rebmix: Finite Mixture Modeling, Clustering & Classification

Marko Nagode, Branislav Panić, Jernej Klemenc & Simon Oman

October 19, 2026

Abstract

The **rebmix** package provides R functions for random univariate and multivariate finite mixture model generation, estimation, clustering, latent class analysis and classification. Variables can be continuous, discrete, independent or dependent and may follow normal, lognormal, Weibull, gamma, binomial, Poisson, Dirac or von Mises parametric families.

1 Introduction

To cite the REBMIX algorithm please refer to (Nagode and Fajdiga, 2011a,b; Nagode, 2015, 2018; Panić et al., 2020a,b,c). For theoretical backgrounds please upload `http://doi.org/10.5963/JA00302001`.

2 What's new in version `2.17.2`

Version `2.17.2` introduces minor bug fixes within R code.

3 Previous versions

Version `2.17.1`, several improvements have been introduced. Additional methods for accelerating the EM algorithm have been implemented, including linear and quadratic acceleration schemes from (Varadhan and Roland, 2008). These can be enabled by setting the `acceleration` slot of the `EM.control` class to the appropriate value. For the linear acceleration scheme, `acceleration` should be set to one of `"stem1"`, `"stem2"`, or `"stem3"`. For the quadratic acceleration scheme, it should be set to `"square1"`, `"square2"`, or `"square3"`. The difference between `"stem1"`, `"stem2"`, and `"stem3"` lies in the estimation of the acceleration parameter α, as described in (Varadhan and Roland, 2008). The same distinction applies to `"square1"`, `"square2"`, and `"square3"`. We have also added the options `"SEM"`, `"SEM-EM"`, and `"ECM-EM"` for the `variant` slot of the `EM.control` class. The `"SEM"` option adds the SEM variant of the EM algorithm described in (Biernacki et al., 2003). The `"SEM-EM"` and `"ECM-EM"` options represent combinations of SEM with EM and ECM with EM, respectively. In these cases, either SEM or ECM is first run until convergence, after which the standard EM algorithm is executed until a local optimum of the log-likelihood is reached. Additionally, we have introduced two new slots to the `EM.control` class. The first slot, `likelihood.tolerance.check`, accepts one of `"absolute"`, `"normalised"`, or `"percentage"`. When set to `"absolute"`, convergence of the log likelihood is assessed as

$$|\log L^{t+1} - \log L^t| \leq \epsilon, \tag{1}$$

where $\log L^{t+1}$ is the new log likelihood value, $\log L^t$ is the previous log likelihood value, and ϵ is the tolerance level.

When `likelihood.tolerance.check` is set to `"normalised"`, convergence is assessed as

$$\frac{|\log L^{t+1} - \log L^t|}{n} \leq \epsilon, \tag{2}$$

where $\log L^{t+1}$ is the new log likelihood value, $\log L^t$ is the previous log likelihood value, n is the number of observations, and ϵ is the tolerance level.

Finally, when `likelihood.tolerance.check` is set to `"percentage"`, convergence is assessed as

$$\frac{|\log L^{t+1} - \log L^t|}{|\log L^{t+1}|} \leq \epsilon, \tag{3}$$

where $\log L^{t+1}$ is the new log likelihood value, $\log L^t$ is the previous log likelihood value, and ϵ is the tolerance level. The second newly introduced slot is `likelihood.estimation.rule`. With this slot, we have implemented Aitken-accelerated estimation of the log likelihood value (McLachlan and Peel, 2000), which can further speed up the EM algorithm. The `likelihood.estimation.rule` accepts one of `"standard"`, `"aitken-bohning"`, `"aitken-lindsay"`, or `"aitken-nicholas"`. The option `"standard"` represents the original estimate of the log likelihood value, while `"aitken-bohning"`, `"aitken-lindsay"`, and `"aitken-nicholas"` all use Aitken-accelerated estimates. The main difference lies in which value is used for the previous estimate of the log likelihood, as described in (McNicholas et al., 2010). To support these changes, much of the old EM estimation code has been refactored, and minor bug fixes have been applied. Finally, we improved the C++ code for rough and enhanced Weibull and gamma parameter estimation, and added a new C++ function `BayesWeibullParameters()`.

Version `2.16.0` introduces handling of warnings and errors. The user can monitor in R code the problems that may occur in C++ code. Some minor changes and bug fixes have been made in C++ and R. The rough estimation procedure has been thoroughly analysed and improved. An additional argument `Mode` has been added to the method `REBMIX`. The detection of the global mode can now be performed in different ways, which can be advantageous when handling different types of datasets.

Version `2.15.0` introduces several auxiliary methods and some improvements to the core methods of the package. Two new auxiliary methods have been added primarily for dealing with digital images and other datasets of spatial interest that are initially processed by flattening or similarly removing spatial information. The newly added `"labelmoments"` method estimates spatial moments for clustering solutions obtained using either the `RCLRMIX` or the `mapclusters` method. Of course, the output of the `RCLRMIX` method or the `mapclusters` method must first be converted into a two-dimensional spatial matrix or a three-dimensional spatial matrix. Then the spatial moments are estimated similarly to widely used image moments Liao and Pawlak (1996). The `"labelmoments"` method also estimates the adjacency matrix between different labels in clustering using already estimated spatial moments. See the manual for more information. The second added method, `"mergelabels"`, uses the spectral clustering described in Ng et al. (2001) to propose another clustering solution that merges the most similar labels from the previous clustering solution. `"mergelabels"` accepts two adjacency matrices. The first one can be created using the method `"labelmoments"` for spatial similarity, and the second can be, for example, the similarity matrix of the different mixture components. Therefore, version `2.15.0` also introduces an improvement to the class `RCLRMIX`, which now also has a slot `A` representing an adjacency matrix (i.e., a similarity matrix) between different mixture components. The adjacency matrix is computed according to the given argument `Rule` in the method `RCLRMIX`. Finally, the C++ functions `CombineComponentsEntropy()` and `CombineComponentsDemp()` are improved and further tested.

Version `2.14.2` introduces several important improvements to the package. Estimation of mixture parameters can now be performed using the object of class `"Histogram"`. Objects of class `"Histogram"` represent the data with bins and bin counts. The object `"Histogram"` is best suited for discrete datasets, such as digital images. The image resolution can be quite high, but there is a finite number of realizations of the image pixel intensity. Often the number of realizations is much smaller than the number of observations. Therefore, it is more useful to represent the original dataset as an ordered pair $(\bar{y}_j, k_j)$, where $\bar{y}_j$ is the jth possible realization and k_j is its count. The continuous datasets can also be shrunk, but some bias may occur. When the number of observations is large, the bias is usually small and vice versa, and when time and memory efficiency is desired, shrinking the dataset is recommended. The original dataset (object of class `data.frame`) is transformed with the newly added methods `"chistogram"`, `"fhistogram"` into a `"Histogram"`. The method `"fhistogram"` is memory intensive, but very fast. It is best suited for one-dimensional datasets. Since it precomputes all possible realizations, even if some of them are not in the dataset, the argument `shrink= TRUE` should be added. The method `"chistogram"` is slower, but memory efficient. When a new realization y_j is found, it is

appended. The two methods "chistogram" and "fhistogram" can be chained in a for loop so that multiple datasets can be shrunk into one "Histogram" object. This can be used when, for example, multiple images are to be loaded. Also, if for example a large dataset is to be processed, only parts of it can be loaded at a time to avoid large memory overhead.

The mapclusters method is added for maximum a posteriori estimation of cluster membership. This method was added to complement the RCLRMIX method. The RCLRMIX method performs clustering and cluster membership prediction as well as cluster merging using different estimators. For large datasets, this can be very time consuming. Using objects of class "histogram" speeds up the RCLRMIX method enormously, but the cluster membership is only determined of the realizations $\bar{y}_j$ and the order is often not preserved either. Thus, we can use the mapclusters method to get the cluster membership on the original dataset of the object of the "data.frame" class.

Finally, all density estimation methods ("demix", "dfmix", "pemix", "pfmix") are modified to include estimation for objects of class "Histogram".

Version 2.14.0 introduces three parametric variant of the Gumbel parametric family

$$f(y) = \frac{1}{\sigma} \exp\left(\frac{y-\mu}{\xi\sigma} - \exp\left(\frac{y-\mu}{\xi\sigma}\right)\right), \tag{4}$$

where $\xi \in \{-1, \mathrm{NA}, 1\}$. If initially ξ is set to NA, the package estimates ξ. Otherwise the estimated ξ equals initial ξ. The package was extended to handle three parameter parametric families. Class "EMMIX.Theta" and method EMMIX are added. They enable parameter estimation for all parametric families available in the package by using the EM algorithm only.

Version 2.13.1 introduces Gumbel probability density

$$f(y) = \frac{1}{\sigma} \exp\left(-\frac{y-\mu}{\sigma} - \exp\left(-\frac{y-\mu}{\sigma}\right)\right). \tag{5}$$

The Gumbel mixture models can now be estimated. In the mixture model the parameters μ and σ are estimated for all components. The corresponding equations for the REBMIX and the EM algorithms have been derived. The EM algorithm has been implemented for all other parametric family types (normal, lognormal, Weibull, gamma, binomial, Poisson and von Mises parametric families), too. Therefore the EM algorithm can now be used with all parametric family types. Additionally, the histogram based EM algorithm has been added to speed up the calculations when the datasets contain large numbers of observations. Finally, the package has been debugged further and some core functions have been improved.

