ROBUST METHODS FOR ANALYZING MULTIVARIATE RESPONSES
WITH APPLICATION TO TIME-COURSE DATA

BY

JI YOUNG KIM

DISSERTATION
Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Statistics
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:
Professor Xuming He, Chair, Director of Research
Professor John I. Marden
Professor Adam T. Martinsek
Professor Feng Liuang
Abstract

Data do not always obey the normality assumption, and outliers can have dramatic impacts on the quality of the least squares methods. We use Huber’s loss function in developing robust methods for time-course multivariate responses. We use spline basis expansion of the time-varying regression coefficients to reduce dimensionality, and downweight the influence of outliers with Huber’s loss function on vectors of residuals. Our research is motivated by time-course microarray experiments to better understand the transcription regulatory network by studying the relationship between gene expressions and transcription factors. The gene expressions are taken as multivariate responses in such studies.

The dissertation consists of three parts. The first part develops a robust score test for linear models by a modification of the well-known Rao’s score test based on Huber’s M estimator. The test statistic is asymptotically normal, and the simulation study suggests that the test has higher power in the presence of outliers than the score test based on the least squares.

In the second part of the dissertation, we propose a robust clustering method based on the EM algorithm applied to a modified multivariate normal density, designed to downweight outliers by Huber’s loss function. We discuss practical algorithms, and assess the performance of the proposed method through Monte Carlo simulations.

Variable selection has received much attention in recent literature. A number of methods have been developed including Lasso. The group Lasso is an extension of the Lasso with the goal of selecting important groups of variables rather than individual variables. In the third part of the dissertation, we propose two robust group Lasso algorithms for the multivariate time-course data, and illustrate the robustness properties of the proposed method for analyzing time-course data.
To my family
Acknowledgments

Standing at this delightful moment and looking back on the past, I am overwhelmed with gratitude to many people. First and foremost I offer my sincerest gratitude to my advisor, Professor Xuming He, who has supported me throughout my Ph.D study with his patience and knowledge. Without his guidance and persistent help, this dissertation would not have been possible. His incredibly sharp intuition on every subject has inspired me to become a motivated thinker in research and scholarship.

I am heartily thankful to Professor Adam Martinsek, for his unforgettable warm encouragement and support. I also would like to express many thanks to Professor John Marden for his support and valuable suggestions on the dissertation, and Professor Feng Liang and Professor Yuguo Chen for their kindness and friendliness. I am very greatful to my friend Guixian for valuable discussions and staff members in the department of Statistics for their kind assistance.

My deepest appreciation to my parents and parents-in-law for their endless support for every single thing I do. Very special thanks from the bottom of my heart to my husband Jung Jin Lee, for his wholehearted support while he was going through his own Ph.D study at the same time. Finally, I thank the most special person to me, Avril Hajin Lee, for enduring the hard time with me, for her incredible understanding and patience, and for her boundless love and trust.
Table of Contents

List of Tables ............................................. vii

List of Figures ............................................. viii

Chapter 1  Robust Score Test for Time-Course Microarray Data ................. 1
  1.1 Introduction ........................................... 1
  1.2 Rao’s score test ........................................ 2
  1.3 Robust score test ...................................... 3
    1.3.1 Model ............................................. 4
    1.3.2 The HS test ...................................... 6
  1.4 Asymptotic normality of the test statistic .......................... 9
  1.5 Simulation study ..................................... 13
  1.6 Application to the yeast cell-cycle microarray data .................. 15
    1.6.1 Backward selection ................................ 15
    1.6.2 Using HS tests and LS tests together ................. 18
  1.7 Conclusion ........................................... 19
  1.8 Figures of Chapter 1 .................................. 20

Chapter 2  Robust EM Clustering via Mixture Model for Multivariate Regression . . 25
  2.1 Introduction .......................................... 25
  2.2 The EM algorithm for Gaussian mixture multivariate regression model .......... 26
    2.2.1 The E-step ........................................ 28
    2.2.2 The M-step ....................................... 28
    2.2.3 Getting the cluster indices from the responsibilities ........... 30
    2.2.4 Example .......................................... 30
  2.3 The robust EM clustering through optimization ........................ 32
    2.3.1 The E-step ........................................ 32
    2.3.2 The M-step ....................................... 33
    2.3.3 Determining c in Huber’s loss function ................... 34
    2.3.4 Examples - Mixtures of multivariate t-distributions with 3 degrees of freedom .. 35
  2.4 The robust EM clustering through weighted least squares estimation ........ 38
    2.4.1 The E-step ........................................ 38
    2.4.2 The M-step ....................................... 39
    2.4.3 Simulation study - Two component mixture with three covariates, generated from
                     multivariate skew t-distributions. ......................... 39
  2.5 Conclusion ........................................... 42
  2.6 Figures of Chapter 2 .................................. 43
# List of Tables

1.1 The power and the type I error estimates for the HS test and the LS test. .................. 15
1.2 The 51 motifs that are already known and experimentally documented, numbered them from 1 to 51. .............................................................. 16
1.3 The Motifs and their p-values in parentheses selected from the HS and the LS backward selection for the cell-cycle data, ordered from the most significant to the least significant. The motifs without p-values indicate their p-values are less than 0.0001. .................. 17
1.4 The significant of the motifs and their p-values in parenthesis before and after removing outliers. 18
2.1 Notation for the contingency table for comparing two partitions. .......................... 31
2.2 The chart for the $x$ and $y$ labels in Figure 2.7. .................................................. 42
3.1 Algorithm of the R Lasso 1. ..................................................................................... 59
3.2 Algorithm of the R Lasso 2. ..................................................................................... 62
3.3 The 96 transcription factors in $X$. For convenience, we numbered them from 1 to 96. .... 64
3.4 The transcription factors selected by R Lasso 1, R Lasso 2 and LS Lasso. The numbers represent the order at which the transcription factor is selected. ................................. 66
### List of Figures

1.1 The normal Q-Q plot of the residuals from the robust linear regression using Huber’s loss function with \( c = 0.01 \) for one of the generated datasets (a) and estimated powers of the HS test and the LS test (b) in Section 1.5. ................................................................. 20
1.2 The normal Q-Q plot (a) and the histogram (b) of the residuals from the robust regression with Huber’s loss function at \( c = 0.01 \) for the cell cycle data in Section 1.6. ................................. 20
1.3 The plots of the HS(30%) and LS residuals for motif MIG1. ................................. 21
1.4 The plots of the HS(30%) and LS residuals for motif Gcr1. ................................. 22
1.5 The plots of the HS(30%) and LS residuals for motif MSN24a. ............................... 23
1.6 Venn diagram of motifs numbered from 1 to 51 selected by HS(10%), HS(20%), HS(30%) and the LS backward selection. ................................................................. 24

2.1 The plot of the response \( Y \) produced by 3-dim multidimensional scaling. The dataset was generated from the mixture of three multivariate normal distributions, as described in Section 2.2.4. Numbers represent the cluster indices. ................................................................. 43
2.2 The coefficient functions for each of three mixture components in Model 1 given in Section 2.3.4. For each of the \( m^{th} \) mixture component, \( m = 1, 2, 3 \), the plots of \( \beta^m_p(t) \), for \( t = 3, \cdots, 30 \) and \( p = 1, \cdots, 4 \) are shown. ................................................................. 43
2.3 3-dim multi-dimensional scaling for the responses in Model 1 of Section 2.3.4 with true cluster indices attached. ................................................................. 44
2.4 2-dim multi-dimensional scaling for the responses in Model 2 of Section 2.3.4 with true cluster indices attached. ................................................................. 44
2.5 Log-likelihoods of the data from Model 3 in Section 2.3.4 up to 150 iterations for \( c=8 \). .............. 45
2.6 True indices and indices for the data from Model 3 in Section 2.3.4 from the REM 1 at the 36th iteration. ................................................................. 45
2.7 ARI values from REM 2 against other clustering algorithms. ‘hubercov’ and ‘hubernocov’ represent REM 2 with and without covariates, respectively. ‘lscov’ and ‘lsnocov’ represent LSEM with and without covariates. ‘mc’ represents Mclust, and ‘km’ represents K-means. .... 46

3.1 The distribution of the Mahalanobis distances of 240 genes (a) and the plot of Genes vs. Distances (b). Black circles represent the genes selected for further investigation. ....................... 68
Chapter 1

Robust Score Test for Time-Course Microarray Data

1.1 Introduction

Microarray time-course experiments typically involve gene expression measurements for thousands of genes over a few time points (Tai and Speed, 2005). The measurements at several consecutive time points are correlated, and the within-subject correlation needs to be properly accounted for in statistical inference. Transcription factors (TF) are proteins involved in gene transcriptions, which regulate genes by binding to their recognition sites. The common pattern of the binding sites for a TF is called a motif, usually modeled by a position-specific weight matrix (PWM) or a sequence logo. Identifying important transcription factor binding motifs (TFBMs) that have a significance influence on gene expressions is of great interest in biological research to understand the transcriptional regulatory network of a genome. Discovered set of important motifs can reveal important biological stories behind the genetic mechanism. De novo motif finding algorithms such as AlignACE (Hughes et al., 2000), MEME (Liu et al., 2002), MDscan (Bailey and Elkan, 1994) and Weeder (Pavesi et al., 2004) typically identifies huge number of motifs. Hence variable selection is a key step to obtain a subset of meaningful motifs, which motivated our work in this chapter.

Hypothesis testing is one of the traditional variable selection methods. We can determine whether a variable is important by testing whether its coefficient is zero. Among well-known large-sample tests, Rao’s score test is often preferred because of its advantage that it can be carried out by estimating the parameters under the null hypothesis, unlike other tests such as the likelihood ratio test or the Wald test, which require estimating under the full model. Since modeling the relationship between gene expressions at several time points and TFBMs involve multivariate regression models with a large number of parameters, the Rao’s score test would be the best choice to reduce the computational cost.

The assumption of the Gaussian errors has been frequently used for statistical modeling because of its computational convenience. However, real data usually do not completely satisfy the Gaussian assumptions, which can have effects on the quality of the classical normality-based statistical analysis. Especially, microarray data have potentially many outliers from various sources - array-specific effects, gene-specific...
effects, dye-specific effects, background noise and artifacts and preparation effects, etc. The consequences of violation of the normality assumption can be severe in such cases.

In this chapter, we propose a robust score test (HS test) for a multivariate regression model by borrowing the idea of the Rao’s score test under the multivariate Gaussian assumption. The goal is to identify the important TFBMs by modeling the gene expression levels on gene-specific TFBM scores. While the traditional Rao’s score test involves computing the maximum likelihood estimator (MLE) of the parameter (or equivalently, the least squares estimator under the assumption of Gaussian errors), the HS test uses a robust estimate of the regression parameter by downweighting genes with large residuals.

Our study is specialized in the time-course microarray data, which led us to develop a functional response model with varying coefficients. The functional structure of the time course data allows us to reduce the dimensions of the parameters by variety of approximations, and the originality of the HS test from the Rao’s score test enables us to estimate the parameters under the null hypothesis, which provides even more dimension reduction.

The derivation of the HS test is given in Section 1.3, along with the main results in Section 1.4. The test statistic has nice asymptotic properties, and the simulation study in Section 1.5 shows that the HS test is well-defined and can outperform the score test with the least squares estimates (the LS test) when the data are not normally distributed. In the application to the Yeast cell-cycle data in Section 1.6, we perform a series of the HS and the LS tests through the backward elimination and illustrate differences between the HS and the LS test. We show that both tests are useful in finding a subset of important motifs. We start the chapter with a brief review of the traditional Rao’s score test and M-estimators in the following section.

1.2 Rao’s score test

Suppose that $\theta = (\alpha, \beta)$ with $\alpha \in \mathbb{R}^p$ and $\beta \in \mathbb{R}^q$, and that we test

$$H_0 : \alpha = \alpha_0 \quad \text{vs} \quad H_1 : \alpha \neq \alpha_0$$

for some constant $\alpha_0$.

Let $l(\theta; X)$ be the log likelihood, with the score function

$$\Psi_n(\alpha, \beta) = \frac{\partial}{\partial \alpha} l(\theta; X).$$
Then the test statistics for the Rao's score test (Rao, 1948) is given by

$$R_n = n^{-1}\Psi_n(\alpha_0, \hat{\beta})' S_n^{-1} \Psi_n(\alpha_0, \hat{\beta}),$$

where $S_n$ is the covariance matrix of $n^{-1/2} \Psi_n(\alpha_0, \hat{\beta})$, and $\hat{\beta}$ is the MLE of $\beta$ under $H_0$. In other words, $\hat{\beta}$ satisfies

$$\sum_{i=1}^{n} \frac{\partial}{\partial \beta} \log f(\alpha_0, \beta)(x_i) = 0.$$

It is well-known that $n^{-1/2} \Psi_n(\alpha_0, \hat{\beta})$ is approximately $N(0, S_n)$, and thus $R_n$ is approximately $\chi_p^2$ distributed.

### 1.3 Robust score test

Even a single outlier can dominate the least squares estimator (Rousseeuw and Leroy, 1987). One step toward a more robust regression estimator is the use of M-estimators (Huber, 1981). They are based on the idea of replacing the squared residuals by another function of the residuals, leading to the problem of

Minimizing $\sum_{i=1}^{n} \rho(r_i)$ with respect to $\hat{\theta}$, \hspace{1cm} (1.1)

where $\rho$ is a symmetric function (i.e., $\rho(-t) = \rho(t)$ for all t) with a unique minimum at zero. Differentiating this expression with respect to the regression coefficients $\theta$ yields

$$\sum_{i=1}^{n} \psi(r_i)x_i = 0,$$ \hspace{1cm} (1.2)

where $\psi$ is the derivative of $\rho$, and $x_i$ is the row vector of explanatory variables of the $i$th case.

The consistency of the M-estimators is well-known provided that the root of the estimating equation (1.2) is unique and isolated (Lehmann and Casella (1998), p.513). To lower the sensitivity in the LS objective function to the presence of data contamination, Huber proposed to use

$$\rho(r) = \begin{cases} r^2 & \text{for } |r| \leq c \\ 2c|r| - c^2 & \text{for } |r| > c, \end{cases} \hspace{1cm} (1.3)$$

with a threshold value $c$, which plays a role of a robustness tuning parameter.

As shown in section 1.2, the Rao’s score test estimates the parameters in the test statistic under the null
hypothesis, which distinguishes itself from other tests that require estimation of the full model. Our proposed test borrows the idea from this traditional Rao’s score test under the assumption of Gaussian multivariate error, but in computing the score statistic, we use Huber’s M-estimators instead of the maximum likelihood estimators. M-estimators with the loss function in (3.24) are statistically more efficient at models with Gaussian errors than the $L_1$ regression, while at the same time they are still robust with respect to outliers (Huber, 1981), by moderating the speed of increase of the squared norm of the residuals, and therefore deflating the undesirable effects of outliers on the test. Huber’s M-estimators also provide nice properties, which enables us to derive the asymptotic distribution of the test statistic. However, our test can be generalized to the test with other M-estimators under suitable regularity conditions.

In the follow section, we introduce our robust score test, which we named as the HS test, followed by the section of the asymptotic distribution of the HS test. In Section 1.5, we illustrate the performance of the HS test through a simulation study, and show that the test has higher power than the traditional one based on the least squares estimator in the presence of outliers. We report its application to the real data in Section 1.6.

1.3.1 Model

Let $Y$ be an $n \times T$ matrix of the multivariate response, $X^o$ an $n \times (P + 1)$ matrix of the column of ones and the $P$ covariates, and $\alpha^o$ the $(P+1) \times T$ regression coefficient. Our model is given by

$$y_i = x_i^o \alpha^o + e_i,$$

where $y_i$ and $x_i^o$ are the $i$-th rows of $Y$ and $X^o$, respectively and independent errors $e_i$ have mean 0 and variance-covariance matrix $\Sigma_{T \times T}$. We assume that the distribution of $e_i$ is symmetric.

The coefficient $\alpha^o$ can be regarded as the usual coefficient matrix in the multivariate regression model when we do not know in what conditions the data were collected. However, when the data are measurements from temporal experiments, it is often more desirable to consider its ordering in time. In the time-course data, the responses are likely to be time-varying, and so are the coefficients. By treating those data only as multivariate data with certain correlation structure, we may not benefit from its important time-varying relationships.

To take advantage of the time-varying relationship in the data, we employ the basis representation of the coefficients for approximation. For each $\alpha^o_p = (\alpha_p(1)^o, \ldots, \alpha_p(T)^o)$, consider a basis representation of
\( \alpha^o \) through B-splines. Let \( K \) be the number of basis functions, and suppose that

\[
\alpha_p(t)^o = \sum_{k=1}^{K} \beta_{pk}^o B_k(t),
\]

where \( B_k(t) \) is the \( k \)th basis function at time \( t \), and \( \beta_{pk}^o \) is the coefficient of \( B_k(t) \). Then we can represent \( \alpha^o \) in terms of \( \beta_{pk}^o \)'s and \( B_k(t)'s \) in the following way:

\[
\alpha^o = \beta^o F, \tag{1.5}
\]

where

\[
\beta^o = \begin{bmatrix}
\beta_0(1) & \cdots & \beta_0(K) \\
\vdots & \ddots & \vdots \\
\beta_p(1) & \cdots & \beta_p(K)
\end{bmatrix}
\quad \text{and} \quad
F = \begin{bmatrix}
B_1(1) & \cdots & B_1(T) \\
\vdots & \ddots & \vdots \\
B_K(1) & \cdots & B_K(T)
\end{bmatrix}. \tag{1.6}
\]

By plugging in (1.5) into (1.4), we get

\[
y_i = x_i^o \beta^o F + e_i. \tag{1.7}
\]

Consider the singular value decomposition \( F = UDV' \), where \( U \) and \( V' \) are orthogonal matrices and \( D \) is a diagonal matrix. If we take \( H' = V D^{-1} U' \), then \( FH' = I_{K \times K} \). By multiplying both sides of (1.7) by \( H \), we get

\[
y_i H' = x_i^o \beta^o + e_i H'. \tag{1.8}
\]

Since the transformation was done for each \( Y_i \) without involving other rows, the new response vectors \( y_{i,\text{new}} = y_i H', \ i = 1, \cdots , n \) are still independent with mean \( x_i \beta^o \). The covariance matrix is now changed from \( \Sigma \) to \( H' \Sigma H \). Writing in the matrix form, the model is

\[
Y_i = X_i^o \beta^o + \mathcal{E}_i, \tag{1.9}
\]

where \( Y_i = y_i H' \) and the independent errors \( \mathcal{E}_i = e_i H' \) have mean 0 and variance covariance \( H' \Sigma H_{K \times K} \) for \( i = 1, \cdots , n \).

We wish to test whether a particular variable is significant on the response. In other words, for a
particular $p$th covariate, we want to test the null hypothesis

$$H_0 : \beta_p(k) = 0 \text{ for } k = 1, 2, \cdots, K. \quad (1.10)$$

against the alternative $H_a : \text{not all of the component of } \beta_p \text{ is 0}$. This is equivalent to testing

$$H_0 : \alpha_p(t) = 0 \text{ for } t = 1, 2, \cdots, T,$$

since $X^o$ has not been changed.

### 1.3.2 The HS test

The score statistic for our model is derived from the traditional Rao test when the errors are from the multivariate normal distribution with mean 0 and the variance-covariance matrix $H' \Sigma H$. For convenience, we go back to original notations by denoting $Y_i = y_i$ and $E_i = e_i$ from now on. Recall that the $X^o$ is the matrix of the column of ones and the $P$ covariates, which can be written as

$$X^o = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1P} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{nP} \end{pmatrix}$$

and $\beta^o$ is the matrix of the coefficients in (1.9), as shown in (1.6).

In order to perform the Rao’s score test for testing (1.10), we need to define the design matrix and the regression coefficient under $H_0$ as follows:

$$X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1,p-1} & x_{1,p+1} & \cdots & x_{1P} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{n,p-1} & x_{n,p+1} & \cdots & x_{nP} \end{pmatrix} \quad \text{and } \beta^o = \begin{bmatrix} \beta_0(1) & \cdots & \beta_0(K) \\ \vdots & \ddots & \vdots \\ \beta_{p-1}(1) & \cdots & \beta_{p-1}(K) \\ \beta_{p+1}(1) & \cdots & \beta_{p+1}(K) \\ \vdots & \ddots & \vdots \\ \beta_P(1) & \cdots & \beta_P(K) \end{bmatrix},$$

where $X$ is the matrix of covariates without the $p^{th}$ covariate, and $\beta$ is the matrix of the regression coefficients without the row for the $p^{th}$ covariate.
The distribution of \( y_i \) under the multivariate normal assumption is
\[
 f_\Theta(y_i) = \frac{1}{(2\pi)^T/2|(H'\Sigma H)|^{1/2}} \exp \left( -\frac{1}{2}(y_i - x_i^o \beta_o)'(H'\Sigma H)^{-1}(y_i - x_i^o \beta_o) \right),
\]
where \( \Theta = (\beta_o, \Sigma) = (\tilde{\beta}_p ; \beta, \Sigma) \). The likelihood of \( Y \) is
\[
 f_\Theta(Y) = \prod_{i=1}^n f_\Theta(y_i) = \left[ \frac{1}{(2\pi)^T/2|(H'\Sigma H)|^{1/2}} \right]^n \exp \left( -\frac{1}{2} \sum_{i=1}^n (y_i - x_i^o \beta_o)'(H'\Sigma H)^{-1}(y_i - x_i^o \beta_o) \right).
\]

The score statistic in the original Rao's score test is given by
\[
 \Psi(\tilde{\beta}_p = \tilde{0}; \beta, \Sigma) = \frac{\partial}{\partial \tilde{\beta}_p} \log f_\Theta(Y)|_{\tilde{\beta}_p = \tilde{0}} = \sum_{i=1}^n x_{ip}(y_i - x_i \beta)'(H'\Sigma H)^{-1}.
\]

If \( \beta \) and \( \Sigma \) are replaced by MLE's under \( H_0 \), we arrive at the score for the Rao test. The MLE under the Gaussian model is the least squares estimator.

