Robust mixture regression using mean-shift penalisation

by

Anika Wessels

Submitted in fulfillment of the requirements for the degree
Master of Science (Advanced Data Analytics)
In the Faculty of Natural and Agricultural Sciences
University of Pretoria
Pretoria

November 2021

Robust mixture regression using mean-shift penalisation

by

Anika Wessels
E-mail: anikawessels92@gmail.com

Abstract

The purpose of finite mixture regression (FMR) is to model the relationship between a response and feature variables in the presence of latent groups in the population. The different regression structures are quantified by the unique parameters of each latent group. The Gaussian mixture regression model is a method commonly used in FMR since it simplifies the estimation and interpretation of the model output. However, it is highly affected if outliers are present in the data. Failing to account for the outliers may distort the results and lead to inappropriate conclusions. We consider a mean-shift robust mixture regression approach to address this. This method uses a component specific mean-shift parameterisation which contributes towards both the successful identification of outliers as well as robust parameter estimation. The technique is demonstrated by a simulation study and a real-world application. The mean-shift regression method proves to be highly robust against outliers.

Keywords: Mixture regression models, penalised likelihood, robust parameter estimation, outlier detection.

Supervisors : Dr. F.J. Kanfer
Dr. S.M. Millard
Department : Department of Statistics
Degree : Master of Science Advanced Data Analytics

Declaration

I, Anika Wessels, declare that this dissertation, which I hereby submit for the degree MSc Advanced Data Analytics at the University of Pretoria, is my own work and has not previously been submitted by me for a degree at this or any other tertiary institution.

SIGNATURE:

DATE: 2021

Acknowledgements

I wish to extend a huge appreciation and thank you to the following people and institutions:

- To my manager, Darryl Lahner, you have been extremely supportive during the three years that I've been studying towards this, and here it is, the final piece of my masters degree.
- To my employer, First National Bank. Thank you for the financial support and believing that I can accomplish this part time.
- To my colleague, Dewald Beket. You have been my shoulder to cry on when things went bad, nevertheless I persisted. I appreciate all your support.
- To my supervisors, Dr. Frans Kanfer and Dr. Sollie Millard. Thank you for all the guidance, advice, inspiration and motivation. My Tuesdays are going to feel incomplete without our meeting.
- To Renate Thiede, thank you for taking the time to provide suggestions and corrections to this mini-dissertation.
- To my study buddy, Charl Arthur Henry Cowley. This year marks a decade of knowing you. Thank you for this journey, I would not have achieved this without you.
- To my parents and siblings, I dedicate this mini-dissertation to you. You have always been my motivation and support. Words cannot describe my appreciation for all you have done for me throughout my life.
- Lastly, to my charming husband, Werner. Thank you for being my rock and for holding my hand every step of the way. You are awesome.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

1.1 Motivation

Mixture regression models, made popular by Goldfeld and Quandt (1973), are used to capture unobserved heterogeneity in the effects of the feature variables on the response. A mixture model consists of a combination of probability density functions (pdf) or mass functions (pmf). If it is suspected that the data can be clustered into different latent groups, each group can be modelled with its own regression component. Latent groups are unobservable groupings of data points with similar group membership. Each group shares a meaningful pattern of responses on the measure of interest. To "identify" these groups, we use the observed data to obtain the probability that an observation belongs to a group. In mixture regression we also refer to these groups as components. Suppose we have m latent groups. We can model these groups with m linear regression components, which is refered to as a finite mixture regression model. If $(y, \mathbf{x})$ belongs to the j^{th} component with j=*1, 2, ..., m* and $y = \mathbf{x}^T\boldsymbol{\beta}_j + \epsilon_j$, then the conditional density of y given $\mathbf{x}$ is

$$f(y \mid \mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^{m} \pi_j \phi(y; \mathbf{x}^T\boldsymbol{\beta}_j, {\sigma_j}^2),$$

where

$\boldsymbol{\beta}_j$ is a vector of dimension p and contains the regression coefficients of component j,

$\epsilon_j \sim N(0, {\sigma_j}^2)$ with ${\sigma_j}^2 > 0$, is a random error term of component j,

$\phi(.;\mu,\sigma^2)$ indicates a Gaussian pdf with mean μ and variance σ^2,
π_j is the mixing probability of component j and
$\boldsymbol{\theta} = (\pi_1,\ \beta_1,\ \sigma_1,\ \ldots\ \pi_m,\ \beta_m,\ \sigma_m)$ is the collection of unknown parameters.

In the presence of latent variables, the maximum likelihood estimates of the parameters are determined by the Expectation Maximisation (EM) algorithm (Dempster et al. (1977)). This is imperative as no closed form expression exists to estimate the unknown parameters. The formula to estimate $\boldsymbol{\theta}$ by using maximum likelihood estimation (MLE) is

$$\hat{\boldsymbol{\theta}}_{mle} = \underset{\theta}{argmax} \sum_{i=1}^{n} \log \left\{ \sum_{j=1}^{m} \pi_j \phi(y_i;\ \mathbf{x}_i{}^T \boldsymbol{\beta}_j, \sigma_j{}^2) \right\}.$$

As a result of the normality-assumption based MLE, which is extremely sensitive to outliers, there is a clear need to re-assess the estimation process. An outlier is defined as an observation with an unusual response value y. In other words, the observation falls outside the general trend of the other response values. Outliers in the sample lead to inaccurate parameter estimation and incorrect conclusions. An observation that differs from the others in the $\mathbf{x}$ direction, i.e. in the predictor variables, is called a high leverage point. Based on the above, a robust method to estimate the parameters needs to be explored.

Robust regression can overcome the limitations associated with parametric and non-parametric regression methods in the presence of outliers. Robust estimation refers to the "less affected" estimation results if the data consist of outliers. The robustness of an estimator can be determined by the break down point (BDP). The BDP determines the proportion of "unusual observations" (or outliers) which the estimation process can accommodate, otherwise it will produce incorrect estimation results. Stated differently, it measures the smallest proportion of outliers in the sample that causes the estimator to break down or become less useful. A higher BDP indicates a more robust estimator. In Section 2.4 we discuss different model performance measures that will be used in the application.

Many robust mixture regression methods have been developed since the late 1990s. Gershenfeld (1997) introduced the cluster-weighted model (CWM), which jointly models the random covariates with the response variable. Markatou (2000) and Shen et al. (2004)

developed robust methods which uses weighted likelihood to create a robust estimation procedure. Neykov et al. (2007) proposed a trimmed likelihood estimator (TLE) to achieve robust mixture estimates. García-Escudero et al. (2010) presented a robust clusterwise linear regression method to trim high leverage points through a process of "second" trimming. Bai et al. (2012) adopted a robust rule in the maximisation step of the traditional EM algorithm. Bashir and Carter (2012) introduced a new class of robust estimators using S-estimators. These have the same flexibility and asymptotic properties as that of M-estimators, of which maximum likelihood estimation is a special case.

Both Song et al. (2014) and Yao et al. (2014) proposed to use a heavy-tailed distribution to model the error density. Song et al. (2014) used the Laplace distribution and Yao et al. (2014) the t-distribution. In addition, Yao et al. (2014) further proposed a trimming of the t-distribution.

Zeller et al. (2016) used a scale mixture of skew-normal distributions which accommodates asymmetry and heavy tails. Hu et al. (2016, 2017) developed a robust EM algorithm for log-concave mixture regression models. García-Escudero et al. (2017) achieved robust mixture regression estimation with random covariates using both constraints and trimming. Punzo and McNicholas (2017) proposed robust clustering in the regression contex using a contaminated Gaussian cluster weighted approach.

The aim is to explore some of these robust regression methods contained in the literature. The main focus of this mini-dissertation is the robust mixture regression using mean-shift penalisation (RM^2) approach by Yu et al. (2017). This method can detect outliers and yields robust parameter estimation, thereby allowing for flexibility to the previous model by Yu et al. (2015).

Some challenges arise when using regularisation methods under the general structure of mixture regression models. In short, regularisation refers to the tuning of the objective function by adding a penalty term which controls for severe fluctuation and in turn prevents extreme parameter estimates. The RM^2 method accounts for this. Computation becomes complex when maximising the mixture likelihood since it is a non-convex problem. To compensate for the non-convexity of the penalised mixture likelihood, a thresholding-embedded EM algorithm is used. If observations have different outlying

effects across the mixture components, it tends to misconstrued the conception. In this case the definition of an outlier becomes unclear. The mean-shift penalisation approach also addresses the unequal variances in the mixture components. Each observation is free to have different outlying effects across the components. The mean-shift parameterisation creates sparsity and is scale dependent. This enable us to follow the "simple normal mixture regression model" assumption.

We consider the mean-shift parameters for each observation in each mixture component in the RM^2 method. The crux of this approach is underlined by the sparsity structures created by the mean-shift parameters. This assist in identifying and accommodating the outliers. The interpretation of a mean-shift parameter is the number of standard deviations the outlying observation (outlier) is shifted from the mean structure of the regression component. We validate the findings of the RM^2 approach by comparing it to the outcome of other robust mixture regression methods. This method proves to be highly robust when outliers are present, both in the simulation study and in a real-world application.

1.2 Objectives

The research objectives are outlined below:

- To investigate the purpose of the mean-shift parameters,
- compare the RM^2 method to other robust mixture regression methods,
- apply the RM^2 method to a data set, and to
- determine and discuss the performance of the RM^2 method.

1.3 Outline

The structure of this mini-dissertation is as follows:

- **Chapter 2** focusses on documenting the robust mixture regression method using mean-shift penalisation. We give a comprehensive description of the model and show how it performs if applied to a simulated data set.
- **Chapter 3** covers a simulation study where we compare the performance of the RM^2 method to other robust mixture regression methods.
- **Chapter 4** illustrates the performance of the RM^2 method in a real-world regression problem.
- **Chapter 5** provides a conclusion on the findings.
- **Appendix A** lists the acronyms used in this mini-dissertation and the associated definitions.
- **Appendix B** provides extracts of the code used in the real-world application.

Chapter 2

Robust Mixture Regression using Mean-Shift Penalisation (RM^2)

2.1 Introduction

This chapter considers the RM^2 model proposed by Yu et al. (2017). Two prominent features make this approach appealing: 1) the method assumes a normal mixture regression model to simplify computation and interpretation, and 2) the mean-shift parameters create different outlying effects for each observation across the components. In Section 2.2 and Section 2.3, we formulate the regression model in the Gaussian mixture setting and for the RM^2 method, respectively. We explain the thresholding-embedded EM algorithm for this method and derive the equations to be used to estimate the unknown parameters. Model performance measures are given in Section 2.4. In Section 2.5 we demonstrate this approach on simulated examples and provide a brief overview of two other robust regression methods.

2.2 Gaussian mixture regression

2.2.1 Model formulation

Assume we have a random vector $\boldsymbol{y}$ and let

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} \mathbf{x}_i^T\boldsymbol{\beta}_1 + \epsilon_{i1} & \text{with probability} & \pi_1 \\ \mathbf{x}_i^T\boldsymbol{\beta}_2 + \epsilon_{i2} & \text{with probability} & \pi_2 \\ \vdots & & \vdots \\ \mathbf{x}_i^T\boldsymbol{\beta}_j + \epsilon_{ij} & \text{with probability} & \pi_j, \end{pmatrix}$$

where
y_i the i^{th} observation of the response ($i = 1, 2, ..., n$),
$\mathbf{x}_i^T$ a vector of dimension p of feature variables,
$\boldsymbol{\beta}_j$ a vector of dimension p with regression coefficients of component j ($j = 1, 2, ..., m$),
π_j the mixing probability of component j and
$\epsilon_{ij} \sim N(0, {\sigma_j}^2)$ random error terms.
Note that $\mathbf{x} = \begin{pmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix}$ is an $n \times p$ matrix.

If $y_i \sim N(\mathbf{x}_i^T\boldsymbol{\beta}_j, {\sigma_j}^2)$ we have a Gaussian mixture regression model. The mixture distribution of $\boldsymbol{y}$ is

$$f(\boldsymbol{y}) = f(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^{m} \pi_j \phi(\boldsymbol{y}; \mathbf{x}^T\boldsymbol{\beta}_j, {\sigma_j}^2), \tag{2.1}$$

where π_j is the j^{th} mixing proportion, $\mathbf{x}^T\boldsymbol{\beta}_j$ is the conditional mean, ${\sigma_j}^2$ is the variance of component j, $\phi(.)$ indicates a Gaussian pdf and $\boldsymbol{\theta} = (\pi_1,\ \beta_1,\ \sigma_1,\ ...\ \pi_m,\ \beta_m,\ \sigma_m)$ is the collection of unknown parameters.

The likelihood function is constructed from the joint distribution as a function of the parameters only. This means that we fix the random variables at its observed values. The likelihood determines how good the model fits the data for given values of the unknown parameters. We are interested in the combination of model parameter values

which maximises the likelihood function. If the samples $Y_1, Y_2, ..., Y_n$ are independent, then the likelihood function is

$$L(\boldsymbol{\theta}|\mathbf{y}) = \prod_{i=1}^{n} f_{\boldsymbol{Y}}(y_i) = \prod_{i=1}^{n}\sum_{j=1}^{m} \pi_j \phi(y_i; \mathbf{x}_i{}^T\boldsymbol{\beta}_j, \sigma_j{}^2).$$

Instead of using the likelihood, we use the log-likelihood as it changes the product of probabilities into the sum of probabilities in the computation process. This is possible because the natural logarithm is a monotonically increasing concave function, hence the location of the maximum does not change. The log-likelihood is

$$l(\boldsymbol{\theta}|\mathbf{y}) = log \prod_{i=1}^{n} \left(\sum_{j=1}^{m} \pi_j \phi(y_i; \mathbf{x}_i{}^T\boldsymbol{\beta}_j, \sigma_j{}^2) \right) = \sum_{i=1}^{n} log \left(\sum_{j=1}^{m} \pi_j \phi(y_i; \mathbf{x}_i{}^T\boldsymbol{\beta}_j, \sigma_j{}^2) \right).$$

Let $\boldsymbol{z} = (z_{ij})$ be the collection of binary latent variables such that

$$z_{ij} = \begin{cases} 1 & \text{if observation } i \text{ is from component } j \\ 0 & \text{otherwise,} \end{cases} \tag{2.2}$$

and $P(z_{ij} = 1|\boldsymbol{\theta}) = \pi_j$. Each observation has only one z_{ij}, i.e. each observation belongs to only one component. The complete data are $(\boldsymbol{y}, \mathbf{x}, \boldsymbol{z})$. The joint distribution of $\boldsymbol{y}$ and $\boldsymbol{z}$ is

$$P(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) = P(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{\theta})P(\boldsymbol{z}|\boldsymbol{\theta}) = \prod_{i=1}^{n} P(y_i|\boldsymbol{z}_i, \boldsymbol{\theta})P(\boldsymbol{z}_i|\boldsymbol{\theta})$$

which, since $P(y_i|\boldsymbol{z}_i, \boldsymbol{\theta}) = f(y_i)$, yields

$$P(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{\theta}) = \prod_{i=1}^{n}\prod_{j=1}^{m} [f(y_i)\ \pi_j]^{z_{ij}}.$$

It follows that the complete data log-likelihood is

$$l_c(\boldsymbol{\theta}) = \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij} log\{\pi_j \phi(y_i; \mathbf{x}_i{}^T\boldsymbol{\beta}_j, \sigma_j{}^2)\}.$$

2.2.2 Model estimation

One common method to estimate the parameters is maximum likelihood estimation using the EM algorithm. A detailed discussion of the EM algorithm is given in Section 2.3.2 for the RM^2 model outlined in Section 2.3.1.

2.3 RM^2 model

2.3.1 Model formulation

Yu et al. (2017) proposed a robust mixture regression model given by

$$f(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}, \boldsymbol{\gamma}_i) = \sum_{j=1}^{m} \pi_j \phi(y_i;\ \mathbf{x}_i{}^T \boldsymbol{\beta}_j + \gamma_{ij}\sigma_j, \sigma_j{}^2) \qquad \text{for } i = 1, 2, ..., n, \tag{2.3}$$

where $\boldsymbol{\theta} = (\pi_1,\ \beta_1,\ \sigma_1,\ \ldots\ \pi_m,\ \beta_m,\ \sigma_m)$ is the unknown parameter vector, $\boldsymbol{\gamma}_i = (\gamma_{i1}\ \ \gamma_{i2}\ \ \ldots\ \ \gamma_{im})^T$ is a vector of mean-shift parameters for observation i, m is the number of regression components and $\phi(\,.\,)$ indicates a Gaussian pdf. The model in (2.3) is known as a mean-shifted normal mixture regression model, because each observation is assigned a mean-shift parameter γ_{ij}. This parameter is added to the mean structure for each observation in each mixture component. The magnitude of this shift is proprotional to the scale of the regression component.

It is clear that without any contraints imposed on these new parameters $\boldsymbol{\gamma}$, the model is over-parameterised. The crux of the model is in the structure of the mean-shift parameters. For non-outlying observations the mean-shift parameters are zero, with only a few non-zero mean-shift parameters for each of the outliers, i.e. a sparse structure. By introducing sparsity in the mean-shift parameters, we can identify and accommodate outliers in the model. The outlying effect of observation i to component j is modelled by $\gamma_{ij}\sigma_j$. This model achieves robust estimation by making use of well-known penalised estimation approaches, whilst retaining the simplicity of a normal mixture model.

