Gaussian Mixture Models in R

by Bastien Chassagnol, Antoine Bichat, Cheïma Boudjeniba, Pierre-Henri Wuillemin, Mickaël Guedj, Gregory Nuel, and Etienne Becht

Abstract Gaussian mixture models (GMMs) are widely used for modelling stochastic problems. Indeed, a wide diversity of packages have been developed in R. However, no recent review describing the main features offered by these packages and comparing their performances has been performed. In this article, we first introduce GMMs and the EM algorithm used to retrieve the parameters of the model and analyse the main features implemented among seven of the most widely used R packages. We then empirically compare their statistical and computational performances in relation with the choice of the initialisation algorithm and the complexity of the mixture. We demonstrate that the best estimation with well-separated components or with a small number of components with distinguishable modes is obtained with REBMIX initialisation, implemented in the **rebmix** package, while the best estimation with highly overlapping components is obtained with *k*-means or random initialisation. Importantly, we show that implementation details in the EM algorithm yield differences in the parameters' estimation. Especially, packages **mixtools** (Young et al. 2020) and **Rmixmod** (Langrognet et al. 2021) estimate the parameters of the mixture with smaller bias, while the RMSE and variability of the estimates is smaller with packages **bgmm** (Ewa Szczurek 2021) , **EMCluster** (W.-C. Chen and Maitra 2022) , **GMKMcharlie** (Liu 2021), **flexmix** (Gruen and Leisch 2022) and **mclust** (Fraley, Raftery, and Scrucca 2022). The comparison of these packages provides R users with useful recommendations for improving the computational and statistical performance of their clustering and for identifying common deficiencies. Additionally, we propose several improvements in the development of a future, unified mixture model package.

Introduction to Mixture modelling

Formally, let's consider a pair of random variables (X, S) with $S \in \{1, \ldots, k\}$ a discrete variable and designing the component identity of each observation. When observed, S is generally denoted as the labels of the individual observations. k is the number of mixture *components*. Then, the density distribution of X is given in Equation (1):

$$\begin{aligned} f_\theta(X) &= \sum_S f_\theta(X, S) \\ &= \sum_{j=1}^{k} p_j f_{\zeta_j}(X), \quad X \in \mathbb{R} \end{aligned} \tag{1}$$

where $\theta = (p, \zeta) = (p_1, \ldots, p_k, \zeta_1, \ldots, \zeta_k)$ denotes the parameters of the model: p_j is the proportion of component j and ζ_j represents the parameters of the density distribution followed by component j. In addition, since S is a categorical variable parametrized by p, the prior weights must enforce the unit simplex constraint (Equation (2)):

$$\begin{cases} p_j \geq 0 \quad \forall j \in \{1, \ldots, k\} \\ \sum_{j=1}^{k} p_j = 1 \end{cases} \tag{2}$$

In terms of applications, mixture models can be used to achieve the following goals:

- *Clustering*: hard clustering consists in determining a complete partition of the n observations $x_{1:n}$ into k disjoint non-empty subsets. In the context of *mixture model-based clustering*, this is done by assigning each observation i to the cluster $\hat{s_i} = \arg\max_j \eta_i(j)$ that maximises the posterior distribution (MAP) (see Equation (3)):

$$\eta_i(j) := \mathbb{P}_\theta(S_i = j | X_i = x_i) \tag{3}$$

- *Prediction*: the purpose is to predict a response variable Y from an explanatory variable X. The dependent variable Y can either be discrete, taking values in classes $\{1, \ldots, G\}$ (*classification* task) or continuous (*regression* task). In this paper, we do not extensively discuss application of mixture models to regression purposes but refer the reader to Bouveyron and Girard (2009) for mixture classification and Shimizu and Kaneko (2020) for mixtures of regression models.

In section Univariate and multivariate Gaussian distributions in the context of mixture models, we describe the most commonly used family, the Gaussian Mixture Model (GMM). Then, we present the MLE estimation of the parameters of a GMM, introducing the classic EM algorithm in section Parameter estimation in finite mixtures models. Finally, we introduce bootstrap methods used to evaluate the quality of the estimation and metrics used for the selection of the best model in respectively appendices *Derivation of confidence intervals in GMMs* and *Model selection*.

Univariate and multivariate Gaussian distributions in the context of mixture models

We focus our study on the finite Gaussian mixture models (GMM) in which we suppose that each of the k components follows a Gaussian distribution.

We recall below the definition of the Gaussian distribution in both univariate and multivariate context. In the finite univariate Gaussian mixture model, the distribution of each component $f_{\zeta j}(X)$ is given by the following univariate Gaussian p.d.f. (probability density function) (Equation (4)):

$$f_{\zeta j}(X = x) = \varphi_{\zeta_j}(x|\mu_j, \sigma_j) := \frac{1}{\sqrt{2\pi}\sigma_j} \exp^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}} \tag{4}$$

which we note: $X \sim \mathcal{N}(\mu_j, \sigma_j)$.

In the univariate case, the parameters to be inferred from each component, ζ_j, are: μ_j, the *location* parameter (equal to the mean of the distribution) and σ_j, the *scale* parameter (equal to the standard deviation of the distribution with a Gaussian distribution).

Following parsimonious parametrisations with respect to univariate GMMs are often considered:

- *homoscedascity*: variance is considered equal for all components, $\sigma_j = \sigma, \forall j \in \{1, \ldots, k\}$, as opposed to heteroscedascity where each sub-population has its unique variability.
- *equi-proportion* among all mixtures: $p_j = \frac{1}{k} \quad j \in \{1, \ldots, k\}$ [1]

In the finite multivariate Gaussian mixture model, the distribution $f_{\zeta j}(\boldsymbol{X})$ of each component j, where $\boldsymbol{X} \in \mathbb{R}^D = (X_1, \ldots, X_D)^\top$ is a multivariate random variable of dimension D, is given by the following multivariate Gaussian p.d.f. (probability density function) (Equation (5)):

$$f_{\zeta j}(\boldsymbol{X} = \boldsymbol{x}) = \det(2\pi\boldsymbol{\Sigma}_j)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_j)\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x} - \boldsymbol{\mu}_j)^\top\right) \tag{5}$$

which we note $\boldsymbol{X} \sim \mathcal{N}_D(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$. The parameters to be estimated for each component can be decomposed into:

- $\boldsymbol{\mu}_j = \begin{pmatrix} \mu_{1j} \\ \vdots \\ \mu_{Dj} \end{pmatrix} \in \mathbb{R}^D$, the D-dimensional mean vector.
- $\boldsymbol{\Sigma}_j$, the $\mathcal{M}_D(\mathbb{R})$ positive-definite [2]covariance matrix, whose diagonal terms are the individual variance of each feature and the off-diagonal terms the pairwise covariance terms.

Three families of multivariate GMMs are often considered:

- the *spherical* family, $\boldsymbol{\Sigma}_j = \sigma_j^2 \boldsymbol{I}_D$, with $\sigma_j \in \mathbb{R}_+^*$, refers to GMMs whose covariance matrix is diagonal with an unique standard deviation term. The corresponding volume representation is a $D-$*hypersphere* of radius σ_j.

[1] A rarer constraint considered implies to enforce a linear constraint over the clusters' means, of the following general form: $\sum_{j=1}^{k} a_j \mu_j = 0$, with $\{a_1, \ldots, a_k\}$. For instance, the R package **epigenomix** considers a $k = 3$ component mixture in the context of transcriptomic (differential analyses) and epigenetics (histone modification) to automatically identify undifferentiated, over and under-expressed genes between case and control samples. A common constraint then is to enforce the distribution of fold changes corresponding to the undifferentiated expressed genes to have a distribution centred on 0. Combining equality of means and equality of variances is irrelevant, as the model is then degenerate. Additionally, setting constraints on the means makes the estimation of the parameters challenging, as detailed in Appendix *Extensions of the EM algorithm to overcome its limitations*.

[2] The positive-definiteness constraint can be interpreted from a probabilistic point of view as a necessary condition such that the generalised integral of the multivariate distribution is defined and sum-to-one over $\mathbb{R}$ or from the statistical definition of the covariance. A symmetric real matrix $\boldsymbol{X}$ of rank D is said to be *positive-definite* if for any non-zero vector $\mathbf{v}, \in \mathbb{R}^D$, the following constraint $\mathbf{v}^\top \boldsymbol{X}\mathbf{v} > 0$ is enforced.

- the *diagonal* family, $\mathbf{\Sigma}_j = \text{diag}\left(\sigma_{1j}^2, \ldots, \sigma_{1D}^2\right)$, with $\sigma_j \in \mathbb{R}_+^D$, refers to GMMs whose covariance matrix is diagonal. Its associated volume representation is an ellipsoid whose main axes are aligned with the D canonical basis of $\mathbb{R}^D$. Of note, the null constraint imposed over the off-diagonal terms in the spherical and diagonal families imply that the multivariate distribution can be further decomposed and analysed as the product of univariate independent Gaussian realisations.
- the *ellipsoidal* family, also named the *general* family, refer to GMMs whose covariance matrix, $\mathbf{\Sigma}_j$, can be any arbitrary positive-definite $D \times D$ matrix. Thus, the corresponding clusters for each component J are ellipsoidal, centred at the mean vector $\boldsymbol{\mu}_j$, and volume and orientation respectively determined by the eigenvalues and the eigenvectors of the covariance matrix $\mathbf{\Sigma}_j$.

In the multivariate setting, the volume, shape, and orientation of the covariances can be constrained to be equal or variable across clusters, generating 14 possible parametrisations with different geometric characteristics (Banfield and Raftery 1993; Celeux and Govaert 1992). We review them in Appendix *Parameters estimation in a high-dimensional context* and Table 5. Of note, the correlation matrix can be easily derived from the covariance matrix with the following normalisation: $\text{cor}(\boldsymbol{X}) = \left(\frac{\text{cov}(x_l, x_m)}{\sqrt{\text{var}(x_l)} \times \sqrt{\text{var}(x_m)}}\right)_{(l,m) \in D \times D}$. Correlation if strictly included between -1 and 1, the strength of the correlation is given by its absolute value while the type of the interaction is returned by its sign. A correlation of 1 or -1 between two features indicates a strictly linear relationship.