**Working Assumption of Independence**

He et al. (2003) found that the estimator under working assumption of independence is computationally simple and yet has good relative performance when covariates are invariant over time or when the within-subject correlations are small. They also found that its relative performance in finite samples is also found to be more favorable than suggested by the asymptotic comparisons. Following their idea, we derive the test under the working assumption of the constant variance and the independence of the \( e_i's \) such that \( \Sigma = \sigma^2 I \), where \( \sigma \) is to be estimated from the data.

**The score statistic of the HS test**

Let \( \Omega_{K\times K} = H'\Sigma H = \sigma^2 H' H \). Instead of using the MLE that is sensitive to outliers, we estimate \( \beta \), or equivalently \( \beta^o \) under \( H_0 \), by minimizing
\[
 R_1 = \sum_{i=1}^n \rho \left( \sqrt{(y_i - x_i \beta)' \Omega^{-1}(y_i - x_i \beta)} \right),
\]
where \( \rho \) is the Huber’s loss function with a threshold value \( c \).

Optimizing (1.11) is computationally intensive, so we can instead minimize
\[
 R_2 = \sum_{m=1}^{nK} \rho(u_m - v_m),
\]
(1.12)
where $\omega_{jk}$ represents the component of $\Omega^{-1}$ located at the $j^{th}$ row and the $k^{th}$ column,

$$
u = \begin{bmatrix}
\sum_{k=1}^{K} y_1(k) \omega_{k1} \\
\vdots \\
\sum_{k=1}^{K} y_n(k) \omega_{k1} \\
\vdots \\
\sum_{k=1}^{K} y_1(k) \omega_{kK} \\
\vdots \\
\sum_{k=1}^{K} y_n(k) \omega_{kK}
\end{bmatrix}, \quad v = \begin{bmatrix}
x_1 \omega_{11} & \cdots & x_1 \omega_{1K} \\
\vdots & \ddots & \vdots \\
x_n \omega_{11} & \cdots & x_n \omega_{1K} \\
\vdots & \ddots & \vdots \\
x_1 \omega_{K1} & \cdots & x_1 \omega_{KK} \\
\vdots & \ddots & \vdots \\
x_n \omega_{K1} & \cdots & x_n \omega_{KK}
\end{bmatrix},
$$

and $\gamma = [\beta(1)', \beta(2)', \ldots, \beta(K)']'$, where $\beta(k) = (\beta_0(k), \ldots, \beta_P(k))'$ for $k = 1, \ldots, K$. Given the weights $\omega_{jk}'$ for $j, k = 1, \ldots, K$, the rlm function in R can be used to solve $R_2$.

With $\hat{\beta}$ estimated either from $R_1$ or from $R_2$, we propose a score statistic for the HS test

$$\Psi_n = \sum_{i=1}^{n} x_{ip} \psi \left( (y_i - x_i \hat{\beta}) \Omega^{-1/2} \right)$$

$$= \sum_{i=1}^{n} x_{ip} \psi \left( (y_i - x_i \hat{\beta}) \Omega^{-1/2} \right),$$

where $\psi(e)$ is the first derivative of Huber’s loss function applied to each component of $(y_i - x_i \hat{\beta}) \Omega^{-1/2}$, which can be written as

$$\psi(t) = \min(c, \max(t,-c)). \quad (1.13)$$

By taking Huber’s loss on each component of the residuals, we can downweight outliers. From now on, we denote the robust score test as the IHS test and the original score test based on the least squares as the LS test.

**Determining $c$**

The choice of the threshold value $c$ in Huber’s loss function depends on how many outliers we wish to downweight. In the univariate case, we typically choose $c$ between 1 and 2 times a scale estimate of the residuals, but there is no common choice of $c$ in multivariate case. To determine the value of $c$ for the HS test, we take a two-step procedure. We first set the threshold to a number close to zero and fit the model. This choice of the threshold downweights most of the residuals, and most of the outliers are revealed as high residuals. Then we save these residuals and set the value of $c$ by taking the absolute value of the 100\text{th}
and $100(1-q)$th percentile of the residuals whichever is larger, where $q \in (0, 1)$ is the desired proportion of downweighting.

### 1.4 Asymptotic normality of the test statistic

The test statistic of the HS test is given by

$$R_H = n^{-1} \Psi_n \hat{\Sigma}^{-1} \Psi_n', \quad (1.14)$$

where $\hat{\Sigma}$ is the covariance matrix of $n^{-1/2} \Psi_n$. The following lemma and theorem imply that $R_H$ is asymptotically $\chi^2_K$ distributed.

**Lemma 1.4.1.** Consider a model $y_i = x_i \beta_0 + e_i$, where $e_i$ are i.i.d with zero mean, and $W$ is the square root of the covariance matrix of $e_i$. Let $\beta_0 \in \Theta$ be the true parameter such that $\sum_{i=1}^n E x_i \psi((y_i - x_i \beta_0)W) = 0$ for some known matrix $W$. Let $f(e)$ be the p.d.f. of $e_i$, and $W_\ell$, $\ell = 1, \cdots, K$ be the $\ell$th column of $W$.

Assume that

(A1) $\sqrt{n} \lVert \hat{\beta} - \beta_0 \rVert^2 \overset{P}{\rightarrow} 0$, as $n \rightarrow \infty$,

(A2) For $\ell = 1, \cdots, K$, $x_\ell^o$ and $\frac{\partial^2}{\partial \beta^2} E \psi((y_i - x_i \beta)W_\ell)$ are bounded. In other words, there exist positive constants $M_1$ and $M_2$ such that $\lVert x_\ell^o \rVert \leq M_1$ and $\left\| \frac{\partial^2}{\partial \beta^2} E \psi((y_i - x_i \beta)W_\ell) \right\| \leq M_2$, for any $\beta$ where the expectations are taken at the underlying distributions of $y_i$.

(A3) $\frac{\partial^2}{\partial \beta^2} f(e)$ is continuous.

Then

$$\frac{1}{\sqrt{n}} \sum_{i=1}^n x_{ip} \left( \psi((y_i - x_i \hat{\beta})W) - \psi((y_i - x_i \beta_0)W) \right) = o_P(1).$$

**Proof.** First we transform $X^o$ so that $X_p$ and the columns of $X^{o[-p]}$ are orthogonal, that is,

$$\sum_{i=1}^n x_{ip} x_i = 0.$$

Consider the $\ell$th component of the $\frac{1}{\sqrt{n}} \sum_{i=1}^n x_{ip} \left( \psi((y_i - x_i \hat{\beta})W) - \psi((y_i - x_i \beta_0)W) \right)$. We will prove that the theorem holds for each $\ell$, $\ell = 1, \cdots, K$. We start the proof by applying Lemma 4.6 in He and Shao (1996). The following conditions should be met to use the lemma.

(C1) There exist $r > 0$, $d_0 > 0$ and a sequence of positive numbers $a_i$, $i \geq 1$ such that $E u^2(y_i, x_i, \beta, d) \leq a_i^2 d^r$ for $\lVert \beta - \beta_0 \rVert \leq d_0$ and $d \leq d_0$, where $u(y, x, \beta, d) = \sup_{\lVert \tau - \beta \rVert \leq d} \lvert x_{ip} (\psi((y_i - x_i \tau)W_\ell) - \psi((y_i - x_i \beta)W_\ell)) \rvert$. 


(C2) For some decreasing sequence of $d_n > 0$ such that $d_n = O(d_{2n}) = o(1)$,

$$\max_{1 \leq i \leq n} u(y_i, x_i, \beta, d_n) = O(A_n^{1/2}/d_n^{1/2}(\log n)^{-2}) \text{ a.s., where } A_i = \sum_{i=1}^{n} d_i^2.$$ 

Take any $d_0$ such that $0 < d_0 < 1$, and $r = 1$. Let $a_i = |x_{ip}|\|x_i\|\|W_\ell\|$.

The function $u(y, x, \beta, d)$ satisfies

$$u(y_i, x_i, \beta, d) = \sup_{\|\tau - \beta\| \leq d} |x_{ip}(\psi((y_i - x_i \tau)W_\ell) - \psi((y_i - x_i \beta)W_\ell))|
= \sup_{\|\tau - \beta\| \leq d} |x_{ip}|((y_i - x_i \tau)W_\ell - (y_i - x_i \beta)W_\ell|
\leq \sup_{\|\tau - \beta\| \leq d} |x_{ip}|\|x_i\|\|\tau - \beta\|\|W_\ell\|
\leq |x_{ip}|\|x_i\|\|W_\ell\|d
= a_i d.$$ 

Then the condition (C1) holds because $Eu^2(y_i, x_i, \beta, d) \leq a_i^2 d^2 \leq a_i^2 d$ for $d < d_0 < 1$.

Now take $d_n = \frac{1}{(\log n)^{\gamma}} = o(1)$. Then $d_{2n} = \frac{1}{(\log 2n)^{\gamma}} = \frac{1}{(\log 2+\log n)^{\gamma}}$. Since

$$\frac{d_n}{d_{2n}} = \frac{(\log 2 + \log n)^4}{(\log n)^4} \rightarrow 1,$$

it satisfies $d_n = O(d_{2n})$.

Note that for each $i = 1, \ldots, n$,

$$\left| \frac{u(y_i, x_i, \beta_0, d_n)}{A_n^{1/2}/d_n^{1/2}(\log n)^{-2}} \right| \leq \frac{a_i d_n}{(\sum_{i=1}^{n} a_i^2)^{1/2}d_n^{1/2}(\log n)^{-2}} \leq \sqrt{d_n}(\log n)^2 = 1.$$ 

Therefore the condition (C2) holds for our choice of $r$, $d_0$, $d_n$, $a_i$.

Lemma 4.6 in He and Shao (1996) implies that

$$\limsup_{n \to \infty} \sup_{\|\tau - \beta_0\| \leq d_n} \frac{Z_n(\tau, \beta_0)}{(A_n d_n^2 + 1)^{1/2}(\log (n + A_n))^{1/2}} \leq C \text{ a.s.,}$$

for some constant $C < \infty$, where

$$Z_n(\tau, \beta) = \sum_{i=1}^{n} x_{ip}(\psi((y_i - x_i \tau)W_\ell) - \psi((y_i - x_i \beta)W_\ell) - E(\psi((y_i - x_i \tau)W_\ell)) + E(\psi((y_i - x_i \beta)W_\ell))).$$

Therefore we have

$$Z_n(\tau, \beta_0) \leq C(A_n d_n^2 + 1)^{1/2}(\log (n + A_n))^{1/2},$$
and equivalently,

\[
\frac{Z_n(\tau, \beta)}{\sqrt{n}} \leq C_4 \left( \frac{(A_n d_n^* + 1)^{1/2}(\log \log(n + A_n))^{1/2}}{\sqrt{n}} \right).
\]

(1.15)

We can easily show that the right hand side of (1.15) goes to 0 as \( n \to \infty \):

\[
\left( C_4 \frac{(A_n d_n^* + 1)^{1/2}(\log \log(n + A_n))^{1/2}}{\sqrt{n}} \right)^2
\leq C^2 M_1^2 M_2^2 d_n^* \left( \log \log(n + A_n) \right)
\leq C^2 M_1^2 M_2^2 \frac{1}{(\log n)} \log \log(n + A_n) + C_2 \log \log \left( M_1^2 M_2^2 + 1 \right)
\leq 0.
\]

If We plug in \( \hat{\beta} \) into \( \tau \) in (1.15), then we have \( \frac{1}{\sqrt{n}} Z_n(\hat{\beta}, \beta_0) \xrightarrow{P} 0 \). In other words,

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left( \psi((y_i - x_i \hat{\beta}) W_i) - \psi((y_i - x_i \beta_0) W_i) - E(\psi((y_i - x_i \hat{\beta}) W_i)) + E(\psi((y_i - x_i \beta_0) W_i)) \right) \xrightarrow{P} 0.
\]

Now we will show that

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left( E(\psi((y_i - x_i \hat{\beta}) W_i)) - E(\psi((y_i - x_i \beta_0) W_i)) \right) \xrightarrow{P} 0.
\]

(1.16)

We need to show that \( E(\psi(y_i - x_i \beta) W_i) \) is twice differentiable. Because \( y_i = x_i \beta_0 + e_i \), and \( e_i \)'s are i.i.d, we have

\[
E\psi(y_i - x_i \beta)
= \int_{c}^{\infty} \psi(x_i(\beta_0 - \beta) + e) f(e) de
= -c \int_{c}^{\infty} \psi(x_i(\beta_0 - \beta) + e) f(e) de + \int_{c}^{\infty} \psi(x_i(\beta_0 - \beta)) (x_i(\beta_0 - \beta) + e) f(e) de + c \int_{c}^{\infty} \psi(x_i(\beta_0 - \beta)) f(e) de.
\]

(1.17)

Since we assumed that \( \frac{\partial^2}{\partial \gamma^2} f_\beta(y) \) is continuous in condition (A3), by Leibnitz’s Rule (Kaplan, 2003), the first and third integrals of (1.17) are twice differentiable. The second integral of (1.17) is also twice differentiable because the integrand \( (x_i(\beta_0 - \beta) + e) f(e) \) has continuous second derivative. Therefore, \( E\psi(y - x \beta) \) is twice
differentiable, which enables us to use the Taylor’s expansion.

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left[ E(\psi(y_i - x_i \hat{\beta})W_\ell) + E(\psi(y_i - x_i \beta_0)W_\ell) \right]
= \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left[ \frac{\partial^2}{\partial \beta^2} E(\psi(y_i - x_i \beta)W_\ell) \bigg|_{\beta=\beta_0} (-x_i(\hat{\beta} - \beta)W_\ell) + \frac{1}{2} \frac{\partial^2}{\partial \beta \partial \beta} E(\psi(y_i - x_i \hat{\beta})W_\ell) \bigg|_{\beta=\beta_0} (x_i(\hat{\beta} - \beta_0)W_\ell)^2 \right]
\]

(1.18)

for some \( \tilde{\beta} \) between \( \beta \) and \( \hat{\beta} \).

The first part of (1.18) goes to 0 in probability since \( \sum_{i=1}^{n} x_{ip}x_i = 0 \). The second part of (1.18) also goes to 0 in probability since

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left( \frac{1}{2} \frac{\partial^2}{\partial \beta^2} E(\psi((y_i - x_i \beta)W_\ell)) \bigg|_{\beta=\tilde{\beta}} ((x_i(\tilde{\beta} - \beta_0)W_\ell)^2) \right)
\]

\[
\leq \frac{1}{\sqrt{n}} \sup_{x \in \mathbb{R}^p} \left( \frac{1}{2} \frac{\partial^2}{\partial \beta^2} E(\psi((y_i - x_i \beta)W_\ell)) \bigg|_{\beta=\tilde{\beta}} ((x_i(\tilde{\beta} - \beta_0)W_\ell)^2) \right)
\]

\[
\leq \frac{1}{2 \sqrt{n}} \sum_{i=1}^{n} \left( x_{ip} \right)^2 \frac{\partial^2}{\partial \beta^2} E(\psi((y_i - x_i \beta)W_\ell)) \bigg|_{\beta=\tilde{\beta}} \|x_i\|^2 \|\tilde{\beta} - \beta_0\|^2 \|W_\ell\|^2
\]

\[
\leq \frac{1}{2 \sqrt{n}} \sum_{i=1}^{n} M_1 M_3 M_2 \|\tilde{\beta} - \beta_0\|^2 \|W_\ell\|^2
\]

\( \xrightarrow{p} 0 \).

Therefore, (1.16) is true and consequently we prove that

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left( \psi((y_i - x_i \hat{\beta})W_\ell) - \psi((y_i - x_i \beta_0)W_\ell) \right) \xrightarrow{p} 0,
\]

for \( \ell = 1, \ldots, K \). In other words,

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left( \psi((y_i - x_i \hat{\beta})W) - \psi((y_i - x_i \beta_0)W) \right) \xrightarrow{p} 0,
\]

\( \Box \)

**Theorem 1.4.1.** Let the variance-covariance matrix of \( x_{ip} \psi((y_i - x_i \beta_0)W \) be \( \Sigma^*_i \). Assume that there exists \( \Sigma^* \) such that \( \frac{1}{n} \sum_{i=1}^{n} \Sigma^*_i \rightarrow \Sigma^* \) as \( n \rightarrow \infty \). Then \( \frac{1}{\sqrt{n}} \Psi_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \psi((y_i - x_i \hat{\beta})W) \) converges to \( N(0, \Sigma^*) \) in distribution.

**Proof.** Note that \( x_{ip} \psi((y_i - x_i \beta_0)W \) has a mean 0 since we assumed the distribution of \( e_i \) is symmetric about 0 and \( \psi(e_i) \) is also symmetric about 0. The variance of \( x_{ip} \psi((y_i - x_i \beta_0)W \) depends on \( x_{ip} \), so we call it \( \Sigma^*_i \). By therem 1.4.1, we know that

\[
\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \left[ \psi((y_i - x_i \hat{\beta})W) - \psi((y_i - x_i \beta_0)W) \right] \xrightarrow{p} 0.
\]

12
By the generalized multivariate central limit theorem (Serfling, 1980),

$$\frac{1}{n} \sum_{i=1}^{n} x_{ip} \psi_0((y_i - x_i\beta_0)W) \text{ is } AN(0, \frac{1}{n} \Sigma^*),$$

or equivalently,

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \psi_0((y_i - x_i\beta_0)W) \xrightarrow{d} N(0, \Sigma^*).$$

Therefore, by Slutsky’s theorem,

$$\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \psi_0((y_i - x_i\hat{\beta})W) \xrightarrow{d} N(0, \Sigma^*).$$

This completes the proof of Theorem 1.4.1.

We have shown that $\frac{1}{\sqrt{n}} \Psi_n$ is asymptotically normal with mean 0 and the variance-covariance matrix $\Sigma^*$. Next we construct a Chi-square statistic by estimating $\Sigma^*$. Note that we have given the score statistic under the working assumption of independence. However, the correlations among the components of $e_i$ certainly exist, and they need to be considered in the inference. Note that

$$\Sigma^* = \text{Var} \left( \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{ip} \psi_0((y_i - x_i\beta)W) \right) = \frac{1}{n} \sum_{i=1}^{n} x_{ip}^2 \text{Var} \left( \psi_0((y_i - x_i\beta)W) \right).$$

By letting $u_j = \psi_0((y_j - x_j\beta)W)$ and assuming that $\text{Var}(u_j)$ is constant over $j$, we have

$$\hat{\text{Var}}(u_j) = \frac{1}{n-1} \sum_{j=1}^{n} (\hat{u}_j - \bar{\hat{u}})(\hat{u}_j - \bar{\hat{u}}),$$

which leads to

$$\hat{\Sigma}^* = \frac{1}{n(n-1)} \left( \sum_{i=1}^{n} x_{ip}^2 \right) \left( \sum_{j=1}^{n} (u_j - \bar{u})(u_j - \bar{u}) \right).$$

Accordingly, the test statistic $R_H = n^{-1} \Psi_n \hat{\Sigma}^{*-1} \Psi_n'$ is asymptotically $\chi^2$ distributed with $K$ degrees of freedom, where $K$ represents the dimension of $\Psi_n$.

### 1.5 Simulation study

We conducted a simulation study to assess the performance of the HS test for smooth functions of $\beta(t)$. Two hundred random samples were generated from a distribution characterized by the coefficient matrix $\beta$ and
the variance-covariance matrix $\Sigma$. More precisely, we generated the response by

$$y_i = [1, x_{i1}, x_{i2}, x_{i3}] \beta + e_i,$$

(1.19)

where $e_i$ is drawn from the multivariate t distribution with 3 degrees of freedom and covariance matrix $\Sigma = 2I_{10} I'_{10} + 3I_{10}$ for $i = 1, \cdots, n$, $t = 3, \cdots, 30$, where $I$ and $I$ are the vector of ones and the identity matrix, respectively.

