ROBUST MIXTURE REGRESSION MODEL FITTING BY LAPLACE DISTRIBUTION

by

YANRU XING

B.Eng., Tianjin Polytechnic University, China, 2002
LL.B., China University of Political Science and Law, China, 2005

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
2013

Approved by:

Major Professor
Weixing Song

Copyright

Abstract

A robust estimation procedure for mixture linear regression models is proposed in this report by assuming the error terms follow a Laplace distribution. EM algorithm is implemented to conduct the estimation procedure of missing information based on the fact that the Laplace distribution is a scale mixture of normal and a latent distribution. Finite sample performance of the proposed algorithm is evaluated by some extensive simulation studies, together with the comparisons made with other existing procedures in this literature. A sensitivity study is also conducted based on a real data example to illustrate the application of the proposed method.

Key words and phrases: Least Absolute Deviation; EM Algorithm; Mixture Regression Model; Normal Mixture; Laplace Distribution

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest appreciation to Professor Weixing Song, who continually and convincingly conveyed the spirit of insistence and creation in regard to research. His exciting attitude in regard to teaching makes him be born for an instructor. Without his guidance and persistent help, this report would not have been possible.

I also would like to thank Professor Kun Chen for his enthusiastic encouragement and valuble critiques. I am grateful for Professor Weixin Yao's assistance and guidance. I obtain most of the theoretical knowledge of Statistics by taking Professor Yao's classes.

Especially, I appreciate Professor Yao's contribution to part of the programming in the simulation study.

I extend my sincere gratitude to Dr. Guihua Bai for providing me a GRA position.

Finally, I wish to thank my husband Huitao Liu for his support and encouragement throughout my study.

Chapter 1

Introduction

Finite mixture models have been applied widely in practical and theoretical area over the years because of its tractability. It can model complex distributions depending on the choice of its appropriate components. Finite mixtures of distributions have constructed the mathematical-based method to model various random phenomena; therefore, many science fields adopt this kind of model to analysis data set, such as, biology, genetics, economics, physical, medicine, social sciences, and so on. Over 100 years ago, famous biometrician Karl Pearson (1894) used mixture models to analyzed the data set of measurements on the ratio of forehead to body length of n=1000 crabs which were sampled from the Bay of Naples. The mixture model was composed with two normal probability density functions with different means and variances in certain proportion respectively. He applied the method of moments to fit this mixture model and obtained his moments-based estimates of the five parameters.

Least absolute deviation (LAD) regression has been widely used in practice if robust estimates are desired. The research on its computation and theoretical properties is abundant in the literature. A detailed survey on this topic can be found in Deilman (1984, 2005). In this report, LAD will be applied to a class of mixture linear regression models to obtain robust estimates of the regression coefficients.

Wei (2012) proposed a robust estimation procedure for the mixture linear regression models based on t distribution by extending McLachlan and Peel (2000)$'s$ work. The research conducted in this report deals with the same questions as in Wei (2012), but with

the LAD technique, the Laplace distribution, instead of t-distribution, is used for achieving robustness. In addition, the implementation of Wei (2012) procedure needs to specify the degrees of freedom in the t-distribution, however, the laplace distribution does not need to do the step of tuning parameters. Moreover, the natural connection between the LAD procedure and the MLE procedure based on Laplace error makes the proposed procedure more appealing.

1.1 Mixture Model Definition

Let $Y_1, \ldots, Y_n$ denote a random sample of size n from g components and suppose that the mixture density $f(y_j;\theta)$ of Y_j can be expressed as following,

$$f(y_j;\theta) = \sum_{i=1}^{g} \pi_i f_i(y_j;\lambda_i), \tag{1.1}$$

where π_i is the weight or proportion of the i^{th} component, $0 \leq \pi_i \leq 1 (i = 1, \ldots, g)$, and $\sum_{i=1}^{g} \pi_i = 1$. $\theta = (\pi_1, \ldots, \pi_g, \lambda_1, \ldots, \lambda_i)$ is the set of unknown parameters and λ_i denote the parameter vector of the density function $f_i(y_j;\lambda_i)$. The $f_i(y_j;\lambda_i)$ is the density function of the i^{th} component. Let z_j be a categorical random variable over from 1 to g with probabilities $\pi_1, \ldots, \pi_g$, respectively.

1.2 Maximum Likelihood Estimation and EM Algorithm

Maximum likelihood (ML) is the most commonly used approach to the fitting of mixture model. The estimate $\hat{\theta}$ is obtained by an appropriate solution of the likelihood equation in regular situation,

$$\frac{\partial L(\theta)}{\partial \theta} = 0.$$

The likelihood function of θ in model (1.1) is formed by the sample size n and given by

$$L(\theta; y) = \prod_{j=1}^{n} (\sum_{i=1}^{g} \pi_i f_i(y_j; \lambda_i)). \tag{1.2}$$

It is equivalent to maximize the log likelihood function when we want to maximize the likelihood function and the log likelihood function is suggested as

$$\begin{aligned} \log L(\theta; y) &= \log(\prod_{j=1}^{n} (\sum_{i=1}^{g} \pi_i f_i(y_j; \lambda_i))) \\ &= \sum_{j=1}^{n} \log(\sum_{i=1}^{g} \pi_i f_i(y_j; \lambda_i)). \end{aligned} \tag{1.3}$$

The ML estimate of parameters θ can be described as

$$\hat{\theta} = \arg\max \sum_{j=1}^{n} \log(\sum_{i=1}^{g} \pi_i f_i(y_j; \lambda_i)). \tag{1.4}$$

Hence, the solution of the following equation is the ML estimate of θ,

$$\frac{\partial \log L(\theta; y)}{\partial \theta} = 0.$$

It is obvious that the above equation does not have explicit solutions unless in trivial case. However, the solutions of the likelihood equation can be obtained by the application of EM algorithm (Dempster et al., 1977).

EM algorithm is a very useful method to handle the incomplete-Data problem. In the procedure of fitting parametric mixture model with EM algorithm, the observed data vector $y = (y_1, \ldots, y_n)^T$ is viewed as being incomplete as the associated component-label vectors, $z = (z_1, \ldots, z_n)^T$, are not observable. Since each y_j come from one of the g components in the mixture model, z_j is a g-dimensional vector where $z_{ij} = (z_j)_i = 1$ or 0 depending on whether y_j arise from the i^{th} component of the mixture model or not. The complete-data vector is

$$y_c = (y^T, z^T)^T.$$

The complete Log likelihood $\log L_c(\theta; y, z)$ for complete-data vector y_c is given by

$$\begin{aligned}\log L_c(\theta; y, z) &= \log \prod_{j=1}^{n} (\prod_{i=1}^{g} (\pi_i f_i(y_j; \lambda_i))^{z_{ij}}) \\ &= \sum_{j=1}^{n} \sum_{i=1}^{g} z_{ij} \log(\pi_i f_i(y_j; \lambda_i)) \\ &= \sum_{i=1}^{g} \sum_{j=1}^{n} z_{ij} (\log \pi_i + \log f_i(y_j; \lambda_i)).\end{aligned} \tag{1.5}$$

The EM algorithm proceeds iteratively in two steps, E-step (expectation step) and M-step (maximization step). E-step handles the unobservable data, which compute the conditional expectation of complete log likelihood given observable vector y and using initial parameter set $\theta^{(0)}$ for θ. The M-step on the $(k+1)^{th}$ iteration maximize the expectation in the E-step with respect to θ and obtain the updated estimates $\theta^{(k+1)}$. The EM algorithm (iterative procedure) can be summarized as:

1. Input initial values $\theta^{(0)} = (\pi_1^{(0)}, \ldots, \pi_g^{(0)}, \lambda_1^{(0)}, \ldots, \lambda_g^{(0)})$ for the parameters.
2. E-step: For the $(k+1)^{th}$ iteration, compute

$$\begin{aligned}Q(\theta, \theta^{(k)}) &= E(\log L_c(\theta; y, z) | y, \theta^{(k)}) \\ &= E(\sum_{j=1}^{n} \sum_{i=1}^{g} z_{ij} \log(\pi_i f_i(y_i; \lambda_i)) | y, \theta^{(k)}) \\ &= \sum_{j=1}^{n} \sum_{i=1}^{g} E(z_{ij} | y, \theta^{(k)}) \log(\pi_i f_i(y_j; \lambda_i)).\end{aligned} \tag{1.6}$$

When compute the expectation of $\log L_c(\theta; y, z)$, we only need to compute the expectation of z_{ij} given y and $\theta^{(k)}$,

$$E(z_{ij} | y, \theta^{(k)}) = \tau_{ij}^{(k+1)} = \frac{\pi_i^{(k)} f_i(y_j; \lambda_i^{(k)})}{\sum_{i=1}^{g} \pi_i^{(k)} f_i(y_j; \lambda_i^{(k)})}. \tag{1.7}$$

3. M-step: compute the estimate of θ which maximizes $Q(\theta, \theta^{(k)})$,

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

4. The E-step and M-step are repeated until certain criterion is satisfied.

1.3 Normal Mixture

Pearson (1894)'s experiment suggests that the normal mixture model is useful in fitting the data which have asymmetrical distributions. We use the normal mixture as an example to conduct EM algorithm. The component normal density has mean μ_i and covariance σ_i^2 and the mixture density function is given by

$$f(y;\theta) = \sum_{i=1}^{g} \pi_i f_i(y;\lambda_i) = \sum_{i=1}^{g} \pi_i \phi_i(y;\mu_i,\sigma_i^2) = \sum_{i=1}^{g} \frac{\pi_i}{\sqrt{2\pi}\sigma_i} e^{-\frac{(y-\mu_i)^2}{2\sigma_i^2}}. \tag{1.8}$$

The EM algorithm for normal mixture model is showed as

1. Input initial values for parameters: $\pi_i^{(0)}$, $\mu_i^{(0)}$, and $\sigma_i^{2(0)}$.

2. E-step: Compute the expectation of the complete log likelihood given $\theta^{(k)}$ and observation y at the $(k+1)^{th}$ iteration. The procedure is the following:

$$Q(\theta,\theta^{(k)}) = E(\log L_c(\theta;y,z)|y,\theta^{(k)}) = \sum_{j=1}^{n}\sum_{i=1}^{g} E(z_{ij}|y,\theta^{(k)}) \log\left(\pi_i f_i(y_j;\lambda_i)\right). \tag{1.9}$$

Let $\tau_{ij}^{(k+1)} = \tau_i(y_j;\theta^{(k)}) = E(z_{ij}|y,\theta^{(k)})$, then

$$Q(\theta,\theta^{(k)}) = \sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}\log\pi_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}\log 2\pi\sigma_i^2 - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}\frac{(y_j-\mu_i)^2}{\sigma_i^2}, \tag{1.10}$$

$$\tau_{ij}^{(k+1)} = \frac{\pi_i^{(k)}\phi_i(y_j;\mu_i^{(k)},\sigma_i^{2(k)})}{\sum_{i=1}^{g}\pi_i^{(k)}\phi_i(y_j;\mu_i^{(k)},\sigma_i^{2(k)})}. \tag{1.11}$$

3. M-step: Take the derivative of $Q(\theta,\theta^{(k)})$ with respect to θ, we obtain

$$\pi_i^{(k+1)} = \sum_{j=1}^{n}\tau_{ij}^{(k+1)}/n, \tag{1.12}$$

$$\mu_i^{(k+1)} = \frac{\sum_{j=1}^{n}\tau_{ij}^{(k+1)}y_j}{\sum_{j=1}^{n}\tau_{ij}^{(k+1)}}, \tag{1.13}$$

$$\sigma_i^{2(k+1)} = \frac{\sum_{j=1}^{n}\tau_{ij}^{(k+1)}(y_j-\mu_i^{(k+1)})^2}{\sum_{j=1}^{n}\tau_{ij}^{(k+1)}}. \tag{1.14}$$

1.4 Mixture of Normal Linear Regression Models

Linear regression model has been studied and applied widely in the analysis of statistical relations between variables. In practice, the data in a data set may originate from different normal population. Hence the mixture of normal linear regression models is very useful to solve the practical problems in many fields, like marketing, biology, agriculture, economics, and so on.

Let Z be a latent class variable and define $Z = i(i = 1, 2, \ldots, g)$. Y is the scalar response variable which depends on the p-dimensional predictor X, and their relations could be constructed by the following linear regression model:

$$Y = X'\beta_i + \varepsilon_i, i = 1, 2, \ldots, g,$$

where $\beta_i = (\beta_{i1},\ldots,\beta_{ip})'$, ε_i and X are independent. The density of ε_i follows normal distribution $f_i(\cdot)$ with mean 0 and variance σ_i^2. Suppose $P(Z = i) = \pi_i$, then the conditional density of Y given X without observing Z is,

$$f(y|x, \theta) = \sum_{i=1}^{g} \pi_i f_i(y; x'\beta_i, \sigma_i^2), \tag{1.15}$$

where $\theta = (\pi_1, \beta_1, \sigma_1^2, \ldots, \pi_g, \beta_g, \sigma_g^2)'$.

Perform the procedures of EM algorithm as in section 1.3 for $f(y|x, \theta)$.

1. Take initial value $\theta^{(0)} = (\pi_1^{(0)}, \ldots, \pi_g^{(0)}, \beta_1^{(0)}, \ldots, \beta_g^{(0)}, \sigma_1^{2(0)}, \ldots, \sigma_g^{2(0)})$.

2. E-step: For the $(k+1)^{th}$ iteration, compute the conditional expectation of the complete log likelihood,

$$E[Z_{ij}|y, \theta^{(k)}] = \tau_{ij}^{(k+1)} = \frac{\pi_i^{(k)} f_i(y_j; x_j'\beta_i^{(k)}, \sigma_i^{2(k)})}{\sum_{i=1}^{g} \pi_i^{(k)} f_i(y_j; x_j'\beta_i^{(k)}, \sigma_i^{2(k)})}, \tag{1.16}$$

where

$$f_i(y_j; x_j'\beta_i^{(k)}, \sigma_i^{2(k)}) = \frac{1}{\sigma_i^{(k)}\sqrt{2\pi}} e^{-\frac{(y_j - x_j'\beta_i^{(k)})^2}{2\sigma_i^{2(k)}}}. \tag{1.17}$$

3. M-step: Update the value of parameters depending on $\theta^{(k)}$.

$$\pi_i^{(k+1)} = \sum_{j=1}^{n} \frac{\tau_{ij}^{(k+1)}}{n}, \tag{1.18}$$

$$\beta_i^{(k+1)} = \arg\min \sum_{j=1}^{n} \tau_{ij}^{(k+1)} (y_j - x_j'\beta_i)(y_j - x_j'\beta_i)' = (\sum_{j=1}^{n} \tau_{ij}^{(k+1)} x_j x_j')^{-1} \sum_{j=1}^{n} \tau_{ij}^{(k+1)} x_j y_j, \quad (1.19)$$

$$\sigma_i^{2(k+1)} = \frac{\sum_{j=1}^{n} \tau_{ij}^{(k+1)} (y_j - x_j'\beta_i^{(k+1)})^2}{\sum_{j=1}^{n} \tau_{ij}^{(k+1)}}. \quad (1.20)$$

4. Iterate E-step and M-step until converge to certain criterion.

Chapter 2

EM Algorithm for Robust Mixture Regression

2.1 Laplace Distribution

Let X be a p-dimensional vector of explanatory variables, and Y be a scalar response variable. We use a linear regression model to investigate the relationship between Y and X. For the mixture linear regression setup, we assume that (X', Y) come from one of the following $g \geq 2$ linear regression models with probability π_i,$i = 1, 2, \ldots, g$,

$$Y = X'\beta_i + \sigma_i \varepsilon_i, \quad i = 1, 2, \ldots, g, \tag{2.1}$$

where $\sum_{i=1}^{g} \pi_i = 1$, β_i's are unknown p-dimensional vectors of regression coefficients, σ_i's are unknown positive scalars. The random error ε_i's are assumed to be independent of X_i's. It is commonly assumed that the density functions of ε_i's are members in a location-scale family with mean 0 and variances 1. In this report the design variable X is assumed to be random, but the proposed estimation procedure also works for the fixed design.

If $g = 1$, LAD estimate of β is the minimizer of the target function $Q(\beta) = \sum_{j=1}^{n} |Y_j - X_j'\beta|$, where $(X_j', Y_j)_{j=1}^{n}$ is a sample from model (2.1). However, if $g > 1$, the formulation of LAD target function is not straightforward since for a sample, we simply do not know which regression model an observation arises from. Our formulation of the LAD target function is motivated by the fact that the maximum likelihood estimate of the regression coefficients

given double exponentially distributed random error is indeed the LAD estimator for $g = 1$. Therefore, for $g \geq 2$ case, we assume that ε_i follows a double exponential distribution with location 0 and scale parameter $1/\sqrt{2}$, which makes the variance of ε_i being 1, $i = 1, 2, \ldots, g$. Then it is easily seen that for a sample $S = \{(X_j', Y_j), j = 1, 2, \ldots, n\}$ from the model (2.1), the log likelihood function of $\theta = (\beta_1, \sigma_1^2, \pi_1, \beta_2, \sigma_2^2, \pi_2, \ldots, \beta_g, \sigma_g^2, \pi_g)$ can be written as

$$\log L(\theta; S) = \sum_{j=1}^{n} \log \sum_{i=1}^{g} \frac{\pi_i}{\sqrt{2}\sigma_i} exp(-\frac{\sqrt{2}|Y_j - X_j'\beta_i|}{\sigma_i}). \tag{2.2}$$

Thus the maximum likelihood estimate of θ can be obtained by maximizing $\log L(\theta; S)$ with respect to θ. Usually no explicit solution can be obtained, and some numerical method will be needed.