For the sake of simplicity and tractability, we will only consider the fully unconstrained model in both the univariate (heteroscedastic and unbalanced classes) and multivariate dimension (unbalanced and complete covariance matrices for each cluster) in the remainder of our paper.

Parameter estimation in finite mixtures models

A common way for estimating the parameters of a parametric distribution is the *maximum likelihood estimation* (MLE) method. It consists in estimating the distribution parameters by maximising the likelihood, or equivalently the log-likelihood of a sample. In what follows, $\ell(\theta|x_{1:n}) = \log(f(x_{1:n}|\theta))$ is the log-likelihood of a n-sample. When all observations are independent, it simplifies to $\ell(\theta|x_{1:n}) = \sum_{i=1}^{n} \log(f(x_i|\theta))$. The MLE consists in finding the parameter estimate $\hat{\theta}$ maximising the log-likelihood $\hat{\theta} = \arg\max \ell(\theta|x_{1:n})$.

Recovering the maximum of a function is generally performed by determining from which values its derivative cancels. The MLE in GMMs has interesting properties, as opposed to the *moment estimation* method: it is a consistent, asymptotically efficient and unbiased estimator (J. Chen 2016; McLachlan and Peel 2000).

When S is completely observed, for pairs of observations $(x_{1:n}, s_{1:n})$, the log-likelihood of a finite mixture model is simply given by Equation (6):

$$\ell(\theta|X_{1:n} = x_{1:n}, S_{1:n} = s_{1:n}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \left[\log\left(f_{\zeta_j}(x_i, s_i = j)\right) + \log(p_j)\right]_{\mathbf{1}_{s_i=j}} \tag{6}$$

where an analytical solution can be computed provided that a closed-form estimate exists to retrieve the parameters ζ_j for each components' parametric distribution. The MLE maximisation, in this context, amounts to estimate clusterwise each components' parameter, ζ_j while the corresponding proportions, p_j, is simply the ratio of the observations assigned to cluster j over the total number of observations n.

However, when S is unobserved, the log-likelihood, qualified as incomplete with respect to the previous case, is given by Equation (7):

$$\ell(\theta|x_{1:n}) = \sum_{i=1}^{n} \log\left(\underbrace{\sum_{j=1}^{k} p_j f_{\zeta_j}(x_i)}_{\text{sum of of logs}}\right) \tag{7}$$

The sum of terms embed in the log function (see underbrace section in Equation (7)) makes it intractable in practice to derive the null values of its corresponding derivative. Thus, no closed form of the MLE is available, including for the basic univariate GMM model. This is why most parameter estimation methods derive instead from the *EM algorithm*, first described in Dempster, Laird, and Rubin (1977). We detail in the next section its main theoretical properties, the reasons of its popularity as well as its main limitations.

The EM algorithm

When both S and the parameters of the model are unknown, no closed-form solution exists to jointly optimise the log-likelihood (Equation (7)) parametrised by (θ, S). However, when either S or θ are known, the estimation of the other parameters is straightforward. Hence, the general principle of EM-like algorithms is splitting this complex non-closed joint MLE estimation of (S, θ) into the iterative estimation of S_q from $(\hat{\theta}_{q-1}, X)$ (expectation phase, or *E-step* of the algorithm) and the estimation of $\hat{\theta}_q$ from (S_q, X) (maximisation phase, or *M-step*), with $\hat{\theta}_{q-1}$ being the estimated parameters at the previous step $q-1$, until we reach the convergence.

The EM algorithm sets itself apart from other commonly used methods by taking into account all possible values taken by the latent variable S. To do so, it computes the expected value of the log likelihood of θ, conditioned by the posterior distribution $\mathbb{P}_{\hat{\theta}_{q-1}}(S|X)$, also named as the *auxiliary function*. Making profit of the independence assumption between the observations of a mixture model, the general formula of this proxy function of the incomplete log-likelihood is given in finite mixture models by Equation (8).

$$\begin{aligned} Q(\theta|\hat{\theta}_{q-1}) &:= \mathbb{E}_{S_{1:n}|X_{1:n},\hat{\theta}_{q-1}}\left[\ell(\theta|X_{1:n}, S_{1:n})\right] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{k} \eta_i(j)\left(\log(p_j) + \log(\mathbb{P}(X_i|S_i = j, \theta))\right) \end{aligned} \tag{8}$$

with $\hat{\theta}_{q-1} = \hat{\theta}$ the current estimated parameter value.

In practice, the EM algorithm consists in performing alternatively E-step and M-step until convergence. We supplied below a pseudocode version:

The EM algorithm

- *step E*: determine the posterior probability function $\eta_i(j)$ for each observation of X for each possible discrete latent class, using the initial estimates $\hat{\theta}_0$ at step $q = 0$, or the previously computed estimates $\hat{\theta}_{q-1}$. The general formula is given by Equation (9):

$$\eta_i(j) = \frac{p_j f_{\zeta_j}(x_i)}{\sum_{j=1}^{k} p_j f_{\zeta_j}(x_i)} \tag{9}$$

- *step M*: compute the mapping function $\hat{\theta}_q := M(\theta|\hat{\theta}_{q-1}) = \arg\max Q(\theta|\hat{\theta}_{q-1})$ which maximises the auxiliary function. One way of retrieving the MLE associated to the auxiliary function is to determine the roots of its derivative, namely solving Equation (10)[a]:

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \theta} = 0 \tag{10}$$

[a] To ensure that we reach a maximum, we should assert that the Hessian matrix evaluated at the MLE is indeed negative definite.

Interestingly, the decomposition of the incomplete log-likelihood associated to a mixture model $\ell(\theta|X)$ reveals an entropy term and the so-called auxiliary function (Dempster, Laird, and Rubin 1977). It can be used to prove that maximising the auxiliary function at each step induces a bounded increase of the incomplete log-likelihood. Namely, the convergence of the EM algorithm, defined by comparisons of consecutive log-likelihood, is guaranteed, provided the mapping function returns the maximum of the auxiliary function. Yet, the convergence of the series of estimated parameters $(\theta_q)_{q\geq 0} \underset{i\to+\infty}{\longrightarrow} \hat{\theta}$ is harder to prove but is considered asserted for the probability family of *exponential laws* (a superset of the Gaussian family), as stated in Dempster, Laird, and Rubin (1977).

Additionally, the EM algorithm is *deterministic*, meaning that for a given initial estimate θ_0 the parameters returned by the algorithm at a given step q are fixed. Yet, it requires the user to provide an initial guess θ_0 on the estimate parameters and to set the number of components of the mixture. We review some classic initialisation methods in Initialisation of the EM algorithm and some algorithms used to overcome the main limitations of the EM algorithm in the Appendix *Extensions of the EM algorithm to overcome its limitations*.

Finally, the prevalent choice of Gaussian distributions to model the distribution of random ob-

servations proceeds from a strong set of interesting properties and from their strong tractability. In particular, Geary (1936) has shown that the Normal distribution is the only distribution for which the Cochran's theorem (Cochran 1934) is guaranteed, namely for which the the mean and variance of the sample are independent of each other. Additionally, similar to any distribution proceeding from the exponential family, the MLE statistic is *sufficient*[3].

Initialisation of the EM algorithm

EM-like algorithms require an initial parameter estimate θ_0 to optimise the maximum likelihood. *Initialisation* is a crucial step, as a bad initialisation would possibly lead to a local sub-optimal solution or trap the algorithm in the boundary of the parameter space. The simplest initialisation methods do not require any other initialisation algorithm, while meta-methods include as well an initialisation step. The commonly-used initialisation methods are:

- The *Model-based Hierarchical Agglomerative Clustering* (MBHC) is an agglomerative hierarchical clustering based on MLE criteria applied to GMMs (Scrucca and Raftery 2015). First, the MBHC is initialised by assigning each observation to its own cluster. Next, the pair of clusters that maximises the likelihood of the underlying statistical model among all possible pairs is merged. This procedure is repeated until all clusters are merged. The final resulting clusters are then simply the last k cuts of the resulting dendrogram. In the homoscedastic case in the univariate setting, and the diagonal family with shared variance across all components, the underlying merging criteria reveals similar to the *Ward's criterion*, with the merged pair of clusters being the one minimising the sum of squares. As opposed to other initialisation methods described after, MBHC is a deterministic method that hence does not require careful calibration of hyperparameters. However, as acknowledged by the creator of the method itself of the method (Fraley 1998), the resulting partitions are generally suboptimal compared to other initialisation methods.
- The standard *random* initialisation, used classically for the initialisation step of the k-means algorithm, consists in randomly selecting k distinct observations, referred to as *centroids* and then assigns each observation to the closest centroid (Biernacki, Celeux, and Govaert 2003). Doing so is close from the C-step of the CEM algorithm. This is the method used in this paper, unless otherwise stated. Alternative versions of this method have been developed: for instance, the package **mixtools** draws the proportions of the components from a Dirichlet distribution, whose main advantage lies on respecting the unit simplex constraint (Equation (2))[4], but uses binning methods to guess the means and standard deviations of the components. In paper Kwedlo (2013), the means of the components are randomly chosen but with additional constraint of maximising the Mahalanobis distance between the selected centroids. This enables to cover a larger portion of the parameters' space.
- k-means is a CEM algorithm, in which the additional assumption of balanced classes and homoscedascity implies that each observation in the E-step is assigned to the cluster with the nearest mean (the one with the shortest Euclidean distance). K-means is initialised by randomly selecting k points, the *centroids*. It is often chosen for its fast convergence and memory-saving consumption.
- The *quantile* method sorts each observation x_i by increasing order and splits them into equi-balanced quantiles of size $1/k$. Then, all observations for a given quantile are assumed to belong to the same component. [5]
- The *Rough-Enhanced-Bayes mixture* (REBMIX) algorithm is implemented in the **rebmix** (Nagode 2022) package and the complete pseudo-code is described thoroughly in (Nagode 2015; Panic, Klemenc, and Nagode 2020). First, the observations are processed using one of the three available methods: k-nearest neighbours (KNN), Parzen kernel density estimation or binned intervals, the method we used for this paper. In this method, firstly, data are binned in $\sqrt{n}^D$ intervals of equal lengths. Then, the mode of one of the components' distribution is given by the

[3]The Pitman–Koopman–Darmois theorem (Koopman 1936) states that only the exponential family provides distributions whose statistic can summarize arbitrary amounts of iid draws using a finite number of values

[4]Without prior knowledge favouring one component over another, the Dirichlet distribution is generally parametrised by $\alpha = \frac{1}{k}$, implicitly stating that any observation has equal chance to proceed from a given cluster. In that case, the corresponding distribution is parametrised by a single scalar value α, called the *concentration parameter*.