Version 2.12.0 introduces the Knuth algorithm (Knuth, 2019) as an effective way of optimal number of bins search. This affects the time efficiency of the REBMIX method considerably. The user can now enter different numbers of bins for different random variables. Two accompanied methods are added optbins and bins. The C++ code is optimized regarding memory allocation and efficiency. The R code is debugged and further improved regarding wrong user input messaging. Further debugging has been done. Outlier detection has been simplified. The REBMIX method now delivers some more components, but identifies also components with very low probability of occurrence, which is important regarding our future plans. The plot method is improved. The RCLRMIX method has been debugged and improved. The same holds for the REBMIX method.

Version 2.11.0 introduces the Expectation-Maximization (EM) algorithm for the improved estimation of Gaussian mixture model parameters (with diagonal and unrestriced covariance matrices). Here the REBMIX algorithm is used to assess the initial parameters of the EM algorithm. Two different variants of the EM algorithm are implemented, namely the original EM algorithm from (Dempster et al., 1977) and a k-means like variant of the EM algorithm (Classification EM) as described in (Celeux and Govaert, 1992). As the REBMIX algorithm estimates a wide range of parameters for the Gaussian mixture model for different numbers of components, three different strategies, named **exhaustive**, **best** and **single**, have been implemented. The **exhaustive** strategy is used to run the EM algorithm (or variant) on each solution of Gaussian mixture model parameters provided by the REBMIX algorithm. The **best** strategy utilizes a voting scheme for the estimated parameters from the REBMIX algorithm and runs the EM algorithm only on selected optimal parameters. The best candidates are

chosen based on the value of the likelihood function (the highest one) for each number of components c from a minimum specified `cmin` to a maximum specified `cmax`. The **single** strategy is useful when the single value of, for example, the number of bins in histogram preprocessing is supplied as input for the REBMIX algorithm. Otherwise, when multiple numbers of bins k are supplied, this strategy is the same as the **exhaustive** strategy. To tackle the slow linear convergence of the EM algorithm, simple acceleration methods are implemented, which can be controlled with parameter `acceleration` and `acceleration.multiplier`. The increment of the EM algorithm in each iteration can be written as

$$\Delta\boldsymbol{\Theta} = \boldsymbol{\Theta}^{(i+1)} - \boldsymbol{\Theta}^{(i)} \tag{6}$$

Instead of using a standard EM increment $\Delta\boldsymbol{\Theta}$ to reduce the number of iterations needed for the EM algorithm, this increment can be multiplied with some multiplier a_{EM}, which is referred to as `acceleration.multiplier`. Therefore the update in each EM iteration now becomes

$$\boldsymbol{\Theta}^{i+1} = \boldsymbol{\Theta}^{(i)} + a_{\mathrm{EM}}\Delta\boldsymbol{\Theta} \tag{7}$$

The safe range for the a_{EM} multiplier lies between 1.0 and 2.0, where 1.0 gives a standard EM increment and 2.0 doubles the EM increment. However, this does not necessarily mean that multiplication by a value of 2.0 will double the speed of the EM algorithm (i.e. by reducing the required number of iterations by 2). Here, 1.5 is a safe value which mostly speeds up the EM algorithm whilst retaining good results for the estimated parameters. A value of 1.9 can significantly speed up the estimation process, yet it can also deteriorate the quality of the resulting estimated parameters. Therefore, the value of the multiplicator needs to be set carefully. This value is set with `acceleration.multiplier` parameter. The other parameter `acceleration` controls how the a_{EM} multiplier is handled and can be one of **fixed**, **line** and **golden**. Selecting the **fixed** option means that the a_{EM} multiplier is specified via the `acceleration.multiplier` parameter and for each iteration of the EM algorithm the increment is increased by a specified value of a_{EM}. The **line** and **golden** options perform a, line and golden search (respectively) for the optimal value of a_{EM} for which the highest increase in the likelihood function of each EM iteration is achieved.

EM handling is carried out using the newly introduced class `"EM.Control"`. Classes `"REBMIX"` and `"REBMVNORM"` and its signature method `REBMIX` now accept the `"EM.Control"` object via the argument called `"EMcontrol"`. The class `EM.Control` has the same name convection for slots as the input argument `EMcontrol` (`strategy`, `variant`, `acceleration`, `tolerance`, `acceleration.multiplier` and `maximum.iterations`) as well as all accessor functions with the same name convention as `a.`*`slot name`* and setter function `a.`*`slot name`*`<-`.

Methods `Zp` and `coef` have been replaced by `a.Zp`, `a.theta1.all` and `a.theta2.all` getters. All slots can be accessed via accessors. Their names are generally composed of `a.` followed by the slot name and are used to read the slots. Class `"RNGMIX.Theta"` has been added to simplify random finite mixture model generation. Method `show` has been added for `"RCLS.chunk"` class. The minimum number of components `cmin` was added to `REBMIX` arguments and to the `"REBMIX"` class. The `"Parzen window"` preprocessing has been renamed to more commonly known `"kernel density estimation"`. Rough parameter estimation for binomial and Poisson parametric families has also been improved and the package is now broadened to latent class analysis in version 2.10.3. Method `split` has been improved and examples for its proper use are added.

GCC 8.1 notes and warnings in C++ functions have been eliminated in version 2.10.2. Cholesky decomposition is now used to calculate the logarithm of the determinant and inverse of variance-covariance matrices instead of LU decomposition. Special attention has been paid to resolving numerical problems related to high dimensional datasets.

Version 2.10.1 is the further debugged version of 2.10.0. Large `K` in combination with large dimension d can lead to histograms with numerous nonempty bins v. In order to restrain v, the well known RootN rule (Velleman, 1976) may intuitively be extended to multidimensions

$$v_{\max} = \frac{1+d}{d} n^{\frac{d}{1+d}}. \tag{8}$$

If $d = \infty$, then $v_{\max} = n$. If $d = 1$, then $v_{\max} = 2\sqrt{n}$. Minor debugging and function improvements have also been carried out in version 2.10.0. The acceleration rate is now progressively increasing.

Each time the inner loop starts, the counter I_2 (see Nagode, 2015, for details) is initiated and constant

$$A = \left.\frac{1 - a_{\mathrm{r}}}{a_{\mathrm{r}}(D_l w_l - D_{\min})}\right|_{I_2=1} \tag{9}$$

is calculated. The acceleration rate a_{r} at $I_2 = 1$ always equals the value stored in the input argument `ar`. Otherwise

$$a_{\mathrm{r}} = \left.\frac{1}{A(D_l w_l - D_{\min}) + 1}\right|_{I_2>1}. \tag{10}$$

The Newton-Raphson root finding in C++ functions was improved in version 2.9.3. This affects only Weibull, gamma and von Mises parametric families. A circular von Mises parametric family has been added and further debugging carried out in version 2.9.2. Version 2.9.1 is a further debugged version 2.8.4. The R code has been extended and rewritten in S4 class system. The background C code has also been extended and rewritten as object-oriented C++ code. The package can now more easily be extended to other parametric families. Multivariate normal mixtures with unrestricted variance-covariance matrices have been added. Clustering has also been added and classification improved.

4 Examples

To illustrate the use of the REBMIX algorithm, univariate and multivariate datasets are considered. The **rebmix** is loaded and the prompt before starting new page is set to `TRUE`.

```
R> library(rebmix)
R> devAskNewPage(ask = TRUE)
```

4.1 Gamma datasets

Three gamma mixtures are considered (Wiper et al., 2001). The first has four well-separated components with means 2, 4, 6 and 8, respectively

$$\begin{array}{lll} \theta_1 = 1/100 & \beta_1 = 200 & n_1 = 100 \\ \theta_2 = 1/100 & \beta_2 = 400 & n_2 = 100 \\ \theta_3 = 1/100 & \beta_3 = 600 & n_3 = 100 \\ \theta_4 = 1/100 & \beta_4 = 800 & n_4 = 100. \end{array}$$

The second has equal means but different variances and weights

$$\begin{array}{lll} \theta_1 = 1/27 & \beta_1 = 9 & n_1 = 40 \\ \theta_2 = 1/270 & \beta_2 = 90 & n_2 = 360. \end{array}$$

The third is a mixture of a rather diffuse component with mean 6 and two lower weighted components with smaller variances and means of 2 and 10, respectively

$$\begin{array}{lll} \theta_1 = 1/20 & \beta_1 = 40 & n_1 = 80 \\ \theta_2 = 1 & \beta_2 = 6 & n_2 = 240 \\ \theta_3 = 1/20 & \beta_3 = 200 & n_3 = 80. \end{array}$$

4.1.1 Finite mixture generation

```
R> n <- c(100, 100, 100, 100)
R> Theta <- new("RNGMIX.Theta", c = 4, pdf = "gamma")
R> a.theta1(Theta) <- rep(1/100, 4)
R> a.theta2(Theta) <- c(200, 400, 600, 800)
R> gamma1 <- RNGMIX(Dataset.name = "gamma1", n = n, Theta = a.Theta(Theta))
R> n <- c(40, 360)
R> Theta <- new("RNGMIX.Theta", c = 2, pdf = "gamma")
```

```
R> a.theta1(Theta) <- c(1/27, 1/270)
R> a.theta2(Theta) <- c(9, 90)
R> gamma2 <- RNGMIX(Dataset.name = "gamma2", n = n, Theta = a.Theta(Theta))
R> n <- c(80, 240, 80)
R> Theta <- new("RNGMIX.Theta", c = 3, pdf = "gamma")
R> a.theta1(Theta) <- c(1/20, 1, 1/20)
R> a.theta2(Theta) <- c(40, 6, 200)
R> gamma3 <- RNGMIX(Dataset.name = "gamma3", rseed = -4, n = n,
+      Theta = a.Theta(Theta))
```