For the case $g = 1$, many algorithms are developed in the literature to tackle the minimization problem $\hat{\beta} = \arg\min Q(\beta)$, such as the linear programming, least angle regression, the modified maximum likelihood method by Li and Arce (2004), and others. An often adopted but ad-hoc scheme for finding the solution β is to directly take the derivative of $Q(\beta)$ with respect to β, and set it equal to 0. Here σ^2 is treated as a nuisance parameter. By doing this, we obtain

$$\frac{\partial Q(\beta)}{\partial \beta} = -\sum_{j=1}^{n} X_j sgn(Y_j - X_j'\beta) = 0, \tag{2.3}$$

where $sgn(\cdot)$ is the sign function which takes -1, 0, 1 if the argument is negative, 0 and positive, respectively. Let $w_j = 1/|Y_j - X_j'\beta|$, and rewrite the equation (2.3) as $\sum_{j=1}^{n} w_j X_j (Y_j - X_j'\beta) = 0$. Thus by supplying an initial value β_0 for β, the updated value β can be found by the weighted least square solution

$$\beta_1 = (\sum_{j=1}^{n} w_j X_j X_j')^{-1} \sum_{j=1}^{n} w_j X_j Y_j. \tag{2.4}$$

By iterating the above procedure, one can eventually find an approximate solution to $\arg\min_\theta Q(\theta)$.

A very interesting connection between the iterated weighted least square procedure

stated above and an EM algorithm in conjunction with the Laplace distribution is found in Phillips (2002). For the sake of completeness, we briefly describe the procedure of Phillip (2002) here.

Andrews and Mallows (1974) showed that a Laplace distribution in fact can be expressed as a mixture of a normal distribution and another distribution related to exponential distribution. To be specific, supposed Z and V be two random variables, V has a distribution with density function $v^{-3}exp(-(2v^2)^{-1})$, $v > 0$, and given $V = v$, the conditional distribution of Z is normal with mean 0 and variance $\sigma^2/(2v^2)$. The joint density function $f(z, v)$ of Z and V can be written as

$$f(z,v) = \frac{v}{\sqrt{\pi}\sigma} exp(-\frac{v^2 z^2}{\sigma^2}) \frac{1}{v^3} exp(-\frac{1}{2v^2}). \tag{2.5}$$

Hence, Z marginally has a Laplace distribution with density function $h_\varepsilon(z) = exp(-\sqrt{2}|z|/\sigma)$

$/(\sqrt{2}\sigma)$. Based on this finding, Phillips (2002) developed an EM algorithm to search for the minimizer of $Q(\beta)$.

Considering V as a latent variable, if V could be observed, then the complete log-Likelihood function of $\theta = (\beta, \sigma^2)$, based on the sample $P = (X_j, Y_j, V_j)_{j=1}^n$, is

$$\log L(\theta; P) = -\frac{n}{2}\log \pi\sigma^2 - \frac{1}{\sigma^2}\sum_{j=1}^n V_j^2 (Y_j - X_j'\beta)^2 - \sum_{j=1}^n \log V_j^2 - \frac{1}{2}\sum_{j=1}^n \frac{1}{V_j^2}. \tag{2.6}$$

Following the two steps in EM algorithm procedure, assume that $\theta^{(k)} = (\beta^{(k)}, \sigma^{2(k)})$ is the value in the k^{th} iteration, then in the $(k+1)^{th}$ iteration, we have to compute the conditional expectation of the complete log likelihood function $\log L(\theta; P)$ first, given the observed data set $(Y_j, X_j)_{j=1}^n$ and $\theta = \theta^{(k)}$, the procedure is showed as

$$\begin{aligned} E[\log L(\theta; P)|S] = &-\frac{n}{2}\log \pi\sigma^2 - \frac{\sum_{j=1}^n E[V_j^2|\theta^{(k)}, (X_j, Y_j)_{j=1}^n](Y_j - X_j'\beta)^2}{\sigma^2} \\ &- \sum_{j=1}^n E[\log V_j^2|\theta^{(k)}, (X_j, X_j)_{j=1}^n] - \frac{1}{2}\sum_{j=1}^n E[\frac{1}{V_j^2}|\theta^{(k)}, (X_j, Y_j)_{j=1}^n]. \end{aligned}$$

For the next step, maximize the conditional complete log likelihood expectation with respect to θ. Denote $w_j = E[V_j^2|\theta^{(k)}, (X_j, Y_j)_{(j=1)}^n]$, and we notice that the third and fourth terms on the right hand side of the above function do not involve the unknown regression parameters, such that maximizing the above conditional expectation is equivalent to maximizing the following terms with respect to θ,

$$-\frac{n}{2}\log\sigma^2 - \frac{\sum_{j=1}^n w_j(Y_j - X_j'\beta)^2}{\sigma^2}.$$

Phillips (2002) showed $w_j = E[V_j^2|\theta^{(k)}, (X_j, Y_j)_{j=1}^n] = \sigma^{(k)}/(\sqrt{2}|Y_j - X_j'\beta^{(k)}|)$, this implies that the solution of $\beta^{(k+1)}$ indeed is the same as the one based on (2.4). It is also easy to see that $\sigma^{2(k+1)}$ can be estimated by $\sigma^{2(k+1)} = 2\sum_{j=1}^n w_j(Y_j - X_j'\beta^{(k+1)})^2/n$. The above methodology will be extended to mixture regression setting.

2.2 Mixture Model with Laplace Distribution

In model (2.1), assume that ε_i's follow a Laplace distribution with mean 0 and scale parameter $1/\sqrt{2}$. For $i = 1, 2, \ldots, g, j = 1, 2, \ldots, n$, denote Z_{ij} as latent Bernoulli variables such that

$$Z_{ij} = \begin{cases} 1 & \text{if } j\text{th observation } (X_j, Y_j) \text{ is from } i\text{th component;} \\ 0 & \text{otherwise.} \end{cases}$$

If the full data set $T = \{(X_j, Y_j, Z_{ij})\}_{i=1,2,\ldots,g;j=1,2,\ldots,n}$ are observable, then the complete log likelihood function of $\theta = (\beta_1, \sigma_1^2, \pi_1, \beta_2, \sigma_2^2, \pi_2, \ldots, \beta_g, \sigma_g^2, \pi_g)$ can be written as

$$\log L(\theta; T) = \sum_{j=1}^n \sum_{i=1}^g Z_{ij} \log \frac{\pi_i}{\sqrt{2}\sigma_i} exp(-\frac{\sqrt{2}|Y_j - X_j'\beta_i|}{\sigma_i}). \tag{2.7}$$

From Andrews and Mallows (1974) mentioned above, we know that a Laplace distributed random variable is a scale mixture of a normal random variable and another variable related to exponential distribution. Denote V_j, coupled with (X_j, Y_j), as the latent scale variable, $j = 1, 2, \ldots, n$, then the complete log likelihood function of θ, based on $D =$

$\{X_j, Y_j, V_j, Z_{ij}\}_{i=1,2,\ldots,g;j=1,2,\ldots,n}$, has the form

$$\begin{aligned}\log L(\theta; D) &= \sum_{j=1}^{n}\sum_{i=1}^{g} Z_{ij}\log\pi_i \frac{V_j}{\sqrt{\pi}\sigma_i} exp(-\frac{V_j^2(Y_j - X_j'\beta_i)^2}{\sigma_i^2})\frac{1}{V_j^3} exp(-\frac{1}{2V_j^2}) \\ &= \sum_{j=1}^{n}\sum_{i=1}^{g} Z_{ij}\log\pi_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g} Z_{ij}\log\pi\sigma_i^2 - \sum_{j=1}^{n}\sum_{i=1}^{g} Z_{ij}\frac{V_j^2(Y_j - X_j'\beta_i)^2}{\sigma_i^2} \\ &- \sum_{j=1}^{n}\sum_{i=1}^{g} Z_{ij}\log V_j^2 - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\frac{Z_{ij}}{V_j^2}.\end{aligned} \tag{2.8}$$

Based on EM algorithm principle, for the E-step, we have to calculate the conditional expectation $E[L(\theta; D)|S, \theta^{(0)}]$, where $\theta^{(0)} = (\beta_1^{(0)}, \sigma_1^{2(0)}, \pi_1^{(0)}, \ldots, \beta_g^{(0)}, \sigma_g^{2(0)}, \pi_g^{(0)})$ is a proper initial value for θ and $S = \{(X_j, Y_j)\}_{j=1}^n$. Since the last two terms in (2.8) do not involve the unknown regression parameters, we can simply drop them from the analysis. Thus, to find $E[L(\theta; D)|S, \theta^{(0)}]$, we only need to calculate the following two terms

$$\tau_{ij} = E[Z_{ij}|S, \theta^{(0)}], \quad \delta_{ij} = E[V_j^2|S, \theta^{(0)}, Z_{ij} = 1]. \tag{2.9}$$

Refer to section 1.3, we obtain

$$\tau_{ij}^{(1)} = \frac{\pi_i^{(0)}\sigma_i^{-1(0)} exp(-\sqrt{2}|Y_j - X_j'\beta_i^{(0)}|/\sigma_i^{(0)})}{\sum_{m=1}^{g}\pi_m^{(0)}\sigma_m^{-1(0)} exp(-\sqrt{2}|Y_j - X_j'\beta_m^{(0)}|/\sigma_m^{(0)})}. \tag{2.10}$$

Follow the same thread as in Phillips (2002) to compute δ_{ij}

$$\delta_{ij}^{(1)} = \frac{\sigma_i^{(0)}}{\sqrt{2}|Y_j - X_j'\beta_i^{(0)}|}. \tag{2.11}$$

In the M-step, the following expression will be maximized with respect to π_i's, β_i's, and σ_i^2's,

$$\sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}\log\pi_i - \frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{g}\tau_{ij}\log\sigma_i^2 - \sum_{j=1}^{n}\sum_{i=1}^{g}\frac{\tau_{ij}\delta_{ij}(Y_j - X_j'\beta_i)^2}{\sigma_i^2}. \tag{2.12}$$

The maximizers of parameters will be used for the next iteration.

2.3 EM Algorithm

We propose the following EM algorithm to maximize function (2.12).

1. Choose an initial value for $\theta = (\beta_1, \sigma_1^2, \pi_i, \ldots, \beta_g, \sigma_g^2, \pi_g)$.

2. E-Step: at the $(k+1)^{th}$ iteration, calculate $\tau_{ij}^{(k+1)}$ and $\delta_{ij}^{(k+1)}$ from equation (2.10) and (2.11) with (0) replaced by (k).

3. M-Step: at the $(k+1)$th iteration, use the following formulas to calculate the maximizer of (2.12):

$$\pi_i^{(k+1)} = \frac{1}{n}\sum_{j=1}^{n} \tau_{ij}^{(k+1)}, \tag{2.13}$$

$$\beta_i^{(k+1)} = (\sum_{j=1}^{n} \tau_{ij}^{(k+1)} \delta_{ij}^{(k+1)} X_j X_j')^{-1} (\sum_{j=1}^{n} \tau_{ij}^{(k+1)} \delta_{ij}^{(k+1)} X_j Y_j), \tag{2.14}$$

and

$$\sigma_i^{2(k+1)} = \frac{2\sum_{j=1}^{n} \tau_{ij}^{(k+1)} \delta_{ij}^{(k+1)} (Y_j - X_j'\beta_i^{(k+1)})^2}{\sum_{j=1}^{n} \tau_{ij}^{(k+1)}}. \tag{2.15}$$

4. Repeat steps 2 and 3 until the convergence is obtained.

If we further assume that all σ_i^2 are equal, then in the above EM algorithm, a common initial value for σ_i^2 should be used, and σ^2 can be updated in M-Step by

$$\sigma^{2(k+1)} = \frac{2\sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}^{(k+1)} \delta_{ij}^{(k+1)} (Y_j - X_j'\beta_i^{(k+1)})^2}{n}. \tag{2.16}$$

The robustness of the above EM procedure follows from the adoption of LAD regression. It is also obvious from the formulae of the updated β_i's in each iteration. Note that the factor reciprocally related to the term $|Y_j - X_j'\beta_i^{(k)}|$, meaning that larger residuals give smaller values of $\delta_{ij}^{(k+1)}$, hence, down-weight the corresponding observations when calculating the estimates.

It is easy to see that when updating $\beta_i^{(k+1)}$, the weight δ_{ij} can be simplified to $\delta_{ij} = 1/|Y_j - X_j'\beta_i^{(k)}|$. After updating $\beta_i^{(k+1)}$'s in the $(k+1)^{th}$ iteration, similar to Phillips (2002)

in one population case, we can update σ_i using formula

$$\sigma_i^{(k+1)} = \frac{\sqrt{2}\sum_{j=1}^{n} \tau_{ij}^{(k+1)} \delta_{ij}^{(k+1)} |Y_j - X_j' \beta_i^{(k+1)}|}{\sum_{j=1}^{n} \tau_{ij}^{(k+1)}}. \tag{2.17}$$

Accordingly, when all σ_i^2ś are assumed to be equal, then one can update the common variance by

$$\sigma^{(k+1)} = \frac{\sqrt{2}\sum_{j=1}^{n}\sum_{i=1}^{g} \tau_{ij}^{(k+1)} \delta_{ij}^{(k+1)} |Y_j - X_j' \beta_i^{(k+1)}|}{n}. \tag{2.18}$$

The EM algorithm proposed above for calculating the estimate of β indeed is an iterated reweighted least square (IRLS) procedure, as the one proposed in Schlossmacher (1973) for one population case and the weights are given by $\tau_{ij}^{(k+1)}\delta_{ij}^{(k+1)}$ in the $(k+1)^{th}$ iteration. Extra attention should be paid when programming the proposed EM algorithm. In the case of $g = 1$, Schlossmacher (1973) warned that if a perfect LAD fit occurs, i.e., $Y_j - X_j'\hat{\beta}_i = 0$ for some i, j and $\hat{\beta}_i$, then the algorithm will eventually gives $Y_j - X_j'\beta_i^k \approx 0$ when iteration proceeds. As a result, $\delta_{ij}^{(k+1)}$ which is reciprocally related to $|Y_j - X_j'\beta_i^k|$ will be very large, and numerical instability would follow. Although Phillips (2002) noticed that this problem rarely arises in the case of $g = 1$, this does occur often in our case, which is not out of expectation, simply because more than one regression models provide more chance for a perfect LAD fitting. But simply adopting Schlossmacher (1973)'s weight scheme by setting $\delta_{ij}^{(k+1)} = 0$ whenever $|Y_j - X_j'\beta_i^k| < e$ for a pre-assigned $e > 0$ is not quite reasonable. It makes much sense to allocate big weights for small residuals and small weights for big residuals. A cogent argument on this issue is provided in Phillips (2002). In our simulation study, we simply adopt a hard threshold rule to control the extremely small LAD residuals in each iteration step. Under this rule, $\delta_{ij}^{(k+1)}$ will be assigned a value of 10^6 for any perfect LAD fit. We also tried other threshold values, such as 10^8, 10^{10} in the simulation, all these choices generate almost identical results. For the sake of brevity, we only report the simulation results by using 10^6 as the threshold value.

It is well known that in IRLS procedure, numerical instability could occur if the weights are very small. A common way to deal with this issue is to impose a hard threshold on $\tau_{ij}^{(k+1)}$

obtained in the $(k+1)^{th}$ iteration. Namely, for a pre-specified value e say, if $\tau_{ij}^{(k+1)} > e$, then $\tau_{ij}^{(k+1)}$ itself will be used for the next iteration; otherwise, e will be used as the weight for the next iteration. Same technique is used in Wei (2012). In our simulation study, $e = 10^{-6}$ is adopted.

2.4 Trim high leverage points

Similar to the traditional M-estimate for linear regression and Wei (2012)'s mixture regression by t-distribution, the above EM algorithm based on Laplace distribution is robust against outliers along y-direction, but not in x-direction, which is also confirmed by the sensitivity study conducted in Chapter 3. As a consequence, if there are any high leverage points in the data sets, then the proposed EM algorithm might not follow our expectation, and certain modification would be necessary. An obvious modification is first to identify these high leverage points, then just discard them.

A commonly used method is to calculate the leverage value for each observation using the following formula,

$$h_{jj} = n^{-1} + (n-1)^{-1} MD_j,$$

where $MD_j = (X_j - \bar{X})' S^{-1} (X_j - \bar{X})$, $\bar{X}, S$ are the sample mean and sample covariance matrix of X_j's, respectively. The jth observation will be identified as a high leverage point if $h_{jj} > 2p/n$, where p is the dimension of X. To avoid the masking effect caused by using $\bar{X}$ and S in detecting the high leverage points, like some high leverage points may influence other high leverage points to be identified, some robust estimation of the population mean and covariance matrix of X can be used instead of the sample mean and sample covariance. Wei (2012) adopted the minimum covariance determinant (MCD) estimators for the population mean and covariance matrix, which is implemented by the Fast MCD algorithm developed in Rousseeuw and Van Driessen (1999). Certainly, other robust estimates of the population mean and covariance matrix could be also used for this purpose, for example, the Stahel-Donoho (SD) estimator from Stahel (1981) and Donoho (1982). The jth observation

will be considered as a high leverage point if the resulting MD_j exceeds the threshold $\chi^2_{p,0.975}$. This threshold is proposed by Pison et al. (2002). In the simulation studies, we apply the proposed EM algorithm based on Laplace distribution after removing the observations with $MD_j > \chi^2_{p,0.975}$ using MCD estimator and to calculate MD_j.

Chapter 3

Numerical Studies

In this chapter, an extensive simulation study will be conducted to assess the finite sample performance of the proposed robust estimation procedure. It is well known that the label switching is always an issue when evaluating different estimation methods in mixture models, and there is no widely accepted labeling standard. In our simulation, similar to Wei (2012), we simply choose the labels by minimizing the distance to the true parameter values. The effects of labeling schemes on comparison different estimation procedures deserve an independent research in the future.

3.1 Simulation Studies

In the simulation study, let all components have equal variance. The reason for doing this is that, if the variances are not the same, the log-likelihood function (2.7) is unbounded and goes to infinity if one observation exactly lies on one component line and its corresponding variance goes to 0, which makes the simulation very unstable.

