RobMixReg: an R package for robust, flexible and high dimensional mixture regression

Wennan Chang[1, 2], Changlin Wan[1, 2], Chun Yu[3], Weixin Yao[4], Chi Zhang[1,2*], and Sha Cao[2*]

[1]Department of Electrical and Computer Engineering, Purdue University, Indianapolis, IN, 46202, USA

[2]Department of Medical and Molecular Genetics, Department of Biostatistics, Center for Computational Biology and Bioinformatics, Indiana University School of Medicine, Indianapolis, IN,46202, USA.

[3]School of Statistics, Jiangxi University of Finance and Economics, Nanchang, China.

[4]Department of Statistics, University of California, Riverside, CA, USA.

*To whom correspondence should be addressed. Email: czhang87@iu.edu. shacao@iu.edu.

Abstract

Motivation: Mixture regression has been widely used as a statistical model to untangle the latent subgroups of the sample population. Traditional mixture regression faces challenges when dealing with: 1) outliers and versatile regression forms; and 2) the high dimensionality of the predictors. Here, we develop an R package called RobMixReg, which provides comprehensive solutions for robust, flexible as well as high dimensional mixture modeling.

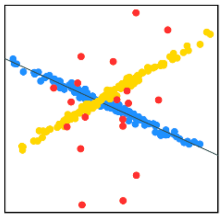
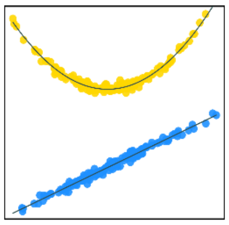
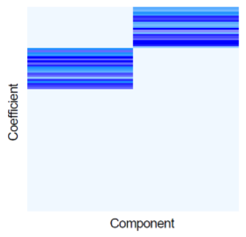
Availability and Implementation: RobMixReg R package and associated documentation is available at CRAN: https://CRAN.R-project.org/package=RobMixReg.

1 Introduction

The big “Omics” data era has profoundly changed biology by scaling data acquisition, and transformed the biomedical research ecosystem from traditional case-based studies to modern large scale data driven research. For example in cancer research, the Cancer Genome Atlas (TCGA) has collected biospecimens and matched clinical phenotypes for over 10,000 cancer patients (Weinstein, et al., 2013). Driven by the increasing availability of phenotypic and multiple omics data, researchers have gained unprecedented opportunity to interrogate biology in novel and creative ways. Regression-based association analysis remains to be the most popular data mining tool for hypothesis generation, owing to its ease of interpretability. However, the traditional linear regression model cannot adequately fit the complexity of the biomedical data caused by: 1) high leverage noise and outliers resulted from the large scale technical experiments; 2) the presence of subpopulations among the wide spectrum of collected samples; 3) the high-dimensionality of the molecular features. This raises an urgent demand of a toolkit to specifically handle these challenges, and more importantly, evaluate the level of outlier contamination, the level of heterogeneity, as well as the level of sparseness of the features, or in other words, the most optimal model to fit the data.

We have developed an R package called “RobMixReg” to provide a comprehensive solution to mining the latent relationships in biomedical data. The major contribution of RobMixReg lies in the following key aspects: 1) for low dimensional predictors, RobMixReg allows for robust parameter estimations, and detection of the outliers with adaptive trimming; 2) RobMixReg allows different mixture components to have flexible forms of predictors; 3) RobMixReg handles the high-dimensional predictors by regularizing the regression coefficients of each component, with a data-driven level of sparsity; 4) RobMixReg provides the capability for order selection. To the best of our knowledge, there is no software package having these comprehensive functionalities and flexibility with similar purposes as RobMixReg in mining the biomedical data. We believe RobMixReg will be a valuable addition to the biomedical research community.

Table 1. The functionalities of RobMixReg

	Robust mixture regression	Flexible mixture regression	High dimensional mixture regression
Options	Methods: *ltsReg, flexmix*, *CAT*, *TLE*, *mixbi*, *mixLp* Order selection: BIC	Method: *mixtureReg* Model and order selection: BIC	Method: *CSMR* Order selection: BIC, CV
Illustra-tions			

2 Methods

Finite Mixture Gaussian Regression (FMGR) is a widely used model to explore the latent relationship between the response and predictors (Böhning, 1999). When fitting a K-component FMGR model with N observations of response variable y and P-variate predictor $\boldsymbol{x}$, we usually consider maximizing the following likelihood function:

$$\operatorname{argmax} \Sigma_{i=1}^{N} \log\left(\Sigma_{k=1}^{K} \pi_k \phi\left(y_i - \boldsymbol{x}_i^T \boldsymbol{\beta}_k; 0, \sigma_k^2\right)\right) \quad (1)$$

where $\pi_k, \boldsymbol{\beta}_k, \sigma_k$ are the prior probability, the regression coefficients and the standard deviation of the k-th component; $\phi(\cdot)$ is the normal density function. The unknown parameters in the above FMGR model is usually estimated by the EM algorithm (Dempster, et al., 1977),

RobMixReg deals with regression scenarios for both low and high dimensional predictors. As shown in Figure 1, for low dimensional predictors, RobMixReg considers robust and flexible mixture modeling; for high dimensional predictors, RobMixReg considers adaptive feature selection. In addition, for both cases, it provides a way for order selection. More details on implementation of these methods could be found in the supplementary document.

2.1 Robust mixture regression

In RobMixReg, the traditional EM algorithm was implemented in by the *flexmix* function in the *flexmix* package (Leisch, 2004). The traditional EM algorithm is very sensitive to outliers due to the least square criterion used in the M step. Several robust approaches were proposed to estimate the FMGR parameters, including the trimmed likelihood estimator (*TLE*) (Neykov, et al., 2007), the Component-wise Adaptive Trimming (*CAT*) algorithm (Chang, et al., 2020), the a robust bi-square criterion method (*mixbi*) (Bai, et al., 2012), the robust mixture of t-distributions (*mixt*) (Yao, et al., 2014), robust mixture of Laplace distribution (*mixLp*) (Song, et al., 2014), and a mean shift mixture regression model (*RM2*) (Yu, et al., 2017). In the RobMixReg package, for $K = 1$, the robust mixture regression degenerates to robust linear regression, and the function *ltsReg* in the package *robustbase* is implemented. For $K > 1$, methods including the *flexmix*, *TLE*, *CAT*, *mixbi*, and *mixLp* are all implemented.