So then, in summising that $\boldsymbol{\Gamma} = (\boldsymbol{\gamma}_1{}^T\ \ \boldsymbol{\gamma}_2{}^T\ \ \ldots\ \ \boldsymbol{\gamma}_n{}^T)^T$ is the collection of mean-shift parameters, the penalised log-likelihood to estimate the unknown parameters in (2.3) is given by

$$(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\Gamma}}) = \underset{\boldsymbol{\theta}, \boldsymbol{\Gamma}}{argmax}\ J_n(\boldsymbol{\theta}, \boldsymbol{\Gamma})$$

where

$$J_n(\boldsymbol{\theta}, \boldsymbol{\Gamma}) = l_n(\boldsymbol{\theta}, \boldsymbol{\Gamma}) - \sum_{i=1}^{n} P_\lambda(\boldsymbol{\gamma}_i) \tag{2.4}$$

and

$$l_n(\boldsymbol{\theta}, \boldsymbol{\Gamma}) = \sum_{i=1}^{n} log \left\{ \sum_{j=1}^{m} \pi_j \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2) \right\}. \tag{2.5}$$

Similar to the Gaussian mixture regression model formulation, the log-likelihood in (2.5) is obtained by taking the log of the likelihood using the product rule for logarithms, $log(\prod(\,.\,)) = \sum log(\,.\,)$. The argument of the penalty function, $P_\lambda(\boldsymbol{\gamma}_i)$ in (2.4), is a vector with a tuning parameter λ which controls the degree of penalisation. The penalty function addressess the over-parameterisation by penalising the optimisation function, i.e. the log-likelihood.

The penalty function can be chosen to produce *vector-wise* sparsity or *element-wise* sparsity. *Vector-wise* sparsity promotes the entire vector, $\boldsymbol{\gamma}_i$, to be zero (across mixture components). The group lasso penalty, $P_\lambda(\boldsymbol{\gamma}_i) = \lambda\|\boldsymbol{\gamma}_i\|_q$, and the group l_0 penalty, $P_\lambda(\boldsymbol{\gamma}_i) = \frac{\lambda^2}{2} I(\|\boldsymbol{\gamma}_i\|_q \neq 0)$, are examples of penalty functions that produce *vector-wise* sparsity. $I(.)$ represents the indicator function and if $q = 2$, it indicates that the l_2 norm is penalised. In the course of this work, we focus on *element-wise* sparsity. To achieve this, we set $P_\lambda(\boldsymbol{\gamma}_i) = \sum_{j=1}^{m} P_\lambda(|\boldsymbol{\gamma}_{ij}|)$ in (2.4). Examples of *element-wise* sparsity are the l_1 norm penalty

$$P_\lambda(\boldsymbol{\gamma}_i) = \lambda \sum_{j=1}^{m} |\boldsymbol{\gamma}_{ij}| \tag{2.6}$$

and the l_0 norm penalty

$$P_\lambda(\boldsymbol{\gamma}_i) = \frac{\lambda^2}{2} \sum_{j=1}^{m} I(\boldsymbol{\gamma}_{ij} \neq 0). \tag{2.7}$$

Other choices of penalty functions are the SCAD penalty (Fan and Li (2001)) and the MCP (Zhang (2010)).

The penalised log-likelihood in (2.4) is unbounded as in the case of the Gaussian mixture model in (2.1). This means that the penalised log-likelihood tends to infinity when $y_i = \mathbf{x}_i{}^T \boldsymbol{\beta}_j + \gamma_{ij}\sigma_j$ and $\sigma_j \to 0$. To avoid this, we restrict the scale parameters $(\sigma_1, \sigma_2, ..., \sigma_m) \in \Omega_\sigma$, where Ω_σ is given by

$$\begin{aligned} \Omega_\sigma = \{(\sigma_1, \sigma_2, ..., \sigma_m) : \sigma_j > 0 \ for\ 1 \leq j \leq m; \\ \sigma_j/\sigma_l \geq \epsilon \ \ for\ j \neq l\ and\ 1 \leq j,\ l \leq m\}, \end{aligned} \tag{2.8}$$

and $\epsilon > 0$ is a small value (see Hathaway (1985)). The parameter space of $\boldsymbol{\theta}$ is therefore defined as

$$\Omega = \{(\pi_j, \beta_j, \sigma_j), j = 1, 2, ..., m : 0 \leq \pi_j \leq 1, \sum_{j=1}^{m} \pi_j = 1, (\sigma_1, \sigma_2, ..., \sigma_m) \in \Omega_\sigma\}. \quad (2.9)$$

2.3.2 Model estimation

The modelling problem involves estimating the conditional density of y given the predictor variables x, for a given data set. Density estimation can be performed using MLE given that the relevant interacting random variables are present, i.e. we have a complete data set. Simply put, a complete data set contains all the variables relevant to the problem. The EM algorithm is a common maximisation approach to MLE when there are latent variables present. It first estimates the values for the latent variables in the E-step, and then optimises the model in the M-step. These two steps are repeated until convergence, i.e. until a stopping criterion is satisfied.

The EM algorithm for the RM^2 model by Yu et al. (2017) is a thresholding-embedded EM algorithm which maximises the penalised log-likelihood in (2.4). Consider again

$$(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\Gamma}}) = \underset{\boldsymbol{\theta} \in \Omega, \boldsymbol{\Gamma}}{argmax} \left\{ \sum_{i=1}^{n} log \left\{ \sum_{j=1}^{m} \pi_j \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2) \right\} - \sum_{i=1}^{n} \sum_{j=1}^{m} P_\lambda(|\boldsymbol{\gamma}_{ij}|) \right\} \quad (2.10)$$

where $P_\lambda(\,.\,)$ can be chosen as the l_1 or l_0 penalty function in (2.6) and (2.7), respectively. To simplify (2.10), we introduce a latent variable $\mathbf{z}$ as in (2.2). We denote the complete data by $(y_i, \mathbf{x}_i, \mathbf{z}_i : i = 1, 2, ..., n)$. The complete data log-likelihood function is

$$l_n^c(\boldsymbol{\theta}, \boldsymbol{\Gamma}) = \sum_{i=1}^{n} \sum_{j=1}^{m} z_{ij}\, log\{\pi_j \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)\}.$$

For a given i there are j latent variables. However, only one of them is equal to one, while the others are zero. Hence, the penalised complete log-likelihood function is

$$J_n^c(\boldsymbol{\theta}, \boldsymbol{\Gamma}) = l_n^c(\boldsymbol{\theta}, \boldsymbol{\Gamma}) - \sum_{i=1}^{n} \sum_{j=1}^{m} P_\lambda(|\boldsymbol{\gamma}_{ij}|). \quad (2.11)$$

Since we do not observe the values of the latent variables, the expected value of the complete log-likelihood is calculated in the E-step. The responsibility of each observation

belonging to component j is

$$\begin{aligned} p_{ij} = E[z_{ij} \,|\, y_i; \boldsymbol{\theta}, \boldsymbol{\Gamma}] &= P(z_{ij} = 1 \,|\, y_i; \boldsymbol{\theta}, \boldsymbol{\Gamma}) \\ &= \frac{P(z_{ij} = 1, y_i \,|\, \boldsymbol{\theta}, \boldsymbol{\Gamma})}{P(y_i \,|\, \boldsymbol{\theta}, \boldsymbol{\Gamma})} \\ &= \frac{P(y_i \,|\, z_{ij}; \boldsymbol{\theta}, \boldsymbol{\Gamma}) P(z_{ij} = 1 \,|\, \boldsymbol{\theta}, \boldsymbol{\Gamma})}{P(y_i \,|\, \boldsymbol{\theta}, \boldsymbol{\Gamma})} \\ &= \frac{\pi_j \, \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)}{\sum_{j=1}^{m} \pi_j \, \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)} \,. \end{aligned} \tag{2.12}$$

The conditional expectation of the penalised complete log-likelihood to use in the M-step is obtained by replacing z_{ij} in $l_n^c(\boldsymbol{\theta}, \boldsymbol{\Gamma})$ in (2.11) with p_{ij} from (2.12) which gives

$$Q(\boldsymbol{\theta}, \boldsymbol{\Gamma}) = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} \, log\{\pi_j \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)\} - \sum_{i=1}^{n} \sum_{j=1}^{m} P_\lambda(|\boldsymbol{\gamma}_{ij}|). \tag{2.13}$$

In the M-step, we maximise (2.13) with respect to $(\boldsymbol{\theta}, \boldsymbol{\Gamma})$, i.e. we update $(\boldsymbol{\theta}, \boldsymbol{\Gamma})$ until convergence of the Q function.

The estimating equations

We derive the equations used in the EM algorithm of the RM^2 model. For fixed $\boldsymbol{\Gamma}$ and σ_j, by using a weighted least squares procedure, each $\boldsymbol{\beta}_j$ can be explicitly solved. From the penalised complete log-likelihood in (2.13), we differentiate with respect to $\boldsymbol{\beta}$:

$$\frac{\partial Q}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{n} \sum_{j=1}^{m} p_{ij} \, \frac{1}{\phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)} \underbrace{\frac{\partial}{\partial \boldsymbol{\beta}} \phi(y_i - \mathbf{x}_i{}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)}_{(*)} \tag{2.14}$$

Let us first consider (*),

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{\beta}}\phi(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2) \\
&= \frac{\partial}{\partial \boldsymbol{\beta}} \frac{1}{\sqrt{2\pi\sigma_j{}^2}} exp\left(\frac{-(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)^2}{2\sigma_j{}^2}\right) \\
&= \frac{1}{\sqrt{2\pi\sigma_j{}^2}} exp\left(\frac{-(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)^2}{2\sigma_j{}^2}\right) \cdot \left(\frac{2(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)}{2\sigma_j{}^2}\right) \cdot \mathbf{x}_i \\
&= \frac{\mathbf{x}_i \,.\, exp\left(\frac{-(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)^2}{2\sigma_j{}^2}\right) \cdot \left(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j\right)}{\sqrt{2\pi\sigma_j{}^2} \,.\, \sigma_j{}^2} \\
&= \frac{\mathbf{x}_i \,.\, exp\left(\frac{-(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)^2}{2\sigma_j{}^2}\right) \cdot \left(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j\right)}{\sqrt{2\pi}\,(\sigma_j{}^2)^{\frac{3}{2}}}.
\end{aligned} \tag{2.15}$$

If we replace (*) in (2.14) with (2.15) we get

$$\begin{aligned}
\frac{\partial Q}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \frac{\sqrt{2\pi}\,(\sigma_j{}^2)^{\frac{1}{2}}}{exp\left(\frac{-(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)^2}{2\sigma_j{}^2}\right)} \\
&\qquad \times \frac{\mathbf{x}_i \,.\, exp\left(\frac{-(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)^2}{2\sigma_j{}^2}\right) \cdot \left(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j\right)}{\sqrt{2\pi}\,(\sigma_j{}^2)^{\frac{3}{2}}} \\
0 &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} \frac{\mathbf{x}_i \,.\, (y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j)}{\sigma_j{}^2} \\
0 &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\ \left(\mathbf{x}_i y_i - \mathbf{x}_i\mathbf{x}_i{}^T\boldsymbol{\beta}_j - \mathbf{x}_i\gamma_{ij}\sigma_j\right) \\
\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\,\mathbf{x}_i\mathbf{x}_i{}^T\boldsymbol{\beta}_j &= \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\ (\mathbf{x}_i y_i - \mathbf{x}_i\gamma_{ij}\sigma_i) \\
\boldsymbol{\beta}_j &= \left(\sum_{i=1}^{n} p_{ij}\,\mathbf{x}_i\mathbf{x}_i{}^T\right)^{-1} \left(\sum_{i=1}^{n} p_{ij}\,\mathbf{x}_i(y_i - \gamma_{ij}\sigma_i)\right).
\end{aligned}$$

For fixed $\boldsymbol{\Gamma}$ and $\boldsymbol{\beta}_j$, there is no explicit solution for σ_j as it appears in the mean structure, hence there is no closed-form expression to estimate it. This is solved by using a non-linear optimisation algorithm. We briefly discuss the R-package, *nloptr* (see Conn et al. (1991)), that will be used to solve for $\sigma_1, \sigma_2, ..., \sigma_m$ (see Algorithm 2.1).

The *nloptr* package is an R interface to *NLopt* (see Johnson (2012); Ypma (2014)). *NLopt* optimises non-linear equations of the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \quad such\ that \\ &g(x) \leq 0, \\ &h(x) = 0, \\ &x_L \leq x \leq x_U, \end{aligned}$$

where $f(.)$ is the function to optimise and x is the n optimisation parameters. The following constraints are optional: the bound constraints, x_L and x_U, can take values between $-\infty$ and ∞, the non-linear inequality constraints can be specified in $g(.)$ and the equality constraints can be specified in $h(.)$. Note that the inequality constraints need to be specified in the *nloptr* function in the form $g(.) \leq 0$.

The basic inputs to *nloptr* for our problem are the lower bound imposed on the standard deviations and the objective function. The lower bound imposed on the standard deviations is given in (2.8) and the objective function to maximise is given in Algorithm 2.1. Note that the *nloptr* package minimises the objective function, therefore in our problem we will multiply the objective function with negative one in order to maximise the function. The constrained estimation is typically performed in practice when the ratio conditions are violated. We follow Yu et al. (2017) and disregard it in the applications in this mini-dissertation, i.e. the standard deviation is set equal to a small value (0.1) if the estimate thereof is close to zero.

To update the mixing probabilities π_j, and enforce the constraint on the π_j's given in (2.9), we need to maximise the Lagrangian function

$$Q^*(\boldsymbol{\theta}, \boldsymbol{\Gamma}) = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\, log\{\pi_j \phi(y_i - \mathbf{x}_i{}^T\boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, \sigma_j{}^2)\} - \sum_{i=1}^{n}\sum_{j=1}^{m} P_\lambda(|\boldsymbol{\gamma}_{ij}|) - \delta(\sum_{j=1}^{m}\pi_j - 1)$$

where δ is the Lagrange multiplier. Differentiating with respect to π_j and δ gives

$$\begin{aligned} \frac{\partial Q^*(\boldsymbol{\theta}, \boldsymbol{\Gamma})}{\partial \pi_j} &= \sum_{i=1}^{n} \frac{p_{ij}}{\pi_j} - \delta \overset{set}{=} 0 \\ \frac{\partial Q^*(\boldsymbol{\theta}, \boldsymbol{\Gamma})}{\partial \delta} &= \sum_{i=1}^{n} \pi_j - 1 \overset{set}{=} 0. \end{aligned} \tag{2.16}$$

If we sum over j in the first equation in (2.16) and multiply by π_j we get

$$\begin{aligned} \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} - \delta \sum_{j=1}^{m} \pi_j &\stackrel{set}{=} 0 \\ \therefore \quad \delta &= n \end{aligned} \tag{2.17}$$

since $\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij} = \sum_{i=1}^{n} 1 = n$ and $\sum_{j=1}^{m} \pi_j = 1$. To solve for π_j we substitute (2.17) into (2.16),

$$\begin{aligned} \frac{\partial Q^*(\boldsymbol{\theta}, \boldsymbol{\Gamma})}{\partial \pi_j} = \sum_{i=1}^{n} \frac{p_{ij}}{\pi_j} - n &\stackrel{set}{=} 0 \\ \therefore \quad \pi_j &= \sum_{i=1}^{n} \frac{p_{ij}}{n} \\ &= \frac{n_j}{n} \end{aligned}$$

where $n_j = \sum_{i=1}^{n} p_{ij}$ is the number of observations designated to component j.

For fixed $\boldsymbol{\theta}$, $\boldsymbol{\Gamma}$ is updated by maximising

$$\sum_{i=1}^{n}\sum_{j=1}^{m} {p_{ij}}^{(l+1)} log\, \phi(y_i - {\mathbf{x}_i}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j;\ 0, {\sigma_j}^2) - \sum_{i=1}^{n}\sum_{j=1}^{m} P_\lambda(|\gamma_{ij}|). \tag{2.18}$$

The optimisation involves estimation of each mean-shift parameter, hence the equation in (2.18) is separable in each γ_{ij}. Simplifying yields

$$\begin{aligned} &{p_{ij}}^{(l+1)} log \left(\frac{1}{\sqrt{2\pi}\sigma_j} exp \left(-\frac{1}{2} \left(\frac{y_i - {\mathbf{x}_i}^T \boldsymbol{\beta}_j - \gamma_{ij}\sigma_j}{\sigma_j} \right)^2 \right) \right) - P_\lambda(|\gamma_{ij}|) \\ &= {p_{ij}}^{(l+1)} log \left(\frac{1}{\sqrt{2\pi}\sigma_j} \right) - {p_{ij}}^{(l+1)} \frac{1}{2} \left(\gamma_{ij} - \frac{y_i - {\mathbf{x}_i}^T \boldsymbol{\beta}_j}{\sigma_j} \right)^2 - P_\lambda(|\gamma_{ij}|). \end{aligned}$$

This is a function for optimisation of γ_{ij} and therefore p_{ij} is considered to be a constant. To maximise, we only keep the terms which contains γ_{ij} and re-arrange to aid in the computation, i.e. maximise

$$-\frac{1}{2} \left(\gamma_{ij} - \frac{y_i - {\mathbf{x}_i}^T \boldsymbol{\beta}_j}{\sigma_j} \right)^2 - \frac{1}{{p_{ij}}^{(l+1)}} P_\lambda(|\gamma_{ij}|)$$

which is equivalent to minimising

$$\frac{1}{2} \left(\gamma_{ij} - \frac{y_i - {\mathbf{x}_i}^T \boldsymbol{\beta}_j}{\sigma_j} \right)^2 + \frac{1}{{p_{ij}}^{(l+1)}} P_\lambda(|\gamma_{ij}|). \tag{2.19}$$

The problem in (2.19) has an exact solution depending on the chosen penalty.