To compare our proposed method with some existing estimation procedures, we generate sample data $(X_{j1}, X_{j2}, Y_j)_{j=1}^n$ from the following two-component mixture regression models which are also used in Wei (2012):

$$Y = \begin{cases} 0 + X_1 + X_2 + \varepsilon_1, & \text{if Z=1,} \\ 0 - X_1 - X_2 + \varepsilon_2, & \text{if Z=2,} \end{cases}$$

where Z indicates which component an observation comes from. That is, the data are generated from a two-component mixture linear regression model with $\beta_1 = (\beta_{10}, \beta_{11}, \beta_{12})' = (0, 1, 1)'$ and $\beta_2 = (\beta_{20}, \beta_{21}, \beta_{22})' = (0, -1, -1)'$. The predictors $X_1 \sim N(0, 1)$ and $X_2 \sim N(0, 1)$ are independent. The random error ε_1 and ε_2 are also independent and have the same distribution as ε. To check the effects of different distributions of ε and the high leverage outliers in x-direction on various estimation methods, the following six cases are considered:

Case I : $\varepsilon \sim N(0, 1)$, standard normal distribution.

Case II: $\varepsilon \sim$ Laplace distribution with mean 0 and variance 1.

Case III: $\varepsilon \sim t_1$, t-distribution with 1 degree of freedom or the Cauchy distribution.

Case IV: $\varepsilon \sim t_3$, t-distribution with 3 degrees of freedom.

Case V: $\varepsilon \sim 0.95N(0, 1) + 0.05N(0, 25)$, a mixture with two normal distributions.

Case VI: $\varepsilon \sim N(0, 1)$ with 5% high leverage outliers being $X_1 = X_2 = 20$ and $Y = 100$.

Case I is often used to evaluate the efficiency of different estimation methods compared to the traditional MLE when the error is exactly normally distributed and there are no outliers. For Case II, the estimation methods proposed in this report will provide the MLE of unknown parameters, which, as in the first case, would serve a reference line to evaluate the performance of other estimation procedures. Both Case III and IV are heavy tailed distributions and often used in literature to mimic the outlier situations. Case V would produce 5% data likely to be low leverage outliers, and in Case VI, 5% observations are replicated serving as the high leverage outliers, which will be used to check the robustness of estimation procedures against the high leverage outliers.

Seven estimation methods will be compared in the simulation study:

1. Maximum likelihood estimate based on normality assumption (MLE).

2. Trimmed likelihood estimator (TLE) proposed by Neykov et al. (2007).

3. Robust modified EM algorithm based on bisquare (Bisquare) proposed by Bai et al. (2012).

4. Robust mixture regression based on t-distribution (Mixregt) proposed by Wei (2012).

5. Trimmed mixture regression based on t-distribution with MCD trimming method (Mixregt-MCD).

6. The proposed robust EM mixture regression based on Laplace-distribution (MixregL).

7. Trimmed mixture regression based on Laplace-distribution with MCD trimming method (MixregL-MCD).

In all simulation studies, the iteration is terminated whenever the difference of the likelihood functions from the current step and the previous step is less than 10^{-6}.

The simulation results are shown in the following tables (Table 1, 2, and 3), which report the mean squared errors (MSE) and bias (Bias) of the parameter estimates for each case in different estimate methods. Sample sizes are 100, 200, and 400 for each case. From the simulation studies, we can see that if the true distribution of ε is normal, the MSEs of MLE procedure are slightly bigger than our proposed method for the first regression component when the sample size is 100, but MLE will show its superiority over all other methods when the sample size gets bigger. For other cases when the distribution of ε has a heavier tail, contaminated by some outliers, or there are high leverage outlier in the data set, then MLE fails to provide reasonable estimates regardless of sample size.

TLE and Bisquare perform very well for all the cases where ε has a lighter tail, except case III where ε has a t-distribution with 1 degree of freedom. The overall performance of the Mixregt proposed by Wei (2012) is satisfying when sample size gets bigger except for the Case VI when high leverage points present in the data set, but this disadvantage could be remedied by the modified procedure Mixregt-MCD.

The simulation results also suggest that the proposed method in this report outperforms or at least is comparable to any other methods. It is rather unexpected that our proposed method performs better than the Mixregt and Mixregt-MCD procedures even when ε has a t-distribution. The bigger MSEs in the latter two procedures might be resulted from the extra step involved in the algorithm, the selection of v, which is the degrees of freedom of the t-distribution.

MCD estimator is mentioned above, which is used in Mixregt-MCD and MixregL-MCD to remove high leverage outliers. For case VI, Mixregt-MCD performs better than Mixregt, but MixregL-MCD doesn't show obvious improvment to MixregL in this simulation.

A common criticism about the EM algorithm is its slow convergence. This can also be seen from our simulation study. The average numbers of iterations to achieve convergence using the MixregL procedure are 96, 97, 78, 98, 102, and 10 for cases from I to VI, respectively, when the sample size is 100; 110, 99, 65, 105, 117, 14 when the sample size is 200; and 124, 111, 41, 123, 119, 17 when the sample size is 400. The number of iterations depend on the choice of the stopping rule. For example, if the iteration is terminated whenever the difference of the likelihood functions from the current step and the previouse step is less than 10^{-4}, then the average numbers of iterations to achieve convergence using the MixregL procedure are 62, 62, 36, 63, 64, 10 for cases from I to VI, respectively, when the sample size is 100. Some accelerating methods might be used here to speed up the convergence, but we shall not seek this possibility in the current report.

3.2 Real Data Example

In this section, a sensitivity study will be conducted which is based on a real data set to compare how outliers affect various different estimation procedures. A typical real data set suitable for mixture regression modeling is the tone data collected in a tone perception experiment of Cohen (1984). In the experiment, a pure fundamental tone was played to a trained musician and electronically generated overtones were added, determined by a

	MLE	TLE	Bisquare	Mixregt	Mixregt-MCD	MixregL	MixregL-MCD
				Case I: $\sim N(0,1)$			
β_{10}	0.130(0.011)	0.139(0.033)	0.143(0.011)	0.124(0.021)	0.163(0.029)	0.093(0.079)	0.090(0.069)
β_{11}	0.160(-0.025)	0.212(-0.195)	0.157(-0.022)	0.130(-0.032)	0.175(-0.115)	0.094(-0.015)	0.113(-0.103)
β_{12}	0.135(-0.034)	0.284(-0.195)	0.171(-0.048)	0.123(-0.004)	0.247(-0.031)	0.088(0.008)	0.165(-0.039)
β_{20}	0.018(-0.003)	0.038(-0.004)	0.021(-0.001)	0.022(-0.012)	0.022(0.008)	0.028(-0.026)	0.027(-0.001)
β_{21}	0.021(-0.016)	0.030(0.011)	0.023(-0.017)	0.021(-0.006)	0.029(-0.011)	0.027(-0.001)	0.035(-0.021)
β_{22}	0.018(-0.009)	0.024(0.034)	0.019(-0.014)	0.021(-0.010)	0.030(-0.020)	0.026(-0.010)	0.042(-0.017)
π_1	0.005(0.003)	0.007(0.025)	0.005(0.005)	0.005(0.013)	0.007(0.016)	0.005(0.017)	0.007(0.022)
				Case II: $\sim$ Laplace(1)			
β_{10}	0.177(0.006)	0.075(0.007)	0.137(-0.016)	0.085(0.012)	0.123(-0.001)	0.058(0.022)	0.060(0.020)
β_{11}	0.145(-0.040)	0.097(-0.107)	0.142(-0.054)	0.084(-0.029)	0.150(-0.033)	0.050(-0.024)	0.080(-0.033)
β_{12}	0.152(-0.009)	0.084(-0.077)	0.126(-0.000)	0.080(-0.021)	0.150(-0.026)	0.055(-0.006)	0.063(-0.020)
β_{20}	0.016(-0.002)	0.013(0.004)	0.013(-0.002)	0.011(-0.007)	0.016(-0.019)	0.010(-0.010)	0.015(-0.026)
β_{21}	0.021(-0.017)	0.013(0.007)	0.014(-0.019)	0.012(-0.008)	0.018(-0.030)	0.011(-0.004)	0.019(-0.020)
β_{22}	0.016(-0.006)	0.013(0.019)	0.013(-0.002)	0.012(-0.002)	0.020(0.009)	0.012(0.003)	0.026(0.018)
π_1	0.004(0.004)	0.004(0.019)	0.004(0.016)	0.004(0.015)	0.005(0.012)	0.003(0.013)	0.005(0.009)
				Case III: $\sim t_1$			
β_{10}	242.992(-0.120)	3.200(-0.150)	1.683(-0.116)	1.708(-0.026)	0.945(-0.075)	0.163(0.061)	0.122(0.034)
β_{11}	174.667(-1.568)	1.886(-0.170)	1.571(-0.347)	1.990(-0.252)	1.621(-0.535)	0.521(-0.377)	0.561(-0.430)
β_{12}	148.108(-1.770)	1.797(-0.033)	1.642(-0.306)	2.410(-0.447)	1.538(-0.360)	0.548(-0.412)	0.418(-0.405)
β_{20}	244.822(0.172)	1.526(0.065)	0.910(0.024)	0.113(-0.020)	3.237(-0.173)	0.032(-0.024)	0.025(-0.038)
β_{21}	175.583(-1.080)	0.774(-0.129)	0.489(-0.088)	0.079(-0.041)	0.949(-0.102)	0.032(0.052)	0.047(0.081)
β_{22}	142.862(-0.454)	0.773(-0.065)	0.580(-0.116)	0.112(-0.049)	0.968(-0.028)	0.037(0.052)	0.048(0.054)
π_1	0.084(0.213)	0.039(0.060)	0.047(0.105)	0.023(0.093)	0.028(0.108)	0.022(0.070)	0.023(0.083)
				Case IV: $\sim t_3$			
β_{10}	1.568(-0.129)	0.238(0.007)	0.460(0.006)	0.529(0.031)	0.475(0.126)	0.131(0.065)	0.130(0.108)
β_{11}	0.997(-0.234)	0.264(-0.135)	0.341(-0.041)	0.361(0.010)	0.772(-0.109)	0.176(-0.021)	0.183(-0.041)
β_{12}	1.240(-0.024)	0.239(-0.096)	0.375(-0.058)	0.394(-0.010)	0.804(-0.040)	0.132(0.013)	0.186(-0.046)
β_{20}	0.723(-0.029)	0.038(-0.008)	0.063(0.013)	0.034(0.002)	0.077(-0.018)	0.032(-0.005)	0.030(-0.009)
β_{21}	0.188(-0.028)	0.034(0.010)	0.085(-0.034)	0.037(-0.005)	0.062(-0.014)	0.042(0.004)	0.052(-0.018)
β_{22}	0.115(0.031)	0.026(-0.010)	0.041(-0.013)	0.029(-0.018)	0.166(-0.027)	0.035(-0.015)	0.048(0.003)
π_1	0.028(0.025)	0.007(0.037)	0.009(0.030)	0.006(0.011)	0.014(0.035)	0.007(0.012)	0.007(0.021)
				Case V: $\sim 0.95N(0,1) + 0.05N(0,25)$			
β_{10}	2.243(-0.020)	0.124(0.046)	0.202(0.042)	0.152(0.015)	0.350(0.037)	0.097(0.034)	0.098(0.042)
β_{11}	1.366(0.054)	0.282(-0.209)	0.225(-0.037)	0.153(-0.029)	0.528(-0.106)	0.100(-0.008)	0.160(-0.056)
β_{12}	2.117(-0.113)	0.221(-0.190)	0.217(-0.056)	0.163(-0.050)	0.705(0.094)	0.099(-0.030)	0.175(0.023)
β_{20}	1.767(0.159)	0.030(0.013)	0.021(0.011)	0.026(0.020)	0.028(-0.004)	0.029(0.008)	0.035(-0.003)
β_{21}	1.277(-0.122)	0.034(0.001)	0.028(-0.023)	0.022(-0.009)	0.035(0.010)	0.026(-0.005)	0.040(0.008)
β_{22}	0.284(-0.006)	0.027(0.011)	0.029(-0.009)	0.120(-0.036)	0.038(-0.017)	0.027(-0.006)	0.044(-0.020)
π_1	0.040(0.015)	0.010(0.034)	0.008(0.020)	0.007(0.015)	0.009(0.012)	0.005(0.006)	0.009(0.013)
				Case VI: $\sim N(0,1)$ with 5% high leverage outliers			
β_{10}	18.364(-2.878)	0.173(0.002)	0.152(0.015)	2.456(0.169)	0.175(-0.032)	0.036(0.080)	0.111(0.092)
β_{11}	5.876(1.422)	0.248(-0.209)	0.200(-0.068)	3.444(1.473)	0.219(-0.055)	0.056(-0.037)	0.133(-0.012)
β_{12}	6.520(1.641)	0.219(-0.168)	0.227(-0.091)	3.589(1.517)	0.262(0.006)	0.042(-0.014)	0.153(-0.046)
β_{20}	11.938(2.451)	0.036(-0.002)	0.023(-0.011)	0.023(0.002)	0.027(0.019)	0.015(-0.058)	0.032(0.011)
β_{21}	12.578(3.316)	0.028(0.000)	0.025(-0.014)	0.053(0.139)	0.027(0.010)	0.013(0.033)	0.042(0.000)
β_{22}	12.561(3.315)	0.022(-0.025)	0.020(-0.019)	0.053(0.136)	0.023(-0.017)	0.012(0.021)	0.046(0.004)
π_1	0.113(0.165)	0.007(0.017)	0.007(0.003)	0.007(-0.074)	0.006(0.005)	0.005(0.030)	0.006(0.011)

Table 3.1: MSE(Bias) of Point Estimates for $n = 100$

	MLE	TLE	Bisquare	Mixregt	Mixregt-MCD	MixregL	MixregL-MCD
				Case I: $\sim N(0,1)$			
β_{10}	0.043(-0.010)	0.073(-0.002)	0.044(-0.010)	0.047(-0.022)	0.052(-0.030)	0.053(0.008)	0.039(0.022)
β_{11}	0.041(-0.007)	0.129(-0.162)	0.044(-0.007)	0.035(0.008)	0.064(-0.005)	0.044(-0.023)	0.070(-0.032)
β_{12}	0.040(-0.020)	0.174(-0.199)	0.044(-0.018)	0.044(-0.007)	0.057(-0.030)	0.051(-0.015)	0.067(-0.037)
β_{20}	0.009(-0.006)	0.018(-0.016)	0.009(-0.007)	0.008(-0.010)	0.009(0.015)	0.013(-0.025)	0.013(0.005)
β_{21}	0.008(-0.006)	0.012(0.020)	0.008(-0.007)	0.009(-0.001)	0.013(-0.009)	0.014(0.013)	0.018(-0.002)
β_{22}	0.010(-0.014)	0.017(0.000)	0.012(-0.015)	0.008(-0.006)	0.013(-0.011)	0.012(0.004)	0.020(-0.001)
π_1	0.002(0.010)	0.004(0.019)	0.003(0.012)	0.002(0.007)	0.002(0.005)	0.002(0.008)	0.002(0.006)
				Case II: $\sim$ Laplace(1)			
β_{10}	0.046(-0.006)	0.039(0.003)	0.033(0.015)	0.027(-0.002)	0.030(-0.005)	0.026(0.015)	0.022(0.013)
β_{11}	0.048(-0.039)	0.033(-0.064)	0.034(0.017)	0.026(-0.021)	0.032(0.000)	0.020(-0.020)	0.024(-0.007)
β_{12}	0.043(0.009)	0.028(-0.058)	0.030(-0.004)	0.033(0.018)	0.036(-0.012)	0.020(0.002)	0.024(-0.011)
β_{20}	0.009(-0.007)	0.007(-0.007)	0.007(-0.010)	0.006(-0.004)	0.005(-0.001)	0.005(-0.012)	0.005(-0.006)
β_{21}	0.008(-0.020)	0.007(0.007)	0.007(-0.019)	0.005(-0.005)	0.007(-0.010)	0.004(-0.004)	0.007(-0.010)
β_{22}	0.009(-0.006)	0.006(0.007)	0.006(-0.009)	0.005(0.006)	0.009(-0.009)	0.005(0.007)	0.009(-0.004)
π_1	0.002(0.004)	0.002(0.019)	0.002(0.023)	0.002(0.006)	0.002(0.005)	0.002(0.005)	0.002(0.003)
				Case III: $\sim t_1$			
β_{10}	286.806(1.711)	1.026(-0.123)	1.256(-0.042)	0.369(0.025)	0.411(0.049)	0.067(0.049)	0.048(0.074)
β_{11}	36.053(-0.902)	0.906(0.103)	0.981(-0.222)	0.716(-0.436)	0.808(-0.471)	0.268(-0.362)	0.406(-0.471)
β_{12}	85.816(-0.726)	0.904(-0.024)	1.031(-0.222)	0.957(-0.483)	0.810(-0.485)	0.289(-0.349)	0.434(-0.506)
β_{20}	283.651(1.486)	0.774(0.128)	0.587(-0.018)	0.048(0.001)	0.060(-0.065)	0.013(-0.052)	0.013(-0.042)
β_{21}	30.042(1.056)	0.201(0.052)	0.273(-0.012)	0.043(0.008)	0.063(-0.009)	0.017(0.078)	0.028(0.109)
β_{22}	49.441(0.368)	0.253(-0.032)	0.281(0.019)	0.043(-0.012)	0.047(-0.004)	0.019(0.074)	0.028(0.101)
π_1	0.067(0.240)	0.020(0.031)	0.033(0.094)	0.016(0.067)	0.025(0.087)	0.025(0.076)	0.033(0.106)
				Case IV: $\sim t_3$			
β_{10}	0.600(-0.069)	0.080(-0.020)	0.121(-0.030)	0.108(-0.024)	0.155(0.036)	0.064(0.017)	0.078(0.022)
β_{11}	0.486(-0.167)	0.082(-0.082)	0.096(0.019)	0.113(0.009)	0.181(-0.041)	0.072(0.005)	0.112(-0.049)
β_{12}	0.778(-0.050)	0.082(-0.095)	0.078(-0.005)	0.132(-0.056)	0.194(-0.020)	0.071(-0.034)	0.103(-0.017)
β_{20}	3.107(-0.153)	0.019(-0.008)	0.016(0.002)	0.015(-0.005)	0.015(-0.007)	0.015(-0.015)	0.017(-0.015)
β_{21}	0.459(-0.026)	0.017(0.014)	0.016(-0.021)	0.013(-0.006)	0.020(-0.014)	0.014(0.006)	0.021(-0.013)
β_{22}	0.227(0.046)	0.014(0.009)	0.016(-0.043)	0.018(-0.001)	0.018(-0.020)	0.017(0.002)	0.019(-0.017)
π_1	0.029(0.018)	0.004(0.035)	0.004(0.031)	0.004(0.009)	0.004(0.012)	0.003(0.004)	0.004(0.010)
				Case V: $\sim 0.95N(0,1) + 0.05N(0,25)$			
β_{10}	1.077(-0.002)	0.072(0.029)	0.051(0.006)	0.076(-0.022)	0.100(0.024)	0.056(0.004)	0.064(0.023)
β_{11}	0.834(0.031)	0.095(-0.142)	0.046(0.014)	0.084(-0.036)	0.113(0.000)	0.054(-0.029)	0.075(-0.024)
β_{12}	0.675(-0.121)	0.097(-0.125)	0.055(0.006)	0.073(-0.002)	0.096(0.007)	0.060(0.008)	0.068(0.013)
β_{20}	0.348(0.062)	0.013(0.007)	0.010(0.012)	0.016(-0.012)	0.013(-0.019)	0.014(-0.020)	0.017(-0.026)
β_{21}	0.042(0.072)	0.011(0.014)	0.009(0.001)	0.012(0.003)	0.014(0.010)	0.016(0.007)	0.017(0.009)
β_{22}	0.036(0.067)	0.012(0.016)	0.010(-0.005)	0.012(0.005)	0.015(-0.023)	0.014(0.008)	0.018(-0.019)
π_1	0.016(-0.023)	0.003(0.020)	0.003(0.014)	0.003(0.004)	0.003(0.003)	0.002(-0.001)	0.003(0.000)
				Case VI: $\sim N(0,1)$ with 5% high leverage outliers			
β_{10}	12.459(-2.191)	0.061(0.006)	0.044(-0.008)	1.745(0.011)	0.054(-0.015)	0.021(0.057)	0.050(-0.004)
β_{11}	4.875(1.543)	0.078(-0.093)	0.060(-0.021)	3.111(1.523)	0.065(-0.031)	0.025(-0.041)	0.064(-0.043)
β_{12}	4.678(1.468)	0.087(-0.132)	0.056(-0.033)	2.967(1.475)	0.067(0.013)	0.031(-0.037)	0.067(-0.028)
β_{20}	15.169(2.671)	0.012(-0.013)	0.010(-0.007)	0.010(-0.009)	0.010(0.000)	0.009(-0.063)	0.016(-0.023)
β_{21}	12.212(3.243)	0.010(0.015)	0.008(0.007)	0.031(0.134)	0.013(0.006)	0.009(0.037)	0.015(-0.006)
β_{22}	13.057(3.364)	0.016(-0.004)	0.012(-0.002)	0.027(0.133)	0.014(-0.022)	0.007(0.027)	0.017(-0.006)
π_1	0.147(0.221)	0.003(0.013)	0.003(0.004)	0.008(-0.085)	0.002(0.007)	0.005(0.027)	0.003(0.003)

