Mining the Latent Relationship among Variables using Mixture Model – Supplementary Materials

Wennan Chang Chi Zhang* Sha Cao*

S1. Introduction

Coming soon.

S2. Case Study

S2.0 Preparation: Install the RobMixReg Pacakge

The RobMixReg package publised on the CRAN. That means the package has better stability and easy to install on several platform (e.g. windows, linux) because CRAN has stricter requirement than github.

```
#install.packages('RobMixReg')
library(RobMixReg)
```

The RobMixReg package also has the most updated version on github.

```
#library("devtools")
#devtools::install_github("changwn/RobMixReg")
```

S2.1 Example 1: Low dimension, $K = 1$ with outlier

Linear regression is the classical model to explorer the relationship between two variables (e.g. gene and patient outcome). However, general linear model is very unstable when there are outliers caused by measurement error or patient internal heterogeneity. To deal with this problem, robust mixture regression became a common way during the statistic analysis (e,g. drug sensitivity prediction et al.). The example 1 simulate the situation which contains a single regression line and several outliers. The implemented algorithm is Least Trimmed Squares Robust Regression (see Table 1 in maintext).

- Mathmatical Model

$$\begin{cases} y_i - \beta_1 x_i - \beta_0 - \epsilon \leq \eta & \text{if} \quad (x_i, y_i) \in \textit{regression line} \\ y_i - \beta_1 x_i - \beta_0 - \epsilon > \eta & \text{if} \quad (x_i, y_i) \textit{ is outlier} \end{cases}$$

Data Simulation:

In this section, we introduce the data generation process. The goal of simulation is to generate single regression line and several outliers. The independent variable *gene* is a vector and its length is 300. $gene \sim N(5, 1)$.The coefficient is controled by variable *slope* and intercept is controled by variable *inter*.

The generation of dependent variable consist of two steps. Firstly, generate a strict regression line based on formula $y_i = \beta x_i + \beta_0$. Then, the Gaussian white noise, $\epsilon \sim N(0, 0.1^2)$,added to the line. Thus, the least square regression line is $y_i = \beta x_i + \beta_0 + \epsilon$. In order to adding the outliers, we sample the index of all data point and select 10% percentage of all points as outliers. Huge variation was added to the line to mimic the outliers caused by measurement error et al. The huge variation from another normal distribution which is $\eta \sim N(0, 2^2)$.

```
# set the seed for randomization
set.seed(12345)
gene = rnorm(n=300, mean=5, sd=1)
inter = 1 ; slope = 0.8
outcome = slope * gene + inter
outcome = outcome + rnorm(n=length(outcome),mean=0, sd=0.3)
# add the outliers
# 10% of variable should be outliers, user can customize the percentage
id_outlier = sample(1:length(outcome), size = 0.1*length(outcome))
outcome[id_outlier] = outcome[id_outlier] + rnorm(length(id_outlier), mean=0,sd=2)
```

All methods mentioned in the paper were integrated in the package in a very user-freindly way. If the user want to perform the robust regression in example 1. The user can run the Least Trimmed Saured Robust Regression by just calling *MLM* main function and setting the *ml.method* parameter as 'a'.

```
res.a = MLM(ml.method='a', outcome, gene)
```

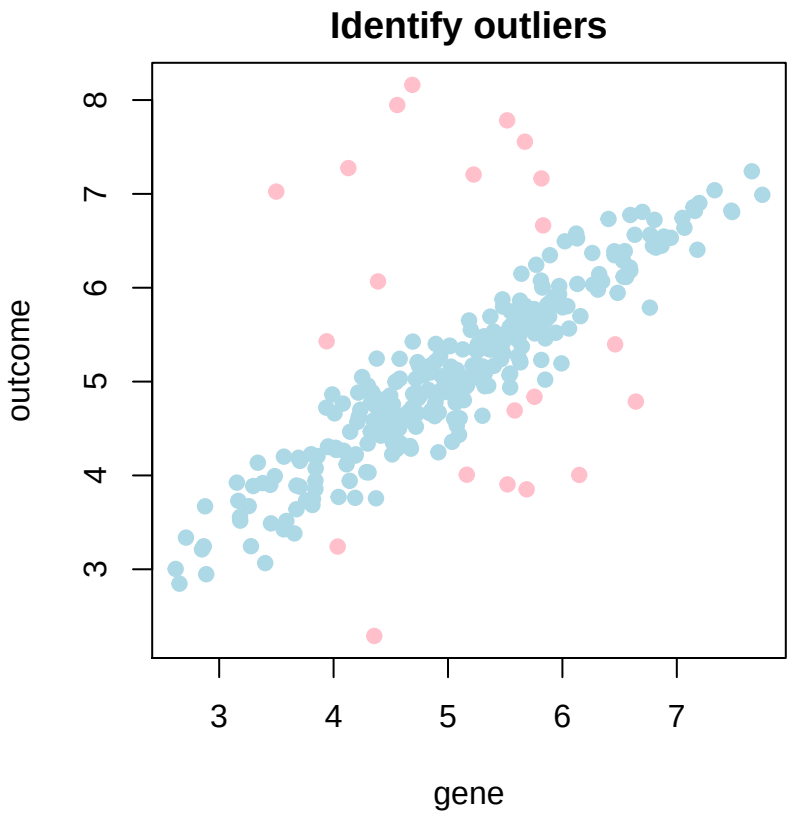
Once we get the result from the method 1, we can extract the sample information that which sample should be a outlier or not.

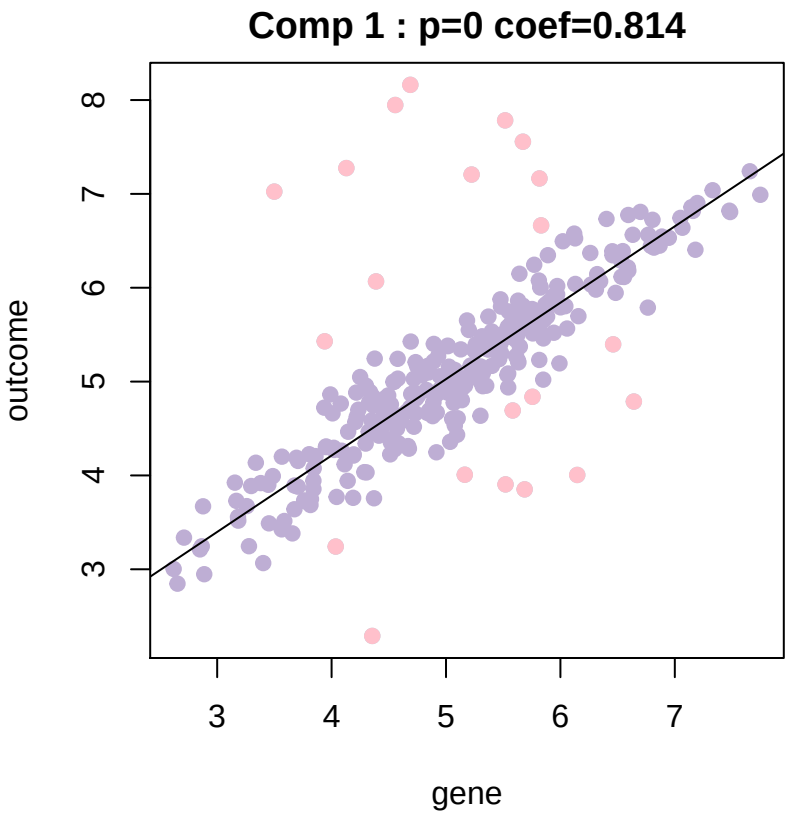
```
inds_out = which(res.a$lts.wt == 0)
#print the data id which is outlier.
print(inds_out)
```

```
##  [1]  14  16  23  43  46  49  69  91  96 101 131 144 198 205 215 226 276 278 282
## [20] 283 290
```