The first 3 independent variables $x_i$, $i = 1, 2, 3$ are the true covariates and generated as follows: $x_1$ is sampled uniformly from $[0.1, 2.1]$; $x_2$, conditioning on $x_1$, is gaussian with mean 0 and variance $(1 + x_1)/(2 + x_1)$; $x_3$, independent of $x_1$ and $x_2$, is a Bernoulli random variable with success rate 0.6. In addition to $x_1$, $x_2$, $x_3$, 20 non-significant variables $x_4$, $x_5$, $\cdots$, $x_{23}$ are simulated to demonstrate the performance of the variable selection, where each $x_p$, independent of each other, is a random realization of Gaussian distribution with mean 0 and variance 4. The functions of coefficients are given by

$$\beta_0(t) = 15 + 2 \sin(\pi t/6),$$

$$\beta_1(t) = 2.2 + 3 \cos(\pi (t - 25)/150)(t - 1),$$

$$\beta_2(t) = 6 + \frac{2(t+1)^2}{10^2} + 20,$$

$$\beta_3(t) = 4 + \frac{20-t^2}{50} + 20,$$

(1.20)

where $t$ was set to be the sequence of numbers between 0 to 30 with length 10. The remaining coefficients, corresponding to the other variables, are given by $\beta_p(t) = 0$ for $p = 3, 4, \cdots, 23$.

For the spline basis expansion of the regression coefficients, we computed a basis matrix $F$ from cubic B-splines with $df=5$, which corresponds to placing two knots and found $H'$ from singular value decomposition such that $F_{5\times10} H'_{10\times 5} = I_{5\times 5}$. We examined the residuals from the robust regression with Huber’s M-estimators for $c = 0.01$ which minimizes (1.12). We downweighted the smallest and the largest 10% of the residuals by choosing $c$ to be $c = \max(|r_{0.01}^0|, |r_{0.9}^0|)$, where $r_{0.01}^0$ is the 0.01th percentile of the residuals from the robust regression with $c_0 = 0.01$. Again, we extract $\sigma$ from the results of rlm with this choice of $c$. Figure 1.1(a) shows the normal Q-Q plot of the residuals from the robust regression using Huber’s loss with $c = 0.01$ for one of the 200 generated datasets. They are fairly far from the normal Q-Q line, which strongly suggests that the data are not normally distributed. The type I error estimate for $n = 100$ estimated from 200 datasets with the nominal level of significance at 0.05 is 0.049, which suggests the test is valid even when $n$ is less than four times the number of covariates.

We compared the performance of the HS test with the LS test through the power estimation. The powers
are computed by averaging the percentage of rejections of 200 datasets generated from the model in (1.19) with the regression coefficients in (1.20). The powers from the HS test grew a lot faster to 1 than those from the LS test. See Table 1.1 and Figure 1.1(b) for comparison. The HS test makes use of the functional structure of the regression coefficients by representing them as sum of basis functions, and also downweight outliers with respect to the data by using Huber’s loss function in the score statistic. Therefore, the HS test outperforms the LS test computed based on the normal distribution when the data are not from normal.

Table 1.1: The power and the type I error estimates for the HS test and the LS test.

<table>
<thead>
<tr>
<th></th>
<th>n=60</th>
<th>n=70</th>
<th>n=80</th>
<th>n=90</th>
<th>n=100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (HS)</td>
<td>0.8150</td>
<td>0.9033</td>
<td>0.9683</td>
<td>0.9966</td>
<td>1</td>
</tr>
<tr>
<td>Power (LS)</td>
<td>0.6416</td>
<td>0.7466</td>
<td>0.9133</td>
<td>0.9900</td>
<td>0.9983</td>
</tr>
<tr>
<td>Type I (HS)</td>
<td>0.0363</td>
<td>0.0388</td>
<td>0.0465</td>
<td>0.0415</td>
<td>0.049</td>
</tr>
<tr>
<td>Type I (LS)</td>
<td>0.0415</td>
<td>0.0370</td>
<td>0.0388</td>
<td>0.0398</td>
<td>0.0543</td>
</tr>
</tbody>
</table>

1.6 Application to the yeast cell-cycle microarray data

In this section, we analyze the relationship between a microarray data and transcription factor binding scores of *Saccharomyces Cerevisiae* (baker’s yeast) genes. The response variable is the cell cycle data for 5668 genes in Orlando et al. (2008), which can be accessed at NCBI under the accession number GSE8799. The data contain fifteen samples taken at sixteen minute intervals covering 2 cell cycles in wild-type. For covariates, we use the motif score data obtained from the lab of Beer and Tavazoie (Beer and Tavazoie, 2004), which contains the gene-specific motif score data for 2587 genes. The motifs include 51 known motifs and 615 unknown motifs by the time Beer and Tavazoie (2004) was published. We used the first 51 motifs that are already known and experimentally documented transcription factor binding sites (Hughes et al.,2000; Lee et al.,2002) for our covariates for the analysis. The names of the transcription factors and their numbers we used in the analysis are given in Table 1.2. The goal is to find out which among the 51 motifs have statistically significant effects on the gene expressions. In the following sections, we illustrate the difference between the HS tests and the LS tests by testing each motif conditionally on other motifs for the significance on the gene expressions, separately.

1.6.1 Backward selection

There are 2499 genes to be analyzed that are present both in expression data and in motif scores. To incorporate the functional structure of the regression coefficients, we used the cubic B-spline bases with 7 degrees of freedom. Figure 1.2(a) and Figure 1.2(b) show the normal Q-Q plot and the histogram of the
Table 1.2: The 51 motifs that are already known and experimentally documented, numbered them from 1 to 51.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABF1</td>
<td>ACE2</td>
<td>AFT1</td>
<td>ARG80</td>
<td>AT-repeat</td>
<td>BAS1</td>
<td>CAD1</td>
<td>CBF1</td>
<td>CIN5</td>
<td>CSRE</td>
</tr>
<tr>
<td>11</td>
<td>DAL81</td>
<td>FKH1</td>
<td>GAL4</td>
<td>GCN4</td>
<td>Gcr1</td>
<td>Gis1</td>
<td>HAP4</td>
<td>HIR2</td>
<td>HSF1</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
</tr>
<tr>
<td>LEU3</td>
<td>LYS14</td>
<td>MAC1</td>
<td>MBP1</td>
<td>MCM1</td>
<td>MET31</td>
<td>MET4</td>
<td>MIG1</td>
<td>MSN24</td>
<td>MSN24a</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>ndt80</td>
<td>NRG1</td>
<td>OAF1</td>
<td>PAC34</td>
<td>PDR3</td>
<td>PHO4</td>
<td>RAP1</td>
<td>REB1</td>
<td>RFX1</td>
<td>RPN4</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
<td>41</td>
</tr>
<tr>
<td>RRPE</td>
<td>SKN7</td>
<td>STE12</td>
<td>STRE</td>
<td>SUM1</td>
<td>SWI4</td>
<td>Ume6</td>
<td>YAP1</td>
<td>Zap1</td>
<td>TATA</td>
</tr>
<tr>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td>XBP1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

residuals from the robust regression with Huber’s loss function at $c = 0.01$. The Kolmogorov-Smirnov test for normal distribution with mean and variance estimated from the residuals gave p-value near 0.

Among many approaches for the variable selection, we choose to use a backward elimination to illustrate difference between the HS and the LS test in finding a subset of important motifs. We started from the full model with 51 motifs and removed at each time the motif with the largest P-value from the tests. The motifs were deleted one by one until all of the motifs in the model are significant under the FDR control at 0.05. We chose $c$ in the HS test to downweight up to 15% of the positive residuals and up to 15% of the negative residuals for the HS backward selection. Since the residuals in this example are skewed to the right, the actual percentage of downweighting in both tails is around 17%. We denote the HS backward selection with $q\%$ of downweighting each side of the residuals as the HS(2$q\%$).

Table 1.3 shows the final sets of motifs selected by the HS(30%) and the LS backward elimination, respectively. The HS(30%) selected 27 motifs, while the LS backward elimination selected 25 motifs as shown on the table. The results from the HS(10 %) and the HS(20%) will be added in Section 1.6.2.

The motifs selected by the HS(30%) only are MIG1, ARO80, STE12, OAF1, MET31 and NRG1, and those selected by the LS only are Gcr1, LEU3 and SKN7, ordered from the more significant to the less significant. We now take a closer look at why the two tests selected different sets of motifs. The LS tests are influenced by outlying genes. The motifs selected only by the HS(30%) reflect the influence on the majority of the genes. On the other hand, the motifs selected only by the LS tests could be driven by a small number
Table 1.3: The Motifs and their p-values in parentheses selected from the HS and the LS backward selection for the cell-cycle data, ordered from the most significant to the least significant. The motifs without p-values indicate their p-values are less than 0.0001.

<table>
<thead>
<tr>
<th>Selected</th>
<th>Motifs (P-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HS(30%)</td>
<td>MSN24a, PAC, RAP1, RPN4, FKH1, SWI4, MBP1, MCM1, Gis1, RRPE, ABF1, YAP1, REB1, MIG1(0.0001), XBP1(0.0002), ACE2(0.0002), HAP4(0.0003), TATA(0.0010), HSF1(0.0015), ARO80(0.0045), GAL4(0.0055), Ume6(0.0089), STE12(0.0103), OAF1(0.0111), MET31(0.0129), PHO4(0.0281), NRG1(0.0406)</td>
</tr>
<tr>
<td>LS</td>
<td>MBP1, MSN24a, PAC, RAP1, SWI4, RRPE, FKH1, HAP4, RPN4, ACE2, ABF1, Gis1, Gcr1, MCM1, LEU3(0.0002), TATA(0.0037), HSF1(0.0046), BAS1(0.0057), REB1(0.0058), YAP1(0.0074), GAL4(0.0113), XBP1(0.0141), SKN7(0.0203), Ume6(0.0292), PHO4(0.0441)</td>
</tr>
</tbody>
</table>

of genes. As an example, we detail the analysis for motif MIG1 and motif Gcr1. To further understand the differences, recall that

\[
\Psi_{HS} = \sum_{i=1}^{n} x_{ip} \psi(r_{i}^{HS}) \quad \text{and} \quad \Psi_{LS} = \sum_{i=1}^{n} x_{ip} r_{i}^{LS}
\]

(1.21)

where \(r_{i}^{HS}\) and \(r_{i}^{LS}\) are the scaled residuals from the robust regression and the least squares regression, respectively. It is interesting to see the scatterplots of \(\psi(r_{i}^{HS})\) and \(r_{i}^{LS}\) plotted against \(x_{ip}\). Since the scores for each test are the product of the residuals and \(x_{ip}\), the value of the score will be determined by the relationship between these residuals and \(x_{ip}\). Figures 1.3 and 1.4 show the componentwise residuals \(r_{i}^{LS} = y_{i} - x_{i} \hat{\beta}_{LS}\) (by circles) where \(\hat{\beta}_{LS}\) represents the least squares estimate of \(\beta\), and \(\psi(r_{i}^{R}) = \psi(y_{i} - x_{i} \hat{\beta}_{R})\) (by triangles), for motif MIG1 and Gcr1, respectively, where \(\hat{\beta}_{R}\) represents the M-estimate of \(\beta\) using Huber’s loss function with \(c\) chosen to downweight 30% of the residuals.

We notice that there are a number of outlying LS residuals with nonzero motif scores. The genes corresponding to those outlying LS residuals for motifs MIG1, Gcr1 and MSN24a have remarkably different characteristics. The sample means at each time point of the gene expressions of the outlying genes are about 5 times bigger than those of the remaining genes. The motifs scores for the outlying genes are about twice as much as those for non-outlying genes at their grand means. The genes with high gene expressions are often considered important. Although they are outlying in the data, it is interesting to know which motifs are more related to those genes or to the other genes with lower gene expressions.

The outlying genes can inflate \(\Psi_{LS}\), as well as the variance of the LS score. For motif MIG1, the outlying
residuals inflate the variance more to drive the LS test statistic closer to 0. To verify, we find out what would happen if we remove outlying genes. Among genes with nonzero motifs scores, we removed those with scaled residuals outside (-2, 2). The reason we didn’t examine the genes with zero motif scores is that the scores wouldn’t be influenced by those genes. As a result, 35 genes were removed for motif MIG1, and 40 genes for motif Gcr1. We performed the LS test again with the outliers removed from the data. As a consequence, the LS test after removing outliers led to significant result for MIG1 with the p-value 0.0002. This suggests that MIG1 is playing an important role in controlling the expression levels for majority of the genes. And motif Gcr1, which was originally declared significant in the LS test, was excluded from the group of significant motifs after removing outliers from the data. The p-value for Gcr1 was 0.0637. This implies that Gcr1 was mostly related to outlying genes, not the majority of the genes. On the other hand, the results from the HS(30%) test led to the same conclusion, which verifies its robustness against outliers.

Motif MSN24a was selected by both tests. It was the most significant motif among motifs selected by HS(30%)backward selection, and the second most significant among those selected by LS backward selection. Figure 1.5 shows the LS and HS(30%) residuals for MSN24a, from which we can notice outlying genes. We deleted genes with nonzero motif scores whose residuals are outside (-2 and 2), which counts 114 genes out of 2499. Both HS(30%) and LS tests after removing these outliers still resulted in small p-values almost 0. This suggests that the outlying genes didn’t inflate the variance as much for MSN24a to come out insignificant.

Table 1.4: The significant of the motifs and their p-values in parenthesis before and after removing outliers.

<table>
<thead>
<tr>
<th>Tests for motifs</th>
<th>Presence of outliers</th>
<th>HS(30%) test</th>
<th>LS test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motifs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIG1</td>
<td>Yes</td>
<td>significant (0.0001)</td>
<td>insignificant (26th deleted, 0.063)</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>significant (&lt; 0.0000)</td>
<td>significant (0.0002)</td>
</tr>
<tr>
<td>Gcr1</td>
<td>Yes</td>
<td>insignificant (8th deleted, 0.383)</td>
<td>significant (&lt;0.0000)</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>insignificant (0.1519)</td>
<td>insignificant (0.0637)</td>
</tr>
<tr>
<td>MSN24a</td>
<td>Yes</td>
<td>significant (&lt; 0.0000)</td>
<td>significant (&lt;0.0000)</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>significant (&lt; 0.0000)</td>
<td>significant (&lt;0.0000)</td>
</tr>
</tbody>
</table>

1.6.2 Using HS tests and LS tests together

One interesting discovery in this analysis is that the HS test and the LS test can be used to distinguish motifs using the following four scenarios. If a motif is significant both in the HS test and the LS test, then we keep the results. If it is significant neither in the HS test nor in the LS test, then we have no evidence to claim its significance to any gene used in the analysis. If the motif is significant in the HS test, but not significant in the LS test, then we say it is significant to a majority of genes. Lastly, if the motif is not significant in the HS test, but significant in the LS test, then there are two possibilities - either the HS test
is not sufficiently powerful or the significance of the LS test is driven by a few outliers.

Since the optimal choice of $c$ for the HS test is unknown, we report additional HS tests with different percentages of downweighting. The Venn-diagram in 1.6 shows the all motifs partitioned by HS(10%), HS(20%), HS(30%) and LS tests. The HS(10%) selected 47 motifs while HS(20%) selected 37 motifs. Note that for the yeast cell-cycle gene expression data, motifs 6, 15 and 42 (BAS1, Gcr1 and SKN7) are selected by the LS tests, but not by the HS tests(20%). Although the HS test missed these four motifs, it could be due to the choice of $c$ in the HS tests. Downweighting too much could hurt the power of the HS test.

1.7 Conclusion

In this chapter, we developed a robust score test for linear models by a modification of the well-known Rao’s score test based on Huber’s M-estimator. We proved that the test statistic is asymptotically normal and showed from a simulation study that the test has higher power than the score test based on the least squares, in presence of outliers. In the application to the *Saccharomyces Cerevisiae* data, we have shown that the HS test can add more insights and caution to the LS test. Using both the HS test and the LS test provides a better and safer option for identifying important variables when departures from the normality assumption are a concern.
1.8 Figures of Chapter 1

Figure 1.1: The normal Q-Q plot of the residuals from the robust linear regression using Huber’s loss function with $c = 0.01$ for one of the generated datasets (a) and estimated powers of the HS test and the LS test (b) in Section 1.5.

Figure 1.2: The normal Q-Q plot (a) and the histogram (b) of the residuals from the robust regression with Huber’s loss function at $c = 0.01$ for the cell cycle data in Section 1.6.
Figure 1.3: The plots of the HS(30%) and LS residuals for motif MIG1.
Figure 1.4: The plots of the HS(30%) and LS residuals for motif Gcr1.
Figure 1.5: The plots of the HS(30%) and LS residuals for motif MSN24a.
Figure 1.6: Venn diagram of motifs numbered from 1 to 51 selected by HS(10%), HS(20%), HS(30%) and the LS backward selection.
Chapter 2

Robust EM Clustering via Mixture Model for Multivariate Regression

2.1 Introduction

Clustering is one of the fundamental data mining techniques. The general purpose is to detect clusters in observations and to assign individual observations to the clusters. Groups of observations that are similar to one another form clusters. Among a wide variety of clustering techniques, mixture models are commonly used because of their usefulness as a flexible statistical option to model heterogeneous data. They provide a soft clustering of the data based on the fitted posterior probability of belonging to each mixture component. For multivariate data, attention has been paid to the use of multivariate normal components because of their computational convenience. They can be fitted by maximum likelihood (ML) via the expectation-maximization (EM) algorithm (Dempster et al., 1977; Yuille et al., 1994; Xu and Jordan, 1995; Bradley et al., 1999; Roweis and Ghahramani, 1999; Ordonez and Cereghini, 2000), as the iterations in the M-step are given in closed form.

However, for many applied problems, the tails of the normal distribution are often shorter than required. Also, the estimates of the component means and covariance matrices can be affected by observations that are atypical of the components in the normal mixture model being fitted (McLachlan and Peel, 2000). Hence, it is desirable for clustering methods to be robust. By robustness, we mean that the method is not affected significantly by small departures from the assumed model, such as the presence of outliers.

The problem of providing protection against outliers in multivariate data is a very difficult one and the difficulty increases with the dimension of the data (Rocke and Woodruff, 1996). A common way in which robust fitting of normal mixture models has been undertaken, is by using M-estimates to update the component estimates on the M-step of the EM algorithm, as in McLachlan and Basford (1988) and Campbell (1984). EM clustering for the ML estimation of mixtures of multivariate t-distributions proposed in McLachlan and Peel (1998) can also provide a robust approach to clustering. The use of t-components in the mixture model provides less extreme estimates of the posterior probabilities of cluster membership.

Here we consider the fitting of regression mixtures of modified multivariate normal components, designed
to downweight extreme outliers by Huber’s loss function. The threshold value in Huber’s loss function plays a role as a robustness tuning parameter. The parameters in each component can be estimated through the EM algorithm, with a direct optimization (REM 1) or the iteratively reweighted least squares estimation (REM 2) in the M-step. This provides a more robust approach to the fitting of normal mixture models, as observations that are atypical of a component are given reduced weight in the calculation of its parameters. It is distinguished from other existing methods in that it is based on the regression mixture that can borrow cluster information from covariates. It can also take advantage of the functional structure of the response, if any, by transforming the response and regression coefficients using splines as described in Chapter 1.

We start the chapter by introducing the M-component mixture of multivariate Gaussian regression model, and the associated EM algorithm for carrying out the maximum likelihood estimation and clustering. In the later sections, we extend it to the mixtures of modified multivariate normal distributions. The performances of the proposed robust clustering methods are assessed through simulations with various comparisons with other existing methods.

2.2 The EM algorithm for Gaussian mixture multivariate regression model

The goal of the EM clustering is to estimate the parameters for each cluster so as to maximize the likelihood of the observed data. The E-step of the EM algorithm assigns ‘posterior probabilities’ for each data point based on its relative density under each mixture component, while the M-step recomputes the component density parameters based on the current probabilities.

Suppose we have \( n \) observations of the \( d \)-dimensional multivariate responses that we wish to cluster, and \( n \) observations of \( p \) covariates that we wish to use for clustering. Let \( Y_{n \times d} \) and \( X_{n \times p} \) be the response and design matrix, respectively. Consider a Gaussian mixture regression model for multivariate normal with \( M \) components. Let \( \beta_m \) be the \( p \times d \) matrix of regression coefficients, and \( \Sigma_m \) be the \( d \times d \) variance-covariance matrix of the \( m^{th} \) components. We define \( \theta = (\theta_1, \cdots, \theta_M) \), where the parameter for the \( m \)th component distribution is defined by \( \theta_m = (\beta_m, \Sigma_m) \) for \( m = 1, \cdots, M \).

