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** [1] and **Rmixmod** [2] estimate the parameters of the mixture with smaller bias, while the RMSE and variability of the estimates is smaller with packages **bgmm** [3] , **EMCluster** [4] , **GMKMcharlie** [5], **flexmix** [6] and **mclust** [7]. 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.

1 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 [8] for mixture classification and [9] 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.
- the *diagonal* family, $\boldsymbol{\Sigma}_j = \text{diag}\left(\sigma_{1j}^2, \ldots, \sigma_{1D}^2\right)$, 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

[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 Section Appendix B: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms.

[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.

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 [10], [11]. We review them in supplementary 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 [12], [13].

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 [14]. 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 [14]. 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 [14].

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 Appendix B: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms.

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, [15] has shown that the Normal distribution is the only distribution for which the Cochran's theorem [16] 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 [18]. 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 [19], 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 [20]. 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 [21], 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** [22] package and the complete pseudo-code is described thoroughly in [23], [24]. 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 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

[3]The Pitman–Koopman–Darmois theorem [17] 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.

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, [23] 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: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms), 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 [20].
- 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 [25]).

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.

2 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: comprehensive report from the univariate and bivariate benchmark. 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** [26] and **HDclassif** [27] 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 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

[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 [23], p. 16.

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 Table 5 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 Parsimonious parametrisation of multivariate GMMs.
- 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: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms and row 3 of Table 2. Especially, GMMs estimation is particularly impacted by the presence of outliers, justifying a specific benchmark (see [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 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 General workflow. Code for *RGMMBench* is provided in 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 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 [20] 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: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms).
- 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 11.

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}.

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

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:

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

We may generalise this definition to k components by averaging the individual components' overlap. We use the function `MixSim::overlap` from the **MixSim** package [28] 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 [28]. 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 7 for the univariate setting and Table 12 for the bivariate 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 in Appendix section *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 7), the parameter estimates are identical in most cases across initialisation methods and packages, notably 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 (Figure 6 and Table 8), with various optimisations returning only local maxima far from the optimal estimate.

Similar scenarios in the bivariate setting (scenarios B6-B10 in Table 12), with a focus on B6, B7 and B10 visualised in Supplementary Figures 18, 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 7), 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 (Figure 7 and Table 9), 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 7: the scenarios B12 (Figure 14 and Table 14) and B14 (Figure 15 and Table 15). Similarly, scenarios B16, B17 and B20 (Table 12) with similar characteristics are summarised in supplementary Figure 19. For all these scenarios, neither the initialisation method nor the package have a statistical significant impact on the overall performance.

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 7) are obtained with the *k*-means initialisation (Figure 8 and Table 10) 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 12), with the best performance reached with the *k*-means initialisation method, followed by MBHC. This is

[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.

emphasised in supplementary Figure 17, 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.

Unbalanced and overlapping components

With unbalanced components and high OVL (scenario U9 in Table 7), 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 (Table 11). 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 (Figure 9 and Table 11), consistently with this method's assumption to use modes to initialize the components [23]. 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 12) 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 (Figure 13). 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 Tables 13 and 16). 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, Figure 16).

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 7 in the univariate setting, scenario B11, Table 12, for the bivariate simulation and scenario HD8, Table at the upper part of Appendix section *Figures and Tables in the HD simulation 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 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

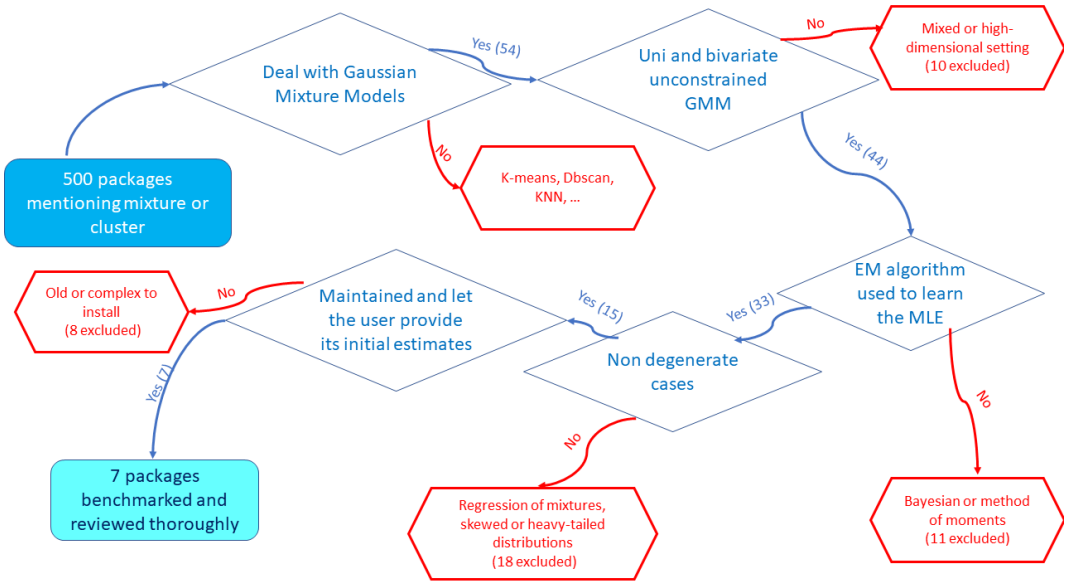


Figure 1: A minimal roadmap used for the selection of the packages reviewed in our benchmark.