The RobMixReg package has a plotting module for visualization in a easy way. For the example 1, we draw all data point and color them based on whether it is a outlier. Then, we also print another figure which contains the fitted regression line and metric (coeffient and related significant p value).

```
inds_in = which(res.a$lts.wt == 1)
# set the figure format and margin
par(mfrow=c(1,1),mar = c(5, 8, 2, 8))
# use plot module to draw the data points and the regression line.
plot_CTLE(outcome~gene,data=data.frame(gene,outcome),nc=1,inds_in=inds_in)
```


Identify outliers
outcome
gene
3
4
5
6
7
8
3
4
5
6
7



S2.2 Example 2: Low dimension, $K = 2$ without outlier

The relationship between gene expression and patient outcome is complex. Besides the outlier problem, the heterogeneity among the subject is also a challenge. Due to subgroup patient may behave similar property. Mixture model (e.g. mixture regression line) were widely used in real application. However, to the best of our knowledge, there is no package provide a flexible mixture regression modeling expet the 'mixtureReg' (see maintext reference). The RobMixReg package modified and optimized the several functions in 'mixtureReg' which meet the publish requirement of CRAN.

- Mathmatical Model

$$y_i = \beta_k x_i + \beta_{k0} + \epsilon_{ki},$$

where

$$\epsilon_{ki} \sim N(0, \sigma_k^2), k = 1, 2; i = 1, ..., n.$$

Data Simulation:

To demonstrate flexible mixture regression model, we simulated two lines through below script. The user can change the two coefficient parameter by setting the *slope1* and *slope2*. Also, two intercepts controlled by *inter1* and *inter2*. The Gaussian white noisy added to two lines form a normal distribution as $\epsilon_{ki} \sim N(0, 0, 1^2)$. For giving a balanced number of mixture model, we selected odd position data points as the first regression line and even position data point as the second regression line.

```
gene = rnorm(n=300, mean=5, sd=1)
outcome = rep(0, length(gene))
slope1 = 0.8; inter1 = 0.2; slope2 = -0.01; inter2 = 4
n <- length(gene)
outcome[seq(n) %% 2 == 1] = slope1 * gene[seq(n) %% 2 == 1] + inter1
outcome[seq(n) %% 2 == 1] = outcome[seq(n) %% 2 == 1] + rnorm(length(outcome)/2,mean=0,sd=0.1)
outcome[seq(n) %% 2 == 0] = slope2 * gene[seq(n) %% 2 == 0] + inter2
outcome[seq(n) %% 2 == 0] = outcome[seq(n) %% 2 == 0] + rnorm(length(outcome)/2,mean=0,sd=0.1)
```

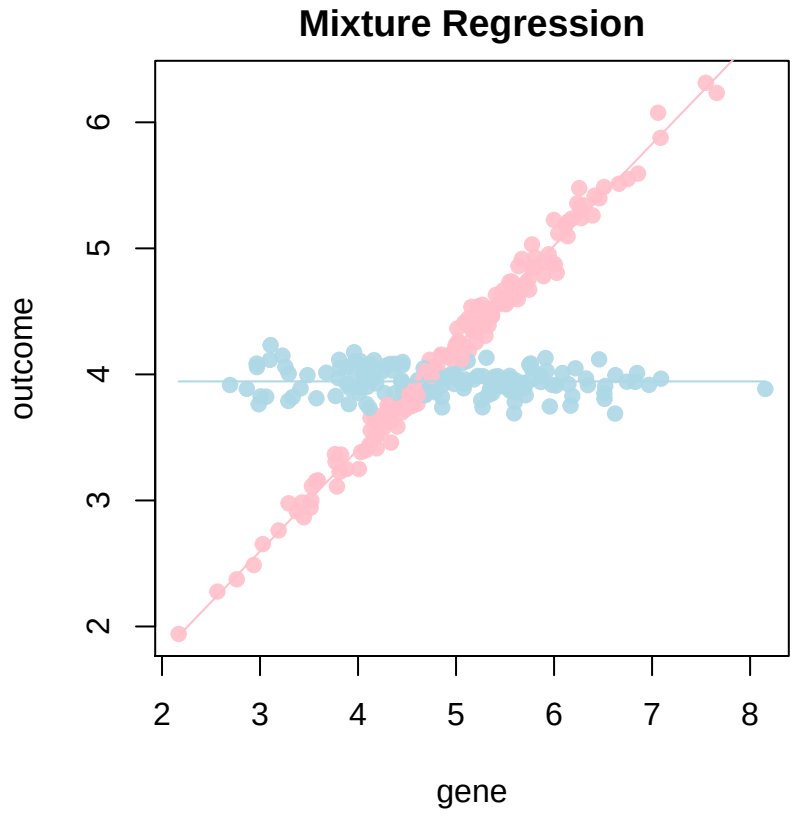
The package own the capability of fitting flexible mixture regression. User can customize the several regression format by setting the *b.formulaList* parameter.

```
res.b = MLM(ml.method='b', gene, outcome,
            b.formulaList = list(formula(outcome ~ gene),formula(outcome ~ 1)))
```

```
## diff =  -3.691596e-09
## iter =  14
## restart =  0
## log-likelihood =  90.17514
```