4.1.2 Finite mixture estimation

```
R> gamma1est <- REBMIX(Dataset = a.Dataset(gamma1), Preprocessing = "kernel density estimation",
+      cmax = 8, Criterion = "BIC", pdf = "gamma")
R> gamma2est <- REBMIX(Dataset = a.Dataset(gamma2), Preprocessing = "histogram",
+      cmax = 8, Criterion = "BIC", pdf = "gamma")
R> gamma3est <- REBMIX(Dataset = a.Dataset(gamma3), Preprocessing = "histogram",
+      cmax = 8, Criterion = "BIC", pdf = "gamma", K = 23:27)
```

4.1.3 Plot method

```
R> plot(gamma3est, pos = 1, what = c("pdf", "marginal cdf"), ncol = 2,
+      npts = 1000, family = "sans")
```

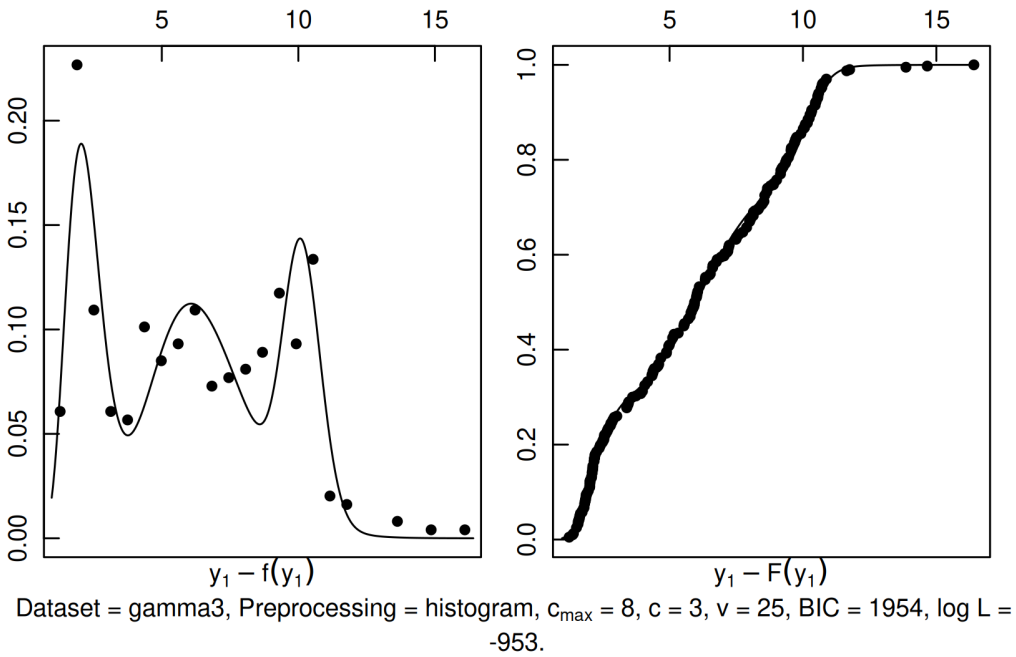

Figure 1: Gamma 3 dataset. Empirical density (circles) and predictive gamma mixture density in black solid line.

4.1.4 Summary, a.theta1.all and a.theta2.all methods

```
R> summary(gamma2est)

  Dataset Preprocessing Criterion c v/k    IC logL M
1  gamma2     histogram       BIC 2  40 -1348  689 5
Maximum logL = 689 at pos = 1.
```

```
R> a.theta1.all(gamma1est, pos = 1)

            [,1]
theta1.1 0.01026
theta1.2 0.00930
theta1.3 0.01150
theta1.4 0.00868

R> a.theta2.all(gamma1est, pos = 1)

         [,1]
theta2.1  195
theta2.2  433
theta2.3  521
theta2.4  920
```

4.1.5 Bootstrap methods

```
R> gamma3boot <- boot(x = gamma3est, pos = 1, Bootstrap = "p", B = 10)
R> gamma3boot

An object of class "REBMIX.boot"
Slot "c":
 [1] 4 3 3 3 3 3 3 3 3 3
Slot "c.se":
[1] 0.316
Slot "c.cv":
[1] 0.102
Slot "c.mode":
[1] 3
Slot "c.prob":
[1] 0.9

R> summary(gamma3boot)

w.cv
[1] 0.150 0.169 0.220
             [,1]
theta1.1.cv 0.663
theta1.2.cv 1.076
theta1.3.cv 0.463
            [,1]
theta2.1.cv 2.05
theta2.2.cv 0.68
theta2.3.cv 1.65
            [,1]
theta3.1.cv   NA
theta3.2.cv   NA
theta3.3.cv   NA
Mode probability = 0.9 at c = 3 components.
```

4.1.6 Estimation of the mixture parameters using the Best REBMIX&EM strategy

```
R> EM <- new("EM.Control", strategy = "best", variant = "EM", acceleration = "fixed",
+      acceleration.multiplier = 1, tolerance = 1e-04, maximum.iterations = 1000,
```

```
+      K = 0)
R> gamma1est.em <- REBMIX(Dataset = a.Dataset(gamma1), Preprocessing = "kernel density estimatio
+      cmax = 8, Criterion = "BIC", pdf = "gamma", EMcontrol = EM)
R> gamma2est.em <- REBMIX(Dataset = a.Dataset(gamma2), Preprocessing = "histogram",
+      cmax = 8, Criterion = "BIC", pdf = "gamma", EMcontrol = EM)
R> gamma3est.em <- REBMIX(Dataset = a.Dataset(gamma3), Preprocessing = "histogram",
+      cmax = 8, Criterion = "BIC", pdf = "gamma", K = 23:27, EMcontrol = EM)
R> summary(gamma1est.em)

  Dataset             Preprocessing Criterion c v/k   IC logL  M
1  gamma1 kernel density estimation       BIC 4  11 1054 -494 11
Maximum logL = -494 at pos = 1.

R> summary(gamma2est.em)

  Dataset Preprocessing Criterion c v/k    IC logL M
1  gamma2     histogram       BIC 2  40 -1354  692 5
Maximum logL = 692 at pos = 1.

R> summary(gamma3est.em)

  Dataset Preprocessing Criterion c v/k   IC logL M
1  gamma3     histogram       BIC 3  25 1926 -939 8
Maximum logL = -939 at pos = 1.
```

4.2 Poisson dataset

Dataset consists of $n = 600$ two dimensional observations obtained by generating data points separately from each of three Poisson distributions. The component dataset sizes and parameters, which are those studied in Ma et al. (2009), are displayed below

$$\begin{array}{ll} \boldsymbol{\theta}_1 = (3,2)^\top & n_1 = 200 \\ \boldsymbol{\theta}_2 = (9,10)^\top & n_2 = 200 \\ \boldsymbol{\theta}_3 = (15,16)^\top & n_3 = 200 \end{array}$$

For the dataset Ma et al. (2009) conduct 100 experiments by selecting different initial values of the mixing proportions. In all the cases, the adaptive gradient BYY learning algorithm leads to the correct model selection, i.e., finally allocating the correct number of Poissons for the dataset. In the meantime, it also results in an estimate for each parameter in the original or true Poisson mixture which generated the dataset. As the dataset of Ma et al. (2009) can not exactly be reproduced, 10 datasets are generated with random seeds r_{seed} ranging from -1 to -10.

4.2.1 Finite mixture generation

```
R> n <- c(200, 200, 200)
R> Theta <- new("RNGMIX.Theta", c = 3, pdf = rep("Poisson", 2))
R> a.theta1(Theta, 1) <- c(3, 2)
R> a.theta1(Theta, 2) <- c(9, 10)
R> a.theta1(Theta, 3) <- c(15, 16)
R> poisson <- RNGMIX(Dataset.name = paste("Poisson_", 1:10, sep = ""),
+      n = n, Theta = a.Theta(Theta))
```

4.2.2 Finite mixture estimation

```
R> poissonest <- REBMIX(Dataset = a.Dataset(poisson), Preprocessing = "histogram",
+      cmax = 10, Criterion = "MDL5", pdf = rep("Poisson", 2), K = 1)
```

4.2.3 Plot method

4.2.4 Clustering

4.2.5 Summary, a.theta1.all and a.theta2.all methods

```
R> summary(poissonest)

      Dataset Preprocessing Criterion c v/k   IC  logL M
1   Poisson_1     histogram      MDL5 3   1 7062 -3403 8
2   Poisson_2     histogram      MDL5 3   1 7116 -3430 8
3   Poisson_3     histogram      MDL5 3   1 7068 -3406 8
4   Poisson_4     histogram      MDL5 3   1 6992 -3368 8
5   Poisson_5     histogram      MDL5 3   1 6993 -3368 8
6   Poisson_6     histogram      MDL5 3   1 7052 -3398 8
7   Poisson_7     histogram      MDL5 2   1 7304 -3572 5
8   Poisson_8     histogram      MDL5 3   1 7123 -3434 8
9   Poisson_9     histogram      MDL5 3   1 7091 -3418 8
10 Poisson_10     histogram      MDL5 3   1 7042 -3393 8
Maximum logL = -3368 at pos = 4.

R> a.theta1.all(poissonest, pos = 1)

          [,1]  [,2]
theta1.1  3.55  2.59
theta1.2 14.35 15.06
theta1.3  7.89 10.09

R> a.theta2.all(poissonest, pos = 1)

         [,1] [,2]
theta2.1   NA   NA
theta2.2   NA   NA
theta2.3   NA   NA
```

