary 3.3.4.

Step 3

In Step 1 we defined

$$R_{\min} = \frac{\omega_{\min}}{2\sigma^3_{\max}}.$$

Let $c = (2/3)R_{\min}$. Suppose by way of contradiction that this c is insufficient to be able to claim $\widehat{\mathcal{S}} \supseteq \mathcal{S}_T$. This would mean that there exists some pair $(j_1^*, j_2^*) \in \mathcal{S}_T$, yet $(j_1^*, j_2^*) \notin \widehat{\mathcal{S}}$. It then follows that we must have

$$\widehat{R}_{j_1^*,j_2^*} \leq (2/3)R_{\min}$$

while at the same time having (as shown in Step 1)

$$R_{j_1^*,j_2^*} > R_{\min}.$$

From this we can conclude that

$$|\widehat{R}_{j_1^*,j_2^*} - R_{j_1^*,j_2^*}| > (1/3)R_{\min},$$

which implies that

$$\max_{1\leq j_1<j_2\leq p} |\widehat{R}_{j_1,j_2} - R_{j_1,j_2}| > (1/3)R_{\min}$$

as well. However, we know by the uniform consistency of $\widehat{R}_{j_1,j_2}$ that by letting $\varepsilon = 1/3R_{\min}$, we have

$$\mathbb{P}(\widehat{\mathcal{S}} \not\supseteq \mathcal{S}_T) \leq \mathbb{P}\left(\max_{1\leq j_1<j_2\leq p} |\widehat{R}_{j_1,j_2} - R_{j_1,j_2}| > (1/3)R_{\min}\right) \to 0 \quad \text{as } n \to \infty.$$

This is a contradiction to the assumption of non containment above. So indeed, we have that

$$\mathbb{P}(\widehat{\mathcal{S}} \supseteq \mathcal{S}_T) \to 1 \quad \text{as } n \to \infty.$$

This proves Theorem 3.3.2, and also establishes the forward direction for the statement of Theorem 3.3.1.

To prove the reverse direction for Theorem 3.3.1, suppose (again by way of contradiction) that $\widehat{\mathcal{S}} \not\subseteq \mathcal{S}_T$. Then there is some $(j_1^*, j_2^*) \in \widehat{\mathcal{S}}$, yet $(j_1^*, j_2^*) \notin \mathcal{S}_T$. This means that

$$\widehat{R}_{j_1^*,j_2^*} \geq (2/3)R_{min},$$

while at the same time (by (C4)) having

$$R_{j_1^*,j_2^*} = 0.$$

It now follows that

$$|\widehat{R}_{j_1^*,j_2^*} - R_{j_1^*,j_2^*}| > (2/3)R_{\min}.$$

Set $\varepsilon = (2/3)R_{\min}$. By uniform consistency we have

$$\mathbb{P}(\mathcal{S}_T \not\supseteq \widehat{\mathcal{S}}) \leq \mathbb{P}\left(\max_{1\leq j_1<j_2\leq p} |\widehat{R}_{j_1,j_2} - R_{j_1,j_2}| > (2/3)R_{\min}\right) \to 0 \quad \text{as } n \to \infty.$$

From this we know that

$$\mathbb{P}(\mathcal{S}_T \supseteq \widehat{\mathcal{S}}) \to 1 \quad \text{as } n \to \infty.$$

We can now conclude that for $c = (2/3)R_{\min}$, we have $\mathbb{P}(\mathcal{S}_T = \widehat{\mathcal{S}}) \to 1$ as $n \to \infty$, completing the proof.

CHAPTER 4

Marginal and Interactive Feature Screening for Ultrahigh Dimensional Data with Multivariate Response

4.1 Introduction

Variable selection and feature screening methods for high and ultrahigh dimensional data sets have been an oft explored topic in fields such as regression modelling, machine learning, and classification. Early approaches such as lasso [76], SCAD [22], adaptive-lasso [95], and elastic net [96] focused on penalized regularization regression. These methods were followed by various implementations of sure independence screening, as pioneered by [24]. [See also e.g. 18, 25, 26, 43, 53, 93]. While the sole or principal focus of these feature screening methods is the case when the response is univariate, we are interested here in developing a new approach for feature screening when the feature space is ultrahigh dimensional and the response is multivariate. The ability to feature screen ultrahigh dimensional feature spaces when the response is multivariate can allow us to develop more accurate classification and regression models because we can account for the covariance structure between the components of the response jointly. The ideal multivariate screening method would possess the ability to screen for both marginal and interactive effects.

Throughout this paper we will let n represent the number of observations, p represent the number of covariates, and q represent the number of components in the response vector $\mathbb{Y}$. We will tacitly assume throughout this work that the feature space is ultrahigh dimensional in the classical sense of [24]. Consider the multivariate regression model below:

$$h(\mathbb{Y}) = XB,$$

where $\mathbb{Y} \in \mathbb{R}^{n \times q}$ is the observed $n \times q$ matrix of responses, $X \in \mathbb{R}^{n \times p}$ is the covariate or predictor matrix, $B \in \mathbb{R}^{p \times q}$ is the parameter coefficient matrix, and h is a link function.

The j^{th} row vector of B, B_j, is a vector of length q corresponding to the coefficient vector of the j^{th} predictor, where $j = 1, ..., p$. For screening purposes, we will assume that the true model is sparse, with few predictors (relative to p) having a causative effect on the response. Our goal with this work is to introduce a feature screening method applicable to situations where $\mathbb{Y}$ is multivariate. Moreover, we will seek to not only establish our method as a viable option when screening for marginal effects in a multivariate response model, but also as the only existing method for *interaction* effects screening when $q > 1$.

[73] presents an approach to multivariate response modelling in the setting of classification using microarray data. While a tractable method for ultrahigh dimesional feature screening with respect to classification, the method is not generally applicable to broader multivariate response problems. Moreover, their techniques cannot be used to screen for interactions in any regard. [56] considers a feature screening process when the relationship between q and n is ultrahigh dimensional (but $p < n$). This situation is distinct (and opposite) from our assumptions here and will not be considered. Two well known methods for feature screening ultrahigh dimensional feature spaces for marginal effects when the response is multivariate are the SIRS method of [93], and the distance correlation (DC-SIS) method of [53]. [See also e.g., 14, 55, where SIRS and DC-SIS are put forth as the leading extant methods in marginal multivariate feature screening]. It is important to note here, however, that neither SIRS nor DC-SIS allow for the screening of interactive effects. We will present below a new method for feature screening ultrahigh dimensional feature spaces with multivariate response. Our method will be able to screen for both marginal effects and interactive effects. As a matter of comparison, we will compare the empirical results of our method in the marginal case with the results of SIRS and DC-SIS.

The remainder of this paper will be organized as follows: In Section 4.2 we will present our new feature screening approach, along with the necessary notation and theoretical properties associated with this new method. This will be followed by Section 4.3, wherein we empirically compare our newly proposed method with the existing approaches of SIRS and DC-SIS, as well as demonstrate the application of our method in the setting of interaction

screening. Therein we will also present a real data analysis on a genome-wide-association-study (GWAS) data set for mice. A concluding discussion will be given in Section 4.4. Section 4.5 will be devoted to proving the theoretical results of Section 4.2.

4.2 Feature Screening and Generalized Correlation

Here we present an overview of a screening procedure built upon the concepts of the generalized correlation matrix. We first explore a novel process of screening for marginal effects of individual covariates on a multivariate response. This is then followed by the presentation of a yet hereunto unseen method for feature interaction screening when the response is multivariate.

4.2.1 Marginal Feature Screening Via Generalized Correlation

Let Z_1 and Z_2 be univariate random variables. The linear (Pearson) correlation between Z_1 and Z_2 is given by

$$\varrho = \frac{\text{Cov}(Z_1, Z_2)}{\sqrt{\text{Var}(Z_1)\text{Var}(Z_2)}},$$

where the standard definitions of the variance and covariance are used. This concept of correlation can be generalized to greater than two variables in the following manner. Let $\boldsymbol{Z} = [Z_1, Z_2, Z_3, \ldots, Z_r]^T$ be an r-dimensional random variable with any distribution. Let Σ be the standard variance-covariance matrix associated with $\boldsymbol{Z}$. We then can obtain a correlation matrix for $\boldsymbol{Z}$ as follows:

$$\text{corr}(\boldsymbol{Z}) = \left[\text{diag}\left(\Sigma\right)\right]^{-1/2} \Sigma \left[\text{diag}\left(\Sigma\right)\right]^{-1/2}.$$

See Chapter 7 of [49] for a further discussion on the correlation matrix.

This work with the correlation matrix and generalized correlation can be specifically extended to marginal effects in the following manner: For any one of the covariates, X_j, consider the correlation matrix of the random variable vector

$$\mathcal{V}_j = [X_j, Y^{(1)}, Y^{(2)}, \ldots, Y^{(q)}],$$

where $Y^{(m)}$ is the m^{th} component of $\mathbb{Y}$. The variance-covariance matrix of $\mathcal{V}_j$ can be written in block matrix form as below:

$$\Sigma_j = \left(\begin{array}{c|c} \mathrm{Var}(X_j) & \mathcal{C}^T \\ \hline \mathcal{C} & \mathrm{Cov}(\mathbb{Y}) \end{array}\right),$$

where $\mathcal{C}$ is a q-vector whose mth entry is the covariance between X_j and $Y^{(m)}$, and $\mathrm{Cov}(\mathbb{Y})$ represents the variance-covariance matrix for the components of $\mathbb{Y}$. Let $\mathcal{H}_j$ be the generalized correlation for $\mathcal{V}_j$:

$$\mathcal{H}_j = \left[\mathrm{diag}\left(\Sigma_j\right)\right]^{-1/2} \Sigma_j \left[\mathrm{diag}\left(\Sigma_j\right)\right]^{-1/2}.$$

This allows us to now construct a population quantity of a utility measure for covariate ranking. Let $\varphi_j = \|\mathcal{H}_j\|_{\mathfrak{p}}$, with $\|\cdot\|_{\mathfrak{p}}$ being any $\ell_{\mathfrak{p}}$ norm with $1 \leq \mathfrak{p} < \infty$. (Of note, $\|\cdot\|_\infty$ cannot be directly employed here as it will always be equal to one in this case). The $\mathfrak{p}$ used here is not to be confused with the p used to denote the dimension of the covariate space. Herein we will examine empirical results under the taxi-cab (i.e. ℓ_1) and Frobenius (i.e. ℓ_2) matrix norms. Where necessary, these norms will be denoted by $\|\cdot\|_T$ and $\|\cdot\|_F$ respectively. It should be noted, however, that any $\ell_{\mathfrak{p}}$-norm (with $\mathfrak{p} < \infty$) can be employed here from a theoretical standpoint. This will be reflected in the proofs (see Section 4.5) of the underlying theory to follow below. The use of other matrix norms beyond the previously discussed entrywise $\ell_{\mathfrak{p}}$ norms (e.g. induced matrix norms, Schatten norms, $\ell_{\mathfrak{p},\mathfrak{q}}$ norms, etc.) is reserved for a later work and extends beyond the scope of what we will examine in this paper. For further reference on the aforementioned taxi-cab and Frobenius norms, as well as any other $\ell_{\mathfrak{p}}$ and matrix norms, see e.g. [40].

We can create an estimator $\widehat{\varphi}_j$ of each φ_j as follows:

- The variance of each X_j is estimated by

$$\widehat{\mathrm{Var}}(X_j) = \frac{1}{n-1}\sum_{i=1}^{n}(X_{ij} - \overline{X}_j)^2,$$

where $\overline{X}_j$ is the sample mean of X_j.

- Each covariance in $\mathcal{V}$ is estimated by

$$\widehat{\mathrm{Cov}}(Y^{(m)}, X_j) = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_{ij} - \overline{X}_j\right)\left(Y_i^{(m)} - \overline{Y}^{(m)}\right),$$

where $\overline{X}_j$ is still the sample mean of X_j and $\overline{Y}^{(m)}$ is the sample mean of $Y^{(m)}$.

- The covariance matrix of $\mathbb{Y}$ is to be estimated in the usual way, where

$$\widehat{\mathrm{Cov}}\left(Y^{(\ell)}, Y^{(m)}\right) = \frac{1}{n-1}\sum_{i=1}^{n}\left(Y_i^{(\ell)} - \overline{Y}^{(\ell)}\right)\left(Y_i^{(m)} - \overline{Y}^{(m)}\right).$$

- This process ultimately results in the ability to create an estimator $\widehat{\Sigma}_j$ of the matrix Σ_j. We can then in turn create the following matrix:

$$\widehat{\mathcal{H}}_j = \left[\mathrm{diag}\left(\widehat{\Sigma}_j\right)\right]^{-1/2}\widehat{\Sigma}_j\left[\mathrm{diag}\left(\widehat{\Sigma}_j\right)\right]^{-1/2}.$$

The matrix $\widehat{\mathcal{H}}_j$ is a natural estimator of the matrix $\mathcal{H}_j$ presented above.

- Application of the desired matrix norm to $\widehat{\mathcal{H}}_j$ in order to produce the resulting $\widehat{\varphi}_j$ is a matter of routine calculation. Explicitly, we define

$$\widehat{\varphi}_j = \|\widehat{\mathcal{H}}_j\|_{\mathfrak{p}}.$$

Once $\widehat{\varphi}_j$ is calculated for each $j = 1, 2, \ldots, p$, we then rank all candidate predictors according to their associated $\widehat{\varphi}_j$ value, from largest to smallest. Covariates associated with larger $\widehat{\varphi}_j$ values are taken as having a larger association with the response $\mathbb{Y}$. Herein, we

will call this newly proposed method GenCorr, in reference to the integral use of generalized correlation in the method. When reference to a specific norm is necessary, we will indicate as such. GenCorr-T will refer to instances where the taxi-cab norm is used. GenCorr-F will refer to instances where the Frobenius norm is used.

4.2.2 Extensions to Interaction Feature Screening

From a purely theoretical vantage, this proposed method can be further extended to screen for r-way (with $r \geq 2$) interactions between predictors as follows:

$$\text{Let } \Sigma_{j_1,j_2,\ldots,j_r} = \left(\begin{array}{c|c} \prod_{s=1}^{r} [\text{Var}(X_j)] & \mathcal{K}^T \\ \hline \mathcal{K} & \text{Cov}(\mathbb{Y}) \end{array}\right),$$

where $\mathcal{K}$ is a q-vector whose mth entry is given by

$$\mathcal{K}_m = \kappa_{r+1}\left(Y^{(m)},\ X_{j_1},\ X_{j_2},\ \ldots,\ X_{j_r}\right), \quad m = 1, 2, 3, \ldots, q.$$

Here k_{r+1} represents the $(r+1)$-way joint cumulant. For a further discussion on joint cumulants, see Chapter 3, [62], and [42]. Note that by viewing the covariance of a component of $\mathbb{Y}$ and a given X_j as the two-way joint cumulant, the previously outlined application of GenCorr to screen for marginal effects is really just a subcase of the method presented here for feature interaction. As such, much of the relevant theory and proofs relating to GenCorr will be presented in the form of r-way interaction screening. The easily conceivable case where $r = 1$ will account for the desired theoretical properties of GenCorr when applied to marginal effect screening. This ensuing presentation of GenCorr solely via the broader framework of r-way interactions is done to avoid duplicating nearly identical theorems for marginal screening and interaction screening separately.

As a matter of notation, define the following set of r-tuples with integer entries

$$\mathcal{I} = \{(j_1, j_2, \ldots, j_r) \mid 1 \leq j_1 < j_2 < \cdots < j_r \leq p\}.$$

The set $\mathcal{I}$ contains all combinations of the covariate indices for which we can have an r-way interaction.

Define the following:

$$\mathcal{H}_{j_1,j_2,\ldots,j_r} = \left(\left[\operatorname{diag} \left(\Sigma_{j_1,j_2,\ldots,j_r} \right) \right]^{-1/2} \Sigma_{j_1,j_2,\ldots,j_r} \left[\operatorname{diag} \left(\Sigma_{j_1,j_2,\ldots,j_r} \right) \right]^{-1/2} \right).$$

We can now create the following interaction utility measure for r-way covariate interaction ranking:

$$\text{Let} \quad \Phi_{j_1,j_2,\ldots,j_r} = \| \mathcal{H}_{j_1,j_2,\ldots,j_r} \|_{\mathfrak{p}},$$

where once again $\|\cdot\|_{\mathfrak{p}}$ is any (entry-wise) $\ell_{\mathfrak{p}}$ norm with $\mathfrak{p} < \infty$.

As was the case with the construction of $\widehat{\varphi}_j$, it is a simple exercise to construct an estimator of $\Phi_{j_1,j_2,\ldots,j_r}$:

- The variance of each X_j is estimated as before for $\widehat{\varphi}_j$.
- The $(r+1)$-way joint cumulant is estimated by

$$\widehat{\kappa}_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}) = \frac{1}{n} \sum_{i=1}^{n} \left[\prod_{s=1}^{r} \left(X_{ij_s} - \overline{X}_{j_s} \right) \right] \left(Y_i^{(m)} - \overline{Y}^{(m)} \right),$$

  where $\overline{X}_{j_s}$ represents the sample mean of X_{j_s} and $\overline{Y}^{(m)}$ is the sample mean of $Y^{(m)}$.
- The covariance matrix of $\mathbb{Y}$ is to be estimated in the usual way, as was done in the case of $\widehat{\varphi}_j$.
- Quite like was done in the marginal effects screening case, one can now create an estimator $\widehat{\mathcal{H}}_{j_1,j_2,\ldots,j_r}$ of the matrix $\mathcal{H}_{j_1,j_2,\ldots,j_r}$. From this, the values of an estimator,

denote it by $\widehat{\Phi}_{j_1,\ldots,j_r}$, are a straight forward calculation. Explicitly, we define

$$\widehat{\Phi}_{j_1,\ldots,j_r} = \|\widehat{\mathcal{H}}_{j_1,j_2,\ldots,j_r}\|_{\mathfrak{p}}.$$

After calculating $\widehat{\Phi}_{j_1,\ldots,j_r}$ for each r-tuple in $\mathcal{I}$, we then can rank each interaction as most to least important based on the associated values of $\widehat{\Phi}_{j_1,\ldots,j_r}$. Larger $\widehat{\Phi}_{j_1,\ldots,j_r}$ values are taken as having larger association with $\mathbb{Y}$. In this way, $\widehat{\Phi}_{j_1,\ldots,j_r}$ will act as a screening utility for feature screening.

In line with the concept of sparsity in ultrahigh dimensional feature screening, we can assume that only a small number of the interactions between covariates have a truly causative association with the response. Let $\mathcal{S}_F = \mathcal{I}$ represent what we will call the "full model." This model is a model which admits every possible (r-way) interaction between the covariates in the feature space. Let $\mathcal{S} \subseteq \mathcal{S}_F$ denote an arbitrary model to be taken under examination. We will also define $X_{(\mathcal{S})}$ to be the set of all covariate interactions whose r-tuple indices are contained in $\mathcal{S}$. Given a positive constant $c > 0$, we can define an estimated model:

$$\widehat{\mathcal{S}} = \{(j_1, \ldots, j_r) \in \mathcal{I} \ \mid \ \widehat{\Phi}_{j_1,\ldots,j_r} > c\}.$$

Here $\widehat{\mathcal{S}}$ represents a model selected by GenCorr given a predetermined cutoff $c > 0$. Multiple approaches exist for determining this cutoff [see e.g. 43, 48, 91, 93]. We suggest examining these methods for use with GenCorr, but will not explore the determination of a cutoff further.

Let $\mathcal{D}\left(\mathbb{Y} \mid X_{(\mathcal{S})}\right)$ represent the conditional distribution of $\mathbb{Y}$ given $X_{(\mathcal{S})}$. We will say that a model $\mathcal{S}$ is *sufficient* if

$$\mathcal{D}\left(\mathbb{Y} \mid X_{(\mathcal{S}_F)}\right) = \mathcal{D}\left(\mathbb{Y} \mid X_{(\mathcal{S})}\right).$$

The full model $\mathcal{S}_F$ is clearly sufficient. Ultimately, we are really only interested in finding the smallest (in terms of cardinality) sufficient model. The smallest sufficient model is also

known as the *true model*. We will denote the true model by $\mathcal{S}_T$. We will denote the number of features in a given model by $|\mathcal{S}|$. (So the number of true or causative features would be written as $|\mathcal{S}_T|$). The principal aim in feature screening is producing an estimated model which not only *contains* the true model, but moreover is the *smallest* such model to contain all the true features (or, as the case may be, all true interactions).

The most common situation where our approach to interaction screening can be applied in practice is in the case where $r = 2$ (i.e. screening for two-way interactions). Computational limitations currently hedge what feasibly can be done with ultrahigh dimensional feature spaces when r is greater than two. Owing to these limitations, all simulations pertaining to interaction screening presented herein (see Section 4.3, Simulation 5) will only deal with two-way interactions.

4.2.3 Theoretical properties

We first establish several conditions to facilitate the technical proofs that will be presented in Section 4.5.

(C1) *Bounds on the standard deviations.* We denote the variances of each X_j and the variance of the components of $\mathbb{Y}$ as follows: $\text{Var}(X_j) = \sigma_j^2$ and $\text{Var}(Y^{(m)}) = \sigma_{(m)}^2$. Assume that there exists a positive constant σ_{min} such that for every $1 \leq j \leq p$ and $1 \leq m \leq q$,

$$\sigma_j, \sigma_{(m)} \geq \sigma_{min}.$$

This allows us to exclude any covariates that are constant and thus have a standard deviation of zero.

(C2) *Entries of $\mathcal{H}_{j_1,\ldots,j_r}$ for $(j_1, \ldots, j_r)$ in $\mathcal{S}_T$.* Assume that for each $(j_1, \ldots, j_r)$ in $\mathcal{S}_T$ there exists some positive constant $\omega_{j_1,\ldots,j_r} > 0$ and some m in $\{1, 2, \ldots, q\}$ such that

$$\left| \frac{\kappa_{r+1}\left(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}\right)}{\sigma_{(m)}\sigma_{j_1} \cdots \sigma_{j_r}} \right| > \omega_{j_1,\ldots,j_r} > 0.$$

This condition ensures that for every true or causative interaction, there is at least one component of $\mathbb{Y}$ with which the causative interaction has a non-zero association as determined by the joint cumulant.

(C3) $\Phi_{j_1,\ldots,j_r}$ *for* $(j_1,\ldots,j_r)$ *not in* $\mathcal{S}_T$. Define the following constant.

$$\gamma = (q+1) + \sum_{1\leq \ell,m\leq q} |\rho_{\ell m}|^2,$$

where $\rho_{\ell m}$ is the correlation between $Y^{(\ell)}$ and $Y^{(m)}$. Note that γ minimally equals $q+1$, meaning that γ is necessarily positive. We will assume that $\Phi_{j_1,\ldots,j_r} = \sqrt{\gamma}$ for any $(j_1,\ldots,j_r) \notin \mathcal{S}_T$.

When some or all of these aforementioned conditions hold, we can state the following theorems.

Theorem 4.2.1. *(Sure Screening) Given that conditions (C1) and (C2) hold, there exists a positive constant* $c > 0$ *such that if*

$$\widehat{\mathcal{S}} = \{(j_1,\ldots,j_r) \in \mathcal{I} \;\mid\; \widehat{\Phi}_{j_1,\ldots,j_r} > c\},$$

then we have

$$\lim_{n\to\infty} \mathbb{P}\left(\mathcal{S}_T \subseteq \widehat{\mathcal{S}}\right) = 1.$$

Note, however, that we have no guarantee in this case of $\mathcal{S}_T$ asymptotically containing the estimated model $\widehat{\mathcal{S}}$.

Theorem 4.2.2. *(Strong Sure Screening) Given that conditions (C1), (C2), and (C3) all hold, there exists a positive constant* $c > 0$ *such that if*

$$\widehat{\mathcal{S}} = \{(j_1,\ldots,j_r) \in \mathcal{I} \;\mid\; \widehat{\Phi}_{j_1,\ldots,j_r} > c\},$$

then we have

$$\lim_{n\to\infty} \mathbb{P}\left(\mathcal{S}_T = \widehat{\mathcal{S}}\right) = 1.$$

This strong sure screening property given in Theorem 4.2.2 is naturally more difficult to obtain than the (weak) sure screening property of Theorem 4.2.1. However, strong sure screening ensures that we not only obtain (asymptotically) an estimated model that *contains* the true model, but furthermore that we obtain an estimated model that asymptotically *equals* $\mathcal{S}_T$ with probability equal to one. Beyond the theorems on sure screening stated above, we can also obtain the following corollaries.

Corollary 4.2.3. *There exists a value* $\Phi_{\min}$ *such that*

$$\Phi_{j_1,\ldots,j_r} > \Phi_{\min} > \sqrt{\gamma} > 0$$

whenever $(j_1, \ldots, j_r) \in \mathcal{S}_T$.

Corollary 4.2.4. *The estimator* $\widehat{\Phi}_{j_1,\ldots,j_r}$ *converges* uniformly *in probability to* $\Phi_{j_1,\ldots,j_r}$. *That is to say, for any* $\varepsilon > 0$,

$$\lim_{n\to\infty} \mathbb{P}\left(\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r}\right| > \varepsilon\right) = 0.$$

These corollaries will come as a natural result of the proofs of Theorems 4.2.1 and 4.2.2 (See Section 4.5).

4.3 Simulations and Empirical Data Analysis

We will first evaluate the empirical performance of GenCorr via a collection of five groups of simulations. These simulations are then followed by a real data analysis on data from a genome-wide association mapping of outbred mice. The first four groups of simulations pertain to screening for marginal effects. We will compare the performance of GenCorr with two other methods for marginal effects screening when the response is multivariate: Sure Independence Ranking Screening (SIRS) [93] and Distance Correaltion (DC-SIS) [53].

Two distinct matrix norms for GenCorr will also be considered. We will denote GenCorr under the taxicab norm as GenCorr-T; GenCorr under the Frobenius norm will be denoted by GenCorr-F.

4.3.1 Simulation 1

For this simulation we wish to examine the ability of GenCorr, SIRS, and DC-SIS to successfully screen for a small set of causative or "true" predictors in a linear model with normally distributed predictors. This simulation will be split into three parts. Each part will explore the effect of different methods for generating the coefficients for the causative predictors.