[5]This method is only available in the univariate framework, since it is not possible to define a unique partition of the observable space into k-splits. For example, in bivariate setting, a binning with $k = 2$ components on each axis leads to a total of $2 \times 2 = 4$ binned regions, which raises the selection issue of the best k hyper-squared volumes for the initial parameters estimation. More generally, $\binom{D}{k}$ binning choices are possible in the multivariate setting.

centre of the interval with the highest frequency. This interval is also used to define a "rough" parameter estimation of its associated component. All the other observations and intervals are then iteratively assigned to the currently estimated component or to the residual components that were not estimated. The algorithm switches to the estimation of another component when at least 25% of the currently non assigned intervals are associated to the currently estimated component. The decision of assigning an interval to either a currently estimated component or residuals relies on the measurement of the deviation between the observed and the expected frequency of the interval, a low value supporting the choice of the algorithm to assign the interval to the currently estimated component. Finally, all intervals assigned to the estimated component are used to determine the parameters of the associated Gaussian distribution (computation of the first and second moments of the distribution). Since this step relies on a higher number of observations to estimate the parameters, Nagode (2015) refers to it as "enhanced" components parameter estimation. The algorithm stops when all intervals are assigned to a cluster and the parameters of the several distributions' components are estimated. Accordingly, the rebmix algorithm can be interpreted as a natural and more general extension of the quantile method with a more rigorous statistical support. Two drawbacks of the algorithm are the need of an intensive calibration of the hyperparameters and its inadequacy for the estimation of highly overlapping mixture distributions (uncertainty of the cluster assignment is not taken into account).[6].

- The *meta-methods* consist generally in short runs of EM-like algorithms, namely CEM, SEM and EM (see Appendix B: *Extensions of the EM algorithm to overcome its limitation*), with alleviated convergence criterion. The main idea is to use several random initial estimates with shorter runs of the algorithm to explore a larger parameter space, and avoid being trapped in a local maximum. Yet, they highly depend on the choice of the initialisation algorithm to start the optimisation (Biernacki, Celeux, and Govaert 2003).
- In the high-dimensional setting, if the number of dimensions D exceeds the number of observations n, all previous methods must be adjusted, usually by first projecting the dataset into a smaller, suitable subspace and then inferring prior parameters in it. In particular, **EMMIXmfa**, in the mixture of common factor analysers (MCFA) approach, initialises the shared projection matrix $\boldsymbol{Q}$ by either keeping the first d eigen vectors generated from standard principal component analysis or uses custom random initialisations (described in further details in the Appendix of Baek, McLachlan, and Flack (2010)).

After this theoretical introduction, we evaluate empirically the computational and statistical performances of the R packages in relation with the choice of the initialisation algorithm and the complexity of the simulation in the next section. The method used to compare the seven reviewed packages is detailed in Methods, while the key results are reported in section Results. We conclude by providing a general simplified framework to select the combination of package and initialisation method best suited to its objectives and the nature of the distribution of the dataset.

A comprehensive benchmark comparing estimation performance of GMMs

We searched CRAN and Bioconductor mirrors for packages that can retrieve parameters of GMM models. Briefly, out of 54 packages dealing with GMMs estimation, we focused on seven packages that all estimate the MLE in GMMs using the EM algorithm, were recently updated and let the user provide its own initialisation estimates: **bgmm**, **EMCluster**, **flexmix**, **GMKMcharlie**, **mclust**, **mixtools** and **Rmixmod**. The complete inclusion process is detailed in Appendix C, *the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms*. The flowchart summarising our choices is represented in Figure 1.

In parallel, we include two additional packages dedicated specifically to high-dimensional settings (they are not tailored to deal with univariate or bivariate dimensions), namely **EMMIXmfa** (Rathnayake et al. 2019) and **HDclassif** (Berge, Bouveyron, and Girard 2019) to compare their performance with standard multivariate approaches in complex, but non degenerate cases. We summarise the main features and use cases of the seven + two reviewed packages in Table 1. The three most commonly used packages are **mixtools**, **mclust** and **flexmix**. However, the **mclust** package is by far the most

[6]The method we describe here to preprocess the observations in order to estimate the empirical density estimation, namely the "histogram method" is not well suited for high dimensional data, as the exponential growth of the volume with respect to dimensionality leads to data sparsity, related to the well-known issue of the "curse of dimensionality". Indeed, $\sqrt{n}^D$ distinct intervals will be parsed by the method and the probability with an increasing number of features and decreasing number of observations that no clear local maximum emerges converges to 1. In high-dimensional context, the Parzen window or the KNN method should be favoured, see (Nagode 2015), p. 16.

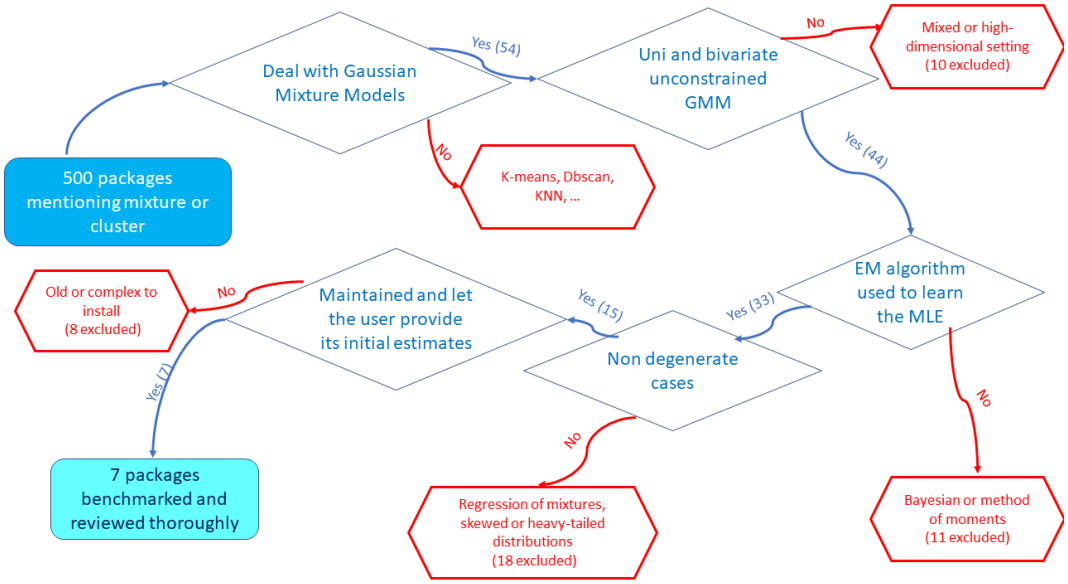

Figure 1: A minimal roadmap used for the selection of the packages reviewed in our benchmark.

complete with many features provided to visualise and evaluate the quality of the GMM estimate. **bgmm** requires the greatest number of packages for its installation, making its upkeep a harder task, while **mclust** only depends of base R packages implemented by default in each new R version. Additionally, in parallel to clustering tasks, **flexmix** and **mixtools** packages perform regression of mixtures and implement mixture models using other parametric distributions or non-parametric methods via kernel-density estimation.

Table 1: Main features of the reviewed packages, sorted by decreasing number of daily downloads. *Downloads per day* returns the daily average number of downloads for each package on the last 2 years. *Recursive dependencies* column counts the complete set of non-base packages required, as first-order dependencies may require as well installation of other packages.

Package	Version	Regression	Implemented models	Downloads per day	Last update	Imports	Recursive dependencies	Language
mclust	5.4.7	☒	☒	5223	31/10/2022	R (≥ 3.0)	0	Fortran
flexmix	2.3-17	☑	Poisson, binary, non-parametric, semi-parametric	3852	07/06/2022	R (≥ 2.15.0), modeltools, nnet, stats4	3	R
mixtools	1.2.0	☑	multinomial, gamma, Weibull, non-parametric, semi-parametric	178	05/02/2022	R (≥ 3.5.0), kernlab, segmented, survival	6	C
Rmixmod	2.1.5	☒	☒	39	18/10/2022	R (≥ 2.12.0), Rcpp, RcppEigen	4	C++
EMCluster	0.2-13	☒	☒	33	12/08/2022	R (≥ 3.0.1), Matrix	3	C
bgmm	1.8.4	☒	☒	27	10/10/2021	R (≥ 2.0), mvtnorm, combinat	77	R
GMKMcharlie	1.1.1	☒	☒	12	29/05/2021	Rcpp, RcppParallel, RcppArmadillo	3	C++
EMMIXmfa	2.0.11	☒	☒	12	16/12/2019	NA	0	C
HDclassif	2.2.0	☒	☒	35	12/10/2022	rARPACK	13	R

We further detail features specifically related to GMMs in Table 2. We detail row after row its content below:

- The parametrisations used to provide parsimonious estimation of the GMMs are reviewed in Parameter estimation in finite mixtures models and summarised in rows 1 and 2 (Table 2) for the univariate and multivariate setting. We refer to them as *canonical* when both homoscedastic and heteroscedastic parametrisations in the univariate setting, and the 14 parametrisations listed in Supplementary Table 3 in the multivariate setting, are implemented. Adding the additional constraint of equi-balanced clusters results in a total to $14 \times 2 = 28$ distinct models and $2 \times 2 = 4$ parametrisations, respectively in the univariate and multivariate setting. Since **EMMIXmfa** and **HDclassif** are dedicated to the analysis of high-dimensional datasets, they project the observations in a smaller subspace and are not available in the univariate setting. Given an user-defined or prior computed intrinsic dimension, we can imagine using any of the standard parametrisations available for instance in the **mclust** package, and listed in Appendix

Parsimonious parametrisation of multivariate GMMs. In addition, **HDclassif** allows each cluster j to be represented with its own subspace intrinsic dimension d_j, as we describe in further details in Appendix *Parameters estimation in a high-dimensional context*.