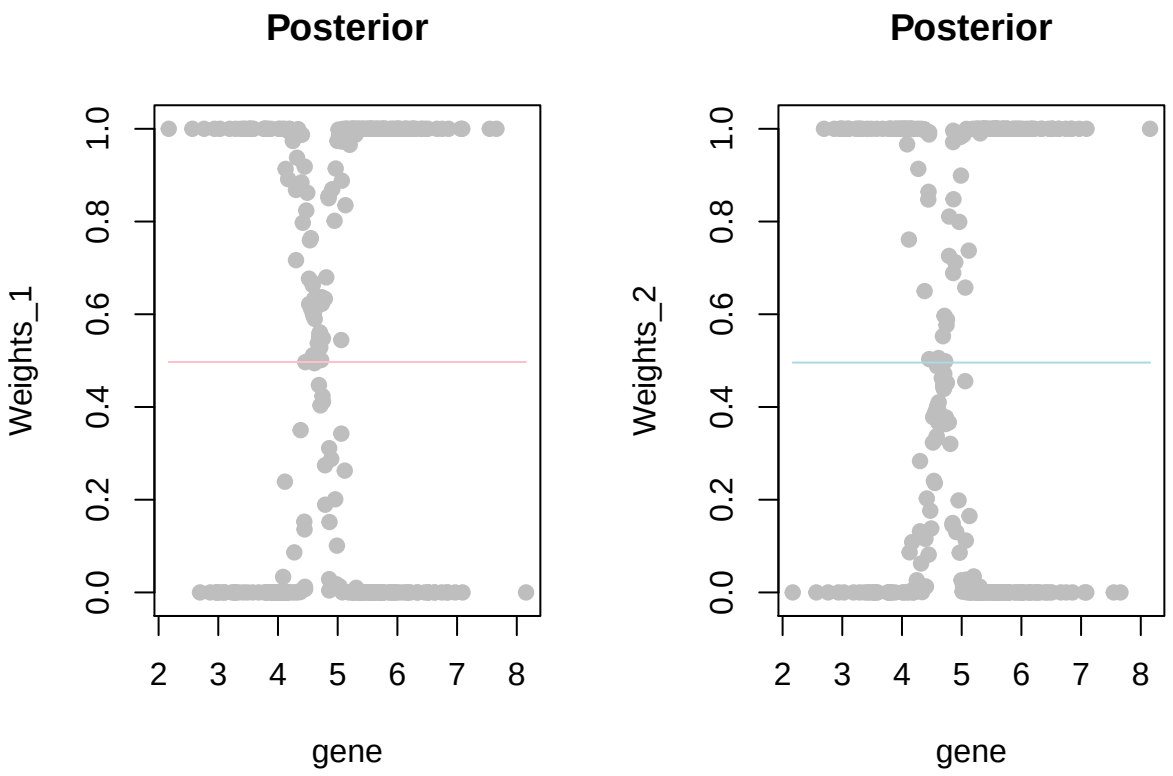
The plot function provide the two modules for the example 2. The first modele will draw two mixture regression lines with different color.

```
par(mar = c(5, 8, 2, 8))
plot_mixtureReg(res.b, which = 1)
```



The second module give the posterior for each data point. The pposterior decide which data point belong to which regression line in mixture model. In both figures, some data points around 5 have low prior because the intersection point is around 5 show as the last figure. It means it is difficult to cluster the point near the intersection point of two lines.

```
par(mfrow=c(1,2))
plot_mixtureReg(res.b, which = 2)
```



S2.3 Example 3: Low dimension, $K = 2$ with outlier

As described in last two examples, we explorer the outlier and heterogeneity among the patient. In this example 3, we will demonstrate using our in-house algorithm CAT to detect the outlier and estimate the parameter of mixture model simultaneously.

- Mathmatical Model

$$\begin{cases} y_i - \beta_k x_i - \beta_{k0} - \epsilon_{ki} \leq \eta & \text{if} \quad (x_i, y_i) \in \textit{regression line} \\ y_i - \beta_k x_i - \beta_{k0} - \epsilon_{ki} \leq \eta & \text{if} \quad (x_i, y_i) \textit{ is outlier} \end{cases}$$

Data Simulation:

The simulation of example 3 combines the process of example 1 and example 2. Two regression lines can be customized by setting the coefficient and intercept parameter. The outliers were selected randomly and the number of outlier is controlled by parameter *size*.

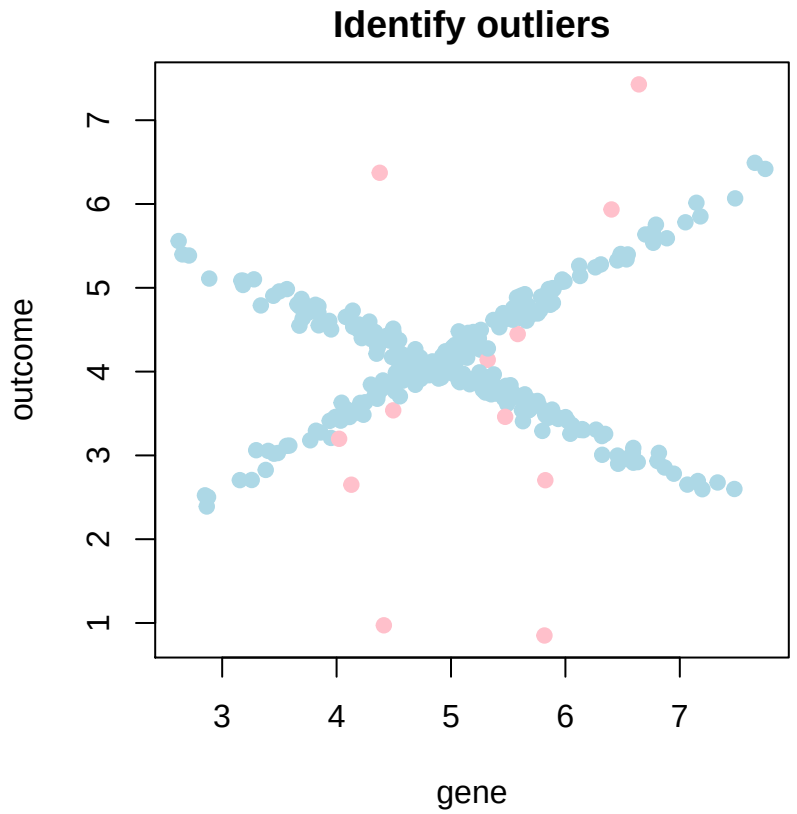
```r
set.seed(12345)
gene = rnorm(n=300, mean=5, sd=1)
outcome = rep(0, length(gene))
slope1 = 0.8; inter1 = 0.2; slope2 = -0.6; inter2 = 7
n <- length(gene)
outcome[seq(n) %% 2 == 1] = slope1 * gene[seq(n) %% 2 == 1] + inter1
outcome[seq(n) %% 2 == 1] = outcome[seq(n) %% 2 == 1] + rnorm(length(outcome)/2,mean=0,sd=0.1)
outcome[seq(n) %% 2 == 0] = slope2 * gene[seq(n) %% 2 == 0] + inter2
outcome[seq(n) %% 2 == 0] = outcome[seq(n) %% 2 == 0] + rnorm(length(outcome)/2,mean=0,sd=0.1)
# The size parameter control the number of outliers during simulation.
id_outlier = sample(1:length(outcome), size = 10)
outcome[id_outlier] = outcome[id_outlier] + rnorm(length(id_outlier), mean=0,sd=2)
```