Simulation 1.A

Here we take a sample size of $n = 60$ of each of $p = 3000$ predictors. First we generate each covariate, X_j as follows:

$$X_j \sim \mathrm{N}(0, \sigma = 5), \quad \text{sampled 60 times.}$$

Let Σ be a 6×6 matrix given by

$$\Sigma = \left[0.5^{|\ell - m|}\right]_{m,\ell}.$$

Next we create the following matrix B:

$$\text{For } k = 1, 2, 3 \text{ sample } (\beta_{k1}, \beta_{k2}, \beta_{k3}, \beta_{k4}, \beta_{k5}, \beta_{k6}) \sim \mathrm{MVN}(\mathbf{0}, \Sigma),$$

$$\text{and let } B = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} & \beta_{15} & \beta_{16} \\ \beta_{21} & \beta_{22} & \beta_{23} & \beta_{24} & \beta_{25} & \beta_{26} \\ \beta_{31} & \beta_{32} & \beta_{33} & \beta_{34} & \beta_{35} & \beta_{36} \end{pmatrix}.$$

We then construct the values of $\mathbb{Y}$ with $q = 6$ as follows:

$$\mathbb{Y} = \begin{pmatrix} | & | & | & | & | & | \\ Y^{(1)} & Y^{(2)} & Y^{(3)} & Y^{(4)} & Y^{(5)} & Y^{(6)} \\ | & | & | & | & | & | \end{pmatrix} = \begin{pmatrix} | & | & | \\ X_1 & X_2 & X_3 \\ | & | & | \end{pmatrix} B.$$

Written more explicitly,

$$Y^{(m)} = \beta_{1m} X_1 + \beta_{2m} X_2 + \beta_{3m} X_3.$$

This construction will mean that X_1, X_2 and X_3 are to be considered as causative, while the remaining X_j will be taken as noise. We ran 400 replications of this simulation.

We first report the means of the best, medial, and worst rankings of the three causative predictors by each method in question. In doing this, we are not concerned with tracking the individual ranks of X_1, X_2, and X_3, but rather we focus our efforts on recording the ranks of the best, medial, and worst ranked true predictors, irrespective of which causative covariate was which. This allows us to gain overall insight into the minimal model size required (on average) to positively include, respectively, one, two, or all three of the causative predictors. The average worst ranking for a method informs us of the average minimum model size that would be necessary to assure all three of the true predictors are present. Hence, a mean worst rank of 46 would mean that, on average, we would need a final model size of 46 to guarantee that we have included all three of the true predictors in our model. A perfect score of best, medial, and worst rankings would be (1,2,3), which corresponds to a minimum model size of three. By reporting each of the best, medial, and worst rankings, we can observe not only how large of a model we must on average have, but also how small of a model we can have and still retain one or more of the causative variables. The mean rank results for each of the four methods (GenCorr-T, GenCorr-F, SIRS, DC-SIS) are given in Table 4.1.

Both SIRS and DC-SIS are observably inadequate as a screening method in this setting,

as neither is able to obtain any semblance of returning a viable result consistently. However, under the taxicab norm, our GenCorr method performs admirably here, with an average worst ranking (i.e. average minimum necessary model size) just over 21. When the Frobenius norm is used, the results are even more impressive, with a minimum required model size just under ten. In either case, this means that GenCorr has successfully reduced the feature space to a collection with dimension less than $n = 60$.

It can be beneficial to also examine the median of the best, medial, and worst rankings. (See Table 4.2). From this we see that SIRS is actually able to determine some of the true features with acceptably high accuracy at least half of the time. However, because SIRS seems especially vulnerable to error when the sample size is small (relative to p), the method struggles to produce consistent results on average. (Median worst ranks of 3 for both applications of GenCorr compared to a median worst rank of 423.50 for SIRS).

Simulation 1.B

We again employ a sample size of $n = 60$ for each of $p = 3000$ predictors. First we generate each covariate, X_j as follows:

$$X_j \sim \mathrm{N}(3, \sigma = 1), \quad \text{sampled 60 times.}$$

As before, let Σ be a 6×6 matrix given by

$$\Sigma = \left[0.5^{|\ell - m|}\right]_{m,\ell}.$$

The rows of the coefficient matrix B are given as follows:

$$\text{For } k = 1, 2, 3 \text{ sample } (\beta_{k1}, \beta_{k2}, \beta_{k3}, \beta_{k4}, \beta_{k5}, \beta_{k6}) \sim \mathrm{MVN}(\mathbf{3}, \Sigma).$$

Thus, as opposed to part A of the simulation, the coefficients are now generated from the multivariate normal distribution with means three instead of zero. Tables 4.1 and 4.2 report the mean and median rankings for Simulation 1.B.

With the adjustment in the means of the predictors and their coefficients, we see an increased accuracy over part A on the part of SIRS (mean worst rank of 57.81 versus a mean worst rank of 890.155 in part A). Both implementations of GenCorr again yield the smallest minimum required model sizes on average (35.5425 and 22.65 for the taxicab and Frobenius norms, respectively). While the performance of DC-SIS improves under the current settings, it lags well behind GenCorr in terms of both mean and median minimum required model size.

Simulation 1.C

For this simulation, we set n equal to 120 and let p be 1500. This change in sample size is done to better avoid the vulnerabilities of SIRS under small (relative to p) sample size. Here we set q equal to four. As was done in part A, we let $X_j \sim N(0, \sigma = 5)$.

Based on Example 1 of DC-SIS (who in turn follow [24]), for m from 1 to 4 (inclusive) and $j = 1, 2, 3$ we define

$$\beta_{jm} = (-1)^U(a + |Z|), \quad \text{where } a = \frac{4\log(n)}{\sqrt{n}},\ U \sim \text{Bernoulli}(0.4),\ \text{and } Z \sim N(0, 1).$$

As before, we let

$$Y^{(m)} = \beta_{1m}X_1 + \beta_{2m}X_2 + \beta_{3m}X_3.$$

Tables 4.1 and 4.2 depict the mean and median ranks of the three methods on question under these settings.

This method for constructing the coefficients of the causative features bears perfect scores for GenCorr (under both of the norms used). While both SIRS and DC-SIS improve upon their results from part A, these two methods lag far behind GenCorr in overall accuracy. (GenCorr achieves nearly perfect scores under both matrix norms; SIRS and DC-SIS obtain mean minimum model sizes over 100 times larger than perfect acquisition).

Table 4.1: Simulation 1 Mean Ranks

Part A			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0000	2.4000	21.3025
GenCorr-F	1.0000	2.0300	9.715
SIRS	185.4550	397.4375	890.1550
DC-SIS	22.8325	296.1950	1236.3275
Part B			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0000	2.1650	35.5425
GenCorr-F	1.0000	2.0925	22.6500
SIRS	1.0000	2.2950	57.8100
DC-SIS	2.4450	32.8375	342.5025
Part C			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0000	2.0000	3.0000
GenCorr-F	1.0000	2.0000	3.0050
SIRS	66.2025	169.6350	327.8850
DC-SIS	10.6100	102.1875	522.7250

Table 4.2: Simulation 1 Median Ranks

Part A			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.00	2.00	3.00
GenCorr-F	1.00	2.00	3.00
SIRS	1.00	19.50	423.50
DC-SIS	1.00	39.00	1098.00
Part B			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.00	2.00	4.00
GenCorr-F	1.00	2.00	3.00
SIRS	1.00	2.00	5.00
DC-SIS	1.00	3.00	52.00
Part C			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.00	2.00	3.00
GenCorr-F	1.00	2.00	3.00
SIRS	1.00	2.00	9.00
DC-SIS	1.00	3.00	294.00

4.3.2 Simulation 2

Simulation 2 is constructed quite similarly to Simulation 1, however we now test the ability of each of GenCorr, SIRS, and DC-SIS to detect an *exponential* relationship between

$\mathbb{Y}$ and X_1, X_2 , and X_3. We still retain $n = 60$, $p = 3000$, and $q = 6$ from before.

Simulation 2.A

With each X_j being constructed using the approach from Simulation 1.A, now define $\mathbb{Y}$ as follows:

$$Y^{(m)} = \exp\Big\{\beta_{1m}X_1 + \beta_{2m}X_2 + \beta_{3m}X_3\Big\},$$

with the matrix B being created in the same manner as found in Simulation 1.A. Once again we run 400 replications of this simulation.

The results for Simulation 2.A are given in Table 4.3.

Table 4.3: Simulation 2 Mean Ranks

	Best Rank	Medial Rank	Worst Rank
GenCorr-T	9.3100	97.2650	545.0075
GenCorr-F	8.7075	77.8800	493.4925
SIRS	194.2475	443.1475	921.8525
DC-SIS	464.4850	1134.8150	2016.2150

We also report in Table 4.4 the median of each of the best, medial, and worst ranks.

Table 4.4: Simulation 2 Median Ranks

	Best Rank	Medial Rank	Worst Rank
GenCorr-T	2.00	36.00	324.50
GenCorr-F	3.00	31.00	239.00
SIRS	1.00	17.00	383.00
DC-SIS	283.50	1025.00	2162.00

While none of the methods produce breathtaking results, GenCorr (under both the taxicab and Frobenius norms) is consistently the superior method in terms of producing the smallest required mean minimum model size (GenCorr achieves mean minimum model sizes that are about half that achieved by SIRS, and about four times smaller than the same for DC-SIS). Here we must keep in mind that, at times, feature screening is not so much concerned with the matter of selecting the true predictors, but rather removing those predictors which are definitively unimportant. In cases such as this, the aim often is not to directly determine the smallest sufficient model, but rather to screen the set of p-many predictors into a collection of reduced size. [See, for example, the discussions in 21, 27, on this topic.]. Penalized regression methods such as those mentioned in Section 4.1 can then be applied to the reduced set of predictors.

Because SIRS is only focused on ranking the covariates based their empirical cumulative distribution function, any monotonicly increasing transformation (e.g. an exponetional transformation) of the covariates will produce the same feature screening results as obtained from the original, untransformed, features (up to choice of random seed). To that end, it should be noted that the results for SIRS in Simulation 1.A will differ from the results for SIRS here, as different random seeds were used to generate the simulation data sets in Simulation 1 and Simulation 2. This will be the case in Simulations 3 and 4 as well.

4.3.3 Simulation 3

The previous simulations explored the ability of GenCorr to successfully screen for causative covariates when both the response and the covariates are continuous. We now turn our attention to the screening of data with discrete predictors and discrete multivariate response.

Simulation 3.A

Here we use the same general setup as used in Simulation 1.A. The one change we implement is with the distribution of the predictors:

$$X_j \sim \text{Pois}(\lambda = 2).$$

Outside of this change, we retain $n = 60$, $p = 3000$, $q = 6$, and perform 400 replications as before. The coefficient matrix B is generated in the same fashion as was done in Simulation 1.A. The overall simulation model will thus be given by

$$\mathbb{Y} = \begin{pmatrix} | & | & | & | & | & | \\ Y^{(1)} & Y^{(2)} & Y^{(3)} & Y^{(4)} & Y^{(5)} & Y^{(6)} \\ | & | & | & | & | & | \end{pmatrix} = \begin{pmatrix} | & | & | \\ X_1 & X_2 & X_3 \\ | & | & | \end{pmatrix} B.$$

This will mean that once again, X_1, X_2 and X_3 will be considered causative. The results for Simulation 3.A are given in Table 4.5 (mean ranks) and Table 4.6 (median ranks).

The results for GenCorr are highly encouraging. On average, the required minimum model size produced by GenCorr-T is just below 27, a reduction to a model size less than half of the sample size of 60. GenCorr-F produces even more favorable results, with a minimum required model size just above 15. Moreover, 96.25% of the replicates of GenCorr-F produce a required minimum model size less than 60. By comparison, SIRS only obtains a required minimum model size less than 60 in 29.5% of replicates.

Simulation 3.B

Part B of Simulation 3 uses the same set up that part B of Simulation 1 used, with the one difference being the distribution of the covariates. Like Simulation 3.A, we let

$$X_j \sim \text{Pois}(\lambda = 2).$$

The causative covariates will again be X_1, X_2, and X_3. The results for Simulation 3.B are given in Table 4.5 (mean ranks) and Table 4.6 (median ranks).

On average, GenCorr-T requires a minimum model size about ten to eleven features smaller than that required by SIRS to capture all causative features. GenCorr-F bests all methods, with a mean minimum model size just greater than 25. Under the settings of part B, DC-SIS improves drastically over its performance in part A. However, in spite of this improvement, DC-SIS still lags significantly behind GenCorr and SIRS.

Simulation 3.C

The same overall setup as was used in Simulation 1.C is used here, yet we now take

$$X_j \sim \text{Pois}(\lambda = 2).$$

The values of $n = 120$, $p = 1500$, $q = 4$ and 400 replicates are carried over from Simulation 1.C. We report the results for Simulation 3.C in Table 4.5 (mean ranks) and Table 4.6 (median ranks).

Both versions of GenCorr yield perfect $(1, 2, 3)$ scores in each of the 400 replications. SIRS and DC-SIS produce significantly less favorable results, especially when viewed in the light of mean minimum model size. Although the median ranks for these latter methods are more in line with the GenCorr results, the inconsistencies in the average case are concerning.

Table 4.5: Simulation 3 Mean Ranks

Part A			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0000	2.2375	26.8025
GenCorr-F	1.0000	2.1850	15.3075
SIRS	186.3225	432.5900	837.1225
DC-SIS	46.3400	412.4975	1440.4675
Part B			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0025	2.2000	35.6125
GenCorr-F	1.0025	2.0850	25.3350
SIRS	1.0050	2.3150	46.4925
DC-SIS	1.0450	8.1075	146.0100
Part C			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0000	2.0000	3.0000
GenCorr-F	1.0000	2.0000	3.0000
SIRS	54.7700	140.9275	260.7725
DC-SIS	5.5000	73.9400	466.0725

Table 4.6: Simulation 3 Median Ranks

Part A			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0	2.0	3.0
GenCorr-F	1.0	2.0	3.0
SIRS	1.0	18.5	310.5
DC-SIS	3.0	106.5	1280.0
Part B			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.0	2.0	4.0
GenCorr-F	1.0	2.0	3.0
SIRS	1.0	2.0	5.0
DC-SIS	1.0	2.0	9.0
Part C			
	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.00	2.00	3.00
GenCorr-F	1.00	2.00	3.00
SIRS	1.00	2.00	6.00
DC-SIS	1.00	6.00	176.50

4.3.4 Simulation 4

We now turn our attention to a model exhibiting an exponential relationship between the response and the causative covariates.

Simulation 4.A

As was the theme in each part of Simulation 3, we follow the overall setup of a previous simulation, yet with each covariate coming from the Poisson distribution with mean two. Here we emulate the approach used in Simulation 2.A, with the single difference being

$$X_j \sim \text{Pois}(\lambda = 2).$$

The values of $n = 60$, $p = 3000$, and $q = 6$ are retained from Simulation 2.A. We perform 400 replications. The average rank results for Simulation 4.A are given in Table 4.7. The median rank results are given in Table 4.8

Table 4.7: Simulation 4 Mean Ranks

	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.4350	16.3325	140.0025
GenCorr-F	1.265	11.445	112.070
SIRS	195.7550	431.3475	832.6675
DC-SIS	252.5000	908.6000	1899.4725

Table 4.8: Simulation 4 Median Ranks

	Best Rank	Medial Rank	Worst Rank
GenCorr-T	1.00	2.00	16.00
GenCorr-F	1.00	2.00	14.00
SIRS	1.00	18.50	317.50
DC-SIS	79.00	739.00	2022.50

As was the case with Simulation 2.A, the mean minimum model sizes obtain by each method exceed the number of samples n. However, this is once again a case where the ultimate goal is not to singularly obtain a final model, but rather to reduce the feature set preparatory to performing further feature space reduction methods such as penalized regression. In both the mean and median rank cases, GenCorr looks far more promising in this regard than the other two methods. Of note, GenCorr-F finds at least two of the causative covariates to be within the top 30 most important covariates in 92.75% of the replicates (GenCorr-T follows closely at 90.25%). SIRS obtains such results only 53.75% of the time; DC-SIS obtains such in only 4% of the replicates.

4.3.5 Simulation 5

We now turn our attention to empirically examining the claim of GenCorr to not only screen for marginal effect of individual predictors on a multivariate response, but also the ability of the extended version of GenCorr to screen for *interactive effects* on a multivariate response. We will explore results under both the taxi-cab norm and the Frobenius norm.

Simulation 5.A

Here we take a sample of $n = 100$ of each of $p = 1000$ predictors. First we generate each covariate as follows:

$$X_j \sim \mathrm{N}(0, \sigma = 2), \quad \text{sampled 100 times.}$$

Let B be the 2×4 matrix defined as follows:

$$B = \begin{pmatrix} 1 & -1 & -2.5 & 2 \\ 2 & 1.5 & -2 & 1 \end{pmatrix}.$$

We also define an $n \times q$ error matrix

$$E = \begin{pmatrix} | & | & | & | \\ \epsilon_1, & \epsilon_2 & \epsilon_3 & \epsilon_4 \\ | & | & | & | \end{pmatrix}, \text{ with each } \epsilon_m \overset{i.i.d}{\sim} \text{MVN}(\mathbf{0}, I_n).$$

For $q = 4$, we then construct the values of $\mathbb{Y}$:

$$\mathbb{Y} = \begin{pmatrix} | & | & | & | \\ Y^{(1)} & Y^{(2)} & Y^{(3)} & Y^{(4)} \\ | & | & | & | \end{pmatrix} = \begin{pmatrix} | & | \\ X_1X_2 & X_3X_4 \\ | & | \end{pmatrix} B + E,$$

where $X_{j_1}X_{j_2}$ represents the product of X_{j_1} and X_{j_2}.

This construction will mean that the interaction between X_1 and X_2, as well as the interaction between X_3 and X_4, are to be considered as causative. All other pairwise interactions will act as noise. For the time being, we will omit any causative marginal effects from the model. A model that also includes causative marginal effects will be considered in part C of this simulation. We ran 400 replications of this simulation.

The proportion of replicates for which each individual causative interaction is within the top five interactions is represented by $\mathcal{P}_{j_1,j_2}$. We will denote by $\mathcal{P}_{\text{top}}$ the proportion of replicates for which one of the true interactions is found to be *the* most important interaction. The proportion of replicates for which *both* causative interactions are determined to be within the top five most important interactions is given under $\mathcal{P}_{\text{a}}$. These proportions are given in Table 4.9. We then also report the 25%, 50%, 75% and 90% quantiles of the required number of interactions to contain the true interactions. These results are given in Table 4.10.

The values for both $\mathcal{P}_{1,2}$ and $\mathcal{P}_{3,4}$ are highly encouraging. Moreover, the $\mathcal{P}_{\text{top}}$ value tells us that in all but one of the replications GenCorr determines one of the true interactions to be the interaction most strongly associated with the response. (In the one replicate where neither true interaction was found to be the top interaction, the interaction between X_1

and X_2 was nevertheless found to be the *second* most important interaction). Under the taxi-cab norm, GenCorr results in both causative interactions being found in the top five most important interaction 93.75% of the time. GenCorr with the Frobenius norm improves upon this, finding both causative interactions to be within the top five most important interactions 95.50% of the time. The quantile values for part A in Table 4.10 indicate that in a significant majority of cases, GenCorr (under either norm) can locate the two true causative interactions with a high level of accuracy.

Simulation 5.B

In this part of Simulation 5, we again use a sample size of $n = 100$ and let $p = 1000$. Like with part A, we will be examining a model without causative marginal effects. However, unlike in part A, where the model coefficients were fixed for each replicate, we now will generate the coefficients on the true interactions anew for each of 400 replications of the simulation. Once again, each covariate is sampled from the normal distribution centered at zero and having standard deviation of two:

$$X_j \sim \mathrm{N}(0, \sigma = 2).$$

Let Σ be a 4×4 matrix defined by

$$\Sigma = \left[0.5^{|\ell - m|}\right]_{m,\ell}.$$

Next we create the coefficient matrix B:

$$\text{For } k = 1, 2 \text{ sample } (\beta_{k1}, \beta_{k2}, \beta_{k3}, \beta_{k4}) \sim \mathrm{MVN}(\mathbf{3}, \Sigma),$$

$$\text{and let } B = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ \beta_{21} & \beta_{22} & \beta_{23} & \beta_{24} \end{pmatrix}.$$

The values of $\mathbb{Y}$ with $q = 4$ are the constructed as follows:

$$\mathbb{Y} = \begin{pmatrix} | & | & | & | \\ Y^{(1)} & Y^{(2)} & Y^{(3)} & Y^{(4)} \\ | & | & | & | \end{pmatrix} = \begin{pmatrix} | & | \\ X_1X_2 & X_3X_4 \\ | & | \end{pmatrix} B.$$

Written more explicitly, this model is

$$Y^{(m)} = \beta_{1m}X_1X_2 + \beta_{2m}X_3X_4.$$

This construction will mean that the interaction between X_1 and X_2 and the interaction between X_3 and X_4 will be considered as causative. All remaining pairwise interactions will be seen as noise.

Like in part A, we report $\mathcal{P}_{j_1,j_2}$, $\mathcal{P}_{\text{top}}$, $\mathcal{P}_{\text{a}}$, q_{25}, q_{50}, q_{75}, and q_{90}. These results are given in Tables 4.9 (selection proportions) and 4.10 (quantiles).

Even under the additional challenge of handling varying coefficients on the causative interactions, GenCorr yet obtains consistently accurate results for both the taxi-cab and the Frobenius norms. These results strengthen our trust in GenCorr as a feature interaction screening method when the response is multivariate. It should be noted that although the $\mathcal{P}_{\text{a}}$ values hover just above 0.6, it is easy to confirm by use of the well known addition rule in probability that (outside of a single replication), whenever one of the causative interactions is not found to be within the top five most important interactions, the remaining causative interaction exercises a strong effect on the response and is positively identified in the top five most important predictors. Such an event occurs when one of the causative interactions dominates over the other causative interaction (likely due to random selection of a coefficient vector with comparatively small values being assigned to the non-dominant interaction).

Simulation 5.C

For this part of Simulation 5, we will once again use a sample size of $n = 100$ and let

$p = 1000$. As always, we will run 400 replications of this part of the simulation. Once again, each covariate is sampled from the normal distribution centered at zero and having standard deviation of two:

$$X_j \sim \mathrm{N}(0, \sigma = 2).$$

Unlike parts A and B, however, we now incorporate marginal effects into our model. This is done as follows.

Let Σ be a 4×4 matrix defined by

$$\Sigma = \left[0.5^{|\ell - m|}\right]_{m,\ell}.$$

Next we create the coefficient matrix B:

$$\text{For } k = 1, 2, 3, 4, 5, 6 \text{ sample } (\beta_{k1}, \beta_{k2}, \beta_{k3}, \beta_{k4}) \sim \mathrm{MVN}(\mathbf{3}, \Sigma),$$

$$\text{and let } B = \begin{pmatrix} \beta_{11} & \beta_{12} & \beta_{13} & \beta_{14} \\ \beta_{21} & \beta_{22} & \beta_{23} & \beta_{24} \\ \beta_{31} & \beta_{32} & \beta_{33} & \beta_{34} \\ \beta_{41} & \beta_{42} & \beta_{43} & \beta_{44} \\ 3\beta_{51} & 3\beta_{52} & 3\beta_{53} & 3\beta_{54} \\ 3\beta_{61} & 3\beta_{62} & 3\beta_{63} & 3\beta_{64} \end{pmatrix}.$$

The values of $\mathbb{Y}$ with $q = 4$ are then constructed as follows:

$$\mathbb{Y} = \begin{pmatrix} | & | & | & | \\ Y^{(1)} & Y^{(2)} & Y^{(3)} & Y^{(4)} \\ | & | & | & | \end{pmatrix} = \begin{pmatrix} | & | & | & | & | & | \\ X_1 & X_2 & X_3 & X_4 & X_1X_2 & X_3X_4 \\ | & | & | & | & | & | \end{pmatrix} B.$$

Component-wise for each component of $\mathbb{Y}$, we have

$$Y^{(m)} = \beta_{1m}X_1 + \beta_{2m}X_2 + \beta_{3m}X_3 + \beta_{4m}X_4 + 3\beta_{5m}X_1X_2 + 3\beta_{6m}X_3X_4.$$

This construction will again allow the interaction between X_1 and X_2 and the interaction between X_3 and X_4 to be taken as causative. The addition of the marginal effects of X_1, X_2, X_3, and X_4 means that GenCorr will now be faced with the heightened challenge of detecting the truly causative interactions while being confronted with the specious interactions involving only one of the marginal covariates. Such "mixed" interactions (non-causative interactions involving one of X_1, X_2, X_3, or X_4; e.g. X_1X_7, X_3X_9, X_2X_4) can sometimes appear to be strongly associated with the response, when in fact the marginal effect of the causative covariate(s) alone is causing such an association. It should be noted that the marginal effects are constructed to be overall rather weak, and, consequently, will not be strongly detected by a marginal screening method. Once again, we report $\mathcal{P}_{j_1,j_2}$, $\mathcal{P}_{\text{top}}$, and $\mathcal{P}_{\text{a}}$, as well as the 25%, 50%, 75% and 90% quantiles of the minimum number of interactions required to capture both causative interactions. These results are given in Table 4.9 (selection proportions) and 4.10 (quantiles). Even with the addition of main effects, we see no drop in the overall accuracy of GenCorr in determining the true set of interactions. These results indicate that GenCorr (under either norm) is yet quite capable of detecting the causative interactions regardless of the addition of marginal effects.