- The EM algorithm is the most commonly used algorithm to retrieve the parameters of the GMMs but faster or variants complementary of the EM algorithm are reviewed in Appendix B: *Extensions of the EM algorithm to overcome its limitations* and row 3 of Table 2. Especially, GMMs estimation is particularly impacted by the presence of outliers, justifying a specific benchmark (see Appendix *A small simulation to evaluate the impact of outliers*). We briefly review the most common initialisation algorithms in section Initialisation of the EM algorithm and row 4 of Table 2, a necessary and tedious task for both the EM algorithm and its alternatives.
- To select the best parametrisations and number of components that fit the mixture, several metrics are provided by the reviewed packages (*Model selection* and row 5). Due to the complexity of computing the true distribution of the parameters estimated, bootstrap methods are commonly used used to derive confidence intervals (see Appendix *Derivation of confidence intervals in GMMs* and row 6 in Table 2).
- Six packages supply several visualisations, summarised in the last row of Table 2, to display either the distributions corresponding to the estimated parameters or compare quickly the performance between the packages. However, **mclust** is by far the most complete one with performance plots to quickly set apart several parametrisations with respect to a performance metric, density plots (in the univariate setting) and isodensity plots (bi-dimensional in the bivariate setting or in higher dimensions after appropriate dimensionality reduction), with the additional possibility of integrating custom confidence intervals and critical regions, and finally boxplot bootstrap representations to take in at a glance the distribution of the benchmarked estimated parameters.

High-dimensional packages provide specific representations adjusted to the high-dimensional settings, notably allowing the user to visualise the projected factorial representation of its dataset in a two or three-dimensional subspace. They also provide specialised performance plots, notably scree plots or BIC scatter plots to represent in a compact way numerous projections and parametrisations.

Table 2: Custom features associated to GMMs estimation for any of the benchmarked packages.

	mclust	**flexmix**	**mixtools**	**Rmixmod**	**EMCluster**	**bgmm**	**GMKMcharlie**	**EMMIXmfa**	**HDclassif**
Models Available (univariate)	canonical	unconstrained	canonical	canonical	unconstrained	canonical	unconstrained	NA	NA
Models Available (multivariate)	canonical	unconstrained diagonal or general	unconstrained	canonical	unconstrained	4 models (diagonal and general, either component specific or global)	unconstrained	4 models (either component-wise or common, on the intrinsic and diagonal residual error covariance matrices)	canonical on projected dimension
Variants of the EM algorithm	VBEM	SEM, CEM	ECM	SEM, CEM	☒	☒	CW-EM, MML	AECM	SEM, CEM
Initialisation	hierarchical clustering, quantile	short-EM, random	random	random, short-EM, CEM, SEM	random, short-EM	k-means, quantile	k-means	k-means, random, heuristic	short-EM, random, k-means
Model or Cluster Selection	BIC, ICL, LRTS	AIC, BIC, ICL	AIC, BIC, ICL, CAIC, LRTS	BIC, ICL, NEC	AIC, BIC, ICL, CLC	GIC	☒	☒	BIC, ICL, CV
Bootstrap Confidence Intervals	☑	☑	☑	☒	☒	☒	☒	☒	☒
Visualisation	performance, histograms and boxplots of bootstrapped estimates, density plots (univariate), scatter plots with uncertainity regions and boundaries (bivariate), isodensity (bivariate , 2D projected PCA or selecting coordinates)	☒	density curves	density curves, scatter plots with uncertainty boundaries	☒	performance, scatter plots with uncertainty boundaries	☒	projected factorial map	projected factorial map, performance (Cattell's scree plot, BIC performance, slope heuristic)

Methods

In addition to the the seven packages selected for our benchmark, we include a custom R implementation of the EM algorithm used as baseline, referred to as *RGMMBench*, and for the high-dimensional setting we select packages **EMMIXmfa** and **HDclassif**, on the basis of criteria detailed in Appendix C, *General workflow*. Code for *RGMMBench* is provided in Appendix *Application of the EM algorithm to*

GMMs. To compare the statistical performances of these packages, we performed *parametric bootstrap* (*Derivation of confidence intervals in GMMs*) and built an experimental design to cover distinct mixture distributions parameter configurations, using prior user-defined parameters.

For each experiment, we assign each observation to an unique cluster by drawing n labels $S_{1:n}$ from a multinomial distribution whose parameters were the prior user-defined proportions $p = (p_1, \ldots, p_k)$. Then, each observation x_i assigned to hidden component j is drawn respectively using the R function `stats::rnorm()` for the univariate distribution and `MASS::mvrnorm` for the multivariate distribution. The complete code used for simulating data is available on GitHub at RGMMBench. Finally, we obtain an empirical distribution of the estimated parameters by computing the MLE of each randomly generated sample.

For all the packages, we have used the same combination of criterion threshold of 10^{-6} and maximal number, 1000, of iterations as a numerical proof of convergence. We simulated data with $n = 200$ draws in the univariate setting and $n = 500$ in the bivariate framework to lower the probability of generating a sample without drawing any observation from one of the components, especially in case of a highly-unbalanced scenario, look out for specificities in Appendix *Practical details for the implementation of our benchmark*. Unless stated explicitly, we keep the default hyper-parameters and custom global options provided by each package. For instance, the **flexmix** package has a default option, *minprior*, set by default to 0.05 and which removes any component present in the mixture with a ratio below 0.05. Besides, we only implement the fully unconstrained model in both univariate and multivariate settings, as it is the only parametrisation implemented in all the seven packages and the most popular to perform classic GMM clustering, since fewer hypothesis are required.

We compared the packages' performances using five initialisation methods: random, quantile, *k*-means, rebmix and hierarchical clustering in the univariate setting. We benchmarked the same initialisation methods in the multivariate setting, except for the quantile method which has no multivariate equivalent (see section Initialisation of the EM algorithm):

- We used the function `EMCluster::rand.EM()` with 10 random restarts and required minimal individual cluster size of 2 for the random initialisation. The method implemented by **EMCluster** is the most commonly used, described in details in Biernacki, Celeux, and Govaert (2003) and in section Initialisation of the EM algorithm.
- We used the function `stats::kmeans()` function with a 10^{-2} stopping criterion and 200 maximal number of iterations to implement the *k*-means initialisation method. The initial centroid and covariance matrix for each component are then computed by restricting to the sample observations assigned to the corresponding component. The approach is close to the one adopted by the CEM algorithm (see Appendix B: *Extensions of the EM algorithm to overcome its limitations*).
- We use the `mclust::hcV()` function for the MBHC algorithm. This method has two main limitations: just like the *k*-means implementation, it only returns a cluster assignment to each observation instead of the posterior probabilities, and the splitting process to generate the clusters results sometimes in clusters composed of only one observation. In that case, we add a small epsilon to each posterior probability to avoid Dirac degenerated distributions.
- We used in the univariate setting `bgmm::init.model.params` for the quantiles initialisation.
- To implement the rebmix method, we use the `rebmix::REBMIX` function, using the *kernel density estimation* for the estimation of the empirical density distribution coupled with *EMcontrol* set to one to prevent the algorithm from starting EM iterations.
- Any of the seven packages could be used to implement the small EM method. We decided to use the `mixtools::normalmixEM` as it is the closest one to our custom implementation, setting 10 random restarts, a maximal number of iterations of 200 and with an alleviated absolute threshold of 10^{-2}. However, preliminary experiments suggest us to discard the small EM initialisation method, as it tends to smooth the differences between the packages, as illustrated in supplementary Figure 9.

We sum up in Table 3 the general configuration used to run the scripts. Additionally, all simulations were run with the same stable R version 4.0.2 (2020-06-22) using an OS system under Linux and numerical precision of 2.22×10^{-16}.

Preliminary experiments suggested that the quality of the estimation of a GMM is mostly affected by the overlap between components' distribution and level of unbalance between components. We quantified the overlap between two components by the following overlap score (OVL) (see Equation (11)), with a smaller score denoting well-separated components:

$$\text{OVL}(i, j) = \int \min(f_{\zeta_i}(x), f_{\zeta_j}(x))dx \quad \text{with } i \neq j \tag{11}$$

Table 3: Global options shared by all the benchmarked packages.

Initialisation methods	Algorithms	Criterion threshold	Maximal iterations	Number of observations
midrule hc, kmeans, small EM,rebmix, quantiles, random	EM R, Rmixmod, bgmm, mclust, flexmix, EMCluster, mixtools, GMKMCharlie	10^{-6}	1000	100, 200, 500, 1000, 2000, 5000, 10000

We may generalise this definition to k components by averaging the individual components' overlap. We use the function `MixSim::overlap` from the **MixSim** package (Melnykov, Chen, and Maitra 2021) that approximates this quantity using a Monte-Carlo based method (see appendices *An analytic formula of the overlap for univariate Gaussian mixtures* and *Practical details for the implementation of our benchmark* for further details).

The level of imbalance may be evaluated with the entropy measure Equation (12), with equi-balanced clusters having an entropy of 1:

$$H(S) = -\sum_{j=1}^{k} p_j \log_k(p_j) \tag{12}$$

with k the number of components and $p_j = \mathbb{P}(S = j)$ the frequency of class j.

We considered 9 distinct configuration parameters, associated with distinct values of OVL and entropy in the univariate setting, 20 scenarios in the bivariate setting and 16 scenarios in the high-dimensional setting. Briefly, we compute any combination, with the same components means (0, 4, 8 and 12), three sets of ratio parametrisations $[(0.25, 0.25, 0.25, 0.25); (0.2, 0.4, 0.2, 0.2); (0.1, 0.7, 0.1, 0.1)]$ and three variances: $(0.3, 1, 2)$ in the univariate setting. In the bivariate setting, we consider two sets of proportions: $[(0.5, 0.5); (0.9, 0.1)]$, two sets of coordinate centroids: $[(0; 20), (20, 0)]$ and $[(0; 2), (2, 0)]$, the same variance of 1 for each feature and for each component for illustrative purposes of the direct relation linking the correlation and the level of OVL and five correlation sets:$[(-0.8, -0.8); (0.8, -0.8); (-0.8, 0.8); (0.8, 0.8); (0, 0)]$.