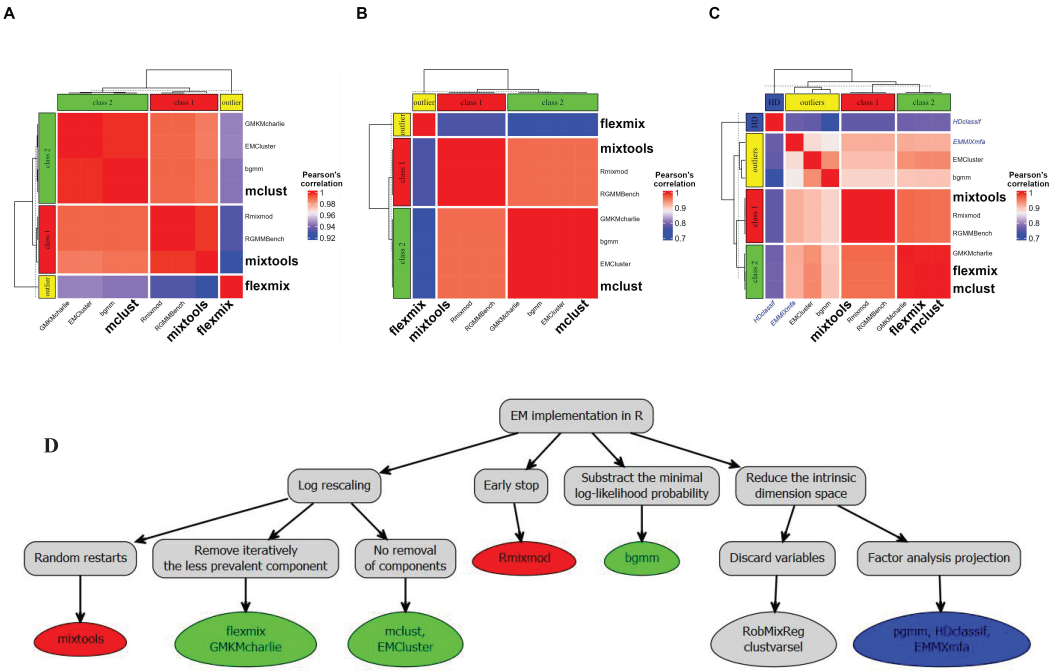


Figure 2: Panels A, B and C show respectively the heatmap of the Pearson correlation in the univariate, bivariate and high-dimensional framework between the parameters estimated by the packages, evaluated for the most discriminating and complex scenario. The correlation matrix was computed using the 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.

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 12), 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 estimations with the random initialisation method in scenario B11 (Table 12) and 80% failed estimations in the univariate setting[9] with the rebmix initialisation with scenario U9, Table 7.

3 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 [29]–[32], 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 [30]–[32]. This seems to contradict the statement from [31] 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 [23]). 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 [33] that the probability for the EM algorithm to

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

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

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 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 12.

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 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.

4 Bibliography

[1] D. Young, T. Benaglia, D. Chauveau, *et al.*, *Mixtools: Tools for analyzing finite mixture models*. 2020.

[2] F. Langrognet, R. Lebret, C. Poli, *et al.*, *Rmixmod: Classification with mixture modelling*. 2021.

[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

[3] P. B. & Ewa Szczurek, *Bgmm: Gaussian mixture modeling algorithms and the belief-based mixture modeling*. 2021.

[4] W.-C. Chen and R. Maitra, *EMCluster: EM algorithm for model-based clustering of finite mixture gaussian distribution*. 2022.

[5] C. W. Liu, *GMKMcharlie: Unsupervised gaussian mixture and minkowski and spherical k-means with constraints*. 2021.

[6] B. Gruen and F. Leisch, *Flexmix: Flexible mixture modeling*. 2022.

[7] C. Fraley, A. E. Raftery, and L. Scrucca, *Mclust: Gaussian mixture modelling for model-based clustering, classification, and density estimation*. 2022.

[8] C. Bouveyron and S. Girard, "Robust Supervised Classification with Mixture Models: Learning from Data with Uncertain Labels," *Pattern Recognition*, 2009, doi: 10.1016/j.patcog.2009.03.027.

[9] N. Shimizu and H. Kaneko, "Direct inverse analysis based on Gaussian mixture regression for multiple objective variables in material design," *Materials & Design*, 2020, doi: 10.1016/j.matdes.2020.109168.

[10] J. D. Banfield and A. E. Raftery, "Model-Based Gaussian and Non-Gaussian Clustering," *Biometrics*, 1993, doi: 10.2307/2532201.

[11] G. Celeux and G. Govaert, "A classification EM algorithm for clustering and two stochastic versions," *Computational Statistics & Data Analysis*, 1992, doi: 10.1016/0167-9473(92)90042-E.

[12] J. Chen, "Consistency of the MLE under mixture models," 2016. doi: 10.1214/16-STS578.

[13] G. McLachlan and D. Peel, *Finite Mixture Models: McLachlan/Finite Mixture Models*. John Wiley & Sons, Inc., 2000. doi: 10.1002/0471721182.

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

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

[16] W. G. Cochran, "The distribution of quadratic forms in a normal system, with applications to the analysis of covariance," *Mathematical Proceedings of the Cambridge Philosophical Society*, 1934, doi: 10.1017/S0305004100016595.

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

[18] L. Scrucca and A. E. Raftery, "Improved initialisation of model-based clustering using Gaussian hierarchical partitions," *Advances in data analysis and classification*, 2015, doi: 10.1007/s11634-015-0220-z.

[19] C. Fraley, "Algorithms for Model-Based Gaussian Hierarchical Clustering," *SIAM Journal on Scientific Computing*, 1998, doi: 10.1137/S1064827596311451.

[20] C. Biernacki, G. Celeux, and G. Govaert, "Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models," *Computational Statistics & Data Analysis*, 2003, doi: 10.1016/S0167-9473(02)00163-9.

[21] W. Kwedlo, "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*, R. Burduk, K. Jackowski, M. Kurzynski, M. Wozniak, and A. Zolnierek, Eds., Springer International Publishing, 2013. doi: 10.1007/978-3-319-00969-8_8.

[22] M. Nagode, *Rebmix: Finite mixture modeling, clustering & classification*. 2022.

[23] M. Nagode, "Finite mixture modeling via REBMIX," *Journal of Algorithms and Optimization*, 2015.

[24] B. Panic, J. Klemenc, and M. Nagode, "Improved initialization of the EM algorithm for mixture model parameter estimation," *Mathematics*, 2020, doi: 10.3390/math8030373.

[25] J. Baek, G. J. McLachlan, and L. K. Flack, "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*, 2010, doi: 10.1109/TPAMI.2009.149.

[26] S. Rathnayake, G. McLachlan, D. Peel, *et al.*, *EMMIXmfa: Mixture models with component-wise factor analyzers*. 2019.

[27] L. Berge, C. Bouveyron, and S. Girard, *HDclassif: High dimensional supervised classification and clustering*. 2019.

[28] V. Melnykov, W.-C. Chen, and R. Maitra, *MixSim: Simulating data to study performance of clustering algorithms*. 2021.

[29] D. Nityasuddhi and D. Böhning, "Asymptotic properties of the EM algorithm estimate for normal mixture models with component specific variances," *Computational Statistics & Data Analysis*, 2003, doi: 10.1016/S0167-9473(02)00176-7.

[30] S. Lourens, Y. Zhang, J. D. Long, *et al.*, "Bias in Estimation of a Mixture of Normal Distributions," *Journal of biometrics & biostatistics*, 2013, doi: 10.4172/2155-6180.1000179.

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

[32] D. Xu and J. Knight, "Continuous Empirical Characteristic Function Estimation of Mixtures of Normal Parameters," *Econometric Reviews*, 2010, doi: 10.1080/07474938.2011.520565.

[33] C. Jin, Y. Zhang, S. Balakrishnan, *et al.*, "Local Maxima in the Likelihood of Gaussian Mixture Models: Structural Results and Algorithmic Consequences." Curran Associates, Inc., 2016. doi: https://doi.org/10.48550/arXiv.1609.00978.

[34] G. R. Walsh, *Methods of Optimization*. Wiley, 1975.