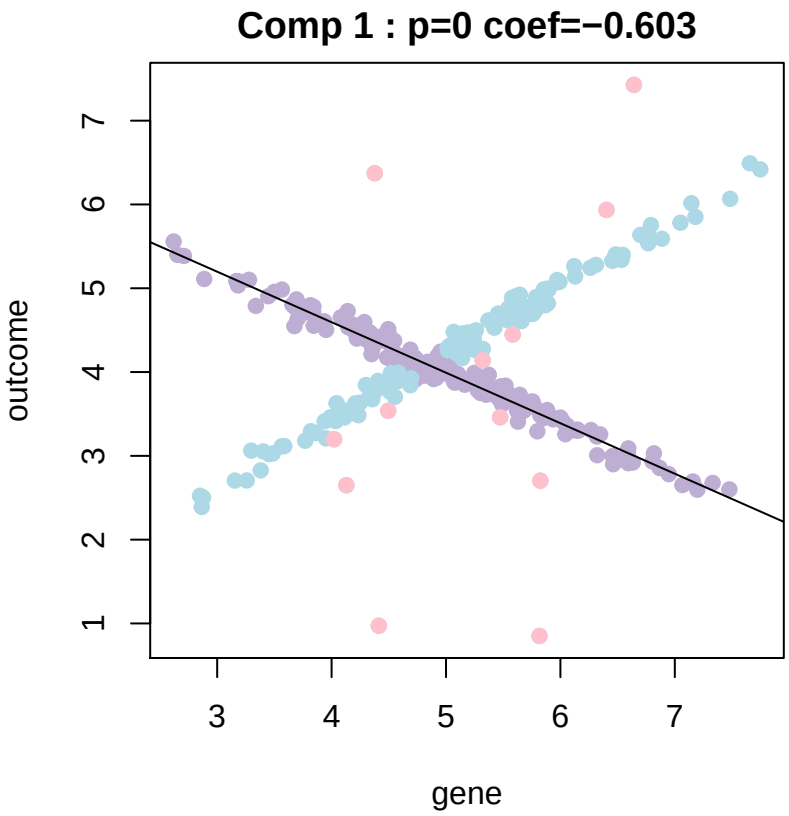
Run the Component-wise Adaptive Trimming method to detect outlier and estimate the parameter simultaneously.

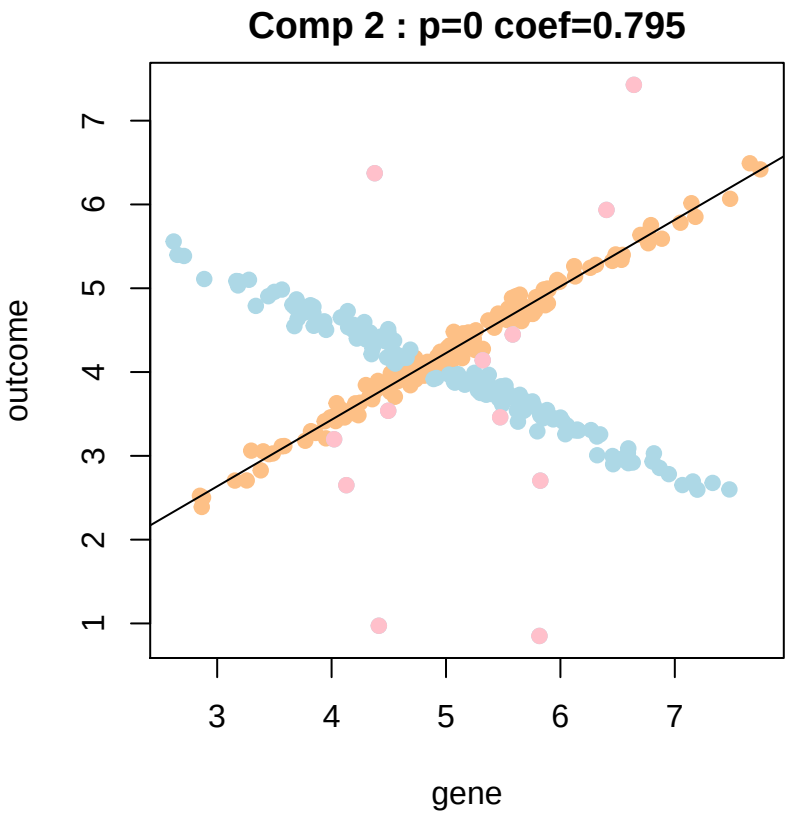
```r
res.c = MLM(ml.method="c", gene, outcome)
```

Extract the outlier information and plot the mixture regression line one by one.

```r
inds_in=res.c@inds_in
par(mfrow=c(1,1),mar = c(5, 8, 2, 8))
plot_CTLE(outcome~gene,data.frame(gene,outcome),nc=2,inds_in=inds_in)
```




Comp 1 : p=0 coef=−0.603
outcome
gene
1
2
3
4
5
6
7
3
4
5
6
7



S2.4 Example 4: High dimension, $K = 2$

The gene expression data is usually in high dimension. Only modeling the relationship between two vectors is not enough. For example, alternatively actived genes or pathways in different patient subpopulation subtantially hurdled the computational capability in studying the disease complexities and optimizing therapeutic strategy, which is fundamental question in precision health study. The set of genes used to execute a biological pathway of clinical response may very likely exist in more than one alternative form, thus lead to patient subpopulation exhibiting varied level of sensitivity and clinical response different cytotoxic drugs, posing a challenge in identifying gene features that are predictive for clinical response to existing but unknown patient subpopulation. Thus, we proposed the RBSL algorithm, following the supervised methods, to enable the identification of subsets genes. The detailed motivation and algorithm please refer to the RBSL paper.

- Mathmatical Model

$$argmin||y_{J_K} - X^T_{:,J_k}\beta_k - \beta_{k0}||, s.t.||\beta_k|| \leq \lambda_k$$

Data Simulation:

In total, we simulate a matrix which is 400 patients by 100 genes. The first twenty genes contributed to the first 200 patients. The continued twenty genes contributed to the last 200 patients. The simulation process is simply executed by a function *simu_data_sparse*, please see the package manual for more details.

```
n=400
bet1=bet2=rep(0,101)
bet1[2:21]=sign(runif(20,-1,1))*runif(20,2,5)
bet2[22:41]=sign(runif(20,-1,1))*runif(20,2,5)
bet=rbind(bet1,bet2)
pr=c(1,1)*0.5
sigs=c(1,1)####need to loop through 0.5, 1, 2
tmp_list = simu_data_sparse(n=n,bet=bet,pr=pr,sigma=sigs)
nit=1
nc=2
max_iter=50
x=tmp_list$x
print("Dimension of matrix (x):");dim(x)
```

```
## [1] "Dimension of matrix (x):"
```

```
## [1] 400 100
```

```
y=tmp_list$y
print("Length of outcome variable (y):"); length(y)
```

```
## [1] "Length of outcome variable (y):"
```

```
## [1] 400
```