If we use the l_1 norm penalty function, the solutions of γ_{ij} are

$$\begin{aligned}
&\text{for } \gamma_{ij} > 0,\\
&\qquad \frac{\partial}{\partial\gamma}\left[\frac{1}{2}\left(\gamma_{ij} - \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j}\right)^2 + \frac{\lambda}{p_{ij}{}^{(l+1)}}\gamma_{ij}\right]\\
&\qquad = \gamma_{ij} - \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} + \frac{\lambda}{p_{ij}{}^{(l+1)}} \overset{set}{=} 0\\
&\therefore \quad \gamma_{ij} = \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} - \frac{\lambda}{p_{ij}{}^{(l+1)}} \quad \Rightarrow \quad \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} - \frac{\lambda}{p_{ij}{}^{(l+1)}} > 0 \quad \text{and}\\
&\text{for } \gamma_{ij} < 0,\\
&\qquad \frac{\partial}{\partial\gamma}\left[\frac{1}{2}\left(\gamma_{ij} - \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j}\right)^2 + \frac{\lambda}{p_{ij}{}^{(l+1)}}\gamma_{ij}\right]\\
&\qquad = \gamma_{ij} - \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} + \frac{\lambda}{p_{ij}{}^{(l+1)}} \overset{set}{=} 0\\
&\therefore \quad \gamma_{ij} = \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} - \frac{\lambda}{p_{ij}{}^{(l+1)}} \quad \Rightarrow \quad \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} - \frac{\lambda}{p_{ij}{}^{(l+1)}} < 0\\
&\qquad\qquad \Rightarrow \quad -\left(\frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j} - \frac{\lambda}{p_{ij}{}^{(l+1)}}\right) > 0.
\end{aligned} \tag{2.20}$$

The general form for the two solutions in (2.20) that corresponds to a soft thresholding rule, Θ_{soft}, is

$$\widehat{\gamma}_{ij} = \Theta_{soft}(\xi_{ij}; \lambda_{ij}{}^*) = sgn(\xi_{ij})(|\xi_{ij}| - \lambda_{ij}{}^*)_+ \tag{2.21}$$

where $\lambda_{ij}{}^* = \frac{\lambda}{p_{ij}{}^{(l+1)}}$, $\xi_{ij} = \frac{y_i - \mathbf{x}_i{}^T\beta_j}{\sigma_j}$ and $c_+ = max(c, 0)$.

If we use the l_0 norm penalty function the general form for the solution of γ_{ij} that corresponds to a hard thresholding rule, Θ_{hard}, is

$$\widehat{\gamma}_{ij} = \Theta_{hard}(\xi_{ij}; \lambda_{ij}{}^*) = \xi_{ij} I(|\xi_{ij}| > \lambda_{ij}{}^*) \tag{2.22}$$

where $\lambda_{ij}{}^* = \frac{\lambda}{\sqrt{p_{ij}{}^{(l+1)}}}$ and $I(.)$ is an indicator function.

The thresholding-embedded EM algorithm is given in Algorithm 2.1 with λ as the tuning parameter. Note that the penalised log-likelihood in (2.4) is an increasing function, that is

$$J_n(\widehat{\boldsymbol{\theta}}^{(l+1)}, \widehat{\boldsymbol{\Gamma}}^{(l+1)}) \geq J_n(\widehat{\boldsymbol{\theta}}^{(l)}, \widehat{\boldsymbol{\Gamma}}^{(l)})$$

Algorithm 2.1: The thresholding-embedded EM algorithm for the RM^2 model

Initialise $\boldsymbol{\theta}^{(0)}$ and $\boldsymbol{\Gamma}^{(0)}$, and set $l \leftarrow 0$.

repeat

1. E-step: Calculate $Q(\boldsymbol{\theta}, \boldsymbol{\Gamma} | \boldsymbol{\theta}^{(l)}, \boldsymbol{\Gamma}^{(l)})$ as in (2.13).
2. M-step: Maximise $Q(\boldsymbol{\theta}, \boldsymbol{\Gamma} | \boldsymbol{\theta}^{(l)}, \boldsymbol{\Gamma}^{(l)})$ by updating $\pi_j^{(l+1)} = (\sum_{i=1}^{n} {p_{ij}}^{(l+1)})/n$ and the other unknown parameters by iterating the steps below ($i = 1, 2, ..., n,\ j = 1, 2, ..., m$):

   2.1 $\quad {\boldsymbol{\beta}_j}^{(l+1)} \leftarrow \Big(\sum_{i=1}^{n} {p_{ij}}^{(l+1)} \mathbf{x}_i {\mathbf{x}_i}^T \Big)^{-1} \Big(\sum_{i=1}^{n} {p_{ij}}^{(l+1)} \mathbf{x}_i (y_i - \gamma_{ij}\sigma_j) \Big)$,

   2.2 $\quad (\sigma_1, ..., \sigma_m)^{(l+1)} \leftarrow \underset{(\sigma_1,...,\sigma_m) \in \Omega_\sigma}{argmax} \sum_{i=1}^{n} \sum_{j=1}^{m} {p_{ij}}^{(l+1)} log\, \phi(y_i - {\mathbf{x}_i}^T {\boldsymbol{\beta}_j}^{(l+1)} - {\gamma_{ij}}^{(l)}\sigma_j;\ 0, {\sigma_j}^2)$,

   2.3 $\quad {\gamma_{ij}}^{(l+1)} \leftarrow \Theta(\xi_{ij}; \lambda_{ij}^*)$

   where Θ is a thresholding rule given in (2.21) or (2.22) which depends on the choice of penalty.

$l \leftarrow l + 1$

until convergence

for all $l > 0$. Algorithm 2.1 converges as a result of this property. If component variances are equal, the algorithm can be adapted in the m component mixture model in (2.3) by setting $\sigma_j = \sigma$, $\sigma > 0$.

We use the Bayesian Information Criterion (BIC) by Yi et al. (2015); Yu et al. (2017) to select an appropriate value for the tuning parameter λ, given as

$$BIC(\lambda) = -l(\lambda) + log(n)\, df(\lambda),$$

where $l(\lambda)$ is the log-likelihood of the mixture model determined at the solution of the tuning parameter λ and $df(\lambda)$ is the degrees of freedom of the estimated model. It is calculated as the sum of the number of non-zero elements in $\widehat{\boldsymbol{\Gamma}}$ (the collection of mean-shift vectors) and the number of parameters across all the components pertaining to the mixture model. We then follow the approach by Zou (2006) in choosing the "best" value for λ; we fit the model for a range of tuning parameter values in equal intervals on the log scale. The smallest tuning parameter, λ_{min}, is the λ for which $\pm$ 50% of the values

in $\mathbf{\Gamma}$ are non-zero and the largest tuning parameter, λ_{max}, is the λ for which $\mathbf{\Gamma}$ results in a zero matrix.

2.4 Model performance measures

The mean squared error (MSE) is used to determine the parameter estimation performance and the below measures are used to determine the performance of outlier identification:

1. the proportion of good points labelled as outliers (I),
2. accuracy (A),
3. specificity (S) and
4. recall (R).

We calculate these performance measures when we compare robust mixture regression methods in a simulation study in Chapter 3.

2.5 Simulated example

We consider observations simulated from a mixture regression model which we then contaminate with additive outliers. Since the focus is on investigating the outlier detection performance, we keep the number of regression components small by choosing $m = 2$.

2.5.1 Initial values setup for parameter estimates

The RM^2 approach is illustrated with a simulation in the case of equal and unequal component variances. A vital point in parameter estimation remains choosing the initial values for each parameter in the EM algorithm. We follow the approach by Yu et al. (2017). Start by drawing $(10 \times m)$ random samples with replacement as to mimic the m regression components, with the sample size of each the same as that of the data. The parameters are estimated with ordinary least squares (OLS) estimation. For equal

component variances we use a pooled standard deviation. To find initial estimates for the mean-shift parameters γ_{ij}, we select 20% of the points with the smallest ordered densities. If the response value for these points with the smallest ordered densities is greater than the maximum of the m component means, we assign the mean-shift parameter for that observation equal to the difference between the response and the maximum of the m component means. In other words, an 'unusual' observation close to the maximum component mean is assigned a value $y - max(\mu_{1,i}, ..., \mu_{m,i})$ if it is greater than the maximum component means, and a value $y - min(\mu_{1,i}, ..., \mu_{m,i})$ otherwise. As stated previously, $m = 2$ in the following examples.

2.5.2 Equal component variances

The response $y_i, \quad i = 1, 2, ..., n$, is independently generated with

$$y_i = \begin{cases} 1 - 2x_{i1} + \gamma_{i1}\sigma + \epsilon_{i1} & \text{if } z_{i1} = 1 \\ 1 + 4x_{i1} + \gamma_{i2}\sigma + \epsilon_{i2} & \text{if } z_{i1} = 0, \end{cases} \tag{2.23}$$

where z_{i1} is a Bernoulli random variable indicating the component that observation i came from with $P(z_{i1} = 1) = 0.3$, x_{i1} is independently generated from a standard normal distribution and the error terms ϵ_{i1} and ϵ_{i2} are independently generated from a $N(0, \sigma^2)$ where $\sigma^2 = 1$.

We generate $n = 200$ observations from the model in (2.23) with all mean-shift parameters (MSP) equal to zero. We then replace 5% of the observations with outliers. The MSPs are generated from a $UNI(11, 13)$ distribution and the absolute value of a non-zero MSP i.e. $|\gamma_{ij}|$ is used as follows:
We randomly replace 2 observations from component one with $y_i = 1 - 2x_{i1} - |\gamma_{i1}|\sigma + \epsilon_{i1}$, and 8 observations from component two with $y_i = 1 + 4x_{i1} + |\gamma_{i2}|\sigma + \epsilon_{i2}$, where $\sigma = 1$ and $x_{i1} = 2$ for both components.

Data exploration

The purpose of data exploration is to assess the data to inform and guide the model building process. One will usually have questions regarding the data at hand which can

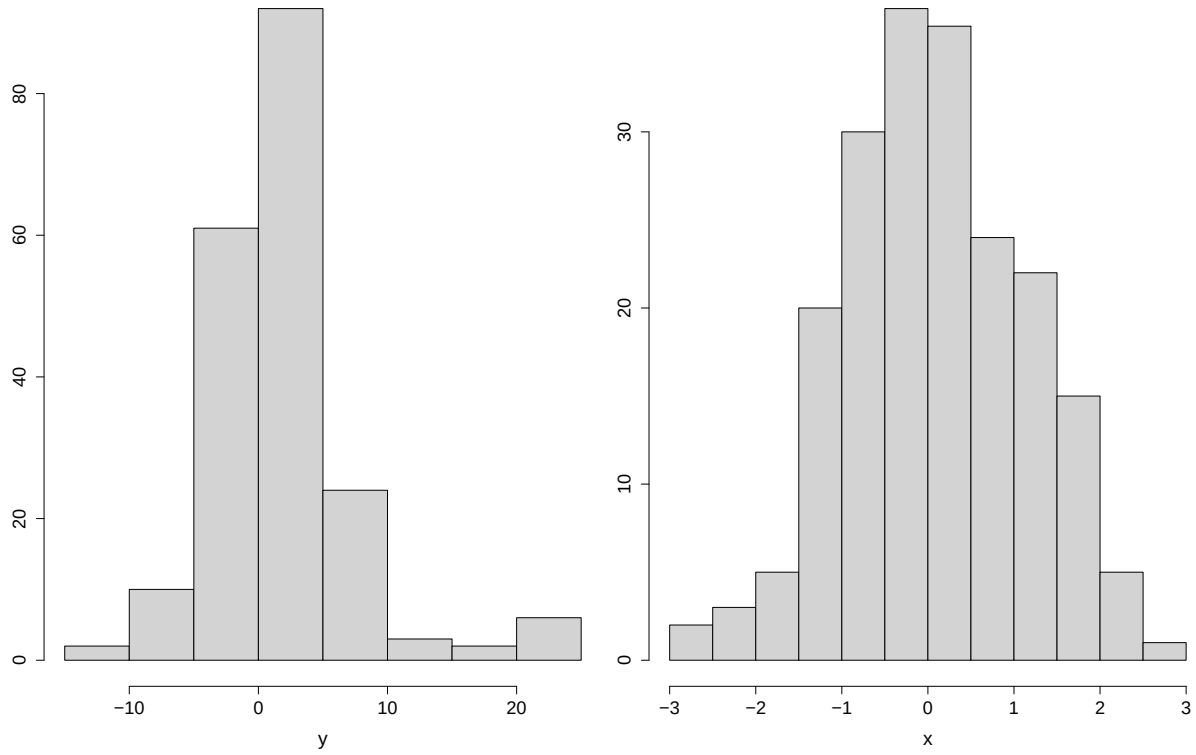
be resolved (partially or completely) by visualising, transforming and then modelling the data. Data exploration assists in refining the questions regarding the target definition (what we want to achieve with the model). There is no strict set of rules, hence the type of problem at hand dictates the best process to follow. In our simulation example, the truth is known, however in practice when the truth is unknown, data exploration is a good starting point. Below are the steps we use in the examples:

1. Draw visualisations of the variables.
2. Search for unusual observations.
3. Check for variation in the observations.
4. Establish patterns.

We start by visualising histograms of our response and feature variables in Figure 2.1. From Figure 2.1a, it is not entirely evident that there are outliers present in the response. Also, one might hesitate to believe that the response is normally distributed. The scatterplot in Figure 2.2 gives a clear indication of the relationship between the response and feature variable i.e. the trends/patterns present.

Model fitting and results

We model the distribution of the response with a Gaussian mixture regression model (using MLE) to illustrate how the estimation is distorted in the presence of outliers and then compare the estimation results to that when fitting the RM^2 model. We first estimate the parameters by fitting a Gaussian mixture regression model with two components with no added outliers and then fit the same model when the outliers are present, see Figure 2.2. The estimation results are given in Table 2.1. With reference to the original data, the RM^2 method yields remarkable similarities in parameter estimates compared to that of the MLE. It is clear from the estimation results that MLE fails when there are outliers present, MLE (with outliers). The RM^2 method yields better results than that of the MLE in the presence of outliers, RM^2 (with outliers). The RM^2 model fit is given in Figure 2.3.



(a) **(b)**

Figure 2.1: Histograms of the response (a) and feature (b) variables for equal component variances.

Table 2.1: Estimation results for the MLE and the RM^2 model fit on simulated data from a mixture regression model with two components with equal variances.

Method	π_1	π_2	$\beta_{0,1}$	$\beta_{1,1}$	$\beta_{0,2}$	$\beta_{1,2}$	σ
MLE (no outliers)	0.3117	0.6883	1.1634	-1.9849	0.8750	4.0977	0.9969
MLE (with outliers)	0.2875	0.7125	0.6177	-2.4212	1.5614	5.0492	1.6389
RM^2 (no outliers)	0.3085	0.6915	1.1653	-1.9894	0.8751	4.0933	1.0000
RM^2 (with outliers)	0.2570	0.7430	1.4398	-2.1186	0.8441	4.0619	1.1544

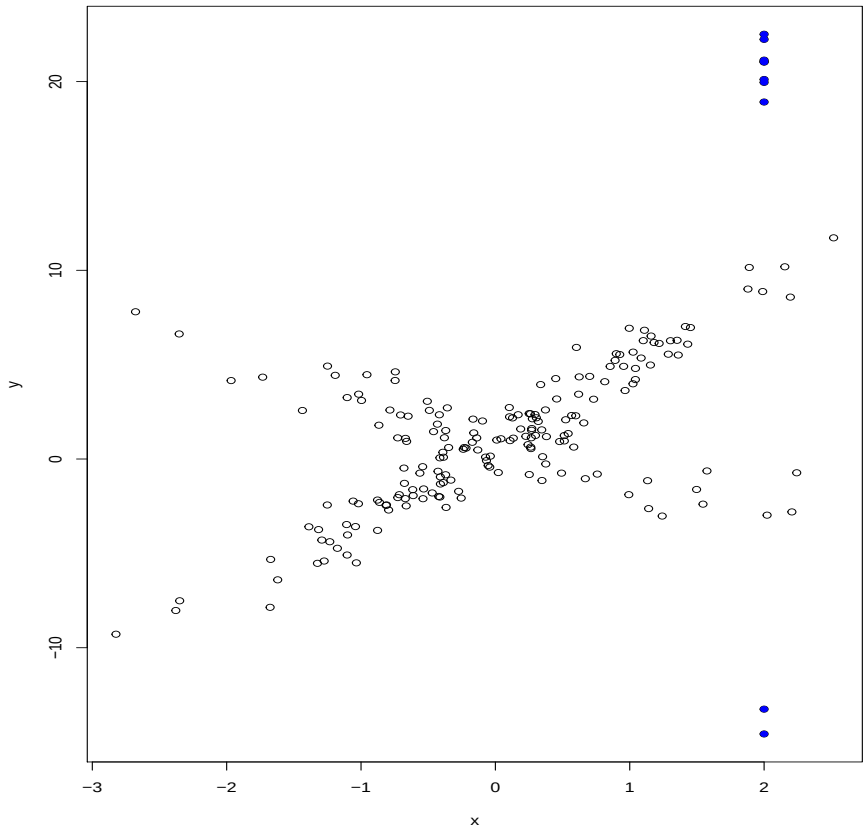


Figure 2.2: A scatter plot of simulated data from a mixture regression model with two components with equal variances and contaminated with outliers (in blue).