[35] K. B. Petersen and M. S. Pedersen, "The matrix cookbook." Technical University of Denmark, 2008.

[36] R. A. Redner and H. F. Walker, "Mixture Densities, Maximum Likelihood and the Em Algorithm," *SIAM Review*, 1984, doi: 239https://doi.org/10.1137/1026034.

[37] L. Scrucca, M. Fop, T. B. Murphy, *et al.*, "Mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models," *The R journal*, 2016, doi: 10.32614/RJ-2016-021.

[38] R. P. Browne and P. D. McNicholas, "Estimating common principal components in high dimensions," *Advances in Data Analysis and Classification*, 2014, doi: 10.1007/s11634-013-0139-1.

[39] G. Schwarz, "Estimating the Dimension of a Model," *The Annals of Statistics*, 1978, doi: 10.1214/aos/1176344136.

[40] Y. Yang, "Can the strengths of AIC and BIC be shared? A conflict between model identification and regression estimation," *Biometrika*, 2005, doi: 10.1093/biomet/92.4.937.

[41] J. Fonseca, "The application of mixture modeling and information criteria for discovering patterns of coronary heart disease," *Journal of Applied Quantitative Methods*, 2008, doi: 10.1109/EMS.2008.36.

[42] G. Celeux, S. Fruewirth-Schnatter, and C. P. Robert, "Model selection for mixture models - perspectives and strategies." arXiv, 2018. doi: 10.48550/ARXIV.1812.09885.

[43] S. Bacci, S. Pandolfi, and F. Pennoni, "A comparison of some criteria for states selection in the latent Markov model for longitudinal data," 2012. Available: http://arxiv.org/abs/1212.0352

[44] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*. 1993.

[45] K. Basford, D. Greenway, G. Mclachlan, *et al.*, "Standard errors of fitted component means of normal mixtures," *Computational Statistics*, 1997, Available: https://www.researchgate.net/publication/37625647/_Standard/_errors/_of/_fitted/_component/_means/_of/_normal/_mixtures

[46] T. Jaki, T.-L. Su, M. Kim, *et al.*, "An evaluation of the bootstrap for model validation in mixture models," *Communications in statistics: Simulation and computation*, 2018, doi: 10.1080/03610918.2017.1303726.

[47] T. A. Louis, "Finding the Observed Information Matrix when Using the EM Algorithm," *Journal of the Royal Statistical Society*, 1982, doi: 10.1111/j.2517-6161.1982.tb01203.x.

[48] D. Oakes, "Direct calculation of the information matrix via the EM," *Journal of the Royal Statistical Society*, 1999, doi: 10.1111/1467-9868.00188.

[49] X. Cao and J. C. Spall, "Relative performance of expected and observed fisher information in covariance estimation for maximum likelihood estimates." 2012. doi: 10.1109/ACC.2012.6315584.

[50] L. Meng, "Method for computation of the fisher information matrix in the expectation-maximization algorithm." arXiv, 2016. doi: 10.48550/ARXIV.1608.01734.

[51] M. Delattre and E. Kuhn, "Estimating Fisher Information Matrix in Latent Variable Models based on the Score Function," 2019. doi: https://doi.org/10.48550/arXiv.1909.06094.

[52] M. Prates, V. Lachos, and C. Cabral, *Mixsmsn: Fitting finite mixture of scale mixture of skew-normal distributions*. 2021.

[53] A. Marandon, T. Rebafka, E. Roquain, *et al.*, "False clustering rate control in mixture models." arXiv, 2022. doi: 10.48550/ARXIV.2203.02597.

[54] M. Alberich-Carramiñana, B. Elizalde, and F. Thomas, "New algebraic conditions for the identification of the relative position of two coplanar ellipses," *Computer Aided Geometric Design*, 2017, doi: 10.1016/j.cagd.2017.03.013.

[55] R. Maitra and V. Melnykov, "Simulating Data to Study Performance of Finite Mixture Modeling and Clustering Algorithms," *Journal of Computational and Graphical Statistics*, 2010, doi: 10.1198/jcgs.2009.08054.

[56] M. Pastore and A. Calcagnì, "Measuring Distribution Similarities Between Samples: A Distribution-Free Overlapping Index," *Frontiers in Psychology*, 2019, doi: 10.3389/fpsyg.2019.01089.

[57] E. Nowakowska, J. Koronacki, and S. Lipovetsky, "Tractable Measure of Component Overlap for Gaussian Mixture Models," 2014. doi: 10.48550/arXiv.1407.7172.

[58] C. Biernacki, G. Celeux, and G. Govaert, "Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2000, doi: 10.1109/34.865189.

[59] H. Kurban, M. Jenne, and M. M. Dalkilic, "Using data to build a better EM: EM*for big data," *International Journal of Data Science and Analytics*, 2017, doi: 10.1007/s41060-017-0062-1.

[60] G. Celeux, S. Chrétien, and F. Forbes, "A Component-wise EM Algorithm for Mixtures," 2012.

[61] J. F. Bobb and R. Varadhan, *turboEM: A suite of convergence acceleration schemes for EM, MM and other fixed-point algorithms*. 2021.

[62] K. Murphy, *Machine Learning A Probabilistic Perspective*. Adaptive Computation; Machine Learning series, 2012. doi: 10.1080/09332480.2014.914768.

[63] M. A. T. Figueiredo and A. K. Jain, "Unsupervised learning of finite mixture models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002, doi: 10.1109/34.990138.

[64] X.-L. MENG and D. Rubin, "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework," *Biometrika*, 1993, doi: 10.1093/biomet/80.2.267.

[65] X.-L. Meng and D. Van Dyk, "The EM Algorithm–an Old Folk-song Sung to a Fast New Tune," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 1997, doi: 10.1111/1467-9868.00082.

[66] H.-U. Klein and M. Schaefer, *Epigenomix: Epigenetic and gene transcription data normalization and integration with mixture models*. 2022.

[67] A. T. Jones, *EMMIXgene: A mixture model-based approach to the clustering of microarray expression data*. 2020.

[68] L. Fallah and J. Hinde, *Pcensmix: Model fitting to progressively censored mixture data*. 2017.

[69] A. Komárek, *mixAK: Multivariate normal mixture models and mixtures of generalized linear mixed models including model based clustering*. 2022.

[70] N. Pocuca, R. P. Browne, and P. D. McNicholas, *Mixture: Mixture models for clustering and classification*. 2022.

[71] M. Thrun, O. Hansen-Goos, and A. Ultsch, *AdaptGauss: Gaussian mixture models (GMM)*. 2020.

[72] X. Li, Y. Fu, X. Wang, *et al.*, *MMDvariance: Detecting differentially variable genes using the mixture of marginal distributions*. 2018.

[73] A. Rauschenberger, *Semisup: Semi-supervised mixture model*. 2022.