2.2 Flexible mixture regression

When fitting two or more lines through the data, there could be different predictors and their functional forms in different lines. For example, two lines may involve different subsets of the predictors; or a line could include nonlinear transformations of the predictors to relax the "linearity" constraints. We provide the capability for flexible mixture regression models such that the user could specify the regression functions for different components. Different from traditional FMGR model, where the predictors consist of only themselves, in flexible mixture regression, users could specify the predictors and the transformations of the predictors. Such a flexibility is important as the complexity of the model could be well controlled. We include a flexible method in our package to enable users to decide what predictors to be used for each mixing component developed in (Zhou, 2017).

2.3 High dimensional mixture regression

While mixture regression is capable of handling the heterogeneous relationships, it does not work well in the case of high dimensional molecular features, where the total number of parameters to be estimated is far more than the total number of observations. Hence, restrictions on the sparsity level of $\boldsymbol{\beta}_k$ needs to be introduced to shrink the insignificant regression coefficients. RobMixReg implemented the *CSMR* algorithm (Chang, et al., 2020), which performs simultaneous sample clustering and feature selection. It has a built-in functionality for the automatic selection of the sparsity level, which substantially improves both the computational efficiency and biological interpretability.

2.4 Order selection

Determining the order (number of components, K) for mixture models has long been a challenging and important problem. We use the Bayesian information criterion (BIC) for order selection, which is calculated as:

$$BIC(\theta_K^*) = -2\mathcal{L}(\theta_K^*) + \log(N)\, d_{\theta_K^*}$$

Here, θ_K^* denotes the parameter estimates for component number K, and $\mathcal{L}(\theta_K^*)$ is the associated log likelihood; N is the total number of observations; $d_{\theta_K^*}$ is the total number of parameters, and $d_{\theta_K^*} = B_0 + (K-1) + K + N_0$. Here, B_0 is the total number of non-zero regression coefficients; $K-1$, the number of component proportions; K, the number of standard deviations for the K components; N_0, the total number of outliers detected. Note that, we currently do not consider outliers for the flexible and high dimensional settings, hence, $N_0 = 0$ in these cases. BIC criterion could be also used for model selection in flexible mixture regression. In addition to the BIC criteria, RobMixReg also offers a cross validation algorithm for the selection of K for the high dimensional case.

3 Results

We used four simulation and four real data examples to demonstrate the capability of RobMixReg (see supplementary file), in dealing with robust mixture regression, flexible mixture regression and high dimensional mixture regression. We provided the solution for order selection in each scenario.

Funding

This work has been supported by NSF grant No.1850360 (Cao).

References

Bai, X., Yao, W. and Boyer, J.E. Robust fitting of mixture regression models. *Computational Statistics & Data Analysis* 2012;56(7):2347-2359.

Böhning, D. Computer-assisted analysis of mixtures and applications: meta-analysis, disease mapping and others. CRC press; 1999.

Chang, W., *et al.* Supervised clustering of high dimensional data using regularized mixture modeling. *arXiv preprint arXiv:2007.09720* 2020.

Chang, W., *et al.* A New Algorithm using Component-wise Adaptive Trimming For Robust Mixture Regression. *arXiv preprint arXiv:2005.11599* 2020.

Dempster, A.P., Laird, N.M. and Rubin, D.B.J.J.o.t.R.S.S.S.B. Maximum likelihood from incomplete data via the EM algorithm. 1977;39(1):1-22.

Leisch, F. Flexmix: A general framework for finite mixture models and latent glass regression in R. 2004.

Neykov, N.*, et al.* Robust fitting of mixtures using the trimmed likelihood estimator. 2007;52(1):299-308.

Song, W., Yao, W. and Xing, Y. Robust mixture regression model fitting by Laplace distribution. *Computational Statistics & Data Analysis* 2014;71:128-137.

Weinstein, J.N.*, et al.* The cancer genome atlas pan-cancer analysis project. *Nature genetics* 2013;45(10):1113.

Yao, W., Wei, Y. and Yu, C. Robust mixture regression using the t-distribution. *Computational Statistics & Data Analysis* 2014;71:116-127.

Yu, C., Yao, W. and Chen, K. A new method for robust mixture regression. *Can J Stat* 2017;45(1):77-94.

Zhou, T. Mixture of Linear Regressions. https://github.com/txzhou/mixtureReg. 2017.

RobMixReg: an R package for robust, flexible and high dimensional mixture regression – Supplementary Materials

Wennan Chang Changlin Wan Chun Yu Weixin Yao Chi Zhang
Sha Cao

Contents

S1. Introduction

Regression-based association analysis remains to be the most popular data mining tool for hypothesis generation, owing to its ease of inter-pretability. However, the traditional linear regression model cannot adequately fit the complexity of the biomedical data caused by: 1) high leverage noise and outliers resulted from the large scale technical experiments; 2) the presence of subpopulations among the wide spectrum of collected samples; 3) the high-dimensionality of the molecular features. Finite Mixture Gaussian Regression (FMGR) is a widely used model to explore the latent relationship between the response and predictors, and the goal of mixture regresison is to maximize the following likelihood:

$$argmax \sum_{i=1}^{N} log(\sum_{k=1}^{K} \pi_k \phi(y_i - \mathbf{x}_i^T \beta_k; 0, \sigma_k^2))$$

Here, we are modeling the relationship between $\mathbf{x}$ and y as a mixture of K regression lines, $K \geq 1$. $(\mathbf{x_i}, y_i)$ is the i-th observations $i = 1, ..., N$, β_k are the regression coefficients for the k-th regression line; ϵ_{ik} is the error term; σ_k is the standard deviation of ϵ_{ik}.