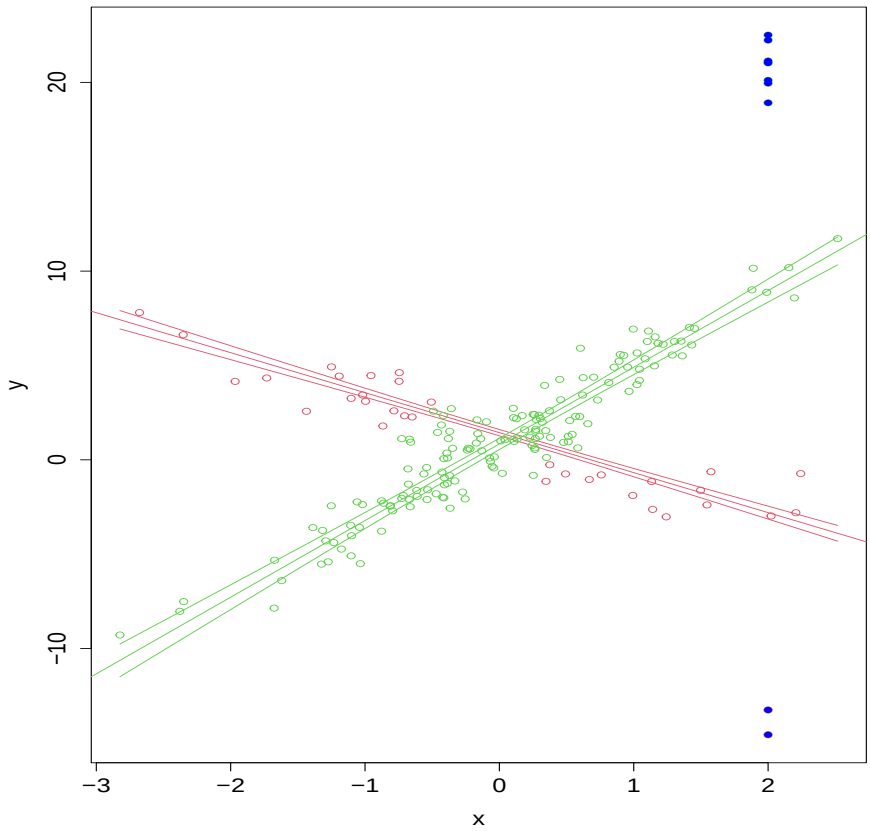


Figure 2.3: RM^2 model fit on simulated data from a two component regression model with equal component variances and 95% confidence bands.

2.5.3 Unequal component variances

The response $y_i, \quad i = 1, 2, ..., n$, is independently generated with

$$y_i = \begin{cases} 1 - 2x_{i1} + \gamma_{i1}\sigma_1 + \epsilon_{i1} & \text{if } z_{i1} = 1 \\ 1 + 4x_{i1} + \gamma_{i2}\sigma_2 + \epsilon_{i2} & \text{if } z_{i1} = 0, \end{cases} \tag{2.24}$$

where z_{i1} is a Bernoulli random variable indicating the component that observation i came from with $P(z_{i1} = 1) = 0.3$, x_{i1} is independently generated from a standard normal distribution and the error terms ϵ_{i1} and ϵ_{i2} are independently generated from a $N(0, {\sigma_1}^2)$ and a $N(0, {\sigma_2}^2)$ with ${\sigma_1}^2 = 4$ and ${\sigma_2}^2 = 1$, respectively.

We generate $n = 200$ observations from the model in (2.24) with all mean-shift parameters (MSP) equal to zero. We then replace 5% of the observations with outliers. The MSPs are generated from a $UNI(11, 13)$ distribution and the absolute value of a non-zero MSP i.e. $|\gamma_{ij}|$ is used as follows:
We randomly replace 2 observations from component one with $y_i = 1 - 2x_{i1} - |\gamma_{i1}|\sigma_1 + \epsilon_{i1}$, where $\sigma_1 = 2$ and $x_{i1} = 2$, and 8 observations from component two with $y_i = 1 + 4x_{i1} + |\gamma_{i2}|\sigma_2 + \epsilon_{i2}$, where $\sigma_2 = 1$ and $x_{i1} = 2$.

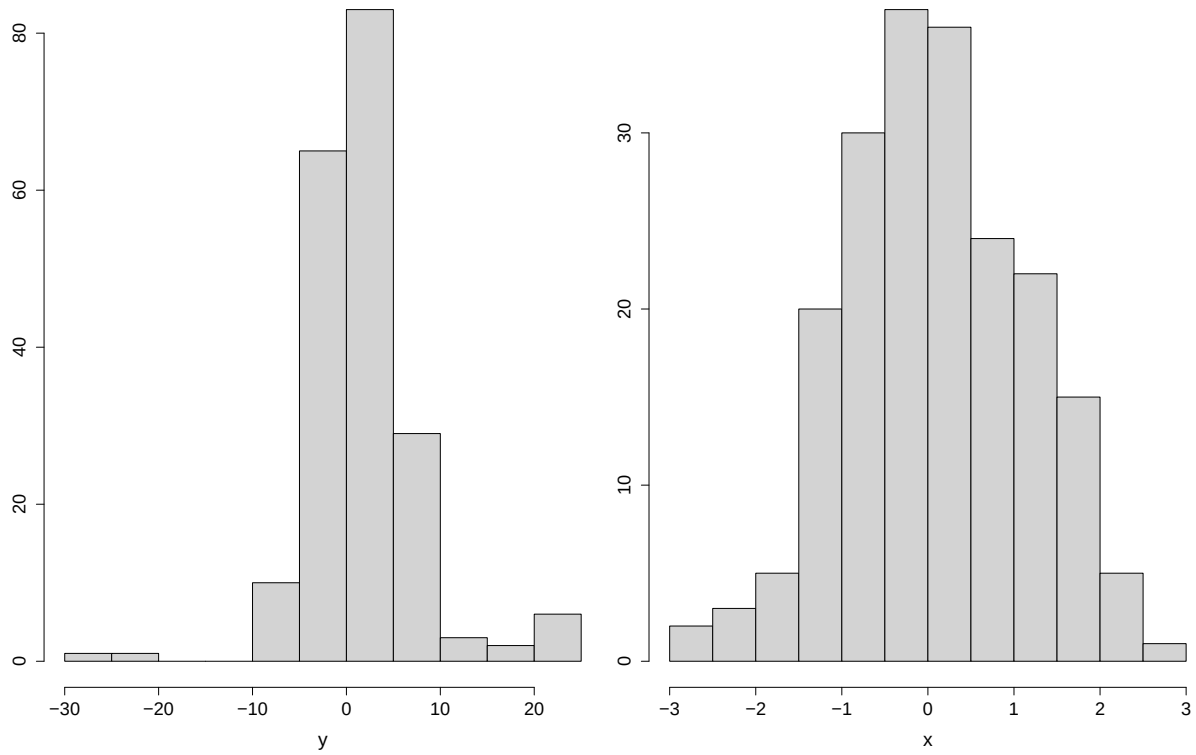
Data exploration

We start by visualising histograms of our response and feature variables in Figure 2.4. From Figure 2.4a, it is evident that there are outliers present in the response. The scatterplot in Figure 2.5 gives a clear indication of the relationship between the response and feature variable i.e. the trends/patterns present. The two regression trends are obvious and the difference in the component variations is visible.

Model fitting and results

Similar to the equal component variance example, we model the distribution of the response with a Gaussian mixture regression model (using MLE) to illustrate how the estimation is distorted in the presence of outliers and then show the estimation results when fitting the RM^2 model.

We first estimate the parameters using a Gaussian mixture regression model with two components before adding outliers to the data and then fit the same model when the



(a) **(b)**

Figure 2.4: Histograms of the response (a) and feature (b) variables for unequal component variances.

outliers are present, see Figure 2.5. The estimation results are given in Table 2.2. With reference to the original data, the RM^2 method yields similarities in parameter estimates compared to that of the MLE. However, the parameter estimates for the component variances are slightly different. This is due to the detection of possible outliers in the data by the RM^2 method. Again, it is clear from the estimation results where there are outliers present, MLE fails. The RM^2 method yields similar results when outliers are present to that of the MLE if there are no outliers, thereby verifying the efficacy of detecting outliers. The RM^2 model fit is given in Figure 2.6.

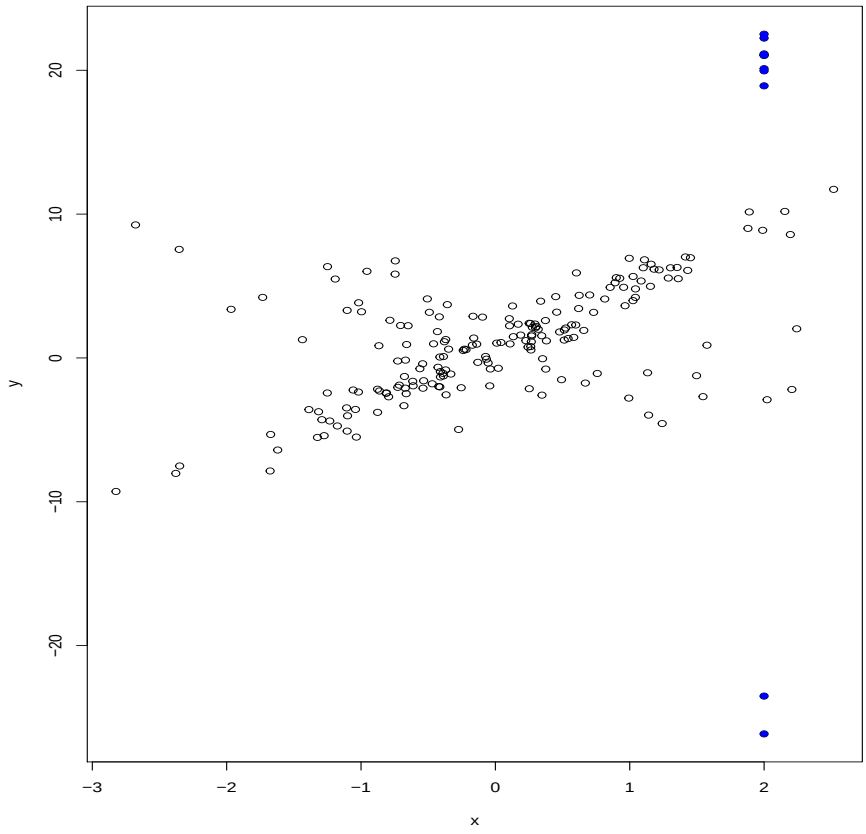


Figure 2.5: A scatter plot of simulated data from a mixture regression model with two components with unequal variances and contaminated with outliers (in blue).

2.5.4 Other robust methods

In the simulation study in Chapter 3, we compare the results of the RM^2 with two selected robust methods: the TLE and robust mixture regression making use of the t-distribution. These methods are briefly discussed here. For completeness, let $\mathbf{z}$ be defined as in (2.2). Let $\mathbf{z}_i = (z_{i1} \quad z_{i2} \quad \dots \quad z_{im})^T$ the collection of latent variables for observation i, where m indicates the number of regression components.

Table 2.2: Estimation results for the MLE and the RM^2 model fit on simulated data from a two component mixture regression model with unequal variances.

Method	π_1	π_2	$\beta_{0,1}$	$\beta_{1,1}$	$\beta_{0,2}$	$\beta_{1,2}$	σ_1	σ_2
MLE (no outliers)	0.3076	0.6924	1.1722	-1.9826	0.8717	4.0878	2.2280	0.9680
MLE (with outliers)	0.3200	0.6801	2.7979	0.8390	0.8701	4.0251	8.7347	1.0104
RM^2 (no outliers)	0.2885	0.7115	1.2326	-2.3289	0.8722	4.0879	1.3760	0.9946
RM^2 (with outliers)	0.3625	0.6375	1.1668	-1.9251	0.8724	4.0916	2.1302	0.9813

Trimmed Likelihood Estimator (TLE)

Neykov et al. (2007) proposed a TLE method which is defined as

$$\widehat{\theta}_{TLE} = \underset{\theta, S_h}{argmax} \sum_{i \in S_h} log \left\{ \sum_{j=1}^{m} \pi_j \phi(y_i; \mathbf{x}_i{}^T \beta_j, \sigma_j{}^2) \right\}$$

where $\mathbf{x}$ is a vector with p dimensions, S_h is a subset of observations $(1, 2, ..., n)$ with $h = n(1-\alpha)$ the number of elements in the subset and α is the trimming percentage of the data. Estimating the parameters $\widehat{\theta}_{TLE}$ involves two steps namely "trial" and "refinement". The final estimates are those where the negative log-likelihood is the smallest. The proportion of trimming α needs to be predetermined. Caution needs to be taken when choosing the trimming proportion. The TLE loses efficiency if α is too large and if it is too small (smaller than the outliers' percentage), the TLE fails. The two step procedure for estimation is given in Algorithm 2.2.

t-distribution

Yao et al. (2014) proposed to model the error density with a t-distribution. The density function of y_i given $\mathbf{x}_i$ is $f(y_i; \mathbf{x}_i, \theta) = \sum_{j=1}^{m} \pi_j h_t(y_i - \mathbf{x}_i^T \beta_j; \sigma, \nu_j)$ where the component error density $h_t(\epsilon; \sigma, \nu_j)$ is

$$h_t(\epsilon; \sigma, \nu_j) = \frac{\Gamma(\frac{\nu+1}{2})\sigma^{-1}}{(\pi\nu)^{\frac{1}{2}}\Gamma(\frac{\nu}{2})\left\{1 + \frac{\epsilon^2}{\nu\sigma^2}\right\}^{\frac{1}{2}(\nu+1)}}$$

and $h_t \sim t(\nu)$ with scale parameter σ. The unknown parameter θ can be estimated by

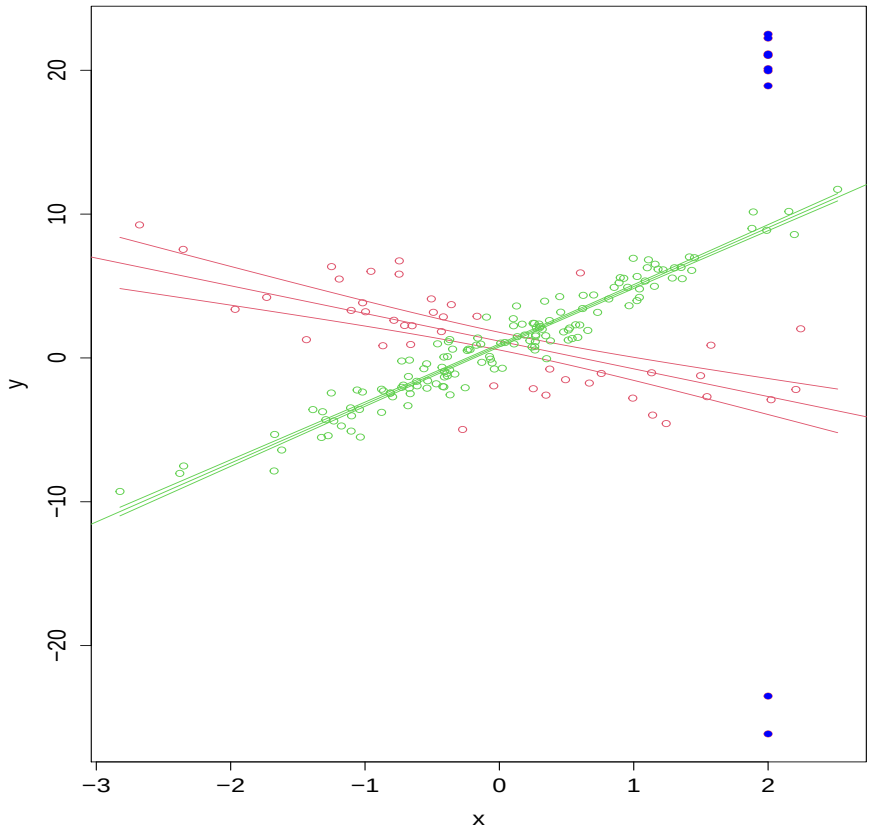


Figure 2.6: RM^2 model fit on simulated data from a two component regression model with unequal component variances and 95% confidence bands.

maximising the log-likelihood

$$l(\theta) = \sum_{i=1}^{n} log \left\{ \sum_{j=1}^{m} \pi_j h_t(y_i - \mathbf{x}_i^T \beta_j; \sigma, \nu_j) \right\} \tag{2.25}$$

Yao et al. (2014) proposed a simplified EM algorithm to maximise (2.25). It makes use of a t-distribution consisting of a mixture of continuous scale normal distributions. If u is a latent variable with a $gamma(\frac{1}{2}\nu, \frac{1}{2}\nu)$ distribution and $\epsilon|u \sim N(0, \frac{\sigma^2}{u})$, then the density of a variable $gamma(\alpha, \gamma)$ is $f(u; \alpha, \gamma) = \Gamma(\alpha)^{-1}\gamma^{\alpha}u^{\alpha-1}e^{-\gamma u}, u > 0$. As a result, ϵ has a marginal t-distribution with ν degrees of freedom and σ the scale parameter.