Table 3.2: MSE(Bias) of Point Estimates for $n = 200$

	MLE	TLE	Bisquare	Mixregt	Mixregt-MCD	MixregL	MixregL-MCD
				Case I: $\sim N(0,1)$			
β_{10}	0.018(-0.006)	0.041(0.012)	0.020(-0.005)	0.019(0.004)	0.027(0.008)	0.025(0.018)	0.031(0.014)
β_{11}	0.020(-0.002)	0.108(-0.178)	0.021(-0.001)	0.018(-0.014)	0.028(-0.014)	0.024(-0.028)	0.034(-0.029)
β_{12}	0.018(-0.006)	0.096(-0.171)	0.020(0.000)	0.016(0.008)	0.031(0.012)	0.029(-0.001)	0.042(-0.012)
β_{20}	0.004(0.003)	0.009(0.002)	0.004(0.002)	0.005(-0.006)	0.005(0.012)	0.008(-0.010)	0.008(0.014)
β_{21}	0.004(0.004)	0.007(0.020)	0.004(0.002)	0.004(-0.009)	0.006(-0.002)	0.006(-0.005)	0.009(0.002)
β_{22}	0.004(-0.005)	0.006(0.013)	0.004(-0.006)	0.005(-0.004)	0.006(0.000)	0.007(0.003)	0.008(0.009)
π_1	0.001(0.000)	0.002(-0.001)	0.001(0.002)	0.001(0.001)	0.002(0.005)	0.001(0.000)	0.002(0.006)
				Case II: $\sim$ Laplace(1)			
β_{10}	0.022(-0.005)	0.012(0.012)	0.015(-0.003)	0.012(-0.004)	0.013(0.003)	0.010(0.007)	0.012(0.010)
β_{11}	0.014(0.008)	0.013(-0.041)	0.010(0.005)	0.012(0.003)	0.018(-0.013)	0.011(0.005)	0.017(-0.007)
β_{12}	0.016(-0.006)	0.017(-0.050)	0.012(-0.004)	0.011(-0.013)	0.016(0.000)	0.008(-0.007)	0.014(0.005)
β_{20}	0.004(-0.003)	0.003(-0.003)	0.003(-0.003)	0.002(0.001)	0.002(0.000)	0.002(0.002)	0.002(-0.001)
β_{21}	0.004(-0.013)	0.003(0.005)	0.003(-0.015)	0.003(-0.009)	0.004(-0.003)	0.003(-0.004)	0.004(-0.001)
β_{22}	0.004(-0.011)	0.004(0.012)	0.003(-0.009)	0.003(-0.003)	0.004(-0.006)	0.002(-0.003)	0.003(-0.003)
π_1	0.001(0.002)	0.001(0.016)	0.001(0.022)	0.001(0.004)	0.001(0.006)	0.001(0.001)	0.001(0.004)
				Case III: $\sim t_1$			
β_{10}	313.757(-0.917)	0.735(-0.040)	0.631(-0.083)	0.170(0.026)	0.154(0.002)	0.016(0.073)	0.017(0.076)
β_{11}	278.219(-3.135)	0.398(0.097)	0.607(-0.187)	0.538(-0.240)	0.485(-0.257)	0.194(-0.352)	0.322(-0.454)
β_{12}	455.172(-1.369)	0.399(0.059)	0.716(-0.146)	0.428(-0.219)	0.484(-0.200)	0.197(-0.361)	0.351(-0.462)
β_{20}	313.757(-0.917)	0.021(-0.001)	0.514(-0.052)	0.024(-0.010)	0.021(-0.002)	0.008(-0.061)	0.008(-0.067)
β_{21}	269.680(-1.135)	0.032(0.003)	0.047(0.034)	0.016(0.009)	0.022(-0.003)	0.011(0.092)	0.015(0.099)
β_{22}	453.695(0.630)	0.093(-0.009)	0.083(0.014)	0.017(0.012)	0.020(-0.002)	0.012(0.094)	0.016(0.102)
π_1	0.061(0.247)	0.008(0.003)	0.016(0.062)	0.010(0.036)	0.008(0.037)	0.037(0.160)	0.038(0.161)
				Case IV: $\sim t_3$			
β_{10}	0.301(0.020)	0.037(-0.008)	0.038(-0.010)	0.039(-0.014)	0.059(-0.016)	0.033(0.002)	0.044(0.005)
β_{11}	0.211(-0.046)	0.039(-0.070)	0.044(0.049)	0.034(-0.013)	0.071(-0.008)	0.028(-0.019)	0.049(-0.033)
β_{12}	0.227(-0.049)	0.037(-0.081)	0.034(0.021)	0.046(0.000)	0.045(0.009)	0.031(0.008)	0.048(-0.043)
β_{20}	0.066(0.018)	0.008(-0.017)	0.007(-0.007)	0.006(-0.007)	0.006(0.011)	0.008(-0.011)	0.006(0.008)
β_{21}	0.069(0.055)	0.007(0.001)	0.006(-0.025)	0.007(-0.005)	0.009(-0.008)	0.007(0.003)	0.010(0.005)
β_{22}	0.069(0.055)	0.009(0.006)	0.008(-0.025)	0.008(0.009)	0.010(-0.001)	0.008(0.011)	0.012(0.003)
π_1	0.010(-0.017)	0.002(0.023)	0.002(0.023)	0.002(0.004)	0.003(0.007)	0.002(-0.001)	0.003(0.003)
				Case V: $\sim 0.95N(0,1) + 0.05N(0,25)$			
β_{10}	0.098(0.000)	0.041(0.005)	0.024(0.004)	0.029(-0.007)	0.038(0.015)	0.034(0.009)	0.042(0.028)
β_{11}	0.394(0.028)	0.048(-0.095)	0.021(0.027)	0.022(0.011)	0.044(-0.012)	0.025(0.003)	0.040(-0.012)
β_{12}	0.081(-0.050)	0.051(-0.119)	0.022(0.014)	0.026(0.001)	0.045(0.012)	0.032(0.000)	0.048(-0.001)
β_{20}	0.041(0.015)	0.006(0.003)	0.005(0.002)	0.006(-0.002)	0.006(0.006)	0.008(-0.006)	0.008(0.003)
β_{21}	0.088(0.046)	0.006(0.010)	0.005(-0.008)	0.006(0.006)	0.009(0.004)	0.008(0.009)	0.011(0.009)
β_{22}	0.135(0.041)	0.007(0.024)	0.004(0.000)	0.005(0.002)	0.008(0.000)	0.007(0.008)	0.011(0.007)
π_1	0.007(-0.033)	0.001(0.003)	0.001(0.006)	0.001(0.000)	0.002(-0.002)	0.002(-0.003)	0.002(-0.007)
				Case VI: $\sim N(0,1)$ with 5% high leverage outliers			
β_{10}	9.355(-1.688)	0.033(0.010)	0.020(-0.010)	1.369(0.227)	0.021(-0.014)	0.013(0.065)	0.029(0.002)
β_{11}	5.188(1.667)	0.049(-0.102)	0.023(-0.011)	2.478(1.473)	0.027(-0.002)	0.014(-0.049)	0.033(-0.037)
β_{12}	4.187(1.307)	0.039(-0.098)	0.021(-0.007)	2.606(1.514)	0.029(0.007)	0.017(-0.034)	0.031(-0.015)
β_{20}	11.697(2.305)	0.005(0.002)	0.004(0.003)	0.005(0.005)	0.005(0.004)	0.007(-0.047)	0.007(-0.002)
β_{21}	11.586(3.309)	0.006(0.011)	0.005(0.012)	0.021(0.125)	0.006(0.004)	0.004(0.026)	0.009(0.005)
β_{22}	12.442(3.437)	0.006(0.003)	0.005(0.003)	0.020(0.122)	0.006(-0.005)	0.005(0.028)	0.010(0.000)
π_1	0.140(0.204)	0.002(0.004)	0.001(-0.006)	0.008(-0.089)	0.001(0.005)	0.040(0.020)	0.001(0.002)

Table 3.3: MSE(Bias) of Point Estimates for $n = 400$

stretching ratio (stretchratio). A value of 2 for the stretch ratio corresponds to the harmonic pattern usually heard in tratitional definite pitched instruments. The musician was asked to tune an adjustable tone to the octave above the fundamental tone, and a measurement called "tuned" gives the ratio of the adjusted tone to the fundamental. 150 pairs of (tuned, stretchratio) values are obtained for the same musician. The variable "strechratio" is treated as a response variable and "tuned" as a predictor. The setup of the experiment indicates two mixture components in the model, and the scatter plot of the data collected from the experiment confirms this point. To investigate the impact of different types of outliers on various procedures, we first add 5 original pairs, (3, 4), to the original data set as outliers in the y-direction. The circles in all plots denote the original data points, and the star denotes the outilers. The right-hand plots in all the figures below have the same y-scales as in the left-hand plots.

The left-hand plot in Figure 3.1 clearly shows that the fitting by MixregL and Bisquare are almost identical, and they all provide very good fit including Mixregt. In the right-hand plot of Figure 3.1, Bisquare fit is looked as baseline to compare to methods TLE and MLE. The fittings of MLE and TLE are affected severely by the outliers. 43 iterations are use for MixregL in this set up to achieve the convergence.

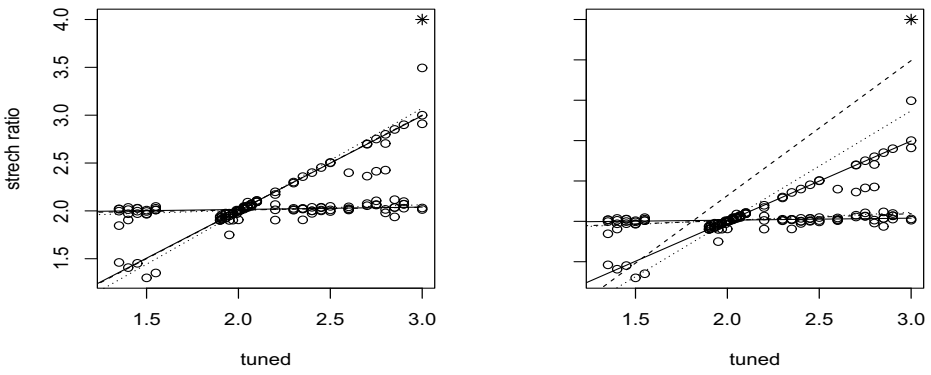

Figure 3.1: Mixture Linear Fitting with Outlier $(3, 4)$

Left panel: solid line – Bisquare, dashed line – MixregL, dotted line – Mixregt,
Right panel: solid line – Bisquare, dashed – TLE, dotted line – MLE

Increase the y-value of the outliers from 4 to 4.5 to further investigate how the outliers in y direction affect the fitting of various procedures, and we find that the fitting results as shown in Figure 3.2 suggest that the performance of Bisquare, MixregL, and Mixregt procedures are quite stable, while the fitting of TLE and MLE procedure dragged more severely towards the outliers. For this case, 43 iterations are also used for MixregL to achieve convergence.

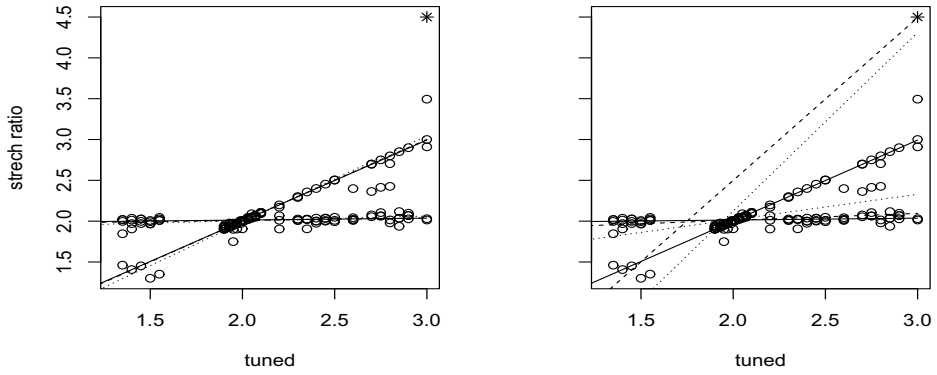

Figure 3.2: Mixture Linear Fitting with Outlier $(3, 4.5)$

Left panel: solid line – Bisquare, dashed line – MixregL, dotted line – Mixregt,
Right panel: solid line – Bisquare, dashed – TLE, dotted line – MLE

Then we add 10 identical pairs, (0, 3), to the original data set as high leverage outliers. The left-hand plot in Figure 3.3 shows that both Bisquare and MixregL give a reasonable fit, but surprisingly the fitting of Mixregt is affected severely by the high leverage outliers, which imply that Mixregt is not quite robust to the outliers in $x-$direction. From the right-hand plot in Figure 3.3, we find that MLE has inferior performance against the outliers, but TLE works better than MLE and the performance of TLE is comparable to MixregL. Here, 77 iterations are needed to get convergence for the MixregL procedure.

Finally 10 identical pairs (0,4) were added to the original data set as outliers both in x and y-direction. The left-hand plot in Figure 3.4 shows that Bisquare continues to provide a robust fit, MixregL barely keeps a vague two-line structure, and Mixregt is affected severely by the outliers. The right-hand plot in Figure 3.4 shows that the fitting of MLE is still the

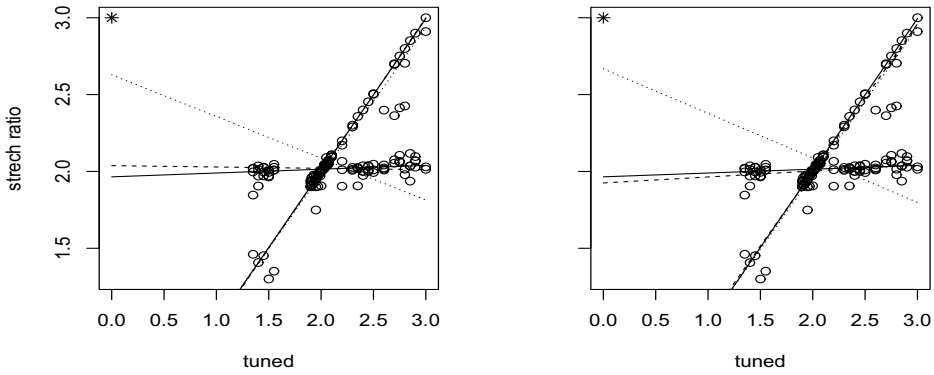

Figure 3.3: Mixture Linear Fitting with Outlier $(0, 3)$

Left panel: solid line – Bisquare, dashed line – MixregL, dotted line – Mixregt,
Right panel: solid line – Bisquare, dashed – TLE, dotted line – MLE

worst, but the fitting of TLE works fine. 22 iterations were used for MixregL to achieve convergence.