'RobMixReg' is a package that provides a comprehensive solution to mining the latent relationships in biomedical data, ranging from the regulatory relationships between different molecular elements, to the the relationships between pheno-types and omic features, in the presence of outliers, latent subgroups and high dimensional molecular features.

The major contribution of RobMixReg lies in the following key aspects: 1) for low dimensional predictors, it handles outliers in modeling the latent relationships, and provides flexible modeling to allow for different predictors for different mixture components; 2) for high dimensional predictors, it detects subgroups and select informative features simultaneously. The RobMixReg package is a powerful exploratory tool to mining the latent relationships in data, which is often obfuscated by high leverage outliers, distinct subgroups and high dimensional features.

The RobMixReg package could be installed from CRAN:

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

It could be also installed from GitHub:

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

S2. Case Studies

S2.0 Data Simulation:

The function *simu_func* is used to generate the synthetic data for various of cases. The input of *simu_func* is :

- β A matrix whose k-th column is the regression coefficients matrix for the k-th component
- σ A vector whose k-th element is the standard deviation of the error term for the k-th regression component
- *alpha* The proportion of the observations being contaminated by the outlier

The output of *simu_func* is :

- x Matrix of the predictors with dimension $N \times (P+1)$, where P is the total number of predictors, and the first column is an all-one vector.
- y Vector of response variable where *alpha* proportion of the observations are contaminated by outliers.

S2.1 Robust mixture regression

We adopt the mean-shift model in the presence of outliers, such that each outlier requires one additional parameter to cancel out the unusually large residuals. The outlier parameters are then regularized to achieve a balance of model complexity and model fitness.

$$argmax \sum_{i=1}^{N} log(\sum_{k=1}^{K} \pi_k \phi(y_i - \mathbf{x}_i^T \beta_k - \gamma_{ik}\sigma_k; 0, \sigma_k^2)) - \sum_{i=1}^{N}\sum_{k=1}^{K} P_\lambda(|\gamma_{ij}|)$$

Here, γ_{ik} models the level of outlierness for the i-th observation; $P_\lambda(|\gamma_{ij}|)$ is a penalty function with a tuning parameter λ controlling the degrees of penalization on the γ_{ik}.

We have intergrated multiple state-of-the-art methods in our package to robustly fit two or more regression lines. The default method is CAT. Other options inlcude TLE, flexmix, mixLp, mixbi, and they could be passsed to the MLM function by the parameter 'rmr.method'. Among them, CAT and TLE could detect the identities of the outliers, and outliers are removed before model estimation; while the rest of the methods perform model estimation in the presence of outliers. For TLE to work, a trimming proportion needs to be provided, and the default is 0.05.

```
# simulate single variable x and y with linear relationship but contaminated by outliers.
set.seed(12345)
beta = rbind(c(0.2, 0.8), c(7, -0.6));  sigma=c(0.1,0.1); alpha=0.05;
data_case3 = simu_func(beta, sigma, alpha)
x3 = data_case3$x
y3 = data_case3$y
# Runing robust mixture regression by setting parameter 'ml.method'
# to 'rmr' in the wrapper function MLM.
res.c = MLM(ml.method="rmr", rmr.method='cat',x=x3, y=y3,nc=2) # CAT method
```

```
# Trimmed likelihod estimation. Here, the uers need to
# provide the ratio of outlier smaples.
res.c.TLE = MLM(ml.method="rmr", rmr.method='TLE', x=x3, y=y3)
res.c.flexmix = MLM(ml.method="rmr", rmr.method='flexmix', x=x3, y=y3)
res.c.Lp = MLM(ml.method="rmr", rmr.method='mixLp', x=x3, y=y3)
res.c.bi = MLM(ml.method="rmr", rmr.method='mixbi', x=x3, y=y3)
```

We can extract the outlier samples from the output of MLM.

```
inds_out = which(res.c$cluMem == -1)
print(inds_out)
```

```
##  [1]  16  28  49  65  75  79  97 101 104 110 131 174 176 215 241 280 283 295 298
```

```
inds_in = which(res.c$cluMem != -1)
```

The mixture regression parameters could be extracted by:

```
res.c$coff
```

```
##                      Comp.1      Comp.2
## coef.(Intercept) 0.24017568  7.02935732
## coef.x           0.79663354 -0.60653274
## sigma            0.08950816  0.09183522
##                  0.49215978  0.50784022
```

Here, the first two rows are regression coefficients, third row for the standard deviations, and fourth row for the component proportions.

We can visualize the fitted regression and the outliers by 'compPlot' function with 'rlr' parameter, shows in figure 1.

```
par(mfrow=c(1,2),mar = c(4, 4, 6, 2))
compPlot(type='rlr', x=x3,y=y3, nc=2, inds_in=inds_in)
```