Algorithm 2.2: Iterative process for obtaining the estimates when fitting a mixture regression model using the TLE method.

1. Trial step

   Randomly select a few subsamples of size $h^\star$ from the data.
   Fit the model to obtain trial MLE sub-samples.

   Note: $h^\star$ should not be smaller than $m(p+1)$.

repeat

2. Refinement step

   2.1 Use each trial MLE as initial values and obtain the h cases with the smallest negative log-likelihood.
   2.2 Fit the model (1.1) to the h cases to obtain an enhanced fit, i.e. a larger trimmed likelihood.

   Note: To achieve highest BDP the refinement subsample size h should be $(n+m(p+1))/2$. If the percentage of contamination is known, a recommended choice for $h = n(1-\alpha)$.

until convergence

Therefore, given X, the complete log-likelihood for (y, u, z) is

$$l_c(\theta; y, u, z) = \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij}\ log\left\{\pi_j \phi\left(y_i; \mathbf{x}_i^T\beta_j, \frac{\sigma^2}{u_i}\right) f\left(u_i; \frac{1}{2}\nu_j, \frac{1}{2}\nu_j\right)\right\},$$

where $u = (u_1, u_2, ..., u_n)$; z and u are independent. The EM algorithm for this method is given in Algorithm 2.3.

2.6 Summary

In this section we explained the details and showed the computations relating to the RM^2 model. The crux of this method lies in the sparsity produced by the mean-shift parameters to simultaneously perform outlier detection and robust estimation. We recorded the estimation results after fitting a Gaussian mixture regression model using MLE and

Algorithm 2.3: EM algorithm for modelling the error density in the mixture regression model with a t-distribution.

Initialise $\boldsymbol{\theta}^{(0)}$, and set $l \leftarrow 0$.

repeat

1. E-step: Calculate

   ${p_{ij}}^{(l+1)} = E(z_{ij}|X, y, \theta^{(l)}) = \frac{{\pi_j}^{(l)} h_t(y_i - {\boldsymbol{x_i}}^T {\beta_j}^{(l)}; {\boldsymbol{\sigma}}^{(l)}, \nu_j)}{\sum_{j=1}^{m} {\pi_j}^{(l)} h_t(y_i - {\boldsymbol{x_i}}^T {\beta_j}^{(l)}; {\boldsymbol{\sigma}}^{(l)}, \nu_j)}$

   and

   ${\delta_{ij}}^{(l+1)} = E(u_i|X, y, \theta^{(l)}, z_{ij} = 1) = \frac{\nu+1}{\nu + \Delta^2(y_i - {\boldsymbol{x_i}}^T {\beta_j}^{(l)}; {\sigma^2}^{(l)})}$

   where $\Delta^2(y_i - {\boldsymbol{x_i}}^T \beta_j; \sigma^2) = (y_i - {\boldsymbol{x_i}}^T \beta_j)^2 / \sigma^2$.

2. M-step: Update parameter estimates

   2.1 ${\pi_j}^{(l+1)} \leftarrow \sum_{i=1}^{n} {p_{ij}}^{(l+1)} / n$,

   2.2 ${\beta_j}^{(l+1)} \leftarrow \Big(\sum_{i=1}^{n} \mathbf{x}_i {\mathbf{x}_j}^T {p_{ij}}^{(l+1)} {u_{ij}}^{(l+1)} \Big)^{-1} \Big(\sum_{i=1}^{n} \mathbf{x}_i y_i \ {p_{ij}}^{(l+1)} {u_{ij}}^{(l+1)} \Big), \ j = 1, 2, ..., m,$

   2.3 ${\sigma^2}^{(l+1)} \leftarrow \frac{\sum_{i=1}^{n} {p_{ij}}^{(l+1)} {u_{ij}}^{(l+1)} (y_i - {\mathbf{x}_i}^T {\beta_j}^{(l+1)})^2}{n}$,

   2.4 ${\nu_j}^{(l+1)} \leftarrow \underset{\nu_j}{argmax} \sum_{i=1}^{n} {p_{ij}}^{(l+1)}$
   $\times [-log\Gamma(\frac{1}{2}\nu_j) + \frac{1}{2}\nu_j \ log(\frac{1}{2}\nu_j) + \frac{1}{2}\nu_j \left\{ {\nu_{ij}}^{(l+1)} - {u_{ij}}^{(l+1)} \right\} - {\nu_{ij}}^{(l+1)}]$

   where ν_j is the degrees of freedom for component j.

$l \leftarrow l + 1$

until convergence

the RM^2 method to a simulated data set with two regression components for equal and unequal component variances. The estimation results for the Gaussian mixture regression model using MLE revealed its sensitivity to outliers present in the data. The RM^2 method proved to be highly robust against outliers. The outlier detection performance may yield different results when making use of other penalty functions. In our examples

we have used the l_0 penalty. We also briefly discussed two other robust mixture regression methods which are used in a simulation study in the next chapter. For further detail concerning the robustness properties of the RM^2 method see Yu et al. (2017).

Chapter 3

Simulation Study

3.1 Introduction

We consider data simulated from a mixture regression model where outliers were added to the observations. Since the focus is on testing the performance of detecting the outliers, we keep the component structure simple i.e. only two components, with two feature variables. We illustrate the case when component variances are equal. To correct the label switching issues in the simulation study, we assign the estimates to the component where the distance to the true parameter values, is minimised.

3.2 Simulation design

The response $y_i, \quad i = 1, 2, ..., n$ is independently generated with

$$y_i = \begin{cases} 1 - 2x_{i1} + x_{i2} + \gamma_{i1}\sigma + \epsilon_{i1} & \text{if } z_{i1} = 1 \\ 1 + 5x_{i1} + x_{i2} + \gamma_{i2}\sigma + \epsilon_{i2} & \text{if } z_{i1} = 0, \end{cases} \tag{3.1}$$

where z_{i1} is a Bernoulli random variable indicating the component that observation i came from with $P(z_{i1} = 1) = 0.3$, x_{i1} and x_{i2} are generated independently from a standard normal distribution and the error terms ϵ_{i1} and ϵ_{i2} are generated independently from a $N(0, \sigma^2)$ with $\sigma^2 = 1$.

We generate $n = 200$ observations from the model in (3.1) with all mean-shift parameters (msp) equal to zero. We then replace 5% of the observations with outliers. The msp's are generated from a $UNI(11, 13)$ distribution and the absolute value of a non-zero msp i.e. $|\gamma_{ij}|$ is used as follows:
We randomly replace 2 observations from component one with $y_i = 1 - 2x_{i1} + x_{i2} - |\gamma_{i1}|\sigma + \epsilon_{i1}$, and 8 observations from component two with $y_i = 1 + 5x_{i1} + x_{i2} + |\gamma_{i2}|\sigma + \epsilon_{i2}$, where $\sigma = 1$, $x_{i1} = 2$ and $x_{i2} = 2$. We only repeat this process a 100 times in light of computational expense and time.

We provide visualisations for a single data set simulated by the model in (3.1). A visualisation of the relationship between the response and feature variables is given in Figure 3.1 with the outliers marked in blue.

It is reasonably visible from the plot that the data points lie along two regression planes. A mixture regression model with two components is fitted to the data with each method (in Section 3.3). Selecting the number of regression components plays a critical role in mixture modelling and remains a challenge in practice. An ordinary approach is to run the model for a different number of components and then choose the model which yields the lowest information criterion.

3.3 Model fitting

We compare the RM^2 estimator (using the l_0 penalty) to the trimmed likelihood estimator, the mixture regression using the t-distribution and the Gaussian model using MLE, per the below:

1. The MLE of a Gaussian mixture regression model (MLE).
2. The TLE with the trimming percentage set to 10% ($TLE_{0.1}$).
3. The MLE of a mixture regression model that assumes the error has a t-distribution with 3 degrees of freedom (Mixregt).
4. The RM^2 element-wise estimator utilising the l_0 penalty ($RM^2(l_0)$).

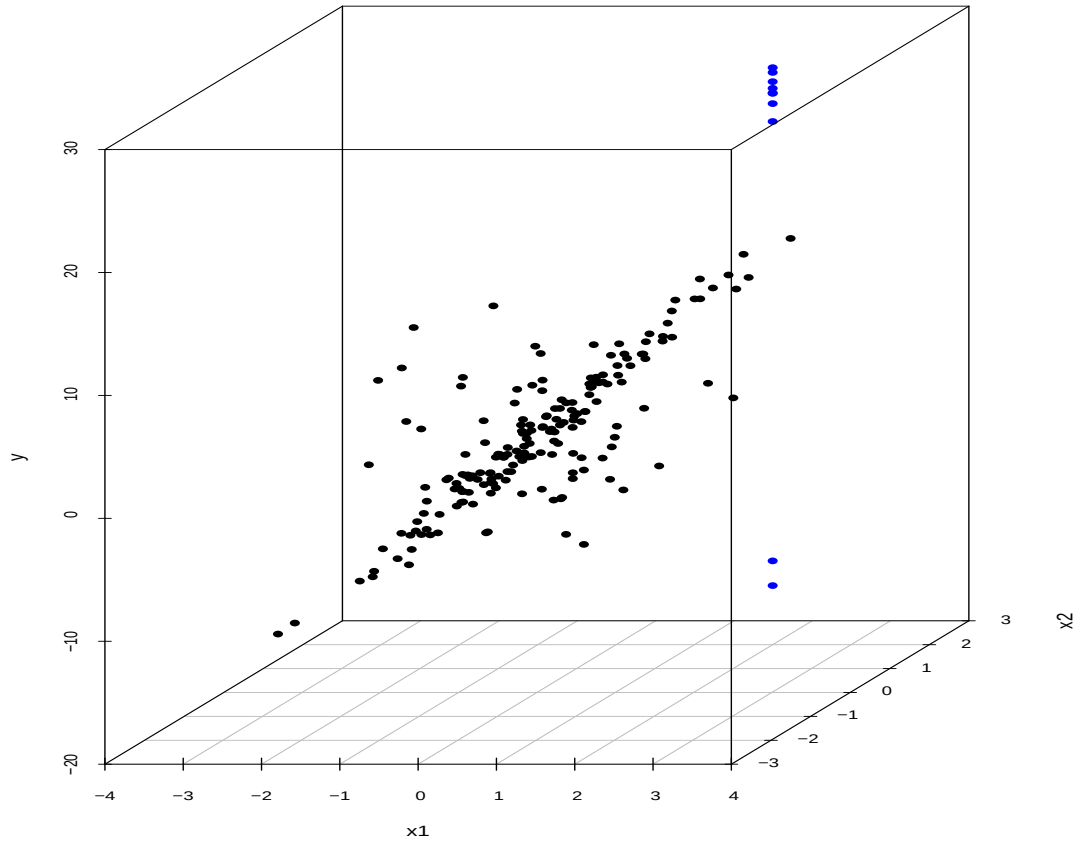


Figure 3.1: A 3D scatter plot for a single data set in the simulation study contaminated with outliers.

For these methods we use code by Yu et al. (2017), however, for the RM^2 method we include a non-linear optimisation algorithm to estimate the component variances, as discussed in Section 2.3.2.

Table 3.1: MSE of the parameter estimates in the simulation study

Method	$\pi_1 = 0.3$	$\beta_{0,1}$	$\beta_{1,1}$	$\beta_{2,1}$	$\beta_{0,2}$	$\beta_{1,2}$	$\beta_{2,2}$	σ
MLE	0.0026	0.2982	0.2667	1.0982	0.3148	0.6286	1.2117	2.5921
$TLE_{0.1}$	0.0017	0.0429	0.0387	0.0443	0.0122	0.0119	0.0105	0.0304
Mixregt	0.0013	0.0257	0.0241	0.0336	0.0119	0.0132	0.0130	0.0100
$RM^2(l_0)$	0.0016	0.0256	0.0205	0.0359	0.0092	0.0093	0.0078	0.0003

3.4 Results

Estimator accuracies relating to the above methods are compared using the MSE of the parameter estimates. The mean of the squared difference in the true and predicted parameter values is measured by the MSE, hence a lower MSE indicates a better model fit. The MSE results are given in Table 3.1. It is evident that MLE fails when there are outliers present. The other methods perform well, but overall the RM^2 model presents the lowest MSE. The RM^2 model fit for a single data set as per the model in (3.1) is shown in Figure 3.2 (an interactive graph is available [1]). We show the distribution of the parameter estimates for the second feature variable of component two i.e. $\beta_{2,2}$, fitted with each of the different estimators, in Figure 3.3. This shows how the outliers distort the estimation results in the case of MLE and fluctuate around the true parameter value for the other methods. However, the estimation is the most consistent in the $RM^2(l_0)$ case.

[1] file:///C:/Users/anika/Documents/2021/University/WST%20895/graphs/RM2-3D-fit-html.html

