Journal of Statistical Software

October 2015, Volume 67, Issue 6. *doi: 10.18637/jss.v067.i06*

Rmixmod: The R Package of the Model-Based Unsupervised, Supervised, and Semi-Supervised Classification Mixmod Library

Rémi Lebret
Université de Technologie de Compiègne & CNRS

Serge Iovleff
Université Lille 1 & CNRS

Florent Langrognet
CNRS & Université de Franche-Comté

Christophe Biernacki
Université Lille 1 & CNRS

Gilles Celeux
Inria Saclay

Gérard Govaert
Université de Technologie de Compiègne & CNRS

Abstract

Mixmod is a well-established software package for fitting mixture models of multivariate Gaussian or multinomial probability distribution functions to a given dataset with either a clustering, a density estimation or a discriminant analysis purpose. The **Rmixmod** S4 package provides an interface from the R statistical computing environment to the C++ core library of **Mixmod** (**mixmodLib**). In this article, we give an overview of the model-based clustering and classification methods implemented, and we show how the R package **Rmixmod** can be used for clustering and discriminant analysis.

Keywords: model-based clustering, discriminant analysis, mixture models, visualization, R, **Rmixmod**.

1. Introduction

Clustering and discriminant analysis (or classification) methods are among the most important techniques in multivariate statistical learning. The goal of cluster analysis is to partition the observations into groups ("clusters") so that the pairwise dissimilarities between observations assigned to the same cluster tend to be smaller than observations in different clusters. The goal of classification is to design a decision function from a learning dataset to assign new data to groups *a priori* known. *Mixture modeling* supposes that the data are an i.i.d. sample from some population described by a probability density function. This density function is a finite mixture of parametric component density functions, each component modeling one of the clusters. This model is fit to the data by maximum likelihood (McLachlan and Peel 2000).

The **Mixmod** package (**Mixmod** Team 2008) is primarily devoted to clustering using mixture models and, to a lesser extent, to discriminant analysis (supervised and semi-supervised situations). Many options are available to specify the models and the strategies to be run. **Mixmod** allows to fit 28 multivariate Gaussian mixture models for quantitative data and 10 multivariate multinomial mixture models for qualitative data. Estimation of the mixture parameters is performed via the EM, the stochastic EM or the classification EM algorithms. These three algorithms can be chained and initialized in several different ways leading to original strategies (see Section 2.3). The model selection criteria BIC (Bayesian information criterion), ICL (integrated classification likelihood), NEC (normalized entropy criterion), and cross-validation are proposed depending on the modeling purpose (see Section 2.4).

Mixmod, developed since 2001, is a package written in C++. Its core library **mixmodLib** can be interfaced with any other software packages or libraries, or can be used from the command line. It has been already interfaced with Scilab (Scilab Enterprises 2015) and MATLAB (The MathWorks Inc. 2014), see Biernacki, Celeux, Govaert, and Langrognet (2006). So far it was lacking an interface to R (R Core Team 2015). The **Rmixmod** package provides a bridge between the R statistical computing environment and the C++ core library of **Mixmod** and. Both cluster analysis and discriminant analysis can be now performed in R using **Rmixmod**. User-friendly outputs and graphs allow for a relevant and appealing visualization of the results. The package is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=Rmixmod`.

There exists a wide variety of packages in R dedicated to the estimation of mixture models, see also the CRAN Task View "Cluster Analysis & Finite Mixture Models" (Leisch and Grün 2015). Among them let us cite **bgmm** (Biecek, Szczurek, Vingron, and Tiuryn 2012), **flexmix** (Leisch 2004; Grün and Leisch 2007, 2008), **mclust** (Fraley and Raftery 2007b,a), **mixtools** (Benaglia, Chauveau, Hunter, and Young 2009), but none of them offer the large set of possibilities as the newcomer **Rmixmod**.

This paper reviews in Section 2 Gaussian and multinomial mixture models and the **Mixmod** library. An overview of the **Rmixmod** package is then given in Section 3 through a description of the main function and of other related companion functions. The practical use of this package is illustrated in Section 4 on toy datasets for model-based clustering in a quantitative and qualitative setting (Section 4.1) and for discriminant analysis (Section 4.2). Section 5 evokes future works of the **Mixmod** project.

2. Overview of the Mixmod library functionalities

2.1. Model-based classification focus

"X-supervised" classifications

Roughly speaking, the **Mixmod** library is devoted to three kinds of different classification tasks. Its main task is unsupervised classification, but supervised and semi-supervised classifications can benefit from its meaningful models, its efficient algorithms and its model selection criteria.

Unsupervised classification. Unsupervised classification, also called cluster analysis, is concerned with discovering a group structure in an n by d data matrix $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ where $\mathbf{x}_i$ is an individual in $\mathbb{X}_1 \times \ldots \times \mathbb{X}_d$. The space $\mathbb{X}_j$ $(j = 1, \ldots, d)$ depends on the type of data at hand: It is $\mathbb{R}$ for continuous data and it is $\{1, \ldots, m_j\}$ for categorical data with m_j levels. The result provided by clustering is typically a partition $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ of $\mathbf{x}$ into K groups, the $\mathbf{z}_i$'s being indicator vectors or *labels* with $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$, $z_{ik} = 1$ or 0, depending on if $\mathbf{x}_i$ belongs to the kth group or not.

Supervised classification. In discriminant analysis, data are composed by n observations $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ $(\mathbf{x}_i \in \mathbb{X}_1 \times \ldots \times \mathbb{X}_d)$ and a partition of $\mathbf{x}$ into K groups defined with the labels $\mathbf{z}$. The aim is to estimate the group $\mathbf{z}_{n+1}$ of any new individual $\mathbf{x}_{n+1}$ of $\mathbb{X}_1 \times \ldots \times \mathbb{X}_d$ with unknown label. Discriminant analysis in **Mixmod** is divided into two steps. The first step consists of determining a classification rule from the training dataset $(\mathbf{x}, \mathbf{z})$. The second step consists of assigning the other observations to one of the groups.

Semi-supervised classification. Usually all the labels $\mathbf{z}_i$ are completely unknown (unsupervised classification) or completely known (supervised classification). Nevertheless, partial labeling of data is possible, and it leads to the so-called semi-supervised classification. The **Mixmod** library handles situations where the dataset $\mathbf{x}$ is divided into two subsets $\mathbf{x} = (\mathbf{x}^\ell, \mathbf{x}^u)$ with $\mathbf{x}^\ell = \{\mathbf{x}_1, \ldots, \mathbf{x}_g\}$ $(1 \leq g \leq n)$ being units with known labels $\mathbf{z}^\ell = \{\mathbf{z}_1, \ldots, \mathbf{z}_g\}$ and $\mathbf{x}^u = \{\mathbf{x}_{g+1}, \ldots, \mathbf{x}_n\}$ units with unknown labels $\mathbf{z}^u = \{\mathbf{z}_{g+1}, \ldots, \mathbf{z}_n\}$.

Usually, semi-supervised classification is concerned with the supervised classification purpose and it aims at estimating the group $\mathbf{z}_{n+1}$ of any new individual $\mathbf{x}_{n+1}$ of $\mathbb{X}_1 \times \ldots \times \mathbb{X}_d$ with unknown label by also taking profit of the unlabeled data of the learning set.

Model-based classifications

The model-based point of view allows to consider all previous classification tasks in a unified manner.

Mixture models. Let $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be n independent vectors in $\mathbb{X}_1 \times \ldots \times \mathbb{X}_d$, where each $\mathbb{X}_j$ denotes some measurable space, and such that each $\mathbf{x}_i$ arises from a mixture probability distribution with density

$$f(\mathbf{x}_i|\boldsymbol{\theta}) = \sum_{k=1}^{K} p_k h(\mathbf{x}_i|\boldsymbol{\alpha}_k), \tag{1}$$

where the p_k's are the mixing proportions ($0 < p_k < 1$ for all $k = 1, \ldots, K$ and $p_1 + \ldots + p_K = 1$), $h(\cdot|\boldsymbol{\alpha}_k)$ denotes a d-dimensional distribution parameterized by $\boldsymbol{\alpha}_k$. As we will see below, h is for instance the density of a Gaussian distribution with mean $\boldsymbol{\mu}_k$ and variance matrix Σ_k and, thus, $\boldsymbol{\alpha}_k = (\boldsymbol{\mu}_k, \Sigma_k)$. The whole parameter vector (to be estimated) of f is denoted by $\boldsymbol{\theta} = (p_1, \ldots, p_K, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K)$.

Label estimation. From a generative point of view, drawing the sample $\mathbf{x}$ from the mixture distribution f requires to first draw a sample of labels $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, with $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$, $z_{ik} = 1$ or 0, depending on if $\mathbf{x}_i$ is arising from the kth mixture component or not. Depending on if the sample $\mathbf{z}$ is completely unknown, completely known or only partially known, we retrieve an unsupervised, a supervised or a semi-supervised classification problem, respectively. Mixture models are particularly well-suited for modeling these different standard situations since an estimate of any label $\mathbf{z}_i$ ($i = 1, \ldots, n$ for unsupervised classification, $i = n+1$ for supervised or semi-supervised classification) can be easily obtained by the following so-called *maximum a posteriori* (MAP) rule

$$\hat{\mathbf{z}}(\boldsymbol{\theta}) = \mathrm{MAP}(\mathbf{t}(\boldsymbol{\theta})) \quad \Leftrightarrow \quad \hat{z}_{ik}(\boldsymbol{\theta}) = \begin{cases} 1 & \text{if } k = \arg\max_{k' \in \{1,\ldots,K\}} t_{ik'}(\boldsymbol{\theta}) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{t}(\boldsymbol{\theta}) = \{t_{ik}(\boldsymbol{\theta})\}$, $t_{ik}(\boldsymbol{\theta})$ denoting the conditional probability that the observation $\mathbf{x}_i$ arises from group k:

$$t_{ik}(\boldsymbol{\theta}) = \frac{p_k h(\mathbf{x}_i|\boldsymbol{\alpha}_k)}{f(\mathbf{x}_i|\boldsymbol{\theta})}. \tag{2}$$

2.2. Parsimonious and meaningful models

The **Mixmod** library proposes many parsimonious and meaningful models, depending on the type of variables to be considered. Such models provide simple interpretations of groups.