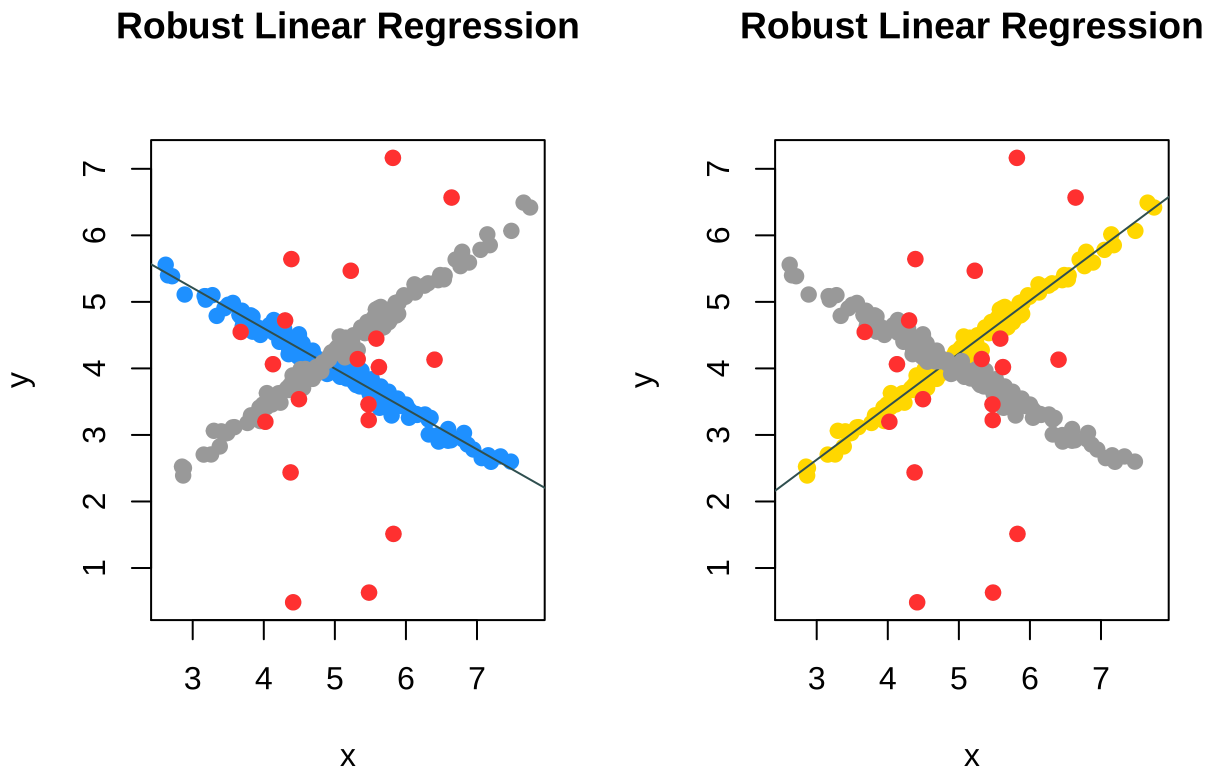


Figure 1: Robust mixture regression result

When $K = 1$, this degenerates to ordinary robust linear regression. We call for robust linear regression in the *MLM* wapper function by letting the 'ml.method' equal to 'rlr'. In the core of the method is the 'ltsReg' function from the 'robustbase' package.

```
# Simulate single variable x and y with linear relationship but contaminated by outliers.
beta = c(1, 0.8); sigma=0.3; alpha= 0.1
data_case1 = simu_func(beta, sigma, alpha)
x1 = data_case1$x
y1 = data_case1$y
# Runing robust linear regression by setting parameter ml.method
# to 'rlr' in the wrapper function MLM.
res.a = MLM(ml.method='rlr', x=x1, y=y1)
```

We can extract the outlier samples from the output of MLM.

```
inds_out = which(res.a$cluMem == 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 regression parameters could be extracted by:

```
res.a$coff
```

```
##                [,1]
## Intercept 0.9569727
## coef.x    0.8153754
## sd        0.2945930
```

We can visualize the fitted regression and the outliers by *compPlot* function in figure 2.

```
inds_in = which(res.a$cluMem == 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.
compPlot(type='rlr', x=x1,y=y1,nc=1, inds_in=inds_in)
```

S2.2 Mixture regression with flexible modeling

When fitting two or more lines through the data, there could be different regimes with the predictors in different lines. For example, two lines may involve different subsets of the predictors; or a line could include nonlinear transformations of the predictors to relax the "linearity" concerns. Different from the traditional mixture regression, RobMixReg allows for different set of predictors and their transformations as predictors for different component.

$$argmax \sum_{i=1}^{N} log(\sum_{k=1}^{K} \pi_k \phi(y_i - g_k^T(\mathbf{x}_i)\beta_k; 0, \sigma_k^2))$$

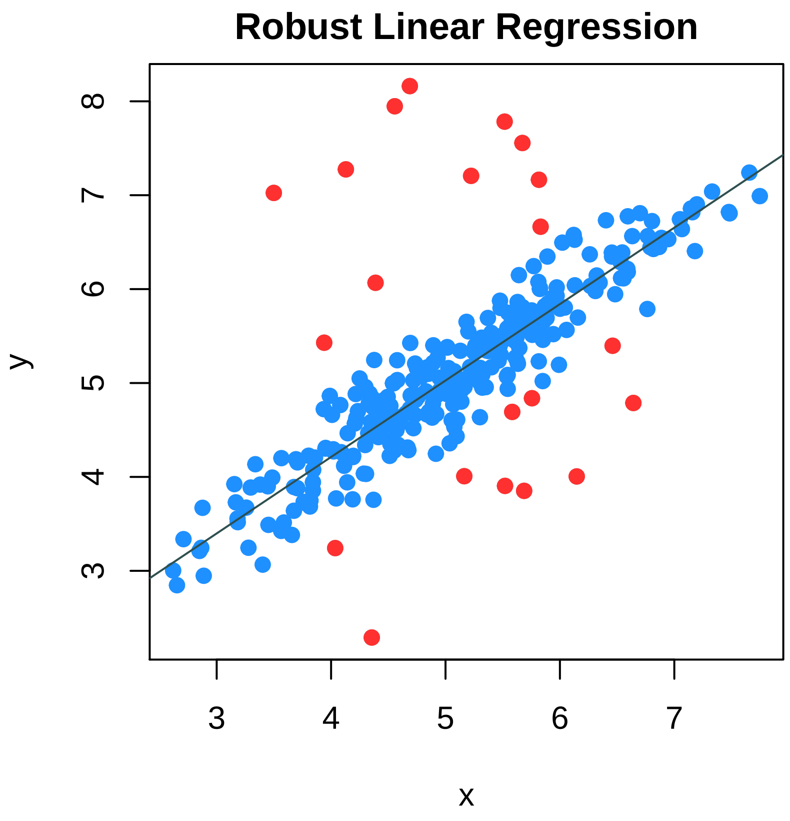


Figure 2: Robust linear regression result

Here, $g_k(\cdot)$ is a vectorized function that transforms the original predictors in regression componetn K, to combinations of the predictors and their transformations.

RobMixReg can easily achieve flexible modeling by specifying the regression formulas for different components. To enable flexible mixture regression, we call robust mixture regression in the *MLM* wapper function by letting the 'ml.method' be 'fmr'. In the core of the method is the 'mixtureReg' function from the 'mixtureReg' package.