[74] P. Papastamoulis, *fabMix: Overfitting bayesian mixtures of factor analyzers with parsimonious covariance and unknown number of components*. 2020.

[75] P. D. McNicholas, A. ElSherbiny, A. F. McDaid, *et al.*, *Pgmm: Parsimonious gaussian mixture models*. 2022.

[76] B. C. Franczak, R. P. Browne, and P. D. McNicholas, *Sensory: Simultaneous model-based clustering and imputation via a progressive expectation-maximization algorithm*. 2016.

[77] N. Dean, A. E. Raftery, and L. Scrucca, *Clustvarsel: Variable selection for gaussian model-based clustering*. 2020.

[78] P. Macdonald and with contributions from Juan Du, *Mixdist: Finite mixture distribution models*. 2018.

[79] R. Mohammadi, *Bmixture: Bayesian estimation for finite mixture of distributions*. 2021.

[80] X. Lu, Y. Li, and T. Love, *Bpgmm: Bayesian model selection approach for parsimonious gaussian mixture models*. 2022.

[81] L. M. Avila, M. R. May, and J. Ross-Ibarra, *DPP: Inference of parameters of normal distributions from a mixture of normals*. 2018.

[82] Y. Xu, P. Mueller, D. Telesca, *et al.*, *Dppmix: Determinantal point process mixture models*. 2020.

[83] A. M. Garay, M. B. Massuia, V. H. Lachos, *et al.*, *BayesCR: Bayesian analysis of censored regression models under scale mixture of skew normal distributions*. 2017.

[84] C. A. Kapourani, *Melissa: Bayesian clustering and imputationa of single cell methylomes*. 2022.

[85] P. Baker, *polySegratioMM: Bayesian mixture models for marker dosage in autopolyploids*. 2018.

[86] E. A. Houseman, Sc.D., D. C. Koestler, *et al.*, *RPMM: Recursively partitioned mixture model*. 2017.

[87] A. T. Jones and H. D. Nguyen, *SAGMM: Clustering via stochastic approximation and gaussian mixture models*. 2019.

[88] F. Shokoohi, *Fmrs: Variable selection in finite mixture of AFT regression and FMR models*. 2022.

[89] R. Turner, *Mixreg: Functions to fit mixtures of regressions*. 2021.

[90] Y. Li and K. Chen, *fmerPack: Tools of heterogeneity pursuit via finite mixture effects model*. 2021.

[91] M. Prates, V. Lachos, and A. Garay, *Nlsmsn: Fitting nonlinear models with scale mixture of skew-normal distributions*. 2021.

[92] S. Cao, W. Chang, and C. Zhang, *RobMixReg: Robust mixture regression*. 2020.

[93] V. Kubicki, C. Biernacki, and Q. Grimonprez, *RMixtComp: Mixture models with heterogeneous and (partially) missing data*. 2021.

[94] D. McParland and I. C. Gormley, *clustMD: Model based clustering for mixed data*. 2017.

[95] C. Viroli and G. J. McLachlan, *Deepgmm: Deep gaussian mixture models*. 2020.

[96] Z. Wang, *IMIX: Gaussian mixture model for multi-omics data integration*. 2022.

[97] A. Rau, *Coseq: Co-expression analysis of sequencing data*. 2022.

[98] Z. McCaw, *MGMM: Missingness aware gaussian mixture models*. 2021.

[99] A. M. Garay, M. B. Massuia, and V. Lachos, *SMNCensReg: Fitting univariate censored regression model under the family of scale mixture of normal distributions*. 2022.

[100] L. Mouselimis, *ClusterR: Gaussian mixture models, k-means, mini-batch-kmeans, k-medoids and affinity propagation clustering*. 2022.

[101] S. Iovleff, *MixAll: Clustering and classification using model-based mixture models*. 2019.

[102] W.-C. Chen and G. Ostrouchov, *Pmclust: Parallel model-based clustering using expectation-gathering-maximization algorithm for finite mixture gaussian model*. 2021.

[103] S. Li, J. Chen, and P. Li., *MixtureInf: Inference for finite mixture models*. 2016.

[104] Y. Yu, *mixR: Finite mixture modeling for raw and binned data*. 2021.

[105] P. Schlattmann, J. Hoehne, and M. Verba, *CAMAN: Finite mixture models and meta-analysis tools - based on c.a.MAN*. 2022.

[106] B. G. Lindsay, "Mixture Models: Theory, Geometry and Applications," *NSF-CBMS Regional Conference Series in Probability and Statistics*, 1995, Available: https://www.jstor.org/stable/4153184

[107] J. Chen and Z. Chen, "Extended Bayesian information criteria for model selection with large model spaces," *Biometrika*, 2008, doi: 10.1093/biomet/asn034.

[108] Y. He and Z. Chen, "The EBIC and a sequential procedure for feature selection in interactive linear models with high-dimensional data," *Annals of the Institute of Statistical Mathematics*, 2016, doi: 10.1007/s10463-014-0497-2.

[109] M. Dai and A. Mukherjea, "Identification of the Parameters of a Multivariate Normal Vector by the Distribution of the Maximum," *Journal of Theoretical Probability*, 2001, doi: 10.1023/A:1007889519309.

[110] C. Robert and G. Casella, *Introducing Monte Carlo Methods with R*. Springer, 2010. doi: 10.1007/978-1-4419-1576-4.

[111] O. Mersmann, *Microbenchmark: Accurate timing functions*. 2021.

[112] T. Benaglia, D. Chauveau, D. R. Hunter, *et al.*, "mixtools: An R package for analyzing finite mixture models," *Journal of Statistical Software*, 2009.

1 Appendix A: In-depth statistical elements about parameters estimation in GMMs

Application of the EM algorithm to GMMs

While solving Equation (10) to retrieve the MLE estimates in the M-step of the EM algorithm, we have to enforce the non-negativity and sum-to-one constraint of the mixture models (Equation (2)). This is enabled by the *Lagrange multipliers* tip, which consists in practice to add the equality constraint over the parameters to estimate, here $-\lambda(\sum_{j=1}^{k} p_j - 1)$, to the function to be optimised [34].

The evaluation of the roots of the derivative of the auxiliary function (see Equation (10)) at the parameter p_j with the additional unit simplex constraint (2) allows to readily compute a MLE estimate of the ratios, valid for any finite mixture model (Equation (13)):

$$\hat{p_j} = \frac{\sum_{i=1}^{n} \eta_i(j)}{n} \tag{13}$$

Additionally, we restrained in both the univariate and multivariate settings to the fully *unconstrained parametrisation*, in which each component follows its own parametric distribution. The general derivative of the auxiliary function with respect to each component parametric distribution ζ_j, is given by Equation (14)[11]:

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \zeta_j} = \sum_{i=1}^{n} \eta_i(j) \frac{\partial \log(f_{\zeta_j}(X_i|S_i = j))}{\partial \zeta_j} \tag{14}$$

Accordingly, if a closed form for the computation of the MLE in supervised cases is known (and fortunately this is the case for both the univariate and multivariate Gaussian distributions), the computation of the maximum of the auxiliary function can be readily calculated.