Continuous variables: Fourteen Gaussian models

In the Gaussian mixture model, each $\mathbf{x}_i$ is assumed to arise independently from a mixture of d-dimensional Gaussian densities with mean $\boldsymbol{\mu}_k$ and variance matrix Σ_k. In this case we have in Equation 1, with $\boldsymbol{\alpha}_k = (\boldsymbol{\mu}_k, \Sigma_k)$,

$$h(\mathbf{x}_i|\boldsymbol{\alpha}_k) = (2\pi)^{-d/2}|\Sigma_k|^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right\}.$$

Thus, clusters associated with the mixture components are ellipsoidal, centered at the means $\boldsymbol{\mu}_k$ and the variance matrices Σ_k determine their geometric characteristics.

Following Banfield and Raftery (1993) and Celeux and Govaert (1995), we consider a parameterization of the variance matrices of the mixture components consisting of expressing the variance matrix Σ_k in terms of its eigenvalue decomposition

$$\Sigma_k = \lambda_k D_k A_k D_k^\top, \tag{3}$$

where $\lambda_k = |\Sigma_k|^{1/d}$, D_k is the matrix of eigenvectors of Σ_k and A_k is a diagonal matrix, such that $|A_k| = 1$, with the normalized eigenvalues of Σ_k on the diagonal in a decreasing order. The parameter λ_k determines the *volume* of the kth cluster, D_k its *orientation* and A_k its *shape*. By allowing some but not all of these quantities to vary between clusters, we obtain parsimonious and easily interpreted models which are appropriate to describe various group situations (see Table 1). More explanations about notation used in this table are given below.

Model	Number of parameters	M step	**Rmixmod** model name
$[\lambda DAD^\top]$	$\alpha+\beta$	CF	`"Gaussian_*_L_C"`
$[\lambda_k DAD^\top]$	$\alpha+\beta+K-1$	IP	`"Gaussian_*_Lk_C"`
$[\lambda DA_kD^\top]$	$\alpha+\beta+(K-1)(d-1)$	IP	`"Gaussian_*_L_D_Ak_D"`
$[\lambda_k DA_kD^\top]$	$\alpha+\beta+(K-1)d$	IP	`"Gaussian_*_Lk_D_Ak_D"`
$[\lambda D_kAD_k^\top]$	$\alpha+K\beta-(K-1)d$	CF	`"Gaussian_*_L_Dk_A_Dk"`
$[\lambda_k D_kAD_k^\top]$	$\alpha+K\beta-(K-1)(d-1)$	IP	`"Gaussian_*_Lk_Dk_A_Dk"`
$[\lambda D_kA_kD_k^\top]$	$\alpha+K\beta-(K-1)$	CF	`"Gaussian_*_L_Ck"`
$[\lambda_k D_kA_kD_k^\top]$	$\alpha+K\beta$	CF	`"Gaussian_*_Lk_Ck"`
$[\lambda B]$	$\alpha+d$	CF	`"Gaussian_*_L_B"`
$[\lambda_k B]$	$\alpha+d+K-1$	IP	`"Gaussian_*_Lk_B"`
$[\lambda B_k]$	$\alpha+Kd-K+1$	CF	`"Gaussian_*_L_Bk"`
$[\lambda_k B_k]$	$\alpha+Kd$	CF	`"Gaussian_*_Lk_Bk"`
$[\lambda I]$	$\alpha+1$	CF	`"Gaussian_*_L_I"`
$[\lambda_k I]$	$\alpha+K$	CF	`"Gaussian_*_Lk_I"`

Table 1: Some characteristics of the 14 models. We have $\alpha = Kd + K - 1$, $* = pk$ in the case of free proportions and $\alpha = Kd$, $* = p$ in the case of equal proportions, and $\beta = \frac{d(d+1)}{2}$. CF means that the M step is in closed form, IP means that the M step needs an iterative procedure.

The general family. First, we can allow the volumes, the shapes and the orientations of clusters to vary or to be equal between clusters. Variations on assumptions on the parameters λ_k, D_k and A_k $(1 \leq k \leq K)$ lead to eight general models of interest. For instance, we can assume different volumes and keep the shapes and orientations equal by requiring that $A_k = A$ (A unknown) and $D_k = D$ (D unknown) for $k = 1, \ldots, K$. We denote this model as $[\lambda_k DAD^\top]$ (or, shortly, $[\lambda_k C]$ where $C = DAD^\top$). With this convention, writing $[\lambda D_k AD_k^\top]$ means that we consider the mixture model with equal volumes, equal shapes and different orientations.

The diagonal family. Another family of interest consists of assuming that the variance matrices Σ_k are diagonal. In the parameterization (3), this means that the orientation matrices D_k are permutation matrices. We write $\Sigma_k = \lambda_k B_k$ where B_k is a diagonal matrix with $|B_k| = 1$. This particular parameterization gives rise to four models: $[\lambda B]$, $[\lambda_k B]$, $[\lambda B_k]$ and $[\lambda_k B_k]$.

The spherical family. The last family of models consists of assuming spherical shapes, namely $A_k = I$, I denoting the identity matrix. In such a case, two parsimonious models are in competition: $[\lambda I]$ and $[\lambda_k I]$.

Remark. The **Mixmod** library provides also some Gaussian models devoted to high dimensional data. We do not describe them here since they are not yet available in the **Rmixmod** package but the reader can refer to the **Mixmod** website `http://www.mixmod.org/` for further information.

Categorical variables: Five multinomial models

We consider now that the data are n objects described by d categorical variables, with respective number of levels $m_1, \dots, m_d$. The data can be represented by n binary vectors $\mathbf{x}_i = (x_i^{jh}; j = 1, \dots, d; h = 1, \dots, m_j)$ $(i = 1, \dots, n)$ where $x_i^{jh} = 1$ if the object i belongs to the level h of the variable j and 0 otherwise. Denoting $m = \sum_{j=1}^{d} m_j$ the total number of levels, the data matrix $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ has n rows and m columns. Binary data can be seen as a particular case of categorical data with d dichotomous variables, i.e., $m_j = 2$ for any $j = 1, \dots, d$.

The latent class model assumes that the d categorical variables are independent given the latent variable: Each $\mathbf{x}_i$ arises independently from a mixture of multivariate multinomial distributions (Everitt 1984). In this case we have in Equation 1

$$h(\mathbf{x}_i|\boldsymbol{\alpha}_k) = \prod_{j=1}^{d} \prod_{h=1}^{m_j} (\alpha_k^{jh})^{x_i^{jh}} \tag{4}$$

with $\boldsymbol{\alpha}_k = (\alpha_k^{jh}; j = 1, \dots, d; h = 1, \dots, m_j)$. In (4), we recognize the product of d conditionally independent multinomial distributions with parameters $\boldsymbol{\alpha}_k^j = (\alpha_k^{j1}, \dots, \alpha_k^{jm_j})$. This model may present problems of identifiability (see for instance Goodman 1974) but most situations of interest are identifiable (Allman, Matias, and Rhodes 2009).

In order to propose more parsimonious models, we present the following extension of the parameterization of Bernoulli distributions used by Celeux and Govaert (1991) for clustering and also by Aitchison and Aitken (1976) for kernel discriminant analysis. The basic idea is to impose the condition on the vector $\boldsymbol{\alpha}_k^j$ to have a unique modal value for one of its components with the other components sharing uniformly the remaining mass probability. Thus, $\boldsymbol{\alpha}_k^j$ takes the form $(\beta_k^j, \dots, \beta_k^j, \gamma_k^j, \beta_k^j, \dots, \beta_k^j)$ with $\gamma_k^j > \beta_k^j$. Since $\sum_{h=1}^{m_j} \alpha_k^{jh} = 1$, we have $(m_j - 1)\beta_k^j + \gamma_k^j = 1$ and, consequently, $\beta_k^j = (1 - \gamma_k^j)/(m_j - 1)$. The constraint $\gamma_k^j > \beta_k^j$ becomes finally $\gamma_k^j > 1/m_j$. Equivalently and meaningfully, the vector $\boldsymbol{\alpha}_k^j$ can be reparameterized by a center $\mathbf{a}_k^j$ and a dispersion ε_k^j around this center with the following decomposition:

- **Center:** $\mathbf{a}_k^j = (a_k^{j1}, \dots, a_k^{jm_j})$ where $a_k^{jh} = 1$ if h indicates the position of γ_k^j (in the following, this position will be denoted $h(k, j)$) and 0 otherwise.
- **Dispersion:** $\varepsilon_k^j = 1 - \gamma_k^j$ the probability that the data $\mathbf{x}_i$, arising from the kth component, are such that $x_i^{jh(k,j)} \neq 1$.

