# ROBUST MIXTURES OF REGRESSION MODELS

by

XIUQIN BAI

Ph.D., Chinese Art Academic Institute, China, 2004

---

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2010

Approved by:

Major Professor
Weixin Yao

# Copyright

**Abstract**

In the fitting of mixtures of linear regression models, the normal assumption has been traditionally used for the error term and then the regression parameters are estimated by the maximum likelihood estimate (MLE) using the EM algorithm. Under the normal assumption, the M step of the EM algorithm uses a weighted least squares estimate (LSE) for the regression parameters. It is well known that the LSE is sensitive to outliers or heavy tailed error distributions. In this report, we propose a robust mixture of linear regression model, which replaces the least square criterion with some robust criteria in the M step of the EM algorithm. In addition, we will use a simulation study to demonstrate how sensitive the traditional mixture regression estimation method is to outliers or heavy tailed error distributions and compare it with our proposed robust mixture regression estimation method. Based on our empirical studies, our proposed robust estimation method works comparably to the traditional estimation method when there are no outliers and the error is normally distributed but is much better if there are outliers or the error has heavy tails (such as t-distribution). A real data set application is also provided to illustrate the effectiveness of our proposed methodology.

# Contents

**Table of Contents** iv

**List of figures** v

**List of tables** vi

**1 Introduction** 1

1.1 Mixture Model . . . 1

1.1.1 History . . . 1

1.1.2 Definition . . . 2

1.1.3 EM Algorithm . . . 3

1.2 Robust Regression . . . 5

1.2.1 History and Application . . . 5

1.2.2 Robust Methods . . . 6

**2 Robust Mixtures Of Linear Regression Models** 11

2.1 Introduction . . . 11

2.2 Robust Estimation Method . . . 14

**3 Simulation Studies and Real Data Application** 17

**4 Conclusion and Future Work** 27

**A R Code** 31

# List of Figures


1.1 Huber's $\psi$ function . . . . . 8

1.2 Bisqure's $\psi$ function . . . . . 9

2.1 The scatter plot of the tone perception data and the fitted two lines by our proposed method. The predictor is actual tone ratio and the response is the perceived tone ratio by a trained musician. . . . . 12

3.1 Case 1: Standard normal distribution . . . . . 18

3.2 Case 2: T distribution with 4 degree freedom . . . . . 19

3.3 Case 3: Contaminated mixture distribution . . . . . 20

3.4 Robust regression and non robust regression for tone data with outliers . . . . . 24

3.5 Robust regression and non robust regression for tone data with increased outliers . . . . . 25

# List of Tables


3.1 MSE's of Case 1: $\epsilon \sim$ Standard normal distribution . . . . . 21

3.2 Efficiency of Case 1: $\epsilon \sim$ Standard normal distribution . . . . . 21

3.3 MSE's of Case 2: $\epsilon \sim$ T distribution with 4 degree freedom . . . . . 22

3.4 Efficiency of Case 2: $\epsilon \sim$ T distribution with 4 degree freedom . . . . . 22

3.5 MSE's of Case 3: $\epsilon \sim 0.95N(0,1) + 0.05N(0,5^2)$ . . . . . 23

3.6 Efficiency of Case 3: $\epsilon \sim 0.95N(0,1) + 0.05N(0,5^2)$ . . . . . 23

# Chapter 1

# Introduction

## 1.1 Mixture Model

### 1.1.1 History

The history of Finite Mixture Models goes back to more than 100 years ago. In the middle of the 19th century, Quetelet did some work involving mixture model which was mentioned explicitly by Galton. More than 30 years latter, Holmes (1892) introduced in the concept of mixtures of populations. The classic paper on mixture models is by the famous biometrician Pearson (1894). This classic paper was the first two monstrous memoirs in a series of "Contributions to the Mathematical Theory of Evaluation "( see Stigler(1986), Chapter 10). Pearson is one of the first major analysts involving the use of mixture models. He used a moment based method to fit a mixture of two heteroscedastic normal components in the paper. The moment based method to fit a mixture by Pearson is pretty tedious. He obtained his moment based estimates of 5 parameters ($\mu_1$ $\mu_2$ $\sigma_1$ $\sigma_2$ $\pi$) by solving a ninth degree polynomial.

The data Pearson worked on in 1894 was provided by the zoologist Walter Frank Raphael Weldon. He had speculated in 1893 that asymmetry in the histogram of these ratios could signal evolutionary divergence. Pearson's approach was to fit a univariate mixture of two normals to the data by choosing the five parameters of the mixture such that the empirical moments matched those of the model.

Although Pearson's method means more work , actually it was still used widely until a new approach was found about 30 years into of the 20th century. But at the same time, some effort aiming at simplifying and extending the method was made. For example, Charlier and Wicksell (1924) extended it to the bivariate normal component case, Doetsch (1928) used it in the case of more than two univariate normal components, Stromgren (1934) considered the use of cumulants, and Rao expanded the use to statistics. Much recently, Cohen (1967) did some work on solving Pearson's ninth degree polynomial by an iterative process.

Considerable advances have been made in the fitting of finite mixture models in last 20 years or so, and maximum likelihood, with the advent of EM algorithm, is one of the most important methods. It is still by far the most widely used approach to the fitting of finite mixture distributions. ML estimation was considered by Tan, Chang, Fryer and Robertson (1972), among some others, for solving Pearson's ninth order polynomial. This method has since been focused and reconsidered by researchers.

### 1.1.2 Definition

In statistics, a mixture model is a probabilistic model for density estimation using a mixture distribution. Mixture model have experienced increased interest over last decades. They are related to cluster analysis, discriminant analysis, survival analysis. Mixture model have applications in a number of areas such including finance, medicine, agriculture and others.

Here is an example: Financial returns often behave differently in normal situations and during crisis times. A mixture model for return data seems reasonable. Some researchers model this is a jump-diffusion model, or as a mixture of two normal distributions.

Finite mixture models are most commonly used for mixture models. They are applied to one sample from a population which consists of several homogeneous or heterogeneous subpopulations. These populations will be called components of the population. Let $(X_1...,X_n)$ be a sample from a population and $z_i$ be the index number of the component to which $X_i$ belongs. The number of components will generally be denoted by $m$, which can be known or unknown. Denote the density function of $j$th component by $f(x;\lambda_j)$, $j = 1, 2, ..., m$. where $\lambda_j$ is a component specific parameter, which can be scalar or vector. For simple notation, $f(\cdot)$ can be the density of discrete or continuous random variables and generally is called *component density*. We have

$$p(x_i|z_i = j) = f(x_i;\lambda_j)(i = 1, ..., n; j = 1, ..., m).$$

where $p(\cdot)$ is the general notation for the density function of both discrete or a continuous probability distribution. The proportion of the total population that is in the $j$th component will be denoted by $\pi_j$ and called the *component weight*. We have

$$P(z_i = j) = \pi_j(i = 1, ..., n; j = 1, ..., m),$$

where $0 \leq \pi_j \leq 1$,$j = 1, ..., m$, and $\sum_{j=1}^{m} \pi_j = 1$. If we could observe the component labels $(z_i, i = 1, ..., n)$, then the variables $(X_i, z_i), i = 1, ..., n.$ are a random sample from the joint density of the form

$$p(x, j) = p(x|z = j)P(z = j) = f(x;\lambda_j)\pi_j.$$

If the component labels $(z_1, ..., z_n)$ are missing, we only observe $(X_1, ..., X_n)$ from the marginal density of $X$, which is the *mixture* density

$$p(x;\theta) = \sum_{j=1}^{m} p(x|z = j)P(z = j) = \sum_{j=1}^{m} \pi_j f(x;\lambda_j), \tag{1.1}$$

where $\theta = (\lambda_1, ..., \lambda_m, \pi_1, ..., \pi_m)$. A family of distributions having density (1.1) are called an $m-component$ $finite\ mixture\ model$. When $m = 1$, there is just one component and $p(\cdot)$ will be called *unicomponent* mixture density. To estimate the unknown parameters $\theta = (\lambda_1 ..., \lambda_m, \pi_1, ..., \pi_m)$ in (1.1), we generally use the maximum likelihood estimator (MLE):I.e, we find $\theta$ to maximize

$$l(\theta; x) = \sum_{i=1}^{n} \log \sum_{j=1}^{m} \pi_j f(x_i; \lambda_j).$$

Since there is no explicit formula for the MLE, we can use EM algorithm to find the MLE for finite mixture models.

### 1.1.3 EM Algorithm

EM algorithm refers to expected-maximization algorithm. It is used for finding maximum likelihood estimates of parameters in probabilistic models, where the model depends on unobserved latent variables. EM is an iterative method which alternates between performing an expectation E step, which computes an expectation of the log likelihood with respect to the current estimate of the distribution for the latent variables, and a maximization M step, which computes the parameters which maximize the expected log likelihood found on the E step. These parameters are then used to determine the distribution of the latent variables in the next E step. Suppose the observed data $x = (x_1, ..., x_n)$ are from $\mathrm{g}(x|\theta)$, where the $\mathrm{g}(x|\theta)$ is any density with parameter $\theta$. The MLE of $\theta$ is $\hat{\theta} = \arg\max_\theta l(\theta)$, where $l(\theta) = \sum_{i=1}^{n} \log\ \mathrm{g}(x_i|\theta)$. Suppose there is a simple or closed form of the ML estimator for the data $y = (x, z)$ with log likelihood $l_c(\theta)$, where $z = (z_1, ..., z_n)$ are the unobserved (or missing) data. We can use EM algorithm to find the $\hat{\theta}$. The EM algorithm is performed as follows:

**1.** Start with initial values $\theta^{(0)}$.

**2.** E step: At the $(k+1)^{th}$ step, compute the expected value of the log-likelihood function, with respect to the conditional distribution of $z$ given $x$ under the current estimate of the parameters $\theta^{(k)}$. Thus

$$Q(\theta, \theta^{(k)}) = E_{\theta^{(k)}}(l_c(\theta)|x). \tag{1.2}$$

**3.** M step: Determine the new estimate $\theta^{(k+1)}$ as the maximizer of $Q(\theta, \theta^{(k)})$ over $\theta$.

$$\theta^{(k+1)} = \arg\max_{\theta} Q(\theta, \theta^{(k)}). \tag{1.3}$$

**4.** Iterate steps 2 and 3 until some convergence criterion is attained.

For finite normal mixture models with $\lambda_j = (\mu_j, \sigma_j^2)$, we can define the missing data as

$$Z_{ij} = \begin{cases} 1, & \text{if the } ith \text{ observation is from the } jth \text{ component;} \\ 0, & \text{otherwise.} \end{cases} \quad 1$$

Then the log-likelihood for the complete data is

$$l_c(\theta) = \sum_{i=1}^{n}\sum_{j=1}^{m} z_{ij}\log(\pi_j\phi(x_i;\mu_j,\sigma_j^2)), \tag{1.4}$$

where $\phi(x;\mu,\sigma^2)$ is the normal density for $N(\mu,\sigma^2)$, i.e..

$$\phi(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\}.$$

Note that $l_c(\theta)$ is a linear function of $z_{ij}$. Therefore, in order to calculate $E_{\theta^{(k)}}(l_c(\theta)|x)$ in the E step, we only need to calculate

$$p_{ij}^{(k+1)} = E_{\theta^{(k)}}(Z_{ij}|x).$$

In the M step, we need to maximize

$$\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\log[\pi_j\phi(x_i;\mu_j,\sigma_j^2)] = \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\log(\pi_j) + \sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}^{(k+1)}\log[\phi(x_i;\mu_j,\sigma_j^2)]. \tag{1.5}$$

Since $\phi$ is a normal density, the above criterion has explicit solutions.

Therefore, the EM algorithm for normal mixture is as follows:

**1.** Take initial guesses of the parameters, say $\mu_j^{(0)}, \sigma_j^{2(0)}, \pi_j^{(0)}, j = 1, ..., m$.

**2.** E step: At the $(k+1)th$ step, compute the classification probability that the observation $x_i$ comes from component $j$ based on the current estimates.

$$p_{ij}^{(k+1)} = E(z_{ij}|x,\theta^{(k)}) = \frac{\pi_j^{(k)}\phi(x_i;\mu_j^{(k)},\sigma_j^{2(k)})}{\sum_{j=1}^{m}\pi_j^{(k)}\phi(x_i;\mu_j^{(k)},\sigma_j^{2(k)})}, i = 1, ..., n, j = 1, ..., m. \tag{1.6}$$

**3.** M step: Compute the component means, variance and the mixing probability:
That is to find $\theta^{(k+1)}$ that can maximize $E_{\theta^{(k)}}(l(\theta)^{(k+1)}|x)$ in (1.5)

$$\mu_j^{(k+1)} = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)} x_i}{\sum_{i=1}^{n} p_{ij}^{(k+1)}}. \tag{1.7}$$

$$\sigma_j^{2(k+1)} = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}(x_i - \mu_j^{(k+1)})^2}{\sum_{i=1}^{n} p_{ij}^{(k+1)}}. \tag{1.8}$$

$$\pi_j^{(k+1)} = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{n}. \tag{1.9}$$

for $j = 1, ..., m$.