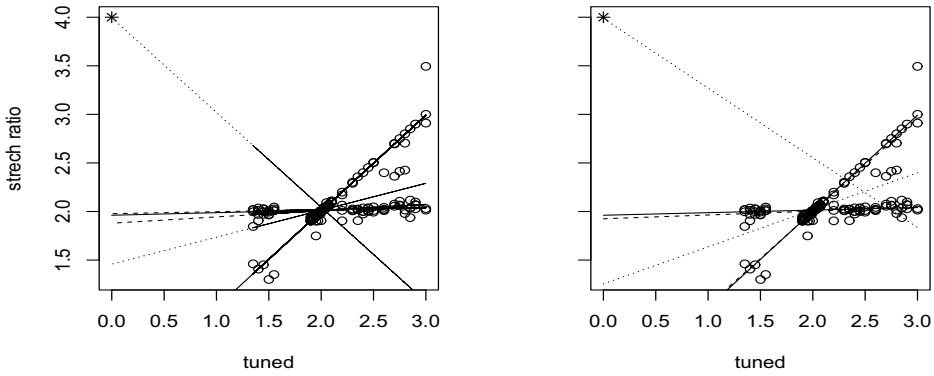

Figure 3.4: Mixture Linear Fitting with Outlier $(0, 4)$

Left panel: solid line – Bisquare, dashed line – MixregL, dotted line – Mixregt,
Right panel: solid line – Bisquare, dashed – TLE, dotted line – MLE

In all the scenatios, the Bisquare performed uniformly better than all the other fitting procedures, although the simulation studies show that Bisquare is less satisfying in some cases, such as when $\varepsilon \sim t-$distributions. Instead of modifying the log likelihood objective function, the Bisquare procedure tries to modify the existing EM algorithm for mixture regression models by replacing the least squares criterion with a robust criteria in the M

step. See Bai (2012) et al. Generally MixregL performed better than Mixret, but both procedures are not quite robust to the high leverage outliers. We also applied Mixregt-MCD and MixregL-MCD to the data set. Both procedures can successfully remove the high leverage outliers and give similar results to the Bisquare.

Chapter 4

Conclusion

This report proposed a new robust estimation procedure tailored to mixture linear regression models by assuming that the random error has a Laplace distribution. The robustness is achieved essentially by the LAD procedure, and implemented by the EM algorithm. Efficiency and effectiveness of the proposed EM algorithm relies upon the fact that the Laplace distribution is a scale mixture of a normal distribution and a distribution related to the exponential distribution. The simulation study shows that the proposed method is superior to or comparable to existing robust estimation procedures in all simulation setups. However, if the proposed method is applied to the real data example and compared with other procedures, the fitting performance of various procedures shows that when high leverage outliers exist, the trimmed version of the proposed procedure should be used.

Bibliography

[1] Andrews, D.F. and Mallows, C.L. (1974). Scale mixtures of normal distributions. *J.R.S.S. (B)*, **36**(1), 99-102.

[2] Bai, X., Yao, W., and Boyer, J. E. (2012). Robust fitting of mixture regression models. *Computational Statistics and Data Analysis*, **56**, 2347-2359.

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

[4] Dielman, T.E. (1984). Leasl absolute value estimation in regression models: An annotated bibliography. *Communications in Statistics - Theory and Methods*, **4**, 513-541.

[5] Dielman, T.E. (2005). Least absolute value regression: recent contributions. *Journal of Statistical Computation and Simulation*, **75**(4), 263-286.

[6] Donoho, D. L. (1982). Breakdown properties of multivariate location estimators. Qualifying paper, Harvard University, Boston.

[7] Li, Y. and Arce, G.R. (2004). A maximum likelihood approach to least absolute deviation regression. *Jouranl of applied signal processing*, **12**, 1762-1769.

[8] McLachlan, G.J., Peel, D., (2000). Finite mixture models. Wiley, New York.

[9] Neykov, N., Filzmoser, P., Dimova, R., and Neytchev, P. (2007). Robust fitting of mixtures using the trimmed likelihood estimator. *Computational Statistics and Data Analysis*, **52**, 299-308.

[10] Phillips, R.F. (2002). Least absolute deviations estimation via the EM algorithm. *Statistics and Computing*, **12**, 281-285.

[11] Pison, G., Van Aelst, S. and Willems, G. (2002). Small sample corrections for LTS and MCD. *Metrika*, **55**, 111-123.

[12] Rousseeuw, P.J., Van Driessen, K., (1999). A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, **41**, 212-223.

[13] Schlossmacher, E.J. (1973). An iterative technique for absolute deviations curve fitting. *JASA*, **68**, 857?59.

[14] Stahel, W. A. (1981). Robuste Schätzungen: Infinitesimale Optimalität und Schätzungen von Kovarianzmatrizen. Ph.D. thesis, ETH Zürich.

[15] Wei, Y. (2012). Robust mixture regression models using t-distribution. Master Report advisor, Dr. Yao, W. Department of Statistics, Kansas State University.

Appendix A

R Code of Simulation Study

1. Leverage Detection Programm

```
lev.p=function(X,method="classic")
 {
   if(method=="sd")
     {
       # Stahel-Donoho Estimator
       # Package needed: rrcov
       Temp=CovSde(X)
       mx=Temp@center
       cx=Temp@cov
     }
   else if(method=="mcd")
     {
      # Fast MCD Estimate
      # Package Needed: robustbase
      Temp=covMcd(X)
      mx=Temp$center
      cx=Temp$cov
     }
   else
     {
       # Sample mean and variance
       mx=apply(X,2,mean)
       cx=var(X)
     }
    list(mean=mx,covar=cx)
  }
```

2. Solving Label Switching

```
slab=function(theta)
 {
   theta0=c(0.25,0,1,1,0.75,0,-1,-1)
   ind1=c(1,2,3,4,5,6,7,8)
   ind2=c(5,6,7,8,1,2,3,4)
   res1=sum((theta[ind1]-theta0)^2)
   res2=sum((theta[ind2]-theta0)^2)
   theta=theta[ind1]*(res1<=res2)+theta[ind2]*(res1>res2)
   return(theta)
 }
```

3. EM algorithm to fitting the Mixture of Linear Regression (Wei and Yao 2012)

```
 #  mixlin estimates the mixture regression parameters by MLE based on
 #    ONE initial value
   mixlin=function(x,y,bet,sig,pr,m=2)
  {
    run=0;
    n=length(y);
    X=cbind(rep(1,n),x);
    if(length(sig)>1 )
     { #the case when the variance is unequal
       r=matrix(rep(0,m*n),nrow=n);
       pk=r;
       lh=0;
       for(j in seq(m))
        {
          r[,j]=y-X%*%bet[j,];
          lh=lh+pr[j]*dnorm(r[,j],0,sig[j]);
        }
       lh=sum(log(lh));
    #E-steps
     repeat
      {
        prest=c(bet,sig,pr);
        run=run+1;
        plh=lh;
        for(j in seq(m))
          {
            pk[,j]=pr[j]*pmax(10^(-6),dnorm(r[,j],0,sig[j]))
          }
            pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
    #M-step
```

```
  np=apply(pk,2,sum);
  pr=np/n;
  lh=0;
  for(j in seq(m))
   {
    w=diag(pk[,j]);
    bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
    r[,j]= y-X%*%bet[j,]; sig[j]=sqrt(t(pk[,j])%*%(r[,j]^2)/np[j]);
    lh=lh+pr[j]*dnorm(r[,j],0,sig[j]);}
    lh=sum(log(lh));
    dif=lh-plh;
    if(dif<1e-6|run>500){break}
   }
  }
else
 { #the case when the variance is equal
  r=matrix(rep(0,m*n),nrow=n);pk=r; lh=0;
  for(j in seq(m))
   {
    r[,j]=y-X%*%bet[j,];lh=lh+pr[j]*dnorm(r[,j],0,sig);}
    lh=sum(log(lh));
   #E-steps
   repeat
    {
      prest=c(bet,sig,pr);run=run+1;plh=lh;
      for(j in seq(m))
       {
         pk[,j]=pr[j]* pmax(10^(-6),dnorm(r[,j],0,sig))
       }
      pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
   #M-step
     np=apply(pk,2,sum);pr=np/n;
    for(j in seq(m))
      {
       w=diag(pk[,j]);
       bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
       r[,j]= y-X%*%bet[j,]; }
       sig=sqrt(sum(pk*(r^2))/n);
       lh=0;
       for(j in seq(m))
        {
          lh=lh+pr[j]*dnorm(r[,j],0,sig);}
          lh=sum(log(lh));
          dif=lh-plh;
          if((dif<1e-6)|(run>500)){break}
```

```
          }
        sig=sig*rep(1,m)
       }
     est=list(theta= matrix(c(bet,sig,pr),nrow=m),likelihood=lh,run=run,
            diflh=dif)
     est
   }
```

4. MLE Based on Normality Assumption

```
#                EM Algorithm
 mix.MLE.normal=function(x,y,beta,sigma,prob,group,tol=1e-6)
   {
     n=length(y);
     X=cbind(rep(1,n),x);
     run=1;
     if(length(sigma)>1)
      {
        gsumi=matrix(0,n,group);
        for(i in seq(group))
          {
           gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma[i]);
          }
        prest=sum(log(apply(gsumi,1,sum)));
       repeat
        {
          tji=matrix(0,n,group)
          for(i in seq(group))
               {
                tji[,i]=prob[i]*pmax(dnorm(y,X%*%beta[i,],sigma[i]),10^(-6));
               }
          tji=solve(diag(apply(tji,1,sum)))%*%tji;
          prob=apply(tji,2,mean);
          sse=rep(0,group)
          for(i in seq(group))
           {
             beta[i,]=lm(y~x,weight=tji[,i])$coefficient;
             sse[i]=sum(tji[,i]*(y-X%*%beta[i,])^2);
             sigma[i]=sqrt(sse[i]/sum(tji[,i]))
           }
         sigma2=sigma^2;
         for(i in seq(group))
          {
           gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma[i]);
```

```
    }
   latest=sum(log(apply(gsumi,1,sum)));
   dif=abs(latest-prest);
   prest=latest;
   run=run+1;
   if(dif<=tol|run>500){break}
  }
 }
else
  {
   gsumi=matrix(0,n,group);
   for(i in seq(group))
     {
      gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma);
     }
   prest=sum(log(apply(gsumi,1,sum)));
  repeat
   {
     tji=matrix(0,n,group)
    for(i in seq(group))
       {
         tji[,i]=prob[i]*pmax(dnorm(y,X%*%beta[i,],sigma),10^(-6));
       }
   tji=solve(diag(apply(tji,1,sum)))%*%tji;
    prob=apply(tji,2,mean);
    sse=rep(0,group);
    for(i in seq(group))
     {
       beta[i,]=lm(y~x,weight=tji[,i])$coefficient;
       sse[i]=sum(tji[,i]*(y-X%*%beta[i,])^2);
     }
   sigma2=sum(sse)/n;
   sigma=sqrt(sigma2);
   for(i in seq(group))
    {
     gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma);
    }
   latest=sum(log(apply(gsumi,1,sum)));
   dif=abs(latest-prest);
   prest=latest;
   run=run+1;
   if(dif<=tol|run>500){break}
  }
 }
 btemp=c(prob[1],beta[1,],prob[2],beta[2,]);
```

```
    btemp=slab(btemp);
    prob=c(btemp[1],btemp[5]);
    beta=rbind(btemp[2:4],btemp[6:8]);
    result=cbind(beta,sigma2,prob)
    result
  }
```

5. Trimmed MLE (Wei and Yao 2012)

```
trimmix=function(x,y,k=2,alpha=0.9,bet,sig,pr,acc=1e-6)
  {
    n=length(y);
    n1=round(n*alpha);
    x=matrix(x,nrow=n);
    a=dim(x);
    p=a[2]+1;
    X=cbind(rep(1,n),x);
    if(dim(bet)[2]==k){bet=t(bet)};
    lh=0
    for (i in seq(k))
      {
       lh=lh+pr[i]*dnorm(y-X%*%bet[i,],0,sig[1])
      }
    ind=order(-lh);run=0;
    obj=sum(log(lh[ind[1:n1]]));
    repeat
      {
       pobj=obj;
       run=run+1;
       x1=x[ind[1:n1],];
       y1=y[ind[1:n1]];
       fit=mixlin(x1,y1,bet,sig,pr,k);
       fit=fit$theta;
       bet=matrix(fit[1:(p*k)],nrow=k);
       sig=fit[p*k+1];
       pr=fit[(p*k+k+1):(p*k+2*k)];
       lh=0;
       for(i in seq(k))
         {
          lh=lh+pr[i]*dnorm(y-X%*%bet[i,],0,sig[1]);
         }
       ind=order(-lh);
       obj=sum(log(lh[ind[1:n1]]));dif=obj-pobj;
       if(dif<acc|run>50){break}
```

```
      }
    if(length(sig)<2){sig=rep(sig,k)};
     btemp=c(pr[1],bet[1,],pr[2],bet[2,]);
     btemp=slab(btemp);
     pr=c(btemp[1],btemp[5]);
     bet=rbind(btemp[2:4],btemp[6:8]);
    theta=matrix(c(bet,sig,pr),nrow=k);
    est=list(theta=theta,likelihood=obj,diflikelihood=dif,run=run)
    est
  }
```

6. Robust Based on Bisquare Function (Bai, Yao, and Boyer 2012)

```
bisquare=function(t,k=4.685)
   {
     out=t*pmax(0,(1-(t/k)^2))^2;
     out
   }
biscalew=function(t)
   {
     t[which(t==0)]=min(t[which(t!=0)])/10;
     out=pmin(1-(1-t^2/1.56^2)^3,1)/t^2;
     out
   }
mixbi2<-function(x,y,bet,sig,pr,m=2)
  {
    run=0;
    acc=10^(-4)*max(abs(c(bet,sig,pr)));
    n=length(y);
    X=cbind(rep(1,n),x);
    p=dim(X)[2];
    if(length(sig)>1)
      {
        r=matrix(rep(0,m*n),nrow=n);
        pk=r;
        for(j in seq(m))
         {
          r[,j]=(y-X%*%bet[j,])/sig[j];
         }
    #E-steps
     repeat
      {
        prest=c(sig,bet,pr);run=run+1;
        for(j in seq(m))
```

```
         {
           pk[,j]=pr[j]*pmax(10^(-6),dnorm(r[,j],0,1))/sig[j]
         }
        pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
        #M-step
        np=apply(pk,2,sum);pr=np/n;
        r[which(r==0)]=min(r[which(r!=0)])/10;
        for(j in seq(m))
         {
          w=diag(pk[,j]*bisquare(r[,j])/r[,j]);
          bet[j,]= solve(t(X)%*%w%*%X+10^(-3)*diag(rep(1,p)))%*%t(X)%*%w%*%y;
          r[,j]= (y-X%*%bet[j,])/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,bet,pr)-prest))
        if(dif<acc|run>500){break}
      }
   }
else
  {
    r=matrix(rep(0,m*n),nrow=n);
    pk=r;
    for(j in seq(m))
      {
       r[,j]=(y-X%*%bet[j,])/sig;
      }
    #E-steps
    repeat
     {
       prest=c(sig,bet,pr);
       run=run+1;
       for(j in seq(m))
        {
          pk[,j]=pr[j]*pmax(10^(-6),dnorm(r[,j],0,1))/sig
        }
       pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
   #M-step
      np=apply(pk,2,sum);
      pr=np/n;
      r[which(r==0)]=min(r[which(r!=0)])/10;
      for(j in seq(m))
       {
        w=diag(pk[,j]*bisquare(r[,j])/r[,j]);
        bet[j,]=solve(t(X)%*%w%*%X+10^(-3)*diag(rep(1,p)))%*%t(X)%*%w%*%y;
        r[,j]=(y-X%*%bet[j,])/sig;
```

```
      }
     sig=sqrt(sum(pk*(r^2*sig[1]^2)*biscalew(r))/n/0.5)
     dif=max(abs(c(sig,bet,pr)-prest))
     if(dif<acc|run>500){break}
    }
   sig=rep(sig,m);
 }
btemp=c(pr[1],bet[1,],pr[2],bet[2,]);
btemp=slab(btemp);
pr=c(btemp[1],btemp[5]);
bet=rbind(btemp[2:4],btemp[6:8]);
theta=matrix(c(bet,sig,pr),nrow=m);
est=list(theta=theta,difpar=dif,run=run)
est
}
```