```
# simulate single variable x and y with linear relationship but contaminated by outliers.
# beta = c(0.8, -0.01); inter = c(0.2, 4)
beta = rbind(c(0.2, 0.8), c(4, -0.01)); sigma = c(1,1)
data_case2 = simu_func(beta, sigma,alpha=NULL)
x2 = data_case2$x
y2 = data_case2$y
# Runing robust mixture regression by setting parameter 'ml.method'
# to 'fmr' in the wrapper function MLM.
res.b = MLM(ml.method='fmr', x = x2, y = y2,
            b.formulaList = list(formula(y ~ x),formula(y ~ 1)))
```

The mixture regression parameters could be extracted by:

```
res.b$coff
```

```
##                    1          2
## Intercept 0.20084960 3.94766052
## coff.x    0.80273914 0.00000000
## sd        0.09104395 0.09222377
## mx        0.50333333 0.49666667
```

Here, the first two rows are regression coefficients, third row for the standard deviations, and fourth row for the component proportions.

We can visualize the fitted regression lines by *compPlot* function and declare the 'type' parameter as 'mr' shows in figure 3.

```
par(mfrow=c(1,1), mar = c(5, 8, 2, 8))
compPlot(type='mr', res=res.b$res)
```

S2.3 High dimensional mixture regression

Given high dimensional predictors, which is often the case in biological data, we have the total number of parameters to be estimated far more than the total number of observations, making the aofrementioned methods fail. In addition, the dense linear coefficients makes it hard to deduce the subgroup specific features and make meaningful interpreta-tions. We need to add regularization to the regression coefficients to achieve sparse mixture regression.

- Mathmatical Model

$$argmax \sum_{i=1}^{N} log(\sum_{k=1}^{K} \pi_k \phi(y_i - \mathbf{x}_i^T \beta_k; 0, \sigma_k^2)) - \sum_{k=1}^{K} \lambda_k (\sum_{j=1}^{P} |\beta_{jk}|)$$

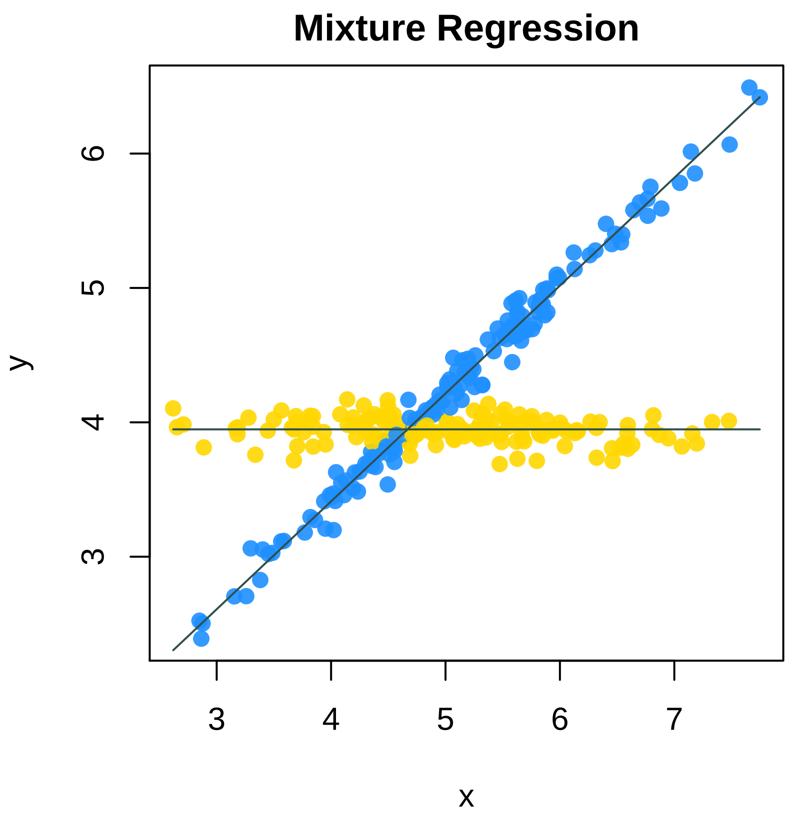


Figure 3: Flexible mixture regression result

Different from the low dimensional setting, we are imposing constraints on the coefficients through the penalty parameter λ_k. The CSMR algorithm is used here, which adaptively selects the penalty parameters λ_k, hence there is no need for model selection with regards to λ_k.

```
# simulate single variable x and y with linear relationship but contaminated by outliers.
set.seed(12345)
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)
tmp_list = simu_func(beta=bet,sigma=c(1,1),alpha=NULL)
x=tmp_list$x
print(paste("Dimension of matrix (x):", dim(x)[1], ', ', dim(x)[2],sep='') )
```

```
## [1] "Dimension of matrix (x):400, 100"
```

```
y=tmp_list$y
print(paste("Length of outcome variable (y):", length(y),sep='') )
```

```
## [1] "Length of outcome variable (y):400"
```

```
sink("nul") ;res.d = MLM(ml.method = 'hrmr',x=x,y=y); sink()
hx=x;hy=y
```

The mixture regression parameters could be extracted by:

```
head(res.d$coff)
```

```
##          [,1]      [,2]
## feature1    0  3.272612
## feature2    0  2.918706
## feature3    0  5.028459
## feature4    0  4.107363
## feature5    0 -3.922227
## feature6    0 -3.189251
```

Here, the first $(P+1)$ rows are regression coefficients, $(P+2)$-th row for the standard deviations, and $(P+3)$-th row for the component proportions.

We visualize the fitted coefficients using heatmap shows in figure 4. The columns are the clusters and the rows are the features.