**4.** Iterate step 2 and 3 until convergence.

## 1.2 Robust Regression

### 1.2.1 History and Application

Robust regression is a form of regression analysis designed to circumvent some limitations of traditional parametric and non-parametric methods. Robust methods have a long history that can be tracked back at least to the end of the nineteenth century with Simon Newcomb (see Stigler, 1973). However the first great steps forward occurred in the 1960s and the early 1970s with the fundamental work of Tukey (1960, 1962), Huber (1964, 1967) and Frank Hampel (1971, 1974). The applicability of the new robust methods proposed by these researchers was made possible by the increased speed and accessibility of computers. In the last four decades the field of robust statistics has experienced substantial growth as a research area, and a large number of articles were published. Influencial books have been written by Huber(1981), Hampel,(1986), Rousseeuw (1987), Staudte(1990).

Why are robust statistics needed? All statistical methods rely explicitly or implicitly on a number of assumptions. The most widely used model formalization is the assumption that the observed data have a *normal* (Gaussian) distribution. This assumption has been present in statistics for two centuries, and has been the framework for most of the classical methods in regression, analysis of variance and multivariate analysis. There have been attempts to justify the assuming of normality with theoretical arguments, such as the central limit theorem. The main justification for assuming a normal distribution is that it is theoretically quite convenient because it allows one to derive explicit formulas for optimal statistical methods such as maximum likelihood and likelihood ratio tests, as well as the sampling distribution of inference quantities such as $t$-statistics. We can call those the *classical* statistical methods, and note that they rely on the assumption that normality holds exactly.

It often happens in practice that an assumed normal distribution model holds approximately in that it describes the majority of observations, but some observations follow a different pattern or no pattern at all. In the case when the randomness in the model is assigned to observational errors, the reality is that while the behavior of many sets of data appeared rather normal, this is held only approximately. There are

still a small proportion of observations which are quite atypical by virtue of being far from the bulk of the data. We call such atypical data "outliers". The kind of"approximately" normal distribution that gives rise to outliers is one that has a normal shape in the central region, but has tails that are heavier or "fatter" than those of normal distribution. We might expect that if such approximate normality holds, then the results of using a n inference tool based on normal distribution theory would also hold approximately. This is unfortunately not true. If the data are assumed to be normally distributed but their actual distribution has heavy tails, then estimates based on the maximum likelihood principle not only cease to be"best" but may have unnecessarily large variance if the tails are symmetric and may have very large bias if the tails are asymmetric.

The robust approach to statistical modeling and data analysis aims at deriving methods that produce reliable parameter estimates, not only when the data follow a given distribution exactly, but also when this happens only approximately in the sense just described above. A more informal data-oriented characterization of robust methods is that they fit the bulk of the data well: if the data contain no outliers, the robust method gives approximately the same results as the classical method, while if a small proportion of outliers are present, the robust method gives approximately the same results as the classical method applied to the"typical" data . As a consequence of fitting the bulk of the data well, robust methods provide a very reliable method of detecting outliers, even in high-dimensional multivariate situations.

### 1.2.2 Robust Methods

Given the observations $\{(x_i, y_i), i = 1, \ldots, n\}$, where $\underline{x}_i$ is a $p$-dimension column vector, we assume that they are linked by the linear relationship

$$y_i = \underline{x}_i^T \beta + \epsilon_i$$

where $\underline{x}_i = (1, x_i^T)^T$, $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^T$, $\{\epsilon_1, \ldots, \epsilon_n\}$ are i.i.d. unobservable random errors with $E(\epsilon_i) = 0, \mathrm{Var}(\epsilon_i) = \sigma^2$, and $x_i$ and $\epsilon_i$ are independent.

The simplest and most widely used estimator of the parameter $\beta$ is the ordinary least squares estimator (LSE)

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \underline{x}_i^T \beta)^2. \tag{1.10}$$

It is well known that the LSE has minimum variance among unbiased estimators and is the most efficient estimator if error is normal. However the least squares criterion is extremely sensitive to outliers and the heavy tailed error distribution. Many robust estimators has been proposed over the last 30 years.

One approach to robustly estimating the regression parameters is to replace the normal distribution with a heavy-tailed distribution. A t-distribution with between 4 and 6 degrees of freedom has been reported to be a good choice in various practical situations. Bayesian robust regression, being fully parametric, relies heavily on such distributions.

Under the assumption of t-distributed errors, the distribution is a location-scale family. The degrees of freedom of the t-distribution is sometimes called the kurtosis parameter. Lange, Little and Taylor (1989) discuss this model in some depth from a non-Bayesian point of view. A Bayesian account appears in Gelman et al (2003).

An alternative parametric approach is to assume that the errors follow a mixture of normal distributions; in particular, a contaminated normal distribution in which the majority of observations are from a specified normal distribution, but a small proportion are from a normal distribution with much higher variance. That is, errors have probability $1 - \varepsilon$ of coming from a normal distribution with variance $\sigma^2$, where $\varepsilon$ is small, and probability $\varepsilon$ of coming from a normal distribution with variance $c\sigma^2$ for some $c > 1$.

$$\epsilon_i \sim (1 - \varepsilon)N(0, \sigma^2) + \varepsilon N(0, c\sigma^2).$$

Typically, $\varepsilon < 0.1$. This is sometimes called the contamination model.

Parametric approaches have the advantage that likelihood theory provides an "off the shelf" approach to inference (although for mixture models such as the -contamination model, the usual regularity conditions might not apply), and it is possible to build simulation models from the fit. However, such parametric models still assume that the underlying model is literally true. As such, they do not account for skewed errors distributions or finite observation precisions

Among various robust estimators which don't assume any parametric density for error, there are three broad classes: L estimators based on the linear combinations of order statistics, R estimators based on ranking techniques, and M estimators based on minimizing a loss function. The M-estimators are the most flexible and commonly used ones. They can be easily generalized to multiparameter problems. The M-estimators replace the least squares criterion with one based on an outlier resistant function, $\rho(\cdot)$, of the form

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} \rho(y_i - \underline{x}_i^T \beta), \tag{1.11}$$

where $\rho(\cdot)$ is an outlier resistant loss function. Denote $\psi(\cdot) = \rho'(\cdot)$. Generally if $\psi(\cdot)$ exists and is bounded, then $\rho(\cdot)$ is an outlier resistant loss function. If we use the $L_1$ loss function $\rho(t) = |t|$, we will get the commonly used robust regression method — median regression. One of the unctions commonly used in this setting is Huber's $\psi$-function (Huber 1981) $\psi_k(t) = \rho'(t) = \max\{-k, \min(k, t)\}$, i.e.

$$\psi_k(t) = \begin{cases} k & \text{where } t > k; \\ t & \text{where } |t| \leq k; \\ -k & \text{where } t < -k. \end{cases} \tag{1.12}$$

If $k = \infty$, the function $\rho(\cdot)$ becomes the $L_2$-loss, and as $k \to 0$, it becomes the $L_1$ loss when suitably

normalized. Another possibility for $\psi(\cdot)$ is Turkey's bisquare function $\psi_k(t) = t\{1-(t/k)^2\}_+^2$, i.e.,

$$\psi_k(t) = \begin{cases} 0 & \text{if } |\frac{t}{k}| > 1; \\ t(1-(\frac{t}{k})^2)^2 & \text{if } |\frac{t}{k}| < 1. \end{cases} \tag{1.13}$$

which weighs down the tail contribution of $t$ by a biweight function. For more detail, see Huber (1973, 1981), Andrews (1974), Beaton and Tukey (1974), Holland and Welsch (1977), and Hampel (1986). Figure 3.1 and 3.2 display the *Huber's* $\psi$ function and *bisquare* $\psi$ function.

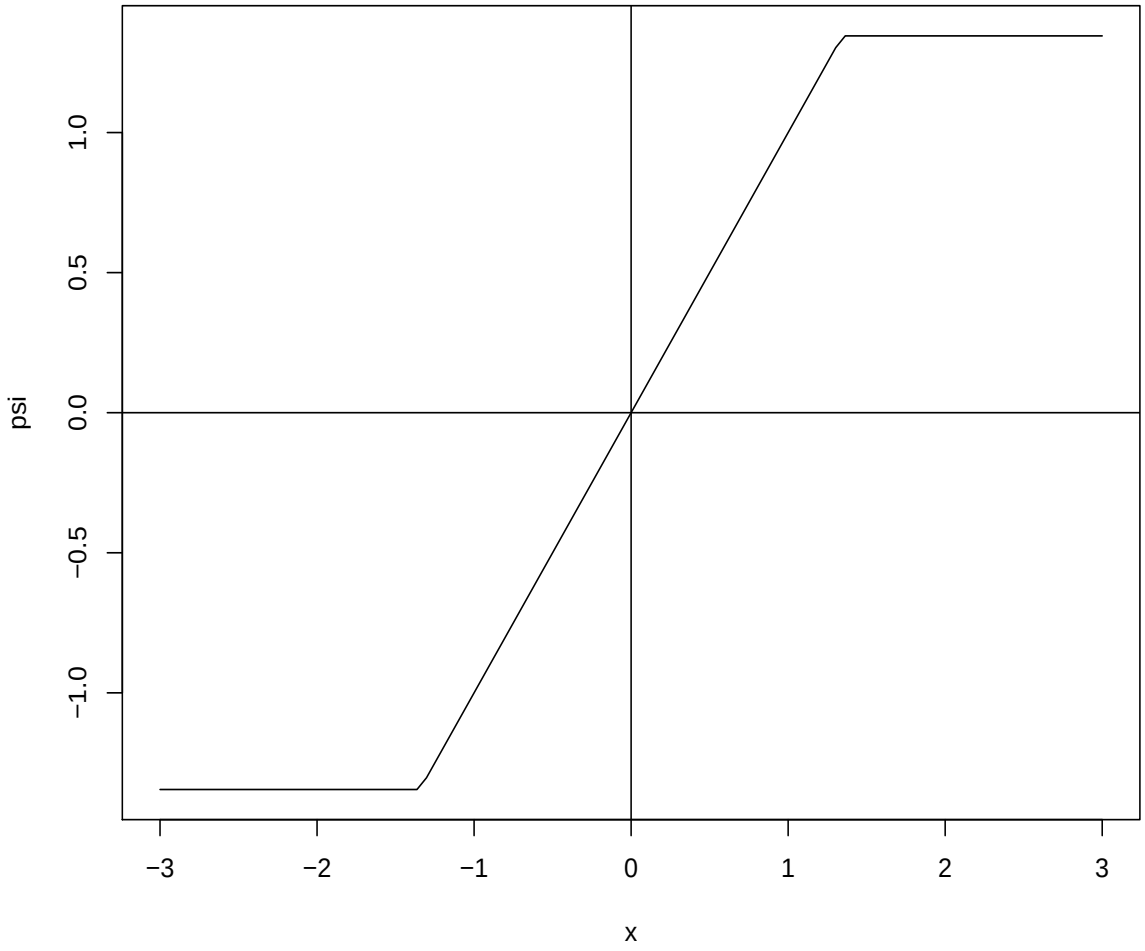


Figure 1.1: Huber's $\psi$ function

Huber's-$\psi$ function is monotonic increasing and it gives a unique solution. But the bisquare-$\psi$ function is not monotonic, so it will generally give multiple solutions. We call the estimate by Huber's-$\psi$ function "monotone M-estimates" and that by bisquare-$\psi$ function "redescending M-estimates."