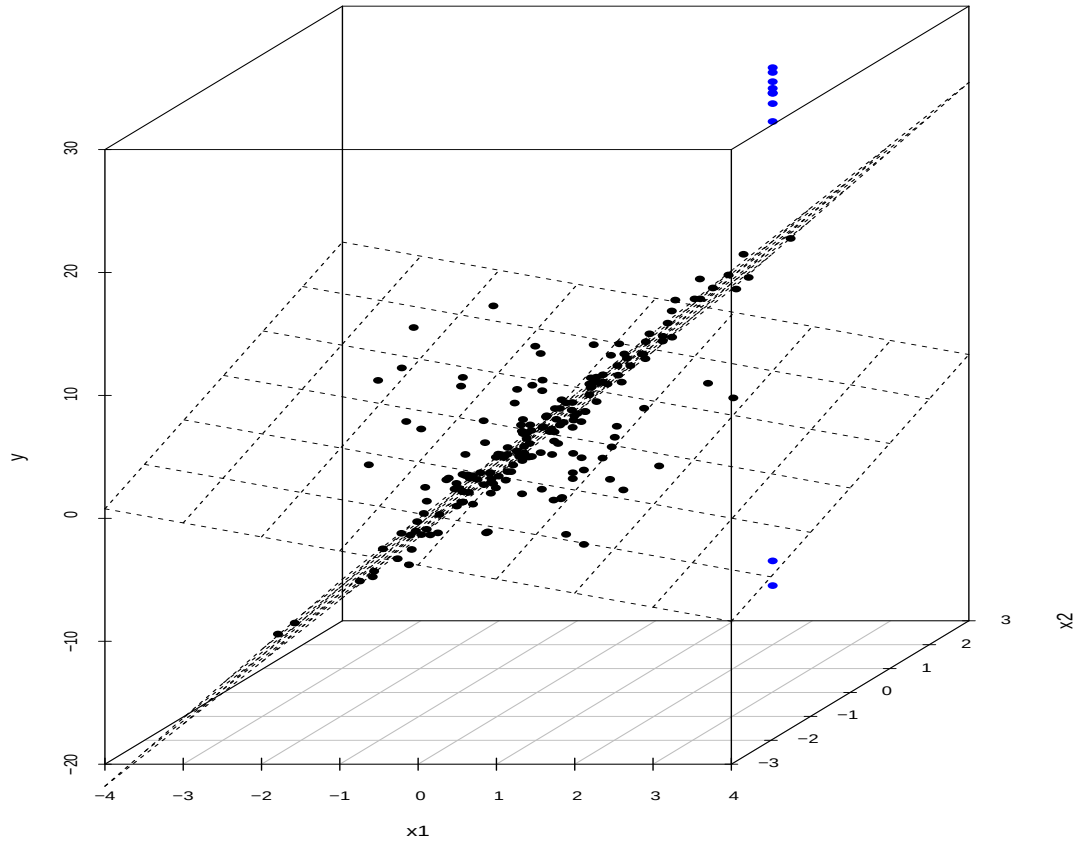
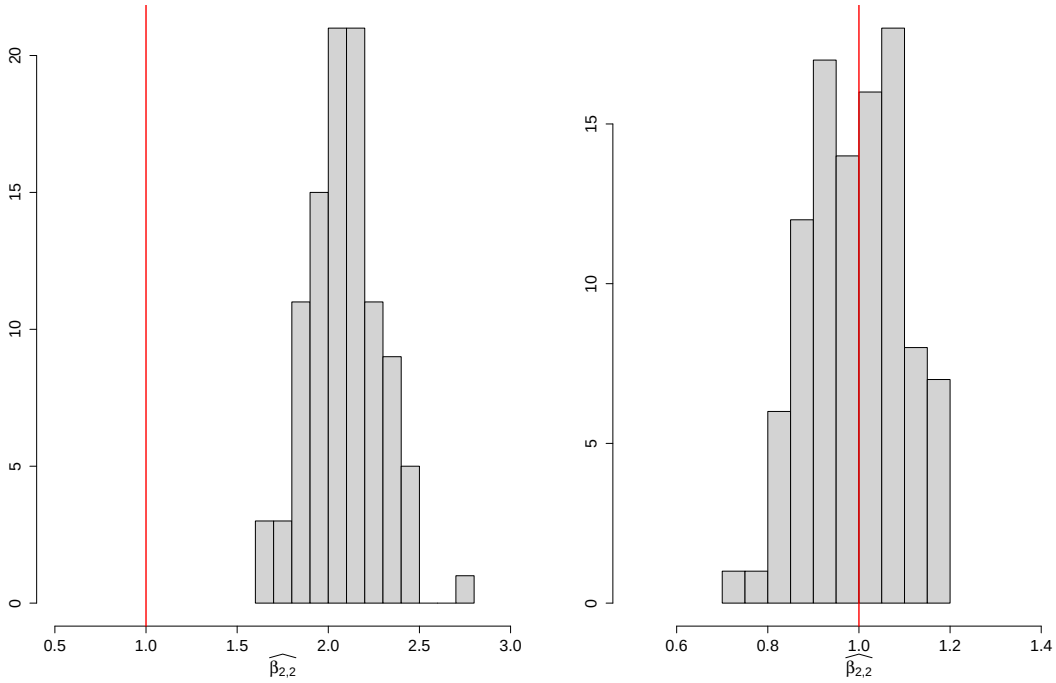
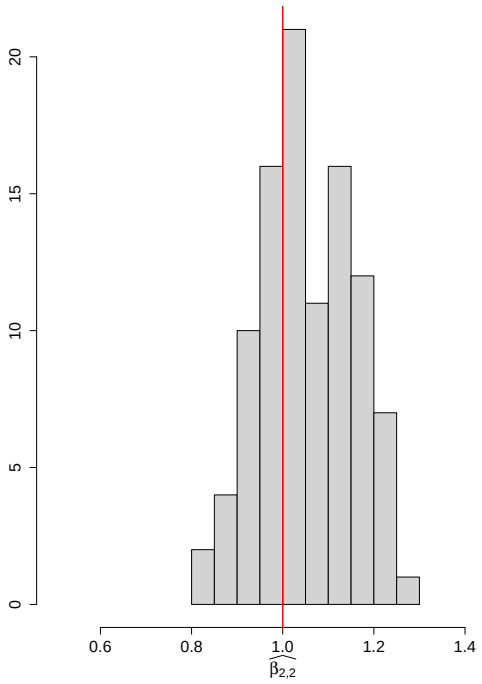


Figure 3.2: A 3D scatter plot for a single data set in the simulation study. The planes are fitted with the $RM^2(l_0)$ method and the 5% outliers are visible in blue.

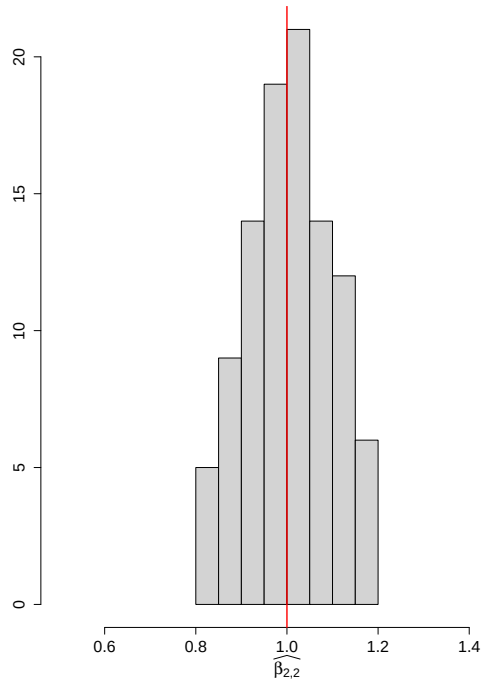


(a) MLE

(b) $TLE_{0.1}$



(c) Mixregt



(d) $RM^2(l_0)$

Figure 3.3: Histograms for the 100 estimates of $\beta_{2,2}$ by fitting each of the methods on the simulated data. The true parameter value is marked with a vertical red line.

Table 3.2: The outlier detection performance of the RM^2 model on simulated data

Measure	
Average proportion of good points labelled as outliers	0.0077
Average accuracy	0.9919
Average precision	0.8866
Average recall	0.9840

The measures to evaluate the outlier detection, as per Section 2.4, is only applied to the RM^2 method since this method explicitly identifies the outliers. To summarise:

- The average recall is 0.9840, which means that the average proportion of undetected outliers is 0.016. Put differently, of all the actual outliers, 98% were correctly predicted as outliers i.e. the success rate.
- The average proportion of good points labelled as outliers (I) is 0.0077. This means that the chance of flagging an observation "incorrectly" as an outlier is a mere 0.8%.
- The average precision of the model is 0.8866, which means that the model has an extremely low false positive rate.
- The average accuracy of the model is 0.9919 which validates that the model give a correct prediction 99% of the time.

These results are evident of a highly accurate robust model. The results are summarised in Table 3.2.

3.5 Summary

We have provided a comparison between a few robust mixture regression methods fitted to a simulated data set with 200 observations, which were repeated a 100 times. The MSE values for the parameter estimates were recorded. The RM^2 model showed an overall better fit than the other methods; a Gaussian mixture regression using MLE, the TLE and the heavy-tailed t-distribution. In addition, the RM^2 method succeeded in identifying the outliers in the data. It is imperative to take cognisance of the label switching issues and formulate a fix when comparing mixture models in a simulation study. For more detail on this see Celeux et al. (2000); Stephens (2000); Yao and Lindsay (2009); Yao (2012).

Chapter 4

Real-World Application

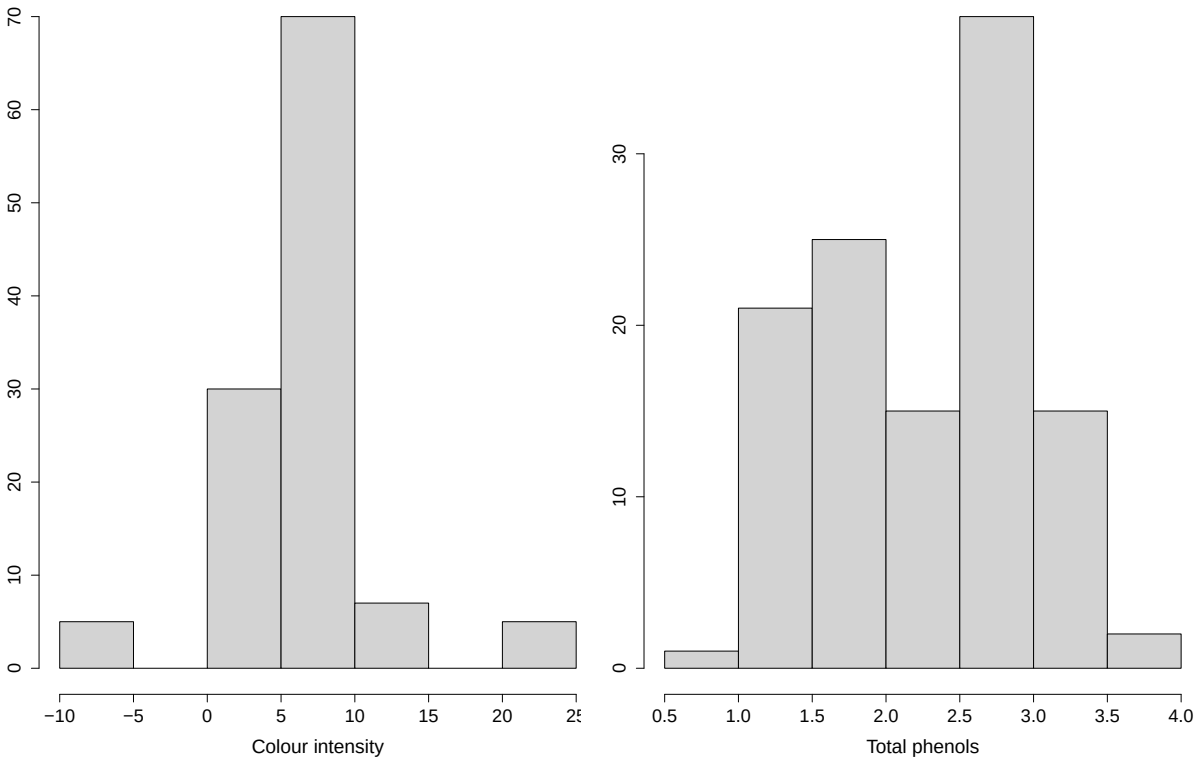
4.1 Introduction

In this chapter we apply the RM^2 method to wine data (see Dua and Graff (2017)). We compare the results of the RM^2 with that of a Gaussian mixture regression model using MLE. These models are fitted on the original data as well as to the data after adding outliers.

4.2 Data description

A chemical analysis was conducted on wines from grapes gathered from three different cultivars, originating from a common region in Italy. In the analysis 13 common ingredients were identified in each of the three types of wines. The data set has 178 observations and 14 variables. For the purpose of illustration we only use two types of wines, class 1 and class 3, therefore we have 107 observations. We will use three variables i.e. 'Class', 'Colour intensity' and 'Total phenols'. The variable 'Class' is only used to verify the prediction results of the model. We therefore choose 'Colour intensity' to be the response variable and 'Total phenols' the feature variable. We add ten outlier pairs to the data with the following rules:

1. $(1.5, a)$ where $a = 20 + 0.1i\,,$ i=1,2,3,4,5
2. $(3.5, b)$ where $b = -5 + 0.1j\,,$ j=1,2,3,4,5.



(a) **(b)**

Figure 4.1: Histograms of the response (a) and feature (b) variables for the wine data.

4.3 Exploratory data analysis

Histograms of both the response and feature variables are visualised in Figure 4.1. The extremes in the distribution of the response in Figure 4.1a is indicative that outliers are present in the vicinity of both -10 Au and 25 Au (colour intensity is measured in absorbance units Au). A scatter plot to visualise the relationship of the response and feature variables is given in Figure 4.2. From this we can easily identify the outliers (blue points).

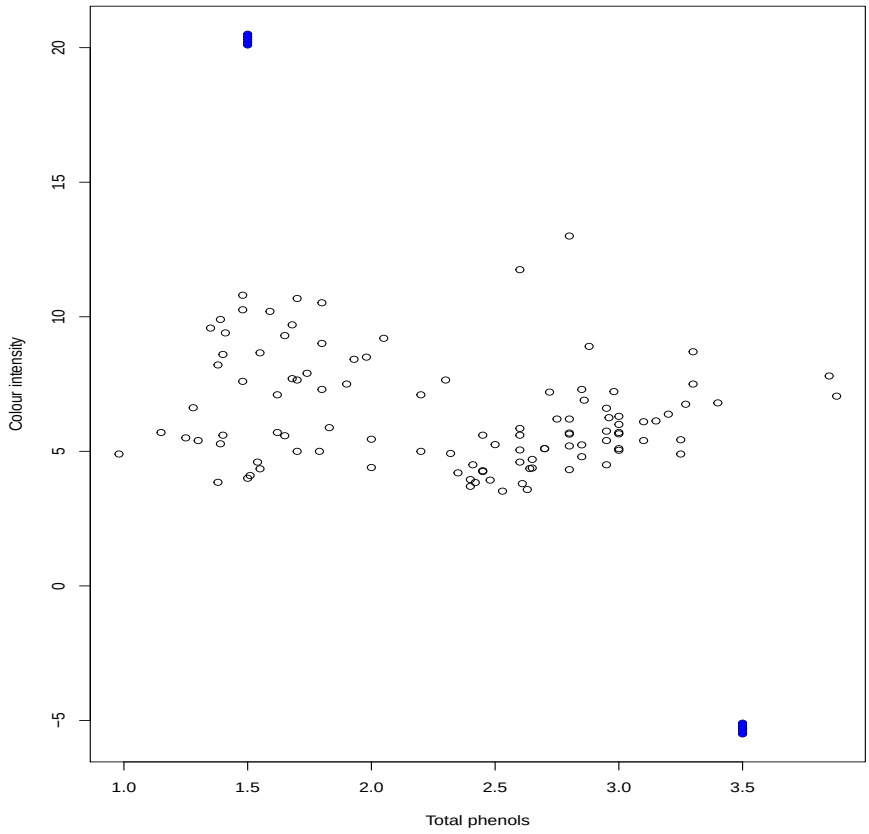


Figure 4.2: A scatter plot of the wine data with added outliers (in blue).

4.4 Model fit and results

The RM^2 method (using the l_0 penalty) is compared to a Gaussian mixture regression model using MLE, by recording the parameter estimates both with and without outliers present. We fit the mentioned Gaussian model with two components.

Firstly we illustrate how the models fit the data without outliers. A Gaussian mixture regression model fitted with two components, is shown in Figure 4.3. The parameter estimates for this model is given in Table 4.1 MLE (no outliers). The RM^2 model performs similar to the MLE when there are no outliers present, see Table 4.1 $RM^2(l_0)$ (no outliers). Now we fit the Gaussian model (also with two components) to the data with outliers. If we refer to Figure 4.4 it is obvious that the outliers have a huge influence

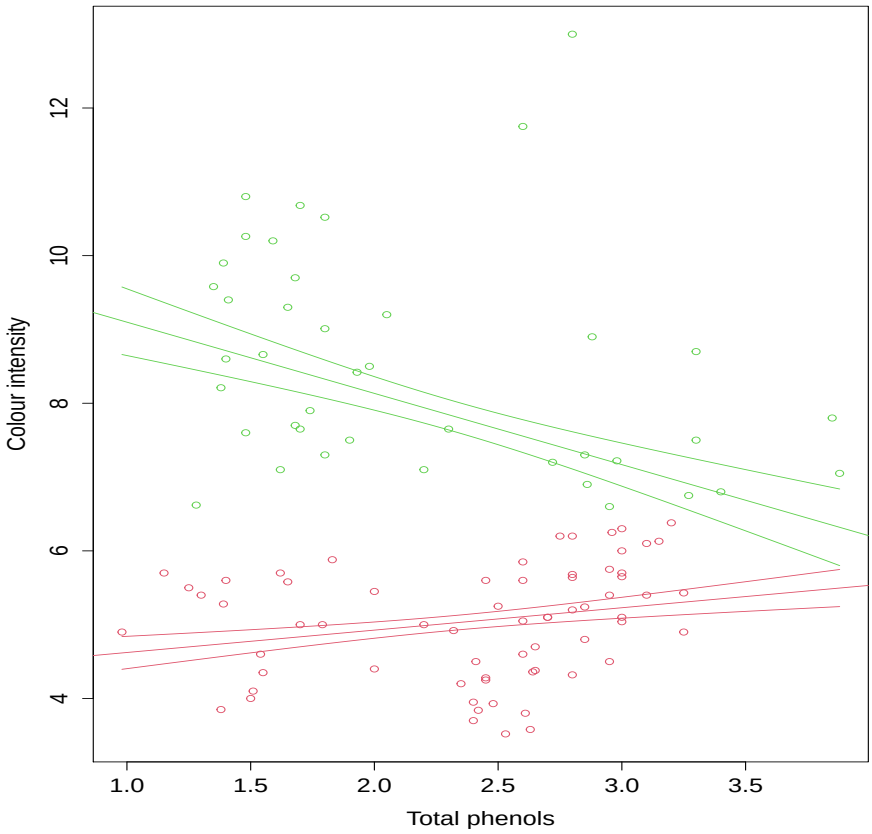


Figure 4.3: A Gaussian mixture model with two components fitted to the wine data without outliers and 95% confidence bands.

on the component structure and hence, on the parameter estimates as also reflected in Table 4.1 MLE (with outliers). The results in Table 4.1 for the RM^2 model shows a similar performance when outliers are present (see $RM^2(l_0)$ (with outliers)), to the MLE without outliers. This model fit is reflected in Figure 4.5 and indicates that the model can detect the outliers and perform accurate parameter estimation.