4.2.6 Estimation of the mixture parameters using the Exhaustive REBMIX&EM strategy

```
R> EM <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "fixed", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> poissonest.em <- REBMIX(Dataset = a.Dataset(poisson), Preprocessing = "histogram",
+      cmax = 10, Criterion = "MDL5", pdf = rep("Poisson", 2), K = 1,
+      EMcontrol = EM)
R> summary(poissonest.em)

      Dataset Preprocessing Criterion c v/k   IC  logL M
1   Poisson_1     histogram      MDL5 3   1 6987 -3365 8
2   Poisson_2     histogram      MDL5 3   1 7039 -3392 8
3   Poisson_3     histogram      MDL5 3   1 6992 -3368 8
4   Poisson_4     histogram      MDL5 3   1 6974 -3359 8
5   Poisson_5     histogram      MDL5 3   1 6989 -3367 8
6   Poisson_6     histogram      MDL5 3   1 7034 -3389 8
7   Poisson_7     histogram      MDL5 2   1 7239 -3539 5
```

```
8   Poisson_8     histogram      MDL5 3   1 7031 -3387 8
9   Poisson_9     histogram      MDL5 3   1 7006 -3375 8
10 Poisson_10     histogram      MDL5 3   1 6994 -3369 8
Maximum logL = -3359 at pos = 4.
```

4.3 Multivariate normal dataset

The `mvnorm` dataset from (Panić et al., 2020c) consists of 250 observations drawn from the 5-component normal mixture.

4.3.1 Finite mixture generation

```
R> n <- c(50, 50, 50, 50, 50)
R> Theta <- new("RNGMVNORM.Theta", c = 5, d = 2)
R> a.theta1(Theta, 1) <- c(2.7, 3.7)
R> a.theta1(Theta, 2) <- c(5.7, 9.1)
R> a.theta1(Theta, 3) <- c(2, 9)
R> a.theta1(Theta, 4) <- c(9.5, 6.6)
R> a.theta1(Theta, 5) <- c(6.3, 0.6)
R> a.theta2(Theta, 1) <- c(0.9, -0.1, -0.1, 0.4)
R> a.theta2(Theta, 2) <- c(2.8, -1.3, -1.3, 1.5)
R> a.theta2(Theta, 3) <- c(0.1, 0, 0, 0.3)
R> a.theta2(Theta, 4) <- c(1.3, -0.4, -0.4, 0.4)
R> a.theta2(Theta, 5) <- c(0.5, 0.3, 0.3, 2.5)
R> mvnorm.simulated <- RNGMIX(model = "RNGMVNORM", Dataset.name = "mvnormdataset",
+      rseed = -1, n = n, Theta = a.Theta(Theta))
```

4.3.2 Finite mixture estimation

```
R> mvnormest <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 20, Criterion = "BIC")
```

4.3.3 Plot method

4.3.4 Clustering

4.3.5 Summary method of estimation

```
R> summary(mvnormest)
```

```
        Dataset Preprocessing Criterion c v/k   IC  logL  M
1 mvnormdataset     histogram       BIC 5   9 2205 -1022 29
Maximum logL = -1022 at pos = 1.
```

4.3.6 Summary method of clustering

```
R> summary(mvnormclu)
```

```
Number of clusters  1          2          3          4
From cluster        2          4          3          5
To cluster          1          2          1          3
Entropy             5.55e-16   8.16e-01   2.00e+00   6.47e+00
Entropy decrease    0.816      1.180      4.473      5.494
```

4.3.7 Estimation of the mixture parameters using the Single REBMIX&EM strategy

The strategy requires the preprocessing parameter `K` to be known. It can be estimated using the `optbins` method. To estimate the optimal parameter `K`, different rules are implemented. More information can be found in (Panić et al., 2020c).

```
R> EM <- new("EM.Control", strategy = "single", variant = "EM",
+      acceleration = "fixed", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> K <- optbins(Dataset = a.Dataset(mvnorm.simulated), Rule = "Knuth equal",
+      kmin = 2, kmax = 100)
R> mvnormest.em <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 20, K = K, Criterion = "BIC",
+      EMcontrol = EM)
R> summary(mvnormest.em)
```

```
        Dataset Preprocessing Criterion c v/k   IC  logL  M
1 mvnormdataset     histogram       BIC 4  11 2146 -1010 23
Maximum logL = -1010 at pos = 1.
```

4.3.8 Clustering with exhaustive REBMIX&ECM strategy and ICL criterion

```
R> CEM <- new("EM.Control", strategy = "exhaustive", variant = "ECM",
+      acceleration = "fixed", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> mvnormest.cem <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 10, Criterion = "ICL",
+      EMcontrol = CEM)
R> mvnorm.clu <- RCLRMIX(model = "RCLRMVNORM", x = mvnormest.cem)
```

4.3.9 Acceleration of the EM algorithm

Standard EM algorithm with fixed acceleration.multiplier of $a_{\mathtt{EM}} = 1.0$:

```
R> EM.normal <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "fixed", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> mvnormestest.em.normal <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.normal)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormestest.em.normal,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormestest.em.normal, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  698 . Value of BIC:  2125
```

Standard EM algorithm with fixed acceleration.multiplier of $a_{\mathtt{EM}} = 1.5$:

```
R> EM.fixed1.5 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "fixed", acceleration.multiplier = 1.5, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> mvnormest.em.fixed1.5 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.fixed1.5)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.fixed1.5,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.fixed1.5, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  644 . Value of BIC:  2125
```

Standard EM algorithm with line search for optimal increment $a_{\mathtt{EM}}$ in each iteration:

```
R> EM.line <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "line", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> mvnormest.em.line <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.line)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.line,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.line, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  637 . Value of BIC:  2125
```

Standard EM algorithm with golden search for optimal increment $a_{\mathtt{EM}}$ in each iteration:

```
R> EM.golden <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "golden", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 0)
R> mvnormest.em.golden <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.golden)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.golden,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.golden, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  4488 . Value of BIC:  2125
```

EM algorithm with linear acceleration scheme using α_1 estimate from Varadhan and Roland (2008):

```
R> EM.stem1 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "stem1", tolerance = 1e-04, maximum.iterations = 1000,
+      K = 0)
R> mvnormest.em.stem1 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.stem1)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.stem1,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.stem1, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  1464 . Value of BIC:  2125
```

EM algorithm with linear acceleration scheme using α_2 estimate from Varadhan and Roland (2008):

```
R> EM.stem2 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "stem2", tolerance = 1e-04, maximum.iterations = 1000,
+      K = 0)
R> mvnormest.em.stem2 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.stem2)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.stem2,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.stem2, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  1275 . Value of BIC:  2125
```

EM algorithm with linear acceleration scheme using α_3 estimate from Varadhan and Roland (2008):

```
R> EM.stem3 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "stem3", tolerance = 1e-04, maximum.iterations = 1000,
+      K = 0)
R> mvnormest.em.stem3 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.stem3)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.stem3,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.stem3, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  1119 . Value of BIC:  2125
```

EM algorithm with quadratic acceleration scheme using α_1 estimate from Varadhan and Roland (2008):

```
R> EM.square1 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "square1", tolerance = 1e-04, maximum.iterations = 1000,
+      K = 0)
R> mvnormest.em.square1 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.square1)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.square1,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.square1, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  1398 . Value of BIC:  2125
```

EM algorithm with quadratic acceleration scheme using α_2 estimate from Varadhan and Roland (2008):

```
R> EM.square2 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "square2", tolerance = 1e-04, maximum.iterations = 1000,
+      K = 0)
R> mvnormest.em.square2 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.square2)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.square2,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.square2, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  1215 . Value of BIC:  2125
```

EM algorithm with quadratic acceleration scheme using α_3 estimate from Varadhan and Roland (2008):

```
R> EM.square3 <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "square3", tolerance = 1e-04, maximum.iterations = 1000,
+      K = 0)
R> mvnormest.em.square3 <- REBMIX(model = "REBMVNORM", Dataset = a.Dataset(mvnorm.simulated),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM.square3)
R> cat("Total number of EM algorithm iterations: ", a.summary.EM(mvnormest.em.square3,
+      pos = 1, col.name = "total.iterations.nbr"), ". Value of BIC: ",
+      a.summary(mvnormest.em.square3, pos = 1, col.name = "IC"))
```

```
Total number of EM algorithm iterations:  1035 . Value of BIC:  2125
```

4.4 Multivariate `sensorlessdrive` dataset

These data are a result of the sensorless drive diagnosis procedure. The features are extracted from electric current drive signals (Bator et al., 2012). The main objective is the sensorless fault detection and classification.

```
R> data(sensorlessdrive)
R> set.seed(5)
R> Drive <- split(p = 0.75, Dataset = sensorlessdrive, class = 4)
```

4.4.1 Finite mixture estimation

```
R> driveest <- REBMIX(model = "REBMVNORM", Dataset = a.train(Drive),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC")
```

4.4.2 Classification

```
R> drivecla <- RCLSMIX(model = "RCLSMVNORM", x = list(driveest),
+      Dataset = a.test(Drive), Zt = a.Zt(Drive))
```