```
#par(mfrow=c(1,1),mar = c(0.1, 0.1, 0.1, 0.1))
compPlot(type='block', res=res.d)
```

S2.4 Order Selection

BIC based order selection

We provide two methods for selecting the number of components, K: Bayesian information criterion (BIC) and cross validation. Meanwhile, in flexible modeling, even with the same K, the models could still differ in

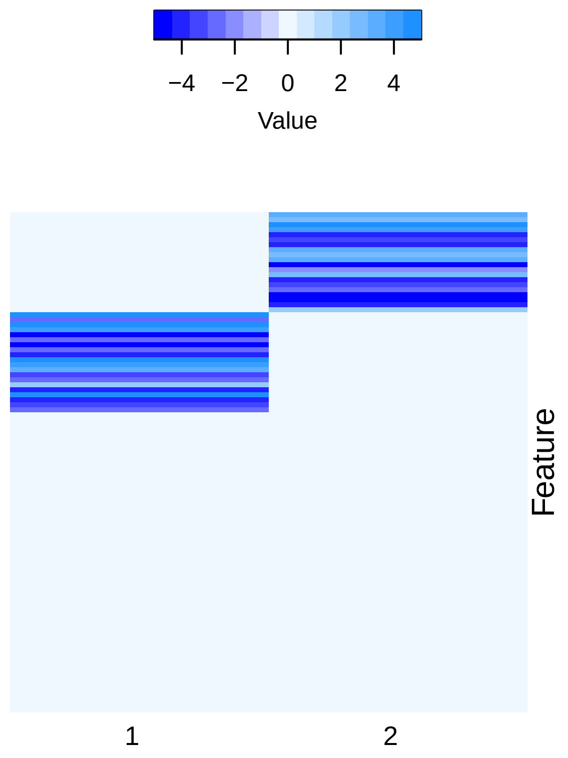


Figure 4: High dimensional feature space robust mixture regression result

their complexity, and we also provided BIC metric for model selection. The BIC of each fitted model could be extraced in slot 'BIC' by declaring the parameter 'ml.method':

Robust mixture regression using BIC

```
res.ba = MLM(ml.method='rlr', x=x1, y=y1)
res.bc = MLM(ml.method="rmr" ,x=x1, y=y1,nc=2)
res.bb = MLM(ml.method="rmr" ,x=x1, y=y1,nc=3)
print(res.ba$BIC)
```

```
## [1] 284.0409
```

```
print(res.bc$BIC)
```

```
## [1] 230.1078
```

```
print(res.bb$BIC)
```

```
## [1] 311.9934
```

Flexbile mixture regression using BIC

```
res.fa = MLM(ml.method='fmr', x = x2, y = y2,
            b.formulaList = list(formula(y ~ x),formula(y ~ 1)))
res.fb = MLM(ml.method='fmr', x = x2, y = y2,
            b.formulaList = list(formula(y ~ x),formula(y ~ x)))
res.fc = MLM(ml.method='fmr', x = x2, y = y2,
            b.formulaList = list(formula(y ~ x),formula(y ~ x+ I(x^2))))
res.fd = MLM(ml.method='fmr', x = x2, y = y2,
            b.formulaList = list(formula(y ~ x),formula(y ~ x),formula(y ~ x)))

# res.b = MLM(ml.method='fmr', x = x2, y = y2,
#              b.formulaList = list(formula(y ~ x),formula(y ~ 1)))
print(res.fa$BIC)
```

```
## [1] 990.9967
```

```
print(res.fb$BIC)
```

```
## [1] 1285.598
```

```
print(res.fc$BIC)
```

```
## [1] 1584.036
```

```
print(res.fd$BIC)
```

```
## [1] 2186.133
```

High dimensional mixture regression using BIC

```r
set.seed(111)
bet1=bet2=bet3=bet4=rep(0,81)
bet1[2:11]=sign(runif(10,-1,1))*runif(10,2,5)
bet2[12:21]=sign(runif(10,-1,1))*runif(10,2,5)
bet=rbind(bet1,bet2)
tmp_list = simu_func(beta=bet,n=200)
x=tmp_list$x
y=tmp_list$y
print(paste("Dimension of matrix (x):", dim(x)[1], ', ', dim(x)[2],sep='') )
```

```
## [1] "Dimension of matrix (x):200, 80"
```

```r
hx=x;hy=y
```

```r
res.h = MLM(ml.method = 'hrmr',x=hx,y=hy,nc=2)
bic1 = res.h$BIC
print(bic1)
```

```
## [1] 5395.74
```

```r
# User can test setting of the component number
bic2 = MLM(ml.method = 'hrmr',x=hx,y=hy,nc=3)$BIC
bic3 = MLM(ml.method = 'hrmr',x=hx,y=hy,nc=4)$BIC
bic4 = MLM(ml.method = 'hrmr',x=hx,y=hy,nc=5)$BIC
```

Cross validation based order selection

For high dimensional predictors, we also provide cross validation for selection of K.A large K will tend to overfit the data with more complex model of higher variance, while smaller K might select a simpler model with larger bias. Take a 5-fold cross validation as an example. For given K, at each repetition, 80% samples are used for training to obtain the mixture regression parameters. Then, for any sample from the rest of the testing data, it is first assigned to a cluster by maximum posterior probability; then a prediction of $\hat{y}$ could be made based on the model parameters given x. Using the testing data, we could decide how to balance the trade-off between bias and variance. To evaluate how the estimated model under K explains the testing data, we could calculate the root-meansquare-error between the observed y and the predicted $\hat{y}$, or Pearson correlation between the two. By repeating this procedure for multiple times, a more robust and stable evaluation of the choice of K should be derived based on the summarized RMSE or Pearson correlations.

The output of cross validation function, *MLM_cv*, is the average correlation or RMSE on the testing data. Higher correlation and lower RMSE indicate a better model with good bias variance tradeoff.