Let \( y_i \) be the \( i^{th} \) row vector of \( Y \) and \( x_i \) be the \( i^{th} \) row vector of \( X \), for \( i = 1, \cdots, n \). The density of the sampling distribution of the response at \( y_i \) can be written as

\[
f(y_i) = \sum_{m=1}^{M} \lambda_m N(x_i \beta_m, \Sigma_m) \quad \text{for } i = 1, 2, \cdots, n
\]
and then the log likelihood for \( y = (y'_1, \cdots, y'_n)' \) is given by

\[
\log(f(y)) = \log \left( \prod_{i=1}^{n} \sum_{m=1}^{M} \lambda_m N(x_i \beta_m, \Sigma_m) \right) = \sum_{i=1}^{n} \log \left( \sum_{m=1}^{M} \lambda_m N(x_i \beta_m, \Sigma_m) \right).
\]

Because a direct maximization of the log likelihood is quite difficult numerically due to the sum of terms inside the logarithm, we consider the unobserved latent vector \( \zeta_i = (\zeta_{i1}, \zeta_{i2}, \cdots, \zeta_{iM}) \) with exactly one of \( \zeta_{im} \) equal to 1 for each \( y_i \). Let \( \lambda = (\lambda_1, \lambda_2, \cdots, \lambda_M) \) be the vector of the mixture proportions and \( \theta = (\theta_1, \theta_2, \cdots, \theta_M) \) be the collection of parameter vectors for all mixture components. The probability of \( y \) given \( \zeta \) and \( \theta \) is given by

\[
P(y|\zeta, \theta) = P(y_1, y_2, \cdots, y_n|\zeta_1, \zeta_2, \cdots, \zeta_n, \theta) = \prod_{i=1}^{n} p(y_i|\zeta_i, \theta)
= \prod_{m=1}^{M} f(y_1|\theta_m)^{\zeta_{i1}} \cdots \prod_{m=1}^{M} f(y_n|\theta_m)^{\zeta_{in}}
= \prod_{i=1}^{n} \prod_{m=1}^{M} f(y_i|\theta_m)^{\zeta_{im}}.
\]

The complete data likelihood is

\[
P(y, \zeta|\theta, \lambda) = P(\zeta|\lambda)P(y|\zeta, \theta) = \prod_{i=1}^{n} \prod_{m=1}^{M} \lambda_{im} \prod_{i=1}^{n} \prod_{m=1}^{M} f(y_i|\theta_m)^{\zeta_{im}} = \prod_{i=1}^{n} \prod_{m=1}^{M} [\lambda_m f(y_i|\theta_m)]^{\zeta_{im}}.
\]

Therefore, the complete data log likelihood is

\[
\log P(y, \zeta|\theta, \lambda) = \sum_{i=1}^{n} \sum_{m=1}^{M} \zeta_{im} (\log \lambda_m + \log f(y_i|\theta_m)).
\]

Let \( P(\zeta_{im} = 1|\theta^{old}, \lambda^{old}, y) = z_{im} \) where \( \theta^{old} \) and \( \lambda^{old} \) are the parameters and mixture proportions from the previous step, respectively. Then \( \zeta_{im}^{old} \), \( \lambda^{old}, y \) follows the Bernoulli distribution with \( z_{im} \) as the probability of being 1, which leads to \( E(\zeta_{im}|\theta^{old}, \lambda^{old}, y) = z_{im} \). Since

\[
P(\zeta|\theta^{old}, \lambda^{old}, y) \propto P(\zeta, y|\theta^{old}, \lambda^{old}),
\]

\[
P(\zeta_{im} = 1, y_i|\theta^{old}, \lambda^{old}) = P(\zeta_{im} = 1|\theta^{old}, \lambda^{old}) P(y_i|\theta^{old}, \lambda_{im}) = \lambda_{im} f_m(y_i|\theta_{im}),
\]

27
we obtain
\[ z_{im} = \frac{\lambda^\text{old}_m f_m(y_i|\theta^\text{old}_m)}{\sum_{m=1}^{M} \lambda^\text{old}_m f_m(y_i|\theta^\text{old}_m)}. \] (2.1)

### 2.2.1 The E-step

The expected complete data log likelihood is
\[ E^\text{old}_\zeta \log P(y, \zeta|\theta, \lambda) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left( \log \lambda_m + \log f_m(y_i|\theta_m) \right) \]

where
\[ f_m(y_i|\theta_m) = \frac{1}{(2\pi)^{T/2} |\Sigma_m|^{1/2}} \exp \left\{ -\frac{1}{2} \left( y_i - x_i\beta_m \right) \Sigma_m^{-1} \left( y_i - x_i\beta_m \right)' \right\}, \]

and \( z_{im} \) given in (2.1), is commonly called the responsibility, or the probability that \( i \)th observation is in the \( m \)th mixture component. We compute these responsibilities in the E-step and plug them into the parameter estimates in the following M-step.

### 2.2.2 The M-step

In the M-step, we must find \( \theta \) and \( \lambda \) to maximize \( E^\text{old}_\zeta \log P(y, \zeta|\theta, \lambda) \). We use the Lagrange multiplier \( L \) with the constraint \( \sum_{m=1}^{M} \lambda_m = 1 \). Let
\[ l(\theta) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left( \log \lambda_m + \log \frac{1}{(2\pi)^{T/2} |\Sigma_m|^{1/2}} \exp \left\{ -\frac{1}{2} \left( y_i - x_i\beta_m \right) \Sigma_m^{-1} \left( y_i - x_i\beta_m \right)' \right\} \right) - L \cdot \left( \sum_{m=1}^{M} \lambda_m - 1 \right). \]

The maximizing steps are shown below.

1. Update \( \lambda = (\lambda_1, \lambda_2, \cdots, \lambda_M) \):
\[ \frac{\partial}{\partial \lambda_m} l(\theta) = \sum_{i=1}^{n} \frac{z_{im}}{\lambda_m} - L = 0 \Rightarrow \hat{\lambda}_m = \frac{1}{L} \sum_{i=1}^{n} z_{im}. \]

Note that \( \sum_{m=1}^{M} \hat{\lambda}_m = 1 \) and \( \sum_{m=1}^{M} z_{im} = 1 \). This implies that \( L = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} = n \), and therefore the estimator for \( \lambda_m \) that maximizes \( l(\theta) \) is
\[ \lambda^\text{new}_m = \frac{\sum_{i=1}^{n} z_{im}}{\sum_{i=1}^{n} \sum_{m=1}^{M} z_{im}} = \frac{\sum_{i=1}^{n} z_{im}}{n} \]

for each \( m = 1, 2, \cdots, M \).
2. Update $\beta = (\beta_1, \beta_2, \ldots, \beta_M)$ with $\Sigma_m$ fixed at the current step:

$$
\frac{\partial}{\partial \beta_m} \ l(\theta) = \sum_{i=1}^{n} z_{im} \frac{\partial}{\partial \beta_m} \left\{ -\frac{1}{2} (y_i - x_i \beta_m) \Sigma_m^{-1} (y_i - x_i \beta_m)' \right\} 
$$

$$
= -\frac{1}{2} \sum_{i=1}^{n} z_{im} \frac{\partial}{\partial \beta_m} \left\{ y_i \Sigma_m^{-1} y_i' - y_i \Sigma_m^{-1} \beta_m' x_i' - x_i \beta_m \Sigma_m^{-1} y_i' + x_i \beta_m \Sigma_m^{-1} \beta_m' x_i' \right\} 
$$

Since

$$
\frac{\partial}{\partial \beta_m} y_i \Sigma_m^{-1} \beta_m' x_i' = x_i' y_i \Sigma_m^{-1}
$$
$$
\frac{\partial}{\partial \beta_m} x_i \beta_m \Sigma_m^{-1} y_i' = x_i' y_i \Sigma_m^{-1}
$$
$$
\frac{\partial}{\partial \beta_m} x_i \beta_m \Sigma_m^{-1} \beta_m' x_i' = 2 x_i' \beta_m x_i' \Sigma_m^{-1},
$$

we get

$$
\frac{\partial}{\partial \beta_m} l(\theta) = \sum_{i=1}^{n} z_{im} x_i' (y_i - x_i \beta_m) \Sigma_m^{-1} = 0.
$$

Solving for $\beta_m$ gives

$$
\beta_m^{new} = \left( \sum_{i=1}^{n} z_{im} x_i' x_i \right)^{-1} \left( \sum_{i=1}^{n} z_{im} x_i' y_i \right).
$$

3. Update $\Sigma = (\Sigma_1, \Sigma_2, \ldots, \Sigma_M)$ with $\beta_m$ fixed at the current step:

Let $\Omega_m = \Sigma_m^{-1}$. Then

$$
\frac{\partial}{\partial \Omega_m} l(\theta) = \sum_{i=1}^{n} \frac{1}{2} z_{im} \frac{\partial}{\partial \Omega_m} \left\{ \log |\Omega_m| - (y_i - x_i \beta_m) \Omega_m (y_i - x_i \beta_m)' \right\}
$$

Since

$$
\frac{\partial}{\partial \Omega_m} \log |\Omega_m| = \Sigma_m
$$
$$
\frac{\partial}{\partial \Omega_m} (y_i - x_i \beta_m) \Omega_m (y_i - x_i \beta_m)' = (y_i - x_i \beta_m)' (y_i - x_i \beta_m),
$$

we get

$$
\frac{\partial}{\partial \Omega_m} l(\theta) = \frac{1}{2} \sum_{i=1}^{n} z_{im} \left\{ \Sigma_m - (y_i - x_i \beta_m)' (y_i - x_i \beta_m) \right\} = 0
$$
Therefore, solving for \( \Sigma_m \) gives

\[
\Sigma_{m}^{\text{new}} = \frac{\sum_{i=1}^{n} z_{im}(y_i - x_i \beta_m)'(y_i - x_i \beta_m)}{\sum_{i=1}^{n} z_{im}}.
\]

The matrix calculus involved in the M-step is based on the differentials of trace of matrices in Petersen and Pedersen (2006).

### 2.2.3 Getting the cluster indices from the responsibilities

We can usually review an actual assignment of observations to clusters, based on the largest responsibility. For the \( i \)th observation, we have \([z_{i1}, \cdots, z_{iM}]\), the probability of \( i \)th observation being in cluster 1, \( \cdots \), \( M \), respectively. We assign the \( i \)th observation to cluster \( K \) if \( z_{ik} = \max\{z_{im}\}_{m=1}^{M} \).

### 2.2.4 Example

A data set with 300 observations was generated to have 3 clusters: \( N(X\beta, \Sigma_m), m = 1, 2, 3 \) on R version 2.6.2. The response is 10-dimensional, and the number of covariates is set to 4. Each covariate is generated independently from the Uniform distribution on \((1,4)\) on seeds 1 to 4. The \( 5 \times 10 \) matrices of the regression coefficients \( \beta_m \)'s are created. Let \( \beta^m_k = (\beta^m_k(t_1), \cdots, \beta^m_k(t_{10})) \) be the \( k \)th row of \( \beta_m \) of length 10 for \( k = 0, 1, \cdots, 4 \), where \( k = 0 \) represents the row for the intercept. The distributions of the 3 mixture components are given as follows:

**Component 1:** \( Y_i \sim N(X_i \beta_1, \Sigma_1) \), where \( \beta_1 \) is a 5 by 10 matrix of coefficients including the row for the intercept, the rows of which were generated by

\[
\begin{align*}
\beta_0^1(t) &= 1 - 1.7 \sin(2\pi t) \\
\beta_1^1(t) &= 0.34 - 1.7 \sin(\frac{3}{2}\pi t) \\
\beta_2^1(t) &= 0.5 - 1.5 \sin(0.5\pi(t - 1.2)) \\
\beta_3^1(t) &= 0.3 - 2 \cos(2\pi t) \\
\beta_4^1(t) &= |t - 0.5|,
\end{align*}
\]

for \( t = 3, 6, \cdots, 30 \), and \( \Sigma_1 \) is a \( 10 \times 10 \) variance-covariance matrix with the compound symmetry structure, with the diagonal elements equal to 2 and the off-diagonal elements 1.

**Component 2:** \( Y_i \sim N(X_i \beta_2, \Sigma_2) \), where \( \beta_2 \) is a 5 by 10 matrix of coefficients, rows of which were generated
by

\[
\begin{align*}
\beta_0^2(t) &= -1.2 + 1.3 \cos(\pi(t + 1.1)) \\
\beta_1^2(t) &= 0.22 + 0.2 \cos(2\pi t) - 1 \\
\beta_2^2(t) &= 1.2 - 1.5 \sin(2.3\pi t) \\
\beta_3^2(t) &= 0.5 - 1.4 \sin(2\pi t) \\
\beta_4^2(t) &= -0.32 + 1.17 \sin((2/3)\pi t),
\end{align*}
\]

and \( \Sigma_2 \) is also compound symmetric, with diagonal elements equal to 3, and the off-diagonal elements 1.

Component 3: \( Y_i \sim N(X_i\beta_3, \Sigma_3) \), where \( \beta_3 \) is a 5 by 10 matrix of coefficients, rows of which were generated by

\[
\begin{align*}
\beta_0^3(t) &= -0.15 + 1.42 \cos(1.5\pi t) \\
\beta_1^3(t) &= t^2 + 3t - 1.2 \\
\beta_2^3(t) &= 1.3 - 1.37 \sin(2\pi t) \\
\beta_3^3(t) &= 1.34 \cos(2\pi t) - 2 \\
\beta_4^3(t) &= 0.34 - 1.7|\sin((2/3)\pi t)|,
\end{align*}
\]

and \( \Sigma_3 \) is also compound symmetric, with diagonal elements equal to 6, and the off-diagonal elements 1.

Each row of \( Y \) was then generated by random draws with equal probabilities from one of the mixture component distributions. Figure 2.1 is the plot of \( Y \) produced by 3-dimensional multi-dimensional scaling.

In order to compare clustering results, a measure of agreement is needed. Given a set of \( n \) objects \( S = \{O_1, \cdots, O_n\} \), suppose \( U \) and \( V \) represent two different partitions of the objects in \( S \). Suppose \( U \) is an external criterion and \( V \) is the clustering result. Let \( n_{ij} \) be the number of objects that are in both class \( u_i \) and cluster \( v_j \). Let \( n_i \) and \( n_j \) be the number of objects in class \( u_i \) and and cluster \( v_j \), respectively. The notations are illustrated in Table 2.1.

Table 2.1: Notation for the contingency table for comparing two partitions.

<table>
<thead>
<tr>
<th>Class \ Cluster</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>\cdots</th>
<th>( v_C )</th>
<th>Sums</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>( n_{11} )</td>
<td>( n_{12} )</td>
<td>\cdots</td>
<td>( n_{1C} )</td>
<td>( n_1 )</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>( n_{21} )</td>
<td>( n_{22} )</td>
<td>\cdots</td>
<td>( n_{2C} )</td>
<td>( n_2 )</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\cdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>( u_R )</td>
<td>( n_{R1} )</td>
<td>( n_{R2} )</td>
<td>\cdots</td>
<td>( n_{RC} )</td>
<td>( n_R )</td>
</tr>
<tr>
<td><strong>Sums</strong></td>
<td>( n_1 )</td>
<td>( n_2 )</td>
<td>\cdots</td>
<td>( n_C )</td>
<td>( n_r = n )</td>
</tr>
</tbody>
</table>
We use the adjusted Rand index proposed by Hubert and Arabie (1985), which is defined by

\[
\frac{\sum_{i,j} \binom{n_{ij}}{2} - \left[ \sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[ \sum_i \binom{n_i}{2} + \sum_j \binom{n_j}{2} \right] - \left[ \sum_i \binom{n_i}{2} \sum_j \binom{n_j}{2} \right] / \binom{n}{2}}.
\]

(2.2)

The adjusted Rand index is the corrected-for-chance version of the Rand index (Rand, 1971). For more details, see the supplement to the paper Yeung and Ruzzo (2000).

We ran the EM algorithm with 10 starting values of the parameters $\beta$, $\Sigma$ and $\lambda$ with 150 iterations. To assess its clustering performance, we computed the Adjusted Rand Index (ARI) between the true and inferred cluster indices. The ARI values from 10 starting values were all around 0.9797, which indicates almost perfect match. The ARI values of the true and inferred indices from Mclust and K-means were around 0.4830 and 0.3855, respectively throughout the 10 runs. The superior performance of the EM clustering based on the Gaussian mixture multivariate regression model comes from the use of covariates. Mclust assumes that the errors are normally distributed and the K-means clustering is also closely related to the EM algorithm for estimating a certain Gaussian mixture model except that EM algorithm makes probabilistic assignments of points to cluster centers while K-means assigns each observation to the closest cluster mean. Covariates add more accuracy to clustering in these cases.

### 2.3 The robust EM clustering through optimization

As pointed out in McLachlan and Peel (2000), although a crude estimate of the within-cluster covariance matrix $\Sigma_i$ often helps clustering (Gnanadesikan, Harvey, and Kettenring, 1993), it can be severely affected by outliers. Hence it is highly desirable to provide robust clustering procedures. In this section, we propose a robust clustering method based on EM algorithm through a multivariate regression model with errors from a modified multivariate normal density. Huber’s loss function $\rho$ in (3.24) is used on the exponent of the density to downweight outliers with respect to the multivariate normal distribution. The proposed clustering method, which we named as REM 1 is distinguished from existing methods in that it makes use of covariates for clustering observations of the response variables. The clustering performance of the REM 1 is compared with the Mclust and the K-means using ARI values in three examples.

#### 2.3.1 The E-step

Consider the Gaussian mixture regression model with $M$ components, with the parameter $\theta_m = (\beta_m, \Sigma_m)$ for the $m$th component distribution. Using Huber’s loss function in the expected complete data log-likelihood,
we get

\[ E_\zeta \log P(y, \zeta | \theta, \lambda) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left( \log \lambda_m + \log f_m(y_i|\theta_m) \right) \]

where

\[ f_m(y_i|\theta_m) = \frac{1}{C^{1/2}} \frac{1}{\sqrt{\Sigma_m}} \exp \left\{ -\frac{1}{2} \rho \left( \left( y_i - x_i \beta_m \right) \Sigma_m^{-1} \left( y_i - x_i \beta_m \right)' \right)^{1/2} \right\} \]

and

\[ z_{im} = \frac{\lambda_m \cdot f_m(y_i|\theta_m)}{\sum_{m=1}^{M} \lambda_m \cdot f_m(y_i|\theta_m)} \]

As in section 2.2.1, \( z_{im} \) is commonly called the responsibility, or the probability that the \( i \)th observation is in the \( m \)th mixture component. We compute these responsibilities in the E-step of the EM algorithm and use them in the following M-step.

### 2.3.2 The M-step

We use the Lagrange multiplier \( L \) with the constraint \( \sum_{m=1}^{M} \lambda_m = 1 \). Let

\[ l(\theta) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left\{ \log \lambda_m + \log \frac{1}{C^{1/2}} \frac{1}{\sqrt{\Sigma_m}} \exp \left\{ -\frac{1}{2} \rho \left( \left( y_i - x_i \beta_m \right) \Sigma_m^{-1} \left( y_i - x_i \beta_m \right)' \right)^{1/2} \right\} \right\} - L \cdot \left( \sum_{m=1}^{M} \lambda_m - 1 \right) \]

We maximize \( l(\theta) \) with respect to the parameters in the following steps:

1. Update \( \lambda = (\lambda_1, \lambda_2, \cdots, \lambda_M) \). The estimation of \( \lambda_m \) remains the same as in section 2.2.2. Since \( \lambda_m \) is not involved in Huber’s loss function, we get the same estimate of \( \lambda_m \) as in 2.2.2:

\[ \lambda_m^{new} = \frac{\sum_{i=1}^{n} z_{im}}{\sum_{i=1}^{n} \sum_{m=1}^{M} z_{im}} = \frac{\sum_{i=1}^{n} z_{im}}{n} \]

for each \( m \in \{1, 2, \cdots, M\} \).

2. Update \( \beta = (\beta_1, \beta_2, \cdots, \beta_M) \) by solving \( \frac{\partial}{\partial \beta_m} l(\theta) = 0 \). For notational convenience, let

\[ r_{im} = \sqrt{(y_{im} - x_{im} \beta_m) \Sigma_m^{-1} (y_{im} - x_{im} \beta_m)'} \]
Then differentiating \( l(\theta) \) with respect to \( \beta_m \) gives

\[
\frac{\partial}{\partial \beta_m} l(\theta) = \sum_{i=1}^{n} z_{im} \left( \frac{1}{2} \rho'(r_{im}) \right) = -\frac{1}{2} \sum_{i=1}^{n} z_{im} \psi'(r_{im}) \frac{1}{2} r_{im}^{-1} (-2)x_i'(yi - x_i \beta_m) \Sigma_m^{-1} = \frac{1}{2} \sum_{i=1}^{n} z_{im} \psi(r_{im}) r_{im}^{-1} x_i'(yi - x_i \beta_m) \Sigma_m^{-1} = 0,
\]

with \( \Sigma_m \) fixed at the current step.