4.4.3 Show and summary methods

```
R> drivecla

An object of class "RCLSMVNORM"
Slot "CM":

        1    2    3    4    5    6    7    8    9   10   11
  1  1025    0    0    0    0  216    0    0   89    0    0
  2     0 1206    0    0    0    0    0    0   17  107    0
  3     0    0 1232   20   77    1    0    0    0    0    0
  4     0    0    9 1280   40    0    0    1    0    0    0
  5     0    0  103   29 1068    0    0  130    0    0    0
  6   145    0    5    0    0  972    0    0  208    0    0
  7     0    0    0    0    0    0 1330    0    0    0    0
  8     0    1   25   32  353    0    0  919    0    0    0
  9   157    8    0    0    1  468    0    0  694    2    0
  10    0  173    0    0    0    0    0    0    3 1154    0
  11    0    0    0    0    0    0    0    0    0    0 1330
Slot "Error":
[1] 0.165
Slot "Precision":
 [1] 0.771 0.907 0.926 0.962 0.803 0.731 1.000 0.691 0.522 0.868 1.000
Slot "Sensitivity":
 [1] 0.772 0.869 0.897 0.940 0.694 0.587 1.000 0.875 0.686 0.914 1.000
Slot "Specificity":
 [1] 1.000 1.004 1.003 1.002 1.016 1.025 1.000 0.979 0.977 0.995 1.000
Slot "Chunks":
[1] 1

R> summary(drivecla)

    Test Predictive Frequency
1      1          1      1025
2      2          1         0
3      3          1         0
```

4	4	1	0
5	5	1	0
6	6	1	145
7	7	1	0
8	8	1	0
9	9	1	157
10	10	1	0
11	11	1	0
12	1	2	0
13	2	2	1206
14	3	2	0
15	4	2	0
16	5	2	0
17	6	2	0
18	7	2	0
19	8	2	1
20	9	2	8
21	10	2	173
22	11	2	0
23	1	3	0
24	2	3	0
25	3	3	1232
26	4	3	9
27	5	3	103
28	6	3	5
29	7	3	0
30	8	3	25
31	9	3	0
32	10	3	0
33	11	3	0
34	1	4	0
35	2	4	0
36	3	4	20
37	4	4	1280
38	5	4	29
39	6	4	0
40	7	4	0
41	8	4	32
42	9	4	0
43	10	4	0
44	11	4	0
45	1	5	0
46	2	5	0
47	3	5	77
48	4	5	40
49	5	5	1068
50	6	5	0
51	7	5	0
52	8	5	353
53	9	5	1
54	10	5	0
55	11	5	0
56	1	6	216

57	2	6	0
58	3	6	1
59	4	6	0
60	5	6	0
61	6	6	972
62	7	6	0
63	8	6	0
64	9	6	468
65	10	6	0
66	11	6	0
67	1	7	0
68	2	7	0
69	3	7	0
70	4	7	0
71	5	7	0
72	6	7	0
73	7	7	1330
74	8	7	0
75	9	7	0
76	10	7	0
77	11	7	0
78	1	8	0
79	2	8	0
80	3	8	0
81	4	8	1
82	5	8	130
83	6	8	0
84	7	8	0
85	8	8	919
86	9	8	0
87	10	8	0
88	11	8	0
89	1	9	89
90	2	9	17
91	3	9	0
92	4	9	0
93	5	9	0
94	6	9	208
95	7	9	0
96	8	9	0
97	9	9	694
98	10	9	3
99	11	9	0
100	1	10	0
101	2	10	107
102	3	10	0
103	4	10	0
104	5	10	0
105	6	10	0
106	7	10	0
107	8	10	0
108	9	10	2
109	10	10	1154

```
110   11          10          0
111    1          11          0
112    2          11          0
113    3          11          0
114    4          11          0
115    5          11          0
116    6          11          0
117    7          11          0
118    8          11          0
119    9          11          0
120   10          11          0
121   11          11       1330
Error = 0.165.
```

4.4.4 Plot method

4.4.5 Estimation of using the histogram based EM algorithm

When dealing with large datasets it is useful to shrink datasets to reduce the computational overload of the EM algorithm. The histogram based EM algorithm can be used for this purpose. The dataset is discretized and the EM is computed with the binned data. Integer parameter `K` from the `EM.Control` object is used to set the shrinkage level. A smaller value means less containers while a larger value implicates more containers.

```
R> EM <- new("EM.Control", strategy = "exhaustive", variant = "EM",
+      acceleration = "fixed", acceleration.multiplier = 1, tolerance = 1e-04,
+      maximum.iterations = 1000, K = 300)
R> driveest <- REBMIX(model = "REBMVNORM", Dataset = a.train(Drive),
+      Preprocessing = "histogram", cmax = 15, Criterion = "BIC",
+      EMcontrol = EM)
R> drivecla <- RCLSMIX(model = "RCLSMVNORM", x = list(driveest),
+      Dataset = a.test(Drive), Zt = a.Zt(Drive))
R> summary(drivecla)
```

```
    Test Predictive Frequency
1      1          1      1025
2      2          1         0
3      3          1         0
4      4          1         0
5      5          1         0
6      6          1       145
7      7          1         0
8      8          1         0
9      9          1       157
10    10          1         0
11    11          1         0
12     1          2         0
13     2          2      1206
14     3          2         0
15     4          2         0
16     5          2         0
17     6          2         0
18     7          2         0
19     8          2         1
```

20	9	2	8
21	10	2	173
22	11	2	0
23	1	3	0
24	2	3	0
25	3	3	1232
26	4	3	9
27	5	3	103
28	6	3	5
29	7	3	0
30	8	3	25
31	9	3	0
32	10	3	0
33	11	3	0
34	1	4	0
35	2	4	0
36	3	4	20
37	4	4	1280
38	5	4	29
39	6	4	0
40	7	4	0
41	8	4	32
42	9	4	0
43	10	4	0
44	11	4	0
45	1	5	0
46	2	5	0
47	3	5	77
48	4	5	40
49	5	5	1068
50	6	5	0
51	7	5	0
52	8	5	353
53	9	5	1
54	10	5	0
55	11	5	0
56	1	6	216
57	2	6	0
58	3	6	1
59	4	6	0
60	5	6	0
61	6	6	972
62	7	6	0
63	8	6	0
64	9	6	468
65	10	6	0
66	11	6	0
67	1	7	0
68	2	7	0
69	3	7	0
70	4	7	0
71	5	7	0
72	6	7	0

73	7	7	1330
74	8	7	0
75	9	7	0
76	10	7	0
77	11	7	0
78	1	8	0
79	2	8	0
80	3	8	0
81	4	8	1
82	5	8	130
83	6	8	0
84	7	8	0
85	8	8	919
86	9	8	0
87	10	8	0
88	11	8	0
89	1	9	89
90	2	9	17
91	3	9	0
92	4	9	0
93	5	9	0
94	6	9	208
95	7	9	0
96	8	9	0
97	9	9	694
98	10	9	3
99	11	9	0
100	1	10	0
101	2	10	107
102	3	10	0
103	4	10	0
104	5	10	0
105	6	10	0
106	7	10	0
107	8	10	0
108	9	10	2
109	10	10	1154
110	11	10	0
111	1	11	0
112	2	11	0
113	3	11	0
114	4	11	0
115	5	11	0
116	6	11	0
117	7	11	0
118	8	11	0
119	9	11	0
120	10	11	0
121	11	11	1330

Error = 0.165.

4.5 Multivariate adult dataset

The adult dataset containing 48842 instances with 16 continuous, binary and discrete variables was extracted from the census bureau database Asuncion and Newman (2007). Extraction was done by Barry Becker from the 1994 census bureau database. The adult dataset is loaded, complete cases are extracted and levels are replaced with numbers.

```
R> data(adult)
R> adult <- adult[complete.cases(adult), ]
R> adult <- as.data.frame(data.matrix(adult))
```

Numbers of unique values for variables are determined and displayed.

```
R> cmax <- unlist(lapply(apply(adult[, c(-1, -16)], 2, unique),
+      length))
R> cmax
```

```
           Age      Workclass         Fnlwgt      Education  Education.Num
            74              7          26741             16             16
Marital.Status     Occupation   Relationship           Race            Sex
             7             14              6              5              2
  Capital.Gain   Capital.Loss Hours.Per.Week Native.Country
           121             97             96             41
```

The dataset is split into train and test subsets for the two incomes and the Type and Income columns are removed.

```
R> Adult <- split(p = list(type = 1, train = 2, test = 1), Dataset = adult,
+      class = 16)
```

4.5.1 Finite mixture estimation

Number of components, component weights and component parameters are estimated assuming that the variables are independent for the set of chunks $y_{1j}, y_{2j}, \ldots, y_{14j}$.

```
R> adultest <- list()
R> for (i in 1:14) {
+      adultest[[i]] <- REBMIX(Dataset = a.train(chunk(Adult, i)),
+          Preprocessing = "histogram", cmax = min(120, cmax[i]),
+          Criterion = "BIC", pdf = "Dirac", K = 1)
+  }
```

4.5.2 Classification

The class membership prediction is based upon the best first search algorithm.

```
R> adultcla <- BFSMIX(x = adultest, Dataset = a.test(Adult), Zt = a.Zt(Adult))
```

4.5.3 Show and summary methods

```
R> adultcla
```

```
An object of class "RCLSMIX"
Slot "CM":

        1     2
  1 10649   711
```

```
  2  1397  2303
Slot "Error":
[1] 0.14
Slot "Precision":
[1] 0.937 0.622
Slot "Sensitivity":
[1] 0.884 0.764
Slot "Specificity":
[1] 1.228 0.943
Slot "Chunks":
[1] 11 12  4  8  1
```

```
R> summary(adultcla)
```

```
  Test Predictive Frequency
1    1          1     10649
2    2          1      1397
3    1          2       711
4    2          2      2303
Error = 0.14.
```

4.5.4 Plot method

5 Summary

The users of the `rebmix` package are kindly encouraged to inform the authors about bugs and wishes.