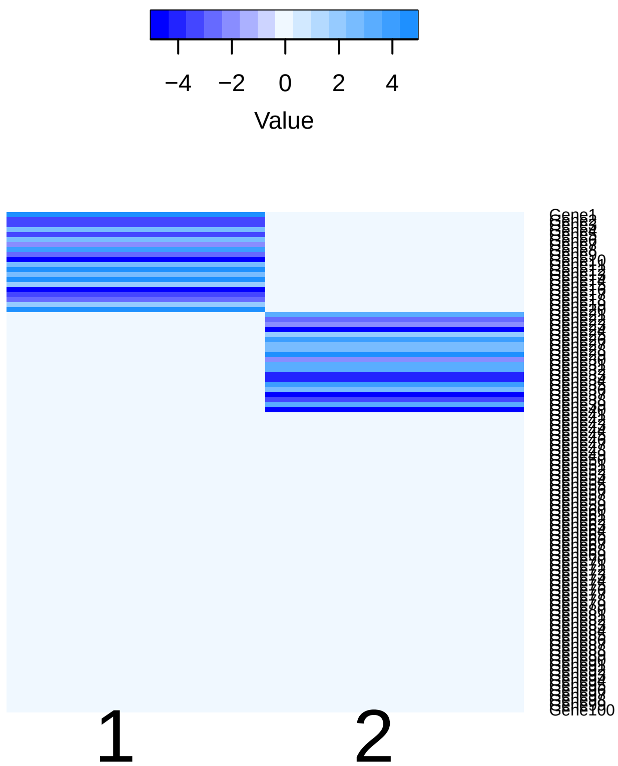
Run the RBSL method to identify the subspaces on which subsets of genes are explainable to the patient outcome variable. The parameter x is a high dimensional matrix and parameter y is the patient outcome variable.

```
res.d = MLM(ml.method = 'd',x=x,y=y)
# The result of the RBSL algorithm consist of 5 objects.
names(res.d)
```

```
## [1] "coffs" "clus"  "x"     "y"     "yhat"
```

Plot the predicted coefficient matrix. The column is the cluster membership and the row is the each cluster's coefficient value of genes.

```
#par(mfrow=c(1,1),mar = c(0.1, 0.1, 0.1, 0.1))
blockMap(res.d)
```



S3. Real Data Application

Place holder Place holder

S3.1 Breast Cancer Multi-Omics Data Filtered Outlier Mutually

Place holder Place holder

How we extract information from RNA-seq and ATAC-seq data.

```
# read the list object built-in the package
list('BRCA_source')
```

```
## [[1]]
## [1] "BRCA_source"
```

```
data_RNAseq_selected = BRCA_source$data_RNAseq_selected
data_ATACseq_selected = BRCA_source$data_ATACseq_selected
RNAseq_marker = BRCA_source$RNAseq_marker
ATACseq_marker = BRCA_source$ATACseq_marker

b4_TCGA<-Compute_Rbase_SVD(data_RNAseq_selected,RNAseq_marker)
b4_ATAC0<-Compute_Rbase_SVD(data_ATACseq_selected,ATACseq_marker)
# generate the RNA-seq and ATAC-seq variable below.
a1<-b4_TCGA[1,colnames(b4_ATAC0)]
b1<-b4_ATAC0[1,]
a2<-b4_TCGA[2,colnames(b4_ATAC0)]
b2<-b4_ATAC0[2,]
a3<-b4_TCGA[3,colnames(b4_ATAC0)]
b3<-b4_ATAC0[3,]
```

These variables also provide in another data frame object.

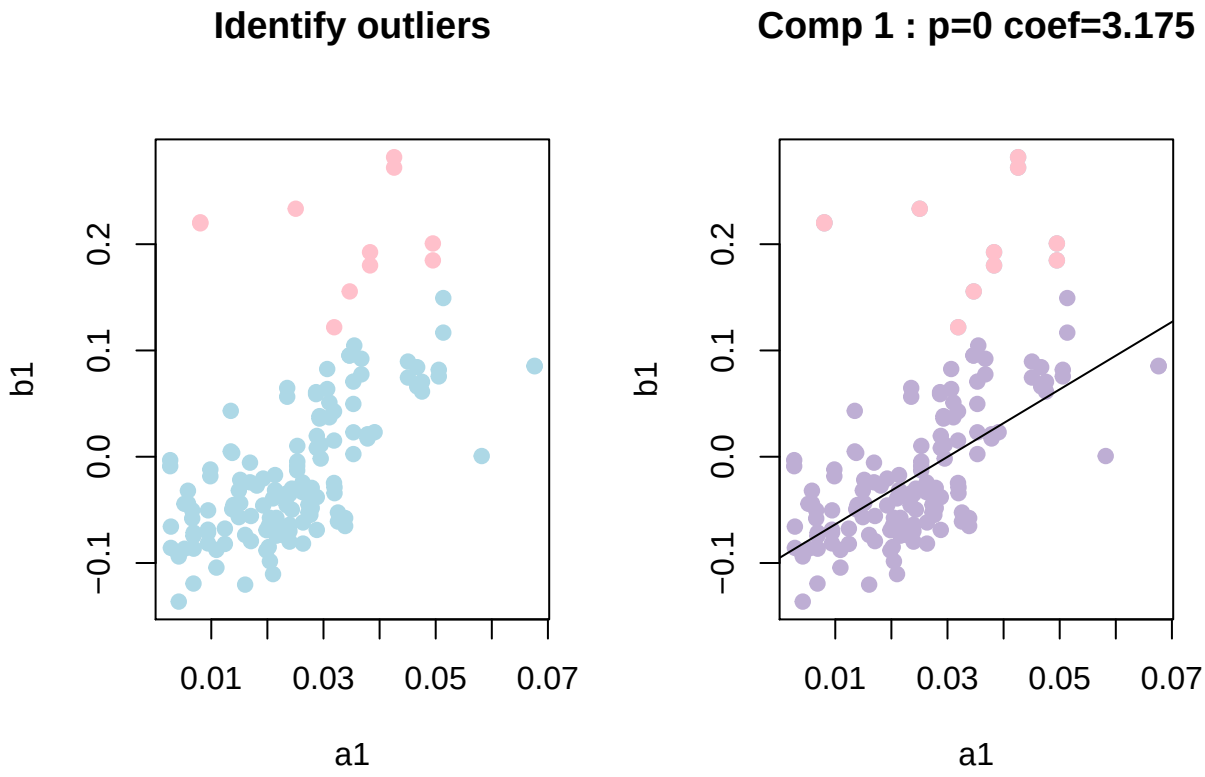
```
data(BRCA_ATAC_RNA_seq)
# First, extract the RNA-seq vectors
a1 = BRCA_ATAC_RNA_seq$a1
a2 = BRCA_ATAC_RNA_seq$a2
a3 = BRCA_ATAC_RNA_seq$a3
# Second, extract the ATAC-seq vectors
b1 = BRCA_ATAC_RNA_seq$b1
b2 = BRCA_ATAC_RNA_seq$b2
b3 = BRCA_ATAC_RNA_seq$b3
```

Call the main function to detect the outliers.