Plug-in the corresponding parametric distribution in the auxiliary function (10) yields the following formula for the univariate GMM (Equation (15)):

$$Q(\theta|\hat{\theta}_{q-1}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_i(j) \left(\log(p_j) - \log(\sigma_j) - \frac{(X_i - \mu_j)^2}{2\sigma_j^2} \right) + K \tag{15}$$

and Equation (16) for the multivariate GMM:

$$Q(\theta|\hat{\theta}_{q-1}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_i(j) \left[\log(p_j) - \frac{1}{2} \left(\log(\det(\boldsymbol{\Sigma}_j)) + (x_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1} (x_i - \boldsymbol{\mu}_j) \right) \right] + K \tag{16}$$

K is a constant with respective values of $\frac{-nD\log(2\pi)}{2}$ and $\frac{-n\log(2\pi)}{2}$ in the univariate and multivariate setting.

In the univariate setting, the individual MLE mean μ_j, and variance, σ_j, estimates are readily available (Equations (17) - (18)):

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \mu_j} = 0 \Leftrightarrow \mu_j = \frac{\sum_{i=1}^{n} \eta_i(j) X_i}{\sum_{i=1}^{n} \eta_i(j)} \tag{17}$$

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \sigma_j} = 0 \Leftrightarrow \sigma_j^2 = \frac{\sum_{i=1}^{n} \eta_i(j)(x_i - \mu_j)^2}{\sum_{i=1}^{n} \eta_i(j)} \tag{18}$$

Before finding the optimum of the auxiliary function in the multivariate setting, we remind the interested reader of some relevant calculus formulas below:

Transpose matrix properties

a. $\det(p\boldsymbol{A}) = p^G \det(\boldsymbol{A})$ b. $\det(A^{-1}) = \frac{1}{\det(A)}$ c. $\left(\boldsymbol{A}^{-1}\right)^\top = \boldsymbol{A}^{-1}$[a]

[a] when $\boldsymbol{A}$ is itself symmetric, as by definition, $\boldsymbol{A}^\top = \boldsymbol{A}$

[11] It is equivalent to compute the MLE of a sample following distribution f_{ζ_j} weighted by the vector of posterior probabilities.

Matrix calculus

Given a symmetric matrix $\boldsymbol{A}$ of full rank D and two vectors $\boldsymbol{x}$ and $\boldsymbol{\mu}$ of size D, the following derivative properties hold:

a. $\frac{\partial \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}}{\partial \boldsymbol{A}} = \boldsymbol{x}\boldsymbol{x}^\top$

b. $\frac{\partial (\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{A}(\boldsymbol{x}-\boldsymbol{\mu})}{\partial \boldsymbol{\mu}} = -2\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{\mu})$

c. $\frac{\partial \log(\det(\boldsymbol{A}))}{\partial \boldsymbol{A}^{-1}} = -\boldsymbol{A}$ [a]

[a] Other matrix calculus formulas and notations are available on Matrix calculus and demonstration details from *The Matrix Cookbook* [35].

Using the calculus formulas derived in the previous boxes, a closed form for the MLE estimate of the mean, $\boldsymbol{\mu}_j$, and covariance, $\boldsymbol{\Sigma}_j$, is readily computed (see Equations (19) - (20)):

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \boldsymbol{\mu}_j} = \sum_{i=1}^{n} \eta_i(j)\boldsymbol{\Sigma}_j^{-1}(x_i - \boldsymbol{\mu}_j) = 0 \Leftrightarrow \mu_j = \frac{\sum_{i=1}^n \eta_i(j)\boldsymbol{x}_i}{\sum_{i=1}^n \eta_i(j)} \tag{19}$$

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \boldsymbol{\Sigma}_j^{-1}} = \frac{1}{2}\sum_{i=1}^{n} \eta_i(j)\left[\boldsymbol{\Sigma}_j - (x_i - \boldsymbol{\mu}_j)(x_i - \boldsymbol{\mu}_j)^\top\right] = 0 \Leftrightarrow \boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^n \eta_i(j)(x_i - \boldsymbol{\mu}_j)(x_i - \boldsymbol{\mu}_j)^\top}{\sum_{i=1}^n \eta_i(j)} \tag{20}$$

Explicitly optimising the equations ((15)-(16)) yield the following MLE parameters in both the univariate and multivariate settings (Table 4), as detailed in [31], [36]:

Table 4: An overview of the practical implementation of the EM algorithm in GMMs.

	Univariate GMM	**Multivariate GMM**
E-step	$\eta_i(j) = \frac{\hat{p}_j^q \mathcal{N}(x_i\|\hat{\mu}_j^q, \hat{\sigma}_j^q)}{\sum_{j=1}^k \hat{p}_j^q \mathcal{N}(x_i\|\hat{\mu}_j^q, \hat{\sigma}_j^q)}$	$\eta_i(j) = \frac{\hat{p}_j^q \mathcal{N}_D(\boldsymbol{x}_i\|\hat{\boldsymbol{\mu}}_j^q, \hat{\boldsymbol{\Sigma}}_j^q)}{\sum_{j=1}^k \hat{p}_j^q \mathcal{N}_D(\boldsymbol{x}_i\|\hat{\boldsymbol{\mu}}_j^q, \hat{\boldsymbol{\Sigma}}_j^q)}$
Ratios estimation	$\hat{p}_j^{q+1} = \frac{\sum_{i=1}^n \eta_i(j)}{n}$	$\hat{p}_j^{q+1} = \frac{\sum_{i=1}^n \eta_i(j)}{n}$
Mean estimation	$\hat{\mu}_j^{q+1} = \frac{\sum_{i=1}^n \eta_i(j)x_i}{\sum_{i=1}^n \eta_i(j)}$	$\hat{\boldsymbol{\mu}}_j^{q+1} = \frac{\sum_{i=1}^n \eta_i(j)\boldsymbol{x}_i}{\sum_{i=1}^n \eta_i(j)}$
(Co)Variance estimation	$\left(\hat{\sigma}_j^2\right)^{q+1} = \frac{\sum_{i=1}^n \eta_i(j)\left(x_i - \hat{\mu}_j^{q+1}\right)^2}{\sum_{i=1}^n \eta_i(j)}$	$\left(\hat{\boldsymbol{\Sigma}}_j^2\right)^{q+1} = \frac{\sum_{i=1}^n \eta_i(j)\left(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{q+1}\right)\left(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}_j^{q+1}\right)^\top}{\sum_{i=1}^n \eta_i(j)}$

In both cases, obtaining the parameters of each component's parametric distribution turn to be equivalent to the computation of the mean and variance of a weighted sample, which can be computed in R with `stats::weighted.mean` and `stats::cov.wt` functions[12]. Importantly, the value of the mapping function only depends on the set of the observations X, but does not depend on the parameter to estimate θ. Indeed, the statistic computed by the EM algorithm is sufficient, which is one of its main advantages.