And finally, we tested eight scenarios in the high-dimensional framework, setting to $D = 10$ the number of dimensions, playing on the level of overlap, imbalance between the components' proportions and the underlying constraint assigned to the covariance matrix (either fully parametrised or spherical), without explicitly providing this restriction to the packages. Each of the parameter configuration tested in the high-dimensional setting was evaluated with $n = 200$ observations and $n = 2000$ observations. Additionally, instead of defining manually interesting parameters for a high-dimensional scenario, we take profit of the simulation framework set up in `MixSim` function, from the **MixSim** package (Melnykov, Chen, and Maitra 2021). This function returns the user a fully parametrised GMM, with a prior defined level of maximum or average overlap[7].

The complete list of parameters used is reported respectively in Table 4 for the univariate setting, Table 5 for the bivariate setting and 6 for the high-dimensional setting. We benchmarked simulations where the components were alternatively very distinct or instead very overlapping, as well as of equal proportions or instead very unbalanced. The adjustments made to meet the specific requirements of high dimensional packages are detailed in *Practical details for the implementation of our benchmark*.

We report the most significant results and features and the associated recommendations in next section Results.

Results

All the figures and performance overview tables are reported in *Supplementary Figures and Tables in the univariate simulation* for the univariate setting, *Supplementary Figures and Tables in the bivariate simulation* for the bivariate scenario and *Supplementary Figures and Tables in the HD simulation* for the high dimensional scenario.

Balanced and non-overlapping components

In the univariate setting, with balanced components and low OVL (scenario U1 in Table 4), the parameter estimates are identical in most cases across initialisation methods and packages, notably

[7]Unfortunately, as discussed in further details in Appendix *An analytic formula of the overlap for univariate Gaussian mixtures*, the MixSim package does not compute the global distribution overlap, but instead returns the mean of pairwise overlap between any component (however, with two components, these two alternative definitions match.) Finally, it is not possible to set the proportions of the components before the generation of the parameters, except for clusters with equal proportions, and contrary to the expect behaviour of additional parameter *PiLow*, supposed to define the smallest mixing proportion.

the same estimates are returned with *k*-means or rebmix initialisation. However, the random initialisation method leads to a higher variance and bias on the parameter estimates than other methods (Supplementary Figure 4 and Supplementary Table 6), with various optimisations returning only local maxima far from the optimal estimate.

Similar scenarios in the bivariate setting (scenarios B6-B10 in Table 5), with a focus on B6, B7 and B10 visualised in Supplementary Figure 16, feature well-separated and balanced components. Consistent with conclusions from the corresponding univariate scenarios, all benchmarked packages return the same estimates across initialisation methods.

Unbalanced and non-overlapping components

However, with unbalanced classes and low OVL (scenario U7 in 4), the choice of the initialisation method is crucial, with quantiles and random methods yielding more biased estimates and proned to fall in other local maxima. Rebmix initialisation provides the best estimates, with the smallest MSE and bias across packages (Supplementary Figure 5 and supplementary Table 7, always associated with the highest likelihood. Overall, with well-discriminated components, most of the differences on the estimation originate from the choice of initialisation method, while the choice of the package has only small impact.

We detail expensively in our bivariate simulations two scenarios featuring both strongly unbalanced and well-separated components, similarly to scenario U3 in Table 4: the scenarios B12 (Supplementary Figure 12 and Table 12) and B14 (Supplementary Figure 13 and Supplementary Table 13). Similarly, scenarios B16, B17 and B20 (Table 5) with similar characteristics are summarised in supplementary Figure 17. For all these scenarios, neither the initialisation method nor the package have a statistical significant impact on the overall performance.

Similarly, scenarios HD1a-HD4b in Table 6) in the high dimensional setting, display well-separated clusters, with a representative outcome represented in Supplementary Figure 19 and Supplementary Table 16. Consistently with results from the corresponding univariate and bivariate scenarios, most of the previously benchmarked packages return sensibly the same estimates with the hierarchical clustering and *k*-means initialisation. However, **bgmm** and **EMCluster** clearly differ by lower performances with the rebmix initialisation method (however, overall, rebmix performs poorly, regardless of the package used for estimation). Notably, initialisations with the rebmix package tend to display a much larger number of poor estimations, some of which can be identified with the local maxima associated with parameter switching between the two classes. Finally, the two additional packages dedicated to high-dimensional clustering display the worst performances, with **EMMIXmfa** returning the most biased parameters and **HDclassif** the most noisy estimates. **EMMIXmfa** is the only package that returned highly biased estimates of the components' proportions in this setting.

Balanced and overlapping components

When the overlap between components increases, the bias and variability of the estimates tends to increase while the choice of initialisation method becomes becomes valuable. The least biased and noisy estimations with balanced components in the univariate setting (scenario U3 in Table 4) are obtained with the *k*-means initialisation (Supplementary Figure 3 and Table 8) while the rebmix initialisation returns the most biased and noisy estimates. Similar results are found in the bivariate setting with a balanced and highly overlapping two-components GMM (scenarios B1-B5 from Table 5), with the best performance reached with the *k*-means initialisation method, followed by MBHC. This is emphasised in supplementary Figure 16, in the top three most complex scenarios, namely B1, B2 and B5. If the shape of the covariance matrix is pretty well-recovered, no matter the package, the Hellinger distances are significantly higher (and thus the estimate further away from the target distribution) with the random and rebmix methods.

Similarly, in the high-dimensional scenario HD7 of Table 6), presenting balanced but highly overlapping clusters with a full covariance structure, the best performances was obtained with *k*-means initialisation, while the rebmix initialisation returns the most biased and noisy estimates. While **EMMIXmfa** performed well when it provided estimates, it returned an error in most cases (see Column *Success* of supplementary Table 17). The least biased estimates are returned by **mixtools** and **Rmixmod** and the least noisy by **flexmix**, **mclust** and **GMKMCharlie** (smaller MSE). Interestingly, in the high-dimensional setting, the packages **EMCluster** and **bgmm** differentiate from the other packages, exhibiting worse performance. In particular, on Panel E of Figure 20, the components' proportions span the $]0-1[^k$ simplex. On the contrary, the **EMCluster** package, and to a lesser extent, the **bgmm** package, perform surprisingly well when datasets were simulated with an underlying spherical covariance structure, even though the estimation was not performed explicitly with this constraint (Supplementary Table 19). Indeed, it seems like that that the off-diagonal terms tend to converge towards 0, as showcased in Supplementary Figure 21, in Panel C, in which the fourth row from top represents the bootstrap intervals associated to the pairwise covariance between dimension 1 and 2 of each cluster.

Unbalanced and overlapping components

With unbalanced components and high OVL (scenario U9 in Table 4), all packages, no matter the initialisation method, are biased, with a higher variability of the parameter estimates compared to other scenarios. The least biased estimates are obtained with *k*-means or random initialisation, but with a higher variability on the estimates with random initialisation (Supplementary Table 9). However, deepening the analysis to the individual components' parameter estimates reveals that the least unbiased estimate is obtained with rebmix initialisation for the two most distinguishable components, namely clusters 2 and 4 (Supplementary Figure 7 and Table 9), consistently with this method's assumption to use modes to initialize the components (Nagode 2015). With highly-overlapping distributions and unbalanced components, both the choice of the initialisation algorithm and the package have a substantial impact on the quality of the estimation of this mixture.

Two scenarios in our bivariate simulation feature distributions with both strong OVL and unbalanced components. Especially, the scenario B11 (Table 5) has the strongest OVL overall, with notably a risk of wrongly assigning minor component 2 to major component 1 of 0.5 (a random method classifying each observation to cluster 1 or 2 would have the same performance).

First, we observe that the the random and rebmix initialisation methods have similar performance, significantly better than *k*-means or MBHC (Supplementary Figure 11). Specifically, the rebmix method returns the least biased estimates, while the random method is associated with the lowest MSE (respectively for scenarios B11 and B15, the supplementary Tables 11 and 14). Second, the estimates differ across packages only in these two complex scenarios, with packages **Rmixmod** and **mixtools** returning more accurate estimates than the others. It it is particularly emphasised in Scenario B15, where the component-specific covariance matrices are diagonal with same non-null input, and thus should present spherical density distributions. However, only the first class of packages correctly recovers the spherical bivariate 95% confidence regions while they are slightly ellipsoidal with the second class of packages (Panel B, Supplementary Figure 14).

With full covariance structures and unbalanced proportions, as depicted in the high-dimensional Scenario HD8a) and b) of Table 6, the general observations stated in the previous subsection for the high dimensional setting hold, namely that the least biased estimates are returned by packages not specifically designed for high-dimensional data, with the *k*-means initialisation (Supplementary Table 12 and supplementary Figure 22). Furthermore, the pkg{EMCluster} and **bgmm** packages and the two packages dedicated to high-dimensional, perform similarly with $n = 200$ observations (sub-scenario a) and $n = 2000$ observations (sub-scenario b), whereas we would expect narrower and less biased confidence intervals by increasing the number of observations by a factor of 10.

Finally, with spherical covariance structures and unbalanced proportions, the best performances, both in terms of bias and variability, are obtained with **flexmix**, **mclust** and **GMKMCharlie**. Indeed, as detailed later in Conclusions, these packages are more sensitive to the choice of the initialisation method and have a greater tendency to get trapped in the neighbourhood of the initial estimates (Supplementary Table 19 and supplementary Figure 22). Accordingly, *k*-means initialisation performs best since it assumes independent and homoscedastic features for each cluster. Furthermore, **EMMIXmfa** is the package that best estimates the off-diagonal terms in this setting, as highlighted in supplementary Table 19.