```r
CV1 = MLM_cv(x=hx, y=hy, nc=2)
```

```
## [1] "Fold 1 done."
## [1] "Fold 2 done."
## [1] "Fold 3 done."
## [1] "Fold 4 done."
## [1] "Fold 5 done."
```

```r
print(CV1$ycor)
```

```
## [1] 0.7849871
```

```r
print(CV1$RMSE)
```

```
## [1] 41.58089
```

```r
# User can test setting of the component number
CV2 = MLM_cv(x=hx, y=hy, nc=3)
CV3 = MLM_cv(x=hx, y=hy, nc=4)
CV4 = MLM_cv(x=hx, y=hy, nc=5)
```

S3. Real Data Application

S3.1 Colon adenocarcinoma disease application

Colon adenocarcinoma is known as a heterogeneous disease with different molecular subtypes. We demonstrate the usage of o*RobMixReg*on collected expression of a few genes and the methylation profiles for some of their CpG sites from the Cancer Genome Atlas (TCGA) cohort.

```r
data(colon_data)
x1 = colon_data$x1
y1 = colon_data$y1
res1 = MLM(ml.method='rlr', x=x1, y=y1)
inds_in = which(res1$cluMem == 1)
par(mfrow=c(1,1),mar = c(5, 8, 2, 8))
compPlot(type='rlr', x=x1,y=y1,nc=1, inds_in=inds_in)
```

```r
data(colon_data)
x2 = colon_data$x2
y2 = colon_data$y2
res.b = MLM(ml.method='fmr', x = x2, y = y2,
            b.formulaList = list(formula(y ~ x),formula(y ~ 1)))
par(mar = c(5, 8, 2, 8))
compPlot(type='mr', res=res.b$res)
```

```r
data(colon_data)
gene_expr = colon_data$x3
methylation = colon_data$y3
res.colon = MLM(ml.method="rmr", rmr.method='cat', x=gene_expr, y=methylation)
inds_in=which(res.colon$cluMem != -1)
par(mfrow=c(1,2),mar = c(4, 4, 6, 2))
compPlot(type='rlr', x=gene_expr,y=methylation,nc=2, inds_in=inds_in)
```

We fitted the data using CAT, the two colored line represented the two subgroups of patients.

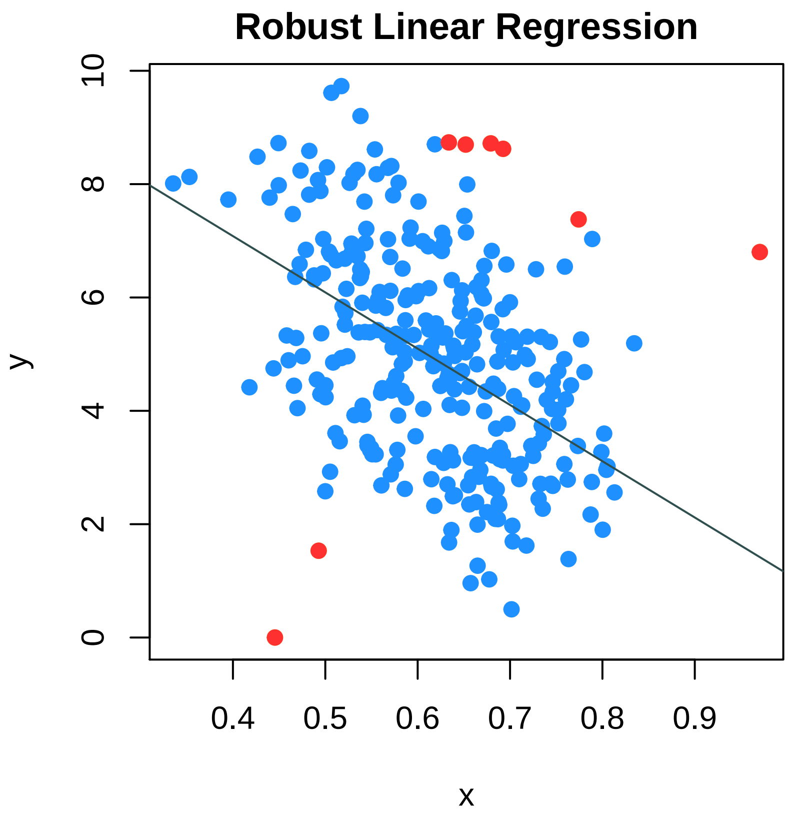


Figure 5: Colon data robust linear regression

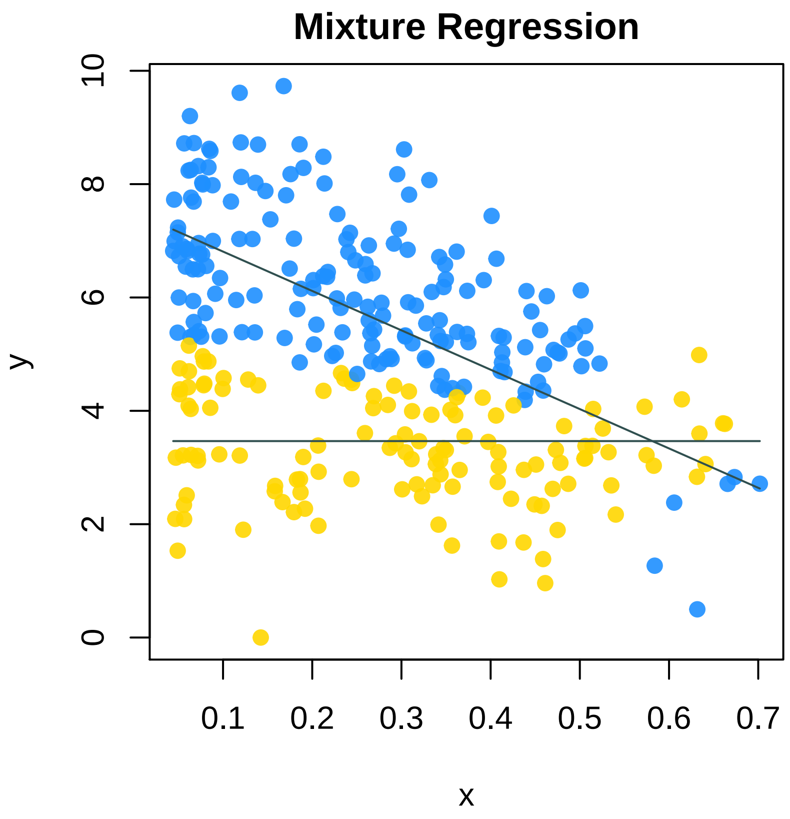


Figure 6: Colon cancer data flexible mixture regression

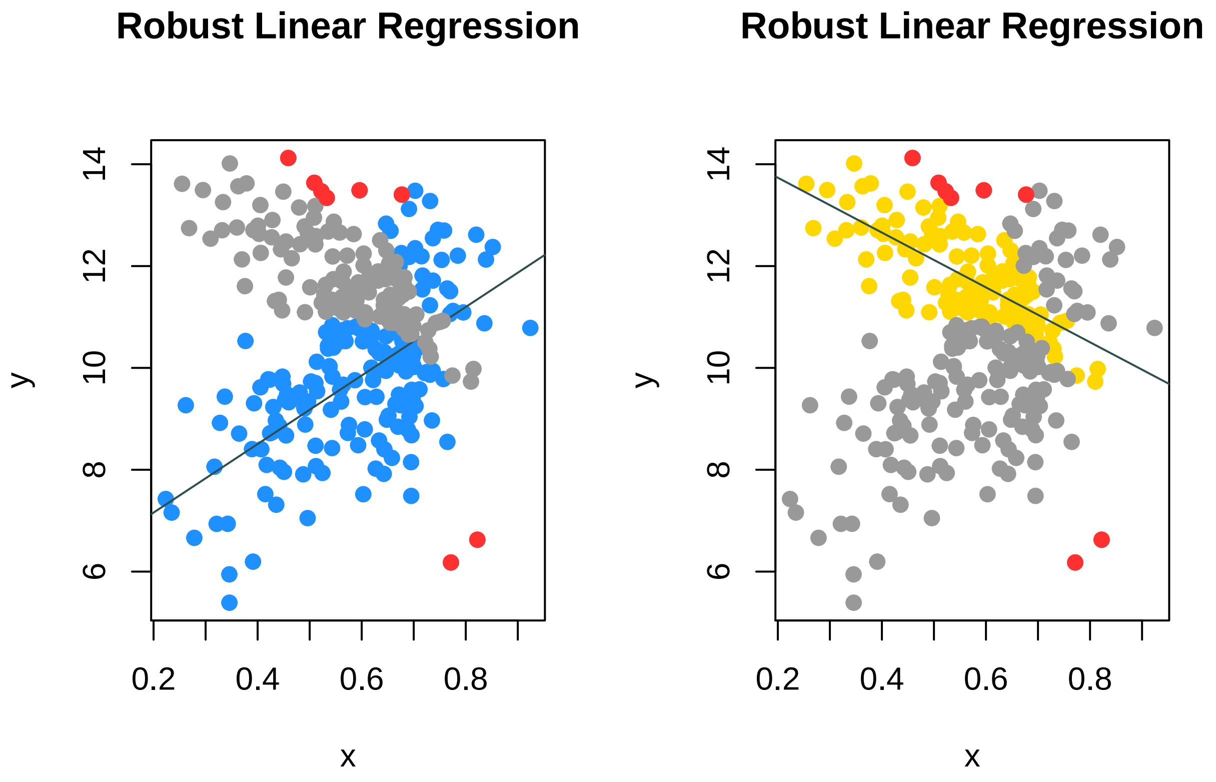


Figure 7: Colon cancer data robust mixture regression

S3.2 Subspace clustering on high dimension dataset - CCLE application

We collected gene expression data of 470 cell lines as well as the cell lines' sensitivity score (AUCC score) for 'AEW541' drug from the Cancer Cell Line Encyclopedia (CCLE) dataset. We demonstrate the usage of our package on this high dimensioanl dataset.