Redescending estimates offer an increase in robustness in the presence of large outliers. This is caused by the different ways the two methods handle with outliers: For the fixed $k$, Huber's $\psi$ function will take the

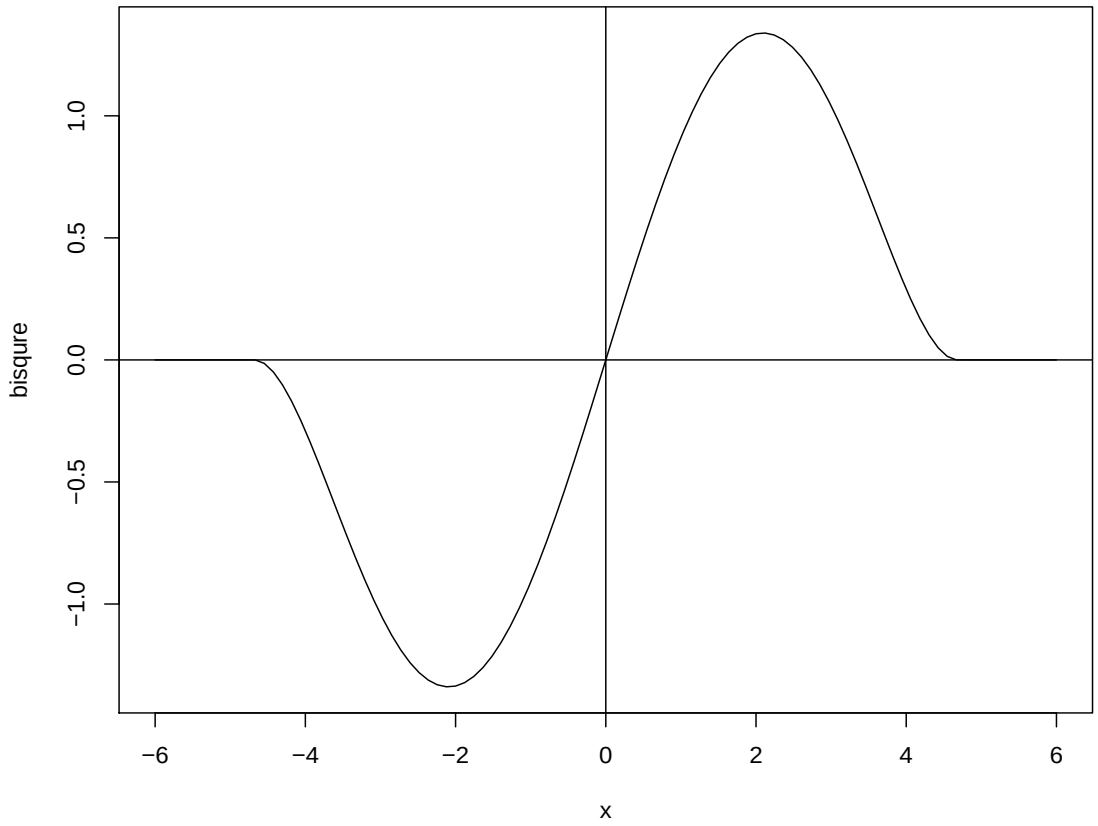


Figure 1.2: Bisqure's $\psi$ function

observations, whose absolute values bigger than absolute $k$ value, as $k$. But bisquare's $\psi$ function will delete those exceptionally large outliers. Hence bisquare's method can be better if large outliers exist. Certainly, the $k$ in the two functions are different. Bisquare's $k$ is usually bigger than $Huber$'s $k$.

Note that the $k$ value usually depends on the scale. If the scale parameter $\sigma$ is unknown, we also need to estimate it robustly. In our report, we introduce the frequently used scale estimate– *bisquare scale* weight. The function is defined as

$$w(t) = \begin{cases} \rho(t)/t^2 & \text{if } t \neq 0; \\ \rho^{''}(0) & \text{if } t = 0. \end{cases}$$

Where

$$\rho(t) = \begin{cases} 1 - [1 - (\frac{t}{k})^2]^3 & \text{if } |t| \leq k; \\ 1 & \text{if } |t| > k. \end{cases}$$

By the definition of $M - estimate\ of\ scale$, the estimate of $\sigma$ should satisfy an equation of the form

$$\frac{1}{n}\sum_{i=1}^{n}\rho(\frac{x_i}{\sigma}) = \delta, \tag{1.14}$$

where $\rho$ is a $\rho - function$ and $\delta = 0.5$. Note: $\rho - function$ is a function with the following properties:

**R1** $\rho(x)$ is a nondecreasing function of $|x|$.

**R2** $\rho(0) = 0$.

**R3** $\rho(x)$ is increasing for $x > 0$ such that $\rho(x) < \rho(\infty)$.

**R4** If $\rho$ is bounded, it is also assumed that $\rho(\infty) = 1$.

See Maronna(2006, section 2.2) for more detials.

Combining the *bisquare scale weight* function and $M - estimate\ sale$ equation, The weighted M-scale estimate of variance satisfies the following equation:

$$\hat{\sigma}^2 = \frac{1}{n\delta}\sum_{i=1}^{n} w(\frac{x_i}{\hat{\sigma}})x_i^2. \tag{1.15}$$

Where $w$ is the *bisquare scale weight* above.

This M-scale estimate of $\hat{\sigma}$ will be used to calculate the regression parameters. This will be discussed in the next chapter.

After standardizing the observations by the weighted M-scale estimate of $\hat{\sigma}$. we can get the fixed $k$ value in $Huber$'s $\psi$ function and *bisquare* $\psi$ function. Basically, we choose $k = 1.345$ in $Huber$'s $\psi$ function, and $k = 4.685$ in *bisquare* $\psi$function. The choices of $k$ in this way can give very robust results for estimating the location parameters. See Maronna(2006, section 2.2). When $k = 1.345$, the efficiency of $Huber$'s $\psi$ function is 0.95. That means if the data follows normal distribution exactly, the ratio of MSE by the classic method over the MSE by $Huber's$ $\psi$ function is 0.95. But if the distribution is not exactly normally distributed, $Huber's$ $\psi$ function is better than the classic method. Similar explanation for $k = 4.685$ in *bisquare* $\psi$ function.

# Chapter 2

# Robust Mixtures Of Linear Regression Models

## 2.1 Introduction

Mixtures of regressions provide a flexible tool to investigative the relationship between variables people are interested coming from several unknown latent components. They are widely used in may fields, including engineering, genetics, biology, econometrics, and marketing. A typical data set is tone perception data (Cohen, 1984), which is shown in Figure 2.1. In the tone perception experiment of Cohen (1984), a pure fundamental tone was played to a trained musician with electronically generated overtones added, which were determined by a stretching ratio. The tuning ratio, which is the ratio between adjusted tone and the fundamental tone, was recorded for 150 trials from the same musician. The purpose of this experiment was to see how this tuning ratio affects the perception of the tone. Furthermore, the experiment was designed to determine if either of two musical perception theories was reasonable (see Cohen (1980) for details). Based on Figure 2.1, two lines are evident which correspond to the behavior indicated by the two musical perception theories. The two regression lines corresponds to correct tuning and tuning to the first overtone, respectively.

The model setting for mixture of regression models can be stated as follows. Let $Z$ be a latent class variable with $P(Z_i = j \mid \underline{x}_i) = \pi_j$ for $j = 1, 2, \cdots, m$, where $\underline{x}_i$ is the p-dimensional vector. Given $Z_i = j$, suppose that the response $y_i$ depends on $\underline{x}_i$ in a linear way

$$y_i = \underline{x}_i^T \boldsymbol{\beta}_j + \epsilon_{ij}, \tag{2.1}$$

$\boldsymbol{\beta}_j = (\beta_{1j}, \ldots, \beta_{pj})^T$, and $\epsilon_{ij} \sim N(0, \sigma_j^2)$. Here, $\underline{x}_i$ includes both the predictors and the constant 1. Then

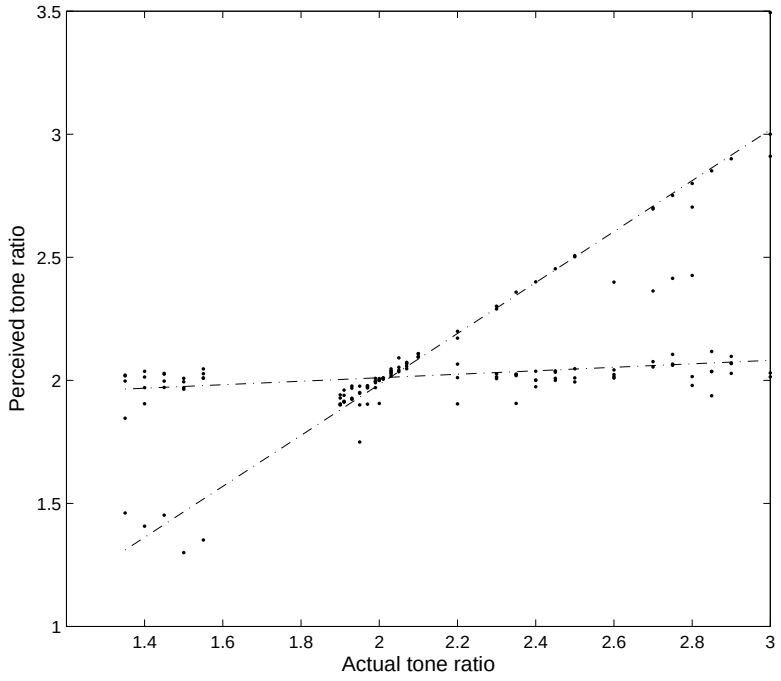


Figure 2.1: The scatter plot of the tone perception data and the fitted two lines by our proposed method. The predictor is actual tone ratio and the response is the perceived tone ratio by a trained musician.

the conditional distribution of $Y_i$ given $\underline{x}_i$ can be written as

$$Y_i|\underline{x}_i \sim \sum_{j=1}^{m} \pi_j N(\underline{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2). \tag{2.2}$$

Therefore, the log-likelihood for the observations $\{\underline{x}_i, y_i, i = 1, \ldots, n\}$ is

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log[\sum_{j=1}^{m} \pi_j \phi(y_i; \underline{x}_i^T \boldsymbol{\beta}_j, \sigma_j^2)], \tag{2.3}$$

where $\boldsymbol{\theta} = (\pi_1, \sigma_1, \boldsymbol{\beta}_1, \ldots, \pi_m, \sigma_m, \boldsymbol{\beta}_m)$. See, for example, Wedel (2000), Skrondal (2004), Jacobs, Jordan, Nowlan, and Hinton (1991), and Jiang and Tanner (1999), for some applications of model (2.2).

The unknown parameters in the model (2.2) can be estimated by the maximum likelihood estimator (MLE), which maximizes (2.3). Note that (2.3) does not have explicit solution. It is usually estimated by the EM algorithm ; see Dempster, Laird, and Rubin (1977).

Based on the EM algorithm theory, after we get the classification probabilities $p_{ij}^{(k+1)}$ in the E step, in the M step $\boldsymbol{\beta}_j, j = 1, \ldots, m$ is updated by

$$\boldsymbol{\beta}_j^{(k+1)} = \arg\min \sum_{i=1}^{n} p_{ij}^{(k+1)} (y_i - \underline{x}_i^T \boldsymbol{\beta}_j)^T (y_i - \underline{x}_i^T \boldsymbol{\beta}_j). \tag{2.4}$$

Consider the first derivative:

$$\frac{\partial}{\partial \boldsymbol{\beta}_j} \sum_{i=1}^{n} p_{ij} (y_i^T y_i - y_i^T \underline{x}_i^T \boldsymbol{\beta}_j - \boldsymbol{\beta}_j^T \underline{x}_i y_i + \boldsymbol{\beta}_j^T \underline{x}_i \underline{x}_i^T \boldsymbol{\beta}_j) = 0$$

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} (-2\underline{x}_i y_i + 2\underline{x}_i \underline{x}_i^T \boldsymbol{\beta}_j) = 0$$

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} \underline{x}_i (y_i - \underline{x}_i^T \boldsymbol{\beta}_j) = 0$$

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} \underline{x}_i \underline{x}_i^T \boldsymbol{\beta}_j = \sum_{i=1}^{n} p_{ij}^{(k+1)} \underline{x}_i y_i$$

Therefore,

$$\boldsymbol{\beta}_j^{(k+1)} = (\sum_{i=1}^{n} p_{ij}^{(k+1)} \underline{x}_i \underline{x}_i^T)^{-1} \sum_{i=1}^{n} p_{ij}^{(k+1)} \underline{x}_i y_i$$

To make the formula neater, we introduce matrix notation for the above formula. Let

$$W_j^{(k+1)} = \begin{pmatrix} p_{1j}^{(k+1)} & 0 & \cdots & 0 \\ 0 & p_{2j}^{(k+1)} & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{nj}^{(k+1)} \end{pmatrix} \tag{2.5}$$

$$X = \begin{pmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \vdots \\ \underline{x}_n^T \end{pmatrix} = \begin{pmatrix} 1 & x_1^T \\ 1 & x_2^T \\ \vdots & \vdots \\ 1 & x_n^T. \end{pmatrix} \tag{2.6}$$

Thus,

$$\boldsymbol{\beta}_j^{(k+1)} = (X^T W_j^{k+1} X)^{-1} X^T W_j^{(k+1)} Y, j = 1, \ldots, m.$$

Therefore, based on the normal assumption of the error density the traditional method uses the following EM algorithm to estimate the regression parameters.