Identification of two classes of packages with distinct behaviours

By summarizing the results obtained across all simulations, we identify two classes of packages with distinct behaviours (Figure 2):

- The first class of packages, represented by **Rmixmod** and **mixtools**, returns similar estimates to our baseline EM implementation. The estimates returned by these packages are less biased but at the extent of a higher variability on the estimates. Additionally, with overlapping mixtures, they tend to be slower compared to the second class, since they require additional steps to reach convergence.
- The second class of packages, composed of the other reviewed packages, is more sensitive to the initialisation method. This leads to more biased but less variable estimates, especially when assumptions done by the initialisation algorithm are not met.

Panels A, B and C display, respectively in the univariate, bivariate and high-dimensional setting, the heatmap of the Pearson correlation between the estimated parameters across the benchmarked packages for the most discriminative scenario (the one featuring the most unbalanced and overlapping components): scenario U9, Table 4 in the univariate setting, scenario B11, Table 5, for the bivariate simulation and scenario HD8, Table 6 for the high-dimensional simulation, with the *k*-means initialisation.

We further identified with this representation minor differences for the estimation of the parameters between **Rmixmod** and **mixtools**, while three subgroups can be identified in the second class of packages: the first subset with **bgmm** and **mclust**, the second subset with **EMCluster** and **GMKMcharlie** packages and the **flexmix** package, which clearly stands out from the others, as being

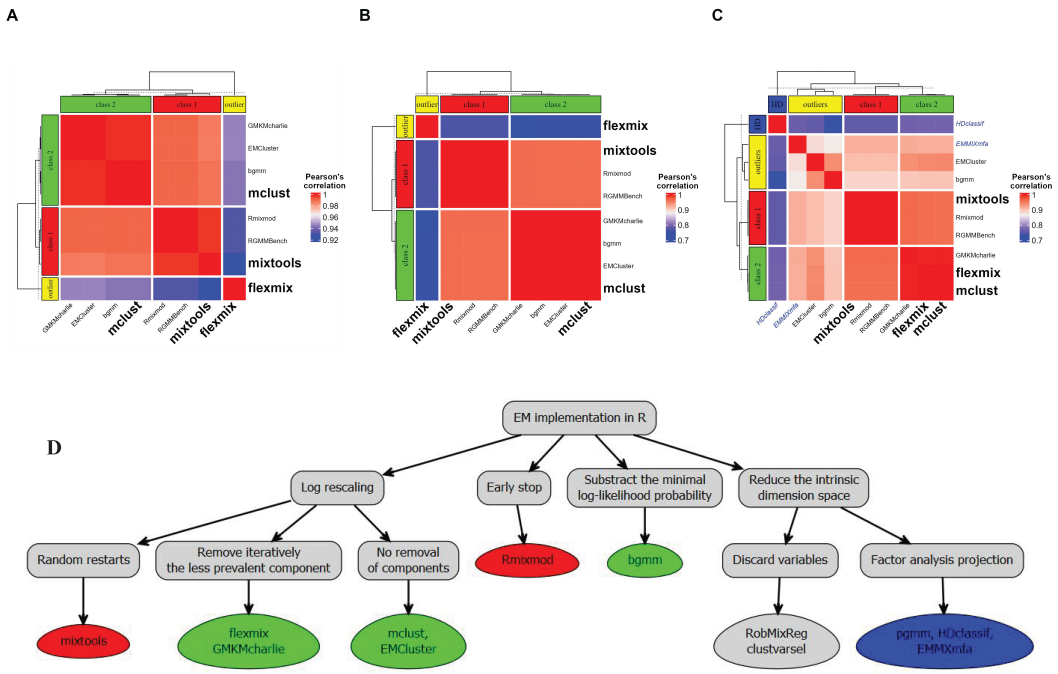

Figure 2: Panels A, B and C show respectively the heatmap of the Pearson correlation in the univariate, bivariate and high-dimensional framework between the parameters estimated by the packages, evaluated for the most discriminating and complex scenario. The correlation matrix was computed using the function `stats::cor` with option *complete* to remove any missing value related to a failed simulation, and the heatmap generated with the Bioconductor package **ComplexHeatmap**. Panel D represents a tree summarising the main differences between the benchmarked packages, in terms of the EM implementation. They are discussed in more detail in Appendix *EM-implementation differences across reviewed packages*.

the one most likely to be trapped at the boundaries of the parameter space. After examining the source codes of the packages, we attribute this differences to custom implementation choices of the EM algorithm, such as the way numerical underflow is managed or the choice of a relative or absolute scale to compare consecutive computed log-likelihoods (see Appendix *EM-implementation differences across reviewed packages* and Panel D, Figure 2). In the high-dimensional setting, the second class of packages showed additional heterogeneity, with **EMCluster** and **bgmm** setting themselves apart from the other three packages.

Failed estimations

Finally, in some cases, neither the specific EM algorithm implemented by each package nor the initialisation method were able to return an estimate with the expected number of components, or even worse fell into a degenerate case (e.g., with infinite or zero variances). In that case, we considered the estimation failed and accordingly we did not include it into the visualizations and the summary metric tables. Most of the failed estimations occurred with the rebmix initialisation. Indeed, an updated version of the package forces the user to provide a set of possible positive integer values for the number of components, with at least a difference of two between the model with the most components and the model with the least components (we therefore set the parameter *cmax* to \$k+1 and *cmin* to $k - 1$). In the most complex scenarios, this constraint leads to an increased risk of returning an overestimation or an underestimation of the actual number of components.

In most cases, this phenomenon occurs with mixtures featuring both strongly unbalanced and highly-overlapping components, with up to 20% initialisations failed for the most difficult scenario B11 (Table 5), 10% for the second most difficult one, namely scenario B15 and on average, an overestimation of three components in 4% of the estimations.

Removing errors proceeding from the initialisation method, only the **flexmix** package failed in returning an estimate matching the user criteria in some scenarios of the univariate and bivariate settings. In both cases, the strong assumption that any cluster with less than 5% of the observations is irrelevant, results in trimming one or more components[8]. This strong agnostic constraint on component proportions led to failures in scenarios featuring strongly overlapping clusters, with up to 20% failed

[8]With a two-components mixture like our bivariate scenario, this even implies to consider an unimodal distribution of the dataset

estimations with the random initialisation method in scenario B11 (Table 5) and 80% failed estimations in the univariate setting[9] with the rebmix initialisation with scenario U9, Table 4.

In a relatively high dimensional framework, as tested on our third benchmark (Table 6), none of the initialisations performed with the random method, carried out with `EMCluster::rand.EM()`, succeed in returning a valid parametrisation. Indeed, over the 16 scenarios tested, the covariance returned during the initialisation was systematically non-positive definite for at least one of the components, violating the properties of covariance matrices. Furthermore, as shown by the comparison of summary metrics with $n = 200$ and $n = 2000$ observations in supplementary Tables 20 and 21, respectively for the simplest scenario HD1 and the most complex one HD8, the rebmix initialisation on the one hand, and the packages dedicated to high dimensionality or those of the second class of packages that show a particular behaviour, present much more failures than the *k*-means or hierarchical clustering initialisation, especially with the first class or the other packages of the second class.

Conclusions

There is a wide diversity of packages that implement the EM algorithm to retrieve the parameters of GMMs. But only few are regularly updated, implement both the unconstrained univariate and multivariate GMM while enabling the user to provide its own initial estimates. Hence, among the 54 packages dealing with GMMs available on CRAN or Bioconductor repositories, we focused our review on 7 packages implementing all these features. We believe that our in-depth review of the packages can help users to quickly find the best package for their clustering pipeline and highlight limitations in the implementation of some packages. Our benchmark covers a much broader range of scenarios than the previously-published studies (Nityasuddhi and Böhning 2003; Lourens et al. 2013; Leytham 1984; Xu and Knight 2010), as we studied the impact of the level of overlap and the imbalance of the mixture proportions on the quality of the estimation.

Interestingly, the EM algorithm sometimes tends to returns biased and inefficient estimators when the components of the GMM under study highly overlap, confirming the experimental observations reported in papers (Lourens et al. 2013; Leytham 1984; Xu and Knight 2010). This seems to contradict the statement from Leytham (1984) that shows the theoretical asymptotic consistency, unbiasedness and efficiency of the MLE of GMMs. However, the key assumption of the demonstration is the definition of a local environment, implying to define borders for which the conditions of the theorem hold. Thus, it can expected that the EM algorithm sometimes fail to reach the global maximum of the distribution, with multiple local extremes.

When all components are well-separated or with a relatively small number of components (three or fewer), we found that the best estimation (lowest MSE and bias) is reached with the latest initialisation method developed, namely the rebmix one. Notably, the global maximum is always properly found in our simulations with distinguishable components. Yet, with overlapping components, the least biased and variable estimates overall are obtained with *k*-means initialisation, enforcing the robustness of the method while the assumptions for using it are not met. On the contrary, with unbalanced and numerous components (above three), the quantiles initialisation leads to the most biased estimates while the rebmix initialisation induces the highest variability. Indeed, rebmix initialisation is not fitted for highly overlapping mixtures, since one of its most restrictive assumption is that each generated interval of the empirical mixture distribution can be associated unambiguously to a component (see Initialisation of the EM algorithm and p15 in Nagode (2015)). Furthermore, rebmix is not particularly adjusted to deal with high-dimensionality, displaying systematically poorer performance compared to other initialisation strategies, such as *k*-means or hierarchical clustering, as illustrated by the summary metrics listed in Appendix *Supplementary Figures and Tables in the HD simulation*. Higher risk of returning a sub-optimal extremum likely arises from the increased data sparsity in high dimensional datasets, which grows as the square root of the number of dimensions $\sqrt{D}$ Convergence of distance definitions. Thus, we expect that most of the equally-large intervals binning the sampling space and that are used to initiate the rebmix algorithm contain either no or only observation, preventing from retrieving the numerically defined mode of the distribution and increasing the risk of initiating the algorithm in a spurious neighbourhood.