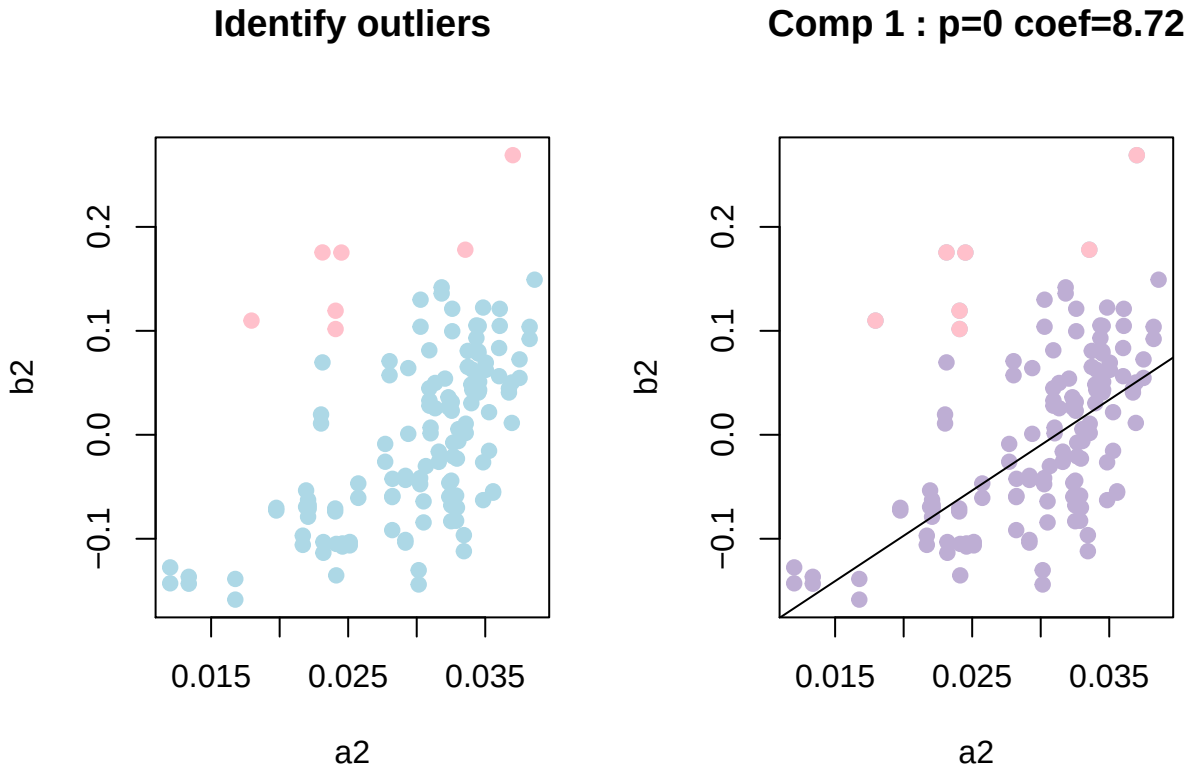
```
brca.1 = MLM(ml.method='c', gene=a1, outcome=b1,nit=10,nc=1)
brca.2 = MLM(ml.method='c', gene=a2, outcome=b2,nit=10,nc=1)
brca.3 = MLM(ml.method='c', gene=a3, outcome=b3,nit=10,nc=1)
```

Once we get the result from the method 1, we can extract the sample information that which sample should be a outlier or not.

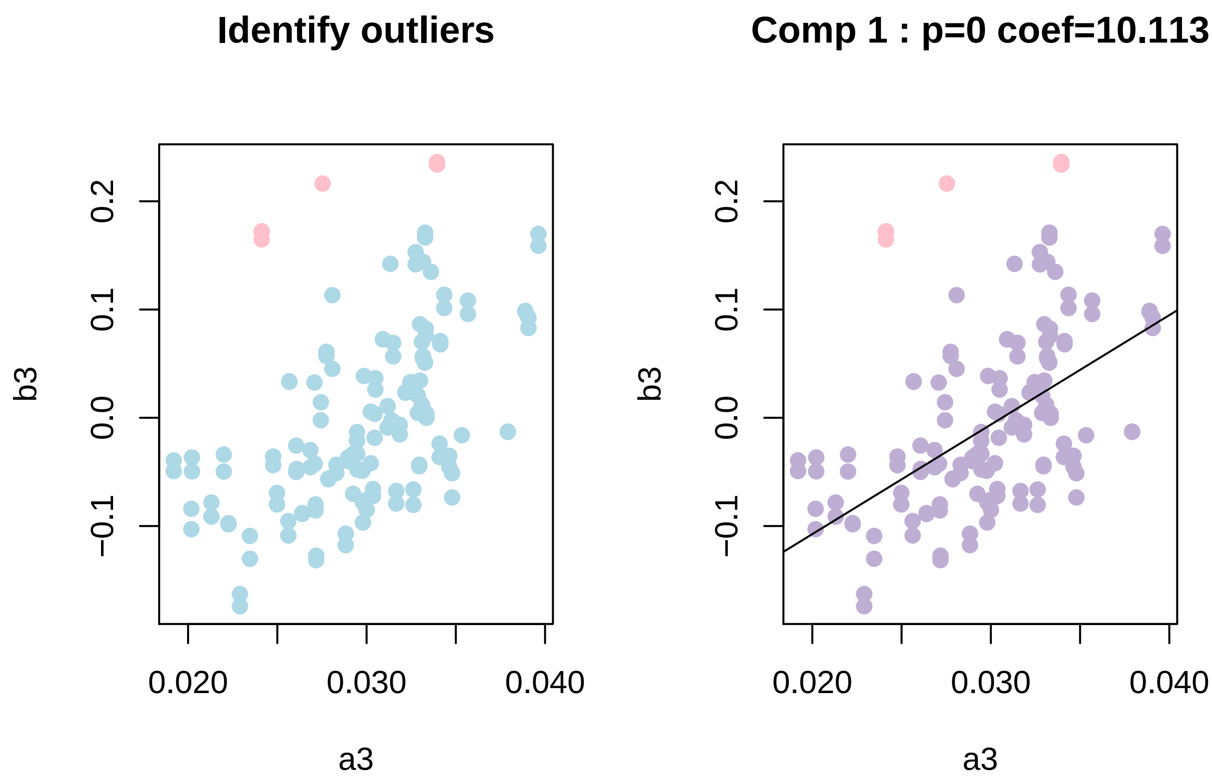
```
inds_in = brca.1@inds_in
par(mfrow=c(1,2),mar = c(4, 4, 6, 2))
plot_CTLE(b1~a1,data=data.frame(a1,b1),nc=1,inds_in=inds_in)
```




```
inds_in = brca.2@inds_in
par(mfrow=c(1,2),mar = c(4, 4, 6, 2))
plot_CTLE(b2~a2,data=data.frame(a2,b2),nc=1,inds_in=inds_in)
```




```
inds_in = brca.3@inds_in
par(mfrow=c(1,2),mar = c(4, 4, 6, 2))
plot_CTLE(b3~a3,data=data.frame(a3,b3),nc=1,inds_in=inds_in)
```



S3.2 Cytokine Dataset

Place holder Place holder

Prepare the built-in cytokine data .