Table 4.9: Simulation 5 Selection Proportions

Model	Method	$\mathcal{P}_{1,2}$	$\mathcal{P}_{3,4}$	$\mathcal{P}_{\text{top}}$	$\mathcal{P}_{\text{a}}$
5.A	GenCorr-T	0.9600	0.9775	0.9975	0.9375
	GenCorr-F	0.9650	0.9900	0.9975	0.9550
5.B	GenCorr-T	0.8075	0.8025	0.9925	0.6125
	GenCorr-F	0.8250	0.8275	0.9925	0.6550
5.C	GenCorr-T	0.8025	0.8300	0.9775	0.6350
	GenCorr-F	0.8150	0.8425	0.9825	0.6600

Table 4.10: Simulation 5 Quantiles

Model	Method	q_{25}	q_{50}	q_{75}	q_{90}
5.A	GenCorr-T	2.0	2.0	2.0	3.0
	GenCorr-F	2.0	2.0	2.0	3.0
5.B	GenCorr-T	2.0	3.0	16.0	167.1
	GenCorr-F	2.0	2.5	11.0	93.0
5.C	GenCorr-T	2.0	3.0	17.0	137.9
	GenCorr-F	2.0	3.0	12.0	90.2

Overall, each part of Simulation 5 demonstrates that GenCorr is an accurate and reliable method for detecting causative interaction for a multivaraite response. In each case, the Frobenius norm produces results that slightly exceed in accuracy those produced when using the taxi-cab norm.

4.3.6 Conclusion on Simulations

Throughout the various simulations presented above, GenCorr under the Frobenius matrix norm consistently obtains the best empirical results. The superiority of the Frobenius norm over the taxi-cab norm is likely due to the former's aforementioned ability to more strongly emphasize a covariate's high association with the response, while at the same time de-emphasizing those covariates which have a weak association with the components of the response. It is our suggestion that the Frobenius norm be favored in any standard implementations of GenCorr.

4.3.7 Real Data Analysis

We now turn our attention to performing a real data analysis on data from a genome-wide association mapping of outbred NMRI mice. This data set comes from the work presented in [89] and is available at http://cgd.jax.org/datasets/phenotype/nmri.shtml. The response has seven components ($q = 7$), as outlined in Table 4.11. These components are observed on each of $n = 288$ individual mice and represent commonly measured phenotypic traits of mice. The values for SBP, DBP, and MAP are missing for two mice. We used the `mice` package in R (an amusing naming coincidence for sure) under the default settings to impute these values for the mice for which they are missing. For a general reference on usage of the `mice` package, see [80]. It should also be noted that we have omitted the ACR (urinary albumin-to-creatinine ratio) values for each mouse, as 259 of the 288 observed mice (nearly 90%) have ACR values equal to 0. The associated covariate space for the mice data consists of $p = 44,428$ SNPs to be examined for genetic association with the recorded phenotypic traits of the observed mice.

Table 4.11: Observed Phenotypic Traits in NMRI Mice

Trait Abbreviation	Description
SBP	Systolic blood pressure
DBP	Diastolic blood pressure
MAP	Mean arterial pressure
HDL	HDL (High-density lipoprotein) cholesterol
CHL	Total cholesterol
TRI	Triglyceride levels
GLU	Glucose levels

Our empirical analysis will be broken down into two stages. In the first stage (4.3.7), we implement an iterative screening method employing GenCorr to find main effects. We will also apply GenCorr to screen for pairwise interactive effects between all $44,428$ SNPs. In the second stage (4.3.7) of the analysis, we will outline a post-screening approach using penalized regression.

First Stage Analysis

In the first stage of this analysis we will use GenCorr-F to screen for both marginal and interactive effects. To screen for marginal effects, we use the same general iterative approach of [92], where we replace their use of DC-SIS with GenCorr. This process is done as follows:

- Apply GenCorr to the full mouse data set. Let $d = 2[n/\log n] = 102$. We select $p_1 < d$ many predictors to act as our set of initial SNPs. As suggested in [92], the value of p_1 is a value strictly between 1 and d such that a linear regression model using the top (as determined by GenCorr) p_1-many SNPs has the minimal mean squared prediction error (MSPE). We determine such a value by iterating over each possible p_1 value ($1 < p_1 < d$), fitting a linear model of $\mathbb{Y}$ regressed on the top p_1 predictors (ordered

by marginal utility score from GenCorr), and then recording the associated MSPE for each p_1. With each choice of p_1, we randomly select a training set consisting of 216 observations (75% of the observed data) to fit the model. This model is then tested on a validation set of 72 observations (25% of the observed data). In our case, we found $p_1 = 85$.

- Denote by X_1 the $n \times p_1$ matrix formed by examining the observations of only the top p_1-many covariates and denote by X_1^c the $n \times (p - p_1)$ matrix formed by the observations of the remaining $(p - p_1)$ covariates not found in the columns of X_1. Let

$$X_{\text{new}} = \left(I_n - X_1(X_1'X_1)^\dagger X_1'\right) X_1^c,$$

where $A^\dagger$ indicates the Moore-Penrose pseudo inverse of a square matrix A. This means that X_{new} will contain the residuals from regressing X_1^c onto X_1. Apply GenCorr to $\mathbb{Y}$ and X_{new}, with $\mathbb{Y}$ still acting as the matrix of responses and X_{new} acting as the matrix of predictor observations. Based on the scores obtained from GenCorr, select the top $(d - p_1)$-many SNPs, as ordered by this most recent run of GenCorr.

We now have a total of $d = 102$ SNPs, selected using an iterative application of GenCorr. This completes the first stage of marginal feature selection. Our attention is next focused on the selection of pairwise interactions. Due to computational limitations, as well as lack of a theoretical basis, we do not use an iterative approach to select pairwise interactions. Instead, we will run the interactive version of GenCorr on the full mouse data and then directly select the top d-many interactions, as ordered by GenCorr. All told we have a set of 204 features (102 marginal, 102 interactive) from this first stage of the analysis. These 204 features will be further examined in stage two of the analysis, as given below.

Second Stage Analysis

In this second analysis stage, we will take the 204 SNPs obtained in stage one and fit several elastic net [96] and lasso [76] models to the data using the `glmnet` package in R. For

further reference on this process see [30]. Three final models will be fit: elastic net with $\alpha = 0.4$, elastic net with $\alpha = 0.8$, and lasso ($\alpha = 1$). Here α is as given in the penalty function

$$((1-\alpha)/2)\,\|B\|_F^2 + \alpha \sum_{j=1}^{2d} \|B_j\|_2,$$

where B is the matrix of coefficients, $\|\cdot\|_F$ is the Frobenius matrix norm, and $\|B_j\|_2$ is a group-lasso penalty on each coefficient vector B_j (the jth row vector of B) for a single predictor or a single pairwise interaction.

In each case, we will select the model associate with the λ on the regularization path that minimizes the mean 10-fold cross validated error. The loss function used here for cross validation is the mean squared error (MSE). We will use a modified version of the corrected Akaike information criterion (AIC_c) to select what we determine to be the best candidate model (with smaller AIC_c values being considered better). The AIC_c has been shown to provide a more accurate estimation of model order than the standard AIC does "when the number of fitted parameters is a moderate to large fraction of the sample size" [7]. The usual multivariate version of the AIC_c is given as follows [71]:

$$\text{AIC}_c = -2\ln(\mathcal{L}) + \frac{2n(qk + q(q+1)/2)}{n-(k+q+1)},$$

where $\mathcal{L}$ is the maximum value of the likelihood function for the model in question. An equivalent (although unusual) definition of AIC_c can be given as follows:

$$\text{sAIC}_c = -2\left(\ln(\mathcal{L}) - \ln(\mathcal{L}_0)\right) + \frac{2n(qk + q(q+1)/2)}{n-(k+q+1)},$$

where $\mathcal{L}_0$ denotes the maximum likelihood for the null model. We will call this version of the AIC_c the *shifted*-AIC_c (abbreviated as sAIC_c). Note that for any candidate model, $\text{sAIC}_c = \text{AIC}_c + 2\ln(\mathcal{L}_0)$. Hence the ordering of candidate models given by the sAIC_c will be equivalent to the ordering given by the usual definition of AIC_c, as the former is just a constant shift of the latter. We use the sAIC_c for our model selection here, as it is easier to obtain from the results given in `glmnet`. The following lemma addresses these thoughts.

Lemma 4.3.1. *Define the deviance of a fitted model by*

$$\mathcal{D} = 2\left(\ln(\mathcal{L}_S) - \ln(\mathcal{L})\right),$$

where $\mathcal{L}_S$ is the maximum likelihood of the saturated model (the model with a free parameter for each observation). Define the null deviance to be

$$\mathcal{D}_0 = 2\left(\ln(\mathcal{L}_S) - \ln(\mathcal{L}_0)\right).$$

Given $\mathcal{D}$ and $\mathcal{D}_0$, but not the log-likelihood of the fitted model directly, we can yet obtain the $sAIC_c$.

Proof. Note that $\mathcal{D}_0 - \mathcal{D} = 2\left(\ln(\mathcal{L}) - \ln(\mathcal{L}_0)\right)$. Thus we now can determine the sAIC_c as below:

$$\text{sAIC}_c = -2\left(\ln(\mathcal{L}) - \ln(\mathcal{L}_0)\right) + \frac{2n(qk + q(q+1)/2)}{n - (k+q+1)}.$$

□

Although the `glmnet` package in R does not provide us with direct access to the log-likelihood of any candidate model, the package does allow us to obtain both $\mathcal{D}$ and $\mathcal{D}_0$ explicitly. By Lemma 4.3.1, we can thus find the associated sAIC_c for any candidate model. To alleviate the issue of having to compare sAIC_c values whose lower order (ones, tens, hundreds) place-values do not make a substantive difference in determining model ordering, we will divide each raw sAIC_c value by 1000 when reporting final results. This scaling of course does not affect the overall ordering of candidate models and makes the results easier to visually parse.

Results of Real Data Analysis.

We report the final outcome of our two stage process below in Table 4.12. We denote the three different models by their method of second stage analysis. (The first stage was the same for each model and was only performed once). As measures of model fitness, we

report the mean 10-fold cross validated MSE and the (scaled, see above) sAIC_c. We also report for each component of the response the number of features (marginal, interactive, total) retained by the individual penalized regression approaches.

Table 4.12: Results of Real Data Analysis

Second Stage	*Features*	*SBP*	*DBP*	*MAP*	*HDL*	*CHL*	*TRI*	*GLU*
El. Net ($\boldsymbol{\alpha = 0.4}$)	Marginal:	102	102	102	102	102	102	102
MSE = 5873.393	Interact:	82	82	82	79	79	81	82
$\text{sAIC}_c = -1558.766$	Total:	184	184	184	181	181	183	184
El. Net ($\boldsymbol{\alpha = 0.8}$)	Marginal:	43	43	43	43	43	43	43
MSE = 5989.832	Interact:	16	16	16	16	16	17	17
$\text{sAIC}_c = -1565.601$	Total:	59	59	59	59	59	60	60
Lasso	Marginal:	29	29	29	29	29	29	29
MSE = 5696.422	Interact:	4	4	4	4	4	4	4
$\text{sAIC}_c = -1566.418$	Total:	33	33	33	33	33	33	33

Except for the few cases in the two elastic net models where the number of retained interactions differs slightly across the components of the response, the same features were retained for each of the seven response variables. In the cases where some components were associated with slightly more non-zero coefficients than the other components, these "extra" non-zero coefficients were the only places where the selected features differ across component. When we used only a moderate mixing parameter in elastic net ($\alpha = 0.4$), few of the features are dropped from the model (and the only dropped features are interactions). This gave us around an 11% reduction in total model size. Because the aim of penalized regularization regression is to obtain a more parsimonious model, the failure to substantively reduce the model size is undesirable. When we increased the mixing parameter to $\alpha = 0.8$,

we were able to obtain a more favorable reduction in total model size. However, while such reduction in size was desirable (see for example the lower sAIC_c for this model compared to the previous model), it also came at the cost of increased mean cross validated MSE, leading us to be hesitant about fully embracing this model. The candidate model that we deemed to be the most advisable overall is that which was obtained by using lasso in the second stage of the analysis. This model possess the lowest sAIC_c value, as well as the lowest mean cross validated MSE. Moreover, this model accomplished by far the largest reduction in model size out of the the three approaches, yielding a model that is not only superior in terms of mean cross validated MSE and sAIC_c, but also salient in its parsimony.

4.4 Discussion

In this paper we proposed a new feature screening approach, called GenCorr, which is applicable to ultrahigh dimensional data with multivariate response. Our method allows us to perform both marginal and interactive screening all within the same overall methodological framework. We have demonstrated the finite performance of GenCorr under a series of empirical simulations. In the marginal case, we compared our results with those of SIRS [93] and DC-SIS [53]. We also presented a real-data analysis, showing the application of GenCorr to data originating from a GWAS setting. From a theoretical perspective, we have shown that GenCorr possess the strong sure screening property, that is, with probability converging to one asymptotically, GenCorr selects the true model exactly.

In this paper we have avoided directly selecting a cutoff for model selection. Several approaches have been proposed for selecting such a cutoff: [93] submitted one possible method for choosing a cutoff for SIRS; [43] developed another approach for choosing a selection cutoff for their Pearson chi-squared-test-based screening method; [48] present an iterative approach to producing an implicit selection cutoff. One may explore these cutoff methods further and adapt them as they see fit. We will not further pursue the topic here, however.

4.5 Proofs of Theorems 4.2.1 and 4.2.2

Here we present the proofs of Theorems 4.2.1 and 4.2.2 as presented in Section 4.2. All proofs here will be presented in the most general form of the context of screening for an r-way interaction between covariates $X_{j_1}, X_{j_2}, \ldots, X_{j_r}$. When $r = 1$, we have the necessary results for marginal effects screening. Throughout this section, $\|\cdot\|_{\mathfrak{p}}$ will refer to any $\mathfrak{p}$-norm with $\mathfrak{p}$ finite. For any positive integer k, $\|\cdot\|_{\mathfrak{p}}$ can be viewed as a function from $\mathbb{R}^k$ to $\mathbb{R}$. This means that $(\mathbb{R}^k, \|\cdot\|_{\mathfrak{p}})$ forms a normed vector space. This leads us to a routine lemma.

4.5.1 Prefacing Lemmas

Lemma 4.5.1. *Let k be any positive integer. The $\mathfrak{p}$-norm $\|\cdot\|_{\mathfrak{p}}$ is a continuous function from $\mathbb{R}^k$ to $\mathbb{R}$.*

Proof. Take any $\varepsilon > 0$. Suppose that $\{\boldsymbol{a}_N\}_{N=1}^{\infty}$ is a sequence in $\mathbb{R}^k$ with $\lim_{N\to\infty} \boldsymbol{a}_N = \boldsymbol{a}$. This means that there exists some positive integer N_0 such that whenever $N > N_0$, we have that

$$\|\boldsymbol{a}_N - \boldsymbol{a}\|_{\mathfrak{p}} < \varepsilon.$$

However, by the reverse triangle inequality, we know that

$$|\|\boldsymbol{a}_N\|_{\mathfrak{p}} - \|\boldsymbol{a}\|_{\mathfrak{p}}| \leq \|\boldsymbol{a}_N - \boldsymbol{a}\|_{\mathfrak{p}}$$

for any positive integer N. Thus there exists some positive integer N_0 (the same one as determined above in fact) such that whenever $N > N_0$, we have

$$|\|\boldsymbol{a}_N\|_{\mathfrak{p}} - \|\boldsymbol{a}\|_{\mathfrak{p}}| < \varepsilon.$$

Thus $\lim_{N\to\infty} \|\boldsymbol{a}_N\|_{\mathfrak{p}} = \|\boldsymbol{a}\|_{\mathfrak{p}}$, completing the proof. □

At multiple times throughout this section we will employ the continuous mapping theorem. For further reference on the continuous mapping theorem, see e.g. [72] and [11]. The well known weak law of large numbers [15] will also be used several times to establish the

consistency of various sample estimators. Below, we provide a slightly modified version of the weak law of large numbers.

Lemma 4.5.2. *Given an independent and identically distributed collection* $\{W_1, W_2, \ldots, W_n\}$ *of n-many samples of a random variable* W *with* $\mathbb{E}W = \mu$ *and* $Var(W) = \sigma^2 < \infty$*, define*

$$\widetilde{W} = \frac{1}{n-1}\sum_{i=1}^{n} W_i.$$

We then have that $\widetilde{W} \xrightarrow{P} \mathbb{E}W$ *as* $n \to \infty$*.*

Proof. Because the samples of W are all identically distributed, we have know that $\mathrm{Var}(W_i) = \sigma^2$ for all i. Let $\overline{W} = \frac{1}{n}\sum_{i=1}^{n} W_i$ be the usual sample mean. Due to the independence of $W_1, W_2, \ldots, W_n$, we have the following:

$$\begin{aligned}
\mathrm{Var}\left(\overline{W}\right) &= \mathrm{Var}\left(\frac{1}{n}(W_1 + W_2 + \cdots + W_n)\right) \\
&= \frac{1}{n^2}\mathrm{Var}\left(W_1 + W_2 + \cdots + W_n\right) \\
&= \frac{n\sigma^2}{n^2} \\
&= \frac{\sigma^2}{n}.
\end{aligned}$$

The common mean of each W_i is the same as the expected value of $\overline{W}$, namely $\mathbb{E}\overline{W} = \mu$. By Chebyshev's inequality and for any $\varepsilon > 0$, we have

$$\mathbb{P}\left(|\overline{W} - \mu| \geq \varepsilon\right) \leq \frac{\sigma^2}{n\varepsilon^2}.$$

This in turn implies the following:

$$\mathbb{P}\left(|\overline{W} - \mu| < \varepsilon\right) = 1 - \mathbb{P}\left(|\overline{W} - \mu| \geq \varepsilon\right) \geq 1 - \frac{\sigma^2}{n\varepsilon^2}. \tag{4.1}$$

Taking the limit as $n \to \infty$ in (4.1), we obtain

$$\lim_{n\to\infty} \mathbb{P}\left(|\overline{W} - \mu| < \varepsilon\right) \geq \lim_{n\to\infty} \left(1 - \frac{\sigma^2}{n\varepsilon^2}\right) = 1 - \lim_{n\to\infty} \frac{\sigma^2}{n\varepsilon^2} = 1.$$

Thus $\overline{W} \xrightarrow{P} \mu$. Note that $\widetilde{W} = \frac{n}{n-1}\overline{W}$. It is easily seen that the numeric sequence $\{\frac{n}{n-1}\}_{n=1}^{\infty}$ converges to one as n approaches infinity. By a standard application of the continuous mapping theorem, we know the following:

$$\widetilde{W} = \frac{n}{n-1}\overline{W} \xrightarrow{P} 1 \cdot \mu = \mu.$$

In conclusion, we have $\widetilde{W} \xrightarrow{P} \mu$, which is the desired result. □

We remind the reader of the matrix $\Sigma_{j_1,\ldots,j_r}$ as previously defined in Section 4.2. The entries of $\Sigma_{j_1,\ldots,j_r}$ are of three main forms:

- The covariance between two components of $\mathbb{Y}$. As given in Section 4.2, this covariance is estimated by the sample covariance given as follows:

$$\widehat{\mathrm{Cov}}\left(Y^{(\ell)}, Y^{(m)}\right) = \frac{1}{n-1}\sum_{i=1}^{n}\left(Y_i^{(\ell)} - \overline{Y}^{(\ell)}\right)\left(Y_i^{(m)} - \overline{Y}^{(m)}\right), \tag{4.2}$$

  where $\overline{Y}^{(\ell)}$ and $\overline{Y}^{(m)}$ are the sample means of $Y^{(\ell)}$ and $Y^{(m)}$ respectively.

- The product of the variances of each of $X_{j_1}, \ldots, X_{j_r}$. As given in Section 4.2, each variance is estimated by the sample variance given below:

$$\widehat{\mathrm{Var}}\left(X_{j_s}\right) = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_{ij_s} - \overline{X}_{j_s}\right)^2, \tag{4.3}$$

  where $\overline{X}_{j_s}$ is the standard sample mean of X_{j_s}.

- The $(r+1)$-way joint cumulant between a component of $\mathbb{Y}$ and the random variables $X_{j_1}, \ldots, X_{j_r}$. The $(r+1)$-way joint cumulant can be estimated as follows:

$$\widehat{\kappa}_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}) = \frac{1}{n}\sum_{i=1}^{n}\left[\prod_{s=1}^{r}\left(X_{ij_s} - \overline{X}_{j_s}\right)\right]\left(Y_i^{(m)} - \overline{Y}^{(m)}\right), \tag{4.4}$$

where $\overline{X}_{j_s}$ represents the sample mean of X_{j_s} and $\overline{Y}^{(m)}$ is the sample mean of $Y^{(m)}$.

It is an overall straightforward exercise to verify that each of (4.2), (4.3), and (4.4) are consistent estimators of their associated population parameters. We present this in the form of a three-part lemma.

Lemma 4.5.3. *With the components of $\mathbb{Y}$ being given by $Y^{(m)}$ $(m = 1, 2, \ldots, q)$, and each covariate in question being denoted by X_{j_s} $(s = 1, 2, \ldots, r)$, we have the following results:*

1. $\widehat{Cov}\left(Y^{(\ell)}, Y^{(m)}\right) \xrightarrow{P} Cov\left(Y^{(\ell)}, Y^{(m)}\right)$;

2. $\prod_{s=1}^{r} \widehat{Var}\,(X_{j_s}) \xrightarrow{P} \prod_{s=1}^{r} Var\,(X_{j_s})$;

3. $\widehat{\kappa}_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}) \xrightarrow{P} \kappa_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r})$.

Proof. Statement 1 can be demonstrated as follows: Expanding the right-hand side of (4.2), we have

$$\begin{aligned}\widehat{\mathrm{Cov}}\left(Y^{(\ell)}, Y^{(m)}\right) = \frac{1}{n-1}\sum_{i=1}^{n} Y_i^{(\ell)} Y_i^{(m)} - \frac{1}{n-1}\sum_{i=1}^{n} Y_i^{(\ell)}\overline{Y}^{(m)} \\ - \frac{1}{n-1}\sum_{i=1}^{n} \overline{Y}^{(\ell)} Y_i^{(m)} + \frac{1}{n-1}\sum_{i=1}^{n} \overline{Y}^{(\ell)}\overline{Y}^{(m)}\end{aligned} \tag{4.5}$$

By applying the weak law of large numbers as presented in Lemma 4.5.2 term-wise to the right hand side of (4.5), we obtain

$$\frac{1}{n-1}\sum_{i=1}^{n} Y_i^{(\ell)} Y_i^{(m)} \xrightarrow{P} \mathbb{E}\left(Y^{(\ell)} Y^{(m)}\right)$$

$$\frac{1}{n-1}\sum_{i=1}^{n} Y_i^{(\ell)}\overline{Y}^{(m)} \xrightarrow{P} \mathbb{E}\left(Y^{(\ell)}\right)\mathbb{E}\left(Y^{(m)}\right)$$

$$\frac{1}{n-1}\sum_{i=1}^{n} \overline{Y}^{(\ell)} Y_i^{(m)} \xrightarrow{P} \mathbb{E}\left(Y^{(\ell)}\right)\mathbb{E}\left(Y^{(m)}\right)$$

$$\frac{1}{n-1}\sum_{i=1}^{n} \overline{Y}^{(\ell)}\overline{Y}^{(m)} \xrightarrow{P} \mathbb{E}\left(Y^{(\ell)}\right)\mathbb{E}\left(Y^{(m)}\right).$$

By the continuous mapping theorem applied to addition and subtraction of estimators, we now have

$$\begin{aligned}\widehat{\mathrm{Cov}}\left(Y^{(\ell)}, Y^{(m)}\right) &\xrightarrow{P} \left[\mathbb{E}\left(Y^{(\ell)}Y^{(m)}\right) - 2\mathbb{E}\left(Y^{(\ell)}\right)\mathbb{E}\left(Y^{(m)}\right) + \mathbb{E}\left(Y^{(\ell)}\right)\mathbb{E}\left(Y^{(m)}\right)\right] \\ &= \mathbb{E}\left(Y^{(\ell)}Y^{(m)}\right) - \mathbb{E}\left(Y^{(\ell)}\right)\mathbb{E}\left(Y^{(m)}\right) \\ &= \mathrm{Cov}\left(Y^{(\ell)}, Y^{(m)}\right).\end{aligned}$$

Statement 2 is a direct corollary of statement 1. Each multiplicand in the product $\prod_{s=1}^{r}\widehat{\mathrm{Var}}\left(X_{j_s}\right)$ can be shown to converge in probability to the associated multiplicand in $\prod_{s=1}^{r}\mathrm{Var}\left(X_{j_s}\right)$. This is done by viewing $\mathrm{Var}\left(X_{j_s}\right)$ as the covariance between X_{j_s} and itself, then proceeding quite similarly as was done in proving statement 1 of the lemma. This gives us that $\widehat{\mathrm{Var}}\left(X_{j_s}\right) \xrightarrow{P} \mathrm{Var}\left(X_{j_s}\right)$ for each $s = 1, 2, \ldots, r$. By a simple application of the continuous mapping theorem, the product of these estimators of the variances converges in probability to the product $\prod_{s=1}^{r}\mathrm{Var}\left(X_{j_s}\right)$.