**Algorithm 2.1.1.** *Based on the initial values of* $\{\pi_j^{(0)}, \boldsymbol{\beta}_j^{(0)}, \sigma_j^{(0)}, j = 1, \ldots, m\}$, *The EM algorithm is to iterate the following E-step and M-step.*

*E-step: Calculate the classification probabilities*

$$p_{ij}^{(k+1)} = \frac{\pi_j^{(k)} \phi(y_i; \underline{x}_i^T \boldsymbol{\beta}_j^{(k)}, \sigma_j^{2(k)})}{\sum_{l=1}^m \pi_l^{(k)} \phi(y_i; \underline{x}_i^T \boldsymbol{\beta}_l^{(k)}, \sigma_l^{2(k)})}, j = 1, \ldots, m$$

*M-step: Update* $\boldsymbol{\theta}$

$$\pi_j^{(k+1)} = \frac{\sum_{i=1}^n p_{ij}^{(k+1)}}{n}, j = 1, \ldots, m.$$

$$\boldsymbol{\beta}_j^{(k+1)} = (X^T W_j^{(k+1)} X)^{-1} X^T W_j^{(k+1)} Y, j = 1, \ldots, m,$$

$$\sigma_j^{2(k+1)} = \frac{\sum_{i=1}^n p_{ij}^{(k+1)} (y_i - \underline{x}_i^T \boldsymbol{\beta}_1^{(k+1)})^2}{\sum_{i=1}^n p_{ij}^{(k+1)}}, j = 1, \ldots, m,$$

*where* $X$ *and* $W_j^{(k+1)}$ *are defined in (2.6) and (2.5), respectively. If we assume the variances are equal, then*

$$\sigma_j^{2(k+1)} = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)} (y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k+1)})^2.$$

Based on (2.4), one can see that in the M step, $\boldsymbol{\beta}$ is updated by minimizing a weighted least square due to the normal error assumption. It is well known that the least square criterion is sensitive to outliers or heavy tail error distribution. In this report, we will provide a robust estimation method for the mixtures of regression models.

## 2.2 Robust Estimation Method

In order to robustly estimate the regression parameters, in the M step, we propose to replace the least squares criterion with a robust criterion $\rho$, i.e., $\boldsymbol{\beta}_j^{(k+1)}, j = 1, \ldots, m$ is the solution of

$$\sum_{i=1}^n p_{ij}^{(k+1)} \underline{x}_i \psi\left(\frac{y_i - \underline{x}_i^T \boldsymbol{\beta}_j}{\sigma_j^{(k)}}\right) = 0,$$

where $\psi(\cdot) = \rho'(\cdot)$. In our report, we will consider both *Huber*'s $\psi$ function defined in (1.12) (with $k = 1.345$) and the *bisquare* $\psi$ fucntion defined in (1.13) (with $k = 4.685$).

Note that

$$\sum_{i=1}^{n} p_{ij}^{(k+1)} \underline{x}_i \psi\left(\frac{y_i - \underline{x}_i^T \boldsymbol{\beta}_j}{\sigma_j^{(k)}}\right) \approx \sum_{i=1}^{n} p_{ij}^{(k+1)} \frac{\psi\{(y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k)})/\sigma_j^{(k)}\}}{(y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k)})/\sigma_j^{(k)}} \underline{x}_i \left(\frac{y_i - \underline{x}_i^T \boldsymbol{\beta}_j}{\sigma_j}\right)$$
$$= \sum_{i=1}^{n} p_{ij}^{*(k+1)} \underline{x}_i \left(\frac{y_i - \underline{x}_i^T \boldsymbol{\beta}_j}{\sigma_j}\right),$$

where

$$p_{ij}^{*(k+1)} = p_{ij}^{(k+1)} \frac{\psi((y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k)})/\sigma_j^{(k)})}{(y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k)})/\sigma_j^{(k)}}.$$

So

$$\boldsymbol{\beta}_j^{(k+1)} = \left(\sum_{i=1}^{n} p_{ij}^{*(k+1)} \underline{x}_i \underline{x}_i^T\right)^{-1} \sum_{i=1}^{n} p_{ij}^{*(k+1)} \underline{x}_i y_i$$

Let

$$W_j^{*(k+1)} = \begin{pmatrix} p_{1j}^{*(k+1)} & 0 & \cdots & 0 \\ 0 & p_{2j}^{*(k+1)} & 0 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{nj}^{*(k+1)} \end{pmatrix} \tag{2.7}$$

Thus,

$$\boldsymbol{\beta}_j^{(k+1)} = (X^T W_j^{*^{(k+1)}} X)^{-1} X^T W_j^{*^{(k+1)}} Y,$$

where $X$ is defined in (2.6).

Based on the above discussions, we propose the following new robust estimation method for the mixtures of regression model of (2.1).

**Algorithm 2.2.1.** *Given the initial values of* $\{\pi_j^{(0)}, \boldsymbol{\beta}_j^{(0)}, \sigma_j^{(0)}, j = 1, \ldots, m\}$, *our new robust EM algorithm is to iterate the following E-step and M-step.*

*E-step: Calculate the classification probabilities*

$$p_{ij}^{(k+1)} = \frac{\pi_j^{(k)} \phi(y_i; \underline{x}_i^T \boldsymbol{\beta}_j^{(k)}, \sigma_j^{2(k)})}{\sum_{l=1}^{m} \pi_l^{(k)} \phi(y_i; \underline{x}_i^T \boldsymbol{\beta}_l^{(k)}, \sigma_l^{2(k)})}$$

*M step: Update* $\boldsymbol{\theta}$

$$\pi_j^{(k+1)} = \frac{\sum_{i=1}^{n} p_{ij}^{(k+1)}}{n}, i = 1...n. j = 1, \ldots, m.$$

$$\boldsymbol{\beta}_j^{(k+1)} = \left(\sum_{i=1}^{n} p_{ij}^{*(k+1)} \underline{x}_i \underline{x}_i^T\right)^{-1} \sum_{i=1}^{n} p_{ij}^{*(k+1)} \underline{x}_i y_i$$

*Using the techniques similar to those in Section 1.2.2, we update the variance by*

$$\sigma_j^{2(k+1)} = \frac{\sum_{i=1}^n p_{ij}^{(k+1)}(y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k+1)})^2 w_{ij}^{(k+1)}}{\delta \sum_{i=1}^n p_{ij}^{(k+1)}}, j = 1, \ldots, m$$

*where*

$$w_{ij}^{(k+1)} = \min[1 - \{1 - (\frac{y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k+1)}}{c\sigma_j^{(k)}})^2\}^3, 1]/(\frac{y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k+1)}}{\sigma_j^{(k)}})^2$$

$\delta = 0.5$, *and* $c = 1.56$.

*If the two variances are equal, the* $(k+1)th$ *estimate of the variance is:*

$$\sigma^{2(k+1)} = \frac{\sum_{i=1}^n \sum_{j=1}^m p_{ij}^{(k+1)}(y_i - \underline{x}_i^T \boldsymbol{\beta}_j^{(k+1)})^2 w_{ij}^{(k+1)}}{n\delta}.$$

In (2.2), if $\underline{x}_i$ only includes the intercept term 1, then the model is the regular normal mixture model. Thus, our above proposed robust estimation method can be also applied to robustly estimate the location parameters in the normal mixture model.

# Chapter 3

# Simulation Studies and Real Data Application

We generate the independent and identically distributed (i.i.d.) data $\{(x_i, y_i), i = 1, \ldots, n\}$ from the model

$$Y = \begin{cases} 0 + 4X + \epsilon_1, & \text{if } Z = 1; \\ 0 - 4X + \epsilon_2, & \text{if } Z = 2. \end{cases},$$

where $Z$ is component indicator of $Y$ with $P(Z = 1) = 0.5$, $X \sim U(0, 1)$, and $\epsilon_1$ and $\epsilon_2$ have the same distribution. We consider the following three cases for the error density $\epsilon$.

*Case 1*: $\epsilon \sim N(0, 1)$ – Standard normal distribution.

*Case 2*: $\epsilon \sim t_4$ – T distribution with degree freedom 4.

*Case 3*: $\epsilon_1 \sim 0.95N(0, 1) + 0.05N(0, 5^2)$ – Contaminated normal mixture.

We use Case 1 to test the efficiency of our robust estimation method compared to the traditional method when the error is exactly normally distributed. Note that when the error is normal, the traditional estimation method by the Algorithm 2.1.1 is the asymptotically most efficient one. Case 2 is a heavy tailed distribution. The $t$-distributions with degrees of freedom from 3 to 5 are often used to represent heavy-tailed distributions. Case 3 is a contaminated normal mixture model, which is often used to mimic the outlier situation. The 5% data from $N(0, 5^2)$ are mostly likely to be outliers. Figure 3.1, 3.2, and 3.3 are the scatter plots of a typical sample size 200 for the three cases, separately.

We use both the traditional estimation method (Algorithm 2.1.1) and our robust estimation method (Algorithm 2.2.1) to estimate the mixture regression parameters. For our robust method, we will consider

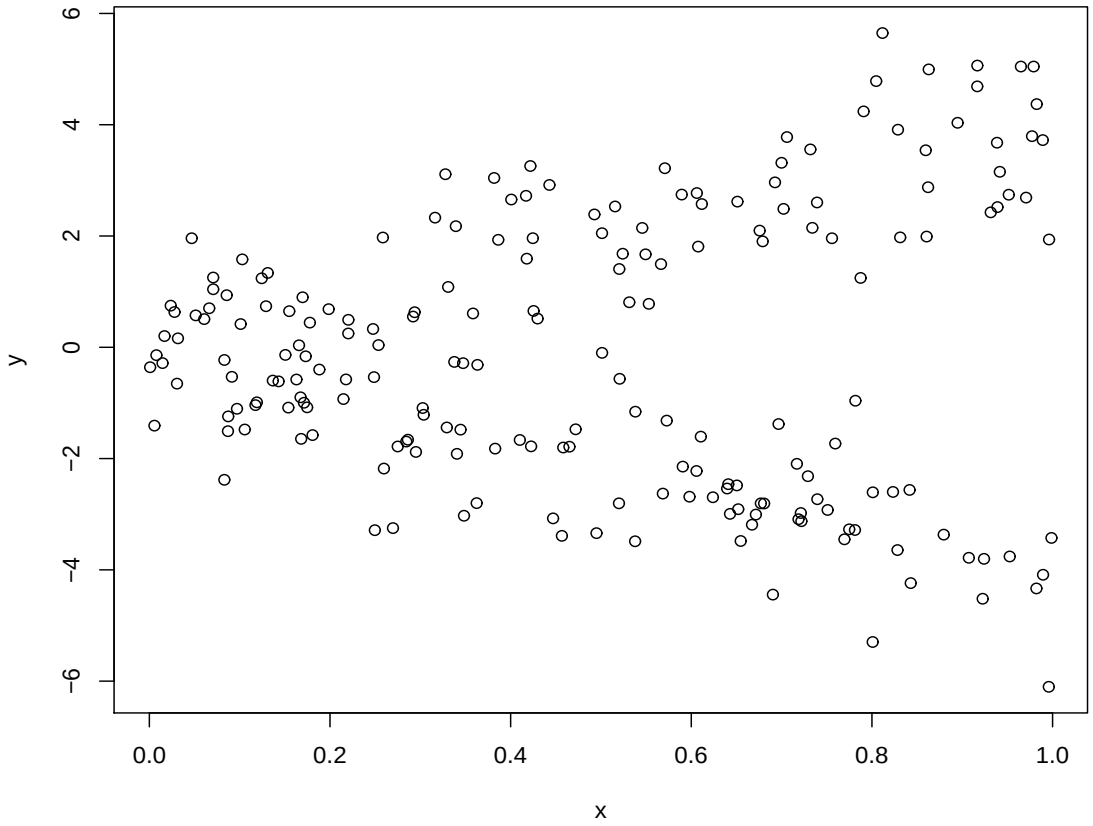


Figure 3.1: Case 1: Standard normal distribution

both Huber's $\psi$ function and Tukey's bisquare function. In addition, we will consider the algorithms of both assuming unequal variance and equal variance in each case.

We got the MSE's for each case under six conditions: no robust assuming equal and unequal variance; robust by *bisquare* method assuming equal and unequal variance; robust by *huberpsi* methods assuming equal and unequal variance, and then compare the MSE by efficiency.Based on the MSE of no robust assuming equal variance, I got the efficiency which is easier to compare the results.