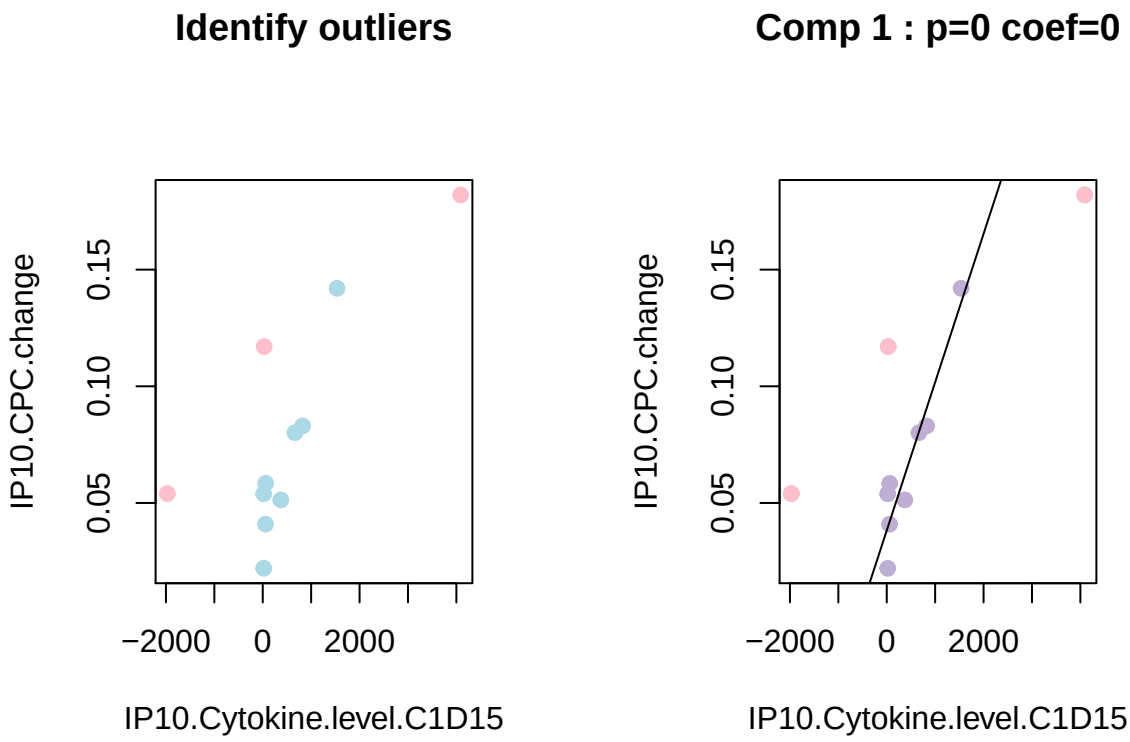
```
data(Cytokine_dataset)
# First, extract the two variable for IP10 gene
IP10.Cytokine.level.C1D15 = Cytokine_dataset$IP10.Cytokine.level.C1D15
IP10.CPC.change = Cytokine_dataset$IP10.CPC.change
# Second, extract the two variable for TNFa gene
TNFa.Cytokine.level.C1D15 = Cytokine_dataset$TNFa.Cytokine.level.C1D15
TNFa.CPC.change = Cytokine_dataset$TNFa.CPC.change
```

Then, call the main function to identify the outliers.

```
IP10.res = MLM(ml.method='c', gene=IP10.Cytokine.level.C1D15, outcome=IP10.CPC.change,nit=10,nc=1)
TNFa.res = MLM(ml.method='c', gene=TNFa.Cytokine.level.C1D15, outcome=TNFa.CPC.change,nit=10,nc=1)
```

Once we get the result from the method 1, we can extract the sample information that which sample should be a outlier or not.

```
inds_in = IP10.res@inds_in
par(mfrow=c(1,2),mar = c(4, 4, 8, 4))
plot_CTLE(IP10.CPC.change~IP10.Cytokine.level.C1D15,
          data=data.frame(IP10.Cytokine.level.C1D15,IP10.CPC.change),
          nc=1,inds_in=inds_in)
```



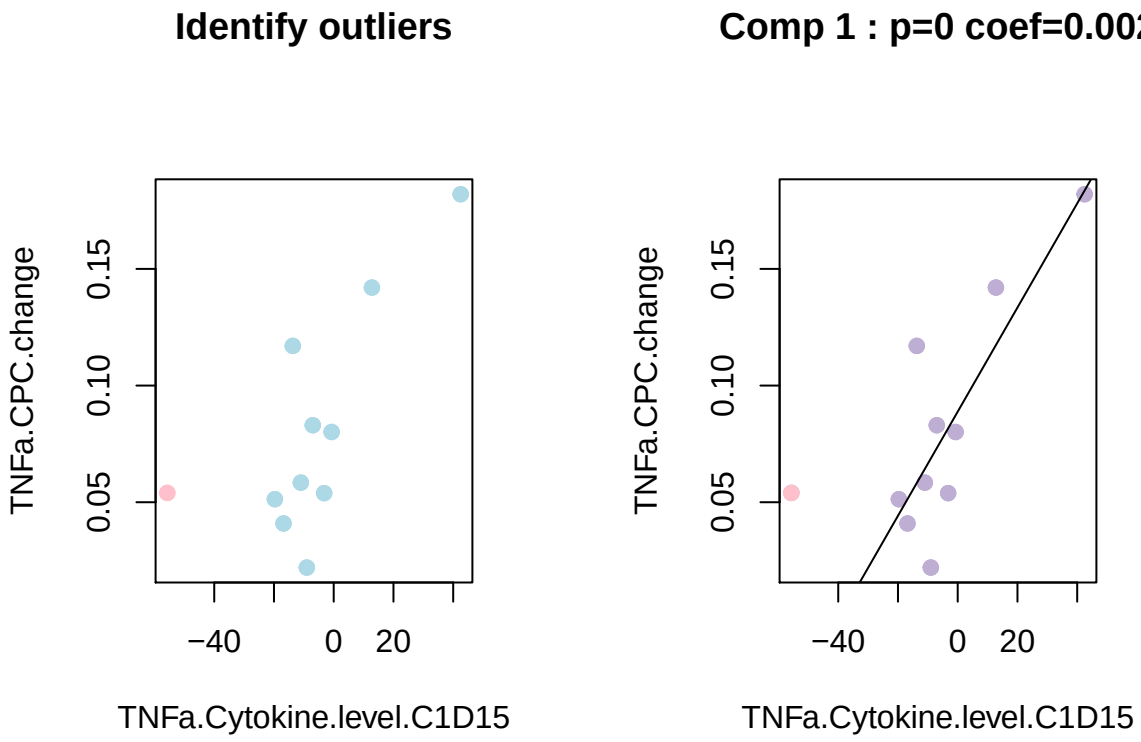
The coefficient is zero because we only take three digits after the decimal point. The approximate coefficient value shows below.

```
lm(IP10.CPC.change~IP10.Cytokine.level.C1D15)
```

```
##
## Call:
## lm(formula = IP10.CPC.change ~ IP10.Cytokine.level.C1D15)
##
## Coefficients:
##                (Intercept)  IP10.Cytokine.level.C1D15
##                  0.0670508                  0.0000258
```