References

A. Asuncion and D. J. Newman. Uci machine learning repository, 2007. URL `http://archive.ics.uci.edu/ml`.

M. Bator, A. Dicks, U. Mönks, and V. Lohweg. Feature extraction and reduction applied to sensorless drive diagnosis. 12 2012. doi: 10.13140/2.1.2421.5689.

C. Biernacki, G. Celeux, and G. Govaert. Choosing starting values for the em algorithm for getting the highest likelihood in multivariate gaussian mixture models. *Computational Statistics & Data Analysis*, 41(3):561–575, 2003. ISSN 0167-9473. doi: https://doi.org/10.1016/S0167-9473(02)00163-9. Recent Developments in Mixture Model.

G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational Statistics & Data Analysis*, 14(3):315–332, 1992. doi: 10.1016/0167-9473(92)90042-E.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38, 1977. URL `https://www.jstor.org/stable/2984875`.

K. H. Knuth. Optimal data-based binning for histograms and histogram-based probability density models. *Digital Signal Processing*, 95:102581, 2019. doi: 10.1016/j.dsp.2019.102581.

S. X. Liao and M. Pawlak. On image analysis by moments. *IEEE Transactions on Pattern analysis and machine intelligence*, 18(3):254–266, 1996. doi: 10.1109/34.485554.

J. Ma, J. Liu, and Z. Ren. Parameter estimation of poisson mixture with automated model selection through byy harmony learning. *Pattern Recognition*, 42(11):2659–2670, 2009. doi: 10.1016/j.patcog.2009.03.029.

G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, New York, 2000.

P. D. McNicholas, T. B. Murphy, A. McDaid, and D. Frost. Serial and parallel implementations of model-based clustering via parsimonious gaussian mixture models. *Computational Statistics & Data Analysis*, 54(3):711–723, 2010. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2009.02.011. Second Special Issue on Statistical Algorithms and Software.

M. Nagode. Finite mixture modeling via rebmix. *Journal of Algorithms and Optimization*, 3(2):14–28, 2015. URL `https://repozitorij.uni-lj.si/Dokument.php?id=127674&lang=eng`.

M. Nagode. Multivariate normal mixture modeling, clustering and classification with the rebmix package. *ArXiv e-prints*, Jan. 2018.

M. Nagode and M. Fajdiga. The rebmix algorithm for the univariate finite mixture estimation. *Communications in Statistics - Theory and Methods*, 40(5):876–892, 2011a. doi: 10.1080/03610920903480890.

M. Nagode and M. Fajdiga. The rebmix algorithm for the multivariate finite mixture estimation. *Communications in Statistics - Theory and Methods*, 40(11):2022–2034, 2011b. doi: 10.1080/03610921003725788.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. volume 14. MIT Press, 2001. URL `https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf`.

B. Panić, J. Klemenc, and M. Nagode. Improved initialization of the em algorithm for mixture model parameter estimation. *Mathematics*, 8(3):373, 2020a. doi: 10.3390/math8030373. URL `https://www.mdpi.com/2227-7390/8/3/373/htm`.

B. Panić, J. Klemenc, and M. Nagode. Optimizing the estimation of a histogram-bin width-application to the multivariate mixture-model estimation. *Mathematics*, 8(7):1090, 2020b. doi: 10.3390/math8071090. URL `https://www.mdpi.com/2227-7390/8/7/1090`.

B. Panić, J. Klemenc, and M. Nagode. Gaussian mixture model based classification revisited: Application to the bearing fault classification. *Journal of Mechanical Engineering*, 66(4):215–226, 2020c. doi: 10.5545/sv-jme.2020.6563. URL `https://www.sv-jme.eu/article/gaussian-mixture-model-based-classification-revisited-application-to-the-bearing-fault-classi`

R. Varadhan and C. Roland. Simple and globally convergent methods for accelerating the convergence of any em algorithm. *Scandinavian Journal of Statistics*, 35(2):335–353, 2008.

P. F. Velleman. Interactive computing for exploratory data analysis i: Display algorithms. In *Proceedings of the Statistical Computing Section*, Washington, D.C., 1976. American Statistical Association.

M. Wiper, D. R. Insua, and F. Ruggeri. Mixtures of gamma distributions with applications. *Journal of Computational and Graphical Statistics*, 10(3):440–454, 2001. URL `http://www.jstor.org/stable/1391098`.

Marko Nagode
University of Ljubljana
Faculty of Mechanical Engineering
Aškerčeva 6
1000 Ljubljana
Slovenia
Marko.Nagode@fs.uni-lj.si.

```
R> plot(poissonest, pos = 1, what = c("pdf", "marginal pdf", "IC",
+       "D", "logL"), nrow = 2, ncol = 3, npts = 1000, family = "sans")
```

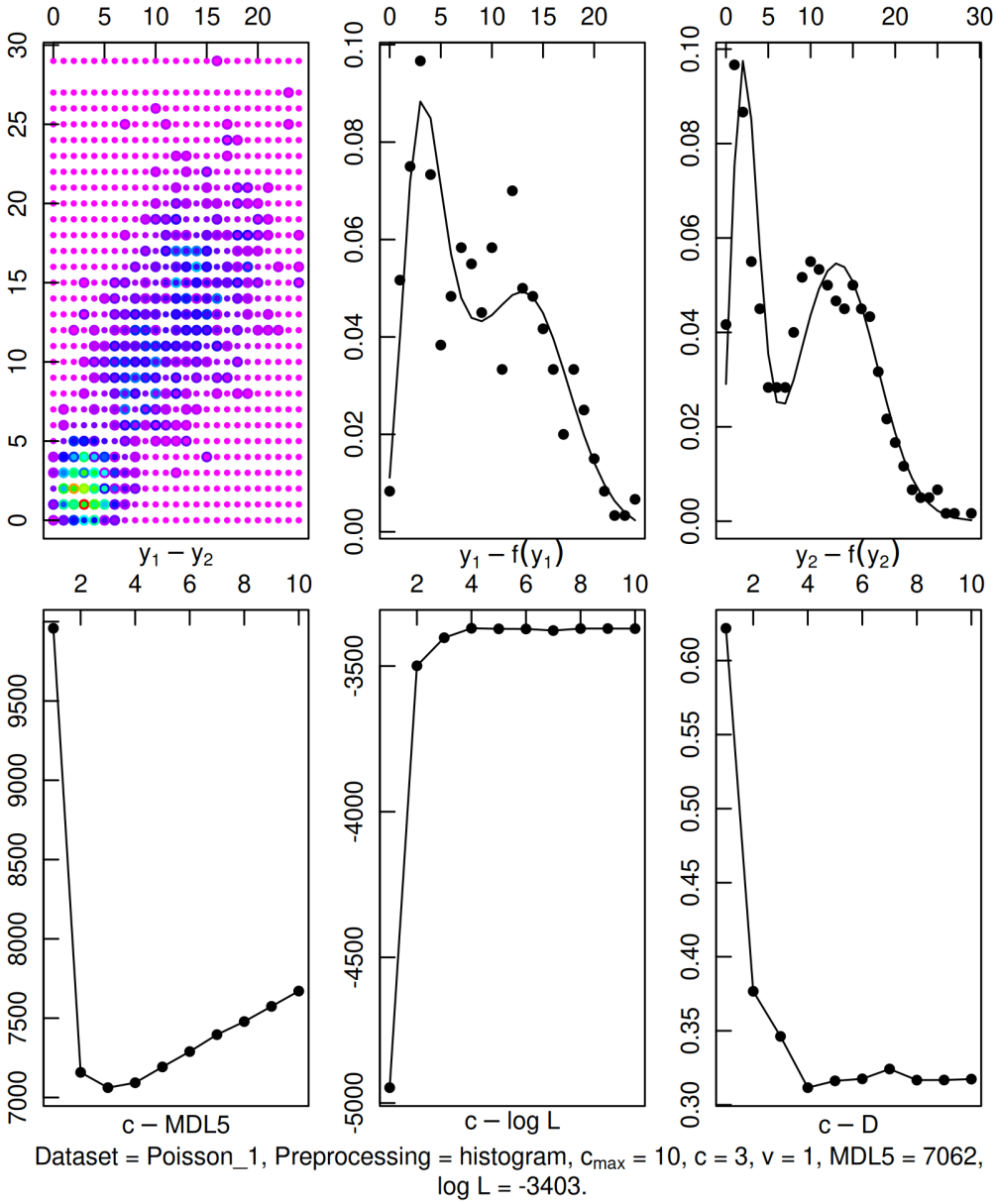

Figure 2: Poisson dataset. Empirical densities (coloured large circles), predictive multivariate Poisson-Poisson mixture density (coloured small circles), empirical densities (circles), predictive univariate marginal Poisson mixture densities and progress charts (solid line).

```
R> poissonclu <- RCLRMIX(x = poissonest, pos = 1, Zt = a.Zt(poisson))
R> plot(poissonclu, family = "sans")
```