The sample size in each case is $n = 200$ and 500 repetitions were generated for each parameter configuration. Similar to Bordes, Chauveau, and Vandekerkhove (2007), we use the true initial values for $\boldsymbol{\theta}$ in our EM algorithm, in order to avoid the possible bias introduced by different starting values among replications or label switching issues (Diebolt and Robert, 1994; Stephens, 2000; Yao and Lindsay, 2009). The true parameter values are $\beta_{10} = \beta_{20} = 0, \beta_{11} = 4, \beta_{21} = -4, \pi_1 = \pi_2 = 0.5$. In **case 1**, $\epsilon_j \sim N(0, 1)$, $\sigma_1 = \sigma_2 = 1$. In **case 2**, $\epsilon_j \sim t_4$, and the standard deviation is $\sqrt{2} = 1.414$. In **case 3**, $\epsilon_j \sim 0.95N(0, 1) + 0.05N(0, 25)$,

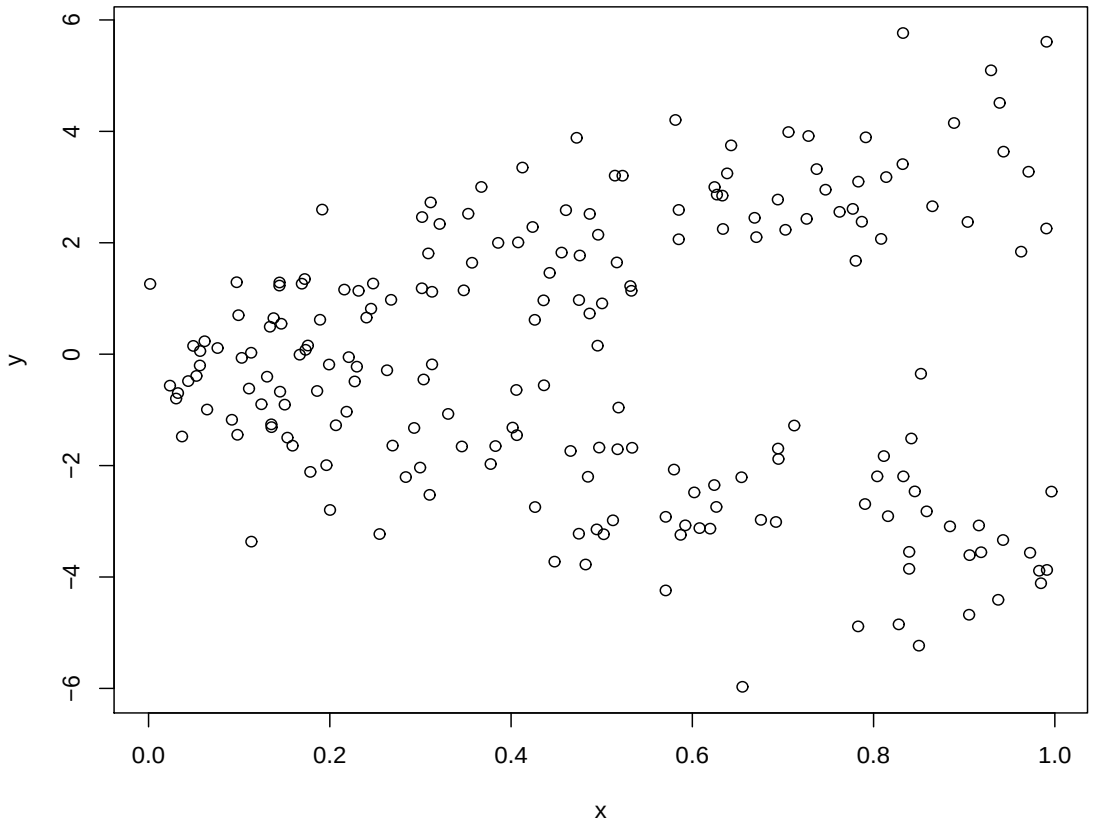


Figure 3.2: Case 2: T distribution with 4 degree freedom

and the standard deviation is $\sqrt{0.95 \times 1 + 0.05 \times 25} = 1.483$.

In Table 3.1, 3.3, and 3.5, we report the MSE for different estimation methods for regression parameters. EM-Reg means the traditional estimation method. Rob-EM-Bi refers to the robust estimation method using Tukey's bisquare function. Rob-EM-Hu refers to the robust estimation method using Huber's $\psi$ function. For better comparison, in Table 3.2, 3.4, and 3.6, we also report the relative efficiency for different estimation methods as compared to the tradition method assuming equal variance. The value greater than 1 means the corresponding method is more efficient than the tradition method assuming equal variance.

For Table 3.1 and 3.2, $\epsilon \sim N(0, 1)$, one can see that all methods estimate the parameters well. Note that in this case, the traditional estimate, which assumes a normal error is the best one. From Table 3.1 and 3.2, one can see that our proposed robust estimation methods are comparable to the traditional methods.

Based on Table 3.3 and 3.4, one can see that our robust estimation method works better than the traditional non-robust method when the error has heavy tail. Both robust methods work closely. But

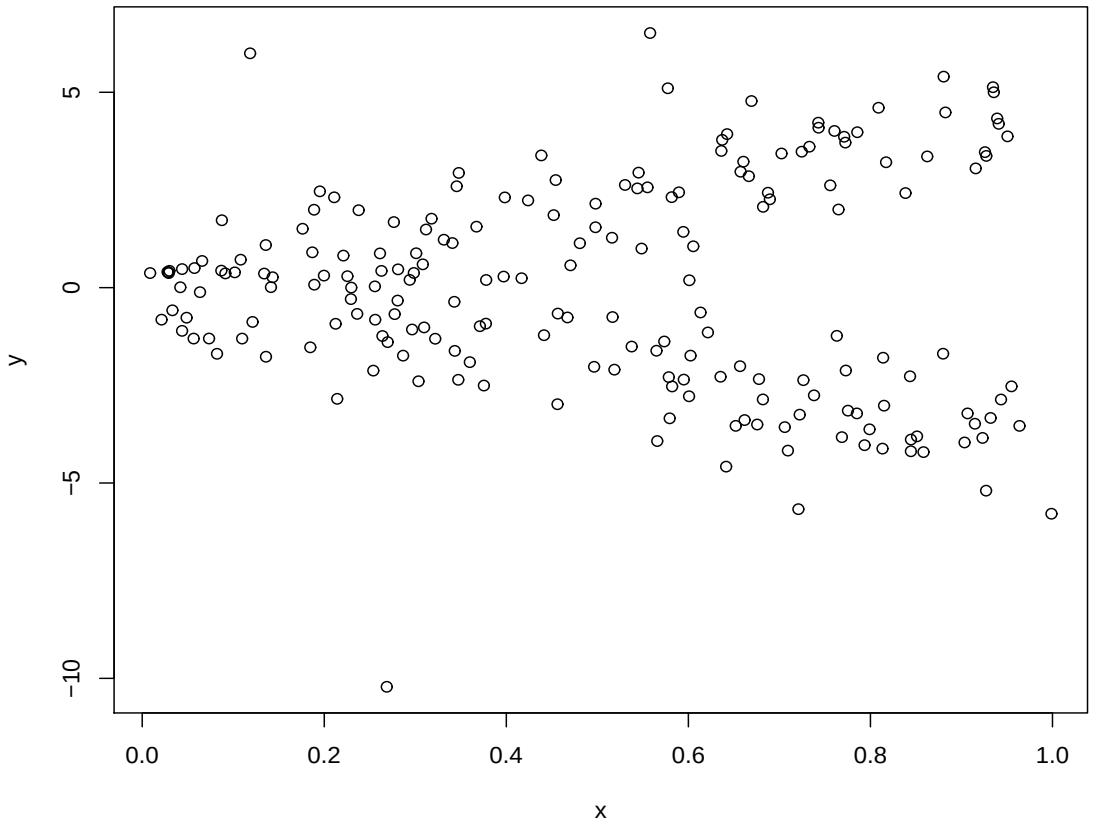


Figure 3.3: Case 3: Contaminated mixture distribution

*bisqure* way is a little bit better than *huber's* $\psi$ method.

In table 3.5 and 3.6, $\epsilon \sim 0.95N(0,1)+0.05N(0,25)$, which mimics the outlier situations. From the tables, one can see that our proposed robust methods work much better than the traditional methods except for proportions. By comparing two methods in robust regression, *bisquare* method and *huber's* $\psi$ method don't have much more difference for almost each parameter in both equal and unequal variance.

| Method | Condition | MSE of Parameters | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\hat{\beta}_{10}$ | $\hat{\beta}_{20}$ | $\hat{\beta}_{11}$ | $\hat{\beta}_{21}$ | $\hat{\pi}_1$ | $\hat{\pi}_2$ |
| EM-Reg | *equ* | 0.073 | 0.109 | 0.192 | 0.225 | 0.001 | 0.001 |
| | *unequ* | 0.076 | 0.118 | 0.197 | 0.235 | 0.001 | 0.001 |
| Rob-EM-Bi | *equ* | 0.079 | 0.115 | 0.209 | 0.240 | 0.001 | 0.001 |
| | *unequ* | 0.082 | 0.122 | 0.213 | 0.247 | 0.001 | 0.001 |
| Rob-EM-Hu | *equ* | 0.079 | 0.115 | 0.209 | 0.240 | 0.001 | 0.001 |
| | *unequ* | 0.082 | 0.123 | 0.213 | 0.247 | 0.001 | 0.001 |

Table 3.1: MSE's of Case 1: $\epsilon \sim$ Standard normal distribution

| Method | condition | efficiency of Parameters | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\hat{\beta}_{10}$ | $\hat{\beta}_{20}$ | $\hat{\beta}_{11}$ | $\hat{\beta}_{21}$ | $\hat{\pi}_1$ | $\hat{\pi}_2$ |
| EM-Reg | *equ* | 1 | 1 | 1 | 1 | 1 | 1 |
| | *unequ* | 0.962 | 0.924 | 0.975 | 0.957 | 1.149 | 1.149 |
| Rob-EM-Bi | *equ* | 0.921 | 0.945 | 0.918 | 0.937 | 0.971 | 0.971 |
| | *unequ* | 0.894 | 0.892 | 0.903 | 0.911 | 1.051 | 1.051 |
| Rob-EM-Hu | *equ* | 0.923 | 0.943 | 0.919 | 0.936 | 0.967 | 0.967 |
| | *unequ* | 0.894 | 0.887 | 0.903 | 0.912 | 1.046 | 1.046 |

Table 3.2: Efficiency of Case 1: $\epsilon \sim$ Standard normal distribution

| Method | Condition | MSE of Parameters | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\hat{\beta}_{10}$ | $\hat{\beta}_{20}$ | $\hat{\beta}_{11}$ | $\hat{\beta}_{21}$ | $\hat{\pi}_1$ | $\hat{\pi}_2$ |
| EM-Reg | *equ* | 0.165 | 0.197 | 0.366 | 0.446 | 0.003 | 0.003 |
| | *unequ* | 0.161 | 0.188 | 0.353 | 0.442 | 0.003 | 0.003 |
| Rob-EM-Bi | *equ* | 0.110 | 0.159 | 0.238 | 0.371 | 0.003 | 0.003 |
| | *unequ* | 0.119 | 0.157 | 0.252 | 0.369 | 0.003 | 0.003 |
| Rob-EM-Hu | *equ* | 0.116 | 0.171 | 0.246 | 0.379 | 0.003 | 0.003 |
| | *unequ* | 0.124 | 0.165 | 0.257 | 0.372 | 0.003 | 0.003 |

Table 3.3: MSE's of Case 2: $\epsilon \sim$ T distribution with 4 degree freedom

| Method | Condition | efficiency of Parameters | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\hat{\beta}_{10}$ | $\hat{\beta}_{20}$ | $\hat{\beta}_{11}$ | $\hat{\beta}_{21}$ | $\hat{\pi}_1$ | $\hat{\pi}_2$ |
| EM-Reg | *equ* | 1 | 1 | 1 | 1 | 1 | 1 |
| | *unequ* | 1.021 | 1.050 | 1.037 | 1.007 | 0.956 | 0.956 |
| Rob-EM-Bi | *equ* | 1.495 | 1.241 | 1.541 | 1.200 | 1.135 | 1.135 |
| | *unequ* | 1.381 | 1.258 | 1.453 | 1.208 | 1.031 | 1.031 |
| Rob-EM-Hu | *equ* | 1.420 | 1.156 | 1.490 | 1.175 | 1.131 | 1.131 |
| | *unequ* | 1.330 | 1.198 | 1.425 | 1.198 | 1.021 | 1.021 |

Table 3.4: Efficiency of Case 2: $\epsilon \sim$ T distribution with 4 degree freedom

| Method | Condition | MSE of Parameters | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\hat{\beta}_{10}$ | $\hat{\beta}_{20}$ | $\hat{\beta}_{11}$ | $\hat{\beta}_{21}$ | $\hat{\pi}_1$ | $\hat{\pi}_2$ |
| EM-Reg | *equ* | 0.310 | 1.014 | 0.596 | 2.251 | 0.005 | 0.005 |
| | *unequ* | 0.240 | 0.959 | 0.476 | 2.18 | 0.004 | 0.004 |
| Rob-EM-Bi | *equ* | 0.199 | 0.260 | 0.390 | 0.617 | 0.004 | 0.004 |
| | *unequ* | 0.206 | 0.304 | 0.378 | 0.702 | 0.004 | 0.004 |
| Rob-EM-Hu | *equ* | 0.202 | 0.365 | 0.365 | 0.832 | 0.004 | 0.004 |
| | *unequ* | 0.213 | 0.425 | 0.375 | 0.934 | 0.004 | 0.004 |