Once we get the result from the method 1, we can extract the sample information that which sample should be a outlier or not.

```
inds_in = TNFa.res@inds_in
par(mfrow=c(1,2),mar = c(4, 4, 8, 4))
plot_CTLE(TNFa.CPC.change~TNFa.Cytokine.level.C1D15,
          data=data.frame(TNFa.Cytokine.level.C1D15,TNFa.CPC.change),
          nc=1,inds_in=inds_in)
```



S4. Integrated Robust Mixture Regression Algorithms and Example

S4.1 Robust Mixture Regression Algorithms

Place holder Place holder

S4.2 Example

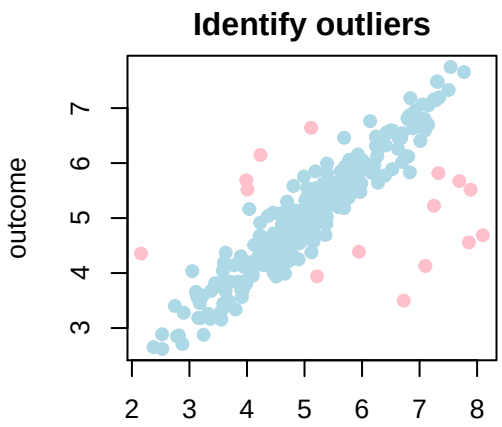
Another important main function in the package named *rmr* which provide five optional algorithms to fit the robust mixture regression model. The implemented algorithms includes: CTLERob stands for Component-wise adaptive Trimming Likelihood Estimation based mixture regression; mixbi stands for mixture regression based on bi-square estimation; mixLstands for mixture regression based on Laplacian distribution; TLE stands for Trimmed Likelihood Estimation based mixture regression. For more detail of the algorithms, please refer to references in the DESCRIPTION file.

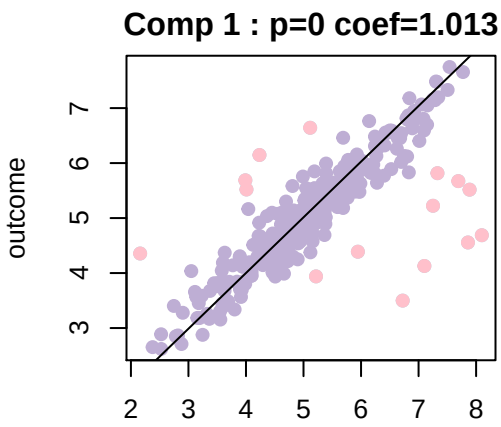
User can select the algorithm by setting the parameter *lr.method*. One example is below. For description of *gaussData* please refer to package manual.

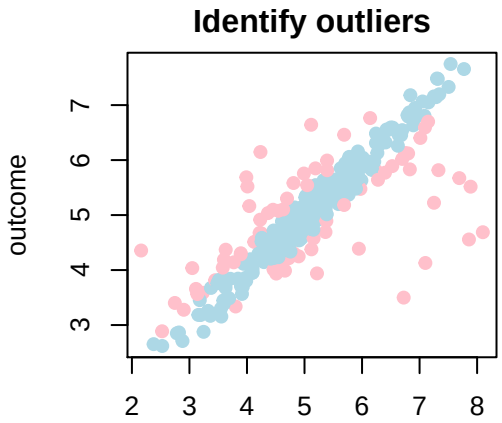
```
# gaussData
x=(gaussData$x);y=as.numeric(gaussData$y);
formula01=as.formula("y~x")
example_data01=data.frame(x,y)
res_rmr = rmr(lr.method='flexmix', formula=formula01, data=example_data01)
res_rmr = rmr(lr.method='TLE', formula=formula01, data=example_data01)
```

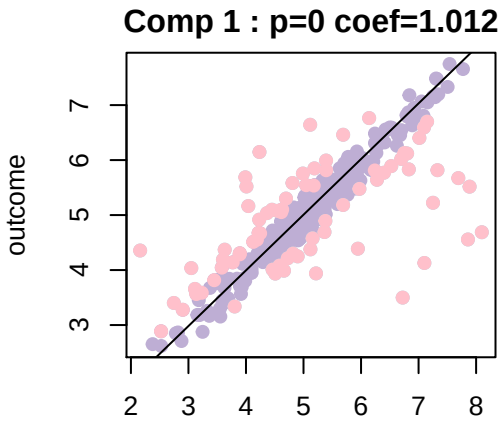
S4.3 Adjust the trimming ratio for robust regression

```
set.seed(12345)
gene = rnorm(n=300, mean=5, sd=1)
inter = 0 ; slope = 1
outcome = slope * gene + inter
outcome = outcome + rnorm(n=length(outcome),mean=0, sd=0.3)
id_outlier = sample(1:length(outcome), size = 0.1*length(outcome))
outcome[id_outlier] = outcome[id_outlier] + rnorm(length(id_outlier), mean=0,sd=2)

# Use tRatio to adjust the ratio of outlier for trimming
res_rmr_r1 = rmr(lr.method='TLE', formula=outcome~gene, data=data.frame(outcome,gene),nc=1,
                 tRatio=0.05)
inds_in_r1 = res_rmr_r1@inds_in
# change tRatio to a larger percentage
res_rmr_r2 = rmr(lr.method='TLE', formula=outcome~gene, data=data.frame(outcome,gene),nc=1,
                 tRatio=0.25)
inds_in_r2 = res_rmr_r2@inds_in

# set the figure format and margin
par(mfrow=c(2,2),mar = c(2, 4, 2, 4))
# use plot module to draw the data points and the regression line.
plot_CTLE(outcome~gene,data=data.frame(outcome,gene),nc=1,inds_in=inds_in_r1)
plot_CTLE(outcome~gene,data=data.frame(outcome,gene),nc=1,inds_in=inds_in_r2)
```


Identify outliers
outcome
3
4
5
6
7
2
3
4
5
6
7
8


Comp 1 : p=0 coef=1.013
outcome
3
4
5
6
7
2
3
4
5
6
7
8


Identify outliers
outcome
3
4
5
6
7
2
3
4
5
6
7
8


Comp 1 : p=0 coef=1.012
outcome
3
4
5
6
7
2
3
4
5
6
7
8