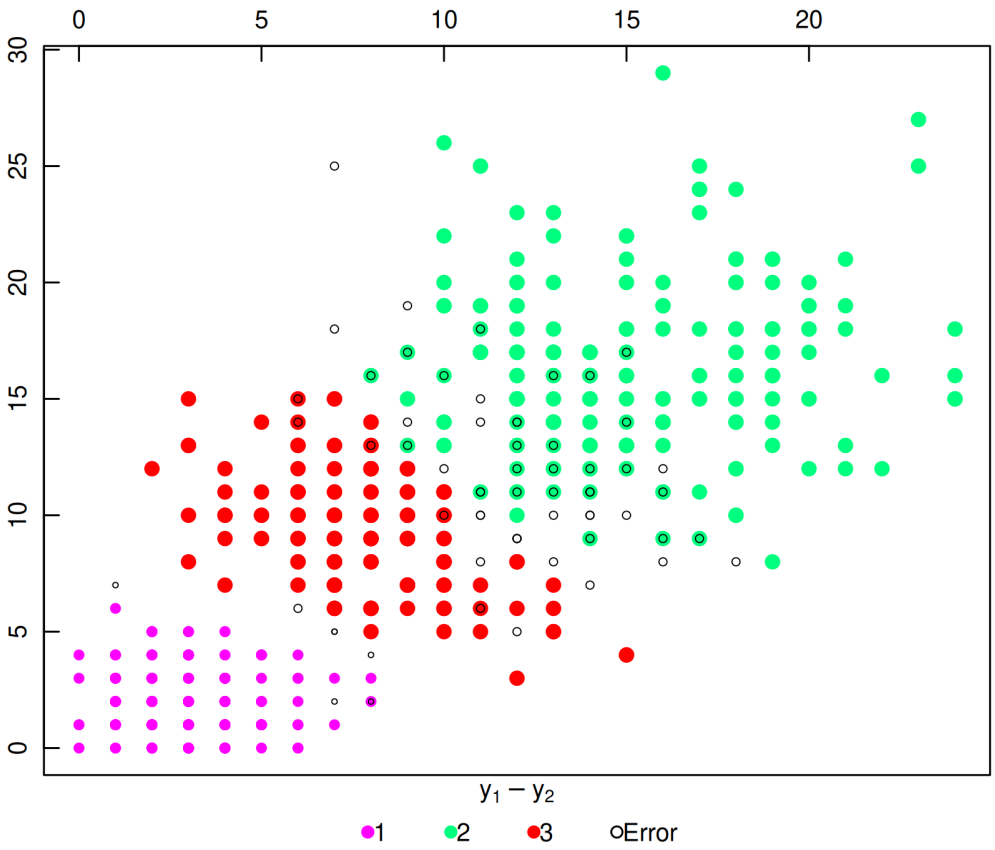

Figure 3: Poisson dataset. Predictive cluster membership (coloured circles), error (black circles).

```
R> plot(mvnormest, family = "sans")
```

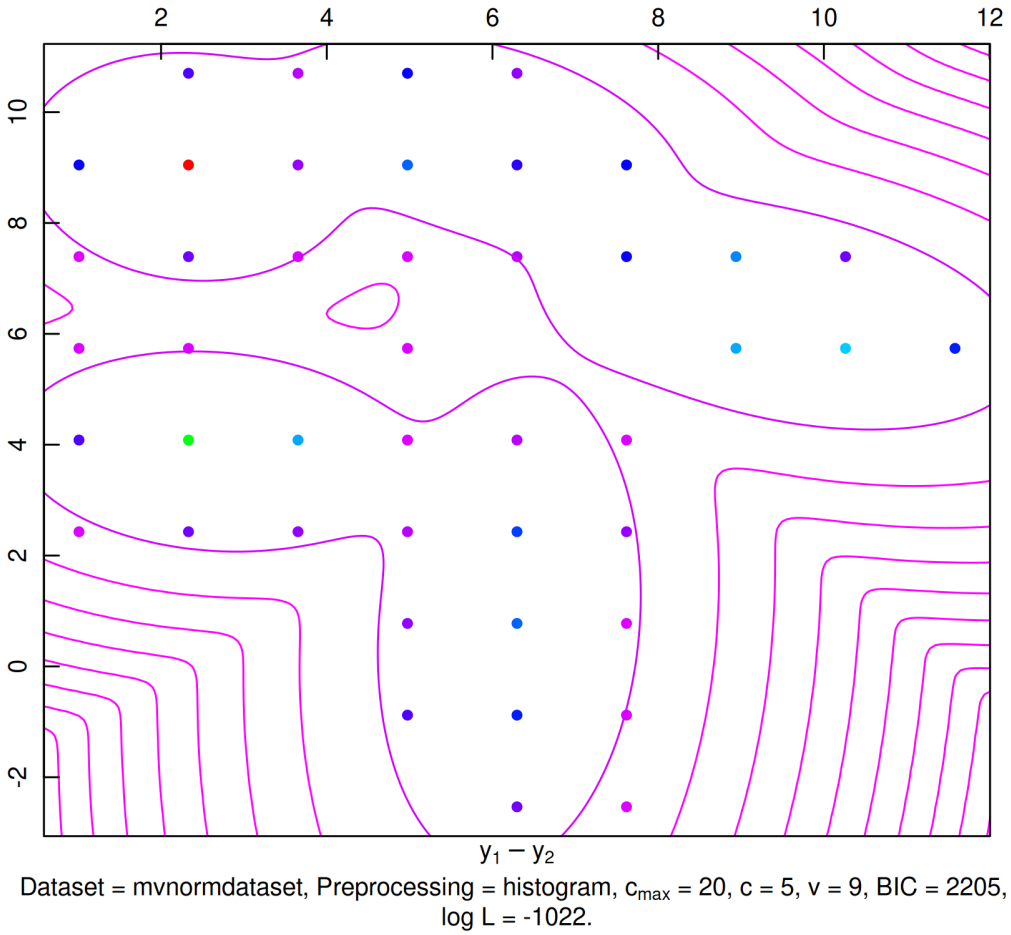

Figure 4: Dataset `mvnorm`. Empirical densities (coloured circles), predictive multivariate normal mixture density (coloured lines).

```
R> mvnormclu <- RCLRMIX(model = "RCLRMVNORM", x = mvnormest)
R> plot(mvnormclu, family = "sans")
```

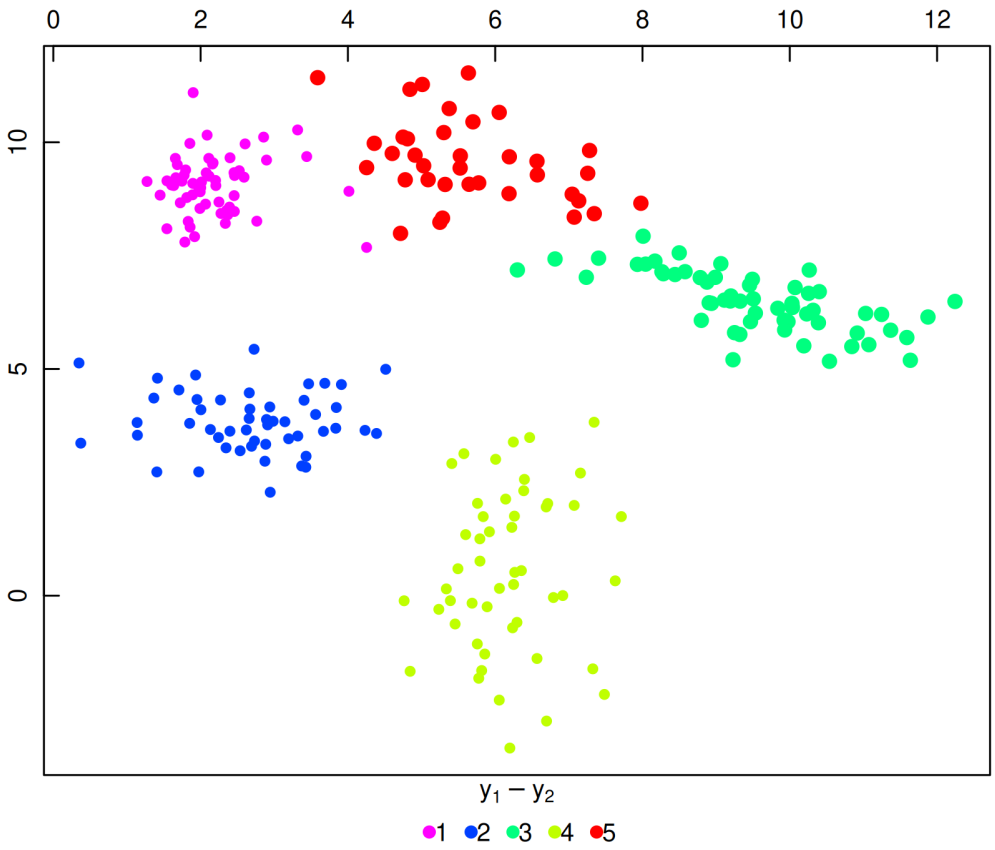

Figure 5: Dataset `mvnorm`. Predictive cluster membership (coloured circles).

```
R> plot(mvnorm.clu, family = "sans")
```