```r
data(CCLE_data)
x = CCLE_data$X
dim(x)
```

```
## [1] 490 500
```

```r
rr = paste('row_', rownames(x), sep='')
cc = paste('col_', colnames(x), sep='')
rownames(x) = rr
colnames(x) = cc

y = CCLE_data$Y
y[is.na(y)] = 0
names(y) = rownames(x)

# suppress the output of warning informaiton
sink("nul") ; CCLE_res=MLM(ml.method="hrmr",x=x,y=y,nit=1,nc=2,max_iter=10); sink()
CCLE_coffs1=CCLE_res$coff

# plot the coefficient matrix
CCLE_res2 = list()
# just print rows (genes) whose coefficient is n
CCLE_res2$coff = CCLE_coffs1[which(apply(CCLE_coffs1,1,sum)!=0),]
compPlot(type='block', res=CCLE_res2)
```

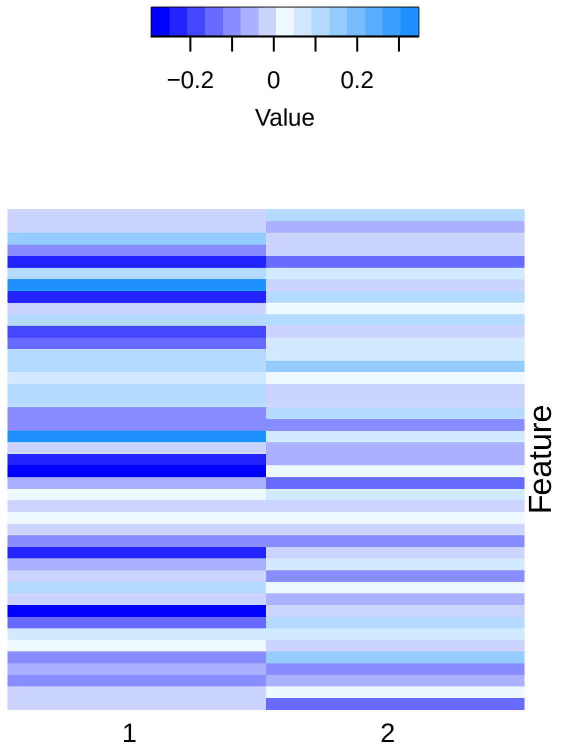


Figure 8: High dimension space feature robust mixture regression result