3. Update \( \Sigma = (\Sigma_1, \Sigma_2, \cdots, \Sigma_M) \).

Let \( \Omega_m = \Sigma_m^{-1} \). Then

\[
\frac{\partial}{\partial \Omega_m} l(\theta) = \sum_{i=1}^{n} \frac{1}{2} z_{im} \frac{\partial}{\partial \Omega_m} \left\{ \log |\Omega_m| - \rho(r_{im}) \right\} = \sum_{i=1}^{n} \frac{1}{2} z_{im} \Sigma_m - \sum_{i=1}^{n} \frac{1}{4} z_{im} \psi'(r_{im}) r_{im}^{-1} (yi - x_i \beta_m)' (yi - x_i \beta_m) = 0,
\]

with \( \beta_m \) fixed at the current step.

### 2.3.3 Determining \( c \) in Huber’s loss function

The performance of REM 1 depends on the choice of the threshold value \( c \) in Huber’s loss function. To determine the value of \( c \) in order for the REM 1 to downweight a certain desired amount of outliers, we first set the threshold to a small value and run the algorithm with the initial parameters estimated from clustering via Mclust. After we obtain the initial results from the REM 1, we compute the following quantities for estimated clusters:

\[
r_{im} = \sqrt{(yi - x_i \beta_m) \Sigma_m^{-1} (yi - x_i \beta_m)^t},
\]

for \( m = 1, \cdots, M \), where \( y_m \) and \( x_m \) are subsets of \( Y \) and \( X \) belonging to the \( m^{th} \) cluster for \( m = 1, \cdots, M \), and \( \beta_m \) and \( \Sigma_m \) are the estimated regression coefficient and the variance-covariance matrix for the \( m^{th} \) cluster, respectively. We choose \( c \) to downweight a certain proportion of \( r_{im} \)’s and and run the EM algorithm again.
2.3.4 Examples - Mixtures of multivariate $t$-distributions with 3 degrees of freedom

In this section, we illustrate the performance of the REM 1 on three different models. We generated one data set from each of the following models: (1) 3 component mixture with 4 covariates, (2) 2 component mixture with 3 covariates, (3) 2 component mixture with 4 covariates. To assess how well the REM 1 works for the heavy-tailed data, we generate each component from a multivariate $t$-distribution. The multivariate $t$-distribution for $x \in \mathbb{R}^d$ with $\nu > 0$ degrees of freedom, the mean $\mu \in \mathbb{R}^d$ and the symmetric and positive-definite variance-covariance matrix $\Sigma_{d \times d}$ has the following density:

$$t_d(x; \nu, \mu, \Sigma) = \frac{\Gamma\left(\frac{\nu + d}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} (\nu \pi)^{d/2} |\Sigma|^{-1/2} \left[ 1 + \frac{1}{\nu} (x - \mu)' \Sigma^{-1} (x - \mu) \right]^{-(\nu+d)/2}.$$  (2.3)

For each $M$ component mixture in our examples, the density of the sampling distribution of the $d$-dimensional $i^{th}$ response vector $Y_i$ is given by

$$f(Y_i) = \sum_{m=1}^{M} \lambda_m t_d(\nu, X_i \beta_m, \Sigma_m),$$  (2.4)

where $X_i$ is the $i^{th}$ row vector of covariates of length $p$, $\beta_m$ is a $p \times d$ matrix of regression coefficients, and $\Sigma_m$ is a $d \times d$ variance-covariance matrix, which will be given later under each model. We choose $\nu = 3$.

In the EM, the expected complete-data log-likelihood increases as iteration goes on. Therefore, we normally run the EM until the likelihood does not significantly increase at each iteration, and take the last values of the parameters as the optimal estimators. However, our main goal is to detect the true clusters. We compute the adjusted rand index (ARI) as a measure for the degree of match between true and estimated indices as we did in Section 2.3. Even though our true parameters would reach the optimal values as iteration proceeds, the value of ARI might not necessarily increase together with the log-likelihoods because it is not computed based on the parameters. It only uses the true and inferred indices to produce a value. We report the results at which the ARI reaches the maximum.

Model 1. Three component mixture with four covariates

Similarly to the simulation study for multivariate normal responses in Section 2.2.4, a dataset with 300 observations was generated to form 3 clusters on R version 2.7.2. The responses are 10-dimensional and there are 4 covariates. The vectors of four covariates $x_1, x_2, x_3$ and $x_4$ were generated in the following way: $x_1$ is sampled uniformly from $[0.1, 2.1]$; $x_2$, conditioning on $x_1$, is Gaussian with mean 0 and variance
(1 + x_1)/(2 + x_1): x_3, independent of x_1, x_2, is a Bernoulli random variable with success rate 0.6.; x_4, conditional on x_2 is Gaussian with mean x_2 and variance 1. We set the seed to i to generate the i^{th} set of covariates for i = 1, · · · , 300. The compound symmetric variance-covariance matrices for each of the 3 components is generated in the following way; Σ_1 is a matrix with the diagonal elements equal to 3 and the off-diagonal elements equal to 2; Σ_2 is a matrix with 6 on the diagonal and 3 otherwise; and lastly, Σ_3 has the diagonal elements of 6 and the off-diagonal elements of 4.

We create the matrices of the regression coefficients β_m’s for m = 1, 2 and 3, each of which is a 5 × 10 matrix including the row for the intercept. Let β_m = (β_m(1), · · · , β_m(10)) be the k^{th} row of β_m for k = 0, 1, · · · , 4, where k = 0 represents the row for the intercept. The equation (2.5) shows β_m(t)’s, for t = 3, 6, · · · , 30 for each m = 1, 2 and 3, that are also graphically shown in Figure 2.2.

\[
\begin{align*}
\beta_1^0(t) &= 1.5 + 2 \sin(\pi t/60) \\
\beta_1^1(t) &= 2 - 3 \cos(\pi(t - 25)/15) \\
\beta_1^2(t) &= 0.6 - 0.12t \\
\beta_1^3(t) &= -4 + (22 - t)^3/1000 \\
\beta_1^4(t) &= 0.1 + (t - 0.3)I(t > 0.3) + (t - 0.7)I(t > 0.7) \\
\beta_1^5(t) &= 3 + 0.2t \\
\beta_1^6(t) &= 0.1 + (t - 4)I(t > 4)/20 + (t - 24)I(t > 24)/15 \\
\beta_1^7(t) &= 5 + (10 - t)^3/2000 \\
\beta_1^8(t) &= -2.2 + 1.3 \cos(\pi(t - 2.5)/12) \\
\beta_1^9(t) &= 1.8 + 1.2 \sin(\pi t/60) \\
\beta_1^{10}(t) &= (15 - t)^3/3000 \\
\beta_2^0(t) &= 2 - 2.2|t - 20|/5 + \sin(\pi t/60) \\
\beta_2^1(t) &= 1.2 - 4.3 \cos(\pi(t - 25)/10) \\
\beta_2^2(t) &= 0.1 + (t - 10)I(t > 10) + (t - 20)I(t > 20) - (t - 2)I(t > 2) \\
\beta_2^3(t) &= 1.5 + 2 \sin(\pi t/60).
\end{align*}
\]

The random errors e_i = [e_i(1), · · · , e_i(T)] are generated from the multivariate t-distribution with df = 3, mean x_iβ_m and variance-covariance matrix Σ_m, m = 1, 2, 3 for each mixture component. Each response is generated by random draws from one of the mixture components with equal probabilities. See Figure 2.3 to see how the responses are clustered in a 3-dimensional space.
We see that cluster 1 is clearly distinguished, but clusters 2 and 3 are a bit ambiguous. To find how many of the true cluster membership indices could be detected by the REM 1, we ran it for \( c = 8 \), which is equivalent to downweighting 1.33% of the outliers. As a result, we reached the optimal point around 36th iteration during the 150 iterations, and the ARI was 0.4505, while K-means and Mclust produced 0.1529 and 0.3002, respectively, for ARI.

**Model 2. Two component mixture with three covariates**

Now we work on a simpler case by reducing the number of mixture components from 3 to 2 and the number of covariates from 4 to 3. We created the clusters by taking only the first 2 components and the first 3 covariates of those in Model 1. The responses are displayed on a 2-dimensional space in Figure 2.4. When \( c \) equals 8, ARI for 2 cluster case reached 0.921378 at the 36th iteration and lasted until 66th. Later it decreased a little. It seems that the algorithm reached the optimal point at the 36th iteration in terms of the ARI values. The ARI’s for K-means and Mclust were 0.2319545 and 0.6387946, respectively.

**Model 3. Two component mixture with four covariates**

We created another dataset with 2 components and 4 covariates by taking the first 2 components and all covariates of those in Model 1. \( c \) is set to 8 to downweight 3 extreme outliers. ARI reached the maximum value of 0.9342 around the 34th iteration during 150 iterations, while the likelihood kept increasing until the end of the iterations. Fig 2.5 shows how the log-likelihood increases. While the log-likelihood consistently increased, the ARI started to decrease after 36th iteration, and increased again later to 0.9213. After that it kept decreasing until it reached 0.8832 at the last iteration. While the maximum of the ARI values from the REM 1 was close to 1 indicating a very good match, the ARI’s from Mclust and K-means were only 0.5967 and 0.2379, respectively. Figure 2.6 shows the true indices and the indices guessed from the REM 1 algorithm at 34th iteration, which produced the highest ARI during 150 iterations.

The three examples above were illustrated to check whether the REM 1 works, and to compare its performance to other existing normality methods which do not use covariates. We found that the REM 1 works better than the K-means and the Mclust on the datasets in the examples. The superior performance of the REM 1 can be attributed to its robustness or the use of covariates. In the next section, we propose another robust clustering method, and investigate the contributions of its robustness and the use of covariates in a detail.
2.4 The robust EM clustering through weighted least squares estimation

The REM 1 in section 2.3 showed a good performance in detecting true cluster indices, but its computational cost is problematic. The algorithm involves optimization steps to estimate $\beta_m$ and $\Sigma_m$, $m = 1, \cdots, M$, which slows down the computing speed. To overcome its computational drawback while still maintaining robustness, we employ the iteratively reweighted least squares in the M-step, and propose another robust clustering method, REM 2. We compute weights for each $(i, m)$, $i = 1, \cdots, n, m = 1, \cdots, M$, using the parameter estimates at the previous iteration, and do the weighted least squares estimation. The weights are computed by taking the ratio between the residuals applied to the Huber’s loss function and the squared residuals. In this way, we can downweight outliers by giving small weights to the residuals whose absolute values are far from $c$. In the next sections, we introduce how the REM 2 works via the EM algorithm.

2.4.1 The E-step

We assume that the $m^{th}$ component has the following density;

$$f_m(y_i|\theta_m) = \frac{1}{C|\Sigma_m|^{1/2}} \exp \left\{ -\frac{1}{2} w_{im} (y_i - x_i \beta_m) \Sigma_m^{-1} (y_i - x_i \beta_m)' \right\},$$

where $C$ is the normalizing constant, and the weights $w_{im}$ are defined by

$$w_{im} = \rho \left( \frac{(y_i - x_i \beta_m) \Sigma_m^{-1} (y_i - x_i \beta_m)'}{(y_i - x_i \beta_m) \Sigma_m^{-1} (y_i - x_i \beta_m)'} \right),$$

(2.6)

where $\rho$ is Huber’s loss function in (3.24), when $y_i - x_i \beta_m \neq 0$. When $y_i - x_i \beta_m = 0$, we assume $w_{im} = 0$ and

$$f_m(y_i|\theta_m) = 1/C|\Sigma_m|^{1/2}.$$

As in previous sections, the expected complete data log-likelihood is

$$E_{\zeta}^{\text{old}} \log P(y, \zeta|\theta, \lambda) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left( \log \lambda_m + \log f_m(y_i|\theta_m) \right),$$

where

$$z_{im} = \frac{\lambda_{im} f_m(y_i|\theta_{im})}{\sum_{m=1}^{M} \lambda_{im} f_m(y_i|\theta_{im})}.$$
2.4.2 The M-step

In the M-step, we must find $\theta$ and $\lambda$ to increase $E^{old}_\zeta \log P(y, \zeta | \theta, \lambda)$. We use the Lagrange multiplier $L$ with the constraint $\sum_{m=1}^{M} \lambda_m = 1$. Let

$$l(\theta) = \sum_{i=1}^{n} \sum_{m=1}^{M} z_{im} \left\{ \log \lambda_m + \log \frac{1}{C^{|\Sigma_m|/2}} - \frac{1}{2} w_{im}(y_i - x_i \beta_m) \Sigma_m^{-1} (y_i - x_i \beta_m)' \right\} - L \cdot \left( \sum_{m=1}^{M} \lambda_m - 1 \right).$$

The weights $w_i$ for each $i$ are computed by using the current values of $\beta_m$ and $\Sigma_m$ throughout the iterations. We maximize $l(\theta)$ with respect to the parameters in the following steps. At each iteration of the M-step,

1. Update $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_M)$.
   
   Since $\lambda_m$ is not involved in the density, we get the same estimate of $\lambda_m$ as in 2.2.2:

   $$\lambda_m^{new} = \frac{\sum_{i=1}^{n} z_{im}}{\sum_{i=1}^{n} \sum_{m=1}^{M} z_{im}} = \frac{\sum_{i=1}^{n} z_{im}}{n}$$

   for each $m \in \{1, 2, \cdots, M\}$.

2. Update $\beta = (\beta_1, \beta_2, \cdots, \beta_M)$ by

   $$\beta_m^{new} = \left\{ \sum_{i=1}^{n} z_{im} w_{im} x_i' x_i \right\}^{-1} \left\{ \sum_{i=1}^{n} z_{im} w_{im} x_i' y_i \right\}.$$

   We have an additional loop at each iteration of the M-step to iteratively update $\beta_m$ by recalculating $w_{im}$ in (2.6) until $\beta_m$ does not change much.

3. Update $\Sigma = (\Sigma_1, \Sigma_2, \cdots, \Sigma_M)$ with $\beta_m$ fixed at the current step.

   Solving $l(\theta)$ for $\Sigma_m$ gives

   $$\Sigma_m^{new} = \frac{\sum_{i=1}^{n} z_{im} w_{im} (y_i - x_i \beta_m)' (y_i - x_i \beta_m)}{\sum_{i=1}^{n} z_{im}}.$$

   As in Step 2, $\Sigma_m$ is iteratively updated by recalculating $w_{im}$ in (2.6) until convergence within an additional loop.

2.4.3 Simulation study - Two component mixture with three covariates, generated from multivariate skew t-distributions.

We conduct a Monte Carlo simulation study to assess the performance of the REM 2 and compare it to other existing methods. A total of 100 datasets with 300 observations partitioned into two true clusters each were...
generated to assess the performance of the algorithm on R version 2.7.2. The seeds to generated 100 datasets are set to 1 to 100. The responses are 10-dimensional and three covariates were generated in the following way: \( x_1 \) is sampled uniformly from \([0, 1, 2, 1]\); \( x_2 \) is Gaussian with mean 0 and variance \((1 + x_1)/(2 + x_1)\); \( x_3 \), independent of \( x_1 \) and \( x_2 \), is a chi-square random variable with 3 degrees of freedom. The 4 \times 10 matrices of coefficients \( \beta_m \)'s for \( m = 1, 2 \) are the same as those in Model 2 in Section 2.3.4. To help reminding, we repeat their formula here. Let \( \beta_k^m = (\beta_k^m(t_1), \cdots, \beta_k^m(t_{10})) \) be the \( k \)th row of \( \beta_m \) for \( k = 0, 1, \cdots, 3 \), where \( k = 0 \) represents the row for the intercept. The equation (2.7) shows \( \beta_k^m (t) \)'s, for \( t = 3, 6, \cdots, 30 \) for each \( m = 1 \) and 2.

\[
\begin{align*}
\beta_0^1 (t) &= 1.5 + 2 \sin(\pi t/60) \\
\beta_1^1 (t) &= 2 - 3 \cos(\pi (t - 25)/15) \\
\beta_2^1 (t) &= 0.6 - 0.12t \\
\beta_3^1 (t) &= -4 + (22 - t)^3/1000 \\
\beta_0^2 (t) &= 3 + 0.2t \\
\beta_1^2 (t) &= 0.1 + (t - 4) I(t > 4)/20 + (t - 24) I(t > 24)/15 \\
\beta_2^2 (t) &= 5 + (10 - t)^3/2000 \\
\beta_3^2 (t) &= -2.2 + 1.3 \cos(\pi (t - 2.5)/12)
\end{align*}
\] (2.7)

The variance-covariance matrices \( \Sigma_1 \) and \( \Sigma_2 \) are also the same as those in Model 2 in Section 2.3.4. \( \Sigma_1 \) is a matrix with the diagonal elements equal to 3 and the off-diagonal elements equal to 2; \( \Sigma_2 \) is a matrix with 6 on the diagonal and 3 otherwise.

To see how well the REM 2 works against non-normal data, we generated datasets based on a mixture of the multivariate skew \( t \)-distribution (Azzalini and Capitanio, 2003). It was mentioned in Azzalini and Capitanio (2003) that the multivariate skew \( t \)-distribution represents mathematically quite a manageable distribution, allowing ample flexibility in skewness and kurtosis, and therefore it appears to be a promising tool for a wide range of practical problems. Its density, properties and its statistical aspects are discussed in great detail in Azzalini and Capitanio (2003). We choose to use the multivariate skew \( t \)-distributions since we believe their tails that are heavier than multivariate normal distributions and their asymmetry feature would provide a realistic approximation to the real data.

The multivariate skew \( t \)-distribution involves degrees of freedom, a vector of the location parameters, a vector of the shape parameters and a positive definite covariance matrix. In our simulation study, the errors
were generated from the multivariate skew \( t \)-distribution with 3 degrees of freedom. The location parameter is \( x_i\beta_m \) for \( m = 1 \) and \( 2 \), and shape parameter is chosen to be a vector of twos with length 10. The cluster indices were randomly drawn from \( \{1, 2\} \) with equal probability. In other words, for the randomly selected index \( I, I = 1, 2 \),

\[
y_i = x_i\beta_I + e_i,
\]

where \( e_i \) is drawn from multivariate skew \( t \)-distribution described above.

As in Section 2.3, to determine the threshold \( c \) in Huber’s loss function, we first set the threshold to a small value and ran the algorithm with the initial parameters estimated from clustering via Mclust. Then we compute the following quantities for estimated clusters:

\[
\begin{align*}
  r_{i1} &= \sqrt{\left(y_{i1} - x_{i1}\beta_1\right)\Sigma_1^{-1}\left(y_{i1} - x_{i1}\beta_1\right)^t}, \\
  r_{i2} &= \sqrt{\left(y_{i2} - x_{i2}\beta_2\right)\Sigma_2^{-1}\left(y_{i2} - x_{i2}\beta_2\right)^t},
\end{align*}
\]

where \( y_m \) and \( x_m \) are subsets of \( Y \) and \( X \) corresponding to the \( m^{th} \) cluster, and \( \beta_m \) and \( \Sigma_m \) are estimated regression coefficient and covariance matrix for cluster \( m, m = 1, 2 \). We choose \( c \) to downweight 15% of \( r_1 \) and \( r_2 \) in this study. When the algorithm stops due to the singularity of \( \Sigma_m \), we add to the diagonal elements of \( \Sigma_m \) a small number, proportional to the maximum eigenvalues of \( \Sigma_m \) and proceed.

We made a various comparisons between the REM 2 and other clustering algorithms using ARI’s introduced in Section 2.3. We wish to find where the improvement comes from - whether it is from including covariates in the model, or from downweighting outliers. To address the question, we performed various comparisons by slight modification of the REM 2 in addition to the comparisons with the Mclust and the K-means. The REM 2 can be easily converted to the multivariate normal mixture with covariates, which we would call the LSEM, by adjusting the value of \( c \) to a large number. We can also run the REM 2 and the LSEM without covariates to see how much contribution the covariates give to the clustering performance. Additionally, we compare the REM 2 with and without covariates to Mclust and to K-means.

Figure 2.7 shows the comparisons between the REM 2 and other clustering methods mentioned above. The ARI values from two different clustering algorithms are plotted against each other. There are 100 points representing 100 data sets. The reference line is the line of \( y = x \), which helps us see how much the REM 2 works better than others in terms of ARI values. More points below the reference line suggest the method on \( x \)-axis detects cluster indices more accurately. See the chart in Table 2.2 for the description of the plots.

In plot 1 of Figure 2.7, the ARI values from the REM 2 with and without covariates are mostly similar,
Table 2.2: The chart for the $x$ and $y$ labels in Figure 2.7.