Statement 3 can be established using a similar approach to that of statement 1. By expanding the product in the right hand side of (4.4) we obtain terms with one of the two following general forms, with the covariates $X_{j_1}, X_{j_2}, \ldots, X_{j_s}$ being (without loss of generality) ordered here to simplify the notation:

$$(-1)^{r-s}\ \frac{1}{n}\sum_{i=1}^{n} X_{ij_1}X_{ij_2}\cdots X_{ij_s}\overline{X}_{j_{(s+1)}}\overline{X}_{j_{(s+2)}}\cdots\overline{X}_{j_r}Y_i^{(m)} \tag{4.6}$$

or

$$(-1)^{r-s-1}\ \frac{1}{n}\sum_{i=1}^{n} X_{ij_1}X_{ij_2}\cdots X_{ij_s}\overline{X}_{j_{(s+1)}}\overline{X}_{j_{(s+2)}}\cdots\overline{X}_{j_r}\overline{Y}^{(m)}. \tag{4.7}$$

The terms (4.6) and (4.7) represent the general form of each summand of the right hand side of (4.4) when expanded out fully. By a standard application of the (more traditional) weak law of large numbers, we have the following:

$$(4.6) \xrightarrow{P} (-1)^{r-s}\, \mathbb{E}\left(X_{j_1} X_{j_2} \cdots X_{j_s} Y^{(m)}\right) \mathbb{E}\left(X_{j_{(s+1)}}\right) \mathbb{E}\left(X_{j_{(s+2)}}\right) \cdots \mathbb{E}\left(X_{j_r}\right);$$

$$(4.7) \xrightarrow{P} (-1)^{r-s-1}\, \mathbb{E}\left(X_{j_1} X_{j_2} \cdots X_{j_s}\right) \mathbb{E}\left(X_{j_{(s+1)}}\right) \mathbb{E}\left(X_{j_{(s+2)}}\right) \cdots \mathbb{E}\left(X_{j_r}\right) \mathbb{E}\left(Y^{(m)}\right).$$

Remembering that

$$\kappa_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}) = \mathbb{E}\left(\left(Y^{(m)} - \mathbb{E}Y^{(m)}\right) \prod_{s=1}^{r} \left(X_{j_s} - \mathbb{E}X_{j_s}\right)\right)$$

and combining the above statements on the convergence of (4.6) and (4.7), then applying the continuous mapping theorem, we finally obtain

$$\widehat{\kappa}_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}) \xrightarrow{P} \kappa_{r+1}(Y^{(m)}, X_{j_1}, \ldots, X_{j_r}).$$

This completes the proof of the three statements of the lemma. □

Lemma 4.5.4. *As an estimator of* $\Phi_{j_1,\ldots,j_r}$, $\widehat{\Phi}_{j_1,\ldots,j_r}$ *is consistent.*

Proof. Lemma 4.5.3 establishes that every entry of $\widehat{\Sigma}_{j_1,\ldots,j_r}$ is a consistent estimator of the associated entry of $\Sigma_{j_1,\ldots,j_r}$. By the continuous mapping theorem, this means that every entry in the matrix

$$\widehat{\mathcal{H}}_{j_1,\ldots,j_r} = \left[\operatorname{diag}\left(\widehat{\Sigma}_{j_1,\ldots,j_r}\right)\right]^{-1/2} \widehat{\Sigma}_{j_1,\ldots,j_r} \left[\operatorname{diag}\left(\widehat{\Sigma}_{j_1,\ldots,j_r}\right)\right]^{-1/2}$$

as defined previously in Section 4.2 is a consistent estimators of the entries of

$$\mathcal{H}_{j_1,\ldots,j_r} = \left[\operatorname{diag}\left(\Sigma_{j_1,\ldots,j_r}\right)\right]^{-1/2} \Sigma_{j_1,\ldots,j_r} \left[\operatorname{diag}\left(\Sigma_{j_1,\ldots,j_r}\right)\right]^{-1/2}.$$

This can be seen by first noting that products and sums of consistent estimators are also consistent estimators themselves and that the diagonal entries of $\Sigma_{j_1,\ldots,j_r}$ are all positive. As the functions $f(t) = 1/t$ and $g(t) = \sqrt{t}$ are both continuous for positive values of t, the

continuous mapping theorem indeed ultimately gives the results on the consistency of the entries of $\widehat{\mathcal{H}}_{j_1,\ldots,j_r}$ as estimators of the respective entries of $\mathcal{H}_{j_1,\ldots,j_r}$.

Note that for our specific application here, $\|\cdot\|_{\mathfrak{p}}$ is a function of $(q+1)^2$-many variables. This comes from the fact that $\mathcal{H}_{j_1,\ldots,j_r}$ is a $(q+1)\times(q+1)$ dimensional matrix. As such, by Lemma 4.5.1 we can apply $\|\cdot\|_{\mathfrak{p}}$ to the matrix $\widehat{\mathcal{H}}_{j_1,\ldots,j_r}$ and obtain (again by continuous mapping theorem) that $\widehat{\Phi}_{j_1,\ldots,j_r}$ is a consistent estimator of $\Phi_{j_1,\ldots,j_r}$. □

4.5.2 Proofs of Main Results

Before proceeding further, we remind the reader of the definition of the following constant:

$$\gamma = (q+1) + \sum_{1\leq \ell,m\leq q} |\rho_{\ell m}|^2,$$

where $\rho_{\ell m}$ is the correlation between $Y^{(\ell)}$ and $Y^{(m)}$. This constant will be referenced several times throughout the proofs of Theorems 4.2.1 and 4.2.2.

The proofs of the main results will be presented in four steps.

- Step 1: We will show that there exists a positive constant $\Phi_{\min}$ such that for any tuple $(j_1,\ldots,j_r)\in\mathcal{S}_T$,

$$\Phi_{j_1,\ldots,j_r} > \Phi_{\min} > \sqrt{\gamma} > 0.$$

  (Note that this is also Corollary 4.2.3). In Condition (C2), we defined the value $\omega_{j_1,\ldots,j_r} > 0$ such that for some component, $Y^{(m)}$, of $\mathbb{Y}$, we have

$$\left|\frac{\kappa_{r+1}\left(Y^{(m)}, X_{j_1},\ldots,X_{j_r}\right)}{\sigma_{(m)}\sigma_{j_1}\cdots\sigma_{j_r}}\right| > \omega_{j_1,\ldots,j_r} > 0,$$

  whenever $(j_1,\ldots,j_r)$ is in $\mathcal{S}_T$. As an aside, note that $|\mathcal{S}_T| \leq \binom{p}{r} < \infty$, as there can only be $\binom{p}{r}$ many r-way interactions formed among p-many covariates. Define a value $\omega_{\min}$ as follows:

$$\omega_{\min} = \min_{(j_1,\ldots,j_r)\in\mathcal{S}_t} \{\omega_{j_1,\ldots,j_r}\}.$$

Because each $\omega_{j_1,\ldots,j_r}$ is positive and because $|\mathcal{S}_T| \leq \infty$, the minimum is a well defined and positive value here. In other words, we have some $\omega_{\min} > 0$ such that $\omega_{j_1,\ldots,j_r} > \omega_{\min}$ for all $(j_1,\ldots,j_r) \in \mathcal{S}_T$. It therefore can be seen that for any $(j_1,\ldots,j_r) \in \mathcal{S}_T$ we have the following chain of inequalities:

$$\begin{aligned}
\Phi_{j_1,\ldots,j_r} &= \|\mathcal{H}_{j_1,\ldots,j_r}\|_{\mathfrak{p}} \\
&\geq \sqrt{(\omega_{j_1,\ldots,j_r})^2 + \gamma} \\
&> \sqrt{\omega_{\min}^2/2 + \gamma} \\
&> \sqrt{\gamma}.
\end{aligned}$$

Let $\Phi_{\min} = \sqrt{\omega_{\min}^2/2 + \gamma}$. Then, for all r-tuples in the true model,

$$\Phi_{j_1,\ldots,j_r} > \Phi_{\min} > \sqrt{\gamma} > 0.$$

This completes Step 1, as well as proves Corollary 4.2.3.

- Step 2: In line with the statement of Corollary 4.2.4, we will now show that $\widehat{\Phi}_{j_1,\ldots,j_r}$ is a uniformly consistent estimator of $\Phi_{j_1,\ldots,j_r}$. Lemma 4.5.3 tells us that $\widehat{\Phi}_{j_1,\ldots,j_r}$ is a consistent estimator of $\Phi_{j_1,\ldots,j_r}$. Thus for any $(j_1,\ldots,j_r) \in \mathcal{I}$ and any $\varepsilon > 0$, we know

$$\lim_{n\to\infty} \mathbb{P}\left(\left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r}\right| > \varepsilon\right) = 0.$$

Let $(J_1,\ldots,J_r)$ be the r-tuple in $\mathcal{I}$ that maximizes $\left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r}\right|$. By Lemma 4.5.3, we know that

$$\lim_{n\to\infty} \mathbb{P}\left(\left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{J_1,\ldots,J_r}\right| > \varepsilon\right) = 0.$$

Thus, for any $\varepsilon > 0$, we have

$$\lim_{n\to\infty} \mathbb{P}\left(\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left| \widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r} \right| > \varepsilon \right) = 0.$$

This establishes that $\widehat{\Phi}_{j_1,\ldots,j_r}$ is a uniformly consistent estimator of $\Phi_{j_1,\ldots,j_r}$. This completes Step 2, as well as the proof of Corollary 4.2.4.

- Step 3: We now show that there is a positive constant c such that

$$\lim_{n\to\infty} \mathbb{P}\left(\mathcal{S}_T \subseteq \widehat{\mathcal{S}} \right) = 1.$$

Let $c = \Phi_{\min}$. By way of contradiction, suppose that this selection of c does not result in $\mathcal{S}_T \subseteq \widehat{S}$. This means that we can find some $(j_1^*, \ldots, j_r^*) \in \mathcal{S}_T$, while at the same time $(j_1^*, \ldots, j_r^*) \notin \widehat{\mathcal{S}}$. By the definition of $\widehat{\mathcal{S}}$, we then know that

$$\widehat{\Phi}_{j_1^*,\ldots,j_r^*} \leq \Phi_{\min},$$

and also

$$\Phi_{j_1^*,\ldots,j_r^*} > \Phi_{\min}.$$

Thus there is some positive number $\zeta > 0$ such that

$$\left| \widehat{\Phi}_{j_1^*,\ldots,j_r^*} - \Phi_{j_1^*,\ldots,j_r^*} \right| > \zeta > 0$$

It now follows that

$$\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left| \widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r} \right| \geq \left| \widehat{\Phi}_{j_1^*,\ldots,j_r^*} - \Phi_{j_1^*,\ldots,j_r^*} \right| > \zeta.$$

However, by uniform consistency shown in Step 2,

$$\mathbb{P}\left(\mathcal{S}_T \not\subseteq \widehat{\mathcal{S}} \right) \leq \mathbb{P}\left(\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left| \widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r} \right| > \zeta \right) \to 0, \text{ as } n \to \infty.$$

This contradicts are previous supposition of non-containment. Thus we have

$$\lim_{n\to\infty} \mathbb{P}\left(\mathcal{S}_T \subseteq \widehat{\mathcal{S}}\right) = 1,$$

finishing the proof to Theorem 4.2.1 and showing the forward direction to Theorem 4.2.2.

- Step 4: We now show the reverse direction for Theorem 4.2.2. Suppose once again by way of contradiction that $\widehat{\mathcal{S}} \not\subseteq \mathcal{S}_T$. Then there exists an r-tuple $(j_1^*, \ldots, j_r^*) \in \widehat{\mathcal{S}}$, yet $(j_1^*, \ldots, j_r^*) \notin \mathcal{S}_T$. It follows that

$$\widehat{\Phi}_{j_1^*,\ldots,j_r^*} > \Phi_{\min} > \sqrt{\gamma}.$$

However, by Condition (C3), we assumed that $\Phi_{j_1^*,\ldots,j_r^*} = \sqrt{\gamma}$. Let $\varepsilon = \left(\Phi_{\min} - \sqrt{\gamma}\right)/2$. This value of ε is verifiably greater than zero. Note that

$$\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r}\right| \geq \left|\widehat{\Phi}_{j_1^*,\ldots,j_r^*} - \Phi_{j_1^*,\ldots,j_r^*}\right| > \varepsilon.$$

However, by uniform consistency, we also have

$$\mathbb{P}\left(\widehat{\mathcal{S}} \not\subseteq \mathcal{S}_T\right) \leq \mathbb{P}\left(\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r}\right| > \varepsilon\right) \to 0, \text{ as } n \to \infty.$$

This causes a contradiction with our assumption of non-containment and

$$\max_{(j_1,\ldots,j_r)\in\mathcal{I}} \left|\widehat{\Phi}_{j_1,\ldots,j_r} - \Phi_{j_1,\ldots,j_r}\right| > \varepsilon.$$

Therefore, we can in fact conclude that

$$\lim_{n\to\infty} \mathbb{P}\left(\widehat{\mathcal{S}} \subseteq \mathcal{S}_T\right) = 1.$$

The combination of Steps 3 and 4 shows that the selection of $c = \Phi_{\min}$ yields the full result of Theorem 4.2.2, namely

$$\lim_{n\to\infty} \mathbb{P}\left(\mathcal{S}_T = \widehat{\mathcal{S}}\right) = 1.$$

This completes the requisite proofs.

CHAPTER 5

Final Comments and Discussion

5.1 Summary

In this work we have introduced three new methods for feature screening ultrahigh dimensional data. Each of these approaches was motivated by an existing, yet heretofore unanswered, need in current feature screening literature. In Chapter 2 we examined a method (TC-SIS) for marginal feature screening when both the covariates and the response are categorical. Via a series of Monte Carlo simulations, we demonstrated that our method consistently outperforms current methods by requiring the smallest mean minimum model size to guarantee acquisition of the true model. We also showed the tractability of TC-SIS on real world data by applying our method to an ultrahigh dimensional data set pertaining to polycystic ovary syndrome (PCOS). This chapter closed with a proof of the strong sure screening property as it relates to TC-SIS.

Chapter 2 was followed by Chapter 3 wherein we developed a new method (JCIS) for screening ultrahigh dimensional data sets for interactions. JCIS is unique among most all extant interaction screening methods in that it does not require first screening for marginal effects and can thus admit all relevant interactions, regardless of whether any component of said interactions was found to be important marginally. This ability of JCIS to detect strong interactive effects among predictors with weak marginal effects is critical due to the prevalence of data where individual features exhibit negligible effect on the response, but in tandem effect an important difference in the response. Once again, we show the computational viability of JCIS by presenting several simulation examples. In each case, JCIS drastically outperformed current interaction screening methods for both categorical, as well as continuous data. Of particular note, we also exhibited a real data analysis on the same PCOS data from above, with the added difficulty of screening for pairwise interactions. Even

with the increased weight of examining several billion possible interactions, JCIS proved computationally capable of handling such a monumental task. Like was the case with TC-SIS, we concluded by proving that JCIS possesses the strong sure screening property.

Our third and final project was presented in Chapter 4. There we once again considered the task of marginal and interaction screening of ultrahigh dimensional feature spaces, but now with the added hurdle of the response being multivariate. This expanded allowance for multivariate response permits us to utilize the more robust analytic insights that can come from examining all components of the response as an interdependent whole rather than individual and disparate parts. We called our method GenCorr in reference to its use of the generalized correlation matrix. Importantly, GenCorr was shown to be the only existing method for multivariate screening that can be directly applied to interaction screening, manifesting the unique applicability of our method in the sphere of feature screening. The effectiveness of GenCorr compared to two other multivariate screening methods was demonstrated over a large collection of empirical simulations. We also presented a real data analysis on a real data set examining the effect of genotype on phenotypic traits in mice. This chapter was closed by our confirming formally that GenCorr exhibits the strong sure screening property for acquisition of the true model.

5.2 Future Work

While the contributions of this work have been no doubt salient, important advances in ultrahigh dimensional feature screening can still be made. Herein we have only examined two-way interaction screening. When data sets are large ($p > 100,000$) and p is much larger than the sample size n, computational limitations often inhibit screening for more than two-way interactions. As computational power improves and with increased focus on optimizing some of the methods proposed in this work, one could explore further availability for higher order interaction screening.

In [56], the authors consider the problem of multivariate response, yet with the number of components to the response being ultrahigh dimensional in relation to the sample size. Future work could consider feature screening methods in such a context. With an added

hurdle of also assuming the covariate space was ultrahigh dimensional, a large sector of unexplored methods could be developed. In line with these thoughts, one new area of possible exploration could be that of response optimization, where a small set of response components are chosen for optimal fit with a number of covariate predictors. The use of the generalized correlation matrix may prove useful in this pursuit.

One other route that has yet to be explored is the application of DC-SIS [53] to interaction screening by generalizing the concept of distance correlation to a function admitting three arguments, much like unto what was done with the JCIS method of Chapter 3. This could allow for the desirable model free aspects of distance correlation to be applied in the interaction screening setting.

REFERENCES

[1] Alan Agresti. *An Introduction to Categorical Data Analysis*. Wiley, Hoboken, NJ, 2 edition, 2007.

[2] Carl A Anderson, Fredrik H Pettersson, Geraldine M Clarke, Lon R Cardon, Andrew P Morris, and Krina T Zondervan. Data quality control in genetic case-control association studies. *Nature protocols*, 5(9):1564–1573, 2010.

[3] P. Armitage. Tests for linear trends in proportions and frequencies. *Biometrics*, 11(3):375–386, 1955.

[4] Krishnakumar Balasubramanian, Bharath K. Sriperumbudur, and Guy Lebanon. Ultrahigh dimensional feature screening via rkhs embeddings. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 31, pages 126–134, Scottsdale, AZ, USA., 2013.

[5] David J Balding. A tutorial on statistical methods for population association studies. *Nature Reviews Genetics*, 7(10):781, 2006.

[6] Stephen Willis Barton. Tutorial for using the center for high performance computing at the University of Utah and an example using random forest. Master's thesis, Utah State University, Logan, UT, 12 2016.

[7] Edward J. Bedrick and Chih-Ling Tsai. Model selection for multivariate regression in small samples. *Biometrics*, 50(1):226–231, 1994.

[8] Harald Binder and Martin Schumacher. Allowing for mandatory covariates in boosting estimation of sparse high-dimensional survival models. *BMC bioinformatics*, 9(1):14, 2008.

[9] G.E. Bredon. *Topology and Geometry*, volume 139 of *Graduate texts in mathematics*. Springer-Verlag, 3rd edition, 1993.

[10] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] G. Casella and R.L. Berger. *Statistical Inference*. Duxbury advanced series in statistics and decision sciences. Thomson Learning, 2002.

[12] E.C. Cho and M.J. Cho. Variance of sample variance. In *JSM Proceedings: Survey Research Methods Section*, pages 1291–1293, Alexandria, VA, 2008. American Statistical Association.

[13] Nam Hee Choi, William Li, and Ji Zhu. Variable selection with the strong heredity constraint and its oracle property. *Journal of the American Statistical Association*, 105(489):354–364, 2010.

[14] Wanghuan Chu, Runze Li, and Matthew Reimherr. Feature screening for time-varying coefficient models with ultrahigh dimensional longitudinal data. *The annals of applied statistics*, 10(2):596, 2016.

[15] E. Çınlar. *Probability and Stochastics*. Graduate Texts in Mathematics. Springer New York, 2011.

[16] William G. Cochran. Some methods for strengthening the common χ^2 tests. *Biometrics*, 10(4):417–451, 1954.

[17] Heather J Cordell. Detecting gene–gene interactions that underlie human diseases. *Nature Reviews Genetics*, 10(6):392, 2009.

[18] Hengjian Cui, Runze Li, and Wei Zhong. Model-free feature screening for ultrahigh dimensional discriminant analysis. *Journal of the American Statistical Association*, 110(510):630–641, 2015.

[19] J. Fan and Y. Fan. High-dimensional classification using features annealed independence rules. *The Annals of Statistics*, 36(6):2605–2637, 2008.

[20] J. Fan, Y. Feng, and R. Song. Nonparametric independence screening in sparse ultra-high dimensional additive models. *Journal of American Statistical Association*, 106(494):544–557, 2011.

[21] J. Fan, F. Han, and H. Liu. Challenges of big data analysis. *National Science Review*, 1:293–314, 2014.

[22] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96:1348–1360, 2001.

[23] J. Fan and R Li. Statistical challenges with high dimensionality: feature selection in knowledge discovery. In M. Sanz-Sole, J. Soria, J.L. Varona, and J. Verdera, editors, *Proceedings of the International Congress of Mathematicians*, volume III, pages 595–622, Zurich, 2006. European Mathematical Society.

[24] J. Fan and J. Lv. Sure independence screening for ultrahigh dimensional feature space. *Journal of the Royal Statistical Society, Series B*, 70(5):849–911, 2008.

[25] J. Fan, R. Samworth, and Y. Wu. Ultrahigh dimensional variable selection: beyond the linear model. *Journal of Machine Learning Research*, 10(3):1829–1853, 2009.

[26] J. Fan and R. Song. Sure independence screening in generalized linear models with np-dimensionality. *Annals of Statistics*, 38(6):3567–3604, 2010.

[27] Jianqing Fan and Jinchi Lv. A selective overview of variable selection in high dimensional feature space. *Statistica Sinica*, 20(1):101–148, 2010.

[28] Yingying Fan, Yinfei Kong, Daoji Li, Zemin Zheng, et al. Innovated interaction screening for high-dimensional nonlinear classification. *The Annals of Statistics*, 43(3):1243–1272, 2015.

[29] Yao-Hwei Fang, Jie-Huei Wang, and Chao A Hsiung. Tsgsis: a high-dimensional grouped variable selection approach for detection of whole-genome snp–snp interactions. *Bioinformatics*, 33(22):3595–3602, 2017.

[30] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[31] Guifang Fu, Gang Wang, and Xiaotian Dai. An adaptive threshold determination method of feature screening for genomic selection. *BMC Bioinformatics*, 18(1):212, 2017.

[32] Damian Gola, Jestinah M. Mahachie John, Kristel van Steen, and Inke R. König. A roadmap to multifactor dimensionality reduction methods. *Briefings in Bioinformatics*, 17(2):293–308, 2016.

[33] Kirk Gosik, Lan Kong, Vernon M Chinchilli, and Rongling Wu. iform/eqtl: an ultrahigh-dimensional platform for inferring the global genetic architecture of gene transcripts. *Briefings in bioinformatics*, 18(2):250–259, 2017.

[34] Casey S. Greene, Daniel S. Himmelstein, Heather H. Nelson, Karl T. Kelsey, Scott M. Williams, Angeline S. Andrew, Margaret R. Karagas, and Jason H. Moore. *Enabling Personal Genomics With an Explicit Test of Epistasis*, pages 327–336. World Scientific, 2012.

[35] Guoyu Guan, Jianhua Guo, and Hansheng Wang. Varying naïve bayes models with applications to classification of chinese text documents. *Journal of Business and Economic Statistics*, 32(3):445–456, 2014.

[36] Lance W. Hahn, Marylyn D. Ritchie, and Jason H. Moore. Multifactor dimensionality reduction software for detecting gene-gene and gene-environment interactions. *Bioinformatics*, 19(3):376–382, 2001.

[37] Ning Hao, Yang Feng, and Hao Helen Zhang. Model selection for high-dimensional quadratic regression via regularization. *Journal of the American Statistical Association*, 0(0):1–11, 2018.

[38] Ning Hao and Hao Helen Zhang. Interaction screening for ultrahigh-dimensional data. *Journal of the American Statistical Association*, 109(507):1285–1301, 2014.

[39] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, New York, 2 edition, 2009.

[40] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 2nd edition, 2012.

[41] F Hoti and MJ Sillanpää. Bayesian mapping of genotype× expression interactions in quantitative and qualitative traits. *Heredity*, 97(1):4, 2006.

[42] Sau-Lon James Hu. Probabilistic independence and joint cumulants. *Journal of Engineering Mechanics*, 117(3):640–652, 1991.

[43] Danyang Huang, Runze Li, and Hansheng Wang. Feature screening for ultrahigh dimensional categorical data with applications. *Journal of Business and Economic Statistics*, 32(2):237–244, 2014.

[44] Hanni P Kärkkäinen, Zitong Li, and Mikko J Sillanpää. An efficient genome-wide multilocus epistasis search. *Genetics*, 201(3):865–870, 2015.

[45] Hanni P Kärkkäinen and Mikko J Sillanpää. Back to basics for bayesian model building in genomic selection. *Genetics*, 191(3):969–987, 2012.

[46] Hyunsoo Kim, Peg Howland, and Haesun Park. Dimension reduction in text classification with support vector machines. *Journal of Machine Learning Research*, 6(Jan):37–53, 2005.

[47] Yongdai Kim, Sunghoon Kwon, and Hosik Choi. Consistent model selection criteria on high dimensions. *Journal of Machine Learning Research*, 13(Apr):1037–1057, 2012.

[48] Jing Kong, Sijian Wang, and Grace Wahba. Using distance covariance for improved variable selection with application to learning genetic risk models. *Statistics in Medicine*, 34(10):1708–1720, 2015.

[49] Michael H Kutner, Chris Nachtsheim, and John Neter. *Applied linear regression models*. McGraw-Hill/Irwin, 4th edition, 2004.

[50] S. Lang. *Real and Functional Analysis*. Number 142 in Graduate Texts in Mathematics. Springer New York, 1993.

[51] Jiahan Li, Kiranmoy Das, Guifang Fu, Runze Li, and Rongling Wu. The bayesian lasso for genome-wide association studies. *Bioinformatics*, 27(4):516–523, 2010.

[52] Jiahan Li, Wei Zhong, Runze Li, and Rongling Wu. A fast algorithm for detecting gene–gene interactions in genome-wide association studies. *The annals of applied statistics*, 8(4):2292, 2014.

[53] R Li, W Zhong, and L Zhu. Feature screening via distance correlation learning. *Journal of the American Statistical Association*, 107(499):1129–1139, 2012.

[54] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[55] JingYuan Liu, Wei Zhong, and RunZe Li. A selective overview of feature screening for ultrahigh-dimensional data. *Science China Mathematics*, 58(10):1–22, 2015.

[56] Yingying Ma, Wei Lan, and Hansheng Wang. Testing predictor significance with ultra high dimensional multivariate responses. *Computational Statistics & Data Analysis*, 83:275–286, 2015.

[57] M. Manetti. *Topology*. UNITEXT. Springer International Publishing, 2015.

[58] H. B. Mann and A. Wald. On stochastic limit and order relationships. *The Annals of Mathematical Statistics*, 14(3):217–226, 1943.

[59] Teri A. Manolio, Francis S. Collins, Nancy J. Cox, David B. Goldstein, Lucia A. Hindorff, David J. Hunter, Mark I. McCarthy, Erin M. Ramos, Lon R. Cardon, Aravinda Chakravarti, Judy H. Cho, Alan E. Guttmacher, Augustine Kong, Leonid Kruglyak,

Elaine Mardis, Charles N. Rotimi, Montgomery Slatkin, David Valle, Alice S. Whittemore, Michael Boehnke, Andrew G. Clark, Evan E. Eichler, Greg Gibson, Jonathan L. Haines, Trudy F C MacKay, Steven A. McCarroll, and Peter M. Visscher. Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753, 10 2009.