The complete code associated to our R implementation is implemented respectively with `enmix_univariate` and `enmix_bivariate` for the univariate and multivariate setting, available on GitHub at RGMMBench, as well as the programs used to generate the several plots and tables of the article. We additionally made two choices not clearly set in the literature:

- The algorithm stops when when the absolute difference between consecutive log-likelihoods falls below a user-defined threshold *epsilon*, with a maximal number of *itmax* iterations allowed to reach this convergence.
- In order to avoid numerical underflows resulting in inconsistent ratios, of type $0/0$, we rely on the fact that Gaussian distributions belong to the exponential family to log-rescale our observations and compute efficiently the posterior probabilities in the E-step of the EM algorithm. First, to avoid null values for highly unlikely observations, those far from the centroids, we use the log attribute of `stats::dnorm` and `mvtnorm::dmvnorm` functions, see Equation (21):

$$\begin{aligned} \ell(\theta|x) &= \log(\sum_{j=1}^{k} p_j f\zeta_j(x)) \\ &= \log\left(\exp\left[\log(p_j) + \log(f\zeta_j(x))\right]\right) \end{aligned} \tag{21}$$

[12] We assign "ML" to the argument *method* to get the biased but true MLE estimate of the covariance

Second, we rewrite our sum of exponentials, the one enclosed into the log, to use the Taylor' series of $\log(1+x)$, with $|x| \ll 1$, see Equation (22):

$$\begin{aligned}\log\left(\sum_{j=1}^{k} e^{a_j}\right) &= \log\left(\exp(a'_j) \times \left[1 + \sum_{j\neq j'} \exp\left(\frac{a_j}{a_{j'}}\right)\right]\right) \\ &= a_{j'} + \text{log1p}\left(\sum_{j\neq j'} \exp\left(\frac{a_j}{a_{j'}}\right)\right), \text{ with } j' = \arg\max_{\forall j\in\{1,\dots,k\}} (e^{a_j})\end{aligned} \tag{22}$$

with `log1p` the R function dedicated for this Taylor's development. The posterior probabilities are then given by Equation (23):

$$\log\left(\mathbb{P}_\theta(S = j|X = x)\right) = \log(p_j) + \log(f_{\zeta_j}) - \ell(\theta|x) \tag{23}$$

- We stop the algorithm early when the estimates are trapped in the boundaries of the parameter space, typically when the ratio of a component or its associated variance tends to zero. This case rarely occurs in our simulations: once in univariate and never in multivariate.

Parsimonious parametrisation of multivariate GMMs

Parsimonious parametrisation of GMMs models are provided by the following *eigenvalue* factorisation of the covariance matrix (Equation (24)):

$$\boldsymbol{\Sigma}_j = \lambda_j \boldsymbol{Q}_j \boldsymbol{D}_j \boldsymbol{Q}_j^\top \tag{24}$$

with $\lambda_j = \det\left(\boldsymbol{\Sigma}_j\right)^{\frac{1}{D}}$ a scalar proportional to the total volume of the ellipsoid (or area in bidimensional setting), $\boldsymbol{D}_j$ a diagonal matrix storing the eigenvalues normalised such that $|\boldsymbol{D}_j| = 1$[13] and $\boldsymbol{Q}_j$ a $\mathcal{M}_D(\mathbb{R})$ *orthogonal matrix* whose columns are D linearly independent *eigenvectors* generating an orthonormal basis in $\mathbb{R}^D$ while $\boldsymbol{Q}_j^\top$ is its corresponding transpose matrix. The existence of the decomposition is guaranteed by the positive definiteness constraint over the covariance matrix while the orthogonality of $\boldsymbol{Q}_j$ results from its symmetry. When the matrix to factorise is positive-definite and symmetric, we also refer to it as *spectral decomposition*, a special case of *eigendecomposition*.

Each of these matrices can be constrained to be equal or variable across clusters, hence this decomposition reveals 14 possible models with different geometric characteristics, namely:

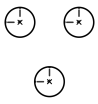
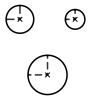
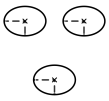
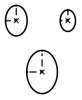
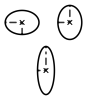
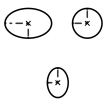
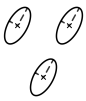
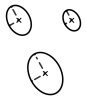
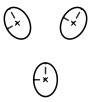
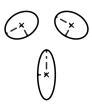
- two models with the *spherical family*, for which only λ_j is used to control the *isotropic* (same radius in any dimension) volume of each component of the corresponding distribution structure
- four models with the *diagonal family*, using λ_j with possibly distinct diagonal elements and $\boldsymbol{D}_j$ to specify the shape of the density contours. In that context, $\boldsymbol{Q}_j$ is henceforth a permutation matrix, whose inputs are only zeros and an unique one per row.
- eight models with the *general family*, using additionally $\boldsymbol{Q}_j$ to determine the orientation of the main axes of the ellipsoids. Indeed, in the last two families described, this matrix was equal to the identity, hence the axis of the ellipsoids were aligned with the standard $\mathbb{R}^D$ basis.

We detail the main characteristics of the 14 parametrisations (28 if we add for each model the equiproportional hypothesis) in Table (12):

- The first column describes in general and understandable terms each parametrisation, with I meaning invariant (alternatively, not used in the parametrisation), E means equal and V variable while the second column matches the corresponding matrix decomposition of the covariance matrix. These 14 models are all included in one of the three super-families: spherical, diagonal and ellipsoidal listed before. As an example, the model VEI has variable volumes λ_j in relation with the cluster, however shares same general shape (as we can note on the Representations, all isodensities are distributed along the x-axis) and invariant directions (in other words, the transition matrix is the identity matrix, entailing that all scatter plots are aligned with the Cartesian coordinate axes).

[13] [2] enforces an additional but, in our opinion, superfluous constraint that the eigen values are sorted by decreasing order

Table 5: The 14 canonical parametrisations of the within-group covariance matrix $\mathbf{\Sigma}_j$ with the corresponding geometric representations.