7. Main Program: Simulation Method MLE, Bisquare, and TLE

```
library(MASS)
begining=Sys.time()
BV=array(0,dim=c(42,6,3));
total=200
kk=1;
for(n in c(100,200,400))
 {
  jj=1;
  for(dist in seq(6))
      {
        set.seed(88888)
        b101=b111=b121=b201=b211=b221=pi11=rep(0,total)
        b102=b112=b122=b202=b212=b222=pi12=rep(0,total)
        b103=b113=b123=b203=b213=b223=pi13=rep(0,total)
        for(k in seq(total))
        {
          g=2;              # Number of Groups
          u=runif(n,0,1);   # A random number for assigning groups
          p1=(u<=0.25);     # probability of group 1
          p2=1-p1;          # probability of group 2
          x1=rnorm(n,0,1);  # 1: normal,
          x2=rnorm(n,0,1);
          e1n=rnorm(n,0,1);
          e2n=rnorm(n,0,1);
          e1L=rexp(n,sqrt(2))-rexp(n,sqrt(2)); # 2: laplace,
          e2L=rexp(n,sqrt(2))-rexp(n,sqrt(2));
```

```
 e1t1=rt(n,1);   # 3: t(1),
 e2t1=rt(n,1);
 e1t3=rt(n,3);   # 4: t(3),
 e2t3=rt(n,3);
 uu=runif(n,0,1)
 e1mn=(uu<0.95)*rnorm(n,0,1)+(uu>=0.95)*rnorm(n,0,5)
              # 5: normal_mixture,
 e2mn=(uu<0.95)*rnorm(n,0,1)+(uu>=0.95)*rnorm(n,0,5)
 e1=e1n*((dist==1)|(dist==6))+e1L*(dist==2)+e1t1*(dist==3)
       +e1t3*(dist==4)+e1mn*(dist==5)
 e2=e2n*((dist==1)|(dist==6))+e2L*(dist==2)+e2t1*(dist==3)
       +e2t3*(dist==4)+e2mn*(dist==5)
 y1=0+x1+x2+e1;
 y2=0-x1-x2+e2;
 x=cbind(x1,x2);
 y=y1*p1+y2*p2;
 if(dist==6)   # 6: normal+0.05outlier
  {
   nout=round(n*0.95)+1;
   x[nout:n,]=20;
   y[nout:n]=100;
  }
# Initial Values
 prob=c(0.5,0.5);
 b1=c(0.1,0.9,0.9);
 b2=c(-0.1,-0.9,-0.9);
 beta=rbind(b1,b2);
 clust=kmeans(y,2)
 sig01=sd(y[clust$cluster==1])
 sig02=sd(y[clust$cluster==2])
 sig0=ifelse((is.na(sig01)|(sig01==0)),sig02,sig01)
 group=2;
 # MLE
 result=mix.MLE.normal(x,y,beta,sig0,prob,group,1e-6);
 b101[k]=result[1,1]
 b111[k]=result[1,2]
 b121[k]=result[1,3]
 b201[k]=result[2,1]
 b211[k]=result[2,2]
 b221[k]=result[2,3]
 pi11[k]=result[1,5]
 # TLE
 result=trimmix(x,y,2,0.9,beta,sig0,prob,1e-6)
 b102[k]=result$theta[1,1]
 b112[k]=result$theta[1,2]
```

```
            b122[k]=result$theta[1,3]
            b202[k]=result$theta[2,1]
            b212[k]=result$theta[2,2]
            b222[k]=result$theta[2,3]
            pi12[k]=result$theta[1,5]
            # Bisquare
            result=mixbi2(x,y,beta,sig0,prob,group);
            b103[k]=result$theta[1,1]
            b113[k]=result$theta[1,2]
            b123[k]=result$theta[1,3]
            b203[k]=result$theta[2,1]
            b213[k]=result$theta[2,2]
            b223[k]=result$theta[2,3]
            pi13[k]=result$theta[1,5]
          }
          tb10=0; tb11=1;  tb12=1;
          tb20=0; tb21=-1; tb22=-1;
          tpi1=0.25;
           # MLE
          Rmle=cbind(b101-tb10,b111-tb11,b121-tb12,b201-tb20,
                              b211-tb21,b221-tb22,pi11-tpi1);
          BV[seq((jj-1)*7+1,jj*7),1:2,kk]=cbind(apply(Rmle,2,
                         function(x){mean(x^2)}),apply(Rmle,2,mean))
          # TML
          Rtml=cbind(b102-tb10,b112-tb11,b122-tb12,b202-tb20,
                            b212-tb21,b222-tb22,pi12-tpi1);
          BV[seq((jj-1)*7+1,jj*7),3:4,kk]=cbind(apply(Rtml,2,
                      function(x){mean(x^2)}),apply(Rtml,2,mean))
          # Bisquare
          Rbsq=cbind(b103-tb10,b113-tb11,b123-tb12,b203-tb20,
                               b213-tb21,b223-tb22,pi13-tpi1);
          BV[seq((jj-1)*7+1,jj*7),5:6,kk]=cbind(apply(Rbsq,2,
                    function(x){mean(x^2)}),apply(Rbsq,2,mean))
          jj=jj+1;
         }
        kk=kk+1;
    }
    ending=Sys.time()
    ending-begining
    BV_MLE_TLE_B2=BV;


8.  Robust Based on t-Distribution (Wei and Yao 2012)

 # Definition of t density
```

```
dent=function(y,mu,sig,v)
 {
  est=gamma((v+1)/2)*sig^(-1)/((pi*v)^(1/2)*gamma(v/2)*
      (1+(y-mu)^2/(sig^2*v))^(0.5*(v+1)));
  est
 }
BV=matrix(0,nrow=42,ncol=2)
n=200    # Sample Size
total=200
for(dist in seq(6))
 {
  set.seed(98765)
  b107=b117=b127=b207=b217=b227=pi17=rep(0,total)
  for(k in seq(total))
    {
      repeat{
      u=runif(n,0,1);   # A random number for assigning groups
      p1=(u<=0.25);     # probability of group 1
      p2=1-p1;          # probability of group 2
      x1=rnorm(n,0,1);
      x2=rnorm(n,0,1);
      e1n=rnorm(n,0,1);  # 1: normal,
      e2n=rnorm(n,0,1);
      e1L=rexp(n,sqrt(2))-rexp(n,sqrt(2)); # 2: laplace,
      e2L=rexp(n,sqrt(2))-rexp(n,sqrt(2));
      e1t1=rt(n,1);   # 3: t(1),
      e2t1=rt(n,1);
      e1t3=rt(n,3);   # 4: t(3),
      e2t3=rt(n,3);
      u=runif(n,0,1)
      e1mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)
                  # 5: normal_mixture,
      e2mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)
      e1=e1n*((dist==1)|(dist==6))+e1L*(dist==2)+e1t1*(dist==3)
            +e1t3*(dist==4)+e1mn*(dist==5)
      e2=e2n*((dist==1)|(dist==6))+e2L*(dist==2)+e2t1*(dist==3)
            +e2t3*(dist==4)+e2mn*(dist==5)
      y1=0+x1+x2+e1;
      y2=0-x1-x2+e2;
      x=cbind(x1,x2);
      y=y1*p1+y2*p2;
      if(dist==6)   # 6: normal+0.05outlier
        {
          nout=round(n*0.95)+1;
          x[nout:n,]=20;
```

```
    y[nout:n]=100;
   }
   # Initial Values
#prob=c(0.5,0.5);
#  b1=c(0.1,0.9,0.9);
#  b2=c(-0.1,-0.9,-0.9);
#  beta=rbind(b1,b2);
prob=c(0.5,0.5);
b1=c(-0.5,-0.9,-0.9);
b2=c(0.5,-0.5,0.5);
beta=rbind(b1,b2);
clust=kmeans(y,2)
sig01=sd(y[clust$cluster==1])
sig02=sd(y[clust$cluster==2])
if(is.na(sig01)){sig01=0};
if(is.na(sig02)){sig02=0};
if((sig01==0)|(sig02=0)){sig0=max(sig01,sig02)} else
  {sig0=(sig01+sig02)/2};
group=2;
X=cbind(rep(1,n),x);
m=2;
maxv=15;
likhod=flag=rep(0,15);
theta=array(0,dim=c(2,5,15));
for(v in seq(maxv))
 {
   bet=beta
   sig=sig0;
   pr=prob;
   run=0;
   n=length(y);
   X=cbind(rep(1,n),x);
   lh=-10^10;
   a=dim(x);
   p=a[2]+1;
   pk=u=r=matrix(rep(0,m*n),nrow=n);
   for(j in seq(m))
    {
      r[,j]=(y-X%*%bet[j,])/sig;
    }
  repeat
   {
    run=run+1;
    prelh=lh;
    for(j in seq(m))
```

```
       {
        pk[,j]=pr[j]*dent(y, X%*%bet[j,],sig,v);
        u[,j]=(v+1)/(v+r[,j]^2)
       }
     lh=sum(log(apply(pk,1,sum)));
     pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
     dif=lh-prelh;
     if(!is.finite(lh)){flag[v]=1; break}
     if(dif<1e-6|run>500){flag[v]=0; break}
     np=apply(pk,2,sum);pr=np/n;
     for(j in seq(m))
      {
       w=diag(pk[,j]*u[,j]);
       bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
       r[,j]=y-X%*%bet[j,];
      }
    sig=sqrt(sum(pk*r^2*u)/sum(pk));
    r=r/sig;
   }
 sig=sig*rep(1,m);
 theta[,,v]= matrix(c(bet,pr,sig),nrow=m);
 likhod[v]=lh;
}
if(all(flag==0)){break}
}
fv=which(likhod==max(likhod))
if(fv<15)
 {result=theta[,,fv]} else
   {
    run=0;
    n=length(y);
    X=cbind(rep(1,n),x);
    r=matrix(rep(0,m*n),nrow=n);
    pk=r;
    lh=0
    for(j in seq(m))
     {
      r[,j]=y-X%*%bet[j,];
      lh=lh+pr[j]*dnorm(r[,j],0,sig);
     }
    lh=sum(log(lh));
    repeat
      {
       prest=c(bet,sig,pr);
       run=run+1;
```

```
          plh=lh;
          for(j in seq(m))
            {
             pk[,j]=pr[j]* pmax(10^(-6),dnorm(r[,j],0,sig))
            }
          pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
          np=apply(pk,2,sum);
          pr=np/n;
          for(j in seq(m))
           {
             w=diag(pk[,j]);
             bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
             r[,j]= y-X%*%bet[j,];
           }
          sig=sqrt(sum(pk*(r^2))/n);
          lh=0;
          for(j in seq(m))
             {
              lh=lh+pr[j]*dnorm(r[,j],0,sig);
             }
          lh=sum(log(lh));
          dif=lh-plh;
          if((dif<1e-6)|(run>500)){break}
         }
       sig=sig*rep(1,m)
       result=matrix(c(bet,pr,sig),nrow=m)
      }
    btemp=c(result[1,4], result[1,1:3], result[2,4], result[2,1:3]);
    btemp=slab(btemp);
    prob=c(btemp[1],btemp[5]);
    beta=rbind(btemp[2:4],btemp[6:8]);
    rest7=cbind(beta,prob)
    b107[k]=rest7[1,1]
    b117[k]=rest7[1,2]
    b127[k]=rest7[1,3]
    b207[k]=rest7[2,1]
    b217[k]=rest7[2,2]
    b227[k]=rest7[2,3]
    pi17[k]=rest7[1,4]
 }
 tb10=0;  tb11=1;   tb12=1;
 tb20=0;  tb21=-1;   tb22=-1;
 tpi1=0.25;
# Mixregt
RL=cbind(b107-tb10,b117-tb11,b127-tb12,b207-tb20,b217-tb21,
```

```
                    b227-tb22,pi17-tpi1);
      BV[seq((dist-1)*7+1,dist*7),1:2]=cbind(apply(RL,2,function(x)
                                                {mean(x^2)}),apply(RL,2,mean))
      print(BV[seq((dist-1)*7+1,dist*7),1:2])
   }
   round(BV,3)
```

9. Robust Based on t-Distribution (Wei and Yao 2012)

```
#      Mixregt_MCD
#      Definition of t density
   dent=function(y,mu,sig,v)
    {
     est=gamma((v+1)/2)*sig^(-1)/((pi*v)^(1/2)*gamma(v/2)*
         (1+(y-mu)^2/(sig^2*v))^(0.5*(v+1)));
     est
    }
   BV=matrix(0,nrow=42,ncol=2)
   n=400;    # Sample Size
   total=200;
   for(dist in seq(6))
    {
     set.seed(98765)
     b107=b117=b127=b207=b217=b227=pi17=rep(0,total)
     for(k in seq(total))
       {
         repeat
        {
         n=400;          # Sample Size
         u=runif(n,0,1);   # A random number for assigning groups
         p1=(u<=0.25);     # probability of group 1
         p2=1-p1;          # probability of group 2
         x1=rnorm(n,0,1);
         x2=rnorm(n,0,1);
         e1n=rnorm(n,0,1);  # 1: normal,
         e2n=rnorm(n,0,1);
         e1L=rexp(n,sqrt(2))-rexp(n,sqrt(2)); # 2: laplace,
         e2L=rexp(n,sqrt(2))-rexp(n,sqrt(2));
         e1t1=rt(n,1);   # 3: t(1),
         e2t1=rt(n,1);
         e1t3=rt(n,3);   # 4: t(3),
         e2t3=rt(n,3);
         u=runif(n,0,1)
         e1mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)  # 5: normal_mixture,
```

```
e2mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)
e1=e1n*((dist==1)|(dist==6))+e1L*(dist==2)+e1t1*(dist==3)
      +e1t3*(dist==4)+e1mn*(dist==5)
e2=e2n*((dist==1)|(dist==6))+e2L*(dist==2)+e2t1*(dist==3)
      +e2t3*(dist==4)+e2mn*(dist==5)
y1=0+x1+x2+e1;
y2=0-x1-x2+e2;
x=cbind(x1,x2);
y=y1*p1+y2*p2;
if(dist==6)   # 6: normal+0.05outlier
  {
    nout=round(n*0.95)+1;
    x[nout:n,]=20;
    y[nout:n]=100;
  }
  # Initial Values
prob=c(0.5,0.5);
b1=c(0.1,0.9,0.9);
b2=c(-0.1,-0.9,-0.9);
beta=rbind(b1,b2);
clust=kmeans(y,2)
sig01=sd(y[clust$cluster==1])
sig02=sd(y[clust$cluster==2])
if(is.na(sig01)){sig01=0};
if(is.na(sig02)){sig02=0};
if((sig01==0)|(sig02=0)){sig0=max(sig01,sig02)} else
  {sig0=(sig01+sig02)/2};
group=2;
mv=lev.p(x,method="mcd");
TX=cbind(x[,1]-mv$mean[1],x[,2]-mv$mean[2]);
ind=which(diag(TX%*%solve(mv$covar)%*%t(TX))>=qchisq(0.975,2));
if(length(ind)>0)
 {
  x=x[-ind,];
  y=y[-ind];
 }
x1=x[,1];
x2=x[,2];
n=length(y);
X=cbind(rep(1,n),x);
m=2;
maxv=15;
likhod=flag=rep(0,15);
theta=array(0,dim=c(2,5,15));
for(v in seq(maxv))
```

```
 {
   bet=beta
   sig=sig0;
   pr=prob;
   run=0;
   n=length(y);
   X=cbind(rep(1,n),x);
   lh=-10^10;
   a=dim(x);
   p=a[2]+1;
   pk=u=r=matrix(rep(0,m*n),nrow=n);
   for(j in seq(m))
    {
      r[,j]=(y-X%*%bet[j,])/sig;
    }
  repeat
   {
    run=run+1;
    prelh=lh;
    for(j in seq(m))
      {
       pk[,j]=pr[j]*dent(y, X%*%bet[j,],sig,v);
       u[,j]=(v+1)/(v+r[,j]^2)
      }
    lh=sum(log(apply(pk,1,sum)));
    pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
    dif=lh-prelh;
    if(!is.finite(lh)){flag[v]=1; break}
    if(dif<1e-6|run>500){flag[v]=0; break}
    np=apply(pk,2,sum);pr=np/n;
    for(j in seq(m))
     {
      w=diag(pk[,j]*u[,j]);
      bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
      r[,j]=y-X%*%bet[j,];
     }
   sig=sqrt(sum(pk*r^2*u)/sum(pk));
   r=r/sig;
  }
 sig=sig*rep(1,m);
 theta[,,v]= matrix(c(bet,pr,sig),nrow=m);
 likhod[v]=lh;
}
if(all(flag==0)){break}
}
```

```
fv=which(likhod==max(likhod))
if(fv<15)
 {result=theta[,,fv]} else
   {
    run=0;
    n=length(y);
    X=cbind(rep(1,n),x);
    r=matrix(rep(0,m*n),nrow=n);
    pk=r;
    lh=0
    for(j in seq(m))
     {
      r[,j]=y-X%*%bet[j,];
      lh=lh+pr[j]*dnorm(r[,j],0,sig);
     }
    lh=sum(log(lh));
    repeat
      {
       prest=c(bet,sig,pr);
       run=run+1;
       plh=lh;
       for(j in seq(m))
         {
          pk[,j]=pr[j]* pmax(10^(-6),dnorm(r[,j],0,sig))
         }
       pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
       np=apply(pk,2,sum);
       pr=np/n;
       for(j in seq(m))
        {
          w=diag(pk[,j]);
          bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
          r[,j]= y-X%*%bet[j,];
        }
       sig=sqrt(sum(pk*(r^2))/n);
       lh=0;
       for(j in seq(m))
          {
           lh=lh+pr[j]*dnorm(r[,j],0,sig);
          }
       lh=sum(log(lh));
       dif=lh-plh;
       if((dif<1e-6)|(run>500)){break}
      }
    sig=sig*rep(1,m)
```

```
            result=matrix(c(bet,pr,sig),nrow=m)
           }
         btemp=c(result[1,4], result[1,1:3], result[2,4], result[2,1:3]);
         btemp=slab(btemp);
         prob=c(btemp[1],btemp[5]);
         beta=rbind(btemp[2:4],btemp[6:8]);
         rest7=cbind(beta,prob)
         b107[k]=rest7[1,1]
         b117[k]=rest7[1,2]
         b127[k]=rest7[1,3]
         b207[k]=rest7[2,1]
         b217[k]=rest7[2,2]
         b227[k]=rest7[2,3]
         pi17[k]=rest7[1,4]
       }
       tb10=0;  tb11=1;   tb12=1;
       tb20=0;  tb21=-1;   tb22=-1;
       tpi1=0.25;
      # Mixregt_MCD
      RL=cbind(b107-tb10,b117-tb11,b127-tb12,b207-tb20,b217-tb21,
             b227-tb22,pi17-tpi1);
      BV[seq((dist-1)*7+1,dist*7),1:2]=cbind(apply(RL,2,function(x)
           {mean(x^2)}),apply(RL,2,mean))
      print(BV[seq((dist-1)*7+1,dist*7),1:2])
   }
   round(BV,3)
```