About the remaining initialisation strategies, we observed that random initialisation yields for some simulations, even in the well-separated case, highly biased estimates, far from the true parameters. Consistent with our observations, it was shown in Jin et al. (2016) that the probability for the EM algorithm to converge from randomly initialised estimates to a local suboptimal maximum is non null above two components, increasing with the number of components. Additionally, the local maximum of the likelihood function obtained can be arbitrarily worse than the global maximum. Finally, hierarchical clustering does not take into account any uncertainty on the assignment for an

[9] the gap proceeds from the stronger level of imbalance and the greater number of components

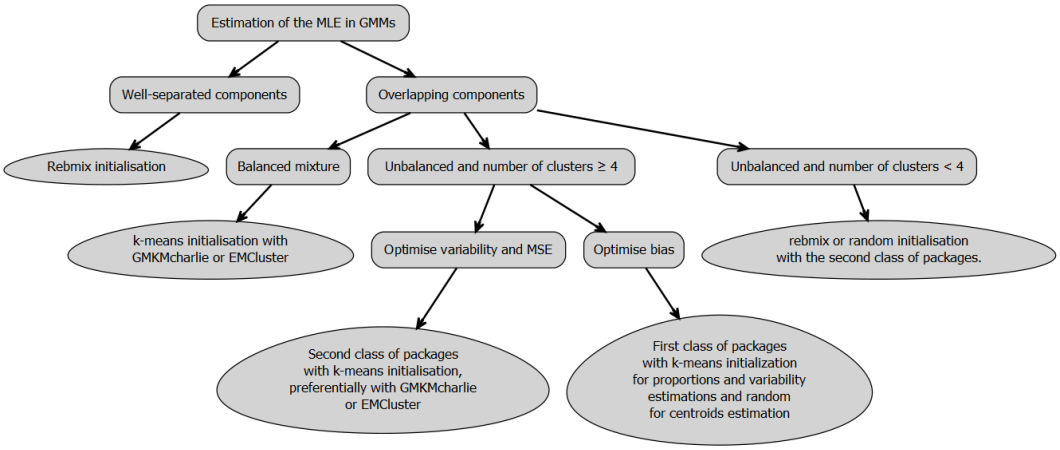

Figure 3: A decision tree to select the best combination of package and initialisation method with respect to the main characteristics of the mixture. It's worth pointing that in both univariate and low dimension multivariate settings, the recommandations are similar.

observation to a given class, which explains its rather bad performances with overlapping components. Overall, there is always an initialisation algorithm performing better than the hierarchical clustering, whereas it is by far the slowest and most computationally intensive initialisation method, as best illustrated in supplementary Figure 10.

Our study reveals differences in the estimates obtained across different packages while the EM algorithm is supposed to be deterministic. We were able to link these differences with custom choices of EM implementations across the benchmarked packages. Specifically, two classes set apart, with distinct choices to deal with some limitations of the EM algorithm: the first one, represented by **mixtools** and **Rmixmod**, tend to provide smaller biased estimates, less sensitive to the choice of initialisation method but with higher variability and longer running times required to reach the convergence. The second one, composed of the remaining packages, provide estimates with reduced MSE, but at the extent of a higher bias on the estimates. One plausible explanation is that the first class of packages, comparing absolute iterations of the function to be maximised, tends on average to perform more iterations. The estimated results are accordingly more consistent and closer to the true MLE estimation but at the expense of an increased risk of getting trapped in a local extremum or a plateau, explaining the greater number of outliers observed. Among them, **flexmix** stands out by choosing an unbiased but non MLE-estimate of the covariance matrix, without any clear improvement of the overall performance in our simulations.

Based on these results, we design a decision tree indicating the best choice of package and initialisation method in relation with the shape of the distribution, displayed in Figure 3. Interestingly, our conclusions are consistent between the univariate and bivariate settings. Furthermore, most of the general recommendations on the best choices of packages with respect to the characteristics of the mixture model generally hold in a relatively higher dimensional setting[10]. From this, we assume that projection into a lower-dimensional space is only beneficial in a really high-dimensional setting, for example when the number of dimensions exceeds the number of observations, or when unrestricted parameter estimation (with the full covariance structure) is practically infeasible for computational reasons.

Comparing all these packages suggest several improvements.

1. The use of C++ code speeds up the convergence of the EM algorithm compared to a full R implementation.
2. All packages dealing with GMMs should use *k*-means for overlapping, complex mixtures and rebmix initialisation for well-separated components, provided that the dimension of the dataset is relatively small. The final choice between these two could be set after a first run or visual inspection aiming at determining roughly the level of entropy across mixture proportions and the degree of overlap between components.
3. The packages should allow the user to set their own termination criteria (either relative or absolute log-likelihood or over the estimates after normalisation). Interestingly, **EMMIXmfa** is the only package among those examined that allows the user to consider an absolute or relative

[10]We should note, however, that a larger sample space revealed that the packages **bgmm** and **EMCluster** display more biased and noisy parameters compared to the other packages benchmarked and that their performance was surprisingly unaffected by the number of simulated realisations

convergence endpoint of the EM algorithm, through the `conv_measure` attribute, with `diff` and `ratio` options respectively.

4. With a great number of components or complex overlapping distributions, the optimal package should integrate partial information when available or a Bayesian estimation of the estimates.

While **mclust** appeared as the most complete package to model GMMs in R, none of the packages reviewed in this report features all the characteristics mentioned above. We thus strongly believe that our observations will help users identify the most suitable packages and parameters for their analyses and guide the development or updates of future packages.

Bibliography

Baek, Jangsun, Geoffrey J. McLachlan, and Lloyd K. Flack. 2010. "Mixtures of Factor Analyzers with Common Factor Loadings: Applications to the Clustering and Visualization of High-Dimensional Data." *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2009.149.

Banfield, Jeffrey D., and Adrian E. Raftery. 1993. "Model-Based Gaussian and Non-Gaussian Clustering." *Biometrics*. https://doi.org/10.2307/2532201.

Berge, Laurent, Charles Bouveyron, and Stephane Girard. 2019. *HDclassif: High Dimensional Supervised Classification and Clustering*.

Biernacki, Christophe, Gilles Celeux, and Gérard Govaert. 2003. "Choosing Starting Values for the EM Algorithm for Getting the Highest Likelihood in Multivariate Gaussian Mixture Models." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/S0167-9473(02)00163-9.

Bouveyron, Charles, and Stéphane Girard. 2009. "Robust Supervised Classification with Mixture Models: Learning from Data with Uncertain Labels." *Pattern Recognition*. https://doi.org/10.1016/j.patcog.2009.03.027.

Celeux, Gilles, and Gérard Govaert. 1992. "A Classification EM Algorithm for Clustering and Two Stochastic Versions." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/0167-9473(92)90042-E.

Chen, Jiahua. 2016. "Consistency of the MLE Under Mixture Models." https://doi.org/10.1214/16-STS578.

Chen, Wei-Chen, and Ranjan Maitra. 2022. *EMCluster: EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution*.

Cochran, W. G. 1934. "The Distribution of Quadratic Forms in a Normal System, with Applications to the Analysis of Covariance." *Mathematical Proceedings of the Cambridge Philosophical Society*. https://doi.org/10.1017/S0305004100016595.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum Likelihood from Incomplete Data Via the *EM* Algorithm." *Journal of the Royal Statistical Society*. https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Ewa Szczurek, Przemyslaw Biecek &. 2021. *Bgmm: Gaussian Mixture Modeling Algorithms and the Belief-Based Mixture Modeling*.

Fraley, Chris. 1998. "Algorithms for Model-Based Gaussian Hierarchical Clustering." *SIAM Journal on Scientific Computing*. https://doi.org/10.1137/S1064827596311451.

Fraley, Chris, Adrian E. Raftery, and Luca Scrucca. 2022. *Mclust: Gaussian Mixture Modelling for Model-Based Clustering, Classification, and Density Estimation*.

Geary, R. C. 1936. "The Distribution of "Student's" Ratio for Non-Normal Samples." *Supplement to the Journal of the Royal Statistical Society*. https://doi.org/10.2307/2983669.

Gruen, Bettina, and Friedrich Leisch. 2022. *Flexmix: Flexible Mixture Modeling*.

Jin, Chi, Yuchen Zhang, Sivaraman Balakrishnan, et al. 2016. "Local Maxima in the Likelihood of Gaussian Mixture Models: Structural Results and Algorithmic Consequences." Curran Associates, Inc. https://doi.org/https://doi.org/10.48550/arXiv.1609.00978.

Koopman, B. O. 1936. "On Distributions Admitting a Sufficient Statistic." *Transactions of the American Mathematical Society*. https://doi.org/10.2307/1989758.

Kwedlo, Wojciech. 2013. "A New Method for Random Initialization of the EM Algorithm for Multivariate Gaussian Mixture Learning." In *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, edited by Robert Burduk, Konrad Jackowski, Marek Kurzynski, Michał Wozniak, and Andrzej Zolnierek. Springer International Publishing. https://doi.org/10.1007/978-3-319-00969-8/_8.

Langrognet, Florent, Remi Lebret, Christian Poli, et al. 2021. *Rmixmod: Classification with Mixture Modelling*.

Leytham, K. M. 1984. "Maximum Likelihood Estimates for the Parameters of Mixture Distributions." *Water Resources Research*. https://doi.org/10.1029/WR020i007p00896.

Liu, Charlie Wusuo. 2021. *GMKMcharlie: Unsupervised Gaussian Mixture and Minkowski and Spherical k-Means with Constraints*.

Lourens, Spencer, Ying Zhang, Jeffrey D Long, et al. 2013. "Bias in Estimation of a Mixture of Normal Distributions." *Journal of Biometrics & Biostatistics*. https://doi.org/10.4172/2155-6180.1000179.

McLachlan, Geoffrey, and David Peel. 2000. *Finite Mixture Models: McLachlan/Finite Mixture Models*. John Wiley & Sons, Inc. https://doi.org/10.1002/0471721182.

Melnykov, Volodymyr, Wei-Chen Chen, and Ranjan Maitra. 2021. *MixSim: Simulating Data to Study Performance of Clustering Algorithms*.

Nagode, Marko. 2015. "Finite Mixture Modeling via REBMIX." *Journal of Algorithms and Optimization*.

———. 2022. *Rebmix: Finite Mixture Modeling, Clustering & Classification*.