Thus, it allows us to give an interpretation similar to the center and the variance matrix used for continuous data in the Gaussian mixture context. The relationship between the initial parameterization and the new one is given by:

$$\alpha_k^{jh} = \begin{cases} 1 - \varepsilon_k^j & \text{if } h = h(k, j), \\ \varepsilon_k^j/(m_j - 1) & \text{otherwise.} \end{cases}$$

Equation 4 can be rewritten with $\mathbf{a}_k = (\mathbf{a}_k^j; j = 1, \dots, d)$ and $\boldsymbol{\varepsilon}_k = (\varepsilon_k^j; j = 1, \dots, d)$ giving

$$h(\mathbf{x}_i|\boldsymbol{\alpha}_k) = \tilde{h}(\mathbf{x}_i|\mathbf{a}_k, \varepsilon_k) = \prod_{j=1}^{d} \prod_{h=1}^{m_j} \left((1 - \varepsilon_k^j)^{a_k^{jh}} (\varepsilon_k^j/(m_j - 1))^{1 - a_k^{jh}} \right)^{x_i^{jh}}.$$

Model	Number of parameters	**Rmixmod** model name
$[\varepsilon]$	$\delta+1$	`"Binary_*_E"`
$[\varepsilon^j]$	$\delta+d$	`"Binary_*_Ej"`
$[\varepsilon_k]$	$\delta+K$	`"Binary_*_Ek"`
$[\varepsilon_k^j]$	$\delta+Kd$	`"Binary_*_Ekj"`
$[\varepsilon_k^{jh}]$	$\delta+K\sum_{j=1}^d(m_j-1)$	`"Binary_*_Ekjh"`

Table 2: Number of free parameters of the five multinomial models. We have $\delta = K-1$, $*$ $= pk$ in the case of free proportions and $\delta = 0$, $* = p$ in the case of equal proportions.

In the following, this model will be denoted as $[\varepsilon_k^j]$. In this context, three other models can be defined. We denote $[\varepsilon_k]$ the model where ε_k^j is independent of the variable j, $[\varepsilon^j]$ the model where ε_k^j is independent of the component k and, finally, $[\varepsilon]$ the model where ε_k^j is independent of both the variable j and the component k. In order to maintain some consistency in the notation, we will denote also with $[\varepsilon_k^{jh}]$ the most general model introduced in the previous section. The number of free parameters associated with each model is given in Table 2.

2.3. Efficient maximum "X-likelihood" estimation strategies

EM and EM-like algorithms focus

Estimation of the mixture parameters is performed either through maximization of the log-likelihood (ML) on $\boldsymbol{\theta}$

$$L(\boldsymbol{\theta}) = \sum_{i=1}^n \ln f(\mathbf{x}_i|\boldsymbol{\theta})$$

via the EM algorithm (*expectation maximization,* Dempster, Laird, and Rubin 1997), the SEM algorithm (*stochastic EM,* Celeux and Diebolt 1985) or through maximization of the completed log-likelihood on both $\boldsymbol{\theta}$ and $\mathbf{z}$

$$L_c(\boldsymbol{\theta},\mathbf{z}) = \sum_{i=1}^n\sum_{k=1}^K z_{ik}\ln(p_k h(\mathbf{x}_i|\boldsymbol{\alpha}_k)), \tag{5}$$

via the CEM algorithm (*classification EM,* Celeux and Govaert 1992). We now describe these three algorithms at iteration q. The choice of the starting parameter $\boldsymbol{\theta}^{\{0\}}$ and of the stopping rules are both described later.

The EM algorithm. It consists of repeating the following E and M steps:

- **E step:** Compute the conditional probabilities $\mathbf{t}(\boldsymbol{\theta}^{\{q\}})$ (see Equation 2).
- **M step:** Compute the parameter $\boldsymbol{\theta}^{\{q+1\}} = \operatorname{argmax}_{\boldsymbol{\theta}} L_c(\boldsymbol{\theta},\mathbf{t}(\boldsymbol{\theta}^{\{q\}}))$ (see Equation 5). Mixture proportions are given by $p_k^{\{q+1\}} = \sum_{i=1}^n t_{ik}(\boldsymbol{\theta}^{\{q\}})/n$. Detailed formulas of the other parameters $\boldsymbol{\alpha}^{\{q+1\}}$ depend on the model at hand and are given in the reference manual of **Mixmod** (**Mixmod** Team 2008).

The SEM algorithm. It is a stochastic version of EM incorporating between the E and M steps a so-called S step restoring stochastically the unknown labels $\mathbf{z}$:

- **E step:** Like EM.
- **S step:** Draw labels $\mathbf{z}^{\{q\}}$ from $\mathbf{t}(\boldsymbol{\theta}^{\{q\}})$ with $\mathbf{z}_i^{\{q\}} \sim \text{multinomial}(t_{i1}(\boldsymbol{\theta}^{\{q\}}), \dots, t_{iK}(\boldsymbol{\theta}^{\{q\}}))$.
- **M step:** Like EM but $\mathbf{t}(\boldsymbol{\theta}^{\{q\}})$ is replaced by $\mathbf{z}^{\{q\}}$.

It is important to notice that SEM does not converge pointwise. It generates a Markov chain whose stationary distribution is more or less concentrated around the ML estimate. A natural estimate from a SEM sequence $(\boldsymbol{\theta}^{\{q\}})_{q=1,\dots,Q}$ of length Q is either the mean $\sum_{q=Q^-,\dots,Q} \boldsymbol{\theta}^{\{q\}}/(Q-Q^-)$ (the first Q^- burn-in iterations are discarded) or the parameter value leading to the highest log-likelihood in the whole sequence.

The CEM algorithm. It incorporates a classification step between the E and M steps of EM, restoring by the MAP estimate the unknown labels $\mathbf{z}$:

- **E step:** Like EM.
- **C step:** Choose the most probable labels $\hat{\mathbf{z}}(\boldsymbol{\theta}^{\{q\}}) = \text{MAP}(\mathbf{t}(\boldsymbol{\theta}^{\{q\}}))$.
- **M step:** Like EM where $\mathbf{t}(\boldsymbol{\theta}^{\{q\}})$ is replaced by $\hat{\mathbf{z}}(\boldsymbol{\theta}^{\{q\}})$.

CEM leads to inconsistent estimates (Bryant and Williamson 1978; McLachlan and Peel 2000, Section 2.21) but has faster convergence than EM since it converges with a finite number of iterations. It allows also to retrieve and generalize standard K-means-like criteria, both in the continuous case (Govaert 2009, Chap. 8) and in the categorical case (Celeux and Govaert 1991).

Remark on the partial labeling case. **Mixmod** allows partial labeling for all algorithms: It is straightforward since known labels $\mathbf{z}^l$ remain fixed in the E step for all of them. In that case the log-likelihood is expressed by

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{g} \ln f(\mathbf{x}_i|\boldsymbol{\theta}) + \sum_{i=g+1}^{n}\sum_{k=1}^{K} z_{ik} \ln(p_k h(\mathbf{x}_i|\boldsymbol{\alpha}_k)) \tag{6}$$

and the completed log-likelihood, denoted now $L_c(\boldsymbol{\theta}, \mathbf{z}^u)$, is unchanged.

Remark on duplicated units. In some cases, it arises that some units are duplicated. Typically, it happens when the number of possible values for the units is low in regard to the sample size. To avoid entering unnecessarily large lists of units, it is also possible to specify a weight w_i for each unit $\mathbf{y}_i$ $(i = 1, \dots, r)$. The set $\mathbf{y}^w = \{(\mathbf{y}_1, w_1), \dots, (\mathbf{y}_r, w_r)\}$ is strictly equivalent to the set with eventual replications $\mathbf{x} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, and we have the relation $n = w_1 + \dots + w_r$.

Remark on spurious solutions. In the Gaussian case, some solutions with (finite) high log-likelihood value can be uninteresting for the user since they correspond to ill-conditioned estimates of covariance matrices for some mixture components. This corresponds to so-called *spurious situations* (McLachlan and Peel 2000, Sections 3.10 and 3.11). As far as we know such spurious solutions cannot be detected automatically and have to be discarded by hand.

Strategies for using EM and CEM

Both likelihood and completed likelihood functions usually suffer from multiple local maxima where EM and CEM algorithms can be trapped. Slow evolution of the objective function can be also encountered sometimes during a long period for some runs, in particular with EM.

Notice that SEM is not concerned by local maxima since it does not converge pointwise but slow evolution towards the stationary distribution cannot be excluded in some cases.

In order to avoid such drawbacks, **Mixmod** can act in three ways: chained algorithms, starting strategies and stopping rules. More details can be found in the **Mixmod** reference manual (**Mixmod** Team 2008).

Chained algorithms strategies. The three algorithms EM, CEM and SEM can be chained to obtain original fitting strategies (e.g., CEM then EM with results of CEM) taking advantage of each of them in the estimation process.

Initialization strategies. The available procedures of initialization are:

- `"random"`: Initialization from a random position is a standard way to initialize an algorithm. This random initial position is obtained by choosing at random centers in the dataset. This simple strategy is repeated several times from different random positions and the position maximizing the likelihood or the completed likelihood is selected.

- `"smallEM"`: A predefined number of EM iterations is split into several short runs of EM launched from random positions. By a short run of EM, we mean that we do not wait for complete convergence but we stop it as soon as the log-likelihood growth is small in comparison to a predefined crude threshold (see details in Biernacki, Celeux, and Govaert 2003). Indeed, it appears that repeating runs of EM is generally profitable since using a single run of EM can often lead to suboptimal solutions.

- `"CEM"`: A given number of repetitions of a given number of iterations of the CEM algorithm is run. One advantage of initializing an algorithm with CEM lies in the fact that CEM converges generally in a small number of iterations. Thus, without consuming a large amount of CPU times, several runs of CEM are performed. Then EM (or CEM) is run with the best solution among all repetitions.

- `"SEMMax"`: A given number of SEM iterations is run. The idea is that a SEM sequence is expected to enter rapidly into the neighborhood of the global maximum of the likelihood function.

Stopping rule strategies. There are two ways to stop an algorithm:

- `"nbIterationInAlgo"`: All algorithms can be stopped after a pre-defined number of iterations.
- `"epsilonInAlgo"`: EM and CEM can be stopped when the relative change of the criterion at hand (L or L_c) is small.

2.4. Purpose dependent model selection

It is of high interest to automatically select a model or the number K of mixture components. However, choosing a sensible mixture model is highly dependent on the modeling purpose. Before describing these criteria, it can be noted that if no information on K is available, it is recommended to vary it between 1 and the smallest integer larger than $n^{0.3}$ (Bozdogan 1993).