<table>
<thead>
<tr>
<th>plots</th>
<th>$x$-axis</th>
<th>$y$-axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>REM 2 with covariates (hubercov)</td>
<td>REM 2 without covariates (hubernocov)</td>
</tr>
<tr>
<td>2</td>
<td>REM 2 with covariates</td>
<td>LSEM with covariates (ls cov)</td>
</tr>
<tr>
<td>3</td>
<td>REM 2 with covariates</td>
<td>LSEM without covariates (lsnocov)</td>
</tr>
<tr>
<td>4</td>
<td>REM 2 with covariates</td>
<td>Mclust (mc)</td>
</tr>
<tr>
<td>5</td>
<td>REM 2 with covariates</td>
<td>K-means (km)</td>
</tr>
<tr>
<td>6</td>
<td>REM 2 without covariates</td>
<td>Mclust</td>
</tr>
<tr>
<td>7</td>
<td>REM 2 without covariates</td>
<td>K-means</td>
</tr>
<tr>
<td>8</td>
<td>REM 2 with covariates</td>
<td>Mclust</td>
</tr>
<tr>
<td>9</td>
<td>REM 2 without covariates</td>
<td>Mclust</td>
</tr>
</tbody>
</table>

but there are slightly more datasets with higher ARI values in the REM 2 with covariates. This implies a mild contribution of the covariates to clustering. Plot 2 shows that the REM 2 with covariates works better than the LSEM with covariates for most of the datasets. This suggests that downweighting outliers improve the clustering to a great degree. From plots 3 to 7, we notice that the REM 2 outperforms other methods for the majority of the simulated datasets. It is also interesting to see plots 8 and 9. Plot 8 is the plot of the LSEM with covariates versus Mclust, and plot 9 is that of the LSEM without covariates versus Mclust. Without covariates, the LSEM is equivalent to Mclust, since both are based on the EM algorithm for the mixture of multivariate normal distribution. But if we include covariates in the model, then the clustering performance can be improved as we observe in plot 8.

2.5 Conclusion

In this chapter, we have proposed two robust clustering algorithms - the REM 1 and the REM 2. Both algorithms are based on the regression mixtures of modified multivariate normal distributions, which involve Huber’s loss function. Parameters are estimated via the EM algorithm, with a direct optimization (REM 1) or the iteratively reweighted least squares (REM 2) in the M-step. The REM 2 is computationally simpler and faster than the REM 1 since the parameters have closed-form solutions. We have shown from various comparisons that the REM 1 and the REM 2 outperform other existing methods in terms of the adjusted rand indices (ARI) when the data are from a mixture of multivariate $t$, or multivariate skew $t$-distributions. The simulation study suggests the the superior performance of the REM 1 and 2 comes mainly from their robustness and partly from their use of covariates.

It remains as a future work to compare the proposed robust clustering methods with the multivariate $t$ clustering, proposed in McLachlan and Peel (1998) and Peel and McLachlan (2000). The $t$-distribution provides a longer-tailed alternative to the normal distribution, and provides a more robust approach to the
fitting of normal mixture models (McLachlan and Peel, 2000). The degrees of freedom of the \( t \)-distributions plays a role as a robustness tuning parameter, as \( c \) does in the REM 1 and the REM 2. The clustering via the regression mixture of the multivariate \( t \)-distribution has not been studied yet, and formulating it and comparing it to the REM 1 and the REM 2 would be our next work.

### 2.6 Figures of Chapter 2

**Figure 2.1:** The plot of the response \( Y \) produced by 3-dim multidimensional scaling. The dataset was generated from the mixture of three multivariate normal distributions, as described in Section 2.2.4. Numbers represent the cluster indices.

**Figure 2.2:** The coefficient functions for each of three mixture components in Model 1 given in Section 2.3.4. For each of the \( m^{th} \) mixture component, \( m = 1, 2, 3 \), the plots of \( \beta_m^p(t) \), for \( t = 3, \ldots, 30 \) and \( p = 1, \ldots, 4 \) are shown.
Figure 2.3: 3-dim multi-dimensional scaling for the responses in Model 1 of Section 2.3.4 with true cluster indices attached.

Figure 2.4: 2-dim multi-dimensional scaling for the responses in Model 2 of Section 2.3.4 with true cluster indices attached.
Figure 2.5: Log-likelihoods of the data from Model 3 in Section 2.3.4 up to 150 iterations for c=8.

Figure 2.6: True indices and indices for the data from Model 3 in Section 2.3.4 from the REM 1 at the 36th iteration.
Figure 2.7: ARI values from REM 2 against other clustering algorithms. ‘hubercov’ and ‘hubernocov’ represent REM 2 with and without covariates, respectively. ‘lscov’ and ‘lsnocov’ represent LSEM with and without covariates. ‘mc’ represents Mclust, and ‘km’ represents K-means.
Chapter 3

Lasso-type Robust Variable Selection for Time-Course Microarray Data

3.1 Introduction

Consider a multivariate linear regression model

\[ Y = \mathbb{1}_n \beta_0 + X \beta + \epsilon, \]

where \( Y \) is an \( n \times d \) matrix of the response variables for \( d \geq 1 \), \( \mathbb{1}_n \) is an \( n \times 1 \) vector of ones, \( \beta_0 \) is a \( 1 \times d \) vector of the intercepts, \( X \) is a \( n \times p \) matrix of \( p \) covariates, \( \beta \) is a \( p \times d \) matrix of the regression coefficients, and \( \epsilon \) is an \( n \times d \) matrix of errors, of which the rows are independent and identically distributed with mean zero. When \( p \) is large, identifying the subset of covariates having nonzero coefficients is one of the important statistical problems of interest.

A variety of procedures of variable selection are available. The hypothesis testing in Chapter 1 is one of the traditional methods of variable selection along with the subset selection methods using information criteria, such as Akaike information criterion and the Bayes information criterion. More recently, the variable selection via the regularized estimation has received much attention since the appearance of the Lasso proposed by Tibshirani (1996). Lasso minimizes the residual sum of squares subject to the sum of the \( L_1 \)-norm of the coefficient being less than a constant. It produces sparse models like the subset selection, and exhibits the stability of the ridge regression (Tibshirani, 1996). Fan and Li (2001) pointed out that the bias created by the Lasso is noticeably large when noise level is not high. They proposed the smoothly clipped absolute deviation (SCAD), which gives a good performance in selecting significant variables without creating excessive biases. Fan and Li (2004) further used the SCAD penalty for longitudinal data analysis when \( \beta(t) \) is assumed to be time-dependent. Zou (2006) proposed the adaptive Lasso as a new version of the Lasso, where adaptive weights are used for penalizing different coefficients in the \( L_1 \) penalty. They showed that the adaptive Lasso has oracle properties, which means that it performs as well as if the true underlying model was known in advance.
Recently, the attention has focused on selecting groups of variables rather than individual variables. The group Lasso of Yuan and Lin (2006) is an extension of the Lasso, designed to select groups of variables by using the sum of the weighted norms of the coefficient vectors on each group of variables as its penalty. The group Lasso received an immediate attention from many researchers leading to a number of publications in a short period. Bach (2008) extended some of the theoretical results of the Lasso to the group Lasso, for finite and infinite dimensional groups, and provided necessary and sufficient conditions for model consistency of the group Lasso. Some variants of the group Lasso include a supervised group Lasso in Ma et al. (2007) which takes the cluster structure in gene expression data into account. Wang and Leng (2008) proposed an adaptive group Lasso to improve the estimation efficiency and the selection consistency of the group Lasso. Jacob et al. (2009) presented a generalization of the group Lasso penalty in the context of graphical models. As an extension of the SCAD to the grouped variable selection, Wang et al. (2007) developed the group SCAD for microarray time course gene expression data. Wang et al. (2008) further developed the gSCAD procedure for general nonparametric varying coefficient models with possible time-dependent covariate processes.

The main focus of this chapter is on developing another variant of the group Lasso. We note that the existing Lasso and the group Lasso methods minimize the $L_2$ norm of the residuals. They work well when the true underlying distribution is normal, but some consideration on robust alternatives are needed due to their sensitivity to the potential outliers, when the errors are from a heavier tailed distribution.

As a robust variant of the Lasso, Rosset and Zhu (2004) proposed ‘Huberized Lasso’, which uses a loss function that has a similar shape to Huber’s loss function. They showed through simulation that the Lasso fails in identifying the correct model in the presence of outliers, while their method identifies it almost exactly with the appropriate choice of the regularization parameter. Other robust variants of the Lasso include Owen (2007), which proposed a robust hybrid of the Lasso and the ridge regression, and Wang et al. (2007), which combined the least absolute deviation (LAD) regression and the Lasso. No robust version of the group Lasso has been proposed.

Another thing we note is that the Lasso and the group Lasso methods are developed mainly for a single response variable. Turlach et al. (2005) extended Lasso so that it can select variables to model several response variables, but the solutions are not sparse, and hence we need a heuristic to select significant explanatory variables. Thus it remains as a task to develop a robust variable selection which can handle multivariate responses and can produce a subset of important covariates without heuristics.

To this end, we propose two robust variable selection methods. The first method, named as the R Lasso 1, is developed by modifying the original group Lasso in Yuan and Lin (2006) with a new objective
function using Huber’s loss function. The penalty function of the original group Lasso is modified as well to incorporate the new objective function. The algorithm developed in Yuan and Lin (2006) can be used to implement our robust group Lasso with transformations of the data. The second method, named as the R Lasso 2, is developed to improve the computational efficiency of the R Lasso 1. It selects variables through a sequence of ridge regressions. It requires fewer transformations than the R Lasso 1, and runs faster than the R Lasso 1.

The robust Lasso methods distinguishes themselves from other existing methods in that it downweights outliers and handles multiple responses by grouping. They can be applied to the general multivariate responses, but we particulary focus on the time-course response variables whose measurements can be viewed as a function of time measured at a few time points. It enables us to achieve the dimension reduction of the regression coefficients by the basis approximation and reduce the computational burden, which is one of the main difficulties in using the multivariate regression models.

In Section 3.2, we present a brief review on the original group Lasso of Yuan and Lin (2006). Sections 3.3 and 3.4 introduce the R Lasso 1 and 2 along with simulation studies to assess their performance compared to the original group Lasso. We report the results from the application to the yeast microarray data and its transcription factors in Section 3.5.

3.2 Original group Lasso

The group Lasso of Yuan and Lin (2006) is the extension of the well-known variable selection method Lasso of Tibshirani (1996). The goal of the group Lasso is to select important groups of variables, which we would call ‘factors’, as in Yuan and Lin (2006), rather than select only individual variables. We summarize the group Lasso below with an extract from Yuan and Lin (2006).

Consider the model with \( J \) factors:

\[
Y = \sum_{j=1}^{J} X_j \beta_j + \varepsilon, \tag{3.2}
\]

where \( Y \) is a \( n \times 1 \) vector, \( \varepsilon \sim N_n(0, \sigma^2 I) \), \( X_j \) is an \( n \times p_j \) matrix corresponding to \( j \)th factor, and \( \beta_j \) is a coefficient vector of size \( p_j \), \( j = 1, \ldots, J \). The response variable and each input variable are centered to eliminate the intercept from Model (3.2). To simplify the description, each \( X_j \) is orthonormalized, i.e., \( X_j'X_j = I_{p_j}, \; j = 1, \ldots, J \), by Gram-Schmidt orthonormalization. Denoting \( X = (X_1, X_2, \ldots, X_J) \) and \( \beta = (\beta_1', \ldots, \beta_J')' \), equation (3.2) can be written as \( Y = X\beta + \varepsilon \). For a vector \( \eta \in R^d, \; d \geq 1 \), and a
symmetric \(d \times d\) positive definite matrix \(K\), denote
\[
\|\eta\|_K = (\eta' K \eta)^{1/2}.
\]

It can be regarded as a norm of \(\eta\) projected onto a space determined by \(K^{1/2}\). If \(K = I_d\), then \(\|\eta\|_K\) is a norm of \(\eta\) in \(R^d\). There are many choices for \(K\), and \(\|\eta\|_K\) is in other words a weighted norm of \(\eta\). For example, if we choose \(K_j = p_j I_{p_j}\), then we can put more weights on the factors with more variables to force their coefficients to become 0 in the group Lasso.

Given positive definite matrices \(K_1, \cdots, K_J\), the group Lasso estimate is defined as the minimizer of
\[
\frac{1}{2} \left\| Y - \sum_{j=1}^J X_j \beta_j \right\|^2 + \lambda \sum_{j=1}^J \|\beta_j\|_{K_j},
\]
where \(\lambda \geq 0\) is a tuning parameter. There are many choices for \(K_j's\), and one reasonable choice when all \(p_j's\) are equal to \(p\) is \(K_j = I_p, j = 1, \cdots, J\), or \(K_j = p_j I_{p_j}, j = 1, \cdots, J\). Yuan and Lin (2006) verified that the solution to (3.3) for \(K_j = p_j I_{p_j}, j = 1, \cdots, p\) can be obtained by iteratively applying
\[
\beta_j = \left(1 - \frac{\lambda \sqrt{p_j}}{\|S_j\|}\right) S_j,
\]
to \(j = 1, \cdots, J\), where \(S_j = X_j'(Y - X \beta_{-j})\), with \(\beta_{-j} = (\beta_1', \cdots, \beta_{j-1}', 0', \beta_{j+1}', \cdots, \beta_J')\).

Although the group Lasso is not robust against outliers due to its squared error term in the objective function, the algorithm has been found to be stable and usually reaches convergence quickly (Yuan and Lin, 2006). The update formula (3.4) requires that each \(X_j\) is orthogonal. Otherwise, this algorithm does not converge to the right solution. In the next section, we describe how we may transform our problem to fit it into the group Lasso.

### 3.2.1 Using the group Lasso for multivariate data

Both the Lasso (Tibshirani, 1996) and the group Lasso (Yuan and Lin, 2006) are developed for univariate response variables. The simultaneous variable selection proposed by Turlach et al. (2004) deals with multivariate response, but the solutions are not sparse. We consider multivariate responses with the time-varying measurements for each case. Although the responses are multivariate, we can implement the original group Lasso by going through the preprocessing steps to be described in detail in this section. In our application, the group Lasso will be used to group the response variables, not to group the covariates.
The structures of the original data we have can be written as

\[
Y = \begin{pmatrix}
  y_1(1) & \cdots & y_1(T) \\
  y_2(1) & \cdots & y_2(T) \\
  \vdots & \ddots & \vdots \\
  y_n(1) & \cdots & y_n(T)
\end{pmatrix},
\quad X = \begin{pmatrix}
  1 & x_{11} & x_{12} & \cdots & x_{1p} \\
  1 & x_{21} & x_{22} & \cdots & x_{2p} \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  1 & x_{n1} & x_{n2} & \cdots & x_{np}
\end{pmatrix}.
\]

Our model for the original data \(Y\) and \(X\) is given by

\[
Y_{n \times T} = X_{n \times (p+1)} \beta_{(p+1) \times T} + \epsilon_{n \times T},
\]

where the independent errors \(\epsilon_i\) have a variance-covariance matrix \(\Omega_{T \times T}\), and the coefficient \(\beta\) is given as

\[
\beta = \begin{pmatrix}
  \beta_0(1) & \cdots & \beta_0(T) \\
  \beta_1(1) & \cdots & \beta_1(T) \\
  \vdots & \ddots & \vdots \\
  \beta_p(1) & \cdots & \beta_p(T)
\end{pmatrix}.
\]

Each \(\beta_j = (\beta_j(1), \cdots, \beta_j(T))\) for \(j = 1, \cdots, p\) is the row vector of coefficients for \(X_j = (X_{1j}, \cdots, X_{nj})'\) at time \(t\), which can be viewed as a functional object. If \(X_j\) is not an important variable for \(Y\), we assume that the \(\beta_j(t) = 0\) for all \(t = 1, \cdots, T\). Our goal is to select important variables, which amounts to deciding whether to set the vector \(\beta_j = (\beta_j(1), \cdots, \beta_j(T))\) to zero for each \(j\). To formulate the problem in the group Lasso form, we need several steps of data transformation. The transformation takes the following steps.

1. To take advantage of the time-varying nature of \(\beta\), we employ the basis representation of \(\beta\). We use the cubic B-splines to approximate \(\beta_j(t)\) with \(d\) degrees of freedom by putting \(d - 3\) equally spaced knots between 1 and \(T\). For example, the expression \(bs(x, df = 7)\) in R for an 15-vector \(x\) generates a \(15 \times 7\) basis matrix of cubic-B-spline functions evaluated at the 15 observations in \(x\), with \(7-3=4\) interior knots at the 20, 40, 60 and 80th percentiles of \(x\).

The cubic B-splines with \(d\) degrees of freedom produces a \(T \times d\) matrix

\[
B_{T \times d} = \begin{pmatrix}
  B_1(1) & \cdots & B_d(1) \\
  \vdots & \ddots & \vdots \\
  B_1(T) & \cdots & B_d(T)
\end{pmatrix}.
\]
Since every $\beta_j(t)$ shares the same time points $t = 1, \cdots, T$, each $\beta_j(t)$ can be expressed as

$$\beta_j(t) = [\alpha_{11}, \cdots, \alpha_{1d}] \begin{pmatrix} B_1(t) \\ \vdots \\ B_d(t) \end{pmatrix},$$

where $B(t) = (B_1(t), \cdots, B_d(t))'$ is the same for all $j$. This enables us to write $\beta$ as

$$\beta = \begin{pmatrix} \alpha_{11} & \cdots & \alpha_{1d} \\ \vdots & \ddots & \vdots \\ \alpha_{p1} & \cdots & \alpha_{pd} \end{pmatrix} \begin{pmatrix} B_1(1) & \cdots & B_1(T) \\ \vdots & \ddots & \vdots \\ B_d(1) & \cdots & B_d(T) \end{pmatrix} = \beta^{[1]} B',$$

and then the model is written as

$$Y = X\beta^{[1]}B' + \epsilon.$$

By singular value decomposition of $B'$, we can find an $H_{T \times d}$ such that $B'H = I_{d \times d}$. Then,

$$YH = X\beta^{[1]} + \epsilon H.$$

By letting $Y^{[1]} = YH$ and $\epsilon^{[1]} = \epsilon H$, we get

$$Y^{[1]} = X\beta^{[1]} + \epsilon^{[1]}.$$ (3.5)

2. Although the basis approximation of $\beta$ reduces the number of parameters to be estimated, the columns of $\epsilon^{[1]}$ are correlated with the variance-covariance matrix $\Sigma_{d \times d} = H'\Omega H$. We estimate $\Sigma$ by using the residuals obtained from the ordinary least squares regressions of each column of $Y^{[1]}$ on $X$. For the original group Lasso, we estimate $\Sigma$ by computing the sample variance-covariance matrix of these residuals. Later in the robust version of the group Lasso, we compute a robust estimate of $\Sigma$. Once $\hat{\Sigma}$ is obtained, then we consider

$$Y^{[1]}\hat{\Sigma}^{-1/2} = X\beta^{[1]}\hat{\Sigma}^{-1/2} + \epsilon^{[1]}\hat{\Sigma}^{-1/2}.$$ (3.6)

By letting $Y^{[2]} = Y^{[1]}\hat{\Sigma}^{-1/2}$, $\beta^{[2]} = \beta^{[1]}\hat{\Sigma}^{-1/2}$ and $\epsilon^{[2]} = \epsilon^{[1]}\hat{\Sigma}^{-1/2}$, we have

$$Y^{[2]} = X\beta^{[2]} + \epsilon^{[2]},$$ (3.7)
where $\epsilon^{[2]}$ has mean 0 and the variance-covariance matrix approximately equal to $I_k$. The correlations between the columns of $Y^{[2]}$ are ignorable after this transformation.

3. Now we eliminate the intercept from the model. We subtract the column mean from each column of $Y^{[2]}$ and $X$ to get the final model

$$Y = X\gamma + \epsilon^{[3]},$$
(3.8)

where $Y_i$, the $i^{th}$ row of $Y$, is $Y_i^{[2]}$ subtracted by the columnwise mean $\overline{Y}^{[2]}$, and $X$ is obtained by subtracting the columnwise mean $\overline{X}$ from each row of $X$, and then deleting the column of ones from $X$. $\gamma$ is basically the same as $\beta^{[2]}$, but with the first row of $\beta^{[2]}$ deleted, to get rid of the coefficients for the intercept.

4. Next we vectorize the response and adjust the design matrix accordingly so that we can have the same model form as in Yuan and Lin (2006). We create the vectorized response and regression coefficient $Z_{nd \times 1}$ and $\Gamma_{pd \times 1}$, respectively, and the corresponding augmented design matrix $M$ as follows:

$$Z = (y_1(1), \cdots, y_1(d), \cdots, y_n(1), \cdots, y_n(d))_{nd \times 1}^{\prime},$$
$$\Gamma = (\gamma_1(1), \cdots, \gamma_1(d), \gamma_2(1), \cdots, \gamma_2(d), \cdots, \gamma_p(1), \cdots, \gamma_p(k))_{pd \times 1}^{\prime},$$
$$M = X \otimes I_d = (X_1 \otimes I_d, \cdots, X_p \otimes I_d)_{nd \times pd},$$
(3.9)

where $X_j$, $j = 1, \cdots, p$ is the $j^{th}$ column of $X_j$, which represents the $j^{th}$ variable. The model is then written as

$$Z_{nd \times 1} = M_{nd \times pd} \Gamma_{pd \times 1} + \epsilon_{nd \times 1},$$
(3.10)

where $\epsilon$ is the vectorized version of $\epsilon^{[3]}$.