10. Robust Mixture Regression By Laplace Distribution

```
#               EM Algorithm
   library(MASS)
   denLp=function(y,mu,sigma)
     {
       exp(-abs(sqrt(2)*(y-mu))/sigma)/(sqrt(2)*sigma)
     }
   BV=matrix(0,nrow=42,ncol=2)
   n=400    # Sample Size
   total=200
   for(dist in seq(6))
     {
       set.seed(98765)
       b107=b117=b127=b207=b217=b227=pi17=rep(0,total)
       for(k in seq(total))
         {
```

```
repeat
{
u=runif(n,0,1);   # A random number for assigning groups
p1=(u<=0.25);     # probability of group 1
p2=1-p1;          # probability of group 2
x1=rnorm(n,0,1);
x2=rnorm(n,0,1);
e1n=rnorm(n,0,1);  # 1: normal,
e2n=rnorm(n,0,1);
e1L=rexp(n,sqrt(2))-rexp(n,sqrt(2)); # 2: laplace,
e2L=rexp(n,sqrt(2))-rexp(n,sqrt(2));
e1t1=rt(n,1);   # 3: t(1),
e2t1=rt(n,1);
e1t3=rt(n,3);   # 4: t(3),
e2t3=rt(n,3);
u=runif(n,0,1)
e1mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)  # 5: normal_mixture,
e2mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)
e1=e1n*((dist==1)|(dist==6))+e1L*(dist==2)+e1t1*(dist==3)
      +e1t3*(dist==4)+e1mn*(dist==5)
e2=e2n*((dist==1)|(dist==6))+e2L*(dist==2)+e2t1*(dist==3)
     +e2t3*(dist==4)+e2mn*(dist==5)
y1=0+x1+x2+e1;
y2=0-x1-x2+e2;
x=cbind(x1,x2);
y=y1*p1+y2*p2;
if(dist==6)   # 6: normal+0.05outlier
  {
    nout=round(n*0.95)+1;
    x[nout:n,]=20;
    y[nout:n]=100;
  }
  # Initial Values
prob=c(0.5,0.5);
b1=c(0.1,0.9,0.9);
b2=c(-0.1,-0.9,-0.9);
beta=rbind(b1,b2);
clust=kmeans(y,2)
sig01=sd(y[clust$cluster==1])
sig02=sd(y[clust$cluster==2])
if(is.na(sig01)){sig01=0};
if(is.na(sig02)){sig02=0};
if((sig01==0)|(sig02=0)){sig0=max(sig01,sig02)} else
  {sigma=(sig01+sig02)/2};
group=2;
```

```
X=cbind(rep(1,n),x);
run=1;
gsumi=matrix(0,n,group);
for(i in seq(group))
  {
    gsumi[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
  }
prest=sum(log(apply(gsumi,1,sum)));
repeat
  {
    tji=dji=adji=matrix(rep(0,group*n),nrow=n);
    for(i in seq(group))
      {
        tji[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
        adji[,i]=abs(y-X%*%beta[i,]);
        dji[,i]=sigma/(sqrt(2)*adji[,i]);
      }
    tji=ginv(diag(apply(tji,1,sum)))%*%tji;
    dji[dji==Inf]=10^6;
    wji=tji*dji;
    prob=apply(tji,2,mean);
    sse=rep(0,group);
    for(i in seq(group))
      {
        beta[i,]=lm(y~x1+x2,weights=wji[,i])$coefficients;
        if(any(is.na(beta[i,])))
          {
            W=diag(wji[,i]);
            beta[i,]=ginv(t(X)%*%W%*%X)%*%t(X)%*%W%*%y
          }
        sse[i]=sum(wji[,i]*(y-X%*%beta[i,])^2);
      }
    sigma2=2*sum(sse)/n;
    sigma=sqrt(sigma2);
    for(i in seq(group))
      {
        gsumi[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
      }
    latest=sum(log(apply(gsumi,1,sum)));
    dif=latest-prest;
    if((latest==-Inf)|(prest==-Inf)){break}
    if((dif<=1e-6)|(run>500)){break}
    prest=latest;
    run=run+1;
  }
```

```
            if((latest!=-Inf)&(prest!=-Inf)){break}
           }
          btemp=c(prob[1],beta[1,],prob[2],beta[2,]);
          btemp=slab(btemp);
          prob=c(btemp[1],btemp[5]);
          beta=rbind(btemp[2:4],btemp[6:8]);
          rest7=cbind(beta,sigma2,prob)
          b107[k]=rest7[1,1]
          b117[k]=rest7[1,2]
          b127[k]=rest7[1,3]
          b207[k]=rest7[2,1]
          b217[k]=rest7[2,2]
          b227[k]=rest7[2,3]
          pi17[k]=rest7[1,5]
      }
       tb10=0;  tb11=1;   tb12=1;
       tb20=0;  tb21=-1;   tb22=-1;
       tpi1=0.25;
       # MixregL
       RL=cbind(b107-tb10,b117-tb11,b127-tb12,b207-tb20,b217-tb21,
              b227-tb22,pi17-tpi1);
       BV[seq((dist-1)*7+1,dist*7),1:2]=cbind(apply(RL,2,function(x)
             {mean(x^2)}),apply(RL,2,mean))
       print(BV[seq((dist-1)*7+1,dist*7),1:2])
    }
    round(BV,3)
```

11. Robust Mixture Regression By Laplace Distribution

```
#               EM Algorithm
#               MixregL-MCD
  library(MASS)
  library(rrcov)
  library(robustbase)
  denLp=function(y,mu,sigma)
    {
      exp(-abs(sqrt(2)*(y-mu))/sigma)/(sqrt(2)*sigma)
    }
  BV=matrix(0,nrow=42,ncol=2)
  n=400    # Sample Size
  total=200
  for(dist in seq(6))
    {
      set.seed(98765)
```

```
b107=b117=b127=b207=b217=b227=pi17=rep(0,total)
for(k in seq(total))
  {
    repeat
    {
    n=400;
    u=runif(n,0,1);   # A random number for assigning groups
    p1=(u<=0.25);     # probability of group 1
    p2=1-p1;          # probability of group 2
    x1=rnorm(n,0,1);
    x2=rnorm(n,0,1);
    e1n=rnorm(n,0,1);  # 1: normal,
    e2n=rnorm(n,0,1);
    e1L=rexp(n,sqrt(2))-rexp(n,sqrt(2)); # 2: laplace,
    e2L=rexp(n,sqrt(2))-rexp(n,sqrt(2));
    e1t1=rt(n,1);   # 3: t(1),
    e2t1=rt(n,1);
    e1t3=rt(n,3);   # 4: t(3),
    e2t3=rt(n,3);
    u=runif(n,0,1)
    e1mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)  # 5: normal_mixture,
    e2mn=(u<0.95)*rnorm(n,0,1)+(u>=0.95)*rnorm(n,0,5)
    e1=e1n*((dist==1)|(dist==6))+e1L*(dist==2)+e1t1*(dist==3)
          +e1t3*(dist==4)+e1mn*(dist==5)
    e2=e2n*((dist==1)|(dist==6))+e2L*(dist==2)+e2t1*(dist==3)
         +e2t3*(dist==4)+e2mn*(dist==5)
    y1=0+x1+x2+e1;
    y2=0-x1-x2+e2;
    x=cbind(x1,x2);
    y=y1*p1+y2*p2;
    if(dist==6)   # 6: normal+0.05outlier
      {
        nout=round(n*0.95)+1;
        x[nout:n,]=20;
        y[nout:n]=100;
      }
      # Initial Values
    prob=c(0.5,0.5);
    b1=c(0.1,0.9,0.9);
    b2=c(-0.1,-0.9,-0.9);
    beta=rbind(b1,b2);
    clust=kmeans(y,2)
    sig01=sd(y[clust$cluster==1])
    sig02=sd(y[clust$cluster==2])
    if(is.na(sig01)){sig01=0};
```

```
if(is.na(sig02)){sig02=0};
if((sig01==0)|(sig02=0)){sig0=max(sig01,sig02)} else
  {sigma=(sig01+sig02)/2};
group=2;
mv=lev.p(x,method="mcd");
TX=cbind(x[,1]-mv$mean[1],x[,2]-mv$mean[2]);
ind=which(diag(TX%*%solve(mv$covar)%*%t(TX))>=qchisq(0.975,1));
if(length(ind)>0)
 {
  x=x[-ind,];
  y=y[-ind];
 }
x1=x[,1];
x2=x[,2];
n=length(y);
X=cbind(rep(1,n),x);
run=1;
gsumi=matrix(0,n,group);
for(i in seq(group))
  {
    gsumi[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
  }
prest=sum(log(apply(gsumi,1,sum)));
repeat
  {
    tji=dji=adji=matrix(rep(0,group*n),nrow=n);
    for(i in seq(group))
      {
        tji[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
        adji[,i]=abs(y-X%*%beta[i,]);
        dji[,i]=sigma/(sqrt(2)*adji[,i]);
      }
    tji=ginv(diag(apply(tji,1,sum)))%*%tji;
    dji[dji==Inf]=10^6;
    wji=tji*dji;
    prob=apply(tji,2,mean);
    sse=rep(0,group);
    for(i in seq(group))
      {
        beta[i,]=lm(y~x1+x2,weights=wji[,i])$coefficients;
        if(any(is.na(beta[i,])))
          {
           W=diag(wji[,i]);
           beta[i,]=ginv(t(X)%*%W%*%X)%*%t(X)%*%W%*%y
          }
```

```
              sse[i]=sum(wji[,i]*(y-X%*%beta[i,])^2);
            }
          sigma2=2*sum(sse)/n;
          sigma=sqrt(sigma2);
          for(i in seq(group))
            {
              gsumi[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
            }
          latest=sum(log(apply(gsumi,1,sum)));
          dif=latest-prest;
          if((latest==-Inf)|(prest==-Inf)){break}
          if((dif<=1e-6)|(run>500)){break}
          prest=latest;
          run=run+1;
        }
        if((latest!=-Inf)&(prest!=-Inf)){break}
      }
      btemp=c(prob[1],beta[1,],prob[2],beta[2,]);
      btemp=slab(btemp);
      prob=c(btemp[1],btemp[5]);
      beta=rbind(btemp[2:4],btemp[6:8]);
      rest7=cbind(beta,sigma2,prob)
      b107[k]=rest7[1,1]
      b117[k]=rest7[1,2]
      b127[k]=rest7[1,3]
      b207[k]=rest7[2,1]
      b217[k]=rest7[2,2]
      b227[k]=rest7[2,3]
      pi17[k]=rest7[1,5]
  }
   tb10=0;  tb11=1;   tb12=1;
   tb20=0;  tb21=-1;   tb22=-1;
   tpi1=0.25;
   # MixregL-MCD
   RL=cbind(b107-tb10,b117-tb11,b127-tb12,b207-tb20,b217-tb21,
                  b227-tb22,pi17-tpi1);
   BV[seq((dist-1)*7+1,dist*7),1:2]=cbind(apply(RL,2,function(x)
              {mean(x^2)}),apply(RL,2,mean))
   print(BV[seq((dist-1)*7+1,dist*7),1:2])
}
round(BV,3)
```

12. Real data example "datatone"

```
   library(MASS)
   library(fpc);
# Robust Mixture Regression By Laplace Distribution
  #  Bootstrap MSE: Laplace
     library(MASS)
     library(fpc);
     data(tonedata)
          denLp=function(y,mu,sigma)
            {
              exp(-abs(sqrt(2)*(y-mu))/sigma)/(sqrt(2)*sigma)
            }
           xx=tonedata$stretchratio
           x=c(xx,rep(3,5)) #10 pairs (0,4)
           yy=tonedata$tuned
           y=c(yy,rep(4,5)) #10 pairs (0,4)
           mcd=lev.p(as.matrix(x),method="classic")
           sres=(x-mcd$mean)/sqrt(mcd$covar);
           #x=x[abs(sres)<1.96] #remove outliers
           #y=y[abs(sres)<1.96] #remove outliers
           n=length(y)
           beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
           prob=c(0.5,0.5);
           sigma=c(0.2797);
           group=2
           X=cbind(rep(1,n),x);
           run=1;
           gsumi=matrix(0,n,group);
           for(i in seq(group))
             {
               gsumi[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
             }
           prest=sum(log(apply(gsumi,1,sum)));
           repeat
             {
               tji=dji=adji=matrix(rep(0,group*n),nrow=n);
               for(i in seq(group))
                 {
                   tji[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
                   adji[,i]=abs(y-X%*%beta[i,]);
                   dji[,i]=sigma/(sqrt(2)*adji[,i]);
                 }
               tji=ginv(diag(apply(tji,1,sum)))%*%tji;
               dji[dji==Inf]=10^6;
               wji=tji*dji;
               prob=apply(tji,2,mean);
```

```
            sse=rep(0,group);
            for(i in seq(group))
              {
                beta[i,]=lm(y~x,weights=wji[,i])$coefficients;
                if(any(is.na(beta[i,])))
                  {
                    W=diag(wji[,i]);
                    beta[i,]=ginv(t(X)%*%W%*%X)%*%t(X)%*%W%*%y
                  }
                sse[i]=sum(wji[,i]*(y-X%*%beta[i,])^2);
              }
            sigma2=2*sum(sse)/n;
            sigma=sqrt(sigma2);
            for(i in seq(group))
              {
                gsumi[,i]=prob[i]*denLp(y,X%*%beta[i,],sigma);
              }
            latest=sum(log(apply(gsumi,1,sum)));
            dif=latest-prest;
            if((dif<=1e-6)|(run>500)){break}
            prest=latest;
            run=run+1;
          }
       rest=cbind(beta,sigma2,prob)
       a1=rest[1,1]
       b1=rest[1,2]
       a2=rest[2,1]
       b2=rest[2,2]
       sse=mean(pmin((y-a1-b1*x)^2,(y-a2-b2*x)^2))
      c(a1,b1,a2,b2,sse)
```

13. Robust Mixture Regression By Laplace Distribution

```
 #  Bootstrap MSE:  T
 # Definition of t density
    dent=function(y,mu,sig,v)
      {
       est=gamma((v+1)/2)*sig^(-1)/((pi*v)^(1/2)*gamma(v/2)*
           (1+(y-mu)^2/(sig^2*v))^(0.5*(v+1)));
       est
      }
           x=tonedata$stretchratio
           x=c(x,rep(0,10))
           y=tonedata$tuned
```

```
y=c(y,rep(4,10))
 mcd=lev.p(as.matrix(x),method="classic")
sres=(x-mcd$mean)/sqrt(mcd$covar);
#x=x[abs(sres)<1.96] # remove outliers
#y=y[abs(sres)<1.96] # remove outliers
n=length(y)
beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
prob=c(0.5,0.5);
sig0=c(0.2797);
group=2
X=cbind(rep(1,n),x);
m=2;
maxv=15;
likhod=flag=rep(0,15);
theta=array(0,dim=c(2,4,15));
for(v in seq(maxv))
 {
   bet=beta;
   sig=sig0;
   pr=prob;
   run=0;
   n=length(y);
   X=cbind(rep(1,n),x);
   lh=-10^10;
   a=dim(x);
   p=a[2]+1;
   pk=u=r=matrix(rep(0,m*n),nrow=n);
   for(j in seq(m))
    {
      r[,j]=(y-X%*%bet[j,])/sig;
    }
  repeat
   {
    run=run+1;
    prelh=lh;
    for(j in seq(m))
      {
       pk[,j]=pr[j]*dent(y, X%*%bet[j,],sig,v);
       u[,j]=(v+1)/(v+r[,j]^2)
      }
    lh=sum(log(apply(pk,1,sum)));
    pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
    dif=lh-prelh;
    if(!is.finite(lh)){flag[v]=1; break}
    if(dif<1e-6|run>500){flag[v]=0; break}
```

```
     np=apply(pk,2,sum);pr=np/n;
     for(j in seq(m))
      {
       w=diag(pk[,j]*u[,j]);
       bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
       r[,j]=y-X%*%bet[j,];
      }
    sig=sqrt(sum(pk*r^2*u)/sum(pk));
    r=r/sig;
   }
 sig=sig*rep(1,m);
 theta[,,v]= matrix(c(bet,pr,sig),nrow=m);
 likhod[v]=lh;
}
if(all(flag==0)){break}
}
fv=which(likhod==max(likhod))
if(fv<15)
 {result=theta[,,fv]} else
   {
    run=0;
    n=length(y);
    X=cbind(rep(1,n),x);
    r=matrix(rep(0,m*n),nrow=n);
    pk=r;
    lh=0
    for(j in seq(m))
     {
      r[,j]=y-X%*%bet[j,];
      lh=lh+pr[j]*dnorm(r[,j],0,sig);
     }
    lh=sum(log(lh));
    repeat
      {
       prest=c(bet,sig,pr);
       run=run+1;
       plh=lh;
       for(j in seq(m))
         {
          pk[,j]=pr[j]* pmax(10^(-6),dnorm(r[,j],0,sig))
         }
       pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
       np=apply(pk,2,sum);
       pr=np/n;
       for(j in seq(m))
```

```
                {
                  w=diag(pk[,j]);
                  bet[j,]=ginv(t(X)%*%w%*%X)%*%t(X)%*%w%*%y;
                  r[,j]= y-X%*%bet[j,];
                 }
               sig=sqrt(sum(pk*(r^2))/n);
               lh=0;
               for(j in seq(m))
                  {
                   lh=lh+pr[j]*dnorm(r[,j],0,sig);
                  }
               lh=sum(log(lh));
               dif=lh-plh;
               if((dif<1e-6)|(run>500)){break}
              }
            sig=sig*rep(1,m)
            result=matrix(c(bet,pr,sig),nrow=m)
           }
        a1=bet[1,1]
        b1=bet[1,2]
        a2=bet[2,1]
        b2=bet[2,2]
        sse=mean(pmin((y-a1-b1*x)^2,(y-a2-b2*x)^2))
         c(a1,b1,a2,b2,sse)
        plot(x,y,xlab="tuned",ylab="strechratio")
        x=seq(0,3,by=0.1)
        yfit1=a1+b1*x;
        yfit2=a2+b2*x;
        lines(x,yfit1)
        lines(x,yfit2)
          c(a1,b1,a2,b2,sse)
```