Density estimation

If a density estimation perspective is pursued, the BIC must be preferred. It consists of choosing the model and/or K minimizing

$$\mathrm{BIC} = -2L(\hat{\boldsymbol{\theta}}) + \nu \ln n$$

with $\hat{\boldsymbol{\theta}}$ the ML estimate and ν the number of parameters estimated. The BIC is an asymptotic approximation of the integrated likelihood, valid under regularity conditions, and has been proposed by Schwarz (1978). Despite the fact that those regularity conditions are not fulfilled for mixtures, it has been proved that the criterion BIC is consistent if the likelihood remains bounded (Keribin 2000) and has been proved to be efficient on practical grounds (see for instance Fraley and Raftery 1998).

Unsupervised classification

In the unsupervised setting, three criteria are available: BIC, ICL and NEC. But when pursuing a cluster analysis perspective, ICL and NEC can provide more parsimonious answers.

The integrated likelihood does not take into account the ability of the mixture model to give evidence for a clustering structure of the data. An alternative is to consider the integrated completed likelihood. Asymptotic considerations lead to the ICL criterion to be minimized (Biernacki, Celeux, and Govaert 2000):

$$\begin{aligned}\mathrm{ICL} &= -2L_c(\hat{\boldsymbol{\theta}}, \mathbf{t}(\hat{\boldsymbol{\theta}})) + \nu \ln n \\ &= \mathrm{BIC} - 2\sum_{i=1}^{n}\sum_{k=1}^{K} t_{ik}(\hat{\boldsymbol{\theta}}) \ln t_{ik}(\hat{\boldsymbol{\theta}}).\end{aligned}$$

Notice that both expressions of ICL above allow to consider ICL either as L_c penalized by the model complexity or as BIC penalized by an entropy term measuring the mixture component overlap.

The NEC criterion measures the ability of a mixture model to provide well separated clusters and is derived from a relation highlighting the differences between the maximum likelihood approach and the classification maximum likelihood approach to the mixture problem. It is defined by

$$\mathrm{NEC}_K = \begin{cases} \frac{-\sum_{i=1}^n \sum_{k=1}^K t_{ik}(\hat{\boldsymbol{\theta}}_K) \ln t_{ik}(\hat{\boldsymbol{\theta}}_K)}{L(\hat{\boldsymbol{\theta}}_K) - L(\hat{\boldsymbol{\theta}}_1)} & \text{if } K > 1 \\ 1 & \text{otherwise} \end{cases}$$

with $\hat{\boldsymbol{\theta}}_K$ the ML estimate of $\boldsymbol{\theta}$ for K components. The index K is used to highlight that NEC is essentially devoted to choosing the number of mixture components K, not the model parameterization (Celeux and Soromenho 1996; Biernacki, Celeux, and Govaert 1999). The chosen value of K corresponds to the lowest value of NEC.

Supervised classification

In the supervised setting, note that only the model (not the number of mixture components) has to be selected. Two criteria are proposed in this situation: BIC and cross-validation. For BIC, the completed log-likelihood (5), where $\mathbf{z}$ is fixed to its known value, has to be used. The cross-validation criterion (CV) is valid only in the discriminant analysis (supervised) context. The model leading to the highest CV criterion value is selected. Cross-validation is a resampling method which can be summarized as follows: Consider random splits of the whole dataset $(\mathbf{x}, \mathbf{z})$ into V independent datasets $(\mathbf{x}, \mathbf{z})^{(1)}, \ldots, (\mathbf{x}, \mathbf{z})^{(V)}$ of approximately equal sizes $n_1, \ldots, n_V$. (If n/V is an integer h, we have $n_1 = \ldots = n_V = h$.) The CV criterion is then defined by

$$\mathrm{CV} = \frac{1}{n} \sum_{v=1}^{V} \sum_{i \in I_v} \delta(\hat{\mathbf{z}}_i(\hat{\boldsymbol{\theta}}^{(v)}), \mathbf{z}_i),$$

where I_v denotes the indices i of data included in $(\mathbf{x}, \mathbf{z})^{(v)}$, δ corresponds to the 0–1 cost and $\hat{\mathbf{z}}_i(\hat{\boldsymbol{\theta}}^{(v)})$ denotes the group to which $\mathbf{x}_i$ is assigned when designing the assignment rule from the entire dataset $(\mathbf{x}, \mathbf{z})$ without $(\mathbf{x}, \mathbf{z})^{(v)}$. When $V = n$, cross-validation is known as the *leave-one-out* procedure, and, in this case, fast estimation of the n discriminant rules is implemented in **Mixmod** in the Gaussian case (Biernacki and Govaert 1999).

Semi-supervised classification

Two criteria are available in the semi-supervised context (supervised purpose): BIC and CV. For BIC, the partial labeled log-likelihood (6) has to be used. For CV, the whole dataset is split at random in V blocks of approximately equal sizes, including both the labeled and the unlabeled units, to obtain unbiased estimates of the error rate (Vandewalle, Biernacki, Celeux, and Govaert 2010). However, note that the CV criterion is quite expensive to be computed in the semi-supervised setting since it requires to run an EM algorithm V times to estimate $\hat{\boldsymbol{\theta}}^{(v)}$.

2.5. Mixmod library implementation and related packages

*The **Mixmod** library*

The **Mixmod** core library (**mixmodLib)** is the main product of the **Mixmod** software package. Developed since 2001, it has been downloaded from the **Mixmod** web site http://www.mixmod.org/ about 300 times per year. Distributed under the GNU GPL license,

mixmodLib has been enhanced and improved for years (Biernacki *et al.* 2006). Important work has been done to improve performance of the **mixmodLib** which can today treat very large datasets quickly with accuracy and robustness. Currently, some arbitrarily large "hard" limits for the sample size and for the variable number are respectively fixed to 1 000 000 and 10 000. It is possible to change them but it requires to recompile the source code. The user must also be aware that to reach these limits in practice will essentially depend on the available computing resources.

It contains about 80 C++ classes and can be used in the command line or can be interfaced with any other software package or library (in accordance with the terms of the GNU GPL license). Some of these C++ classes (top level classes) have been created to easily interface **mixmodLib**. Clustering can be performed with the top level '`XEMClusteringMain`' class (using '`XEMClusteringInput`' and '`XEMClusteringOutput`' classes) and discriminant analysis with the '`XEMLearnMain`' class (using '`XEMLearnInput`' and '`XEMLearnOutput`' classes) for the first step and the '`XEMPredictMain`' class (using '`XEMPredcitInput`' and '`XEMpredictOutput`' classes) for the second step (prediction).

The **Rmixmod** package uses also the **Rcpp** package (Eddelbuettel and François 2011) which provides C++ classes that greatly facilitate interfacing C or C++ code in R packages. Since the **Rcpp** package works only on R versions 2.15 and above, an up-to-date version of R is required for smooth installation of the package.

Existing related packages

To provide a suitable product for an increasingly large and various public, the **Mixmod** team has developed four products, available at `http://www.mixmod.org/`:

- **mixmodLib** (developed since 2001), the core library which can be interfaced with any other software package and can also be used in the command line (for *expert* users).
- **mixmodForMatlab** (developed since 2002), a collection of MATLAB functions to call **mixmodLib** supplemented by some functions to visualize results.
- **mixmodGUI** (developed since 2009), a very user-friendly software package which provides all the clustering functionalities of **mixmodLib**; we plan to make available soon also discriminant analysis functionalities.

3. Overview of the Rmixmod functions

3.1. Main Rmixmod functions

Unsupervised classification and density estimation

Cluster analysis can be performed with the function `mixmodCluster()`. Illustration of use of this function is given in Section 4.1. This function has two mandatory arguments: a data frame `x` and a list of number of groups. Default values for model and strategy will be used unless users specify a list of models with the `models` option (see Section 3.2) or a new strategy with the `strategy` option (see Section 3.3). By default only the BIC criterion is used to select

Input parameter	Description
`data`	Data frame containing quantitative or qualitative data. Rows correspond to observations and columns correspond to variables.
`nbCluster`	Numeric vector indicating the number of clusters.
`dataType`	Character indicating the type of data being either `"quantitative"` or `"qualitative"`. Set as `NULL` by default, type will be guessed depending on variables type.
`models`	A '`Model`' object defining the list of models to run. For quantitative data, the model `"Gaussian_pk_Lk_C"` is called (see `mixmodGaussianModel()` in Section 3.2 for specifying other models). For qualitative data, the model `"Binary_pk_Ekjh"` is called (see `mixmodMultinomialModel()` in Section 3.2 for specifying other models).
`strategy`	A '`Strategy`' object containing the strategy to run. By default `mixmodStrategy()` (see Section 3.3) is called.
`criterion`	Character vector defining the criterion to select the best model. The best model is the one with the lowest criterion value. Possible values: `"BIC"`, `"ICL"`, `"NEC"`, `c("BIC", "ICL", "NEC")`. Default is `"BIC"`.
`weight`	Numeric vector with n (number of individuals) rows. `weight` is optional. This option is to be used when weights are associated with the data.
`knownLabels`	Numeric vector of size n. It will be used for semi-supervised classification when labels are known. Each element corresponds to a cluster assignment.

Table 3: List of all the input parameters of the `mixmodCluster()` function.

models, but users can make a list of criteria by using the `criterion` option. In Table 3 the reader will find a summary of all the input parameters of the `mixmodCluster()` function with its default value if it is not a mandatory parameter.

The `mixmodCluster()` function returns an instance of the '`MixmodCluster`' class. Its two attributes will contain all outputs:

- `results`: A list of '`MixmodResults`' objects containing all the results sorted in ascending order according to the given criterion.
- `bestResult`: A '`MixmodResults`' object containing the best model results.