[60] Jonathan Marchini, Peter Donnelly, and Lon R Cardon. Genome-wide strategies for detecting multiple loci that influence complex diseases. *Nature genetics*, 37(4):413, 2005.

[61] Jonathan Marchini, Bryan Howie, Simon Myers, Gil McVean, and Peter Donnelly. A new multipoint method for genome-wide association studies by imputation of genotypes. *Nature genetics*, 39(7):906, 2007.

[62] Alexandru Nica and Roland Speicher. Lectures on the combinatorics of free probability. In *London Mathematical Society Lecture Note Series*, volume 335. Cambridge University Press, Oxford, UK, 2006.

[63] Yue S. Niu, Ning Hao, and Lingling An. Detection of rare functional variants using group isis. *BMC Proceedings*, 5(9):S108, Nov 2011.

[64] Kristine A. Pattin, Bill C. White, Nate Barney, Jiang Gui, Heather H. Nelson, Karl T. Kelsey, Angeline S. Andrew, Margaret R. Karagas, and Jason H. Moore. A computationally efficient hypothesis testing method for epistasis analysis using multifactor dimensionality reduction. *Genetic Epidemiology*, 33(1):87–94, 2009.

[65] Patrick C Phillips. Epistasis-the essential role of gene interactions in the structure and evolution of genetic systems. *Nature Reviews Genetics*, 9(11):855, 2008.

[66] Monnat Pongpanich, Patrick F. Sullivan, and Jung-Ying Tzeng. A quality control algorithm for filtering snps in genome-wide association studies. *Bioinformatics*, 26(14):1731–1737, 2010.

[67] T Popoviciu. Sur les équations algébriques ayant toutes leurs racines réelles. *Mathematica (Cluj)*, 9:129–145, 1935.

[68] Marylyn D. Ritchie, Lance W. Hahn, Nady Roodi, L. Renee Bailey, William D. Dupont, Fritz F. Parl, and Jason H. Moore. Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. *The American Journal of Human Genetics*, 69(1):138 – 147, 2001.

[69] Murat Sariyar, Isabell Hoffmann, and Harald Binder. Combining techniques for screening and evaluating interaction terms on high-dimensional time-to-event data. *BMC bioinformatics*, 15(1):58, 2014.

[70] Daniel F Schwarz, Inke R König, and Andreas Ziegler. On safari to random jungle: a fast implementation of random forests for high-dimensional data. *Bioinformatics*, 26(14):1752–1758, 2010.

[71] Abd-Krim Seghouane. New aic corrected variants for multivariate linear regression model selection. *IEEE Transactions on Aerospace and Electronic Systems*, 47(2):1154–1165, 2011.

[72] R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, New York, 1980.

[73] Shiquan Sun, Qinke Peng, and Adnan Shakoor. A kernel-based multivariate feature selection method for microarray data classification. *PloS one*, 9(7):e102541, 2014.

[74] Gábor J. Székely and Maria L. Rizzo. Brownian distance covariance. *Annals of Applied Statistics*, 3(4):1236–1265, 12 2009.

[75] Gábor J. Székely, Maria L. Rizzo, and Nail K. Bakirov. Measuring and testing dependence by correlation of distances. *Ann. Statist.*, 35(6):2769–2794, 12 2007.

[76] R. Tibshirani. Regression shrinkage and selection via lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–288, 1996.

[77] Robert Tibshirani, Martin Wainwright, and Trevor Hastie. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

[78] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

[79] Masao Ueki and Gen Tamiya. Ultrahigh-dimensional variable selection method for whole-genome gene-gene interaction analysis. *BMC bioinformatics*, 13(1):72, 2012.

[80] Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software, Articles*, 45(3):1–67, 2011.

[81] Digna R. Velez, Bill C. White, Alison A. Motsinger, William S. Bush, Marylyn D. Ritchie, Scott M. Williams, and Jason H. Moore. A balanced accuracy function for epistasis modeling in imbalanced datasets using multifactor dimensionality reduction. *Genetic Epidemiology*, 31(4):306–315, 2007.

[82] Hansheng Wang. Forward regression for ultra-high dimensional variable screening. *Journal of the American Statistical Association*, 104(488):1512–1524, 2009.

[83] Wen-Hua Wei, Gibran Hemani, and Chris S Haley. Detecting epistasis in human complex traits. *Nature Reviews Genetics*, 15(11):722, 2014.

[84] Stacey J. Winham and Alison A. Motsinger-Reif. An r package implementation of multifactor dimensionality reduction. *BioData Mining*, 4(1):24, Aug 2011.

[85] Chen Xu and Jiahua Chen. The sparse mle for ultrahigh-dimensional feature screening. *Journal of the American Statistical Association*, 109(507):1257–1269, 2014.

[86] Ming Yuan, V Roshan Joseph, and Yi Lin. An efficient variable selection approach for analyzing designed experiments. *Technometrics*, 49(4):430–439, 2007.

[87] Ming Yuan, V Roshan Joseph, and Hui Zou. Structured variable selection and estimation. *The Annals of Applied Statistics*, pages 1738–1757, 2009.

[88] Heping Zhang, Minghui Wang, and Xiang Chen. Willows: a memory efficient tree and forest construction package. *BMC bioinformatics*, 10(1):130, 2009.

[89] Weidong Zhang, Ron Korstanje, Jill Thaisz, Frank Staedtler, Nicole Harttman, Lingfei Xu, Minjie Feng, Liane Yanas, Hyuna Yang, William Valdar, Gary A. Churchill, and Keith DiPetrillo. Genome-wide association mapping of quantitative traits in outbred mice. *G3: Genes, Genomes, Genetics*, 2(2):167–174, 2012.

[90] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *The Annals of Statistics*, pages 3468–3497, 2009.

[91] Sihai Dave Zhao and Yi Li. Principled sure independence screening for cox models with ultra-high-dimensional covariates. *Journal of Multivariate Analysis*, 105(1):397 – 411, 2012.

[92] W Zhong and L Zhu. An iterative approach to distance correlation-based sure independent screening. *Journal of Statistical Computation and Simulation*, pages 1–15, 2014.

[93] Li-Ping Zhu, Lexin Li, Runze Li, and Li-Xing Zhu. Model-free feature screening for ultrahigh dimensional data. *Journal of the American Statistical Association*, 106(496):1464–1475, 2011.

[94] Andreas Ziegler, Anita L DeStefano, Inke R König, and Group 6. Data mining, neural nets, trees-problems 2 and 3 of genetic analysis workshop 15. *Genetic epidemiology*, 31(S1):S51–S60, 2007.

[95] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

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

APPENDICES

APPENDIX A

R Code

A.1 R Code for Chapter 2

Note that PC-SIS is often referred to by some variation of "Huang," which is the first author's surname. Also note that TC-SIS is often referred to in the code as CA-SIS or CAT-SIS (a reference to applications to *categorical* data).

```
#########################
# Functions
#########################

# DC-SIS
dc = function(u, v, n){
  u = as.numeric(u)
  v = as.numeric(v)
  m.u = matrix(u, n, n, byrow = TRUE)
  m.v = matrix(v, n, n, byrow = TRUE)
  s.u = abs(m.u - t(m.u))
  s.v = abs(m.v - t(m.v))
  s1 = sum((s.u) * (s.v)) / (n * n)
  s2 = sum(s.u) * sum(s.v) / (n * n * n * n)
  s3 = sum(rowSums(s.u) * (s.v)) / (n * n * n)
  sqrt(abs(s1 + s2 - 2 * s3))
}

DCSIS = function(data) {
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]

  omega = rep(0, p)
  dcov.y = dc(Y, Y, n)
  for(i in 1:p){
    dcov.xy = dc(X[, i], Y, n)
    dcov.xx = dc(X[, i], X[, i], n)
    dcorr = dcov.xy / sqrt(dcov.xx * dcov.y)
    omega[i] = dcorr * dcorr
  }
  omega
}


# MMLE
MMLE = function(data) {
  Y = data[, 1]
  X = data[, -1]
  p = dim(X)[2]

  Beta_jHatVect = rep(0, p)
  for(i in 1:p) {
```

```
    Beta_jHatVect[i] = glm(Y ~ X[, i], family = "binomial")$
        coefficients[[2]]
  }
  abs(Beta_jHatVect)
}

# Huang et al
Huang = function(data) {
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]

  as.numeric(apply(X, MARGIN = 2, function(X_jColumn, y)
    {return(1 / n * as.numeric(chisq.test(x = X_jColumn, y = Y)$
        statistic))},
     y = Y))
}

# CA-Trend
cum2 = function(a, b = a, n) {
  sum((a - mean(a)) * (b - mean(b))) / n
}

cum2x = function(a, n) {
  if (max(table(a)) > 0.8 * n) {
    return(0)
  } else {
    return(1)
  }
}

CAISIS = function(data, p1) {
  y = data[, 1]
  x = data[, -1]
  p = dim(x)[2]
  n = dim(x)[1]
  omega = rep(0, p)
  final.omega = rep(0, p)
  for (k in 1:p) {
    omega[k] = CA(y, x[, k])
  }
  rank = p + 1 - rank(omega, ties.method = "random")
  final.omega[which(rank %in% 1:p1)] = omega[which(rank %in% 1:p1)]
  index = which(rank %in% (p1 + 1):450)
  tempx = x[, index]
  tempx = residuals(lm(tempx ~ as.matrix(x[, which(rank %in% 1:p1)])
     ))
  for (k in 1:length(index)) {
    final.omega[index[k]] = CA(y, tempx[, k])
  }
  result = cbind(final.omega, rank)

  return(result)
}

# Choose p1
chooseP1 = function(data) {
  y = data[, 1]
  x = data[, -1]
  p = dim(x)[2]
```

```
  n = dim(x)[1]
  omega = rep(0, p)
  for(k in 1:p) {
    omega[k] = CA(y, x[, k])
  }
  rank = p + 1 - rank(omega, ties.method = "random")
  accuracy = rep(0, 200)
  for(index in 1:200) {
    train = sample(1:n, 0.75 * n, replace = TRUE)
    mydata = data.frame(cbind(y, x[, which(rank <= index)]))
    logit = glm(y ~ ., data = mydata[train, ], family = "binomial")
    predict.y = predict(logit, mydata[-train, ], type = "response")
    predict.y[predict.y >= 0.5] = 1
    predict.y[predict.y < 0.5] = 0
    accuracy[index] = length(which(predict.y == y[-train]))
  }

  return(min(which(accuracy == max(accuracy))))
}


CATrend = function(data) {
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]

  K2vec = rep(0, p)
  for (i in 1:p) {
    K2vec[i] =
        cum2(Y, X[, i], n) / sqrt(cum2(Y, n = n) * cum2(X[, i], n =
           n))
  }
  abs(K2vec)
}


#########################
# Simulation 1
#########################

source("functions.R")
n = 200
p = 5000

DCmat = matrix(NA, 500, 10)
MMmat = matrix(NA, 500, 10)
HUmat = matrix(NA, 500, 10)
CAmat = matrix(NA, 500, 10)

for (set in 1:500) {
  pr = runif(1, 0.05, 0.95)
  Y = sample(0:1, n, replace = TRUE, prob = c(pr, 1 - pr))
  prob = matrix(c(0.3, 0.8, 0.6, 0.7, 0.2, 0.6, 0.4, 0.6, 0.4, 0.7,
                  0.6, 0.4, 0.2, 0.4, 0.7, 0.1, 0.7, 0.2, 0.7, 0.4),
                   nrow = 2, byrow = TRUE)
  X = matrix(NA, n, p)
  for (i in 1:10) {
    X[Y == 0, i] = rbinom(length(which(Y == 0)), 2, prob[1, i])
    X[Y == 1, i] = rbinom(length(which(Y == 1)), 2, prob[2, i])
  }
  for (i in 11:p) {
    pr = runif(1, 0.05, 0.95)
    X[, i] = rbinom(n, 2, pr)
```

```
  }
  data = cbind(Y, X)

  # compare results
  DCres = DCSIS(data)
  DCmat[set, ] = (p - rank(DCres, ties.method = "first") + 1)[1:10]
  MMres = MMLE(data)
  MMmat[set, ] = (p - rank(MMres, ties.method = "first") + 1)[1:10]
  HUres = Huang(data)
  HUmat[set, ] = (p - rank(HUres, ties.method = "first") + 1)[1:10]
  CAres = CATrend(data)
  CAmat[set, ] = (p - rank(CAres, ties.method = "first") + 1)[1:10]

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""),
            row.names = FALSE)
  write.csv(DCmat, "DCmat", row.names = FALSE)
  write.csv(MMmat, "MMmat", row.names = FALSE)
  write.csv(HUmat, "HUmat", row.names = FALSE)
  write.csv(CAmat, "CAmat", row.names = FALSE)
}

########################
# Simulation 2
########################

source("functions.R")
n = 200
p = 5000

DCmat = matrix(NA, 500, 10)
MMmat = matrix(NA, 500, 10)
HUmat = matrix(NA, 500, 10)
CAmat = matrix(NA, 500, 10)

for (set in 1:500) {
  pr = runif(1, 0.05, 0.95)
  Y = sample(0:1, n, replace = TRUE, prob = c(pr, 1 - pr))
  Kai = matrix(c(0.35, 0.4, 0.4, 0.3, -0.213, 0.4, 0, 0.35, 0.45,
     0.213,
                 0.55, 0.5, 0.55, 0.6, 1.213, 0.5, 1, 0.65, 0.55,
                    0.787),
               nrow = 2, byrow = TRUE)
  X = matrix(1, n, p)
  for (i in 1:10) {
    tempi = rep(0, n)
    tempi[Y == 0] = rnorm(length(which(Y == 0)), 0, 1)
    tempi[Y == 1] = rnorm(length(which(Y == 1)), 1, 1)
    X[tempi < Kai[1, i], i] = 0
    X[tempi > Kai[2, i], i] = 2
  }
  for (i in 11:p) {
    pr = runif(1, 0.05, 0.95)
    X[, i] = rbinom(n, 2, pr)
  }
  data = cbind(Y, X)

  # compare results
  DCres = DCSIS(data)
  DCmat[set, ] = (p - rank(DCres) + 1)[1:10]
  MMres = MMLE(data)
  MMmat[set, ] = (p - rank(MMres) + 1)[1:10]
  HUres = Huang(data)
  HUmat[set, ] = (p - rank(HUres) + 1)[1:10]
```

```
  CAres = CATrend(data)
  CAmat[set, ] = (p - rank(CAres) + 1)[1:10]

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""),
            row.names = FALSE)
  write.csv(DCmat, "DCmat", row.names = FALSE)
  write.csv(MMmat, "MMmat", row.names = FALSE)
  write.csv(HUmat, "HUmat", row.names = FALSE)
  write.csv(CAmat, "CAmat", row.names = FALSE)


########################
# Simulation 3
########################

source("functions.R")
n = 200
p = 5000

DCmat = matrix(NA, 500, 5)
MMmat = matrix(NA, 500, 5)
HUmat = matrix(NA, 500, 5)
CAmat = matrix(NA, 500, 5)

for (set in 1:500) {
  Y = sample(0:1, n, 1 / 2)
  X = matrix(1, n, p)
  for (i in 1:p) {
    X[, i] = sample(0:2, n, replace = TRUE, prob = rep(1 / 3, 3))
  }
  coef = matrix(c(0, -5, 2, -6, 1, #-6, 0, 3, -1, 4,
                  3, -3, 4, -4, 3, #-4, 1, 2, -2, 3,
                  5, -1, 6, -2, 5), #-2, 2, 1, -3, 2),
                  nrow = 3, byrow = TRUE)
  Y = (X[, 1:5] == 0) %*% coef[1, ] +
    (X[, 1:5] == 1) %*% coef[2, ] +
      (X[, 1:5] == 2) %*% coef[3, ]
  prob = 1 / (1 + exp(-Y))
  Y = rep(0, n)
  Y[prob > 0.5] = 1
  data = cbind(Y, X)

  # compare results
  DCres = DCSIS(data)
  DCmat[set, ] = (p - rank(DCres) + 1)[1:5]
  MMres = MMLE(data)
  MMmat[set, ] = (p - rank(MMres) + 1)[1:5]
  HUres = Huang(data)
  HUmat[set, ] = (p - rank(HUres) + 1)[1:5]
  CAres = CATrend(data)
  CAmat[set, ] = (p - rank(CAres) + 1)[1:5]

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""),
            row.names = FALSE)
  write.csv(DCmat, "DCmat", row.names = FALSE)
  write.csv(MMmat, "MMmat", row.names = FALSE)
  write.csv(HUmat, "HUmat", row.names = FALSE)
  write.csv(CAmat, "CAmat", row.names = FALSE)
}


########################
# Simulation 4
########################
```

```
library(MASS)
source("functions.R")
n = 200
p = 1000

DCmat = matrix(NA, 500, 10)
#MMmat = matrix(NA, 500, 5)
#HUmat = matrix(NA, 500, 5)
CAmat = matrix(NA, 500, 10)

rho = 0.2
Sigma = diag(p)
Sigma = rho ^ abs(row(Sigma) - col(Sigma))

for (set in 1:500) {
  #X = matrix(1, n, p)
  X = mvrnorm(n, rep(0, p), Sigma)
  #X[which(x < -0.67, arr.ind = TRUE)] = 0
  #X[which(x > 0.67, arr.ind = TRUE)] = 2
  coef = c(5, -5, 5.5, -6, 6, 4, 4.5, -5.5, 5, -4)
  Y = X[, 1:10] %*% coef
  #Y = rep(0, n)
  #Y[y > median(y)] = 1
  data = cbind(Y, X)

  # compare results
  DCres = DCSIS(data)
  DCmat[set, ] = (p - rank(DCres) + 1)[1:10]
  #MMres = MMLE(data)
  #MMmat[set, ] = (p - rank(MMres) + 1)[1:5]
  #HUres = Huang(data)
  #HUmat[set, ] = (p - rank(HUres) + 1)[1:5]
  CAres = CATrend(data)
  CAmat[set, ] = (p - rank(CAres) + 1)[1:10]

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""),
            row.names = FALSE)
  write.csv(DCmat, "DCmat", row.names = FALSE)
  #write.csv(MMmat, "MMmat", row.names = FALSE)
  #write.csv(HUmat, "HUmat", row.names = FALSE)
  write.csv(CAmat, "CAmat", row.names = FALSE)
}


#########################
# Analysis:
#       Used for each simulation
#########################

DCmat = read.csv("DCmat") ## DC-SIS
MMmat = read.csv("MMmat")
HUmat = read.csv("HUmat")
CAmat = read.csv("CAmat")

# Mean Minimum Model Size
mean(apply(DCmat, 1, max))
mean(apply(MMmat, 1, max))
mean(apply(HUmat, 1, max))
mean(apply(CAmat, 1, max))

# Power Analysis
d = 10
apply(DCmat, 2, function(x) length(which(x <= d)) / 500)
apply(MMmat, 2, function(x) length(which(x <= d)) / 500)
```

```
apply(HUmat, 2, function(x) length(which(x <= d)) / 500)
apply(CAmat, 2, function(x) length(which(x <= d)) / 500)

d = 15
apply(DCmat, 2, function(x) length(which(x <= d)) / 500)
apply(MMmat, 2, function(x) length(which(x <= d)) / 500)
apply(HUmat, 2, function(x) length(which(x <= d)) / 500)
apply(CAmat, 2, function(x) length(which(x <= d)) / 500)

d = 20
apply(DCmat, 2, function(x) length(which(x <= d)) / 500)
apply(MMmat, 2, function(x) length(which(x <= d)) / 500)
apply(HUmat, 2, function(x) length(which(x <= d)) / 500)
apply(CAmat, 2, function(x) length(which(x <= d)) / 500)


#########################
# Real Data Analysis
#########################

source("https://bioconductor.org/biocLite.R")
biocLite("snpStats")

source("functions.R")
library(snpStats)

# genotypes
bed1 = read.plink("36417/NICHD_PolycysticOvary_c1_PCOS.bed")
bed1 = bed1$genotypes
class(bed1) = "matrix"

bed2 = read.plink("36419/NICHD_PolycysticOvary_c2_NUGENE.bed")
bed2 = bed2$genotypes
class(bed2) = "matrix"

genotype = rbind(bed1, bed2)
X = apply(genotype, 2, as.numeric)
Y = c(rep(1, 1043), rep(0, 3056))
data = cbind(Y, X)

# Choose p1 (191)
set.seed(10000000)
P1 = chooseP1(data) #191

# Iterative CA
result = CAISIS(data, p1 = P1) ##P1 = 191
```