14. Robust Mixture Regression By Laplace Distribution

```
 #  Bootstrap MSE: Bisquare
  bisquare=function(t,k=4.685)
     {
       out=t*pmax(0,(1-(t/k)^2))^2;
       out
     }
  biscalew=function(t)
     {
       t[which(t==0)]=min(t[which(t!=0)])/10;
       out=pmin(1-(1-t^2/1.56^2)^3,1)/t^2;
```

```
     out
   }
mixbi2<-function(x,y,bet,sig,pr,m=2)
  {
    run=0;
    acc=10^(-4)*max(abs(c(bet,sig,pr)));
    n=length(y);
    X=cbind(rep(1,n),x);
    p=dim(X)[2];
    if(length(sig)>1)
      {
        r=matrix(rep(0,m*n),nrow=n);
        pk=r;
        for(j in seq(m))
         {
          r[,j]=(y-X%*%bet[j,])/sig[j];
         }
    #E-steps
     repeat
      {
        prest=c(sig,bet,pr);run=run+1;
        for(j in seq(m))
         {
           pk[,j]=pr[j]*pmax(10^(-6),dnorm(r[,j],0,1))/sig[j]
         }
        pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
        #M-step
        np=apply(pk,2,sum);pr=np/n;
        r[which(r==0)]=min(r[which(r!=0)])/10;
        for(j in seq(m))
         {
          w=diag(pk[,j]*bisquare(r[,j])/r[,j]);
          bet[j,]= solve(t(X)%*%w%*%X+10^(-3)*diag(rep(1,p)))%*%t(X)%*%w%*%y;
          r[,j]= (y-X%*%bet[j,])/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,bet,pr)-prest))
        if(dif<acc|run>500){break}
      }
   }
else
  {
    r=matrix(rep(0,m*n),nrow=n);
    pk=r;
    for(j in seq(m))
```

```
      {
       r[,j]=(y-X%*%bet[j,])/sig;
      }
    #E-steps
    repeat
     {
       prest=c(sig,bet,pr);
       run=run+1;
       for(j in seq(m))
        {
          pk[,j]=pr[j]*pmax(10^(-6),dnorm(r[,j],0,1))/sig
        }
       pk=pk/matrix(rep(apply(pk,1,sum),m),nrow=n);
   #M-step
      np=apply(pk,2,sum);
      pr=np/n;
      r[which(r==0)]=min(r[which(r!=0)])/10;
      for(j in seq(m))
       {
        w=diag(pk[,j]*bisquare(r[,j])/r[,j]);
        bet[j,]=solve(t(X)%*%w%*%X+10^(-3)*diag(rep(1,p)))%*%t(X)%*%w%*%y;
        r[,j]=(y-X%*%bet[j,])/sig;
       }
      sig=sqrt(sum(pk*(r^2*sig[1]^2)*biscalew(r))/n/0.5)
      dif=max(abs(c(sig,bet,pr)-prest))
      if(dif<acc|run>500){break}
     }
    sig=rep(sig,m);
  }
 bet
}
  library(MASS)
  library(fpc);
  x=tonedata$stretchratio
  x=c(x,rep(0,10))
  y=tonedata$tuned
  y=c(y,rep(4,10))
  n=length(y)
        beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
        prob=c(0.5,0.5);
        sigma=c(0.2797);
        group=2
       rest=mixbi2(x1,y1,beta,sigma,prob,m=2)
        bet=c(rest[1,],rest[2,])
     bet0=c(0.1059,0.99485,1.9573,0.0276)
```

```
        ind1=c(1,2,3,4)
        ind2=c(3,4,1,2)
        res1=sum((bet[ind1]-bet0)^2)
        res2=sum((bet[ind2]-bet0)^2)
        theta=bet[ind1]*(res1<=res2)+bet[ind2]*(res1>res2)
         a1=theta[1]
         b1=theta[2]
         a2=theta[3]
         b2=theta[4]
   sse=mean(pmin((y-a1-b1*x)^2,(y-a2-b2*x)^2))
             c(a1,b1,a2,b2,sse)
   plot(x,y,xlab="tuned",ylab="strechratio")
   x=seq(0,3,by=0.1)
   yfit1=a1+b1*x;
   yfit2=a2+b2*x;
   lines(x,yfit1)
   lines(x,yfit2)
  c(a1,b1,a2,b2,sse)
```

15. Robust Mixture Regression By Laplace Distribution

```
 #  Bootstrap MSE: TLE
  trimmix=function(x,y,k=2,alpha=0.9,bet,sig,pr,acc=1e-6)
    {
      n=length(y);
      n1=round(n*alpha);
      x=matrix(x,nrow=n);
      a=dim(x);
      p=a[2]+1;
      X=cbind(rep(1,n),x);
      if(dim(bet)[2]==k){bet=t(bet)};
      lh=0
      for (i in seq(k))
        {
         lh=lh+pr[i]*dnorm(y-X%*%bet[i,],0,sig[1])
        }
      ind=order(-lh);run=0;
      obj=sum(log(lh[ind[1:n1]]));
      repeat
        {
         pobj=obj;
         run=run+1;
         x1=x[ind[1:n1],];
         y1=y[ind[1:n1]];
```

```
      fit=mixlin(x1,y1,bet,sig,pr,k);
      fit=fit$theta;
      bet=matrix(fit[1:(p*k)],nrow=k);
      sig=fit[p*k+1];
      pr=fit[(p*k+k+1):(p*k+2*k)];
      lh=0;
      for(i in seq(k))
        {
         lh=lh+pr[i]*dnorm(y-X%*%bet[i,],0,sig[1]);
        }
      ind=order(-lh);
      obj=sum(log(lh[ind[1:n1]]));dif=obj-pobj;
      if(dif<acc|run>50){break}
     }
   if(length(sig)<2){sig=rep(sig,k)};
   bet
 }
library(MASS)
library(fpc);
  x=tonedata$stretchratio
  x1=c(x,rep(3,5)) # 10 pair (0,4)
  y=tonedata$tuned
  y1=c(y,rep(4,5)) # 10 pair (0,4)
   mcd=lev.p(as.matrix(x),method="classic")
        sres=(x-mcd$mean)/sqrt(mcd$covar);
        x=x[abs(sres)<1.96]
        y=y[abs(sres)<1.96]
  n=length(y)
        beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
        prob=c(0.5,0.5);
        sigma=c(0.2797);
        group=2
     rest=trimmix(x1,y1,k=2,alpha=0.9,beta,sigma,prob,acc=1e-6)
     bet=c(rest[1,],rest[2,])
     bet0=c(0.1059,0.99485,1.9573,0.0276)
     ind1=c(1,2,3,4)
     ind2=c(3,4,1,2)
     res1=sum((bet[ind1]-bet0)^2)
     res2=sum((bet[ind2]-bet0)^2)
     theta=bet[ind1]*(res1<=res2)+bet[ind2]*(res1>res2)
     a1=theta[1]
      b1=theta[2]
      a2=theta[3]
      b2=theta[4]
      sse=mean(pmin((y-a1-b1*x)^2,(y-a2-b2*x)^2))
```

```
 c(a1,b1,a2,b2,sse)
plot(x,y,xlab="tuned",ylab="strechratio")
x=seq(0,3,by=0.1)
yfit1=a1+b1*x;
yfit2=a2+b2*x;
lines(x,yfit1)
lines(x,yfit2)
 c(a1,b1,a2,b2,sse)
```

16. Robust Mixture Regression By Laplace Distribution

```
#   Bootstrap MSE:  MLE
 mix.MLE.normal=function(x,y,beta,sigma,prob,group,tol=1e-6)
   {
     n=length(y);
     X=cbind(rep(1,n),x);
     run=1;
     if(length(sigma)>1)
      {
        gsumi=matrix(0,n,group);
        for(i in seq(group))
          {
           gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma[i]);
          }
        prest=sum(log(apply(gsumi,1,sum)));
       repeat
        {
          tji=matrix(0,n,group)
          for(i in seq(group))
               {
                tji[,i]=prob[i]*pmax(dnorm(y,X%*%beta[i,],sigma[i]),10^(-6));
               }
          tji=solve(diag(apply(tji,1,sum)))%*%tji;
          prob=apply(tji,2,mean);
          sse=rep(0,group)
          for(i in seq(group))
           {
             beta[i,]=lm(y~x,weight=tji[,i])$coefficient;
             sse[i]=sum(tji[,i]*(y-X%*%beta[i,])^2);
             sigma[i]=sqrt(sse[i]/sum(tji[,i]))
           }
         sigma2=sigma^2;
         for(i in seq(group))
          {
```

```
      gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma[i]);
     }
    latest=sum(log(apply(gsumi,1,sum)));
    dif=abs(latest-prest);
    prest=latest;
    run=run+1;
    if(dif<=tol|run>500){break}
  }
 }
else
  {
   gsumi=matrix(0,n,group);
   for(i in seq(group))
     {
      gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma);
     }
   prest=sum(log(apply(gsumi,1,sum)));
  repeat
   {
     tji=matrix(0,n,group)

    for(i in seq(group))
       {
         tji[,i]=prob[i]*pmax(dnorm(y,X%*%beta[i,],sigma),10^(-6));
       }
    tji=solve(diag(apply(tji,1,sum)))%*%tji;
     prob=apply(tji,2,mean);
     sse=rep(0,group);
     for(i in seq(group))
      {
        beta[i,]=lm(y~x,weight=tji[,i])$coefficient;
        sse[i]=sum(tji[,i]*(y-X%*%beta[i,])^2);
      }
    sigma2=sum(sse)/n;
    sigma=sqrt(sigma2);
    for(i in seq(group))
     {
      gsumi[,i]=prob[i]*dnorm(y,X%*%beta[i,],sigma);
     }
    latest=sum(log(apply(gsumi,1,sum)));
    dif=abs(latest-prest);
    prest=latest;
    run=run+1;
    if(dif<=tol|run>500){break}
  }
```

```
    }
  beta
   }
```

17. Plot Fitting of Different Methods with Different Type of Outliers

```
    library(MASS)
    library(fpc);
    x=tonedata$stretchratio
    x=c(x,rep(3,5))
    y=tonedata$tuned
    y=c(y,rep(4.5,5))
    n=length(y)
          beta=matrix(c(-0.5,2,0.89,0.01),nrow=2);
          prob=c(0.5,0.5);
          sigma=c(0.2797);
          group=2
       rest=mix.MLE.normal(x,y,beta,sigma,prob,group,tol=1e-6)
      bet=c(rest[1,],rest[2,])
      bet0=c(0.1059,0.99485,1.9573,0.0276)
      ind1=c(1,2,3,4)
      ind2=c(3,4,1,2)
      res1=sum((bet[ind1]-bet0)^2)
      res2=sum((bet[ind2]-bet0)^2)
      theta=bet[ind1]*(res1<=res2)+bet[ind2]*(res1>res2)
       a1=theta[1]
       b1=theta[2]
       a2=theta[3]
       b2=theta[4]
       sse=mean(pmin((y-a1-b1*x)^2,(y-a2-b2*x)^2))
       c(a1,b1,a2,b2,sse)
       plot(x,y,xlab="tuned",ylab="strechratio")
       x=seq(0,3,by=0.1)
       yfit1=a1+b1*x;
       yfit2=a2+b2*x;
       lines(x,yfit1)
       lines(x,yfit2)
       c(a1,b1,a2,b2,sse)
```

17.1. With 5 pair of outliers (3,4)

```
  library(fpc);
  data(tonedata)
  par(mfrow=c(1,2),pty="s")
```

```
    xx=tonedata$stretchratio
    x=c(xx,rep(3,5))
    yy=tonedata$tuned
    y=c(yy,rep(4,5))
   plot(xx,yy,xlab="tuned",ylab="strech ratio",xlim=c(1.3,3),ylim=c(1.3,4))
   points(3,4,pch=8)
   x=seq(0,3,by=0.1)
# Laplace
   a1=0.0009999494
   b1=1.0000000167
   a2=1.9498823529
   b2=0.0294117647
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# T
   a1=-0.18739840
   b1=1.08683202
   a2=1.88872212
   b2=0.05823153
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
# Bisquare
   a1=0.01670549
   b1=0.9928898
   a2=1.964869
   b2=0.02485191
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
    x=tonedata$stretchratio
    x=c(x,rep(3,5))
    y=tonedata$tuned
    y=c(y,rep(4.5,5))
   plot(xx,yy,xlab="tuned",ylab="",xlim=c(1.3,3),ylim=c(1.3,4.5),yaxt='n')
   axis(2,labels=FALSE)
   points(3,4.5,pch=8)
   x=seq(0,3,by=0.1)
# Bisquare
   a1=0.01670549
   b1=0.9928898
```

```
   a2=1.964869
   b2=0.02485191
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
# TLE
   a1=-1.025159
   b1=1.67224
   a2=1.842036
   b2=0.08432628
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# MLE
   a1=-0.7384855
   b1=1.36924
   a2=1.815414
   b2=0.0982063
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
```

17.2. With 5 pair of outliers (3,4.5)

```
   library(fpc);
   par(mfrow=c(1,2),pty="s")
    xx=tonedata$stretchratio
    x=c(xx,rep(3,5))
    yy=tonedata$tuned
    y=c(yy,rep(4.5,5))
   plot(xx,yy,xlab="tuned",ylab="strech ratio",xlim=c(1.3,3),ylim=c(1.3,4.5))
   points(3,4.5,pch=8)
   x=seq(0,3,by=0.1)
# Laplace
   a1=0.0009999952
   b1=1.0000000016
   a2=1.9498822862
   b2=0.0294117931
   sseL=0.0785645867
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
```

```
   lines(x,fit2,lty=2,lwd=1)
# T
   a1=-0.13638402
   b1=1.06087869
   a2=1.89036535
   b2=0.05730300
   sseT=0.07426869
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
# Bisquare
   a1=0.01817967
   b1=0.99160576
   a2=1.96748687
   b2=0.02305783
   sseB=0.07934474
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
    x=tonedata$stretchratio
    x=c(x,rep(3,5))
    y=tonedata$tuned
    y=c(y,rep(4.5,5))
   plot(xx,yy,xlab="tuned",ylab="",xlim=c(1.3,3),ylim=c(1.3,4.5),yaxt='n')
   axis(2,labels=FALSE)
   points(3,4.5,pch=8)
   x=seq(0,3,by=0.1)
# Bisquare
   a1=0.01817967
   b1=0.99160576
   a2=1.96748687
   b2=0.02305783
   sseB=0.07934474
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
# TLE
   a1=-1.48677313
   b1=1.99274099
   a2=1.84191159
   b2=0.08437009
   sseTLE=0.05288226
```

```
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# MLE
   a1=-2.25079063
   b1=2.18701617
   a2=1.39486335
   b2=0.31127507
   sseTLE=0.04367094
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
```

17.3. With 10 pair of outliers (0,3)

```
   library(fpc);
   par(mfrow=c(1,2),pty="s")
    xx=tonedata$stretchratio
    x=c(xx,rep(0,10))
    yy=tonedata$tuned
    y=c(yy,rep(3,10))
   plot(xx,yy,xlab="tuned",ylab="strech ratio",xlim=c(0,3),ylim=c(1.3,3))
   points(0,3,pch=8)
   x=seq(0,3,by=0.1)
# Laplace
   a1=0.003222222
   b1=0.998888889
   a2= 2.037608221
   b2=-0.008695325
   sseL=0.064030873
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# T
   a1=0.06816041
   b1=0.95571454
   a2=2.62884353
   b2=-0.27165519
   sseT=0.02602522
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
```

```
   lines(x,fit2,lty=3,lwd=1)
# Bisquare
   a1=0.01670549
   b1=0.99288980
   a2=1.96486856
   b2=0.02485191
   sseB=0.07287699
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
    x=tonedata$stretchratio
    x=c(x,rep(0,10))
    y=tonedata$tuned
    y=c(y,rep(3,10))
      plot(xx,yy,xlab="tuned",ylab="",xlim=c(0,3),ylim=c(1.3,3),yaxt='n')
   axis(2,labels=FALSE)
     points(0,3,pch=8)
   x=seq(0,3,by=0.1)
# Bisquare
   a1=0.01670549
   b1=0.99288980
   a2=1.96486856
   b2=0.02485191
   sseB=0.07287699
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
# TLE
   a1=0.070297991
   b1=0.963432814
   a2=1.925161627
   b2=0.038860682
   sseTLE=0.006120743
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# MLE
   a1=0.05153669
   b1=0.96245563
   a2=2.66721859
   b2=-0.28956024
   sseTLE=0.02573221
```

```
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
```

17.4. With 10 pair of outliers (0,4)

```
    library(fpc);
    par(mfrow=c(1,2),pty="s")
    xx=tonedata$stretchratio
    x=c(xx,rep(0,10))
    yy=tonedata$tuned
    y=c(yy,rep(4,10))
   plot(xx,yy,xlab="tuned",ylab="strech ratio",xlim=c(0,3),ylim=c(1.3,4))
   points(0,4,pch=8)
# Laplace
   a1=1.87983051
   b1=0.06440678
   a2=1.97800000
   b2=0.02000000
   sseL=0.32253498
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# T
   a1=1.4581761
   b1=0.2773582
   a2=3.9945974
   b2=-0.9747406
   sseT= 0.0496096
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
# Bisquare
   a1=0.02304719
   b1=0.98966712
   a2=1.96156984
   b2=0.02633991
   sseB=0.26559389
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
```

```
    x=tonedata$stretchratio
    x=c(x,rep(0,10))
    y=tonedata$tuned
    y=c(y,rep(4,10))
      plot(xx,yy,xlab="tuned",ylab="",xlim=c(0,3),ylim=c(1.3,4),yaxt='n')
   axis(2,labels=FALSE)
     points(0,4,pch=8)
   x=seq(0,3,by=0.1)
# Bisquare
   a1=0.02304719
   b1=0.98966712
   a2=1.96156984
   b2=0.02633991
   sseB=0.26559389
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=1,lwd=1)
   lines(x,fit2,lty=1,lwd=1)
# TLE
   a1=0.070297991
   b1=0.963432814
   a2=1.925161627
   b2=0.038860682
   sseTLE=0.006120743
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=2,lwd=1)
   lines(x,fit2,lty=2,lwd=1)
# MLE
   a1=1.25507678
   b1=0.38119961
   a2=3.98833381
   b2=-0.71758717
   sseTLE= 0.03874365
   fit1=a1+b1*x;
   fit2=a2+b2*x;
   lines(x,fit1,lty=3,lwd=1)
   lines(x,fit2,lty=3,lwd=1)
```