Supervised and semi-supervised classification

Supervised and semi-supervised classification can be performed using the `mixmodLearn()` and the `mixmodPredict()` functions. Both functions are illustrated in Section 4.2.

`mixmodLearn()` function. It has two mandatory arguments: a data matrix **`x`** and a vector containing the known labels **`z`**. As for the `mixmodCluster()` function the three arguments `models`, `weight` and `criterion` are available. The default criterion is CV (cross-validation).

Input parameter	Description
`data`	Data frame containing quantitative or qualitative data. Rows correspond to observations and columns correspond to variables.
`knownLabels`	Numeric vector of size equal to the number of observations. Each element corresponds to a cluster assignment. The maximum value corresponds to the number of clusters.
`dataType`	Character indicating the type of data being either `"quantitative"` or `"qualitative"`. Set as `NULL` by default, type will be guessed depending on variables type.
`models`	A '`Model`' object defining the list of models to run. For quantitative data, the model `"Gaussian_pk_Lk_C"` is called (see `mixmodGaussianModel()` in Section 3.2 for specifying other models). For qualitative data, the model `"Binary_pk_Ekjh"` is called (see `mixmodMultinomialModel()` in Section 3.2 for specifying other models).
`criterion`	Character vector defining the criterion to select the best model. Possible values: `"BIC"`, `"CV"` or `c("CV", "BIC")`. Default is `"CV"`.
`nbCVBlocks`	Integer value defining the number of blocks to perform the cross-validation. This value will be ignored if the CV criterion is not chosen. Default value is 10.
`weight`	Numeric vector with n (number of individuals) elements. `weight` is optional. This option is to be used when weights are associated with the data.

Table 4: List of all the input parameters of the `mixmodLearn()` function.

Input parameter	Description
`data`	Data frame containing quantitative or qualitative data. Rows correspond to observations and columns correspond to variables.
`classificationRule`	A '`MixmodResults`' object which contains the classification rule computed in the `mixmodLearn()` or `mixmodCluster()` step.

Table 5: List of the input parameters of the `mixmodPredict()` function.

In Table 4 the reader will find a summary of all the input parameters of the `mixmodLearn()` function and default values for non-mandatory parameters.

The `mixmodLearn()` function returns an instance of the '`MixmodLearn`' class. Its two attributes will contain all outputs:

- `results`: A list of '`MixmodResults`' objects containing all the results sorted in ascending order according to the given criterion (in descending order for the CV criterion).
- `bestResult`: A '`MixmodResults`' object containing the best model results.

`mixmodPredict()` function. It only needs two arguments: a data matrix of the remaining observations and a classification rule (see Table 5). It returns an instance of the '`MixmodPredict`' class which contains predicted partitions and probabilities.

3.2. Companion functions for model definition

Continuous variables: Gaussian models

All the Gaussian models summarized in Table 1 are available in **Rmixmod**. Users can get all the 28 models by calling `mixmodGaussianModel()`.

```
R> all <- mixmodGaussianModel()
R> all

****************************************
*** Mixmod Models:
* list =  Gaussian_pk_L_I Gaussian_pk_Lk_I Gaussian_pk_L_B Gaussian_pk_Lk_B
          Gaussian_pk_L_Bk Gaussian_pk_Lk_Bk Gaussian_pk_L_C Gaussian_pk_Lk_C
          Gaussian_pk_L_D_Ak_D Gaussian_pk_Lk_D_Ak_D Gaussian_pk_L_Dk_A_Dk
          Gaussian_pk_Lk_Dk_A_Dk Gaussian_pk_L_Ck Gaussian_pk_Lk_Ck
          Gaussian_p_L_I Gaussian_p_Lk_I Gaussian_p_L_B Gaussian_p_Lk_B
          Gaussian_p_L_Bk Gaussian_p_Lk_Bk Gaussian_p_L_C Gaussian_p_Lk_C
          Gaussian_p_L_D_Ak_D Gaussian_p_Lk_D_Ak_D Gaussian_p_L_Dk_A_Dk
          Gaussian_p_Lk_Dk_A_Dk Gaussian_p_L_Ck Gaussian_p_Lk_Ck
* This list includes models with free and equal proportions.
****************************************
```

This function has four parameters to specify some particular models in the family:

- `listModels` can be used when users want to use specific models

  ```
  R> list.models <- mixmodGaussianModel(listModels = c("Gaussian_p_L_C",
  +     "Gaussian_p_L_Dk_A_Dk", "Gaussian_pk_Lk_B"))
  ```

- `free.proportions` and `equal.proportions` can be used to include models with free or equal proportions

  ```
  R> only.free.proportions <- mixmodGaussianModel(
  +     equal.proportions = FALSE)
  ```

- `family` allows including models from a specific family (`"general"`, `"diagonal"`, `"spherical"`, `"all"`)

  ```
  R> family.models <- mixmodGaussianModel(
  +     family = c("general", "spherical"), free.proportions = FALSE)
  ```

Categorical variables: Multinomial models

All the multinomial models summarized in Table 2 are available in **Rmixmod**. Users can get all the 10 models by calling `mixmodMultinomialModel()`.

```
R> all <- mixmodMultinomialModel()
R> all
```

```
****************************************
*** Mixmod Models :
* list =  Binary_pk_E Binary_pk_Ekj Binary_pk_Ekjh Binary_pkEj Binary_pk_Ek
          Binary_p_E Binary_p_Ekj Binary_p_Ekjh Binary_p_Ej Binary_p_Ek
* This list includes models with free and equal proportions.
****************************************
```

This function has five arguments. As `mixmodGaussianModel()` this function has the following parameters: `listModels`, `free.proportions` and `equal.proportions`.

```
R> only.free.proportions <- mixmodMultinomialModel(equal.proportions = FALSE)
R> list.models <- mixmodMultinomialModel(listModels = c("Binary_p_E",
+    "Binary_p_Ekjh", "Binary_pk_Ekj", "Binary_pk_Ej"))
```

But users can also use `variable.independency` and `component.independency` in order to include models which are independent of the variable j or independent of the component k.

```
R> var.independent <- mixmodMultinomialModel(variable.independency = TRUE)
R> var.comp.independent <- mixmodMultinomialModel(
+    variable.independency = TRUE, component.independency = TRUE)
```

3.3. Companion function for maximum likelihood estimation strategies

The strategies described in Section 2.3 can be tuned using the `mixmodStrategy()` function. The `mixmodStrategy()` function has no mandatory arguments and the default arguments are the ones specified in the **Mixmod** documentation (**Mixmod** Team 2008). In Table 6 the reader will find a summary of all the input parameters of the `mixmodStrategy()` function.

The `mixmodStrategy()` function returns an instance of the '`MixmodStrategy`' class. A default strategy can be defined in **Rmixmod** with the `mixmodStrategy()` function:

```
R> mixmodStrategy()

****************************************
*** MIXMOD Strategy:
* algorithm            =  EM
* number of tries      =  1
* number of iterations =  200
* epsilon              =  0.001
*** Initialization strategy:
* algorithm            =  smallEM
* number of tries      =  50
* number of iterations =  5
* epsilon              =  0.001
* seed                 =  NULL
****************************************
```

Here are other examples to show different ways to set a strategy:

Input parameter	Description
`algo`	Character vector with the estimation algorithm. Possible values: `"EM"`, `"SEM"`, `"CEM"`, `c("EM", "SEM")`. Default value: `"EM"`.
`nbTry`	Integer value defining the number of tries. `nbTry` must be a positive integer. Default value: 1.
`initMethod`	Character value defining the method of initialization of the algorithm specified in the `algo` argument. Possible values: `"random"`, `"smallEM"`, `"CEM"`, `"SEMMax"`. Default value: `"smallEM"`.
`nbTryInInit`	Integer value defining number of tries in `initMethod` algorithm. `nbTryInInit` must be a positive integer. Option available only if `initMethod` is `"smallEM"` or `"CEM"`. Default value: 50.
`nbIterationInInit`	Integer value defining the number of `"EM"` or `"SEM"` iterations in `initMethod`. `nbIterationInInit` must be a positive integer. Only available if `initMethod` is `"smallEM"` or `"SEMMax"`. Default values: 5 if `initMethod` is `"smallEM"` and 100 if `initMethod` is `"SEMMax"`.
`nbIterationInAlgo`	Integer vector defining the number of iterations if `nbIteration` is used as a stopping rule for the algorithm(s). Default value: 200.
`epsilonInInit`	Numeric value defining the epsilon value in the initialization step. Only available if `initMethod` is `"smallEM"`. Default value: 0.001.
`epsilonInAlgo`	Numeric vector defining the epsilon value for the algorithm. Warning: `epsilonInAlgo` does not make any sense if `algo` is `"SEM"`, so it needs to be set as `NaN` in that case. Default value: 0.001.
`seed`	Random seed used in the random number generator. Default value: `NULL`.

Table 6: List of all the input parameters of the `mixmodStrategy()` function.

```
R> strategy1 <- mixmodStrategy(algo = "CEM", initMethod = "random",
+    nbTry = 10, epsilonInInit = 0.000001)
R> strategy2 <- mixmodStrategy(algo = c("SEM", "EM"),
+    nbIterationInAlgo = c(200, 100), epsilonInAlgo = c(NA, 1e-04),
+    seed = 2408)
```

It is well-known that the number of local maxima of the log-likelihood function increases in conjunction with the number of parameters to be estimated. In such a situation, **Rmixmod** is able to avoid these traps by tuning its previous input parameters, in particular by increasing the value of `nbTry`. The need for this adjustment is easily detected by the fact that multiple re-runs of **Rmixmod** provide unstable results.

However, ability of **Rmixmod** to provide a good search over the log-likelihood function increases the frequency of spurious solutions in the Gaussian case (see a description in Section 2.3.1). Such solutions are then easily manually discarded.

In addition, for some specific reproducibility purposes, note that **Rmixmod** allows also to control exactly the random seed by providing the optional `seed` argument (see Table 6).