5. Let $M_j = X_j \otimes I_k$ be an $nd \times d$ matrix, and $\gamma_j = (\gamma_j(1), \gamma_j(2), \cdots, \gamma_j(d))^{\prime}$ be a $d$-dimensional vector for $j = 1, \cdots, p$. We orthonormalize each $M_j$ through the QR decomposition, which factorizes $M_j$ into the $nd \times d$ orthonormal matrix $Q_j$ and $d \times d$ upper triangular matrix $R_j$. By doing this, we get

$$M \Gamma = (M_1, \cdots, M_p) \Gamma = (Q_1 R_1, \cdots, Q_p R_p) \begin{pmatrix} \gamma_1 \\ \vdots \\ \gamma_p \end{pmatrix} = (Q_1, \cdots, Q_p) \begin{pmatrix} R_1 \gamma_1 \\ \vdots \\ R_p \gamma_p \end{pmatrix}. $$
(3.11)
After all the transformations, the model is written as

$$Z_{ndx1} = Q_{ndxpd} \Delta_{pdx1} + \epsilon_{ndx1},$$  \hfill (3.12)

where \( Q = (Q_1, \cdots, Q_p) \) and \( \Delta = \begin{pmatrix} R_1 \gamma_1 \\ \vdots \\ R_p \gamma_p \end{pmatrix} \).

We estimate \( \Delta \) in (3.12) using the group Lasso. The five steps of transformation enable us to group the time points. Note that we do not need to center each column of \( Q \) again after the QR decomposition. Since \( X_j \) is centered, all columns of \( M_j \) are centered, or equivalently, \( \mathbf{1}'_n D_j = 0 \). Since \( M_j \) is decomposed as \( Q_j R_j \), we have

$$\mathbf{1}'_n M_j = \mathbf{1}'_n Q_j R_j = 0.$$  \hfill (3.13)

The \( R_j \) is invertible because the columns of \( M_j \) are orthogonal and therefore linearly independent. This implies that \( \mathbf{1}'_n Q_j = 0 \), which means \( Q_j \) is still centered.

Then the group Lasso estimate for the time-course response is defined as the solution to

$$\| Z - Q \Delta \|^2 + \lambda \sum_{j=1}^p \| \Delta_j \|_{K_j},$$  \hfill (3.14)

where \( \Delta_j = R_j \gamma_j \). Since we have \( d \) variables in \( Q_j \) for all \( j = 1, \cdots, p \), we choose \( K_j = I_d \) and omit \( K_j \) hereafter.

3.2.2 Do the transformations hurt the group structure?

The answer is no. To determine whether the \( j \)th covariate is important or not, we only need to check whether all components of the \( j \)th final coefficient \( \Delta_j \) are zeros because the zero rows will be preserved throughout the transformations. If \( \Delta_j = R_j \gamma_j = 0 \), then \( \gamma_j = 0 \), because \( R_j \) is invertible. Since \( \beta^{[1]} \) is just \( \gamma \) augmented by the intercept row, the \( (j+1) \)th row of \( \beta^{[2]} \), which corresponds to the \( j \)th covariate is also \( 0 \). Note that

$$\beta_{p \times T} = \hat{\beta}_{p \times d} B'_{d \times T} = \beta^{[2]}_{p \times d} \Sigma_1^{1/2} B'_{d \times T},$$

which ensures that a zero row of \( \beta^{[2]} \) will be still a zero row of \( \beta \). Since the main purpose of the Lasso is to select important groups, rather than to obtain \( \hat{\beta} \), we do not have to transform the estimated coefficients.
back to the original scale. Once we get the important groups of covariates from the group Lasso, we can estimate $\beta$ using ordinary least squares.

### 3.3 Robust Lasso using adjusted penalty

Consider the case where $\varepsilon$ is not normally distributed, but its distribution is bell-shaped with heavier tails. Rosset and Zhu (2004) showed through simulation that the Lasso fails in identifying the correct model in such cases while a robust loss does it better with an appropriate choice of the regularization parameter. They proposed “Huberized Lasso” for the univariate response. In this section, we propose a robust Lasso for multivariate time-varying response, which can be implemented via the original group Lasso algorithm, which we would call the R Lasso 1. As in Section 3.2, the idea is to group the responses by time, by preprocessing the initial datasets so that they are of the form for the group Lasso algorithm. To make the Lasso more robust, we use Huber’s loss function in the first term of the objective function instead of the squared norm used in the original group Lasso. The minimization is carried out using the iteratively weighted least squares. This procedure involves updating the response and covariate using weights at each iteration, raising an issue that the penalty term should be adjusted to account for the weights that changes at every iteration. Therefore, we introduce a modified objective function based on Huber’s loss function and an adjusted penalty.

Suppose we have already preprocessed the original datasets by going through the steps 1, 2 and 3 in Section 3.2.1. However, step 2 should be done differently. In step 2 of the R Lasso 1, we regress each column of $Y^{[1]}$ on $X^{[0]}$ using a robust method. The method we used for the robust regression will be discussed in detail in Section 3.3.1. The residuals from the robust fit are used to compute the minimum covariance determinant estimator of $\Sigma$.

#### 3.3.1 Determining $c$ in Huber’s loss function

Step 2 in the data preprocessing for the R Lasso 1 involves robust regression. Although any robust regression suffices for step 2 as long as it is robust against large residuals, we choose to use an M-estimation with Huber’s loss function to make it consistent with the R Lasso. For Huber’s loss function in (3.24), we need a threshold value $c$. It controls the amount of residuals that we downweight, and therefore is closely related to $P$, the percentage of downweighting we wish to have. To achieve $P\%$ of downweighting, we locate $c$ at the $(100 - P/2)^{th}$ percentile of the residuals.

Then which residuals do we use? We should obtain the residuals from the regression using Huber’s loss...
function. It sounds somewhat contradictory, since we need the $c$ to compute residuals, and we need residuals to determine $c$. The idea to overcome this problem is that we can unveil most of the outliers by initially setting $c$ to a very small value near 0 and examining residuals from that fit. We start from a robust regression with a small $c$ and save the residuals to determine the appropriate threshold based on them.

To implement robust regression, we use the R function rlm in the MASS package which fits a linear model by Huber’s M estimator. We need to determine two threshold values in Huber’s loss function: one for estimating $\Sigma$ in step 2, and the other for the R Lasso 1 loss function. We denote them by $c^s$ and $c^\ell$, respectively.

### Determining $c^s$ to estimate $\Sigma$ in step 2

1. Regress each column of $Y^{[1]}$ on $X^{[0]}$ by Huber’s M-estimator with $c$ near 0, in particular, we use $c = 0.001$.

2. Extract the scaled residuals and find the $(P/2)^{th}$ and $100 - (P/2)^{th}$ quantiles of them.

3. Set $c^s$ to be the maximum of the absolute values of these two quantiles.

4. Refit each column of $Y^{[1]}$ on $X^{[0]}$ by Huber’s M-estimator with $c = c^s$, and extract unscaled residuals.

The saved residuals are used to compute the minimum covariance determinant estimator of $\Sigma$.

### Determining $c^\ell$ in the R Lasso 1

Once we obtain the preprocessed data $Y$ and $X$, we need determine the value of the threshold $c^\ell$ of the Huber’s loss. In addition, we need to have an initial value of $\beta$ since R Lasso 1 is an iterative process. We can do both through the following steps.

1. Regress each column of $Y$ on $X$ with Huber’s M-estimator with a small $c = 0.001$.

2. Extract the scaled residuals from the fit, and find the $(P/2)^{th}$ and $100 - (P/2)^{th}$ quantiles of the saved residuals.

3. Set $c^{s0}$ to be the maximum of the absolute values of these quantiles.

4. Regress each column of $Y$ on $X$ by Huber’s M-estimator with $c = c^{s0}$.

   (1) Extract the unscaled residuals and form a residual matrix $R_{n \times d}$.

   (2) Extract the coefficients from each regression and stack it column by column to form a $p \times d$ matrix of coefficients $\beta^I$. This $\beta^I$ is used as the initial value of $\beta$ for every $\lambda$ in the Lasso.
5. Note that $c^\ell$ will be used for the norms of the residuals in the R Lasso 1. We compute the norm of each row of $R$, and compute the $(100 - P)^{th}$ quantile of those $n$ values as $c^\ell$.

### 3.3.2 Algorithm of the R Lasso 1

Suppose the initial data are preprocessed as described in Section 3.3. Let $Y$ be an $n \times d$ response matrix with the $i^{th}$ row denoted as $y_i$, and $X$ be an $n \times p$ design matrix of $p$ covariates, with the $i^{th}$ row denoted as $x_i$. The columns of $Y$ are not correlated, and each column of $Y$ and of $X$ is centered. The coefficient $\beta$ is a $p \times d$ matrix with the $j^{th}$ row is denoted as $\beta_j$, $j = 1, \cdots, p$. Let $A_j$ be a $d \times d$ matrix for $j = 1, \cdots, p$, which will be defined later. The objective function of the R Lasso 1 is given by

$$R = \sum_{i=1}^{n} \rho (\|y_i - x_i \beta\|) + \lambda \sum_{j=1}^{p} \|A_j \beta_j'\|. \quad (3.15)$$

The use of $A_j$ will enable us to solve (3.15) using the group Lasso algorithm of Yuan and Lin (2006).

The objective function (3.15) is minimized in the following steps. The solution to (3.15) can be obtained by iteratively reweighted least squares (IRLS) using weights

$$w_i = \frac{\rho (\|y_i - x_i \beta\|)}{\|y_i - x_i \beta\|_2^2}. \quad (3.16)$$

At each iteration of the IRLS, we solve

$$R(w) = \sum_{i=1}^{n} w_i \|y_i - x_i \beta\|^2 + \lambda \sum_{j=1}^{p} \|A_j \beta_j'\|, \quad (3.16)$$

instead of (3.15), where $w_i$ is obtained from the previous estimate of $\beta$ in each step of the iteration.

Note that (3.16) is equivalent to

$$\sum_{i=1}^{n} \|y_i^w - x_i^w \beta\|^2 + \lambda \sum_{j=1}^{p} \|A_j^w \beta_j'\|, \quad (3.17)$$

where $y_i^w = \sqrt{w_i} y_i$ and $x_i^w = \sqrt{w_i} x_i$. Each column of the updated matrices $Y^w = (y_1^w, \cdots, y_n^w)'$ and $X^w = (x_1^w, \cdots, x_n^w)'$ need to be centered again in order to satisfy the requirements for the group Lasso algorithm.

Recall that the original Lasso algorithm require the response to be univariate. Let $Z$ be the vectorized
form of $Y^w$, $D$ be the adjusted design matrix from $X^w$, and $\delta$ be the vectorized coefficient. In other words,

$$Z = (y^w_1(1), \ldots, y^w_d, \ldots, y^w_n(1), \ldots, y^w_n(d))_{nd \times 1},$$
$$D = X^w \otimes I_d = (X^w_1 \otimes I_d, \ldots, X^w_p \otimes I_d)_{nd \times pd},$$
$$\delta = (\beta_1, \ldots, \beta_p)_{pd \times 1}. \quad (3.18)$$

This process produces the univariate response variable, and groups the columns of $Y^w$. Each $nd \times d$ matrix $D_j = X^w_j \otimes I_d$ corresponds to the $j^{th}$ factor (the $j^{th}$ group of covariates) for $j = 1, \cdots, p$.

With $Z$, $D$ and $\delta$, we can write (3.17) as

$$\|Z - D\delta\|^2 = \sum_{\ell=1}^{nk} (Z_\ell - \sum_{j=1}^{p} D_{\ell,j} \beta_j')^2 + \lambda \sum_{j=1}^{p} \|A_j^w \beta_j'\|. \quad (3.19)$$

Each $D_j$ can be factorized as $Q_jR_j$ by $QR$ decomposition, where $Q_j$ is an $nd \times d$ orthogonal matrix and $R_j$ is a $d \times d$ upper triangular matrix.

Define $A_j^w = R_j$, and let $\gamma_j = R_j \beta_j' = A_j^w \beta_j'$. Then (3.19) becomes

$$\sum_{\ell=1}^{nk} (Z_\ell - \sum_{j=1}^{p} Q_{\ell,j}R_j \gamma_j)^2 + \lambda \sum_{j=1}^{p} \|A_j^w \beta_j'\| = \sum_{\ell=1}^{nk} (Z_\ell - \sum_{j=1}^{p} Q_{\ell,j} \gamma_j)^2 + \lambda \sum_{j=1}^{p} \|\gamma_j\|. \quad (3.20)$$

As noted in Section 3.2.1, the columns of $Q$ are still centered. Since $X^w_j$ is centered, all columns of $D_j$ are centered, or equivalently, $1'_{nd}D_j = 0$. Since $D_j$ is decomposed as $Q_jR_j$, we have

$$1'_{nd}D_j = 1'_{nd}Q_jR_j = 0. \quad (3.21)$$

The $R_j$ is invertible because the columns of $D_j$ are orthogonal and therefore linearly independent. This implies that $1'_{nd}Q_j = 0$, which means $Q_j$ is still centered. Therefore, the pseudo-data $Z$ and $Q$ in (3.20) satisfy all the requirements for the original group Lasso. We can solve (3.20) using the group Lasso algorithm on $Z$ and $Q$ to estimate $\gamma_j$ for $j = 1, \cdots, p$.

Note that $A_j^w$ is different from the $A_j$ in (3.15). $A_j$ is an unknown matrix employed in the objective function (3.15) to enable us to use the group Lasso algorithm. We minimize $R$ in (3.15) by minimizing $R(w)$ at each iteration of the IRLS, by using $A_j^w$'s which vary with weights.

With pre-determined threshold values $c^*$ and $c^\ell$ in Huber’s loss function, the R Lasso 1 on the pseudo-data $Z$ and $Q$ runs the steps shown in Table 3.1.
Table 3.1: Algorithm of the R Lasso 1.

For each $\lambda$

$\beta^{(0)} = \beta^I$.

While $\text{diff} > 0.00001$, $k \geq 1$

IRLS

1. Compute the weights $w_i$, for $i = 1, \cdots, n$:
$$w_i = \rho\left(\frac{\|y_i - x_i\beta^{(k-1)}\|}{\|y_i - x_i\beta^{(k-1)}\|^2}\right).$$

2. Compute $Y^w_{n \times d}$ and $X^w_{n \times p}$.

3. Center each column of $Y^w$ and $X^w$.

4. Vectorize $Y^w$ into $Z_{nd \times 1}$, and adjust $X^w$ into $D_{nd \times pd}$.

5. QR decomposition on each $D_j$ for $j = 1, \cdots, p$.

6. Run the group Lasso algorithm on $Z$ and $Q$.

7. Compute $\text{diff} = \max_{j=1}^{p} \|\beta_j^{(k-1)}\| - \|\beta_j^{(k-1)}\|$.

} loop for IRLS.

Select $X_j$’s where $\|\beta_j\|$’s are larger than 0.

} loop for $\lambda$.

3.3.3 Simulation study for the R Lasso 1

To compare the performance of the R Lasso 1 with the original group Lasso, we conducted a small simulation study. We generated 100 Monte Carlo datasets with $n = 200$. For each Monte Carlo sample, we generated $X_1$ and $X_2$ independently from the Uniform distribution on $(0,1)$, and generated an $n \times 2$ response matrix $Y$ from the model

$$Y_i(t) = X_{1i}t + e_i(t),$$

for $t = 1$ and 2, where the independent error $e_i(t)$ is from $N(0, 100^2)$ for 5% of the observations $i = 1, \cdots, 10$, and from $N(0, 1)$ for the remaining 95% of the observations $i = 11, \cdots, 200$. Although the percentage of the large errors is only 5%, we downweighted 10% of the large residuals in the R Lasso 1. Among 100 Monte
Carlo data sets, the R Lasso 1 chose \( X_1 \) before it chose \( X_2 \) on 95 datasets, while the original group Lasso chose \( X_1 \) before \( X_2 \) on only 54 datasets. This suggests that the R Lasso 1 selects the true covariate better than the original Lasso when outliers are present.

### 3.4 Robust Lasso through a sequence of ridge regressions

Although the R Lasso 1 introduced in Section 3.3 has an advantage that we can directly apply the algorithm of Yuan and Lin (2006), it requires complicated transformations of the original data to fit the algorithm. The main restriction comes from the fact of the update formula (3.4) works only when the design matrix is orthonormal. Now we propose another algorithm for robust Lasso, which we would call R Lasso 2, through a sequence of ridge regressions. It enables us to compute the closed form of \( \hat{\beta}_j \) and runs much faster than the robust Lasso in Section 3.3, which needs additional iterations to update \( \beta_j \) in the original group Lasso algorithm. The R Lasso 2 requires fewer transformations - It requires the steps 1, 2 and 3 in Section 3.2.1 with step 2 modified to compute the robust estimate of \( \Sigma \) as described in Section 3.3. Each covariate needs to be scaled to have the standard deviation 1. We determine the threshold values of Huber’s loss function as explained in Section 3.3.1,

Let \( Y \) be an \( n \times d \) matrix of \( d \) response variables, and \( X = (X_1, \cdots, X_p) \) be an \( n \times p \) matrix of \( p \) covariates, where \( j^{th} \) covariate \( X_j \) be a column vector of length \( n \) with a standard deviation of 1. Let \( \alpha \) be a small number, say 0.0001, and \( P \) be the percentage of downweighting.

We aim to minimize the following objective function,

\[
R_0 = \sum_{i=1}^{n} \rho(\|y_i - x_i\hat{\beta}\|) + \lambda \sum_{j=1}^{p} \|\beta_j\|, \tag{3.23}
\]

where \( \beta = (\beta_1, \cdots, \beta_p)^{p \times d} \) is the coefficient matrix with \( j^{th} \) row defined by \( \beta_j \) for \( j = 1, \cdots, p \), and \( \rho(\cdot) \) is Huber’s loss function defined by

\[
\rho(r) = \begin{cases} 
  r^2 & \text{for } |r| \leq c \\
  2c|r| - c^2 & \text{for } |r| > c,
\end{cases} \tag{3.24}
\]

where a threshold value \( c \) to be determined by \( P \).

If \( c = \infty \) in Huber’s loss function, it is the LS Lasso that minimizes

\[
L = \sum_{i=1}^{n} \|y_i - x_i\beta\|^2 + \lambda \sum_{j=1}^{p} \|\beta_j\|^2 + \alpha, \tag{3.25}
\]
which is equivalent to the R Lasso 2 with $P = 0$.

The solution to (3.23) can be computed iteratively, by minimizing the following function at each step $k$:

$$R_1 = \sum_{i=1}^{n} \rho(\|y_i - x_i \beta\|) + \lambda \sum_{j=1}^{P} \frac{\|\beta_j\|^2}{\|\beta_j^{(k-1)}\| + \alpha},$$  

(3.26)

where $\beta_j^{(k-1)}$ is $\beta_j$ at iteration step $(k-1)$. The minimization of (3.26) will be carried out using the weighted least squares,

$$R_2 = \sum_{i=1}^{n} w_i \|y_i - x_i \beta\|^2 + \lambda \sum_{j=1}^{P} \frac{\|\beta_j\|^2}{\|\beta_j^{(k-1)}\| + \alpha},$$  

(3.27)

where weights $w_i$'s are given by

$$w_i = \rho \left( \frac{\|y_i - x_i \beta^{(k-1)}\|}{\|y_i - x_i \beta^{(k-1)}\|^2} \right).$$

By using the squared norm of $\beta_j$ in the penalty, we can obtain a closed form of $\hat{\beta}^{(k)}$, and improve the computational efficiency:

$$\hat{\beta}^{(k)} = (A_{p \times p} - B_{p 	imes p}^{-1})C_{p \times d},$$  

(3.28)

where

$$A = (a'_1, \cdots, a'_p)' \text{ with } a_j = \sum_{i=1}^{n} w_i x_i j x_i,$$

$$B = diag(b_1, \cdots, b_p) \text{ with } b_j = (\sum_{i=1}^{n} w_i x_i^2 j) \left( (\sum_{i=1}^{n} w_i x_i^2 j^2) + \frac{\lambda}{\|\beta_j^{(k-1)}\| + \alpha} \right), \text{ and}$$

$$C = (c'_1, \cdots, c'_p)' \text{ with } c_j = \sum_{i=1}^{n} w_i x_i j y_i.$$  

(3.29)

We set $\hat{\beta}_j^{(k)} = 0$ when the norm of $\hat{\beta}_j^{(k)}$ is less than $\alpha$. Note that each $X_j$ is scaled for the penalty function to be properly applied to each $\|\beta_j\|$, $j = 1, \cdots, p$. Let $X_j$ be the $j$th covariate in the model, and $s_j$ be the standard deviation of $X_j$ for $j = 1, \cdots, p$. Then

$$X \beta = [X_1, X_2, \cdots, X_p] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} = \begin{bmatrix} X_1 \\ s_1 \end{bmatrix} \frac{X_2}{s_2} \cdots \frac{X_p}{s_p} \begin{bmatrix} s_1 \beta_1 \\ s_2 \beta_2 \\ \vdots \\ s_p \beta_p \end{bmatrix}$$
So $\beta_j = 0$ is equivalent to $s_j \beta_j = 0$.