Model	Notation	Family	M-step	Number of parameters	Representation
EII	$[\lambda I]$	Spherical	CF	$\alpha + 1$	
VII	$[\lambda_j I]$	Spherical	CF	$\alpha + k$	
EEI	$[\lambda D]$	Diagonal	CF	$\alpha + d$	
VEI	$[\lambda_j D]$	Diagonal	IP	$\alpha + d + k - 1$	
EVI	$[\lambda D_j]$	Diagonal	CF	$\alpha + kd - k + 1$	
VVI	$[\lambda_j D_j]$	Diagonal	CF	$\alpha + kd$	
EEE	$[\lambda QDQ^\top]$	Ellipsoidal	CF	$\alpha + \beta$	
EVE	$[\lambda QD_jQ^\top]$	Ellipsoidal	IP	$\alpha + \beta$	
VEE	$[\lambda_j QDQ^\top]$	Ellipsoidal	IP	$\alpha + \beta + (k-1)(d-1)$	
VVE	$[\lambda_j QD_jQ^\top]$	Ellipsoidal	IP	$\alpha + \beta + d(k-1)$	
EEV	$[\lambda Q_j DQ_j^\top]$	Ellipsoidal	CF	$\alpha + k\beta - d(k-1)$	
VEV	$[\lambda_j Q_j DQ_j^\top]$	Ellipsoidal	IP	$\alpha + k\beta - (k-1)(d-1)$	
EVV	$[\lambda Q_j D_j Q_j^\top]$	Ellipsoidal	CF	$\alpha + k\beta - k + 1$	
VVV	$[\lambda_j Q_j D_j Q_j^\top]$	Ellipsoidal	CF	$\alpha + k\beta$	

- Varying the volume λ_j, given a fixed $\boldsymbol{Q}$ and $\boldsymbol{D}$, amounts to an *enlargement* (when all dimensions of a figure are changed in the same scale, also referred to as *isotropic* transformation), varying the eigenvectors $\boldsymbol{Q}_j$, given a fixed volume λ_j and $\boldsymbol{D}$ is equivalent to a rotation and finally varying the diagonal matrix $\boldsymbol{D}_j$, given the other parameters of Equation (24) are fixed, results in a *distortion* of the representation.
- CF means that the M-step is in closed form while IP entails that the M-step is iterative.
- The number of parameters enumerates the *degrees of freedom*, namely the number of parameters to truly estimate once the sum-to-one constraint is enforced (Equation (2)). In detail, k is the number of components of the GMM model, D its dimension, $\alpha = kD + k - 1$ is the number of parameters required to identify the mean vector of each component (kD) and the ratios $k - 1$ and $\beta = \frac{D(D+1)}{2}$ the number of covariance terms to estimate for a given component (D variance diagonal terms, the remaining terms being the pairwise symmetric covariance terms between the features). Note that the complexity of the covariance matrix in the fully unconstrained model (Model VVV) grows linearly with the number of components while exploding in the order $\mathcal{O}(D)$ with the number of dimensions. Meantime, the complexity of the parametrisation with the homoscedastic spherical family (Model EII) is constant.
- Last column displays the 14 most common GMMs parametrisations, by plotting the ellipses and centroids of a three components bivariate GMM parametrised by the mean vector and covariance of each component. For any additional detail, we refer the interested reader to **mclust** [37] and **Rmixmod** [2] vignettes for a general introduction to GMMs and to [10], [11], [38] for the closed formulas of the models.

Model selection

When comparing several models with several number of components or parametrisations, the likelihood is uninformative as it can be arbitrarily minimised by increasing the complexity of the model or adding components. it is then necessary to penalise for complexity when comparing them. The general form of the penalty metric, *GIC* (for generalised information criteria), is given by Equation (25):

$$\text{GIC}(\theta) = \underbrace{p(\theta)}_{\text{penalty term}} - \underbrace{2\ell(X|\theta)}_{\text{log-likelihood of the model}} \tag{25}$$

Among them, we set apart scores focused on selecting selecting the right number of parameters and components, namely the *degrees of freedom* (d.o.f.) of the model ($3k - 1$ parameters for the univariate unconstrained GMM), and those focusing on retrieving readable clusters.

In the first category, the *AIC* (Akaike information criterion) [39] is a *minimax-rate optimal* (score that minimises the risk in the worst case) but inconsistent metric [40] , proned to overestimate the true number of components. *BIC* (Bayesian Information Criterion), and CAIC (consistent AIC), accounting for both the number of parameters and the sample size, are consistent metrics. Finally, the *MDL*(Minimum Description Length) criterion accounts for the number of parameters, sample size and number of components. Its core objective differs from the others as it aims at reducing the amount of code to encode both parameters and observations but is practically close to the BIC metric. A thorough description of these scores, with their formulas and theoretical properties, can be found in [41], [42].

In the second category, the most commonly implemented is the *ICL (integrated complete-data likelihood)*, a BIC criterion with an additional entropy penalty [13]. As opposed to *BIC*, the entropy term reduces the number of components to a well-separated and readable clustering. Hence, it tends to underestimate their true number when components are overlapping. Alternative similar metrics are the *CLC* (Classification Likelihood Criterion), *AWE* (Approximate Weight of Evidence) and *NEC* (Normalised Entropy Criterion) metrics [43]. The several metrics implemented by the reviewed packages are listed in 2.

The *Likelihood-ratio test* (LRTS) can also be used to compare *nested models*, with additional advantage to possibly derive a p-value yielding the probability that a complex model (with more components) should preferentially be used over a simpler one. Traditionally, common process is to add one component after the other, until hypothesis H_0 can not be rejected anymore. Under standard regularity conditions of Cramer's theorem, Wilk's theorem states that the Likelihood Ratio distribution follows asymptotically a χ^2 distribution, but unfortunately these conditions are not met in mixture models [13]. To counterbalance it, bootstrap inference [13] is often used to derive an empirical distribution of the Likelihood Ratio.

Derivation of confidence intervals in GMMs

Punctual estimation, with a single estimate $\hat{\theta}$ for a given n-sample, is not enough to evaluate the performance of a specific method, as drawing another n-sample using the same parameters is likely to lead to a different distribution and estimation of $\hat{\theta}$. Instead, it can be interesting to retrieve the distribution or at least the variability of the estimated parameters, which can reveal useful to derive confidence intervals. However, obtaining the distribution or even an asymptotic approximation of the distribution of the parameters is not feasible in practice with mixture models [13]. Hence, most authors recommend to use bootstrap methods for the generation of confidence intervals, as suggested in [44], [45].