3.4. Other companion functions

Non-graphical functions

`show`, `print` and `summary` methods have been implemented for the **Rmixmod** S4 classes '`Strategy`', '`Model`', '`GaussianParameter`', '`MultinomialParameter`', '`MixmodResults`', '`MixmodCluster`', '`MixmodLearn`' and '`MixmodPredict`'.

The **Rmixmod** package provides two other utility functions:

1. `nbFactorFromData()`: Allows to get the number of levels of each column of a dataset.
2. `sortbyCriterion()`: After calling the `mixmodCluster()` or `mixmodLearn()` method, results will be sorted into ascending order according to the first given criterion (descending order for the CV criterion). This method is able to reorder the list of results according to a given criterion. The input parameters are
   - `object`: a '`Mixmod`' object;
   - `criterion`: a string containing the criterion name.

Most of these functions will be illustrated in Section 4.

Graphical functions

Methods for `plot`, `hist` and `barplot` have been implemented for the **Rmixmod** S4 class '`MixmodResults`'. `hist` and `barplot` are each specific for quantitative and qualitative data. All these functions will be also illustrated in Section 4.

4. Rmixmod through examples

4.1. Unsupervised classification

Continuous variables: Geyser dataset

The outputs and graphs of clustering with **Rmixmod** are illustrated on the well-known `geyser` dataset (Azzalini and Bowman 1990). It is a data frame containing 272 observations from the Old Faithful Geyser in the Yellowstone National Park. The same version of the dataset as in package **Rmixmod** is also available as dataset `faithful` in the base package **datasets**. A more complete version is provided by the **MASS** package (Venables and Ripley 2002). Each observation consists of two measurements: The duration (in minutes) of the eruption and the waiting time (in minutes) to the next eruption. In this example we ignore the partition and we want to estimate the best Gaussian mixture model fitting the dataset. The following code provides a way to do this by running a cluster analysis for different numbers of clusters (from 2 to 8 clusters), all Gaussian models, the BIC, ICL and NEC model selection criteria, and `strategy2` defined in Section 3.3:

```
R> data("geyser", package = "Rmixmod")
R> xem.geyser <- mixmodCluster(data = geyser, nbCluster = 2:8,
+    criterion = c("BIC", "ICL", "NEC"), models = mixmodGaussianModel(),
+    strategy = strategy2)
```

The xem.geyser object contains information both on input and output of the clustering:

```
R> xem.geyser

****************************************
*** INPUT:
****************************************
* nbCluster =  2 3 4 5 6 7 8
* criterion =  BIC ICL NEC
****************************************
*** MIXMOD Models:
* list =  Gaussian_pk_L_I Gaussian_pk_Lk_I Gaussian_pk_L_B Gaussian_pk_Lk_B
Gaussian_pk_L_Bk Gaussian_pk_Lk_Bk Gaussian_pk_L_C Gaussian_pk_Lk_C
Gaussian_pk_L_D_Ak_D Gaussian_pk_Lk_D_Ak_D Gaussian_pk_L_Dk_A_Dk
Gaussian_pk_Lk_Dk_A_Dk Gaussian_pk_L_Ck Gaussian_pk_Lk_Ck Gaussian_p_L_I
Gaussian_p_Lk_I Gaussian_p_L_B Gaussian_p_Lk_B Gaussian_p_L_Bk
Gaussian_p_Lk_Bk Gaussian_p_L_C Gaussian_p_Lk_C Gaussian_p_L_D_Ak_D
Gaussian_p_Lk_D_Ak_D Gaussian_p_L_Dk_A_Dk Gaussian_p_Lk_Dk_A_Dk
Gaussian_p_L_Ck Gaussian_p_Lk_Ck
* This list includes models with free and equal proportions.
****************************************
* data (limited to a 10x10 matrix) =
      Duration Waiting.Time
 [1,] 3.6      79
 [2,] 1.8      54
 [3,] 3.333    74
 [4,] 2.283    62
 [5,] 4.533    85
 [6,] 2.883    55
 [7,] 4.7      88
 [8,] 3.6      85
 [9,] 1.95     51
[10,] 4.35     85
* ... ...
****************************************
*** MIXMOD Strategy:
* algorithm            =  SEM EM
* number of tries      =  1
* number of iterations =  200 100
* epsilon              =  NaN 1e-04
*** Initialization strategy:
* algorithm            =  smallEM
* number of tries      =  50
* number of iterations =  5
* epsilon             =  0.001
* seed                =  2408
****************************************


****************************************
*** BEST MODEL OUTPUT:
*** According to the BIC criterion
****************************************
* nbCluster   =  3
* model name  =  Gaussian_p_L_C
* criterion   =  BIC(2312.5998) ICL(2434.4125) NEC(0.3837)
* likelihood  =  -1131.0738
****************************************
*** Cluster 1
* proportion =  0.3333
* means      =  4.5545 81.0500
* variances  = |     0.0796     0.5340 |
               |     0.5340    34.2128 |
*** Cluster 2
* proportion =  0.3333
* means      =  2.0390 54.5080
* variances  = |     0.0796     0.5340 |
               |     0.5340    34.2128 |
*** Cluster 3
* proportion =  0.3333
* means      =  3.9755 78.7194
* variances  = |     0.0796     0.5340 |
               |     0.5340    34.2128 |
****************************************
```

A summary of the previous information can also be obtained:

```
R> summary(xem.geyser)
```

```
**************************************************************
* Number of samples    =  272
* Problem dimension    =  2
**************************************************************
*       Number of cluster =  3
*              Model Type =  Gaussian_p_L_C
*               Criterion =  BIC(2312.5998) ICL(2434.4125) NEC(0.3837)
*              Parameters =  list by cluster
*                  Cluster  1 :
                        Proportion =  0.3333
                             Means =  4.5545 81.0500
                         Variances = |     0.0796     0.5340 |
```

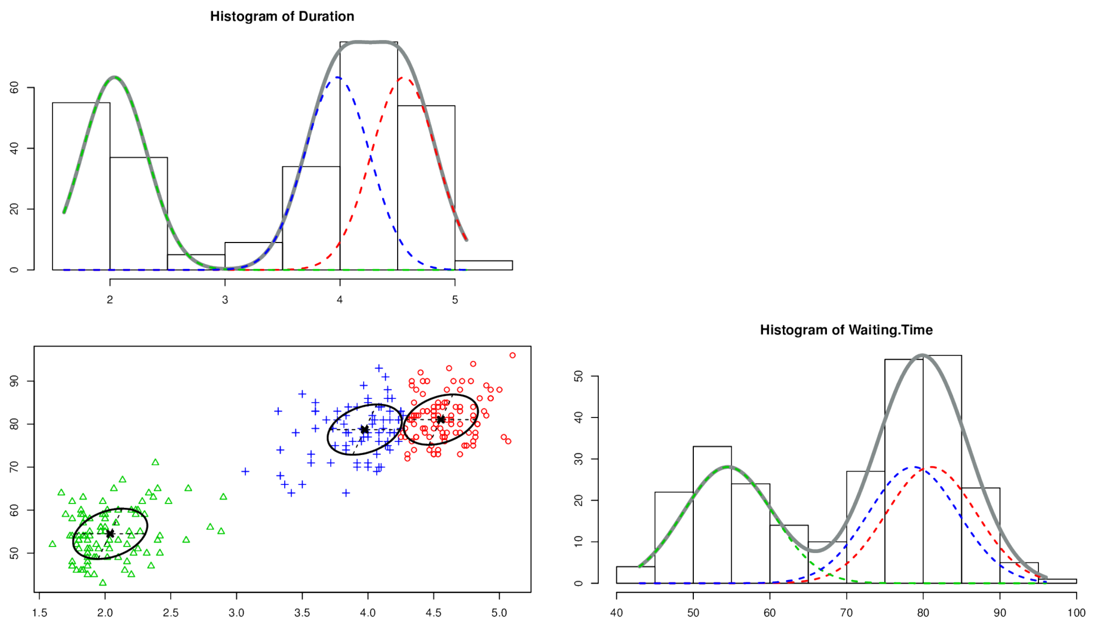


Figure 1: Output displayed by the `plot()` function for the `geyser` dataset.

```
                                        |      0.5340    34.2128 |
*                  Cluster  2 :
                        Proportion =  0.3333
                             Means =  2.0390 54.5080
                         Variances = |      0.0796      0.5340 |
                                     |      0.5340    34.2128 |
*                  Cluster  3 :
                        Proportion =  0.3333
                             Means =  3.9755 78.7194
                         Variances = |      0.0796      0.5340 |
                                     |      0.5340    34.2128 |
*          Log-likelihood =  -1131.0738
**************************************************************
```

A `plot()` method has been defined which gives on the same graph:

- On the diagonal: a 1D representation with densities and data;
- On the lower triangular: a 2D representation with isodensities, data points and partition.

The output of `plot(xem.geyser)` is displayed in Figure 1.