Table 3.2: Algorithm of the R Lasso 2.

Let $\alpha = 0.0001$. For each $\lambda$:

$\beta^{(0)} = \beta^I$.

While $\text{diff} > 0.00001$, $k \geq 1$ {

1. Compute the weights $w_i = \rho \left( \frac{\|y_i - x_i \beta^{(k-1)}_j\|}{\|y_i - x_i \beta^{(k-1)}_j\|} \right)$, for $i = 1, \cdots, n$.

2. Compute $\hat{\beta}^{(k)} = (A_{p \times p} - B_{p \times p}^{-1})C_{p \times d}$, for $j = 1, \cdots, p$ where

$A = (a_1', \cdots, a_p')'$ with $a_j = \sum_{i=1}^{n} w_i x_{ij} x_i$,

$B = \text{diag}(b_1, \cdots, b_p)$ with $b_j = \left( \sum_{i=1}^{n} w_i x_{ij}^2 \right) \left( \sum_{i=1}^{n} w_i x_{ij}^2 + \frac{\lambda}{\|\beta^{(k-1)}_j\| + \alpha} \right)$,

$C = (c_1', \cdots, c_p')'$ with $c_j = \sum_{i=1}^{n} w_i x_{ij} y_i$.

3. Compute $\text{diff} = \max_{j=1}^{p} \| \beta^{(k-1)}_j \| - \| \hat{\beta}^{(k-1)}_j \|$.

} loop for IRLS.

Select $X_j$’s where $\| \beta_j \|$’s are larger than $\alpha$.

} loop for $\lambda$.

3.4.1 Simulation study for the R Lasso 2

In this section, we illustrate how the R Lasso 2 performs compared with the LS Lasso, which is equivalent to the R Lasso with $P = 0$. Recall that the objective function to estimate $\beta$ at step $k$ in the R Lasso 2 is

$$O_1 = \sum_{i=1}^{n} \rho(\|y_i - x_i \beta\|) + \lambda \sum_{j=1}^{p} \frac{\|\beta^{(k-1)}_j\|^2}{\|\beta^{(k-1)}_j\| + \alpha},$$

(3.30)

where $\rho$ is the Huber’s loss function, and $\beta^{(k-1)}_j$ is the $\beta_j$ at step $(k - 1)$. We chose $\alpha = 0.0001$ in our simulation study. While $\rho$ in (3.30) downweights extreme values by using a linear loss function outside $(-c, c)$, the $L_2$ norm in (3.25) inflates large values of $r_i$ and hence has severe sensitivity with respect to outliers. We now illustrate the difference in the cases of one dimensional or two dimensional response variables with two covariates.
**Case I. One dimensional response variable**

Consider a model with a one-dimensional response variable and two covariates. A total of 100 Monte-Carlo samples are generated with the sample size for each dataset of 200. We generated \( X_{1i} \) and \( X_{2i} \) independently from the uniform distribution on \((0,1)\), and generated \( Y_i \) from the model

\[
Y_i = X_{1i} + e_i,
\]

where \( e_i \) is random noise from \( N(0, 100^2) \) for \( i = 1, \cdots , 10 \), and from \( N(0, 1) \) for \( i = 11, \cdots , 200 \). Therefore, 5% of the data points spread out, which would result in large residuals with respect to the model. Note that the response is associated only with \( X_1 \), and we expect Lasso algorithms to choose \( X_1 \) before \( X_2 \) enters as the tuning parameter \( \lambda \) decreases. When \( \lambda \) is large enough for Lasso to choose only one variable, Lasso would choose one between the following two models:

\[
\begin{align*}
Y &= X_1 \beta_1 + e \\
Y &= X_2 \beta_2 + e,
\end{align*}
\]

which gives smaller value of the objective functions when evaluated under that model. The objective functions of the LS Lasso and the R Lasso 2 differ only in their first terms, depending on whether large residuals are downweighted or not. As in Section 3.3.3, we downweighted \( P = 10\% \) of the large residuals in the R Lasso 2. Among 100 Monte Carlo data sets, R Lasso 2 chose \( X_1 \) before it chose \( X_2 \) on 92 datasets, while the LS Lasso chose \( X_1 \) before \( X_2 \) on only 53 datasets.

**Case II. Two dimensional response variable**

We now consider the case of a two-dimensional response variable, which means we have an \( n \times 2 \) response matrix. We generated 100 Monte Carlo datasets with \( n = 200 \) and with exactly the same \( X_1 \) and \( X_2 \) as in Case I. The response \( Y_{n \times 2} \) is generated from the following model:

\[
Y_{i}(t) = X_{1i}t + e_i(t),
\]

for \( t = 1 \) and 2, where the independent error \( e_i(t) \) is from \( N(0, 100^2) \) for 5% of the observations \( i = 1, \cdots , 10 \), and from \( N(0, 1) \) for the remaining 95% of the observations \( i = 11, \cdots , 200 \).

It shows 10 observations with large norms of the residuals. Again, we downweighted \( P = 10\% \) of the large residuals in the R Lasso 2, which amounts to 20 observations. Among 100 Monte Carlo data sets, R
Lasso 2 chose \( X_1 \) before it chose \( X_2 \) on 97 datasets, while LS Lasso chose \( X_1 \) before \( X_2 \) on only 50 datasets, similar to the performances shown in the one-dimensional case.

### 3.5 Application

We consider the gene expression dataset from Luan and Li (2003) with 292 cell-cycle-regulated genes identified based on the \( \alpha \)-factor synchronization cultures of Spellman et al. (1998), which covers two cell-cycle periods, with 18 time points measured at every 7 minutes for 119 minutes. The binding probabilities of 96 transcription factors (TFs) for these 292 genes from Wang et al. (2007) are used as covariates. The number of genes used in Wang et al. (2007) is 297, but we do not have data on 5 genes. After deleting genes containing missing values, we have 240 genes to work with. The names of the transcription factors and their numbers we used in the analysis are given in Table 3.3.

Table 3.3: The 96 transcription factors in \( X \). For convenience, we numbered them from 1 to 96.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABF1</td>
<td>ACE2</td>
<td>ADR1</td>
<td>ARG80</td>
<td>ARG81</td>
<td>ARO80</td>
<td>ASH1</td>
<td>BAS1</td>
<td>CAD1</td>
<td>CBF1</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>CIN5</td>
<td>CRZ1</td>
<td>CUP9</td>
<td>DAL81</td>
<td>DAL82</td>
<td>DIG1</td>
<td>DOT6</td>
<td>FHL1</td>
<td>FKH1</td>
<td>FKH2</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>FZF1</td>
<td>GAL4</td>
<td>GAT1</td>
<td>GAT3</td>
<td>GCN4</td>
<td>GCR1</td>
<td>GCR2</td>
<td>GLN3</td>
<td>GRF10.Pho2.</td>
<td>GTS1</td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>HAL9</td>
<td>HAP2</td>
<td>HAP3</td>
<td>HAP4</td>
<td>HAP5</td>
<td>HIR1</td>
<td>HIR2</td>
<td>HMS1</td>
<td>HSF1</td>
<td>IME4</td>
</tr>
<tr>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>INO2</td>
<td>INO4</td>
<td>IXR1</td>
<td>LEU3</td>
<td>MAC1</td>
<td>MAL13</td>
<td>MATa1</td>
<td>MBP1</td>
<td>MCM1</td>
<td>MET31</td>
</tr>
<tr>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>MET4</td>
<td>MIG1</td>
<td>MOT3</td>
<td>MSN1</td>
<td>MSN4</td>
<td>MSS11</td>
<td>MTH1</td>
<td>NDD1</td>
<td>NRG1</td>
<td>PDR1</td>
</tr>
<tr>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
</tr>
<tr>
<td>PHD1</td>
<td>PHO4</td>
<td>PUT3</td>
<td>RAP1</td>
<td>RCS1</td>
<td>REB1</td>
<td>RFX1</td>
<td>RGM1</td>
<td>RLM1</td>
<td>RME1</td>
</tr>
<tr>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>ROX1</td>
<td>RPH1</td>
<td>RTG1</td>
<td>RTG3</td>
<td>SFP1</td>
<td>SIG1</td>
<td>SIP4</td>
<td>SKN7</td>
<td>SMP1</td>
<td>SOK2</td>
</tr>
<tr>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
</tr>
<tr>
<td>SRD1</td>
<td>STB1</td>
<td>STE12</td>
<td>STP1</td>
<td>STP2</td>
<td>SUM1</td>
<td>SWI4</td>
<td>SWI5</td>
<td>SWI6</td>
<td>YAP1</td>
</tr>
<tr>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>YAP5</td>
<td>YAP6</td>
<td>YFL044C</td>
<td>YJL206C</td>
<td>ZAP1</td>
<td>ZMS1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We aim to identify the TFs that play important roles in regulating yeast cell-cycle gene expressions. We assumed the following model to link the binding probabilities to the gene expression levels.

\[
Y_i(t) = \beta_0(t) + \sum_{j=1}^{96} \beta_j(t)X_{ij} + \varepsilon_i(t),
\]  

(3.34)
where $\beta_j(t)$ expresses the effect of the $j^{th}$ TF on the gene expression at time $t = 1, \cdots, 18$. We assumed that $Y_i(t)$, $i = 1, \cdots, 240$ are independent at the same $t$, and that each $\varepsilon_i = (\varepsilon_i(1), \cdots, \varepsilon_i(18))$ has mean 0 and variance-covariance matrix $\Sigma$. The datasets are preprocessed as previously described. We reduced the number of time points from $T = 18$ to $d = 8$ using the cubic B-splines with 8 degrees of freedom, by putting 5 internal equally spaced knots between 1 and 18. We chose $P = 20\%$ for the R Lasso 1 and 2.

### 3.5.1 Transcription Factors found in the previous literatures

- Wang et al. (2007) identified 71 out of 96 TFs related to the yeast cell-cycle processes, including 19 of the 21 known and experimentally verified cell-cycle-related TFs. These known TFs are numbered as 1, 2, 8, 10, 19, 20, 25, 26, 27, 44, 48, 49, 50, 58, 66, 78, 82, 83, 87, 88 and 89. TFs 10 and 25 were not selected by Wang et al. (2008). The 52 additional TFs selected by Wang et al. (2008) are 5, 6, 7, 11, 12, 13, 14, 17, 18, 21, 23, 24, 29, 30, 31, 32, 33, 34, 35, 37, 38, 39, 40, 41, 45, 46, 47, 51, 52, 53, 55, 57, 59, 61, 63, 67, 68, 69, 71, 73, 74, 75, 76, 77, 79, 80, 81, 84, 85, 91, 92 and 94.

- Wu et al. (2006) identified 40 TFs that regulated genes of the yeast cell cycle, including 30 TFs which are known to be involved in the cell cycle, and 10 putative cell cycle related TFs. The 30 known TFs are numbered as 1, 2, 11, 16, 19, 20, 22, 24, 34, 36, 37, 43, 48, 49, 55, 58, 64, 69, 78, 82, 83, 87, 88, 89, 91 and Cts6, HIR3, Tec1, Ume6 and Yox1. The remaining 10 TFs are 58, 60, 61, 62, 66, 79, 92 and Dat1, Hap1 and Sut1.

### 3.5.2 Results

In general, both methods select TFs in the similar orders. However, we found some interesting differences. The R Lasso 1 and the R Lasso 2 are similarly behaved in general. Nevertheless, we noticed some differences between the R Lasso 1 and the R lasso 2, as well as differences between those two R lasso methods and the LS Lasso. Table 3.4 shows five transcription factors selected at significantly different orders by three Lasso methods. The R Lasso 1 and the R lasso 2 selected TF 9, 37, 49, and 51 at similar orders, while the LS Lasso chose them much earlier or later. The only difference between the R Lasso 1 and 2 is that the TF 36 is the first TF selected by the R Lasso 1 and the LS Lasso, while it is the 10th in the R Lasso 2. Although the difference is not as much as that between the R Lassos and the LS Lasso, but it indicates that the results from any group Lasso type of methods depend somewhat on the choice of the penalties used.

We favor the R lasso 2 over the R Lasso 1, because the R Lasso 2 is computationally simpler and runs much faster than the other, yet the two methods showed similar performance in the simulation study. Although the R Lasso 1 has an advantage that we can directly apply the original group Lasso algorithm, it
requires complicated transformations of the original data to fit the algorithm, since the update formula (3.4) in the original Lasso works only when the design matrix is orthonormal. In the next sections, we report further analysis using the R Lasso 2.

Table 3.4: The transcription factors selected by R Lasso 1, R Lasso 2 and LS Lasso. The numbers represent the order at which the transcription factor is selected.

<table>
<thead>
<tr>
<th></th>
<th>TF 9</th>
<th>TF 36</th>
<th>TF 37</th>
<th>TF 49</th>
<th>TF 51</th>
</tr>
</thead>
<tbody>
<tr>
<td>R Lasso 1</td>
<td>58th</td>
<td>1st</td>
<td>10th</td>
<td>9th</td>
<td>18th</td>
</tr>
<tr>
<td>R Lasso 2</td>
<td>56th</td>
<td>10th</td>
<td>12nd</td>
<td>6th</td>
<td>18th</td>
</tr>
<tr>
<td>LS Lasso</td>
<td>90th</td>
<td>1st</td>
<td>2nd</td>
<td>21st</td>
<td>8th</td>
</tr>
</tbody>
</table>

Sensitivity to outliers

Now we take a closer look at the first 71 TFs among 96 TFs. The R Lasso 2 and the LS Lasso select 71 TFs at $\lambda = 410$ and $\lambda = 395$, respectively. The TFs selected only by the R Lasso 2 but not by the LS Lasso are TFs 9, 16, 62, 75, 83 and 95, and those selected only by the LS Lasso are TFs 14, 40, 57, 64, 74 and 76. The difference seems to be within the normal variability between any two different methods, not necessarily due to the issues of robustness. We illustrate how the two methods would differ when outliers are introduced.

There are 77 TFs which are selected either by the R Lasso 2 or the LS Lasso. We computed the Mahalanobis distances of all 240 genes based on the binding scores of these 77 TFs relative to their mean to measure leverages. The distances range over (2.02, 15.42), with median at 7.531, as shown in Figure 3.1(a). We found that gene 50 has the smallest leverage, gene 48 has leverage around the 40th percentile, genes 23 and 45 has median leverages, and gene 63 has the largest leverages. The black circles in Figure 3.1(b) indicate these genes. The gene expressions of all 240 genes range over (-2.71, 2.53). If we replace the whole gene expression for each of these genes by outlying values, the R Lasso 2 does not perform very differently from the LS Lasso. For example, if we replace all the 18 measurements of each gene by numbers generated from the Uniform distribution on (3,4), then we observe the following.

- **Gene 50 (smallest leverage)**: Both Lasso results change significantly: the R Lasso 2 selects 25 TFs with a new TF 24 added, and the LS Lasso selects 24 TFs dropped from 71. Similar results were observed for Gene 45.

- **Gene 63 (largest leverage)**: The R Lasso 2 selects 71 TFs, which is the same number of TFs selected previously, but with a slight difference - TFs 3, 33, 44, 57 and 95 were newly selected, and the TFs 16, 42, 75, 83 and 95 were removed from the previous group of the 71 TFs. LS Lasso selects 70 TFs with TFs 3 and 75 newly selected, and TFs 15, 40 and 73 removed.
• Gene 23 (median leverage) : We see a difference here. The R Lasso 2 selects 23 TFs with a new TF 64 selected additionally, while the LS Lasso selects 12 TFs with no new TF.

In summary, we concluded that R Lasso 2 does not necessarily perform better than the LS Lasso when we replace all the measurements of a gene. However, if part of a gene is contaminated, then the R Lasso 2 and the LS Lasso start to differ. We replaced the first half (9 measurements) of the genes 50, 48, 23, 45 and 63 by values generated from the Uniform distribution on (3, 4). The results are given below :

• Gene 50 (smallest leverage) : The R Lasso 2 selects 42 TFs with new TFs 8 and 64, while the LS Lasso selects 23 TFs with new TF 8.

• Gene 48 (leverage at 40th percentile) : The R Lasso 2 selects 47 TFs with TF 64 added, while the LS Lasso selects only 12 TFs with no new TF added.

• Gene 23 (median leverage) : The R Lasso 2 selects 51 TFs with TF 64 newly selected, while the LS Lasso selects only 12 TFs with no new TF.

• Gene 45 (median leverage) : The R Lasso 2 selects 52 TFs with TF 64 newly selected, while the LS Lasso selects only 21 TFs with no new TF.

• Gene 63 (largest leverage) : The R Lasso 2 selects 74 TFs including all previously selected 71 TFs and with three new TFs 3, 57 and 93. The LS Lasso selects 66 TFs with TFs 3, 26 and 75 newly selected, and 8 TFs removed from the previous list of the 71 TFs. The outliers introduced in gene 63 seem to have little effect in this case.

As shown above, the R Lasso 2 is less sensitive to outliers than the LS Lasso. Moreover, we see that the difference is much bigger when we change the genes with small to median leverages.

3.5.3 Revisiting the HS and LS tests

In this section, we report the results from the backward selection using the HS and the LS tests in Chapter 1 for the new Yeast cell cycle data. As in Chapter 1, we deleted the least significant TF at each step until all TFs remaining in the model are significant under the false discovery rate (FDR) controlled at 0.05. As a result, the HS backward selection found 12 significant TFs, which are TF 16, 18, 19, 22, 37, 48, 58, 78, 80, 82, 87 and 88. The LS backward selection found 13 TFs, numbered as 2, 19, 37, 48, 58, 60, 61, 78, 79, 80, 83, 87 and 88. Note that the TFs 16, 18, 22 and 82 are selected only by HS backward tests, while the TFs 2, 60, 61, 79 and 83 are selected only by the LS backward tests.
Figure 3.1: The distribution of the Mahalanobis distances of 240 genes (a) and the plot of Genes vs. Distances (b). Black circles represent the genes selected for further investigation.

Comparing the results from the tests with the Lasso methods, we found that HS tests and the R Lasso 2 differ a lot. The results from the LS tests and the LS Lasso were not similar, either. The first 12 TFs selected by the R Lasso 2 are TFs 20, 24, 36, 37, 48, 49, 58, 60, 68, 87, 89 and 91. We see that only 4 TFs 37, 48, 58, 87 are selected both by the HS tests and the R Lasso 2. Likewise, the first 13 TFs selected by the LS Lasso are TFs 20, 24, 36, 37, 48, 51, 58, 60, 68, 79, 87, 89 and 91. Only 6 TFs 37, 48, 58, 60, 79 and 87 are selected both by the LS tests and the LS Lasso.

This is not surprising since the tests are looking for conditional associations between covariates and the gene expressions, while the Lasso methods select any variable that contributes to the gene expressions. In other words, a variable in the final list of the variables after backward tests could represent a group of variables with high correlations. Both approaches may be useful for what they intend to do.

3.6 Conclusion

We have proposed two robust group Lasso methods, the R Lasso 1 and the R Lasso 2. The objective functions for the R Lasso 1 and 2 involve Huber’s loss function, in order to make the selection robust against departures from the normality assumption. Although the R Lasso 1 has an advantage that we can directly apply the original group Lasso algorithm, it requires complicated transformations of the original data to fit the algorithm, because the original group Lasso works only when the design matrix is orthogonal. We favor
the R Lasso 2 over the R Lasso 1 because the R Lasso 2 is computationally simpler and runs much faster than the other, while two methods showed similar performance.

We have shown in our application that the R Lasso 2 is less sensitive than the LS Lasso, when we introduce outliers to the original data. However, as in Chapter 1, it is worth noting that the robust group Lasso methods do not replace the LS Lasso. Their performance depends on the percentage of downweighting, and excessive downweighting can make them less efficient than the LS Lasso. The robust group Lasso methods can complement the least squares-based method and provide more options for more informative conclusions.


Vita

Ji Young Kim graduated from the Korea University in Seoul, Korea, with the B.S. in Mathematics. She obtained the M.S. in Actuarial Science and the M.S. in Statistics at the University of Illinois at Urbana-Champaign. Upon receiving the Ph.D of Statistics in August 2010, she will begin her career at the Mount Holyoke College in Massachusetts, as an Assistant Professor of Statistics, in the department of Mathematics and Statistics.