Table 3.5: MSE's of Case 3: $\epsilon \sim 0.95N(0,1) + 0.05N(0,5^2)$

| Method | Condition | efficiency of Parameters | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\hat{\beta}_{10}$ | $\hat{\beta}_{20}$ | $\hat{\beta}_{11}$ | $\hat{\beta}_{21}$ | $\hat{\pi}_1$ | $\hat{\pi}_2$ |
| EM-Reg | *equ* | 1 | 1 | 1 | 1 | 1 | 1 |
| | *unequ* | 0.777 | 0.946 | 0.798 | 0.969 | 0.733 | 0.733 |
| Rob-EM-Bi | *equ* | 0.646 | 0.256 | 0.653 | 0.274 | 0.649 | 0.649 |
| | *unequ* | 0.665 | 0.300 | 0.634 | 0.312 | 0.794 | 0.794 |
| Rob-EM-Hu | *equ* | 0.654 | 0.360 | 0.612 | 0.370 | 0.683 | 0.683 |
| | *unequ* | 0.687 | 0.419 | 0.629 | 0.415 | 0.771 | 0.771 |

Table 3.6: Efficiency of Case 3: $\epsilon \sim 0.95N(0,1) + 0.05N(0,5^2)$

Mixtures of regressions are widely used in many fields. The tone data in section 2.1 is a good example. We applied our new method to the real data, and compared it with the traditional method. With some outliers added to the data set, the scatter plot and the fitted lines by robust EM algorithm and regular EM algorithm look like this way:

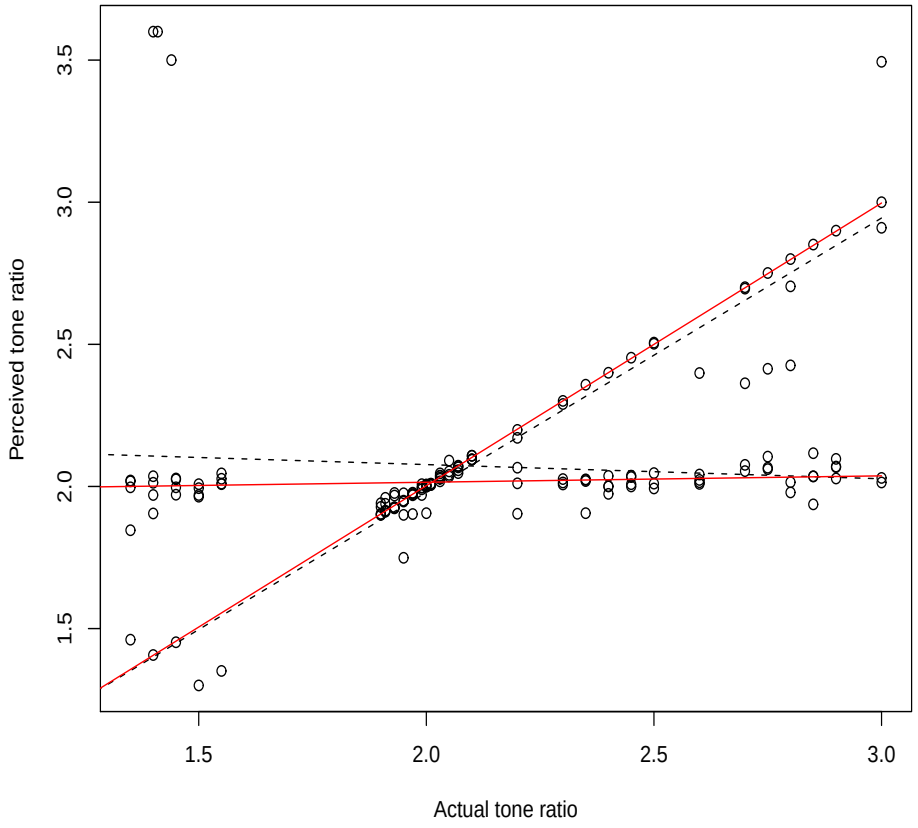


Figure 3.4: Robust regression and non robust regression for tone data with outliers

In figure3.4, we added three outliers which are x=(1.4,1.41,1.44) y=(3.6,3.6,3.5) The red (or solid )lines by robust EM method ignored the outliers. So they are much better than that by regular EM algorithm. Actually comparing with figure 2.1, robust lines in figure 3.4 are very close to the fitted line in figure 2.1. If we increased the values of outliers, which are x=(1.4,1.4,1.4) y=(3.9,3.9,3.9), some dramatic changes happened in figure3.5. Our new method still kept the same as that in figure3.4, while the regular EM algorithm fitted the whole body of data as one group and the outliers as the other group. That means our new method can still work well to the real data.

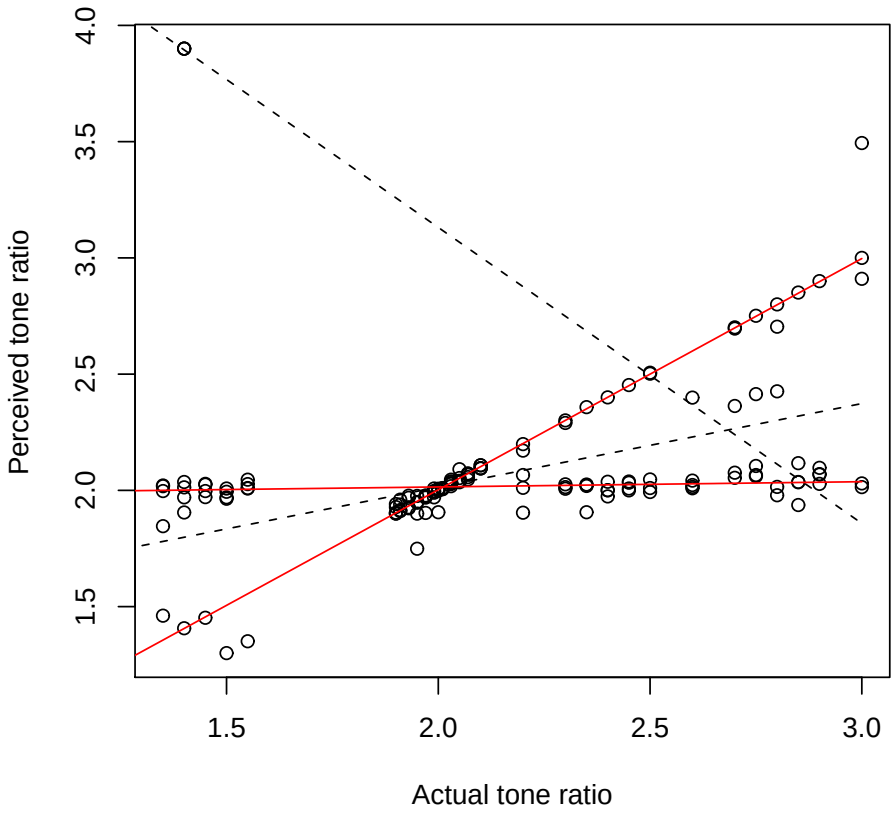


Figure 3.5: Robust regression and non robust regression for tone data with increased outliers

# Chapter 4

# Conclusion and Future Work

The traditional estimation method for mixture of linear regression models is to assume a normal model for the error and then the parameters are estimated by the MLE using the EM algorithm. Under the normal assumption, the EM algorithm uses a LSE to update the regression parameters in the M step. Due to the sensitivity of LSE, in this report, we proposed the robust estimation methods for mixtures of linear regression models which may replace the least square criterion used in the M step of the EM algorithm with some robust criteria. Based on our simulation study, we can see that our robust estimation method does not lose too much efficiency compared to the traditional estimation method when there are no outliers and the error term follows a normal. But when the distribution of error term has heavier tail or there are outliers, our robust estimation method can be much more efficient than the traditional method.

It is well known that the traditional robust regression method is consistent when the error is symmetric about 0. It will be our future task to discuss consistent conditions for our robust mixtures of linear regression models. In addition, we will also consider extending our method to some other mixture models such as mixtures of generalized linear models and mixtures of nonparametric regressions.

# Bibliography


[1] Andrews, D. F. (1974). A Robust Method for Multiple Linear Regression. *Technometrics*, 16, 523-531.

[2] Beaton, A. E. and Tukey, J. W. (1974). The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data. *Technometrics*, 16, 147-185.

[3] Charlier, C. V. L., and Wicksell, S. D. (1924), On the Dissection of Frequency Functions. *Arkiv fiir Mathematik, Astronomi och Fysik*, BD 18, 6.

[4] Bordes, L., Chauveau, D., and Vandekerkhove, P. (2007). An EM algorithm for a semiparametric mixture model. *Computational Statistics and Data Analysis*, 51, 5429-5443.

[5] Cohen, E. (1984). Some effects of inharmonic partials on interval perception. *Music Perception*, 1, 323-349.

[6] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977), Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society*, B39, 1-38.

[7] Diebolt, J. and Robert, C. P. (1994), "Estimation of finite mixture distributions through Bayesian sampling," *Journal of the Royal Statistical Society*, B56, 363-375.

[8] Doetsch, G.(1928). Die elimination des dopplereffekts auf spektroskopische feinstrukturen und exakte bestimmung der komponenten. *Zeitschrift für Physik* 49, 705-730.

[9] Hample, F. R. (1971). A general definition of qualitative robustness. *The Annals of Mathematical Statistics.* 42 1887-1896.

[10] Hample, F. R. (1974). The influence curve and its role in robust estimation. *The Annals of Statistics.* 69, 383-393.

[11] Hample, F. R. (1986). *Robust Statistics.* Wiley, New York.

[12] Holmes, G.K.(1892). Measures of Distribution. *Journal of the American Statistical Association* 3, 141-157.

[13] Holland, P. W. and Welsch, R. E. (1977). Robust Regression Using Iteratively Reweighted Least Squares. *Computations in Statistics*, A6, 813-827.

[14] Huber, P. J. (1964). Robust estimation of a local parameter. *Annals of Mathematical Statistics*, 35. 73-101.

[15] Huber, P. J. (1967). Robust behavior of maximum likelihood estimates under nonstandard conditions. *Processing of the Fifth Brkeley symposium on Mathematics and Statistics Probability*, 35. 73-101.

[16] Huber, P. J. (1973). Robust Regression: Asymptotics, Conjectures, and Monte Carlo. *Annals of Statistics*, 1, 799-821.

[17] Huber, P. J. (1981). *Robust Statistics*. Wiley, New York.

[18] Pearson, K. (1894). Contributions to the mathematical theory of evolution. Philosophical. *Transactions of the Royal Society of London* A 185 , 71-110.

[19] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. "Adaptive mixtures of local experts." *Neural Computation* 3(1):79-87, 1991.

[20] Jiang, Wenxin; Tanner, Martin A.(1999) Hierarchical mixtures-of-experts for exponential family regression models: Approximation and maximum likelihood estimation. *The Annals of Statistics.* 27, No.3, 987-1011.

[21] Ricardo A. Maronna, R. douglas Martin and Victor J. Yohai. *Robust Statistics: Theory and Methods*. Wiley, New York.

[22] Rousseeuw, P.J. and Leroy, a.M.(1987),*Robust Regression and Outier Detection*. Wiley, New York.

[23] Skrondal, A. and Rabe-Hesketh, S. (2004). *Generalized Latent Variable Modeling: Multilevel, Longitudinal and Structural Equation Models*. Boca Raton. Chapman and Hall/CRC.

[24] Staudte, R.G.and sheather,S.J.(1990),*Robust Estimation and Testing*. Wiley, New York.

[25] Stephens, M. (2000), Dealing with label switching in mixture models. *Journal of the Royal Statistical Society*, B62, 795-809.

[26] Stigler, S.M.(1986). *The History of Statistics*. Cambridge, Massachusetts: Belknap Press of Harvard Univesity Press.

[27] Stigler, S.(1973). simon Newcomb, Percy Daniell, and the history of robust estimation 1885-1920. *Journal of American Statistical Association* 68, 872-879.

[28] Stromgren,B.(1934). Tables and diagrams for dissecting a frequency cure into components by the half-invariant method . *Skandinavian Aktuarietidskr*, B62, 795-809.

[29] Tan,W.Y. and Chang,W.C.(1972). Some comparisons of the mehod of moments and the method of maximum likelihood in estimating parameters of a mixture of two normal densities. *Journal of American Statistical Association* 67, 702-708.

[30] Tukey, J.W.(1960). A survey of sampling from contaminated distributions. *Contributions to Probability and statistics.*I. Olkin (ed.), Stanford, CA: Stanford Unversity Press.