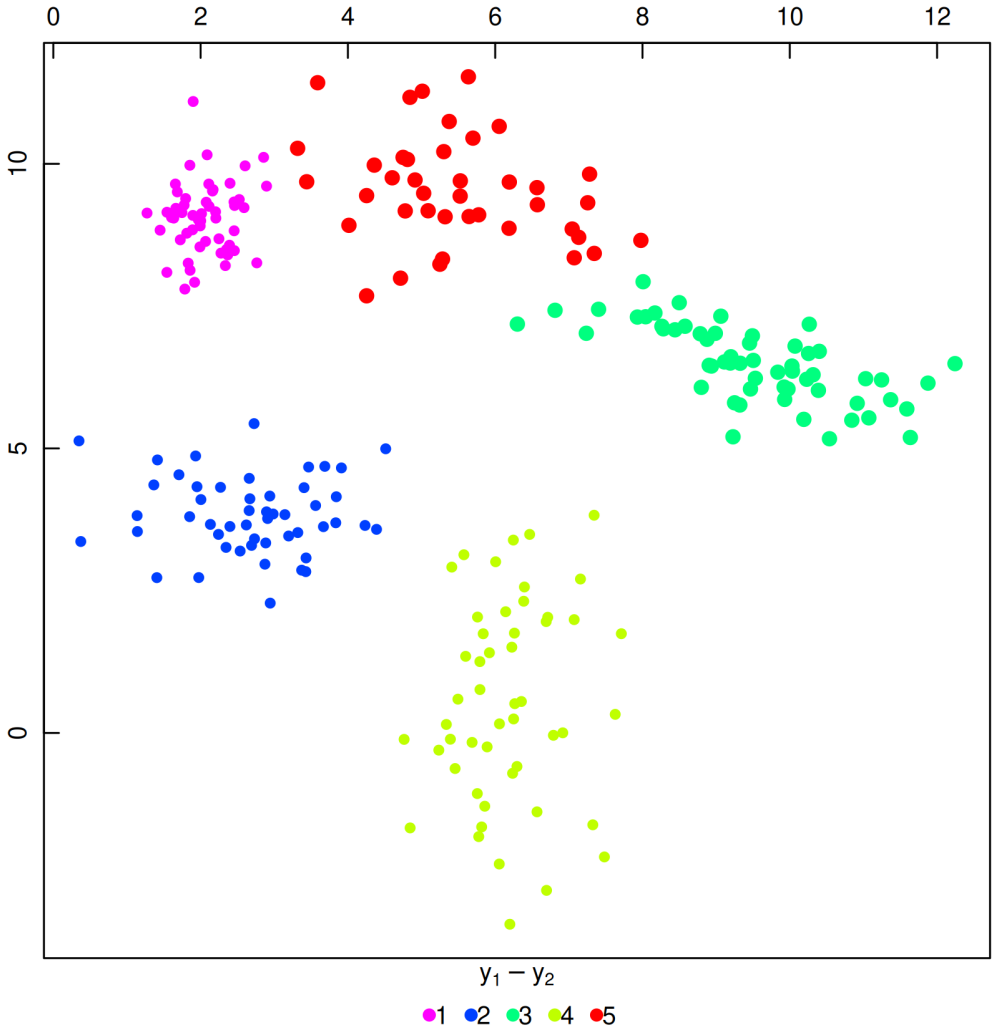

Figure 6: Dataset `mvnorm`. Predictive cluster membership (coloured circles) for exhaustive REBMIX&ECM strategy and ICL criterion.

```
R> plot(drivecla, nrow = 3, ncol = 2, family = "sans")
```

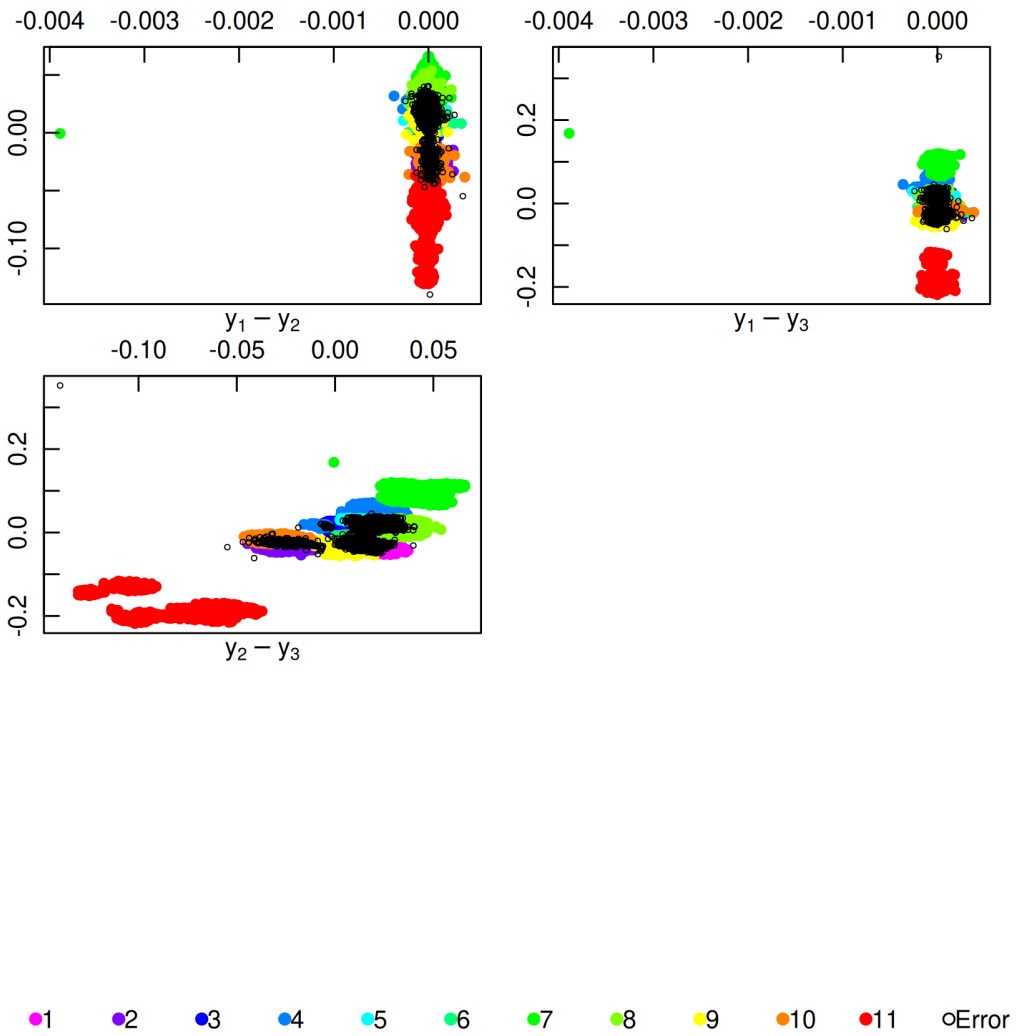

Figure 7: Dataset `iris`. Predictive class membership (coloured circles), error (black circles).

```
R> plot(adultcla, nrow = 5, ncol = 2, family = "sans")
```

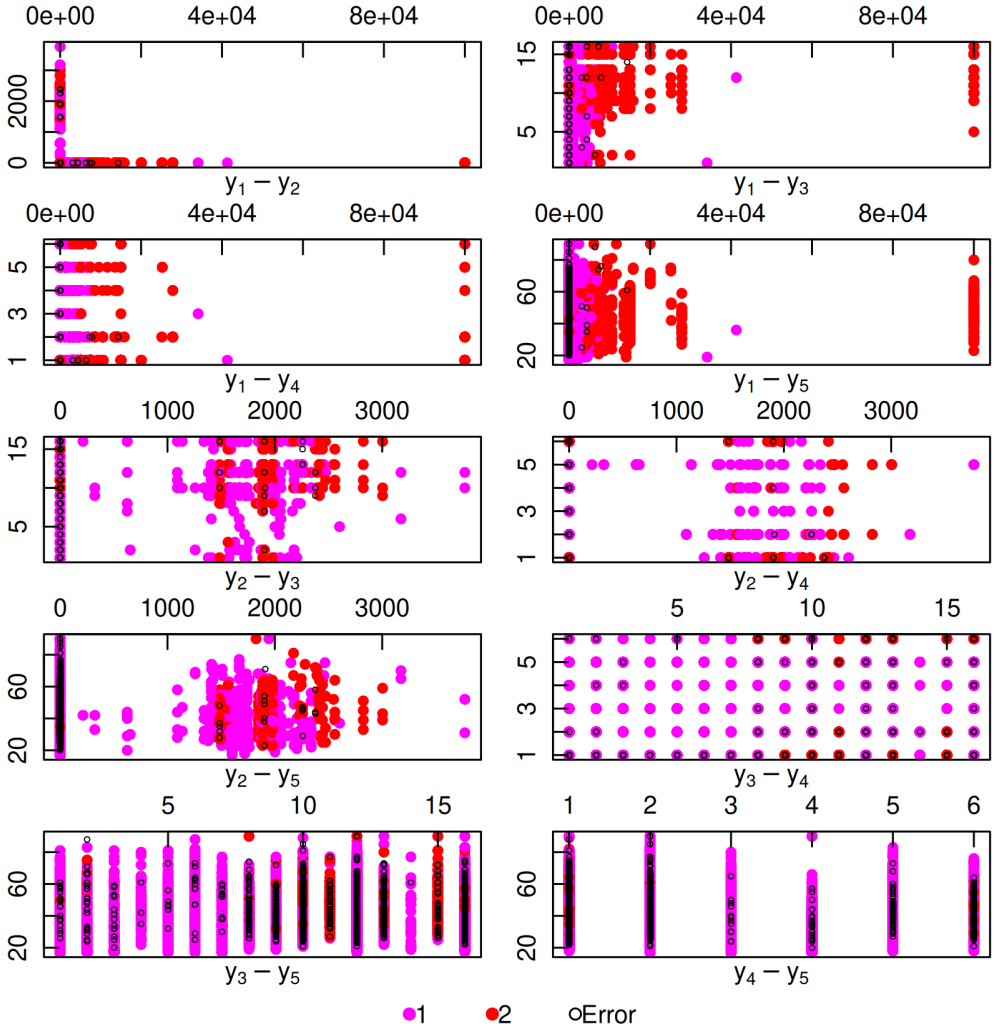

Figure 8: Dataset `adult`. Predictive class membership (coloured circles), error (black circles).