Table 4.1: Parameter estimates for the wine data

Method	π_1	$\beta_{0,1}$	$\beta_{1,1}$	$\beta_{0,2}$	$\beta_{1,2}$	σ_1	σ_2
MLE (no outliers)	0.5367	4.3201	0.3031	10.0632	-0.9649	0.7923	1.7597
MLE (with outliers)	0.0943	39.5061	-12.7973	7.7270	-0.5825	0.1441	1.9774
$RM^2(l_0)$ (no outliers)	0.6405	4.5372	0.2560	10.8815	-1.2440	0.9711	1.0471
$RM^2(l_0)$ (with outliers)	0.6715	4.3622	0.3544	11.1974	-1.4085	0.9937	1.0262

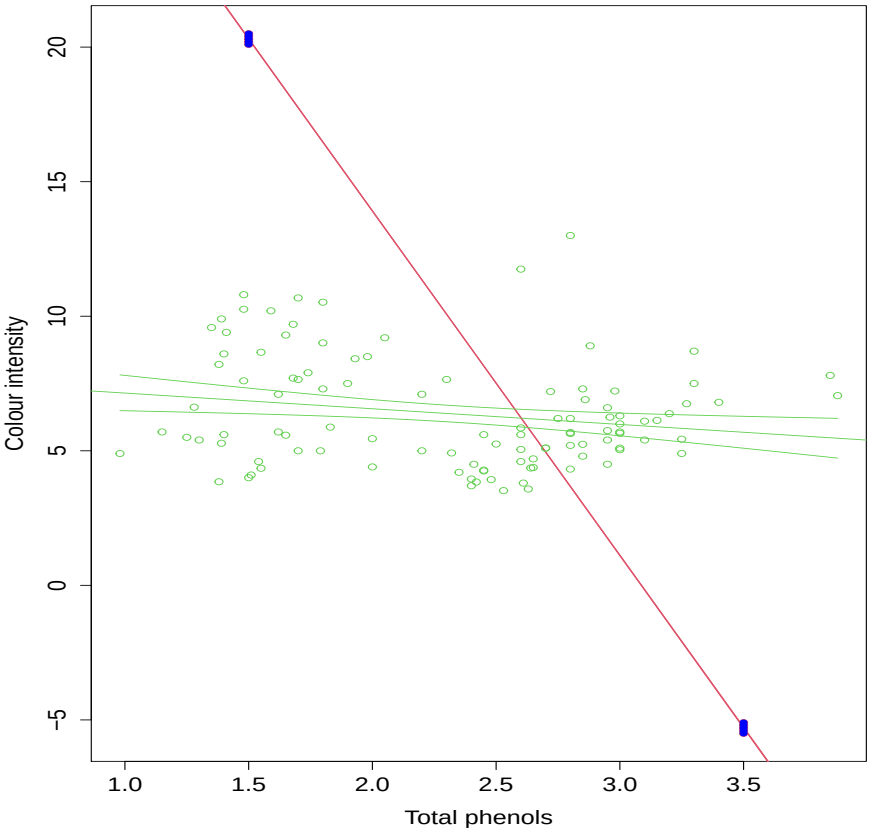


Figure 4.4: A Gaussian mixture model with two components fitted to the wine data with outliers and 95% confidence bands.

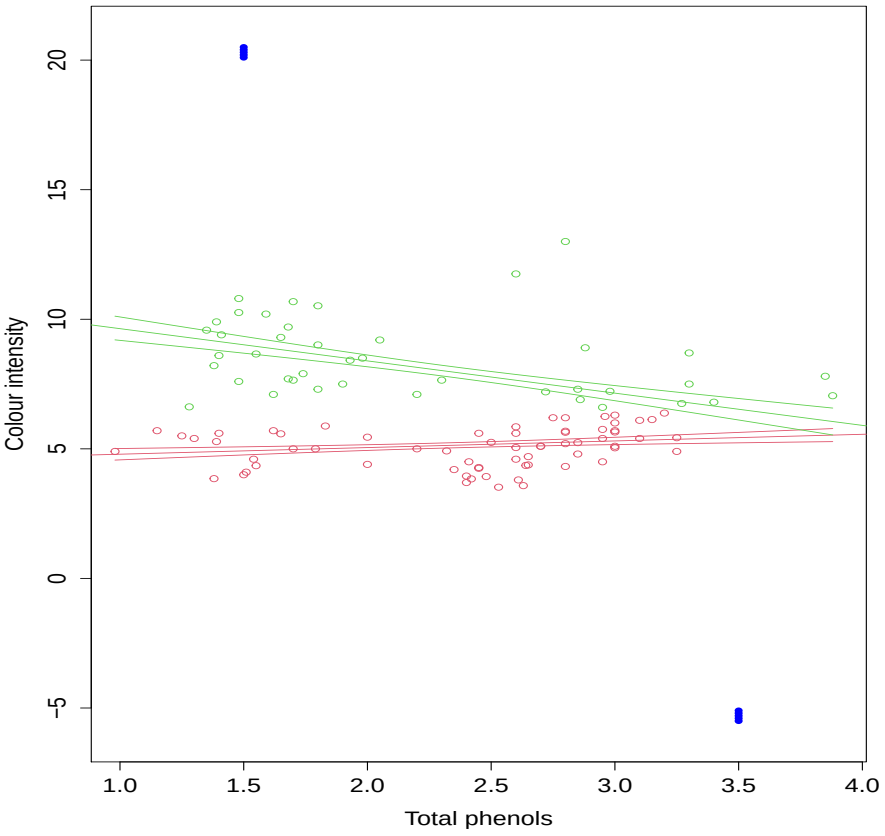


Figure 4.5: The RM^2 model fitted to the wine data with outliers and 95% confidence bands.

4.5 Summary

In this chapter a Gaussian mixture regression model using MLE was fitted on the original wine data (before we added outliers). The results from the original data were compared to the results after deliberate contamination of the data with outliers. It showed that the MLE fails when there are outliers present. To account for the outliers, we use the RM^2 method. This method proved to be robust against outliers, and therefore, estimates the parameters very close to the results of the MLE in the case of no outliers.

Chapter 5

Conclusions

This chaper gives a summary on our findings of exploring robust mixture regression in the presence of outliers. We provide possible future research opportunities related to robust mixture regression models.

5.1 Summary of conclusions

The main focus of this mini-dissertation was to explore robust mixture regression methods and their performance when outliers form part of the data. An in-depth analysis was conducted on the RM^2 method. We opted to investigate the mean-shift parameters in the RM^2 method. During this part of the process we re-affirmed that the essence lies in the sparsity of the mean-shift parameters. If the model identifies an observation as an outlier, the assigned mean-shift parameter adjusts the mean to prevent the regression component from shifting towards the outliers.

A comparison of the RM^2 method, the Gaussian mixture regression model using MLE, TLE and the heavy-tailed t-distribution was performed using a simulation study. The MSE for the RM^2 method was overall the lowest.

In the wine cultivars example we established the effect of total phenols on the colour intensity of the wine. The parameter estimates of the RM^2 method were compared to that of the Gaussian mixture regression model. From the results it was evident that the traditional MLE failed in the presence of outliers, whilst the RM^2 method proved to be

robust. It also showed its ability to identify and accommodate outliers.

This research confirms that the RM^2 method is highly robust against outliers.

5.2 Future work

The RM^2 method was fitted to a mixture regression model with two components to illustrate the concept of outlier identification and robust parameter estimation. It will be valuable to determine the performance of this method when fitting it to a finite mixture regression (FMR) problem with more than two components, as the determination of the number of components is an essential step in the process of mixture modelling. BIC was used to measure the model performance based on the tuning parameter. The performance of the RM^2 method can be enhanced by improving the selection of the tuning parameter. When comparing mixture model performance using a simulation study, there are label switching issues which can be solved with different strategies. The examples in this mini-dissertation had one or two feature variables. In practice, a feature selection method needs to be adopted when fitting the RM^2 method. For the purpose of illustration we have used one penalty function, the l_0 penalty. The method could possibly yield different results when making use of other penalty functions.

Bibliography

Xiuqin Bai, Weixin Yao, and John E Boyer. Robust fitting of mixture regression models. *Computational Statistics & Data Analysis*, 56(7):2347–2359, 2012.

Shaheena Bashir and EM Carter. Robust mixture of linear regression models. *Communications in Statistics-Theory and Methods*, 41(18):3371–3388, 2012.

Gilles Celeux, Merrilee Hurn, and Christian P Robert. Computational and inferential difficulties with mixture posterior distributions. *Journal of the American Statistical Association*, 95(451):957–970, 2000.

Andrew R Conn, Nicholas IM Gould, and Philippe Toint. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, 1991.

Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

Luis Angel García-Escudero, Alfonso Gordaliza, Agustín Mayo-Íscar, and Roberto

San Martín. Robust clusterwise linear regression through trimming. *Computational Statistics & Data Analysis*, 54(12):3057–3069, 2010.

Luis Angel García-Escudero, Alfonso Gordaliza, Francesca Greselin, Salvatore Ingrassia, and Agustín Mayo-Íscar. Robust estimation of mixtures of regressions with random covariates, via trimming and constraints. *Statistics and Computing*, 27(2):377–402, 2017.

Neil Gershenfeld. Nonlinear inference and cluster-weighted modeling. *Annals of the New York Academy of Sciences*, 808(1):18–24, 1997.

Stephen Goldfeld and Richard Quandt. A markov model for switching regressions. *Journal of econometrics*, 1(1):3–15, 1973.

Richard J Hathaway. A constrained formulation of maximum-likelihood estimation for normal mixture distributions. *The Annals of Statistics*, 13(2):795–800, 1985.

Hao Hu, Yichao Wu, and Weixin Yao. Maximum likelihood estimation of the mixture of log-concave densities. *Computational statistics & data analysis*, 101:137–147, 2016.

Hao Hu, Weixin Yao, and Yichao Wu. The robust em-type algorithms for log-concave mixtures of regression models. *Computational statistics & data analysis*, 111:14–26, 2017.

Steven G Johnson. The nlopt nonlinear-optimization package (version 2.3). *URL http://ab-initio. mit. edu/nlopt*, 2012.

Marianthi Markatou. Mixture models, robustness, and the weighted likelihood methodology. *Biometrics*, 56(2):483–486, 2000.

Neyko Neykov, Peter Filzmoser, Rumiana Dimova, and Plamen Neytchev. Robust fitting of mixtures using the trimmed likelihood estimator. *Computational Statistics & Data Analysis*, 52(1):299–308, 2007.

Antonio Punzo and Paul D McNicholas. Robust clustering in regression analysis via the contaminated gaussian cluster-weighted model. *Journal of Classification*, 34(2): 249–293, 2017.

Hong-bin Shen, Jie Yang, and Shi-tong Wang. Outlier detecting in fuzzy switching regression models. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications*, pages 208–215. Springer, 2004.

Weixing Song, Weixin Yao, and Yanru Xing. Robust mixture regression model fitting by laplace distribution. *Computational Statistics & Data Analysis*, 71:128–137, 2014.

Matthew Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, 2000.

Weixin Yao. Model based labeling for mixture models. *Statistics and Computing*, 22(2): 337–347, 2012.

Weixin Yao and Bruce G Lindsay. Bayesian mixture labeling by highest posterior density. *Journal of the American Statistical Association*, 104(486):758–767, 2009.

Weixin Yao, Yan Wei, and Chun Yu. Robust mixture regression using the t-distribution. *Computational Statistics & Data Analysis*, 71:116–127, 2014.

Grace Y Yi, Xianming Tan, and Runze Li. Variable selection and inference procedures for marginal analysis of longitudinal data with missing observations and covariate measurement error. *Canadian Journal of Statistics*, 43(4):498–518, 2015.

Jelmer Ypma. Introduction to nloptr: an r interface to nlopt. *R Package*, 2, 2014.

Chun Yu, Kun Chen, and Weixin Yao. Outlier detection and robust mixture modeling using nonconvex penalized likelihood. *Journal of Statistical Planning and Inference*, 164:27–38, 2015.

Chun Yu, Weixin Yao, and Kun Chen. A new method for robust mixture regression. *Canadian Journal of Statistics*, 45(1):77–94, 2017.

Camila B Zeller, Celso RB Cabral, and Víctor H Lachos. Robust mixture regression modeling based on scale mixtures of skew-normal distributions. *Test*, 25(2):375–396, 2016.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of statistics*, 38(2):894–942, 2010.

Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429, 2006.

Appendix A

Acronyms

AIC	Akaike Information Criterion
BDP	break down point
BIC	Bayesian Information Criterion
CWM	cluster-weighted model
EM	Expectation Maximisation
FMR	finite mixture regression
LAD	least absolute deviation
MCP	minimax concave penalty
MLE	maximum likelihood estimation
MSE	mean squared error
MSP	mean-shift parameter
OLS	ordinary least squares
pdf	probability density function
pmf	probability mass function

RM^2	robust mixture regression using mean-shift penalisation
SCAD	smoothly clipped absolute deviations
TLE	trimmed likelihood estimator