A.2 R Code for Chapter 3

Note that GPC is often referred to by some variation of "Huang," which is the first author's surname. Also, JCIS is sometimes referred to as R3 or Formula3 internally in the code, as this was the working name for JCIS.

```
#########################
# Functions (functions.R)
#########################
library(dplyr)
library(boot)
library(readr)
library(MASS)
```

```
### Huang's paper
prob3 = function(Y, X1, X2) {
  n = length(Y)
  sum = 0
  temp = data.frame(cbind(Y, X1, X2))
  names(temp) = c('Y', 'X1', 'X2')
  ctab = count(temp, c('Y', 'X1', 'X2'))
  #ctab1 = count(cbind(Y, X1), c('Y', 'X1'))
  #ctab2 = count(cbind(Y, X2), c('Y', 'X2'))
  for (k in 1:dim(ctab)[1]) {
    k1 = sum(ctab[(ctab[, 1] == ctab[k, 1] & ctab[, 2] == ctab[k,
        2]), 4])
    k2 = sum(ctab[(ctab[, 1] == ctab[k, 1] & ctab[, 3] == ctab[k,
        3]), 4])
    sum = sum + (k1 * k2 / n ^ 2 - ctab[k, 4] / n) ^ 2 / (k1 * k2 /
        n ^ 2)
  }
  sum
}

huang = function(data) {
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]
  Hmat = matrix(0, nrow = p, ncol = p)
  for (i in 1:(p - 1)) {
    for (j in (i + 1):p) {
      Hmat[i, j] = prob3(Y, X[, i], X[, j])
    }
  }
  Hmat
}

# Zhang's paper
BICn = function(data) {
  #data = data.frame(cbind(Y, X))
  BIC(lm(Y ~ ., data = data))
}

zhang = function(data) {
  data = data.frame(scale(data))
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]

  d = 5
  candidateX = X

  ##print(candidateX)

  modelmat = data.frame(Y)
  select = c()
  main = c()
  candidate = as.character(1:p)
  inter = 0
  while(inter < 5) {
    BICvec= rep(0, dim(candidateX)[2])
    for(i in 1:dim(candidateX)[2])
      {
        print(i)
```

```
      print(dim(candidateX)[2])
        tempmat = data.frame(cbind(modelmat, candidateX[, i]))
        BICvec[i] = BICn(tempmat)
      }
    ns = candidate[which(BICvec == min(BICvec))]
    select = c(select, ns)
    modelmat = data.frame(cbind(modelmat, candidateX[, which(BICvec
       == min(BICvec))]))
    candidate = candidate[-which(BICvec == min(BICvec))]
    candidateX = candidateX[, -which(BICvec == min(BICvec))]
    if(length(grep("_", ns)) == 0) {
      if(length(main) > 0) {
        candidate = c(candidate, paste(ns, main, sep = "_"))
        candidateX = cbind(candidateX, modelmat[, dim(modelmat)[2]]
           * mainX)
        mainX = data.frame(cbind(mainX, modelmat[, dim(modelmat)
           [2]]))
      } else {
        mainX = data.frame(modelmat[, dim(modelmat)[2]])
      }
      main = c(main, ns)
    } else {
      inter = inter + 1
    }
  }
  #print(select)
  select
}

cum2 = function(a, b = a) {
  n = length(a)
  sum((a - mean(a)) * (b - mean(b))) / n
}

R3 = function(data) {
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]
  Rmat = matrix(0, nrow = p, ncol = p)
  K2vec = rep(0, p)
  K2y = cum2(Y)
  for (i in 1:p) {
    K2vec[i] = cum2(X[, i])
  }
  for (i in 1:(p - 1)) {
    for (j in (i + 1):p) {
      Rmat[i, j] = cum3(Y, X[, i], X[, j], unbiased = FALSE) /
         sqrt(K2y * K2vec[i] * K2vec[j])
    }
  }
  Rmat
}


#########################
# Simulation 1
#########################

source("functions.R")
n = 200
```

```
p = 1000

set.seed(7919)
for (set in 1:100) {
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p) {
    X[, i] = sample(0:1, n, replace = TRUE)
  }
  Y = X[, 1] * X[, 2]
  data = cbind(Y, X)

  Rmat = R3(data)
  Hmat = huang(data)

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""), row.
     names = FALSE)
  write.csv(Rmat, paste("Rmat", set, ".csv", sep = ""), row.names =
     FALSE)
  write.csv(Hmat, paste("Hmat", set, ".csv", sep = ""), row.names =
     FALSE)
}

result = data.frame(matrix(NA, 100, 3))
names(result) = c("Set", "JCIS_X1X2", "HuangX1X2")

for(set in 1:100) {
  result[set, 1] = set
  Rmat = read.csv(paste("Rmat", set, ".csv", sep = ""))
  result[set, 2] = dim(which(Rmat >= Rmat[1, 2], arr.ind = TRUE))[1]

  Hmat = read.csv(paste("Hmat", set, ".csv", sep = ""))
  result[set, 4] = dim(which(Hmat >= Hmat[1, 2], arr.ind = TRUE))[1]

}

write.csv(result, "result.csv", row.names = FALSE)

outcomes <- read_csv("~/Interaction␣Simulations/Simulation␣I/result.
   csv")

k3 = sapply(outcomes[,2], mean)
huang = sapply(outcomes[,3], mean)
length(which(huang < k3))

########################
# Simulation 2
########################

source("functions.R")
n = 200
p = 1000

for (set in 1:50) {
  Theta = matrix(c(c(0.3, 0.4, 0.5, 0.3), c(0.95, 0.9, 0.9, 0.95)),
     nrow = 2, byrow = TRUE)
  Y = sample(c(0, 1), n, replace = TRUE, prob = c(0.75, 0.25))
  X = matrix(0, nrow = n, ncol = p)
  for (i in 1:4) {
    prob = (Theta[, i])[Y + 1]
    X[, 2 * i - 1] = unlist(lapply(prob, function(x) sample(1:0, 1,
       prob = c(x, 1 - x))))
    prob2 = apply(cbind(prob, X[, 2 * i - 1]), 1,
                  function(x) ifelse(x[2] == 0,
```

```
                                          0.6 * ifelse(x[1] > 0.5, 1, 0)
                                             + 0.4 * ifelse(x[1] <= 0.5,
                                             1, 0),
                                          0.95 * ifelse(x[1] > 0.5, 1, 0)
                                              + 0.05 * ifelse(x[1] <=
                                             0.5, 1, 0)))
    X[, 2 * i] = unlist(lapply(prob2, function(x) sample(1:0, 1,
       prob = c(x, 1 - x))))
  }
  for (i in 9:p) {
    X[, i] = sample(1:0, n, replace = TRUE)
  }
  data = cbind(Y, X)

  Rmat = R3(data)
  Hmat = huang(data)

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""), row.
     names = FALSE)
  write.csv(Rmat, paste("Rmat", set, ".csv", sep = ""), row.names =
     FALSE)
  write.csv(Hmat, paste("Hmat", set, ".csv", sep = ""), row.names =
     FALSE)
}

result = data.frame(matrix(NA, 100, 9))
names(result) = c("Set", "FX12", "FX34", "FX56", "FX78", "HX12", "
   HX34", "HX56", "HX78")

for(set in 1:100) {
  result[set, 1] = set
  Rmat = read.csv(paste("Rmat", set, ".csv", sep = ""))
  result[set, 2] = dim(which(Rmat >= Rmat[1, 2], arr.ind = TRUE))[1]
  result[set, 3] = dim(which(Rmat >= Rmat[3, 4], arr.ind = TRUE))[1]
  result[set, 4] = dim(which(Rmat >= Rmat[5, 6], arr.ind = TRUE))[1]
  result[set, 5] = dim(which(Rmat >= Rmat[7, 8], arr.ind = TRUE))[1]
  Hmat = read.csv(paste("Hmat", set, ".csv", sep = ""))
  result[set, 6] = dim(which(Hmat >= Hmat[1, 2], arr.ind = TRUE))[1]
  result[set, 7] = dim(which(Hmat >= Hmat[3, 4], arr.ind = TRUE))[1]
  result[set, 8] = dim(which(Hmat >= Hmat[5, 6], arr.ind = TRUE))[1]
  result[set, 9] = dim(which(Hmat >= Hmat[7, 8], arr.ind = TRUE))[1]
}

write.csv(result, "result.csv", row.names = FALSE)


########################
# Simulation 3
########################

source("functions.R")
n = 200
p = 1000

set.seed(7919)

Zmat = matrix(NA, 100, 5)
for (set in 1:100) {
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p) {
    X[, i] = rnorm(n, 0, 2)
    #X[, i] = rnorm(n, 0, 1)
  }
  Y = X[, 1] * X[, 2]  + X[, 3] * X[, 4]
```

```
  #Y = X[, 1] * X[, 2]  #
  data = cbind(Y, X)

  zres = zhang(data)

  Zmat[set, ] = zres[grep("_", zres)]
  Rmat = R3(data)

  write.csv(data, paste("Simu_data", set, ".csv", sep = ""), row.
     names = FALSE)
  write.csv(Rmat, paste("Rmat", set, ".csv", sep = ""), row.names =
     FALSE)
  write.csv(Zmat, "Zmat.csv", row.names = FALSE)
}

result = data.frame(matrix(NA, 100, 5))
names(result) = c("Set", "Formula3_X12", "Formula3_X34", "Zhang_X12"
    , "Zhang_X34")
d = 5 #####You can put this at 3 and still have 100% consistency I
    belive. And at d =2, you have about 98% consistency.
##(This is speaking of my K_3 method, not Zhang)
##Zmat = read.csv("Simulation I/Simulation Ib/ZmatSLURM.csv")
Zmat = read.csv("ZmatSLURM.csv")

for(set in 1:100) {
  result[set, 1] = set
  Rmat = read.csv(paste("Rmat", set, ".csv", sep = ""))
  result[set, 2] = ifelse(dim(which(Rmat >= Rmat[1, 2], arr.ind =
     TRUE))[1] <= d, 1, 0)
  result[set, 3] = ifelse(dim(which(Rmat >= Rmat[3, 4], arr.ind =
     TRUE))[1] <= d, 1, 0)
  setres = as.character(unlist(Zmat[set, ]))
  result[set, 4] = ifelse("1_2" %in% setres | "2_1" %in% setres, 1,
     0)
  result[set, 5] = ifelse("3_4" %in% setres | "4_3" %in% setres, 1,
     0)
}

write.csv(result, "result.csv", row.names = FALSE)

colSums(result)/100


########################
# Simulation 4
########################

source("functions.R")
n = 100
p = 500

Zmat = matrix(NA, 100, 5)
for (set in 1:100) {
  rho = 0.01
  Sigma = diag(p)
  Sigma = rho ^ abs(row(Sigma) - col(Sigma))
  X = mvrnorm(n, rep(0, p), Sigma)
  Y = 3 * X[, 1] * X[, 3] + 3 * X[, 6] * X[, 10] + 1 * X[, 1] + 1 *
     X[, 3] + 1 * X[, 6] + 1 * X[, 10]
  data = data.frame(cbind(Y, X))

  zres = zhang(data)
  Zmat[set, ] = zres[grep("_", zres)]
  Rmat = R3(data)
```

```
  write.csv(data, paste("Simu_data", set, ".csv", sep = ""), row.
     names = FALSE)
  write.csv(Rmat, paste("Rmat", set, ".csv", sep = ""), row.names =
     FALSE)
  write.csv(Zmat, "Zmat.csv", row.names = FALSE)
}

result = data.frame(matrix(NA, 100, 5))
names(result) = c("Set", "Formula3_X13", "Formula3_X610", "Zhang_X13
   ", "Zhang_X610")
d = 5
Zmat = read.csv("Zmat.csv")

for(set in 1:100) {
  result[set, 1] = set
  Rmat = read.csv(paste("Rmat", set, ".csv", sep = ""))
  result[set, 2] = ifelse(dim(which(Rmat >= Rmat[1, 3], arr.ind =
     TRUE))[1] <= d, 1, 0)
  result[set, 3] = ifelse(dim(which(Rmat >= Rmat[6, 10], arr.ind =
     TRUE))[1] <= d, 1, 0)
  setres = as.character(unlist(Zmat[set, ]))
  result[set, 4] = ifelse("1_3" %in% setres | "3_1" %in% setres, 1,
     0)
  result[set, 5] = ifelse("6_10" %in% setres | "10_6" %in% setres,
     1, 0)
}

write.csv(result, "result.csv", row.names = FALSE)

#########################
# Real Data Analysis (Chromosome One shown. Others are very similar)
#########################


cum2 = function(a, b = a) {
  n = length(a)
  sum((a - mean(a)) * (b - mean(b))) / n
}

R3 <- function(data, names = "QQQ",resultFileName = "zzz.csv")
{
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]

  ##print(p)

  Rmat = matrix(0, nrow = p, ncol = p)

  ##print(dimnames(Rmat))

  dimnames(Rmat) <- list(rownames(Rmat)<-names,
                         colnames(Rmat)<-names)
  K2vec = rep(0, p)
  K2y = cum2(Y)
        for (i in 1:p)
        {
            K2vec[i] = cum2(X[, i])
        }
        for (i in 1:(p - 1))
        {
            for (j in (i + 1):p)
                {
```

```
                        if ( K2vec[i] * K2vec[j] ==0)
                        { Rmat[i,j] = 0 } #####Make 0 for real
                            analysis.
                        else
                        {
                                Rmat[i, j] = cum3(Y, X[, i], X[, j],
                                     unbiased = FALSE) / sqrt(K2y *
                                    K2vec[i] * K2vec[j])
                        }

                }
                fwrite(as.data.frame(Rmat[i,]), file =
                    resultFileName, row.names = TRUE, col.names =
                    TRUE, append = TRUE)
                        ####write 1 line at a time.

        }
  Rmat
}


R3LongPrint <- function(data, names = "QQQ",resultFileName = "zzz.
    csv")
{
####The hope here is to reduce the output file size. There is no
    reason to store the data in the matrix format.
####Now that we know that the raw results are seemingly working, the
     matrix format is superfluous.
  Y = data[, 1]
  X = data[, -1]
  n = dim(X)[1]
  p = dim(X)[2]

  ##print(p)

  RLONGmat = matrix(0, nrow = choose(p,2), ncol = 3)

  ##print(dimnames(Rmat))


        colnames(RLONGmat) = c("SNP_1", "SNP_2", "R␣Squared")

  K2vec = rep(0, p)
  K2y = cum2(Y)
        for (i in 1:p)
        {
            K2vec[i] = cum2(X[, i])
        }
        for (i in 1:(p - 1))
        {
            for (j in (i + 1):p)
                {
                        if ( K2vec[i] * K2vec[j] > 0)
                        {
                                Rsquared= cum3(Y, X[, i], X[, j],
                                    unbiased = FALSE) / sqrt(K2y *
                                    K2vec[i] * K2vec[j])
                                RLONGmat[p*(i-1) + j - choose(i+1,2)
                                    ,] = c(i,j,Rsquared)
                        }

                }

        }
```

```
        fwrite(as.data.frame(round(RLONGmat, digits = 6)), file =
           resultFileName, row.names = FALSE, col.names = TRUE,
           append = FALSE)

        print(resultFileName)
}


source("https://bioconductor.org/biocLite.R")

install.packages(c("binomTools","feather", "readr"),lib=c("/uufs/
   chpc.utah.edu/common/home/uXXXXXXXX/software/pkg/RLibs/3.3.2i"),
            repos=c("http://cran.us.r-project.org"),verbose=TRUE)

source("Interact␣Functions.R")
library(snpStats)
library(R.utils)
library(base)
library(boot)
library(binomTools)
library(data.table)
library(dplyr)
library(magrittr)
library(feather) # v0.0.0.9000
library(readr) # v0.2.2

###Although this BIM table is only assciated with the Y = 1 group,
   the BIM table for the control (Y = 0) group is identical.
###Because of this, I just use the BIM table from the Y = 1 group as
    the main BIM table.
BIMTable = fread("NICHD_PolycysticOvary_c1_PCOS.bim",
                 col.names = c("Chromosome", "ID","GeneticDist", "
                    Position", "Allele1", "Allele2"))

BIMTable = BIMTable[,-3] ###Remove the GeneticDistance column. Has
   no information.

#dim(filter(BIMTable, Chromosome %in% 1:23, Position>0))###The "
   cleansed" table we want. 729286 (maybe messed up by find replace)
    by 5

cleanBIMTable = filter(BIMTable, Chromosome %in% 1:23, Position>0)

filter(cleanBIMTable, Chromosome == 1) %>%
  select(ID) ->
  name1

# genotypes
########Clean bed 1
bed1 = read.plink("NICHD_PolycysticOvary_c1_PCOS.bed", select.snps =
    name1[,1])
bed1 = bed1$genotypes

bed0 = read.plink("NICHD_PolycysticOvary_c2_NUGENE.bed", select.snps
    = name1[,1])
bed0 = bed0$genotypes

genotype = rbind(bed1, bed0)


snpsum.col <- col.summary(genotype)
call <- 0.95 ###Why remove low call rate? I have a citation on this
   prescreen.
use <- with(snpsum.col, (!is.na(Call.rate) & Call.rate >= call))
```

```
use[is.na(use)] <- FALSE
cat(ncol(genotype)-sum(use),"SNPs␣will␣be␣removed␣due␣to␣low␣call␣
    rate.\n")
genotype <- genotype[,use]
snpsum.col <- snpsum.col[use,]

minor <- 0.1
use1 <- with(snpsum.col, (!is.na(MAF) & MAF > minor) )
use1[is.na(use1)] <- FALSE
cat(ncol(genotype)-sum(use1),"SNPs␣will␣be␣removed␣due␣to␣low␣MAF␣.\
    n"  )
genotype <- genotype[,use1]
snpsum.col <- snpsum.col[use1,]

name1 = rownames(snpsum.col)###Update the chromsomes we are looking
    at.


class(genotype) = "matrix"

X = apply(genotype, 2, as.numeric)

Y = c(rep(1, 1043), rep(0, 3056))

data = cbind(Y, X)

timestamp()
R3LongPrint(data, names = name1, resultFileName = "chrom1LEAN.csv")
timestamp()
```