By default, all models of the `xem.geyser@results` variable are sorted by the BIC criterion. Alternatively, it is easy to sort this list of models according to the ICL criterion value with the `sortByCriterion()` function. Then, by looking at the best result, we can see that the ICL criterion selects two clusters (contrary to BIC which selects three clusters):

```
R> icl <- sortByCriterion(xem.geyser, "ICL")
R> icl["bestResult"]
```

```
* nbCluster   =  2
* model name  =  Gaussian_pk_Lk_D_Ak_D
* criterion   =  BIC(2320.2833) ICL(2321.3701) NEC(0.0034)
* likelihood  =  -1132.1126
****************************************
*** Cluster 1
* proportion =  0.6432
* means      =  4.2915 79.9892
* variances  = |     0.1588     0.6810 |
               |     0.6810    35.7675 |
*** Cluster 2
* proportion =  0.3568
* means      =  2.0387 54.5040
* variances  = |     0.0783     0.6467 |
               |     0.6467    33.8916 |
****************************************
```

A list with all results is also available, this list being sorted by criterion values:

```
R> xem.geyser["results"]
R> icl["results"]
```

Categorical variables: Birds of different subspecies

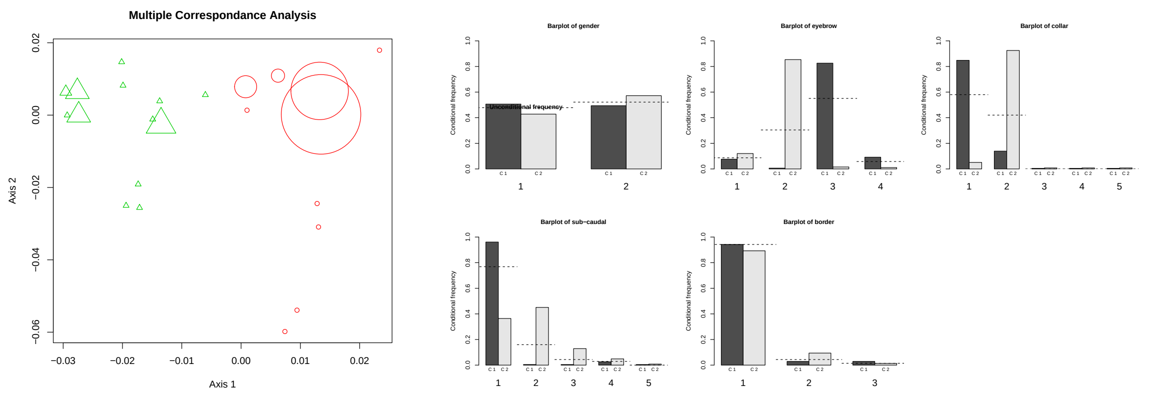
The `birds` dataset (Bretagnolle 2007) provides details on the morphology of birds (puffins). Each bird is described by five qualitative variables: one variable for the gender and four variables giving a morphological description of the birds. There are 69 puffins divided into two sub-classes: *lherminieri* and *subalaris* (34 and 35 individuals, respectively). Here we run a cluster analysis of birds with 2 clusters:

```
R> data("birds", package = "Rmixmod")
R> xem.birds <- mixmodCluster(birds, 2,
+    strategy = mixmodStrategy(seed = 2408))
```

The `plot()` method gives the following output in the qualitative case: A multiple correspondence analysis is performed to get a 2-dimensional representation of the dataset and a bigger symbol is used when observations are similar. The output of `plot(xem.birds)` is displayed in Figure 2(a).

Also a `barplot()` method has also been defined. For each qualitative variable, we obtain:

- a barplot with the frequencies of the modalities;
- for each cluster a barplot with the probabilities for each modality to be in that cluster.



(a) (b)

Figure 2: Output displayed (a) by the `plot()` function and (b) by the `barplot()` function for the `birds` dataset.

The output of `barplot(xem.birds)` is displayed in Figure 2(b).

4.2. Supervised classification

The following example concerns quantitative data. But, obviously, discriminant analysis also works with qualitative datasets in **Rmixmod**.

The outputs and graphs of discriminant analysis with **Rmixmod** are illustrated using an example where the aim is to predict a company's ability to cover its financial obligations (Jardin and Séverin 2010; Lourme and Biernacki 2011). This is an important question that requires a profound knowledge of the mechanism leading to bankruptcy. The original first sample (year 2002) is made up of 216 healthy firms and 212 insolvent firms. The second sample (year 2003) is made up of 241 healthy firms and 220 insolvent firms. Four financial ratios expected to provide some meaningful information about the company's financial health are considered: EBITDA/Total Assets, Value Added/Total Sales, Quick Ratio, Accounts Payable/Total Sales.

First step: Learning

After splitting data into years 2002 and 2003, we learn the discriminant rule on year 2002 and then we have a look at the best result:

```
R> data("finance", package = "Rmixmod")
R> ratios2002 <- subset(finance, Year == 2002, 3:6)
R> health2002 <- subset(finance, Year == 2002, 2, drop = TRUE)
R> ratios2003 <- subset(finance, Year == 2003, 3:6)
R> health2003 <- subset(finance, Year == 2003, 2, drop = TRUE)
R> learn <- mixmodLearn(ratios2002, health2002, nbCVBlocks = nrow(finance))
R> learn["bestResult"]
```

```
* nbCluster   =  2
* model name  =  Gaussian_pk_Lk_C
* criterion   =  CV(0.8201)
* likelihood  =  444.9579
****************************************
*** Cluster 1
* proportion =  0.4953
* means      =  -0.0386 0.2069 0.6089 0.1774
* variances  = |     0.0226     0.0064     0.0186    -0.0023 |
               |     0.0064     0.0166     0.0076    -0.0006 |
               |     0.0186     0.0076     0.2728    -0.0095 |
               |    -0.0023    -0.0006    -0.0095     0.0079 |
*** Cluster 2
* proportion =  0.5047
* means      =  0.1662 0.2749 1.0661 0.1079
* variances  = |     0.0172     0.0049     0.0142    -0.0017 |
               |     0.0049     0.0126     0.0058    -0.0005 |
               |     0.0142     0.0058     0.2076    -0.0073 |
               |    -0.0017    -0.0005    -0.0073     0.0060 |
****************************************
* Classification with CV:
           | Cluster 1 | Cluster 2 |
----------- ----------- -----------
 Cluster 1 |       167 |        32 |
 Cluster 2 |        45 |       184 |
----------- ----------- -----------
* Error rate with CV =  17.99 %

* Classification with MAP:
           | Cluster 1 | Cluster 2 |
----------- ----------- -----------
 Cluster 1 |       212 |         0 |
 Cluster 2 |         0 |       216 |
----------- ----------- -----------
* Error rate with MAP =  0.00 %
****************************************
```

We call now the `plot()` function to a get a visualization of the best result. The output of `plot(learn)` is displayed in Figure 3. It is also allowed to specify a subset of variables to be combined on the figure. For instance the command `plot(learn, c(1, 3))` would display only variables 1 and 3. Equivalently, the names of variables 1 and 3 could be used: `plot(learn, c("EBITDA.Total.Assets", "Quick.Ratio"))`. This functionality could be particularly useful when many variables are available.

Second step: Prediction

We perform predictions on year 2003, then we get a summary (note that `[...]` indicates that output has been truncated) and finally we compare predictions of health 2003 with the true health 2003 (75.7% correct classifications):

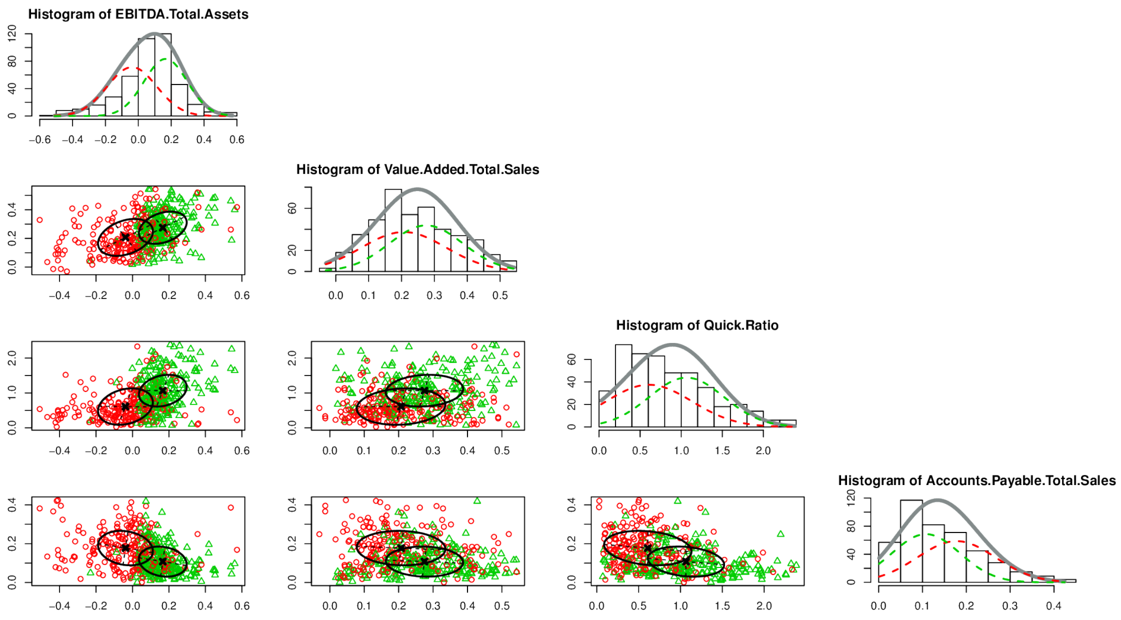


Figure 3: Output displayed by the `plot()` function for the finance dataset.

```
R> prediction <- mixmodPredict(data = ratios2003,
+     classificationRule = learn["bestResult"])
R> summary(prediction)

**************************************************************
* partition      =  2 1 1 1 [...] 1 2
* probabilities = |    0.4966   0.5034 |
                  |    0.8125   0.1875 |
                  |    0.8851   0.1149 |
                  |    0.8329   0.1671 |
                          [...]
                  |    0.5626   0.4374 |
                  |    0.0308   0.9692 |
**************************************************************

R> mean(as.integer(health2003) == prediction["partition"])

[1] 0.7570499
```

5. Further works

The **Rmixmod** package interfaces almost every functionality of the **Mixmod** library. Some particular initializations strategies and models to deal with high-dimensional data have not been implemented in the package. But initialization strategies of most interest are available in

Rmixmod and the package **HDclassif** (Bergé, Bouveyron, and Girard 2012) has been recently released for clustering and discriminant analysis of high-dimensional data.