Nityasuddhi, Dechavudh, and Dankmar Böhning. 2003. "Asymptotic Properties of the EM Algorithm Estimate for Normal Mixture Models with Component Specific Variances." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/S0167-9473(02)00176-7.

Panic, Branislav, Jernej Klemenc, and Marko Nagode. 2020. "Improved Initialization of the EM Algorithm for Mixture Model Parameter Estimation." *Mathematics*. https://doi.org/10.3390/math8030373.

Rathnayake, Suren, Geoff McLachlan, David Peel, et al. 2019. *EMMIXmfa: Mixture Models with Component-Wise Factor Analyzers*.

Scrucca, Luca, and Adrian E. Raftery. 2015. "Improved Initialisation of Model-Based Clustering Using Gaussian Hierarchical Partitions." *Advances in Data Analysis and Classification*. https://doi.org/10.1007/s11634-015-0220-z.

Shimizu, Naoto, and Hiromasa Kaneko. 2020. "Direct Inverse Analysis Based on Gaussian Mixture Regression for Multiple Objective Variables in Material Design." *Materials & Design*. https://doi.org/10.1016/j.matdes.2020.109168.

Xu, Dinghai, and John Knight. 2010. "Continuous Empirical Characteristic Function Estimation of Mixtures of Normal Parameters." *Econometric Reviews*. https://doi.org/10.1080/07474938.2011.520565.

Young, Derek, Tatiana Benaglia, Didier Chauveau, et al. 2020. *Mixtools: Tools for Analyzing Finite Mixture Models*.

Simulation settings

For ease of reading, we reproduce below the parameter configurations used to run the three benchmarks, respectively for the univariate (Table 4), bivariate (5) and high dimensional setting (Table 6).

Table 4: The 9 parameter configurations tested to generate the samples of the univariate experiment, with $k = 4$ components.

ID	Entropy	OVL	Proportions	Means	Correlations
U1	1.00	3.3e-05	0.25 / 0.25 / 0.25 / 0.25	0 / 4 / 8 / 12	0.3 / 0.3 / 0.3 / 0.3
U2	1.00	5.7e-03	0.25 / 0.25 / 0.25 / 0.25	0 / 4 / 8 / 12	1 / 1 / 1 / 1
U3	1.00	2.0e-02	0.25 / 0.25 / 0.25 / 0.25	0 / 4 / 8 / 12	2 / 2 / 2 / 2
U4	0.96	3.3e-05	0.2 / 0.4 / 0.2 / 0.2	0 / 4 / 8 / 12	0.3 / 0.3 / 0.3 / 0.3
U5	0.96	5.8e-03	0.2 / 0.4 / 0.2 / 0.2	0 / 4 / 8 / 12	1 / 1 / 1 / 1
U6	0.96	2.0e-02	0.2 / 0.4 / 0.2 / 0.2	0 / 4 / 8 / 12	2 / 2 / 2 / 2
U7	0.68	2.7e-05	0.1 / 0.7 / 0.1 / 0.1	0 / 4 / 8 / 12	0.3 / 0.3 / 0.3 / 0.3
U8	0.68	4.4e-03	0.1 / 0.7 / 0.1 / 0.1	0 / 4 / 8 / 12	1 / 1 / 1 / 1
U9	0.68	1.5e-02	0.1 / 0.7 / 0.1 / 0.1	0 / 4 / 8 / 12	2 / 2 / 2 / 2

Table 5: The 20 parameter configurations tested to generate the samples of the bivariate experiment.

ID	Entropy	OVL	Proportions	Means	Correlations
B1	1.00	0.15000	0.5 / 0.5	(0,2);(2,0)	-0.8 / -0.8
B2	1.00	0.07300	0.5 / 0.5	(0,2);(2,0)	-0.8 / 0.8
B3	1.00	0.07300	0.5 / 0.5	(0,2);(2,0)	0.8 / -0.8
B4	1.00	0.00078	0.5 / 0.5	(0,2);(2,0)	0.8 / 0.8
B5	1.00	0.07900	0.5 / 0.5	(0,2);(2,0)	0 / 0
B6	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	-0.8 / -0.8
B7	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	-0.8 / 0.8
B8	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	0.8 / -0.8
B9	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	0.8 / 0.8
B10	1.00	0.00000	0.5 / 0.5	(0,20);(20,0)	0 / 0
B11	0.47	0.06600	0.9 / 0.1	(0,2);(2,0)	-0.8 / -0.8
B12	0.47	0.01600	0.9 / 0.1	(0,2);(2,0)	-0.8 / 0.8
B13	0.47	0.05000	0.9 / 0.1	(0,2);(2,0)	0.8 / -0.8
B14	0.47	0.00045	0.9 / 0.1	(0,2);(2,0)	0.8 / 0.8
B15	0.47	0.03900	0.9 / 0.1	(0,2);(2,0)	0 / 0
B16	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	-0.8 / -0.8
B17	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	-0.8 / 0.8
B18	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	0.8 / -0.8
B19	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	0.8 / 0.8
B20	0.47	0.00000	0.9 / 0.1	(0,20);(20,0)	0 / 0

Additional files

- Additional files related to the univariate setting
  - **S1.** Bootstrap distributions of the estimated parameters for each scenario described in 4.
  - **S2.** Mean, standard deviation, bias and MSE for each individually estimated parameter in scenarios listed in 4.
  - **S3.** Distribution of the running times taken for the EM estimation of the parameters of the GMM, across all nine scenarios described in 4, for each benchmarked package. We selected the *k*-means algorithm to initialise the EM algorithm, as being the least variable for a given package and scenario.
  - **S4.** Distribution of the time computations taken by the six initialisation methods listed in Table 3.
- Additional files related to the outliers setting:
  - **S5.** Bootstrap distributions of the estimated parameters used to generate Supplementary Figure 2. We additionally include the **otrimle** package, dedicated to these extreme distributions. Two scenarios were tested, introducing 2% and 4% of outliers drawn from an

Table 6: The 16 parameter configurations tested to generate the samples in a high dimensional context. The first digit of each ID index refers to an unique parameter configuration (identified by its level of overlap, entropy and topological structure, either circular or ellipsoidal, of the covariance matrix, while the lowercase letter depicts the number of observations, a) with $n = 200$ and b) with $n = 2000$.

ID	OVL	Number of observations	Proportions	Spherical
HD1a	1e-04	200	0.5 / 0.5	☑
HD1b	1e-04	2000	0.5 / 0.5	☑
HD2a	1e-04	200	0.19 / 0.81	☑
HD2b	1e-04	2000	0.19 / 0.81	☑
HD3a	1e-04	200	0.5 / 0.5	☒
HD3b	1e-04	2000	0.5 / 0.5	☒
HD4a	1e-04	200	0.21 / 0.79	☒
HD4b	1e-04	2000	0.21 / 0.79	☒
HD5a	2e-01	200	0.5 / 0.5	☑
HD5b	2e-01	2000	0.5 / 0.5	☑
HD6a	2e-01	200	0.15 / 0.85	☑
HD6b	2e-01	2000	0.15 / 0.85	☑
HD7a	2e-01	200	0.5 / 0.5	☒
HD7b	2e-01	2000	0.5 / 0.5	☒
HD8a	2e-01	200	0.69 / 0.31	☒
HD8b	2e-01	2000	0.69 / 0.31	☒

improper uniform distribution.

  - **S6.** Mean, standard deviation, bias and MSE for each individually estimated parameter in both scenarios visualised on Supplementary Figure 2, for each combination of package and initialisation method.

- Additional files related to the bivariate benchmark:

  - **S7.** Bootstrap distributions of the estimated parameters for each scenario described in 5.
  - **S8.** Mean, standard deviation, bias and MSE for each individually estimated parameter in scenarios listed in 5.
  - **S9.** Distribution of the running times taken for the EM estimation of the parameters of the GMM, across all twenty scenarios described in 5, for each benchmarked package. We selected the *k*-means algorithm to initialise the EM algorithm, as being the least variable for a given package and scenario.

- Additional files related to the high-dimensional benchmark:
  - **S10.** Bootstrap distributions of the estimated parameters for each scenario described in 6.
  - **S11.** Mean, standard deviation, bias and MSE for each individually estimated parameter in scenarios listed in 6.
  - **S12.** Distribution of the running times taken for the EM estimation of the parameters of the GMM, across all twenty scenarios described in 6, for each benchmarked package. We selected the *k*-means algorithm to initialise the EM algorithm, as being the least variable for a given package and scenario.

References

Bastien Chassagnol
Laboratoire de Probabilités, Statistiques et Modélisation (LPSM), UMR CNRS 8001
4 Place Jussieu Sorbonne Université
75005, Paris, France
ORCiD: 0000-0002-8955-2391
bastien_chassagnol@laposte.net

Antoine Bichat
Les Laboratoires Servier
50 Rue Carnot
92150, Suresnes, France
https://rdrr.io/github/abichat/abutils/
ORCiD: 0000-0001-6599-7081
antoine.bichat@servier.com

Cheïma Boudjeniba
Systems Biology Group, Dept. of Computational Biology, Institut Pasteur
25 Rue du Dr Roux
75015 Paris
cheima.boudjeniba@servier.com

Pierre-Henri Wuillemin
Laboratoire d'Informatique de Paris 6 (LIP6), UMR 7606
4 Place Jussieu Sorbonne Université
75005, Paris, France
http://www-desir.lip6.fr/~phw/
ORCiD: 0000-0003-3691-4886
pierre-henri.wuillemin@lip6.fr

Mickaël Guedj
Les Laboratoires Servier
50 Rue Carnot
92150, Suresnes, France
https://michaelguedj.github.io/
ORCiD: 0000-0001-6694-0554
mickael.guedj@gmail.com

Gregory Nuel
Laboratoire de Probabilités, Statistiques et Modélisation (LPSM), UMR CNRS 8001
4 Place Jussieu Sorbonne Université
75005, Paris, France
http://nuel.perso.math.cnrs.fr/
ORCiD: 0000-0001-9910-2354
Gregory.Nuel@math.cnrs.fr

Etienne Becht
Les Laboratoires Servier
50 Rue Carnot

92150, Suresnes, France
ORCiD: 0000-0003-1859-9202
etienne.becht@servier.com