[31] Tukey, J.W.(1962). The future of data analysis *The Analysis of Mathematical Statistics.*33, 1-67

[32] Wedel.M and Kamakura.W.A. (2000). *Market Segmentation: Conceptual and Methodological Foundations.* 2nd edition, Norwell, MA: Kluwer Academic Publishers. Journal of Classification. Springer, New York.

[33] Yao and Lindsay (2009). Bayesian mixture labelling by Posterior Density. *Journal of American Statistical Association*, 104, 758-767.

# Appendix A

# R Code

```
################
huberpsi<-function(t,k=1.345){ out=pmax(-k,pmin(k,t));out}
bisquare<-function(t,k=4.685){out=t*pmax(0,(1-(t/k)^2))^2;out}
biscalew<-function(t){out=pmin(1-(1-t^2/1.56^2)^3,1)/t^2;out}
################
#the EM algorithm to fit the mixture of linear regression
#########################
#lin_equa
 mixlin1<-function(x,y,beta,sig1,pr,m=2){
run=0;acc=10^(-4)*max(abs(c(beta,sig1,pr)));n=length(x);
if(length(sig1)>1){
r=matrix(rep(0,2*n),nrow=n);pk=r;
  for(j in seq(m))
    r[,j]=y-beta[j,1]-x*beta[j,2];
#E-steps
 repeat
   {  prest=c(beta,sig1,pr);run=run+1
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,sig1[j])
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
```

```
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   temp=pk[,j]%*%x;
   beta[j,]=solve(matrix(c(np[j],temp,temp,pk[,j]%*%x^2),nrow=2))%*%c(pk[,j]%*%y,pk[,j]%*%(x*y))
   r[,j]=y-beta[j,1]-beta[j,2]*x;
   sig1[j]=sqrt(t(pk[,j])%*%(r[,j]^2)/np[j]);
     }
    dif=max(abs(c(beta,sig1,pr)-prest))
    if(dif<acc|run>500){break}
 }
}
 else{
r=matrix(rep(0,2*n),nrow=n);pk=r; sig1=sig1*c(1,1)
  for(j in seq(m))
    r[,j]=y-beta[j,1]-x*beta[j,2];
#E-steps
 repeat
   {  prest=c(beta,sig1,pr);run=run+1
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,sig1[j])
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   temp=pk[,j]%*%x;
beta[j,]=solve(matrix(c(np[j],temp,temp,pk[,j]%*%x^2),nrow=2))%*%c(pk[,j]%*%y,pk[,j]%*%(x*y));
   r[,j]=y-beta[j,1]-beta[j,2]*x;
     }
    sig1=sqrt(sum(pk*(r^2))/n)*c(1,1)
```

```
    dif=max(abs(c(beta,sig1,pr)-prest))
    if(dif<acc|run>500){break}
} }
  theta=c(beta,sig1,pr);theta
}

################
#the robust EM algorithm to fit the mixture of linear regression
########################
#bi_eq
mixlinrb_bi1<-function(x,y,beta,sig1,pr,m=2){
run=0;acc=10^(-4)*max(abs(c(beta,sig1,pr)));n=length(x);
if(length(sig1)>1)
{
r=matrix(rep(0,2*n),nrow=n);pk=r;
  for(j in seq(m))
    r[,j]=(y-beta[j,1]-x*beta[j,2])/sig1[j];
#E-steps
 repeat
   {  prest=c(sig1,beta,pr);run=run+1;
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,1)/sig1[j]
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   w=pk[,j]*bisquare(r[,j])/r[,j];
   temp=w%*%x;
   beta[j,]=solve(matrix(c(sum(w),temp,temp,w%*%x^2),nrow=2))%*%c(w%*%y,w%*%(x*y))
   r[,j]=(y-beta[j,1]-beta[j,2]*x)/sig1[j];
   sig1[j]=sqrt(sum(r[,j]^2*sig1[j]^2*pk[,j]*biscalew(r[,j]))/np[j]/0.5);
```

```
     }
    dif=max(abs(c(sig1,beta,pr)-prest))
    if(dif<acc|run>500){break}
 }  }
 else{ r=matrix(rep(0,2*n),nrow=n);pk=r;sig1=sig1*c(1,1);
  for(j in seq(m))
    r[,j]=(y-beta[j,1]-x*beta[j,2])/sig1[j];
#E-steps
 repeat
   {  prest=c(sig1,beta,pr);run=run+1;
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,1)/sig1[j]
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   w=pk[,j]*bisquare(r[,j])/r[,j];
   #w=pk[,j]*huberpsi(r[,j])/r[,j];
   temp=w%*%x;
   beta[j,]=solve(matrix(c(sum(w),temp,temp,w%*%x^2),nrow=2))%*%c(w%*%y,w%*%(x*y))
   r[,j]=(y-beta[j,1]-beta[j,2]*x)/sig1[j];
     }
   sig1=sqrt(sum(pk*(r^2*sig1[1]^2)*biscalew(r))/n/0.5)*c(1,1)
    dif=max(abs(c(sig1,beta,pr)-prest))
    if(dif<acc|run>500){break}
}}
theta=c(beta,sig1,pr);theta
}


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

```
#the robust EM algorithm to fit the mixture of linear regression
#########################

#hu_eq:

mixlinrb_hu1<-function(x,y,beta,sig1,pr,m=2){
run=0;acc=10^(-4)*max(abs(c(beta,sig1,pr)));n=length(x);
if(length(sig1)>1)
{
r=matrix(rep(0,2*n),nrow=n);pk=r;
  for(j in seq(m))
    r[,j]=(y-beta[j,1]-x*beta[j,2])/sig1[j];
#E-steps
 repeat
   {  prest=c(sig1,beta,pr);run=run+1;
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,1)/sig1[j]
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   #w=pk[,j]*bisquare(r[,j])/r[,j];
   w=pk[,j]*huberpsi(r[,j])/r[,j];

   temp=w%*%x;
   beta[j,]=solve(matrix(c(sum(w),temp,temp,w%*%x^2),nrow=2))%*%c(w%*%y,w%*%(x*y))
   r[,j]=(y-beta[j,1]-beta[j,2]*x)/sig1[j];
   sig1[j]=sqrt(sum(r[,j]^2*sig1[j]^2*pk[,j]*biscalew(r[,j]))/np[j]/0.5);
     }
    dif=max(abs(c(sig1,beta,pr)-prest))
    if(dif<acc|run>500){break}
```

```
 }  }
 else{ r=matrix(rep(0,2*n),nrow=n);pk=r;sig1=sig1*c(1,1);
  for(j in seq(m))
    r[,j]=(y-beta[j,1]-x*beta[j,2])/sig1[j];
#E-steps
 repeat
   {  prest=c(sig1,beta,pr);run=run+1;
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,1)/sig1[j]
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   #w=pk[,j]*bisquare(r[,j])/r[,j];
   w=pk[,j]*huberpsi(r[,j])/r[,j];
   temp=w%*%x;
   beta[j,]=solve(matrix(c(sum(w),temp,temp,w%*%x^2),nrow=2))%*%c(w%*%y,w%*%(x*y))
   r[,j]=(y-beta[j,1]-beta[j,2]*x)/sig1[j];
     }
   sig1=sqrt(sum(pk*(r^2*sig1[1]^2)*biscalew(r))/n/0.5)*c(1,1)
    dif=max(abs(c(sig1,beta,pr)-prest))
    if(dif<acc|run>500){break}
}}
theta=c(beta,sig1,pr);theta}


bt=Sys.time()

rep=500
mixlin1_eN_no=matrix(rep(0,8*rep),nrow=rep)
mixlin2_eN_no=matrix(rep(0,8*rep),nrow=rep)
```

```
mixlinrb_bi1_eN_no=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_hu1_eN_no=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_bi2_eN_no=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_hu2_eN_no=matrix(rep(0,8*rep),nrow=rep)

#mixlin1_eN_yes=matrix(rep(0,8*rep),nrow=rep)
#mixlin2_eN_yes=matrix(rep(0,8*rep),nrow=rep)
#mixlinrb_bi1_eN_yes=matrix(rep(0,8*rep),nrow=rep)
#mixlinrb_hu1_eN_yes=matrix(rep(0,8*rep),nrow=rep)
#mixlinrb_bi2_eN_yes=matrix(rep(0,8*rep),nrow=rep)
#mixlinrb_hu2_eN_yes=matrix(rep(0,8*rep),nrow=rep)

mixlin1_emix=matrix(rep(0,8*rep),nrow=rep)
mixlin2_emix=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_bi1_emix=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_hu1_emix=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_bi2_emix=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_hu2_emix=matrix(rep(0,8*rep),nrow=rep)


mixlin1_et=matrix(rep(0,8*rep),nrow=rep)
mixlin2_et=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_bi1_et=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_hu1_et=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_bi2_et=matrix(rep(0,8*rep),nrow=rep)
mixlinrb_hu2_et=matrix(rep(0,8*rep),nrow=rep)


n=200
m=2
beta=matrix(c(0,0,4,-4),nrow=2)
pr=c(0.5,0.5)
x=runif(n,0,1)
u=runif(n,0,1)
```

```
 for(ii in seq(rep))
{
#e=normal
#sig2=c(1,1);
#sig1=c(1)
#e=rnorm(n,0,sig2[1]);
#y=(u<=pr[1])*(beta[1,1]+beta[1,2]*x+e)+(u>pr[1])*(beta[2,1]+beta[2,2]*x+e)

#add 8 outliers.
#x=c(x,rep(0,8));
#y=c(y,rep(-6,4));
#y=c(y,rep(6,4))

#e=mix
#sig1=c(1.8439)
#sig2=c(1.8439,1.8439);
#e=(u<=0.90)*rnorm(n,0,1)+(u>0.90)*rnorm(n,0,5)
#y=(u<=pr[1])*(beta[1,1]+beta[1,2]*x+e)+(u>pr[1])*(beta[2,1]+beta[2,2]*x+e)

#sig1=c(1.4832)
#sig2=c(1.4832,1.4832);
#e=(u<=0.95)*rnorm(n,0,1)+(u>0.95)*rnorm(n,0,5)
#y=(u<=pr[1])*(beta[1,1]+beta[1,2]*x+e)+(u>pr[1])*(beta[2,1]+beta[2,2]*x+e)



#e=t(4)
sig1=c(1.414)
sig2=c(1.414,1.414);
e=rt(n,4)
y=(u<=pr[1])*(beta[1,1]+beta[1,2]*x+e)+(u>pr[1])*(beta[2,1]+beta[2,2]*x+e)

#mixlin1_eN_no[ii,]=t(mixlin1(x,y,beta,sig1,pr))
#mixlin2_eN_no[ii,]=t(mixlin2(x,y,beta,sig2,pr))
```

```
#mixlinrb_bi1_eN_no[ii,]=t(mixlinrb_bi1(x,y,beta,sig1,pr))
#mixlinrb_bi2_eN_no[ii,]=t(mixlinrb_bi2(x,y,beta,sig2,pr))
#mixlinrb_hu1_eN_no[ii,]=t(mixlinrb_hu1(x,y,beta,sig1,pr))
#mixlinrb_hu2_eN_no[ii,]=t(mixlinrb_hu2(x,y,beta,sig2,pr))


#mixlin1_eN_yes[ii,]=t(mixlin1(x,y,beta,sig1,pr))
#mixlin2_eN_yes[ii,]=t(mixlin2(x,y,beta,sig2,pr))
#mixlinrb_bi1_eN_yes[ii,]=t(mixlinrb_bi1(x,y,beta,sig1,pr))
#mixlinrb_bi2_eN_yes[ii,]=t(mixlinrb_bi2(x,y,beta,sig2,pr))
#mixlinrb_hu1_eN_yes[ii,]=t(mixlinrb_hu1(x,y,beta,sig1,pr))
#mixlinrb_hu2_eN_yes[ii,]=t(mixlinrb_hu2(x,y,beta,sig2,pr))



#mixlin1_emix[ii,]=t(mixlin1(x,y,beta,sig1,pr))
#mixlin2_emix[ii,]=t(mixlin2(x,y,beta,sig2,pr))
#mixlinrb_bi1_emix[ii,]=t(mixlinrb_bi1(x,y,beta,sig1,pr))
#mixlinrb_bi2_emix[ii,]=t(mixlinrb_bi2(x,y,beta,sig2,pr))
#mixlinrb_hu1_emix[ii,]=t(mixlinrb_hu1(x,y,beta,sig1,pr))
#mixlinrb_hu2_emix[ii,]=t(mixlinrb_hu2(x,y,beta,sig2,pr))



mixlin1_et[ii,]=t(mixlin1(x,y,beta,sig1,pr))
mixlin2_et[ii,]=t(mixlin2(x,y,beta,sig2,pr))
mixlinrb_bi1_et[ii,]=t(mixlinrb_bi1(x,y,beta,sig1,pr))
mixlinrb_bi2_et[ii,]=t(mixlinrb_bi2(x,y,beta,sig2,pr))
mixlinrb_hu1_et[ii,]=t(mixlinrb_hu1(x,y,beta,sig1,pr))
mixlinrb_hu2_et[ii,]=t(mixlinrb_hu2(x,y,beta,sig2,pr))