Appendix B

Code

An example on how to simulate a random sample with added outliers is given here. The code used to achieve robust mixture regression using mean-shift penalisation in the case of equal component variances is also provided.

```
%let perc=0.05;
%put &perc.;

/*Univariate example*/

/*Equal OR unequal component variances data*/

data data_uni;

seed = 1234;
call streaminit(seed);
p=0.3;
n=200;
a=11;
b=13;
```

```
var1=1;
var2=1;
sigma1=sqrt(var1);
sigma2=sqrt(var2);

perc=&perc.; *percentage of outliers;
it=perc*n;
bound1 = floor(0.25*it);
bound2 = ceil(0.75*it);

do obs = 1 to n;
z = rand("Bernoulli", p);
x1 = rand("Normal",0,1);
/* x2 = rand("Normal",0,1);*/
err1 = rand("Normal",0,sigma1);
err2 = rand("Normal",0,sigma2);
gamma1 = a + (b-a)*rand("Uniform");
gamma2 = a + (b-a)*rand("Uniform");
output;
end;

call symput("bound1",bound1);
call symput("bound2",bound2);
call symput("sigma1",sigma1+1);
call symput("sigma2",sigma2+1);

run;
%put &sigma1.; *Only to be used as starting values in proc IML below;
%put &sigma2.; *Only to be used as starting values in proc IML below;
%put &bound1.;
%put &bound2.;
```

```
/*Randomly select outliers from component1*/
proc surveyselect data=data_uni out=outlier_comp1_uni method=srs
sampsize=&bound1. seed=1234 noprint;
where z=1;
run;
proc sql noprint;
select obs into:outl_ind1 separated by "," from outlier_comp1_uni;
quit;
%put &outl_ind1.;

/*Randomly select outliers from component2*/
proc surveyselect data=data_uni out=outlier_comp2_uni method=srs
sampsize=&bound2. seed=1234 noprint;
where z=0;
run;
proc sql noprint;
select obs into:outl_ind2 separated by "," from outlier_comp2_uni;
quit;
%put &outl_ind2.;


data data_uni_final;
set data_uni;

if z=1 then do; *component 1 ;
if obs in (&outl_ind1.) then do;
x1=2;
gamma1=gamma1;
gamma2=0;
```

```
gamma_ind1=1;
gamma_ind2=0;
y = 1 - 2*x1 - abs(gamma1)*sigma1 + err1;
end;
else do;
x1=x1;
gamma1=0;
gamma2=0;
gamma_ind1=0;
gamma_ind2=0;
y = 1 - 2*x1 + 0*sigma1 + err1;
end;
end;

else do; *if z=0, i.e. component 2 ;
if obs in (&outl_ind2.) then do;
x1=2;
gamma1=0;
gamma2=gamma2;
gamma_ind1=0;
gamma_ind2=1;
y = 1 + 4*x1 + abs(gamma2)*sigma2 + err2;
end;
else do;
x1=x1;
gamma1=0;
gamma2=0;
gamma_ind1=0;
gamma_ind2=0;
y = 1 + 4*x1 + 0*sigma2 + err2;
end;
```

```
end;

if gamma_ind1=1 or gamma_ind2=1 then outlier=1;
else outlier=0;

run;

data data_uni_final_nooutl;
set data_uni;

if z=1 then do; *component 1 ;
x1=x1;
gamma1=0;
gamma2=0;
gamma_ind1=0;
gamma_ind2=0;
y = 1 - 2*x1 + 0*sigma1 + err1;
end;

else do; *if z=0, i.e. component 2 ;
x1=x1;
gamma1=0;
gamma2=0;
gamma_ind1=0;
gamma_ind2=0;
y = 1 + 4*x1 + 0*sigma2 + err2;
end;

if gamma_ind1=1 or gamma_ind2=1 then outlier=1;
else outlier=0;
```

```
run;


dm 'odsresults;clear';
title ;

options nodate  ;
ods noproctitle ;

/*MIXTURE MODELING*/
/***************************/
/*EQUAL COMPONENT VARIANCES*/
/***************************/

proc iml;

use data_uni_final_outl;
read all var {"x1"} into X;
read all var {"y"} into y;
read all var {"gamma1" "gamma2"} into gamma;
read all var {"gamma_ind1" "gamma_ind2"} into gamma_ind;
read all var {"err1" "err2"} into err;
read all var {"outlier"} into outlier_ind;
close;
x_f=repeat(1,nrow(X))||x;  *including intercept;
/*print y x gamma;*/

data_mat = y||x_f;


do s = 1 to nrow(lambda1);
```

```
pi_1path = J(k,initial,.);
beta_1path = J(k,k*nrow(lambda1),.);
sig_1path = J(k,initial,.);
gamma_1path = J(n,initial,.);
gamma_1path_ind = J(n,initial,.);
lh_1 = 0;
obj = 0;

if s > 1 then do; *from the second lambda, the initial=10th set of;
*inital pi, mu, sigma and gamma be the solution of the previous gamma;
pi_0[,initial] = pi_path[,s-1];
beta_0[,(k*initial)-1:(k*initial)] = beta_path[,(k*s)-3:(k*s)-2];
sig_0[,initial] = sig_path[,s-1];
gamma_0[,initial] = gamma_path[,s-1];
end;


beta_new_save=J(ncol(x_f)*initial,ncol(x_f),.);
obj=J(initial,1,.);

do num = 1 to initial;

pi0=pi_0[,num];
beta0=beta_0[,(k*num)-1:(k*num)];
sig0=sig_0[,num];
gamma0=gamma_0[,num];
lambda0=lambda1[s];


***Initialize parameters;
```

```
/************************ NB NB NB NB *********************/
***Choose thresholding rule to be used when determining gamma;
***Choose between "hard" or "soft";

threshold_rule="hard";

/**********************************************************/


n=nrow(y);
k=ncol(x_f);
pi=pi0;
beta=beta0;
sigma=sig0; *component variances/std deviations;
sigma_t = sigma‘;
sigma_rep_t = repeat(sigma‘,nrow(X));
iter=nrow(pi);
gamma=gamma0; *mean-shift parameters;
gamma_ind=J(nrow(x_f),1,0);
do i=1 to nrow(x_f);
/* do j=1 to k; */
if gamma[i]^=0 then gamma_ind[i]=1;
else gamma_ind[i]=0;
/* end;*/
end;
lambda=lambda0; *tuning parameter controlling the degrees of penalization;

/* beta_new_save=J(initial+1,ncol(x_f),0);*/

check=J(nrow(y),k,0);
/* check=y-x_f*beta-gamma#sigma_rep_t; *n x m;*/
```

```
check=y-x_f*beta-gamma; *n x m;
**********************;


***Starting value of the log-likelihood;
denom = J(nrow(X),1,.);
Q0 = J(nrow(X),k,.);
pij0 = J(nrow(X),k,.);
pij = J(nrow(X),k,.);
/* lg_pdf = J(nrow(X),k,.);*/
/* lg_pi_t = repeat((log(pi))‘,nrow(X));*/
penalty = J(nrow(X),1,0);
**************************************;


***Initialize the log-likelihood;
/* LLH[1] = 0; */
LLH_diff=0;
diff_beta=0;
diff_sigma=0;
diff_gamma=0;
count=0;
**********************;



***Initialize the Expectation step (E-step);
sums=J(nrow(x),k,.);

do i = 1 to nrow(X);
do j = 1 to k;
```

```
sums[i,j] = pi[j]*pdf("Normal", check[i,j], 0, sigma_t[j]);
end;

/* ind_sum = ind_sum + gamma_old[i,j];*/
if threshold_rule="soft" then do;
penalty[i]=lambda*abs(gamma[i]);
end;
if threshold_rule="hard" then do;
penalty[i]=((lambda##2)/2)*gamma_ind[i];
end;
end;
tot=sums[,+];



lh=J(nrow(x),k,0);
s_lh=J(nrow(x),k,0);
l_lh=J(nrow(x),k,0);
do i = 1 to nrow(X);

do j = 1 to k;

pij0[i,j] = pi[j]*max(10e-100, pdf("Normal", check[i,j], 0, sigma_t[j]));
pij[i,j] = pij0[i,j]/tot[i];

/* Likelihood */
lh[i,j] = pi[j]*max(10e-100, pdf("Normal", check[i,j], 0, sigma_t[j]));


end;
```

```
end;


s_lh = lh[,+];
l_lh = log(s_lh);
/* Q-function */
Q = l_lh[+] - penalty[+];
/* print Q;*/



***EM loop;

LLH0=LLH[iter];
/* run=run+1;*/
diff=0;
oldpi=pi;
oldbeta=beta;
oldsigma=sigma; *component variances/std deviations;
oldsigma_t = oldsigma`;
oldsigma_rep_t = repeat(oldsigma`,nrow(X));
iter=nrow(oldpi);
oldgamma=gamma; *mean-shift parameters;
oldgamma_ind=J(nrow(x_f),1,0);
do i=1 to nrow(x_f);
/* do j=1 to k; */
if oldgamma[i]^=0 then oldgamma_ind[i]=1;
else oldgamma_ind[i]=0;
/* end;*/
end;
```

```
check=J(nrow(y),k,0);
/* check=y-x_f*oldbeta-oldgamma#oldsigma_rep_t; *n x m;*/
check=y-x_f*oldbeta-oldgamma; *n x m;
**********************;


do until(diff < 0.1);

***Expectation step (E-step);
sums=J(nrow(x),k,.);

do i = 1 to nrow(X);
do j = 1 to k;
sums[i,j] = oldpi[j]*pdf("Normal", check[i,j], 0, oldsigma_t[j]);
end;

/* ind_sum = ind_sum + gamma_old[i,j];*/
if threshold_rule="soft" then do;
penalty[i]=lambda*abs(oldgamma[i]);
end;
if threshold_rule="hard" then do;
penalty[i]=((lambda##2)/2)*oldgamma_ind[i];
end;
end;
tot=sums[,+];

lh=J(nrow(x),k,0);
s_lh=J(nrow(x),k,0);
l_lh=J(nrow(x),k,0);
do i = 1 to nrow(X);
```

```
do j = 1 to k;

pij0[i,j] = oldpi[j]*max(10e-100, pdf("Normal", check[i,j], 0, oldsigma_t[j]));
pij[i,j] = pij0[i,j]/tot[i];


/* Likelihood */
lh[i,j] = oldpi[j]*max(10e-100, pdf("Normal", check[i,j], 0, oldsigma_t[j]));


end;

end;

***Update the log-likelihood;
s_lh = lh[,+];
l_lh = log(s_lh);
/* Q-function */
Q0 = l_lh[+] - penalty[+];
/* print Q0;*/

**********************;


***Maximization step (M-step);
n_ks=pij[+,];
pi_new=J(k,1,0);
do j= 1 to k;
pi_new[j] = n_ks[j]/nrow(X);
end;
/* print pi_new;*/
```

```
/* ***Calculate Beta;*/
beta_new=J(ncol(x_f),k,0);
do j=1 to k;

w=diag(pij[,j]);
/* beta_new[,j] = inv(x_f‘*w*x_f)*x_f‘*w*(y-oldgamma#sigma_rep_t[,j]);*/
beta_new[,j] = ginv(x_f‘*w*x_f)*x_f‘*w*(y-oldgamma);

end;
/* print beta_new ;*/



R_data1_uni=pij||y||x_f||gamma;
R_data2_uni=beta_new‘||beta‘||pi_new||pi||sigma;

create R_data1_uni from R_data1_uni[colname={"pi1" "pi2" "y" "x_f1" "x_f2"
"gamma_old1" "gamma_old2"}]; /** create data set **/
append from R_data1_uni;        /** write data in vectors **/
close R_data1_uni;  /** close the data set **/

create R_data2_uni from R_data2_uni[colname={"B0" "B1" "B0_old" "B1_old"
"pi_new" "pi_old" "sigma_old"}]; /** create data set **/
append from R_data2_uni;        /** write data in vectors **/
close R_data2_uni;  /** close the data set **/



***Calculate Sigma;
```

```
call ExportDataSetToR("R_data1_uni", "df1");
call ExportDataSetToR("R_data2_uni", "df2");

submit / r;


# Nonlinear optimization problem
# install.packages("nloptr")
library(nloptr)


#library(readxl)

n <- nrow(df1)
m <- nrow(df2)



# Will work if all pi_s are together
pij <- NULL
for(i in 1:m){
startcol <- which(names(df1) == "pi1")
endcol <- as.integer(startcol + m - 1)
pij <- as.matrix(df1[,startcol:endcol])
}

yi <- NULL
for(i in 1:m){
startcol <- which(names(df1) == "y")
# endcol <- as.integer(startcol + m)
yi <- as.matrix(unlist(df1[,startcol]))
```

```
}

x_f <- NULL
for(i in 1:m){
startcol <- which(names(df1) == "x_f1")
endcol <- as.integer(startcol + 1)
x_f <- as.matrix(df1[,startcol:endcol])
}

gamma <- NULL
for(i in 1:m){
startcol <- which(names(df1) == "gamma_old1")
#endcol <- as.integer(startcol + m - 1)
#gamma <- as.matrix(df1[,startcol:endcol])
gamma <- as.matrix(unlist(df1[,startcol]))
}

beta_new <- NULL
for(i in 1:m){
startcol <- which(names(df2) == "B0")
endcol <- as.integer(startcol + 1)
beta_new <- as.matrix(df2[,startcol:endcol])
}
beta_newt <- t(beta_new)

beta_old <- NULL
for(i in 1:m){
startcol <- which(names(df2) == "B0_old")
endcol <- as.integer(startcol + 1)
beta_old <- as.matrix(df2[,startcol:endcol])
}
```

```
beta_oldt <- t(beta_old)


pi_new <- NULL
for(i in 1:m){
startcol <- which(names(df2) == "pi_new")
# endcol <- as.integer(startcol + m)
pi_new <- as.matrix(unlist(df2[,startcol]))
}
pi_newt <- t(pi_new)


pi_old <- NULL
for(i in 1:m){
startcol <- which(names(df2) == "pi_old")
# endcol <- as.integer(startcol + m)
pi_old <- as.matrix(unlist(df2[,startcol]))
}
pi_oldt <- t(pi_old)


sigma <- NULL
for(i in 1:m){
startcol <- which(names(df2) == "sigma_old")
# endcol <- as.integer(startcol + m)
sigma <- as.matrix(unlist(df2[,startcol]))
}
sigmat <- sigma[1]
#sigmatf <- matrix(rep(sigmat,n), ncol=ncol(sigmat),byrow=T)
```

```
# Objective function

eval_f <- function(sigma)
{
pij <- pij
yi <- yi
x_f <- x_f
beta_newt <- beta_newt
gamma <- gamma
n <- n
m <- m
func <- matrix(rep(0,n*m),ncol=m)
for(i in 1:n)
{
for(j in 1:m)
{

func[i,j] <- -1*pij[i,j]*log(dnorm(yi[i] - x_f[i,]%*%beta_newt[,j] - gamma[i]*sigmat,
0, sigmat ))
func[i,j] <- replace(func[i,j], is.na(func[i,j]), 0)
func[i,j] <- replace(func[i,j], func[i,j]==-Inf, 0)
func[i,j] <- replace(func[i,j], func[i,j]==Inf, 100000)

}
}

final_func <- sum(func)

return (final_func)
```

```
}



# Starting parameter values
x0 <- as.numeric(sigmat)


# Lower and upper bounds
lb <- rep(0.01, 1)
ub <- rep(20, 1)



local_opts <- list( "algorithm" = "NLOPT_LN_COBYLA",
"xtol_rel" = 1.0e-4, "print_level" = 3 )



res1 <- nloptr(x0=x0,
eval_f=eval_f,
# eval_grad_f=grad.norm,
lb = lb,
ub = ub,
# pij=pij,
# yi=yi,
# x_f=x_f,
# beta_newt=beta_newt,
# gamma=gamma,
# n=n,
# m=m,
```

```
# eval_g_ineq = eval_g0,
opts = local_opts)


res1
sigma_new0 <- sqrt(res1$solution)
sigma_new0[sigma_new0<0.1] <- 0.1


endsubmit;


call ImportMatrixFromR(sigma_new0, "sigma_new0");

sigma_new = J(2,1,sigma_new0);
sigma_new_t = sigma_new‘;
sigma_new_rep_t = repeat(sigma_new_t,nrow(X));
sigma_new_t1 = 1/(sigma_new‘);
sigma_new_rep = repeat(sigma_new_t1,n);

/* print sigma_new;*/

***Calculate Gamma;
/* zi = (y-x_f*beta_new)#sigma_new_rep; *sigma_new_rep = 1/sigma_new; */


w0 = J(nrow(X),k,0);
do j = 1 to k;
w0[,j] = pij[,j]/(sigma_new[j,])##2;
end;
w = w0[,+];
```

```
zi0 = ((pij#(y-x_f*beta_new))/(2#sigma_new_t##2))/(0.5#w);
zi = zi0[,+];


lambda_star=J(nrow(X),k-1,0);
do i = 1 to nrow(X);

***lambda to be used in soft thresholding rule (l1 penalty);
if threshold_rule="soft" then do;
/* if pij[i,j]=0 then lambda_star[i]=0 else;*/
lambda_star[i] = lambda/w[i];
end;

***lambda to be used in hard thresholding rule (l0 penalty);
if threshold_rule="hard" then do;
/* if pij[i,j]=0 then lambda_star[i,j]=0; else*/
lambda_star[i] = lambda/sqrt(w[i]);
end;

end;


***Indicator function used in hard thresholding rule;
dum=J(nrow(X),k-1,.);
gamma_new_ind=J(nrow(X),k-1,.);
do i = 1 to nrow(X);
/* do j = 1 to k;*/
dum[i]=(abs(zi[i])-lambda_star[i]);

if dum[i]>0 then gamma_new_ind[i]=1;
else gamma_new_ind[i]=0;
```

```
/* end;*/
end;
/* print zi dum ind lambda_star;*/

***Soft thresholding rule (l1 penalty);
if threshold_rule="soft" then gamma_new = sign(zi)#((abs(zi)-lambda_star) <> 0);


***Hard thresholding rule (l0 penalty);
if threshold_rule="hard" then gamma_new = zi#gamma_new_ind;
/* print gamma_new;*/


***Update the log-likelihood;
check=y-x_f*beta_new-gamma_new; *n x m;
/* print check;*/

sums=J(nrow(x),k,.);

do i = 1 to nrow(X);

if threshold_rule="soft" then do;
penalty[i]=lambda*abs(gamma_new[i]);
end;
if threshold_rule="hard" then do;
penalty[i]=((lambda##2)/2)*gamma_new_ind[i];
end;

end;
```

```
lh=J(nrow(x),k,0);
s_lh=J(nrow(x),k,0);
l_lh=J(nrow(x),k,0);
check_pdf=J(nrow(x),k,0);

do i = 1 to nrow(x);
do j = 1 to k;
/* Likelihood */
lh[i,j] = pi_new[j]*max(10e-100, pdf("Normal", check[i,j], 0, sigma_new[j]));

end;

end;

s_lh = lh[,+];
l_lh = log(s_lh);

/* Q-function */
Q = l_lh[+] - penalty[+];
diff = Q - Q0 ;
/* print s_lh, Q diff;*/



***Set old parameters (step k) equal to new (step k+1) and repeat;
oldbeta = beta_new;
oldsigma = sigma_new;
oldsigma_t = oldsigma`;
oldsigma_rep_t = repeat(oldsigma`,nrow(X));

oldgamma = gamma_new;
```

```
oldgamma_ind=J(nrow(X),1,0);
do i=1 to nrow(X);
/* do j=1 to k; */
if oldgamma[i]^=0 then oldgamma_ind[i]=1;
else oldgamma_ind[i]=0;
/* end;*/
end;


oldpi = pi_new;
iter=iter+1;
check = J(nrow(y),k,0);
check = y - x_f*oldbeta - oldgamma;
pij0 = J(nrow(X),k,.);
pij = J(nrow(X),k,.);
count=count+1;
penalty = J(nrow(X),k,.);

/* print diff;*/

end; *EM loop;

beta_new_save[(2*num)-1:(2*num),] = beta_new;
/* beta_new_savef = beta_new_save[2:(initial+1),];*/
print beta_new_save;
**************************;

pi_1path[,num] = pi_new;
beta_1path[,(2*num)-1:(2*num)] = beta_new;
sig_1path[,num] = sigma_new;
gamma_1path[,num] = gamma_new;
```

```
gamma_1path_ind[,num] = (gamma_1path[,num]^=0)[,+];


obj[num]= Q;
llh[num] = l_lh[+];


end; *num;

sol_llh = obj[<:>];
/* print llh sol_llh;*/

pi_path[,s] = pi_1path[,sol_llh];
beta_path[,(k*s)-1:(k*s)] = beta_new_save[(sol_llh*2)-1:(sol_llh*2),];
sig_path[,s] = sig_1path[,sol_llh];
gamma_path[,s] = gamma_1path[,sol_llh];
gamma_path_ind[,s] = gamma_1path_ind[,sol_llh];
df_lambda[s] = sum(gamma_path_ind[,s]);
BIC[s] = -1*llh[sol_llh] + (log(n))*(df_lambda[s]+(3*k-1));


end; *s;
print pi_path beta_path sig_path gamma_path gamma_path_ind df_lambda;
print s BIC;

final_sol = BIC[>:<];

pi_final = pi_path[,final_sol];
beta_final = beta_path[,(final_sol*2)-1:(final_sol*2)];
sig_final = sig_path[,final_sol];
gamma_final = gamma_path[,final_sol];
```

```
gamma_ind_final = gamma_path_ind[,final_sol];
lambda_final = lambda1[final_sol];

print final_sol pi_final beta_final sig_final gamma_final lambda_final;
quit;
```