A.3 R Code for Chapter 4

```
#########################
# Functions
#########################
library(boot)

### Biased SD
cum2 = function(a, b = a) {
  n = length(a)
  sum((a - mean(a)) * (b - mean(b))) / n
}

indicator <- function(lower, upper) ###As in I(lower < upper)
{
  comps = ifelse(lower < upper, 1,0)

  return(prod(comps))
}

isAllZero <- function(testThisVect)
{
  nonZero = (testThisVect != 0)
  if(sum(nonZero) > 0)
  {
    return(FALSE)
  }
  else
  {return(TRUE)}
}

###There is a built in norm function, but it is slow. Used SVD. Ours
     is UBE.
```

```
norm_vec <- function(x){sqrt(sum(x^2))}

################################
################################
## ZLLZ (aka SIRS)
################################
################################

zllz <-function(data, q)
{

  Y = data[, 1:q]
  X = data[, -(1:q)]
  X = scale(X) ###They assume that the data is standardized.

  n = dim(X)[1]
  p = dim(X)[2]

  omegaHats = apply(X, MARGIN = 2, omegaCalc_jointInd, sampSize = n,
      dataY = Y, q = q)

  if(isAllZero(omegaHats))
  {
    randomRanks = sample(1:p)
    print("All omegas were zero")
    #print(paste("Returning ", 3001-randomRanks[1], 3001-randomRanks
        [2], 3001-randomRanks[3]))
    ###This just makes it some that randomRanks actually has the
        ranks.

    return(randomRanks)#####The simulation will take care of sorting
        these.
  }


  return(omegaHats)

}

omegaCalc_jointInd <-function(X_j, dataY, sampSize, q) ##dataY = Y,
    sampSize = n. Wrappers.
{
  sumMat = matrix(0, nrow = sampSize, ncol = sampSize)


  colSumsSqrd = rep(0, sampSize)


  for(k in 1:sampSize)
  {
    for(i in 1:sampSize)
    {
      if(q>1)
      {sumMat[i,k] = X_j[i]*indicator(dataY[i,], dataY[k,])}
      else
      {sumMat[i,k] = X_j[i]*indicator(dataY[i], dataY[k])}

    }

    colSumsSqrd[k] = sum(sumMat[,k])^2
  }

  omegaHat_j = sum(colSumsSqrd)/(sampSize^3)

  return(omegaHat_j)
}
```

```
#####################################
#####################################
### DC-SIS
#####################################
#####################################

DCSIS_MV<-function(data, q)
{

    Y = data[, 1:q]
    X = data[, -(1:q)]


    n = dim(X)[1]
    p = dim(X)[2]
    dc = rep(0,p)

    dcov.y = DistanceCov(Y,Y, n = n)

    for(j in 1:p)
    {
        X.j = X[, j]
        dcov.xy = DistanceCov(X.j, Y, n = n)
        dcov.xx = DistanceCov(X.j, X.j, n = n)
        dcorr = dcov.xy / sqrt(dcov.xx * dcov.y)
        dc[j] = dcorr * dcorr
    }

    return(dc)

}

DistanceCov <- function(u, v, n)
{
  u = as.matrix(as.numeric(u))###cast vectors into 1 by n column
      matrix.
  v = as.matrix(as.numeric(v))###cast vectors into 1 by n column
      matrix.

  u_q = ncol(u)
  v_q = ncol(v)

  diff_uArr = array(0,dim = c(n,n, u_q))
  diff_vArr = array(0,dim = c(n,n, v_q))

  for(m in 1:u_q)
  {
    ###Make an (n by n)  matrix whose rows consist of repeated
       instances of Y^(m).
     uArr= matrix(u[,m], nrow = n, ncol = n, byrow = TRUE)

     ###@Now take the difference array[,,m] - t(array[,,])
     diff_uArr[,,m] = uArr - t(uArr)

  }

  for(m in 1:v_q)
  {
    ###Make an (n by n)  matrix whose rows consist of repeated
       instances of Y^(m).
    vArr= matrix(v[,m], nrow = n, ncol = n, byrow = TRUE)

    ###Now take the difference array[,,m] - t(array[,,])
    diff_vArr[,,m] = vArr - t(vArr)
```

```
  }

  #m.u = matrix(u, n, n, byrow = TRUE)
  #m.v = matrix(v, n, n, byrow = TRUE)
  s.u = apply(diff_uArr, MARGIN = c(1,2), norm_vec)
  s.v = apply(diff_vArr, MARGIN = c(1,2), norm_vec)


  s1 = sum((s.u) * (s.v)) / (n^2)
  s2 = sum(s.u) * sum(s.v) / (n^4)
  s3 = sum(rowSums(s.u) * (s.v)) / (n^3)

  dcov <- sqrt(abs(s1 + s2 -2 * s3))
  return(dcov)
}

################################################
################################################
### CovarProd (small phi). The cheap phi method.
################################################
################################################

CovarProd_MV_main <- function(data, q)
{ #####This is the function for q many components
  Y = data[, 1:q]
  X = data[, -(1:q)]
  n = dim(X)[1]
  p = dim(X)[2]
  phiVect = rep(0,p)
  covProdVect = rep(0,p)


  K2vec = rep(0, p)
  #K2yTEST = rep(0, q)
  ##Create a vector of the estimated SDs for each component of Y.

  if(q>1)
  {
    K2y = apply(Y, MARGIN =2, cum2)
  }
  else
  {
    K2y = cum2(Y)
  }


  ###Get the SD for each covariate.
  for (j in 1:p) {
    K2vec[j] = cum2(X[, j])
  }


  for (j in 1:p)
  {


    for(m in 1:q)
    {
      covProdVect[j] = prod(apply(Y, MARGIN = 2, cum2, a = X[,j]))
    }
    #print(length(prod_Ycentered))
    #print(length(X[,j] - mean(X[,j])))
    phiVect[j] =  covProdVect[j]/sqrt(K2vec[j]^q*prod(K2y))

  }
```

```
  return(abs(phiVect))
}

##############################################
##############################################
### Correlation Matrix. Generalized correlation. "Big" phi.
##############################################
##############################################

CorrMat_MV_main = function(data, q, normToUse = 1)
{ #####This is the function for q many components
  Y = data[, 1:q]
  X = data[, -(1:q)]
  n = dim(X)[1]
  p = dim(X)[2]
  phiVect = rep(0,p)


  for (j in 1:p)
  {

    covMat = abs(cov(cbind(X[,j], Y)))

    if(normToUse == 1)
    {
      sdVec = (1/sqrt(diag(covMat)))

      phiVect[j] = sdVec %*% covMat %*% sdVec
    }
    else if(normToUse == 2)
    {
      ##print("Using the Frobenius norm")

      sdMat = diag(1/sqrt(diag(covMat)))

      phiVect[j] = norm_vec(sdMat %*% covMat %*% sdMat)

    }


    #print(sdMat %*% covMat %*% sdMat)

  }

  return(abs(phiVect))
}

#####################################################
#####################################################
### Interact Interact. Generalized correlation WITH interactions. "
    Super" phi. INTERACT
#####################################################
#####################################################

CorrMat_MV_Interact = function(data, q, normToUse = 1)
{ #####This is the function for q many components
  Y = data[, 1:q]
  X = data[, -(1:q)]
  n = dim(X)[1]
  p = dim(X)[2]
  PhiMat = matrix(0, nrow = p, ncol = p) ###This can be trimmed to (
     p-1) by (p-1) if need be.
  ##We could even start recording only the combinations that matter.
       The long form like in Real Data, Paper 2.
```

```
  for (j_1 in 1:(p - 1))
  {
    for (j_2 in (j_1 + 1):p)
    {
      ###This is the covariance matrix with the 3 way joint cumulant
          entries padding the first row/col
      fullMat = matrix(0, nrow = q+1, ncol = q+1)

      ##This is approach number 1 for the first row/col.
      ##firstRowCol = c(cov(X[,j_1],X[,j_2]), apply(Y, MARGIN = 2,
         cum3, b = X[,j_1], c = X[,j_2], unbiased = FALSE))

      ## This is approach number 2 for the first row/col.
      firstRowCol = c(var(X[,j_1])*var(X[,j_2]), apply(Y, MARGIN =
         2,cum3, b = X[,j_1], c = X[,j_2], unbiased = FALSE))

      ###covYMat = cov(Y)

      fullMat[2:(q+1), 2:(q+1)] = cov(Y)
      fullMat[1,] = firstRowCol
      fullMat[,1] = firstRowCol ###Overwrite on [1,1], but that
         matters very little here. UBE.

      fullMat = abs(fullMat) ##We only care about the magnitude of
         the relationship, not the directions of the relationships.


      ###if(!(j %% 1000)){print(covMat)}

      if(normToUse == 1)
      {
          sdMat = 1/sqrt(diag(fullMat))
          PhiMat[j_1, j_2] = (sdMat %*% fullMat %*% sdMat)
      }

      else if(normToUse == 2)
      {
        ##print("Using the Frobenius norm")

        sdMat = diag(1/sqrt(diag(fullMat)))

        PhiMat[j_1, j_2] = norm_vec(sdMat %*% fullMat %*% sdMat)

      }

    }
  }

  return(abs(PhiMat))
}

### ********************************************************
CorrMat_MV_Interact_WithinGroup <- function(data, q, normToUse = 2,
   names)
{ #####This is the function for q many components
  ## WithinGroup is used to indicate that the data is
  ##           (Y,X) and we only need to look at one way pairs (i.e.
     groups 1,2,3,4)
  ##           on the diagonal.
  Y = data[, 1:q]
  X = data[, -(1:q)]
  n = dim(X)[1]
  p = dim(X)[2]

  rowNum = 1
```

```
eighty_fifth10k = 0 ##Initialize as 0.

PhiLONGmat = matrix(0, nrow = choose(p,2), ncol = 3)

##print(dimnames(Rmat))


colnames(PhiLONGmat) = c("SNP_1", "SNP_2", "Phi")

K2y = rep(0, q)
K2vec = rep(0, p)

##We could even start recording only the combinations that matter.
    The long form like in Real Data, Paper 2.
for(m in 1:q)
{
  K2y[m] = cum2(Y[,m])
}

for (j in 1:p)
{
  K2vec[j] = cum2(X[, j])
}


for (j_1 in 1:(p - 1))
{
  for (j_2 in (j_1 + 1):p)
  {

    if(rowNum == 10001)#Take the 85th quantile and above.
    {
      eighty_fifth10k = as.numeric(quantile(as.numeric(PhiLONGmat
          [1:10000,3]), probs = 0.85))
    }

    ## This is approach number 2 for the first row/col.
    firstRowCol = apply(Y, MARGIN = 2,cum3, b = X[,j_1], c = X[,j_
       2], unbiased = FALSE)

    normalizedFirstRowCol = firstRowCol/sqrt(K2y*K2vec[j_1]*K2vec[
       j_2])

    if(normToUse == 1)
    {
      Phi_j1_j2 = round(sum(abs(normalizedFirstRowCol)), 7)


      if(rowNum < 10001 | (rowNum > 10000 & Phi_j1_j2 > eighty_
         fifth10k))
      {
        #PhiLONGmat[rowNum,] = c(names[j_1], names[j_2], Phi_j1_j2
           )

        PhiLONGmat[rowNum,1:2] = c(names[j_1], names[j_2])
        PhiLONGmat[rowNum,3] = Phi_j1_j2

        rowNum = rowNum + 1
      }
    }

    else if(normToUse == 2)
    {
      Phi_j1_j2 = round(norm_vec(normalizedFirstRowCol), 7)
      ###We do not need to report 15 digits or whatever.
```

```
        ###This is a simplified version. Only looks at the values
            that will differ.

        ##record the first 10k. Then only record the value if it is
            above the 75th percentile.
        if(rowNum < 10001 | Phi_j1_j2 > eighty_fifth10k)
        {
          #PhiLONGmat[rowNum,] = c(names[j_1], names[j_2], Phi_j1_j2
              )
          PhiLONGmat[rowNum,1:2] = c(names[j_1], names[j_2])
          PhiLONGmat[rowNum,3] = Phi_j1_j2
          rowNum = rowNum + 1
        }

      }

    }
  }


  return(PhiLONGmat[1:(rowNum),]) ###Check for tailing zeros.
  #Remember that rowNum has already been incremented to one past the
      last recordable value.
}

CorrMat_MV_Interact_SeparateGroup= function(data, q, normToUse = 2,
    namesX1, namesX2)
{ #####This is the function for q many components
  ## separateGroup is used to indicate that the data is
  ##           (Y,X1, X2) and we need to look at ALL pairs


  X_1Size = length(namesX1)
  #n = dim(X_1)[1]
  Y = data[, 1:q]
  X_1 = data[,(q+1):(X_1Size + q)]
  X_2 = data[, -(1:(X_1Size + q))]

  # Y = Y
  # X_1 = as.matrix(X1)
  # X_2 = as.matrix(X2)
  #
  # namesX1 =  colnames(X1)
  # namesX2 = colnames(X2)

  p_1 = dim(X_1)[2]
  p_2 = dim(X_2)[2]

  # if(p_1 != p_2)
  # {
  #   print("p_1 should have equaled p_2 in our case! 11107 is the
      group size.")
  # }

  rowNum = 1
  eighty_fifth10k = 0 ##Initialize as 0.

  PhiLONGmat = matrix(0, nrow = max(p_1, p_2)^2, ncol = 3)


  colnames(PhiLONGmat) = c("SNP_1", "SNP_2", "Phi")

  K2y = rep(0, q)
  K2X1vec = rep(0, p_1)
```

```
K2X2vec = rep(0, p_2)

##We could even start recording only the combinations that matter.
#The long form like in Real Data, Paper 2.
for(m in 1:q)
{
  K2y[m] = cum2(Y[,m])
}

for (j in 1:p_1)
{
  #print("Made it to the first loop")
  K2X1vec[j] = cum2(X_1[,j])
}
for (j in 1:p_2)
{
  #print("Made it to the second loop")
  K2X2vec[j] = cum2(X_2[,j])
}


for (j_1 in 1:p_1)
{
  for (j_2 in 1:p_2)
  {

    if(rowNum == 10001)#Take the 85th quantile and above.
    {
      print("Reached 10000th row.")
      #eighty_fifth10k = as.numeric(quantile(PhiLONGmat
          [1:10000,3], probs = 0.85))
      eighty_fifth10k = as.numeric(quantile(as.numeric(PhiLONGmat
          [1:10000,3]),
                                            probs = 0.85, na.rm =
                                               TRUE))
    }

    ## This is approach number 2 for the first row/col.
    firstRowCol = apply(Y, MARGIN = 2,cum3, b = X_1[,j_1], c = X_
       2[,j_2], unbiased = FALSE)

    normalizedFirstRowCol = firstRowCol/sqrt(K2y*K2X1vec[j_1]*
       K2X2vec[j_2])

    if(normToUse == 1)
    {
      Phi_j1_j_2 = round(sum(abs(normalizedFirstRowCol)), 7)


      if(rowNum < 10001 | (rowNum > 10000 & Phi_j1_j2 > eighty_
          fifth10k))
      {
        PhiLONGmat[rowNum,1:2] = c(namesX1[j_1], namesX2[j_2])
         PhiLONGmat[rowNum,3] = Phi_j1_j2

        rowNum = rowNum + 1
      }
    }

    else if(normToUse == 2)
    {
      Phi_j1_j2 = round(norm_vec(normalizedFirstRowCol), 7)

      ###This is a simplified version. Only looks at the values
          that will differ.
```

```
        ##record the first 10k. Then only record the value if it is
            above the 85th percentile.
        if(rowNum < 10001 | Phi_j1_j2 > eighty_fifth10k)
        {
          PhiLONGmat[rowNum,1:2] = c(namesX1[j_1], namesX2[j_2])
          PhiLONGmat[rowNum,3] = Phi_j1_j2
          rowNum = rowNum + 1
        }

        if((rowNum) %% 100000 == 0)
        {
          timestamp()
          print(paste("Benchmark", rowNum))
        }

      }
    }
  }


  return(PhiLONGmat[1:(rowNum),]) ###Check for tailing zeros.
}#Remember that rowNum has already been incremented to one past the
    last recordable value.


#########################
# Simulation 1
#########################

source("/uufs/chpc.utah.edu/common/home/uxxxxxx/GenCorr/
    GenCorrFunct.R")

library(data.table)
library(R.utils)
library(MASS)

arg = cmdArgs()

 n = 60
 p = 3000
 q = 6

 n = 120
 p = 1500
 q = 4

score = matrix(-1, nrow = 100, ncol = 3) ###These are the scores for
     each method.
#### Best rank, mid rank, worst rank. Perfect score is 1,2,3.
colnames(score) = c("GCorr_Best", "GCorr_Mid", "GCorr_Worst")

set.seed(arg[[2]])

for(r in 1:100)
{
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p)
  {
    #X[, i] = rnorm(n, 0, 5)
    X[, i] = rnorm(n, 3, 1)
    #X[, i] = rpois(n, 2)
  }

  Y = matrix(0, nrow = n, ncol = q)
```

```
  # rho = 0.5
  # Sigma = diag(q)
  # Sigma = rho ^ abs(row(Sigma) - col(Sigma))
  #
  # scalarMat = t(mvrnorm(3, rep(0, q), Sigma))

  # rho = 0.5
  # Sigma = diag(q)
  # Sigma = rho ^ abs(row(Sigma) - col(Sigma))
  #
  # scalarMat = t(mvrnorm(3, rep(3, q), Sigma))


 sign = (-1)^(rbinom(n=3*q,size = 1, prob = 0.4))
 scalarVals = sign*(4*log(n)/sqrt(n) + abs(rnorm(n = 3*q,0, 1)))
 scalarMat = matrix(scalarVals, nrow = 3, ncol = q)

####This comes  from DC-SIS. (I've adapted it to a MV setting)

  for(m in 1:q)
  {
    Y[,m] = X[,c(1,2,3)] %*% scalarMat[,m]
  }#

  testData = cbind(Y, X)

  #testData = matrix(rnorm(300, 0,1), ncol = 30, nrow = 10)
  #phi = CovarProd_MV_main(data = testData, q = q)
  phi_num2 = CorrMat_MV_main(data = testData, q = q, normToUse = arg
     [[3]])

  omega = rep(2,3000)
   #timestamp()
   omega = zllz(data = testData, q=q)
   #timestamp()

  dc = rep(1,3000)
  #timestamp()
 dc = DCSIS_MV(data = testData, q=q)
  #timestamp()

   head(phi)
   sorted_phi = sort.int(phi, decreasing  = TRUE, index.return =
      TRUE)
   head(sorted_phi$ix, 10)
   phiMainScores = which(sorted_phi$ix %in% c(1,3,5))

  #head(phi_num2)
  sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
     return = TRUE)
  #print(head(sorted_phi_num2$ix, 10))
  phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1,2,3))

   head(omega)
   sorted_omeg = sort.int(omega, decreasing  = TRUE, index.return =
      TRUE)
   head(sorted_omeg$ix, 30)
   omegaMainScores = which(sorted_omeg$ix %in% c(1,2,3))

  head(dc)
  sorted_dc = sort.int(dc, decreasing  = TRUE, index.return = TRUE)
  head(sorted_dc$ix, 30)
  dcMainScores = which(sorted_dc$ix %in% c(1,2,3))

  score[r, 1:3] = phi_num2MainScores
```

```
  score[r, 4:6] = omegaMainScores
  score[r, 7:9] = dcMainScores

  if(r %% 25 == 0)
  {
    print(paste0("We␣are␣at␣replicate␣", r, "␣of␣Sim␣1␣run␣", arg
        [[1]]))
    print("The␣column␣scores␣are␣")
    print(colMeans(score[1:r,]))

  }


}

fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    ", "Sim1Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)

########################
# Simulation 2
########################

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    GenCorrFunct.R")
##source("GenCorr Functions.R")
#print("pig")

library(data.table)
library(R.utils)
library(MASS)

arg = cmdArgs()
#arg = list("FIRST", 233,2)

n = 60
p = 3000
q = 6

# n = 120
# p = 1500
# q = 4
#

score = matrix(-1, nrow = 100, ncol = 3) ###These are the scores for
     each method.
#### Best rank, mid rank, worst rank. Perfect score is 1,2,3.
colnames(score) = c("GCorr_Best", "GCorr_Mid", "GCorr_Worst")

# score = matrix(-1, nrow = 100, ncol = 9) ###These are the scores
    for each method.
# #### Best rank, mid rank, worst rank. Perfect score is 1,2,3.
# colnames(score) = c("GCorr_Best", "GCorr_Mid", "GCorr_Worst",
#                     "ZLLZ_Best", "ZLLZ_Mid", "ZLLZ_Worst",
#                     "DC_Best", "DC_Mid", "DC_Worst")
#
set.seed(arg[[2]])

for(r in 1:100)
{
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p)
```

```
{
  #X[, i] = rnorm(n, 0, 5)
  X[, i] = rnorm(n, 3, 1)
  #X[, i] = rpois(n, 2)
}

Y = matrix(0, nrow = n, ncol = q)


# rho = 0.5
# Sigma = diag(q)
# Sigma = rho ^ abs(row(Sigma) - col(Sigma))
#
# scalarMat = t(mvrnorm(3, rep(0, q), Sigma))

# rho = 0.5
# Sigma = diag(q)
# Sigma = rho ^ abs(row(Sigma) - col(Sigma))
#
# scalarMat = t(mvrnorm(3, rep(3, q), Sigma))


sign = (-1)^(rbinom(n=3*q,size = 1, prob = 0.4))
scalarVals = sign*(4*log(n)/sqrt(n) + abs(rnorm(n = 3*q,0, 1)))
scalarMat = matrix(scalarVals, nrow = 3, ncol = q)

####This comes  from DC-SIS. (I've adapted it to a MV setting)

for(m in 1:q)
{
  Y[,m] = exp(X[,c(1,2,3)] %*% scalarMat[,m])
}###Should create a nice little linear model like above (around
    lines 28-34).

testData = cbind(Y, X)

#testData = matrix(rnorm(300, 0,1), ncol = 30, nrow = 10)
#phi = CovarProd_MV_main(data = testData, q = q)
phi_num2 = CorrMat_MV_main(data = testData, q = q, normToUse = arg
    [[3]])

omega = rep(1, p)
#timestamp()
#omega = zllz(data = testData, q=q)
#timestamp()

dc = rep(2,p)
#timestamp()
#dc = DCSIS_MV(data = testData, q=q)
#timestamp()
#
# head(phi)
# sorted_phi = sort.int(phi, decreasing  = TRUE, index.return =
    TRUE)
# head(sorted_phi$ix, 10)
# phiMainScores = which(sorted_phi$ix %in% c(1,3,5))

#head(phi_num2)
sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
    return = TRUE)
#head(sorted_phi_num2$ix, 10)
phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1,2,3))

# head(omega)
```

```
  sorted_omeg = sort.int(omega, decreasing  = TRUE, index.return =
     TRUE)
  # head(sorted_omeg$ix, 30)
  omegaMainScores = which(sorted_omeg$ix %in% c(1,2,3))

  #head(dc)
  sorted_dc = sort.int(dc, decreasing  = TRUE, index.return = TRUE)
  #head(sorted_dc$ix, 30)
  dcMainScores = which(sorted_dc$ix %in% c(1,2,3))

  score[r, 1:3] = phi_num2MainScores
  #score[r, 4:6] = omegaMainScores
  #score[r, 7:9] = dcMainScores

  if(r %% 25 == 0)
  {
    timestamp()
    print(paste0("We␣are␣at␣replicate␣", r, "␣of␣Sim␣2␣run␣", arg
       [[1]]))
    print("The␣column␣scores␣are␣")
    print(colMeans(score[1:r,]))

  }
}

fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   ", "Sim2Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)

#########################
# Simulation 3
#########################

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   GenCorrFunct.R")
#source("GenCorr Functions.R")

library(data.table)
library(R.utils)
library(MASS)

arg = cmdArgs()
#arg = list("FIRST", 890,2)

# n = 60
# p = 3000
# q = 6

n = 120
p = 1500
q = 4


score = matrix(-1, nrow = 100, ncol = 3) ###These are the scores for
    each method.
#### Best rank, mid rank, worst rank. Perfect score is 1,2,3.
colnames(score) = c("GCorr_Best", "GCorr_Mid", "GCorr_Worst")

set.seed(arg[[2]])

for(r in 1:100)
{
```

```
X = matrix(NA, nrow = n, ncol = p)
for (i in 1:p)
{
  #X[, i] = rnorm(n, 0, 5)
  X[, i] = rpois(n, 2)
}

Y = matrix(0, nrow = n, ncol = q)


# rho = 0.5
# Sigma = diag(q)
# Sigma = rho ^ abs(row(Sigma) - col(Sigma))
#
# scalarMat = t(mvrnorm(3, rep(0, q), Sigma))

# rho = 0.5
# Sigma = diag(q)
# Sigma = rho ^ abs(row(Sigma) - col(Sigma))
#
# scalarMat = t(mvrnorm(3, rep(3, q), Sigma))
#

sign = (-1)^(rbinom(n=3*q,size = 1, prob = 0.4))
scalarVals = sign*(4*log(n)/sqrt(n) + abs(rnorm(n = 3*q,0, 1)))
scalarMat = matrix(scalarVals, nrow = 3, ncol = q)
#
# ####This comes  from DC-SIS. (I've adapted it to a MV setting)

for(m in 1:q)
{
  Y[,m] = X[,c(1,2,3)] %*% scalarMat[,m]
}###Should create a nice little linear model like above (around
    lines 28-34).

testData = cbind(Y, X)

#testData = matrix(rnorm(300, 0,1), ncol = 30, nrow = 10)
#phi = CovarProd_MV_main(data = testData, q = q)
phi_num2 = CorrMat_MV_main(data = testData, q = q, normToUse = arg
    [[3]])

#timestamp()
omega = zllz(data = testData, q=q)
#timestamp()

omega = rep(1, p)
dc = rep(2,p)

#timestamp()
dc = DCSIS_MV(data = testData, q=q)
#timestamp()


#head(phi_num2)
sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
    return = TRUE)
#head(sorted_phi_num2$ix, 10)
phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1,2,3))

# head(omega)
sorted_omeg = sort.int(omega, decreasing  = TRUE, index.return =
    TRUE)
# head(sorted_omeg$ix, 30)
omegaMainScores = which(sorted_omeg$ix %in% c(1,2,3))
```

```
  #head(dc)
  sorted_dc = sort.int(dc, decreasing  = TRUE, index.return = TRUE)
  #head(sorted_dc$ix, 30)
  dcMainScores = which(sorted_dc$ix %in% c(1,2,3))

  score[r, 1:3] = phi_num2MainScores
  score[r, 4:6] = omegaMainScores
  score[r, 7:9] = dcMainScores

  if(r %% 25 == 0)
  {
    timestamp()
    print(paste0("We are at replicate ", r, " of Sim 3 run ", arg
        [[1]]))
    print("The column scores are ")
    print(colMeans(score[1:r,]))

  }

}

fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXXX/GenCorr
    /", "Sim3Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)

#########################
# Simulation 4
#########################

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    GenCorrFunct.R")
##source("GenCorr Functions.R")
#print("pig")

library(data.table)
library(R.utils)
library(MASS)

arg = cmdArgs()
###arg = list("FIRST", 888, 2)

# n = 60
# p = 3000
# q = 6

n = 120
p = 1500
q = 4

score = matrix(-1, nrow = 100, ncol = 3) ###These are the scores for
     each method.
#### Best rank, mid rank, worst rank. Perfect score is 1,2,3.
colnames(score) = c("GCorr_Best", "GCorr_Mid", "GCorr_Worst")

# score = matrix(-1, nrow = 100, ncol = 9) ###These are the scores
    for each method.
# #### Best rank, mid rank, worst rank. Perfect score is 1,2,3.
# colnames(score) = c("GCorr_Best", "GCorr_Mid", "GCorr_Worst",
#                     "ZLLZ_Best", "ZLLZ_Mid", "ZLLZ_Worst",
#                     "DC_Best", "DC_Mid", "DC_Worst")
```

```
set.seed(arg[[2]])

for(r in 1:100)
{
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p)
  {
    #X[, i] = rnorm(n, 0, 5)
    X[, i] = rpois(n, 2)
  }

  Y = matrix(0, nrow = n, ncol = q)

  #scalarMat = matrix(rnorm(18, 0,2.5), nrow = 6, ncol = 3)

  # rho = 0.5
  # Sigma = diag(q)
  # Sigma = rho ^ abs(row(Sigma) - col(Sigma))
  #
  # scalarMat = t(mvrnorm(3, rep(0, q), Sigma))

  # rho = 0.5
  # Sigma = diag(q)
  # Sigma = rho ^ abs(row(Sigma) - col(Sigma))
  #
  # scalarMat = t(mvrnorm(3, rep(3, q), Sigma))

  sign = (-1)^(rbinom(n=3*q,size = 1, prob = 0.4))
  scalarVals = sign*(4*log(n)/sqrt(n) + abs(rnorm(n = 3*q,0, 1)))
  scalarMat = matrix(scalarVals, nrow = 3, ncol = q)

  ####This comes  from DC-SIS. (I've adapted it to a MV setting)

  for(m in 1:q)
  {
    Y[,m] = exp(X[,c(1,2,3)] %*% scalarMat[,m])
  }###Should create a nice little linear model like above (around
     lines 28-34).

  testData = cbind(Y, X)

  #testData = matrix(rnorm(300, 0,1), ncol = 30, nrow = 10)
  #phi = CovarProd_MV_main(data = testData, q = q)
  phi_num2 = CorrMat_MV_main(data = testData, q = q, normToUse = arg
     [[3]])

  dc = rep(0,p)
  omega = rep(2,p)
  #timestamp()
  omega = zllz(data = testData, q=q)
  #timestamp()

  #timestamp()
  dc = DCSIS_MV(data = testData, q=q)
  timestamp()
  #
   head(phi)
   sorted_phi = sort.int(phi, decreasing  = TRUE, index.return =
      TRUE)
   head(sorted_phi$ix, 10)
   phiMainScores = which(sorted_phi$ix %in% c(1,3,5))

  #head(phi_num2)
  sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
     return = TRUE)
```

```
  #head(sorted_phi_num2$ix, 10)
  phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1,2,3))

  # head(omega)
  sorted_omeg = sort.int(omega, decreasing  = TRUE, index.return =
     TRUE)
  # head(sorted_omeg$ix, 30)
  omegaMainScores = which(sorted_omeg$ix %in% c(1,2,3))

  #head(dc)
  sorted_dc = sort.int(dc, decreasing  = TRUE, index.return = TRUE)
  #head(sorted_dc$ix, 30)
  dcMainScores = which(sorted_dc$ix %in% c(1,2,3))

  score[r, 1:3] = phi_num2MainScores
   score[r, 4:6] = omegaMainScores
   score[r, 7:9] = dcMainScores

  if(r %% 25 == 0)
  {
    timestamp()
    print(paste0("We are at replicate ", r, " of Sim 4 run ", arg
       [[1]]))
    print("The column scores are ")
    print(colMeans(score[1:r,]))
  }

}

fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   ", "Sim4Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)


########################
# Simulation 5
########################


source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   GenCorrFunct.R")
#source("M:/GCorr/GenCorr Functions.R")
#print("pig")

library(data.table)
library(R.utils)
library(MASS)


#arg = list("FIRST", 132,2)
arg = cmdArgs()


n = 100
p = 1000
q = 4
d = 5

score = matrix(-1, nrow = 100, ncol = 4) ###These are the scores for
    each method.
#### Best rank, worst rank. Perfect score is 1,2.
colnames(score) = c("GCorr_Best", "GCorr_Worst", "X_1X_2", "X_3X_4")
```

```
set.seed(arg[[2]])

for(r in 1:100)
{
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p)
  {
    X[, i] = rnorm(n, 0, 2)
    #X[, i] = rpois(n, 2)
  }

  Y = matrix(0, nrow = n, ncol = q)


   Y[,1] = X[, 1]*X[,2] + 2*X[,4]*X[,3] + rnorm(n,0,1)### plus some
       small error!
    Y[,2] = -X[, 1]*X[,2] +  1.5*X[,4]*X[,3] + rnorm(n,0,1)
     Y[,3] = -2.5*X[, 1]*X[,2] - 2*X[,4]*X[,3] + rnorm(n,0,1)
     Y[,4] = 2*X[, 1]*X[,2] +X[,4]*X[,3] + rnorm(n,0,1)

  testData = cbind(Y, X)


  phi_num2 = CorrMat_MV_Interact(data = testData, q = q, normToUse =
      arg[[3]])


  timestamp()

  #head(phi_num2[1:20, 1:20])
  sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
     return = TRUE)


  phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1001,3003))
  X1X2Score = which(sorted_phi_num2$ix == 1001)
  X3X4Score = which(sorted_phi_num2$ix == 3003)


  score[r, ] = c(phi_num2MainScores, X1X2Score, X3X4Score)


   if(r %% 25 == 0)
   {
     ##timestamp()
     print(paste0("We␣are␣at␣replicate␣", r, "␣for␣Sim␣7␣run␣", arg
        [[1]]))
     print("The␣column␣scores␣are␣")
     print(colMeans(score[1:r,]))


   }


}

fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   ", "Sim7Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)
```

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

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    GenCorrFunct.R")
#source("M:/GCorr/GenCorr Functions.R")
#print("pig")

library(data.table)
library(R.utils)
library(MASS)


#arg = list("FIRST", 132, 2)
arg = cmdArgs()

n = 100
p = 1000
q = 4
d = 5

score = matrix(-1, nrow = 100, ncol = 4) ###These are the scores for
     each method.
#### Best rank, worst rank. Perfect score is 1,2.
###Looking at percentiles may be our best bet. The mean is ~not as
    good as we'd maybe want.

colnames(score) = c("GCorr_Best", "GCorr_Worst", "X_1X_2", "X_3X_4")

set.seed(arg[[2]])

for(r in 1:100)
{
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p)
  {
    X[, i] = rnorm(n, 0, 2)
    #X[, i] = rpois(n, 2)
  }

  Y = matrix(0, nrow = n, ncol = q)


  rho = 0.5
  Sigma = diag(q)
  Sigma = rho ^ abs(row(Sigma) - col(Sigma))

  scalarMat = t(mvrnorm(2, rep(3, q), Sigma))

  for(m in 1:q)
  {

    Y[,m] = cbind(X[,1]*X[,2], X[,3]*X[,4]) %*% scalarMat[m,]
  }

  testData = cbind(Y, X)


  phi_num2 = CorrMat_MV_Interact(data = testData, q = q, normToUse =
      arg[[3]])

  timestamp()

  sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
     return = TRUE)
```

```
  phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1001,3003))

  X1X2Score = which(sorted_phi_num2$ix == 1001)
  X3X4Score = which(sorted_phi_num2$ix == 3003)


  score[r, ] = c(phi_num2MainScores, X1X2Score, X3X4Score)


   if(r %% 5 == 0)
   {
     ##timestamp()
     print(paste0("We␣are␣at␣replicate␣", r, "␣for␣Sim␣5␣run␣", arg
         [[1]]))
     print("The␣column␣scores␣are␣")
     print(colMeans(score[1:r,]))


   }

}

fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   ", "Sim5Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)

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

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
   GenCorrFunct.R")
#source("M:/GCorr/GenCorr Functions.R")
#print("pig")

library(data.table)
library(R.utils)
library(MASS)


#arg = list("FIRST", 132)
arg = cmdArgs()

n = 100
p = 1000
q = 4
d = 5

score = matrix(-1, nrow = 100, ncol = 8) ###These are the scores for
    each method.
#### Best rank, worst rank. Perfect score is 1,2.
###Looking at percentiles may be our best bet. The mean is ~not as
   good as we'd maybe want.

colnames(score) = c("GCorr_Best", "GCorr_Worst", "X_1X_2", "X_3X_4",
     "X_1", "X_2", "X_3", "X_4")

set.seed(arg[[2]])

for(r in 1:100)
{
  X = matrix(NA, nrow = n, ncol = p)
  for (i in 1:p)
  {
```

```
    X[, i] = rnorm(n, 0, 2)
    #X[, i] = rpois(n, 2)
  }

  Y = matrix(0, nrow = n, ncol = q)

  rho = 0.5
  Sigma = diag(q)
  Sigma = rho ^ abs(row(Sigma) - col(Sigma))

  scalarMat = t(mvrnorm(6, rep(3, q), Sigma))

  for(m in 1:q)
  {

    Y[,m] = cbind(X[,c(1,2,3,4)], 3*X[,1]*X[,2], 3*X[,3]*X[,4]) %*%
       scalarMat[m,]
  }


  testData = cbind(Y, X)

  phi_num = CorrMat_MV_main(data = testData, q = q, normToUse = arg
     [[3]])
  phi_num2 = CorrMat_MV_Interact(data = testData, q = q, normToUse =
      arg[[3]])

  timestamp()

  sorted_phi = sort.int(phi_num, decreasing  = TRUE, index.return =
     TRUE)

  sorted_phi_num2 = sort.int(phi_num2, decreasing  = TRUE, index.
     return = TRUE)


  phi_num2MainScores = which(sorted_phi_num2$ix %in% c(1001,3003))
  X1X2Score = which(sorted_phi_num2$ix == 1001)
  X3X4Score = which(sorted_phi_num2$ix == 3003)

  X1Score = which(sorted_phi$ix == 1)
  X3Score = which(sorted_phi$ix == 3)
  X2Score = which(sorted_phi$ix == 2)
  X4Score = which(sorted_phi$ix == 4)


  score[r, ] = c(phi_num2MainScores, X1X2Score, X3X4Score, X1Score,
     X2Score, X3Score, X4Score)


   if(r %% 25 == 0)
   {
     ##timestamp()
     print(paste0("We␣are␣at␣replicate␣", r, "␣for␣Sim␣6␣run␣", arg
        [[1]]))
     print("The␣column␣scores␣are␣")
     print(colMeans(score[1:r,]))


   }

}
```

```
fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    ", "Sim6Results_", arg[[1]],".csv")

fwrite(as.data.frame(score),
       file = fileName,
       showProgress = TRUE,
       col.names=TRUE)

#########################
# Real Data Analysis (Stage One)
#########################

library(data.table)
library(dplyr)
library(tidyr)

#if (!require("tibble")){
#  install.packages("tibble")
#}

library(tibble)


# if (!require("VIM")){
#   install.packages("VIM")
# }
# library(VIM)

# if (!require("mice")){
#   install.packages("mice")
#  }


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

library(mice)

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    GenCorrFunct.R")

#pheno_raw <- fread("h:/GCorr/Real Data GenCorr/pheno_raw.csv") ###
    Pheno_raw is the responses.
#snps_dat_collapsed <- fread("h:/GCorr/Real Data GenCorr/snps.dat.
    collapsed.csv") ##snps are the predictors.


### pheno_raw has the following names:
# PedNum is the unique identifier for a subject (i.e. one mouse)
# SBP  systolic blood pressure
# DBP  diastolic blood pressure
# MAP  mean arterial pressure
# ACR  urinary albumin-to-creatinine ratio
# HDL  high-density lipoprotein cholesterol levels
# CHL  total cholesterol levels
# TRI  triglyceride
# GLU  glucose levels

pheno_raw <- fread("M:/GCorr/Real␣Data␣GenCorr/pheno_raw.csv")


###ACR will be thrown out in the end due to almost all the values
    being 0.

sum(pheno_raw$ACR == 0, na.rm = TRUE)/288 ##89.93056%. So
    essentially 90% of the mice have ACR == 0.

snps_dat_collapsed = as_tibble(snps_dat_collapsed)
```

```
pheno_raw = as_tibble(pheno_raw[,c("SBP", "DBP", "MAP", "HDL", "CHL"
    , "TRI", "GLU")])
## Omit ACR and PedNum. Neither is important in the numerical
    analysis.

head(pheno_raw)

proportionMissing <- function(x){sum(is.na(x))/length(x)}
apply(pheno_raw, MARGIN = 2, proportionMissing) ####
sum(apply(snps_dat_collapsed, MARGIN = 2, proportionMissing) > 0)
###Nothing is missing in the SNPs data.