}
Sys.time()-bt
```

```
mixlin1_eN_no
mixlin2_eN_no
mixlinrb_bi1_eN_no
mixlinrb_bi2_eN_no
mixlinrb_hu1_eN_no
mixlinrb_hu2_eN_no


mixlin1_emix
mixlin2_emix
mixlinrb_bi1_emix
mixlinrb_bi2_emix
mixlinrb_hu1_emix
mixlinrb_hu2_emix

mixlin1_et
mixlin2_et
mixlinrb_bi1_et
mixlinrb_bi2_et
mixlinrb_hu1_et
mixlinrb_hu2_et

Sys.time()-bt
#for e=rt(n,4)
#a=b=1.414
# for e=mix
a=b=1.4832
#for e=rnorm(n,0,1)
a=b=1
orig=c(0,0,4,-4,a,b,0.5,0.5)
d=matrix(rep(1,500),nrow=500)
#d=matrix(rep(1,100),nrow=100)
orig_matrix=kronecker(t(orig),d)
```

```
mixlin_eN_no=Xno[,1:8]
mixlinrb_eN_no=Xno[,9:16]
mixlin_eN_yes=Xyes_lin_rb[,1:8]
mixlinrb_eN_yes=Xyes_lin_rb[,9:16]
mixlin_emix=Xemix_lin_rb[,1:8]
mixlinrb_emix=Xemix_lin_rb[,9:16]
mixlin_et=Xet_lin_rb[,1:8]
mixlinrb_et=Xet_lin_rb[,9:16]


diffeN_no_lin1=mixlin1_eN_no-orig_matrix
diffeN_no_lin2=mixlin2_eN_no-orig_matrix
diffeN_no_linrb_bi1=mixlinrb_bi1_eN_no-orig_matrix
diffeN_no_linrb_bi2=mixlinrb_bi2_eN_no-orig_matrix
diffeN_no_linrb_hu1=mixlinrb_hu1_eN_no-orig_matrix
diffeN_no_linrb_hu2=mixlinrb_hu2_eN_no-orig_matrix

diffemix_lin1=mixlin1_emix-orig_matrix
diffemix_lin2=mixlin2_emix-orig_matrix
diffemix_linrb_bi1=mixlinrb_bi1_emix-orig_matrix
diffemix_linrb_bi2=mixlinrb_bi2_emix-orig_matrix
diffemix_linrb_hu1=mixlinrb_hu1_emix-orig_matrix
diffemix_linrb_hu2=mixlinrb_hu2_emix-orig_matrix


diffet_lin1=mixlin1_et-orig_matrix
diffet_lin2=mixlin2_et-orig_matrix
diffet_linrb_bi1=mixlinrb_bi1_et-orig_matrix
diffet_linrb_bi2=mixlinrb_bi2_et-orig_matrix
diffet_linrb_hu1=mixlinrb_hu1_et-orig_matrix
diffet_linrb_hu2=mixlinrb_hu2_et-orig_matrix

mseeN_no_lin1=apply(diffeN_no_lin1^2,2,mean)
mseeN_no_lin2=apply(diffeN_no_lin2^2,2,mean)
```

```
mseeN_no_linrb_bi1=apply(diffeN_no_linrb_bi1^2,2,mean)
mseeN_no_linrb_bi2=apply(diffeN_no_linrb_bi2^2,2,mean)
mseeN_no_linrb_hu1=apply(diffeN_no_linrb_hu1^2,2,mean)
mseeN_no_linrb_hu2=apply(diffeN_no_linrb_hu2^2,2,mean)

mseemix_lin1=apply(diffemix_lin1^2,2,mean)
mseemix_lin2=apply(diffemix_lin2^2,2,mean)
mseemix_linrb_bi1=apply(diffemix_linrb_bi1^2,2,mean)
mseemix_linrb_bi2=apply(diffemix_linrb_bi2^2,2,mean)
mseemix_linrb_hu1=apply(diffemix_linrb_hu1^2,2,mean)
mseemix_linrb_hu2=apply(diffemix_linrb_hu2^2,2,mean)

mseet_lin1=apply(diffet_lin1^2,2,mean)
mseet_lin2=apply(diffet_lin2^2,2,mean)
mseet_linrb_bi1=apply(diffet_linrb_bi1^2,2,mean)
mseet_linrb_bi2=apply(diffet_linrb_bi2^2,2,mean)
mseet_linrb_hu1=apply(diffet_linrb_hu1^2,2,mean)
mseet_linrb_hu2=apply(diffet_linrb_hu2^2,2,mean)

#ratio
rationo_lin11=mseeN_no_lin1/mseeN_no_lin1
rationo_lin21=mseeN_no_lin2/mseeN_no_lin1
rationo_linrb_bi1=mseeN_no_linrb_bi1/mseeN_no_lin1
rationo_linrb_bi2=mseeN_no_linrb_bi2/mseeN_no_lin1
rationo_linrb_hu1=mseeN_no_linrb_hu1/mseeN_no_lin1
rationo_linrb_hu2=mseeN_no_linrb_hu2/mseeN_no_lin1

ratioemix_lin11=mseemix_lin1/mseemix_lin1
ratioemix_lin21=mseemix_lin2/mseemix_lin1
ratioemix_linrb_bi1=mseemix_linrb_bi1/mseemix_lin1
ratioemix_linrb_bi2=mseemix_linrb_bi2/mseemix_lin1
ratioemix_linrb_hu1=mseemix_linrb_hu1/mseemix_lin1
ratioemix_linrb_hu2=mseemix_linrb_hu2/mseemix_lin1
```

```
ratioet_lin11=mseet_lin1/mseet_lin1
ratioet_lin21=mseet_lin2/mseet_lin1
ratioet_linrb_bi1=mseet_linrb_bi1/mseet_lin1
ratioet_linrb_bi2=mseet_linrb_bi2/mseet_lin1
ratioet_linrb_hu1=mseet_linrb_hu1/mseet_lin1
ratioet_linrb_hu2=mseet_linrb_hu2/mseet_lin1

############################
Regression for tone data
############################

###
huberpsi<-function(t,k=1.345){ out=pmax(-k,pmin(k,t));out}
bisquare<-function(t,k=4.685){out=t*pmax(0,(1-(t/k)^2))^2;out}
biscalew<-function(t){out=pmin(1-(1-t^2/1.56^2)^3,1)/t^2;out}

################
#the EM algorithm to fit the mixture of linear regression
########################

 mixlin<-function(x,y,beta,sig,pr,m=2){
run=0;acc=10^(-4)*max(abs(c(beta,sig,pr)));n=length(x);
if(length(sig)>1){
r=matrix(rep(0,2*n),nrow=n);pk=r;
  for(j in seq(m))
    r[,j]=y-beta[j,1]-x*beta[j,2];
#E-steps
 repeat
   {  prest=c(beta,sig,pr);run=run+1
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,sig[j])
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
```

```
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   temp=pk[,j]%*%x;
   beta[j,]=solve(matrix(c(np[j],temp,temp,pk[,j]%*%x^2),nrow=2))%*%c(pk[,j]%*%y,pk[,j]%*%(x*y))
   r[,j]=y-beta[j,1]-beta[j,2]*x;
   sig[j]=sqrt(t(pk[,j])%*%(r[,j]^2)/np[j]);
     }
    dif=max(abs(c(beta,sig,pr)-prest))
    if(dif<acc|run>500){break}
 }
}
 else{
r=matrix(rep(0,2*n),nrow=n);pk=r; sig=sig*c(1,1)
  for(j in seq(m))
    r[,j]=y-beta[j,1]-x*beta[j,2];
#E-steps
 repeat
   {  prest=c(beta,sig,pr);run=run+1
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,sig[j])
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   temp=pk[,j]%*%x;
beta[j,]=solve(matrix(c(np[j],temp,temp,pk[,j]%*%x^2),nrow=2))%*%c(pk[,j]%*%y,pk[,j]%*%(x*y));
   r[,j]=y-beta[j,1]-beta[j,2]*x;
     }
    sig=sqrt(sum(pk*(r^2))/n)*c(1,1)
```

```
    dif=max(abs(c(beta,sig,pr)-prest))
    if(dif<acc|run>500){break}
} }
  theta=c(beta,sig,pr);theta
}

################
#the robust EM algorithm to fit the mixture of linear regression
#########################
mixlinrb<-function(x,y,beta,sig,pr,m=2){
run=0;acc=10^(-4)*max(abs(c(beta,sig,pr)));n=length(x);
if(length(sig)>1)
{
r=matrix(rep(0,2*n),nrow=n);pk=r;
  for(j in seq(m))
    r[,j]=(y-beta[j,1]-x*beta[j,2])/sig[j];
#E-steps
 repeat
   {  prest=c(sig,beta,pr);run=run+1;
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,1)/sig[j]
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   w=pk[,j]*bisquare(r[,j])/r[,j];
   temp=w%*%x;
   beta[j,]=solve(matrix(c(sum(w),temp,temp,w%*%x^2),nrow=2))%*%c(w%*%y,w%*%(x*y))
   r[,j]=(y-beta[j,1]-beta[j,2]*x)/sig[j];
   sig[j]=sqrt(sum(r[,j]^2*sig[j]^2*pk[,j]*biscalew(r[,j]))/np[j]/0.5);
     }
```

```
    dif=max(abs(c(sig,beta,pr)-prest))
    if(dif<acc|run>500){break}
 }  }
 else{ r=matrix(rep(0,2*n),nrow=n);pk=r;sig=sig*c(1,1);
  for(j in seq(m))
    r[,j]=(y-beta[j,1]-x*beta[j,2])/sig[j];
#E-steps
 repeat
   {  prest=c(sig,beta,pr);run=run+1;
   for(j in seq(m))
     {
      pk[,j]=pr[j]*dnorm(r[,j],0,1)/sig[j]
      }
  pk=pk/cbind(apply(pk,1,sum),apply(pk,1,sum));
    #M-step
    np=apply(pk,2,sum);pr=np/n;
   for(j in seq(m))
    {
   w=pk[,j]*bisquare(r[,j])/r[,j];
   #w=pk[,j]*huberpsi(r[,j])/r[,j];
   temp=w%*%x;
   beta[j,]=solve(matrix(c(sum(w),temp,temp,w%*%x^2),nrow=2))%*%c(w%*%y,w%*%(x*y))
   r[,j]=(y-beta[j,1]-beta[j,2]*x)/sig[j];
     }
   sig=sqrt(sum(pk*(r^2*sig[1]^2)*biscalew(r))/n/0.5)*c(1,1)
    dif=max(abs(c(sig,beta,pr)-prest))
    if(dif<acc|run>500){break}
}}
theta=c(beta,sig,pr);theta}
#read tone data in to R;
X=read.table("F:/report/master_report/latex_source/tone/tonedata.txt", header=TRUE)
#tone data without outliers.

x=c(X[,1])
```

```
y=c(X[,2])
m=2;
n=length(x);
beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
pr=c(0.5,0.5);
#sig wothout outliers
sig=c(0.2797);
mix=mixlin(x,y,beta,sig,pr)
mixrb=mixlinrb(x,y,beta,sig,pr)
par(mfrow=c(2,2))
tt=seq(1,3,by=0.01);
ff=mix[1]+mix[3]*tt;
gg=mix[2]+mix[4]*tt
plot(x,y)
lines(tt,ff,lty=2)
lines(tt,gg,lty=2)
ff=mixrb[1]+mixrb[3]*tt;
gg=mixrb[2]+mixrb[4]*tt
plot(x,y)
lines(tt,ff,lty=1)
lines(tt,gg,lty=2)
par(mfrow=c(1,1))
#tone with 3 outliers.
x=c(X[,1],1.4,1.41,1.44)
y=c(X[,2],3.6,3.6,3.5)
m=2;
n=length(x);
beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
pr=c(0.5,0.5);
#sig with 3 outliers
sig=c(0.322);
mix=mixlin(x,y,beta,sig,pr)
mixrb=mixlinrb(x,y,beta,sig,pr)
tt=seq(1,3,by=0.01);
```

```
ff=mix[1]+mix[3]*tt;
gg=mix[2]+mix[4]*tt
plot(x,y,xlab="Actual tone ratio",ylab="Perceived tone ratio")
lines(tt,ff,lty=2)
lines(tt,gg,lty=2)
ff=mixrb[1]+mixrb[3]*tt;
gg=mixrb[2]+mixrb[4]*tt
lines(tt,ff,lty=1,col="red")
lines(tt,gg,lty=1,col="red")
```