We have proposed some tools to visualize outcomes but data visualization in **Rmixmod** can still be enhanced. In addition, supervised and semi-supervised classification currently implemented could be greatly improved by including a variable selection procedure for instance (see Maugis, Celeux, and Martin-Magniette 2011). Moreover, we encourage users to contribute by suggesting new graphics or other utility functions.

In the **Mixmod** project currently some other recent advances in model-based clustering are implemented in order to provide associated efficient R packages. This concerns for instance co-clustering (partitioning simultaneously rows and columns of a dataset) and clustering of mixed data (dealing with quantitative and qualitative data in the same analysis). The next versions of **Rmixmod** will include these latter functionalities.

References

Aitchison J, Aitken C (1976). "Multivariate Binary Discrimination by the Kernel Method." *Biometrika*, **63**(3), 413–420. doi:10.1093/biomet/63.3.413.

Allman E, Matias C, Rhodes J (2009). "Identifiability of Parameters in Latent Structure Models with Many Observed Variables." *The Annals of Statistics*, **37**(6A), 3099–3132. doi:10.1214/09-aos689.

Azzalini A, Bowman A (1990). "A Look at Some Data on the Old Faithful Geyser." *Journal of the Royal Statistical Society C*, **39**(3), 357–365. doi:10.2307/2347385.

Banfield J, Raftery A (1993). "Model-Based Gaussian and Non-Gaussian Clustering." *Biometrics*, **49**(3), 803–821. doi:10.2307/2532201.

Benaglia T, Chauveau D, Hunter D, Young D (2009). "**mixtools**: An R Package for Analyzing Finite Mixture Models." *Journal of Statistical Software*, **32**(6), 1–29. doi:10.18637/jss.v032.i06.

Bergé L, Bouveyron C, Girard S (2012). "**HDclassif**: An R Package for Model-Based Clustering and Discriminant Analysis of High-Dimensional Data." *Journal of Statistical Software*, **46**(6), 1–29. doi:10.18637/jss.v046.i06.

Biecek P, Szczurek E, Vingron M, Tiuryn J (2012). "The R Package **bgmm**: Mixture Modeling with Uncertain Knowledge." *Journal of Statistical Software*, **47**(3), 1–32. doi:10.18637/jss.v047.i03.

Biernacki C, Celeux G, Govaert G (1999). "An Improvement of the NEC Criterion for Assessing the Number of Components Arising from a Mixture." *Pattern Recognition Letters*, **20**(3), 267–272. doi:10.1016/s0167-8655(98)00144-5.

Biernacki C, Celeux G, Govaert G (2000). "Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(7), 719–725. doi:10.1109/34.865189.

Biernacki C, Celeux G, Govaert G (2003). "Choosing Starting Values for the EM Algorithm for Getting the Highest Likelihood in Multivariate Gaussian Mixture Models." *Computational Statistics & Data Analysis*, **41**(3–4), 561–575. doi:10.1016/s0167-9473(02)00163-9.

Biernacki C, Celeux G, Govaert G, Langrognet F (2006). "Model-Based Cluster and Discriminant Analysis with the **Mixmod** Software." *Computational Statistics & Data Analysis*, **51**(2), 587–600. doi:10.1016/j.csda.2005.12.015.

Biernacki C, Govaert G (1999). "Choosing Models in Model-Based Clustering and Discriminant Analysis." *Journal of Statistical Computation and Simulation*, **64**(1), 49–71. doi:10.1080/00949659908811966.

Bozdogan H (1993). "Choosing the Number of Component Clusters in the Mixture-Model Using a New Informational Complexity Criterion of the Inverse-Fisher Information Matrix." In *Information and Classification*, pp. 40–54. Springer-Verlag, Heidelberg.

Bretagnolle V (2007). Personal Communication. Source: Museum.

Bryant P, Williamson J (1978). "Asymptotic Behaviour of Classification Maximum Likelihood Estimates." *Biometrika*, **65**(2), 273–281. doi:10.1093/biomet/65.2.273.

Celeux G, Diebolt J (1985). "The SEM Algorithm: A Probabilistic Teacher Algorithm Derived from the EM Algorithm for the Mixture Problem." *Computational Statistics Quarterly*, **2**(1), 73–82.

Celeux G, Govaert G (1991). "Clustering Criteria for Discrete Data and Latent Class Models." *Journal of Classification*, **8**(2), 157–176. doi:10.1007/bf02616237.

Celeux G, Govaert G (1992). "A Classification EM Algorithm for Clustering and Two Stochastic Versions." *Computational Statistics & Data Analysis*, **14**(3), 315–332. doi:10.1016/0167-9473(92)90042-e.

Celeux G, Govaert G (1995). "Gaussian Parsimonious Clustering Models." *Pattern Recognition*, **28**(5), 781–793. doi:10.1016/0031-3203(94)00125-6.

Celeux G, Soromenho G (1996). "An Entropy Criterion for Assessing the Number of Clusters in a Mixture Model." *Journal of Classification*, **13**(2), 195–212. doi:10.1007/bf01246098.

Dempster A, Laird N, Rubin D (1997). "Maximum Likelihood from Incomplete Data with the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Everitt B (1984). *An Introduction to Latent Variable Models*. Chapman and Hall, London. doi:10.1002/bimj.4710270617.

Fraley C, Raftery A (1998). "How Many Clusters? Which Clustering Method? Answers via Model-Based Cluster Analysis." *Computer Journal*, **41**(8), 578–588. doi:10.1093/comjnl/41.8.578.

Fraley C, Raftery A (2007a). "**mclust** Version 3 for R: Normal Mixture Modeling and Model-Based Clustering." *Technical Report 504*, Department of Statistics University of Washington.

Fraley C, Raftery A (2007b). "Model-Based Methods of Classification: Using the **mclust** Software in Chemometrics." *Journal of Statistical Software*, **18**(6), 1–13. doi:10.18637/jss.v018.i06.

Goodman L (1974). "Exploratory Latent Structure Analysis Using Both Identifiable and Unidentifiable Models." *Biometrika*, **61**(2), 215–231. doi:10.1093/biomet/61.2.215.

Govaert G (2009). *Data Analysis*. John Wiley & Sons. doi:10.1002/9780470611777.

Grün B, Leisch F (2007). "Fitting Finite Mixtures of Generalized Linear Regressions in R." *Computational Statistics & Data Analysis*, **51**(11), 5247–5252. doi:10.1016/j.csda.2006.08.014.

Grün B, Leisch F (2008). "**FlexMix** Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters." *Journal of Statistical Software*, **28**(4), 1–35. doi:10.18637/jss.v028.i04.

Jardin P, Séverin E (2010). "Dynamic Analysis of the Business Failure Process: A Study of Bankruptcy Trajectories." In *Portuguese Finance Network*. Ponte Delgada, Portugual.

Keribin C (2000). "Consistent Estimation of the Order of Mixture Models." *Sankhyā: The Indian Journal of Statistics A*, **62**(1), 49–66.

Leisch F (2004). "**FlexMix**: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**(8), 1–18. doi:10.18637/jss.v011.i08.

Leisch F, Grün B (2015). "CRAN Task View: Cluster Analysis & Finite Mixture Models." Version 2015-07-24, URL http://CRAN.R-project.org/view=Cluster.

Lourme A, Biernacki C (2011). "Simultaneous t-Model-Based Clustering for Data Differing over Time Period: Application for Understanding Companies Financial Health." *Case Studies in Business, Industry and Government Statistics*, **4**(2), 73–82.

Maugis C, Celeux G, Martin-Magniette M (2011). "Variable Selection in Model-Based Discriminant Analysis." *Journal of Multivariate Analysis*, **102**(10), 1374–1387. doi:10.1016/j.jmva.2011.05.004.

McLachlan G, Peel D (2000). *Finite Mixture Models*. Wiley Series in Probability and Statistics, 1st edition. John Wiley & Sons. doi:10.1002/0471721182.

Mixmod Team (2008). ***Mixmod** Statistical Documentation*. CNRS, University Besançon.

R Core Team (2015). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Scilab Enterprises (2015). *SciLab 5.5.2.* URL http://www.scilab.org/.

The MathWorks Inc (2014). *MATLAB – The Language of Technical Computing, Version R2014b.* Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Vandewalle V, Biernacki C, Celeux G, Govaert G (2010). "A Predictive Deviance Criterion for Selecting a Generative Model in Semi-Supervised Classification." *Technical Report RR 7377*, Inria.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S.* 4th edition. Springer-Verlag, New York. doi:10.1007/978-0-387-21706-2.

Affiliation:

Rémi Lebret
Laboratoire Heudiasyc – Université de Technologie de Compiègne & CNRS
Laboratoire Paul Painlevé – Université Lille 1 & CNRS
59655 Villeneuve d'Ascq Cedex, France
E-mail: remi.lebret@math.univ-lille1.fr

Serge Iovleff, Christophe Biernacki
Laboratoire Paul Painlevé – Université Lille 1 & CNRS
Inria Lille – Nord Europe
59655 Villeneuve d'Ascq Cedex, France
E-mail: serge.iovleff@inria.fr, christophe.biernacki@inria.fr

Florent Langrognet
Laboratoire de Mathématiques – CNRS & Université de Franche-Comté
25030 Besançon Cedex, France
E-mail: florent.langrognet@univ-fcomte.fr

Gilles Celeux
Inria Saclay – Île-de-France
Dept. de Mathématiques – Université Paris-Sud
91405 Orsay Cedex, France
E-mail: gilles.celeux@inria.fr

Gérard Govaert
Laboratoire Heudiasyc – Université de Technologie de Compiègne & CNRS
60205 Compiègne Cedex, France
E-mail: gerard.govaert@utc.fr

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2015, Volume 67, Issue 6 *Submitted:* 2012-07-02
doi:10.18637/jss.v067.i06 *Accepted:* 2014-12-16