###Observing missingness using the mice package.

md.pattern(pheno_raw)

aggr_plot <- aggr(pheno_raw, col=c('navyblue','red'),
                  numbers=TRUE, sortVars=TRUE,
                 labels=names(data), cex.axis=.7,
                gap=3, ylab=c("Histogram␣of␣missing␣data","Pattern")
                   )

#marginplot(pheno_raw[c(3,2)])

tempData <- mice(pheno_raw,m=5,maxit=5,meth='pmm',seed=500)

pheno_raw = complete(tempData,1)###Impute with the first imputation.

#### Summarizing the response data.


Y = pheno_raw[,c("SBP", "DBP", "MAP", "HDL", "CHL", "TRI", "GLU")] #
    ##Create The Y Part of the Data.
head(Y)

fwrite(as.data.table(Y), file = "M:/GCorr/Real␣Data␣GenCorr/pheno_
    MiceImputed.csv")
Y = fread("M:/GCorr/Real␣Data␣GenCorr/pheno_MiceImputed.csv")


flippedSNPs = fread("M:/GCorr/Real␣Data␣GenCorr/flippedSNPs.csv",
    sep = ",", header = TRUE)

flippedSNPs[1:10, 1:5]

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

q = 7

realData = cbind(Y, flippedSNPs)

phi_num2 = CorrMat_MV_main(data = realData, q = q, normToUse = 2)

Phi = CorrMat_MV_Interact(data = realData, q = q, normToUse = 2)

mainEff = cbind(colnames(flippedSNPs), phi_num2)

colnames(mainEff) = c("SNP", "phi_j")

fwrite(as.data.table(mainEff), file = "/uufs/chpc.utah.edu/common/
    home/uXXXXXXX/GenCorr/mainEffectsRealDOS.csv", sep = ",")

colnames(Phi) = colnames(flippedSNPs)
rownames(Phi) = colnames(flippedSNPs)

fwrite(as.data.table(Phi), file = "/uufs/chpc.utah.edu/common/home/
    uXXXXXXX/GenCorr/interactEffectsRealDOS.csv",
```

```
       sep = ",", row.names = TRUE, col.names = TRUE)

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

library(data.table)
library(dplyr)
library(tidyr)
library(tibble)
library(R.utils)

library(mice)

source("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr/
    GenCorrFunct.R")

#arg = list(1, 3)

arg = cmdArgs()


pheno_raw <- fread("/uufs/chpc.utah.edu/common/home/uXXXXXXX/GenCorr
    /pheno_raw.csv")


###ACR will be thrown out in the end due to almost all the values
    being 0.

sum(pheno_raw$ACR == 0, na.rm = TRUE)/288 ##89.93056%. So
    essentially 90% of the mice have ACR == 0.

#snps_dat_collapsed = as_tibble(snps_dat_collapsed)
pheno_raw = as_tibble(pheno_raw[,c("SBP", "DBP", "MAP", "HDL", "CHL"
    , "TRI", "GLU")])
## Omit ACR and PedNum. Neither is important in the numerical
    analysis.


tempData <- mice(pheno_raw,m=5,maxit=5,meth='pmm',seed=500)


pheno_raw = complete(tempData,1)###Impute with the first imputation.


#### Summarizing the response data.

#summarise_all(pheno_raw, funs(mean))

Y = pheno_raw[,c("SBP", "DBP", "MAP", "HDL", "CHL", "TRI", "GLU")] #
    ##Create The Y Part of the Data.
head(Y,3)


flippedSNPs = fread(file = '/uufs/chpc.utah.edu/common/home/uXXXXXXX
    /GenCorr/flippedSNPs.csv',
                    sep = ",", header = TRUE)


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

q = 7


lower1 = splitTable[arg[[1]],1]
upper1 = splitTable[arg[[1]],2]

lower2 = splitTable[arg[[2]],1]
upper2 = splitTable[arg[[2]],2]
```

```
X1 = flippedSNPs[,lower1:upper1]
X2 = flippedSNPs[,lower2:upper2]

if(arg[[1]] == arg[[2]])
{
  realData = cbind(Y, X1)
  Phi = CorrMat_MV_Interact_WithinGroup(data = realData, q = q,
     normToUse = 2,
                                        names = colnames(X1))

  fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/
     GenCorr/Real_S"
                    ,arg[[3]], "_S", arg[[4]], ".csv")

  fwrite(as.data.frame(Phi), file = fileName, sep = ',', col.names =
      TRUE)

} else
{
  data = cbind(Y, X1, X2)
  Phi = CorrMat_MV_Interact_SeparateGroup(data = data, q = q,
     normToUse = 2,
                                        namesX1 =  colnames(X1),
                                           namesX2 = colnames(X2))

  fileName = paste0("/uufs/chpc.utah.edu/common/home/uXXXXXXX/
     GenCorr/Real_S"
                    ,arg[[3]], "_S", arg[[4]], "X.csv") #The X is
                       prevent accidental overwrite.

  fwrite(as.data.frame(Phi), file = fileName, sep = ',', col.names =
      TRUE)
}


source("GenCorrFunct.R")

library(SCGLR)
library(data.table)
library(dplyr)
library(MASS)

Y = fread("M:/GCorr/Real␣Data␣GenCorr/pheno_MiceImputed.csv",  sep =
    ",", header = TRUE)
flippedSNPs = fread("M:/GCorr/Real␣Data␣GenCorr/flippedSNPs.csv",
   sep = ",", header = TRUE)


fullData = cbind(Y, flippedSNPs)

#multivariateGlm.fit(Y = Y, comp = X, family = c("Put family of each
    Y component in here"))

###Center each component of Y so that we can a balanced variable for
    each component.
n = 288
q = 7

fam <- rep("gaussian",q)


chooseP1 <- function(data, q = 7, reps = 50) {
  y = scale(data[, 1:q])
  x = data[, -(1:q)]
```

```
  p = dim(x)[2]
  n = dim(x)[1]

  d = 2*round(n/log(n), digits = 0)

  phi = phi_main_scores <- fread("M:/GCorr/Real␣Data␣GenCorr/phi_
     main_scores.csv")


  ranks = p + 1 - rank(phi, ties.method = "random")

  MSPE = matrix(0, ncol = d, nrow = reps)

  fam = rep("gaussian",q)


  for(index in 1:d) ##1:d
  {
    for(rep in 1:reps)
    {
      train = sample(1:n, 0.75 * n, replace = FALSE)


      mydata = cbind(y, dplyr::select(x, colnames(x)[which(ranks <=
         index)]))

      MVform = multivariateFormula(namesY = c("SBP", "DBP", "MAP", "
         HDL", "CHL", "TRI", "GLU"),
                                   namesX =  colnames(x)[which(ranks
                                       <= index)])


      MVregModel = multivariateGlm(formula = MVform ,data = mydata,
         family = fam)

      coefs = sapply(MVregModel,coefficients)

      coefs[is.na(coefs)] = 0 ###Replace NA with 0.

      dplyr::select(x,  colnames(x)[which(ranks <= index)]) %>%
        slice(., -train) ->
        testX

      predict.y = multivariatePredictGlm(Xnew = testX,
                                         family = fam,
                                         beta = coefs)

      MSPE[rep,index] = sum((y[-train,] - predict.y)^2) ###This is
         the trace of the MS(P)E Matrix.


    }
    if(index %% 20 == 0)
    {
      print(paste("We␣are␣at␣index", index))
    }

    meanMSPE = apply(MSPE, MARGIN = 2, mean)
  }

  print(meanMSPE)

  return(min(which(meanMSPE == min(meanMSPE))))
}


timestamp()
set.seed(7919)
```

```
chooseP1(data = fullData, reps = 5)
#### Use p_1 = 85.

timestamp()

GenCorrISIS <- function(data, p1 = 5, q = 7)### Zhong and Zhu
    default t p_1 = 5.
{### They also suggest the dynamic choosing of p_1 in practice
    though.
  y = scale(data[, 1:q])
  x = data[, -(1:q)]
  p = dim(x)[2]
  n = dim(x)[1]

  d = 2*round(n/log(n), digits = 0) ###Should be 102

  phis = fread("M:/GCorr/Real␣Data␣GenCorr/phi_main_scores.csv")


  final.phi = rep(0, p) ###Every marginal feature (the first p_1 and
      the latter as well) has a "final" phi.
  ## The top p_1 features have their original (non-iterative) phis.
  ## The next d-p_1 features have (obstensibly) non-zero phis.


  ranks = p + 1 - rank(phis, ties.method = "random")


  final.phi[which(ranks %in% 1:p1)] = unlist(phis)[which(ranks %in%
     1:p1)]

  X_1 = as.matrix(dplyr::select(x, colnames(x)[which(ranks %in% 1:p1
     )]))
  X_1_c = as.matrix(dplyr::select(x, colnames(x)[which(!(ranks %in%
     1:p1))]))


  hatMatrix = diag(1, nrow = n) - X_1 %*% ginv(t(X_1)%*%X_1) %*% t(X
     _1)
  tempx = hatMatrix %*% X_1_c

  tempData = cbind(y, tempx)

  secondPhi = CorrMat_MV_main(data = tempData, q = q, normToUse = 2)

  secondRanks = p + 1 - rank(secondPhi, ties.method = "random") - p1

  index = which(secondRanks %in% 1:(d-p1))

  for (k in 1:length(index))
  {

    final.phi[index[k]] = secondPhi[k]

    ranks[index[k]] = secondRanks[index[k]] + p1

  }

  result = data.frame(colnames(x), final.phi, ranks)

  rownames(result) = NULL
  colnames(result) = c("SNP", "final.phi", "ranks")


  return(result)
}
```

```
iterResults = GenCorrISIS(data = fullData, p1 = 85, q = 7)
############################################################


finalMarginalEffects = filter(iterResults, final.phi>0)

fwrite(finalMarginalEffects, file = "H:/Gcorr/Real␣Data␣GenCorr/
   finalMargEffects.csv")

########################
# Real Data Analysis (Stage Two)
########################

library(data.table)
library(dplyr)
library(glmnet)
library(readr)


Y = fread("M:/GCorr/Real␣Data␣GenCorr/pheno_MiceImputed.csv",  sep =
    ",", header = TRUE)
flippedSNPs = fread("M:/GCorr/Real␣Data␣GenCorr/flippedSNPs.csv",
   sep = ",", header = TRUE)

flippedSNPs[1:10, 1:5]

Y= as.matrix(Y)

jumboCSV = read_csv("M:/GCorr/Real␣Data␣GenCorr/jumboCSV.csv")
finalMargEffects = read_csv("M:/GCorr/Real␣Data␣GenCorr/
   finalMargEffects.csv")

n = 288
q = 7

d = 2*round(n/log(n), digits = 0)

topInteract = head(arrange(jumboCSV,desc(Phi)), n = 102)
#We just take the top 102 interactions.
#There is no precendent for how to really run an interative approach
    here.

table(topInteract$SNP_1) #77 Unique SNPs
length(table(topInteract$SNP_1)) #77 unique SNPs.

table(topInteract$SNP_2) #13 Unique SNPs
length(table(topInteract$SNP_2)) #13 unique SNPs.

table(finalMargEffects$SNP)
length(table(finalMargEffects$SNP)) #102 unique SNPs.

which(topInteract$SNP_1 %in% finalMargEffects$SNP) ## None

which(topInteract$SNP_2 %in% finalMargEffects$SNP) ## None

length(table(c(topInteract$SNP_2, finalMargEffects$SNP))) ## 115.
##Confirms the above results.

length(table(c(topInteract$SNP_2, topInteract$SNP_1))) ## 83.
## So 83 total SNPs account for 102 interactions.
## We could have as many as 204 unique SNPs if every interaction had
    two
##  previously unseen SNPs.

interactProduct = matrix(-1, nrow = n, ncol = d)
```

```
for(i in 1:d)
{
  ##print(paste(topInteract$SNP_1[i], topInteract$SNP_2[i]))

  SNP_One_Index = which(colnames(flippedSNPs) == topInteract$SNP_1[i
     ])

  SNP_Two_Index = which(colnames(flippedSNPs) == topInteract$SNP_2[i
     ])

  interactProduct[,i] =
    as.matrix(dplyr::select(flippedSNPs, SNP_One_Index)
              *dplyr::select(flippedSNPs, SNP_Two_Index))
}

marginals = matrix(-1, nrow = n, ncol = d)

for(i in 1:d)
{

  SNP_One_Index = which(colnames(flippedSNPs) == topInteract$SNP_1[i
     ])

  marginals[,i] =
    as.matrix(dplyr::select(flippedSNPs, SNP_One_Index))
}

fullData = cbind(marginals, interactProduct)

countNonZeroCoef <- function(coefVect, d = 102)
{
  #Remember to account for the placeholder for the intercept.
  marginalCount = sum(abs(coefVect[2:(d+1)]) > 0)
  interactCount = sum(abs(coefVect[(d+2):(2*d+1)]) > 0)

  print(paste("Marginal Features:", marginalCount))
  print(paste("Interact Features:", interactCount))
  print(paste("TotalNum Features:", marginalCount + interactCount))
  print("-------------------------------------------------------------"
     )

  return(c(marginalCount, interactCount, marginalCount+interactCount
     ))

}

testVect = c(1, 0,2,3, 1, 9,11,0,1)

countNonZeroCoef(testVect, d = 4)

#*****************************************************************
    ************
## alpha = 0.40
    ################################################################
#*****************************************************************
    ************
harold =  cv.glmnet(x = fullData, y = Y, family = "mgaussian",
                    alpha = 0.40, type.measure = "mse", nfolds = 10,
                    standardize.response = TRUE, intercept = FALSE)

minIndex = which(harold$lambda == harold$lambda.min)
harold$cvm[minIndex] ##This is to get the mean 10-fold CV MSE.

#plot(harold)
#harold$lambda.min

coefficients0_40 = coef(harold, s = "lambda.min")
```

```
roundedCoeff = sapply(coefficients0_40,round, digits = 8)

sapply(roundedCoeff, countNonZeroCoef, d = 102)

sum((roundedCoeff$SBP != 0)*(roundedCoeff$DBP != 0)*(roundedCoeff$
    MAP != 0)*(roundedCoeff$HDL != 0)*
  (roundedCoeff$CHL != 0)*(roundedCoeff$TRI != 0)*(roundedCoeff$GLU
     != 0))

###So we need to select a candidate model. I feel that lambda.min
    will be the simplest.

fitHarold = harold$glmnet.fit

tLL = fitHarold$nulldev - deviance(fitHarold)
### deviance() returns the deviance value for each of the 100
    lambdas used.

k = fitHarold$df
n = 288
q = 7

correction = 2*n*(q*k + q*(q+1)/2)/(n - (k + q+1))
sAICc = -tLL+ correction ###A vector of the AICc values.
##We then just need to select which model (lambda) we are using and
    pull out the associated sAICc

##which(sAICc == min(sAICc))
chosenModel = which(fitHarold$lambda == harold$lambda.min)

sAICc[chosenModel]/1000

#################################################################
    ############
## alpha = 0.80
    ##################################################################
#################################################################
    ############
harold =  cv.glmnet(x = fullData, y = Y, family = "mgaussian",
                    alpha = 0.8, type.measure = "mse", nfolds = 10,
                    standardize.response = TRUE, intercept = FALSE)

minIndex = which(harold$lambda == harold$lambda.min)
harold$cvm[minIndex] ##This is to get the mean 10-fold CV MSE.

#plot(harold)
#harold$lambda.min

coefficients0_80 = coef(harold, s = "lambda.min")

roundedCoeff = sapply(coefficients0_80,round, digits = 8)

sapply(roundedCoeff, countNonZeroCoef, d = 102)

sum((roundedCoeff$SBP != 0)*(roundedCoeff$DBP != 0)*(roundedCoeff$
    MAP != 0)*(roundedCoeff$HDL != 0)*
      (roundedCoeff$CHL != 0)*(roundedCoeff$TRI != 0)*(roundedCoeff$
         GLU != 0))

fitHarold = harold$glmnet.fit

tLL = fitHarold$nulldev - deviance(fitHarold)
### deviance() returns the deviance value for each of the 100
    lambdas used.

k = fitHarold$df
n = 288
```

```
q = 7
correction = 2*n*(q*k + q*(q+1)/2)/(n - (k + q+1))
sAICc = -tLL+ correction ###A vector of the AICc values.
##We then just need to select which model (lambda) we are using and
    pull out the associated sAICc

##which(sAICc == min(sAICc))
chosenModel = which(fitHarold$lambda == harold$lambda.min)

sAICc[chosenModel]/1000


#*****************************************************************
    ************
## alpha = 1 LASSO
    ################################################################
#*****************************************************************
    ************
harold =  cv.glmnet(x = fullData, y = Y, family = "mgaussian",
                    alpha = 1, type.measure = "mse", nfolds = 10,
                    standardize.response = TRUE, intercept = FALSE)

minIndex = which(harold$lambda == harold$lambda.min)
harold$cvm[minIndex] ##This is to get the mean 10-fold CV MSE.


coefficientsLASSO = coef(harold, s = "lambda.min")

roundedCoeff = sapply(coefficientsLASSO,round, digits = 8)

sapply(roundedCoeff, countNonZeroCoef, d = 102)

sum((roundedCoeff$SBP != 0)*(roundedCoeff$DBP != 0)*(roundedCoeff$
    MAP != 0)*(roundedCoeff$HDL != 0)*
       (roundedCoeff$CHL != 0)*(roundedCoeff$TRI != 0)*(roundedCoeff$
          GLU != 0))

fitHarold = harold$glmnet.fit

tLL = fitHarold$nulldev - deviance(fitHarold)
### deviance() returns the deviance value for each of the 100
    lambdas used.

k = fitHarold$df
n = 288
q = 7
correction = 2*n*(q*k + q*(q+1)/2)/(n - (k + q+1))
sAICc = -tLL+ correction ###A vector of the AICc values.
##We then just need to select which model (lambda) we are using and
    pull out the associated sAICc

##which(sAICc == min(sAICc))
chosenModel = which(fitHarold$lambda == harold$lambda.min)

sAICc[chosenModel]/1000
```

APPENDIX B

Using the Center for High Performance Computing

Herein I present the basic steps to obtaining access to the Center for High Performance Computing (CHPC) at the University of Utah as well as present some samples of SLURM submissions. This is done to allow readers to gain basic insight into how to work with the CHPC resources for their own cluster computing needs. *Please note:* I am only presenting how I used the CHPC resources in a Windows environment; in many cases, numerous options are available for performing the steps I present below. Much of what I learned initially in regards to CHPC access came from Chapter 5 of (Stephen) Willis Barton's master's thesis [6]. His thesis is still likely the number one resource (outside of the CHPC's own website) for accessing and using CHPC resources. Specifics for how to obtain an account are given on the CHPC website and will not be fully covered here. Briefly, if you are a faculty member at Utah State University, you can register with the CHPC and then request that individuals under your sponsorship or advisement be given access. Each member of your group will need to obtain a uNID (a U of U ID number).

Once a uNID has been obtained, you can log onto the CHPC by using a VPN service to connect to `vpnaccess.utah.edu`. You will need to enter your uNID, the associated password for that account, and then a two-factor-authentication key (as the second "password"). I used DUO-2FA, which provides users with a six digit single use access code each time they connect with the CHPC services. This will vary widely depending on how you choose to access the system. Using the FastX utility one can then interface with the CHPC using a Linux based GUI. [See 6, for the specifics of doing such]. WinSCP can be used to transfer files from a local machine to the CHPC servers. Utah State University has a partition on the Ember cluster. Note that they have 18 total nodes. This is for the *entire* university. Hence, if some well meaning soul in the Engineering Department (not to call them out, but yes, they are the usual culprit) decides he or she needs to use nine nodes for a three day job...and submits

ten such jobs...you get the point: *You will be waiting a long time for your batches to run.* In theory you can contact the CHPC and they can censure people for eating up massive amounts of resources, but then you have to deal with the politics.

B.1 SLURM Submission Examples

Here I present a collection of basic examples for submitting batches to the CHPC clusters using SLURM. Naturally, this only represents a small part of what you can do with SLURM. I am not an expert on SLURM by any means.

Below is a simple R script (`basicScript.R`) that I will be using for the examples:

```
#### basicScript.R

library(data.table)
library(R.utils) ##Need this library to read arguments from the
    command line.

arg = cmdArgs() ###Pulls the command arguments and places them in a
    list.

set.seed(arg[[1]]) #Use the first command line argument to set the
    seed.

fileName = paste0("~directory/", "ourData_", arg[[2]],".csv")
## Use the second command line argument to access the specific data
    we want.

dataSet = fread(fileName)

analyzeData <- function(data, k = 10)
{
        ### Do something in here.
        ###      And then something in here.
}

result = analyzeData(dataSet, k = 50)

## END of basicScript.R
```

We now will use the SLURM submission system to run this script with the desired input. In order to do this we need two files: A `.conf` file and a `.slurm` file. Here is a simple `.conf` file using our `basicScript.R` script.

```
0 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 234  1
1 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 1938 2
2 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 791  3
3 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 2228 4
```

Let's call this file `analysis.conf`. In this file we proscribe that we would like to run an R script and then we list the two command arguments to use for each run of the script. Starting with 0, we also denote a task number for each run of the `basicScript.R` script.

We now will define the `.slurm` file associated with our batch run here:

```
#!/bin/bash
#SBATCH --job-name=TestBatch ##Give the job a name
#SBATCH --time=1-05:00:00        ##Specify the run time. Here we
    have 1 day and 5 hours.
#SBATCH --nodes=1            ##Specify the number of nodes. We are
    using a single node.
#SBATCH --ntasks-per-node=4  ##Specify the number of jobs per node.
    (Should match .conf file)

#SBATCH -o out.%j            ##Create an output file.
#SBATCH -e err.%j            ##Create an error file.

##Specify wheh and if to email you.
## You will be emailed when the job starts, fails, or ends.
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --mail-user=myEmail@gmail.com

##Specify your account information.
## This is the standard account for USU.
#SBATCH --account=usu-em
#SBATCH --partition=usu-em

##Load R
module load R/3.3.2

##Specify what .conf file to use.
srun --multi-prog analysis.conf
```

This is really pretty much just a `bash` script under the hood. Note that `#` does not denote a comment, but instead `##` will act as a comment. Once we have these files constructed, we can now submit the `.slurm` script to SLURM for processing. This is done using the command `sbatch`. For example, if our `.slurm` is `myBatch.slurm`:

```
> sbatch myBatch.slurm
```

SLURM will then place your job in the queue and give you a job ID number. Use the SLURM command `squeue -l` followed by your uNID to output a list of your currently pending and running jobs. This will also tell you how long your job has been running. (The `-l` option gives "long" output).

```
> squeue -l uXXXXXXX
```

The SLURM command `sacct` followed by a specific job ID number can be used to check on specific jobs. The SLURM command `scancel` followed by a job ID number will cancel that specific job. The job needs to be one of your own to cancel it. (Yes, I'll admit, I've tried canceling some of those pesky nine node, three day, super batches).

Here is one more example, this time of a batch using multiple nodes.

```
0 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 234  1
1 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 1938 2
```

```
2 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 791  3
3 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 2228 4
4 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 111  1
5 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 876  2
6 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 135  3
7 Rscript /uufs/chpc.utah.edu/common/home/uXXXXXXX/basicScript.R 753  4
```

In this `.conf` file we list out eight total runs of the R script we wish to run. However, we will split these runs over two nodes now instead of one. We now will define the `.slurm` file associated with our batch run here:

```
#!/bin/bash
#SBATCH --job-name=TestBatch ##Give the job a name
#SBATCH --time=2-15:00:00        ##Specify the run time. Here we
    have 2 days and 15 hours.
#SBATCH --nodes=2            ##Specify the number of nodes. We are
    using two nodes.
#SBATCH --ntasks-per-node=4  ##Specify the number of jobs per node.
    (Should match .conf file)

#SBATCH -o out.%j            ##Create an output file.
#SBATCH -e err.%j            ##Create an error file.

##Specify wheh and if to email you.
##You will be emailed when the job starts, fails, or ends.
#SBATCH --mail-type=BEGIN,FAIL,END
#SBATCH --mail-user=myEmail@gmail.com

##Specify your account information.
##This is the standard account for USU.
#SBATCH --account=usu-em
#SBATCH --partition=usu-em

##Load R
module load R/3.3.2

##Specify what .conf file to use.
srun --multi-prog analysis.conf
```

This batch is submitted to SLURM the same as before. SLURM will split your script runs over two nodes now.

One last note: Parallelizing your computations is the whole point here. While the CHPC nodes are pretty quick and have a decent amount of RAM, they are not a quantum computer or magic black box that is 100 times faster than your desktop. The salience in using the CHPC nodes is that you can parallelize your data analyses. This means that if you need to run a script on a large amount of data, you can partition the data into pieces and run the script on pieces of the data in parallel. ®®

CURRICULUM VITAE

Randall Reese

Contact Information

3900 Old Main Hill
ANSC 321
Logan, UT 84322

3981 S. 4520 W.
West Valley City, UT 84120

801-471-0924
801-556-9803
rreese531@gmail.com
randall.reese@aggiemail.usu.edu
LinkedIn
GitHub: rr1964

Research Interests

Data scaling and variable selection, statistical computing, categorical data analysis, biostatistics, machine learning.

Education

Utah State University, Logan, UT

Ph.D., Statistics, June 2018

- Dissertation Topic: *Feature Selection and Interaction Screening in Ultrahigh Dimensional Feature Spaces.*
- Advisor: Richard Cutler
- GPA: 4.0/4.0

Brigham Young University, Provo, UT

M.S., Mathematics, June 2015

- Thesis: *Topics Pertaining to the Group Matrix: k-Characters and Random Walks.*
- Advisor: Stephen Humphries
- GPA: 4.0/4.0

B.S., Mathematics, April 2013
B.A., Korean, April 2013

- *Magna Cum Laude*
- Minor: Computer Science
- GPA: 3.98/4.00