Bootstrap distributions of the parameters are generally retrieved via *empirical* or *parametric* bootstrap, both available in the **mclust** package. In the *empirical* or *non-parametric* bootstrap [46], we draw iteratively N samples of size n with replacement from the original observed variable $x_{1:n}$. In the *parametric* bootstrap, N simulations are built from the parameter estimated with the available observations of X, via the EM algorithm or any method used for parameter estimation. In both cases, we obtain an empiric distribution of the parameter estimate: $\hat{\theta}_{1:N} = (\hat{\theta}^1, \ldots, \hat{\theta}^N)$. Sample mean and standard deviation (SD) of this empirical distribution can be used to retrieve an asymptotic estimate of the variability of the parameter estimate $\hat{\theta}$, the bias or the MSE of the parameter estimates. To get unbiased estimates of the true standard deviation and mean of the estimates, it is of common practice to compute the empirical covariance matrix of the sample $\text{cov}[\hat{\theta}] = \frac{\sum_{j=1}^{N}(\hat{\theta}_j - \mathbb{E}[\hat{\theta}])(\hat{\theta}_j - \mathbb{E}[\hat{\theta}])^{\mathrm{T}}}{N-1}$, the square roots of its diagonal terms corresponding to the empiric SDs. Symmetric $1-\alpha$ asymptotic confidence intervals using the Central Limit Theorem (CLT) can then be simply derived Equation (26):

$$\mathbb{E}[\hat{\theta}_t] \pm \frac{1}{\sqrt{n}} z_{1-\frac{\alpha}{2}} \sqrt{\text{var}(\hat{\theta}_t}, \quad \forall t \in \{1, \ldots, 3k\} \tag{26}$$

with $z_{1-\frac{\alpha}{2}}$ the $1-\frac{\alpha}{2}$ quantile of the standard Gaussian distribution.

If computing the covariance matrix is not possible analytically, it can be approximated by the expected Fisher Information Matrix $\mathcal{I}_{\text{exp}}(\theta)$ (FIM), given by Equation (27):

$$\left[\mathcal{I}_{\text{exp}}(\theta)\right]_{1\leq i\leq 3k, 1\leq j\leq 3k} = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \ell(\theta|X)\right] \tag{27}$$

Indeed, the Cramér-Rao theorem states that the diagonal elements of the inverse of the FIM are upper bounded by the variability of the parameters: $\text{var}(\hat{\theta}) \geq \frac{1}{\mathcal{I}(\theta)}$. This implies that the ratio between inverse of the FIM and the variance $e(\hat{\theta}) = \frac{\mathcal{I}(\hat{\theta})^{-1}}{\text{var}(\hat{\theta})}$ converges to 1, using the asymptotic efficiency of the MLE estimate of GMMs.

Unfortunately, the computation of the expected FIM is still a hard task. Hence it is generally replaced by the observed FIM, the negative of the Hessian matrix of the incomplete log-likelihood function: $\mathcal{I}_{\text{obs}}(\theta) = -\frac{\partial^2}{\partial\theta_i \partial\theta_j} \ell(\theta|X)$. Exact general formulas are provided for the univariate case in [47] and for the multivariate case in [48]. Yet, it has to be noted that the expected FIM generally outperforms the observed FIM in estimating the covariance matrix of the MLE [49].

However all these methods require to compute second derivatives of the log-likelihood leading to some disadvantages from a computational point of view. More recently, [50] and [51] proposed an accelerated algorithm requiring only computation of first order derivatives. A similar alternative is implemented in the **mixsmsn** package [52]: `mixsmsn::im.smsn`, in which the Hessian matrix is approximated by the cross-product of the gradient of the log-likelihood Equation (28):

$$\mathcal{I}_{\text{obs}}(\theta) \approx -\frac{\partial \log(\ell(\theta|X))}{\partial\theta} \frac{\partial \log(\ell(\theta|X))}{\partial\theta}^T \tag{28}$$

according to an idea developed in paper [45]. For a more general introduction to Gaussian mixtures, including other models and parametrisations in the multivariate case, we refer the reader to the reference book *Gaussian parsimonious clustering models* [11].

An analytic formula of the overlap for univariate Gaussian mixtures

From an analytic point of view, the overlap between k components of variable X is given by Equation (29):

$$\text{OVL}(X) = 1 - \int_{\mathbb{R}} \max_j (p_j \varphi_{\zeta_j}(x)) dx \tag{29}$$

The 1 in Equation (29) corresponds to the integration of probability $f_\theta(X)$ distribution over its domain. The second part is the area under the curve of the component density function maximised on $\mathbb{R}$, with j the index of the component maximised at that point. It should be noted that the definition used here for the overlap is closely related to the definition of the *false clustering rate* (FCR) [53].

Equation (29) simplifies for a two component mixture distribution to Equation (30):

$$\text{OVL}(X) = \int_{\mathbb{R}} \min \left(p_1 \varphi_{\zeta_1}(x), p_2 \varphi_{\zeta_2} dx(x) \right) \tag{30}$$

From a probabilistic point of view, we can rewrite Equation (30) as the overall probability of assigning a wrong label to a given observation. With two components, this simply decomposes as the sum of the probability of mistakenly assigning an observation from component 2 to component 1 and the probability of assigning an observation from component 1 to component 2 Equation (31):

$$\begin{aligned} \text{OVL}(1,2) &= \text{OVL}(1|2) + \text{OVL}(2|1) \\ &= \mathbb{P}\left(p_1 \varphi(X, \mu_1, \sigma_1) \leq p_2 \varphi(X, \mu_2, \sigma_2)\right) + \mathbb{P}\left(p_2 \varphi(X, \mu_2, \sigma_2) \leq p_1 \varphi(X, \mu_1, \sigma_1)\right) \\ &= \int_{\mathbb{R}} p_1 \varphi_{\zeta_1}(x) 1_{p_1 \varphi_{\zeta_1} \leq p_2 \varphi_{\zeta_2}} dx + \int_{\mathbb{R}} p_2 \varphi_{\zeta_2}(x) 1_{p_2 \varphi_{\zeta_2} \leq p_1 \varphi_{\zeta_1}} dx \end{aligned} \tag{31}$$

We illustrate the computation of the overlap in some hard-hitting cases below, showing relation between the level of entropy and the individual standard deviations with the overlap measured in Figure 4. Means of component 1 and 2 are 5.28 and 8.45. Panels A and C correspond to balanced classes, while in panel B and D, class 1 is more abundant with a frequency of 0.9. Finally, in panels A and B, the variance of component 1 is smaller than the variance of component 2 with respective SDs of 1 and 3 and reciprocally for panels B and D. Interestingly, in panel D, using the MAP as defined in Equation (3), all observations issued from class 2 are wrongly assigned to class 1. The red area corresponds to the probability of misclassifying component 1 as component 2, while the green area corresponds to the probability of misclassifying component 2 as component 1. Total overlap is since the sum of red and green area, in Figure 4.

There are two intersection points, x_1 and x_2 , with $\mu_1 < \mu_2$ when solving equation Equation (32):

$$p_1 \varphi(x, \mu_1, \sigma_1) = p_2 \varphi(x, \mu_2, \sigma_2) \tag{32}$$

in following case: if $\sigma_2 > \sigma_1$, then we must have $p_1 > \frac{\sigma_1}{\sigma_1 + \sigma_2}$, else if $\sigma_2 < \sigma_1$, then $p_1 < \frac{\sigma_1}{\sigma_1 + \sigma_2}$. In that case, they are given by following formula Equation (33):

$$(x_1, x_2) = \left(\frac{\sigma_1^2 \mu_2 - \sigma_2^2 \mu_1 \pm \sigma_1 \sigma_2 \sqrt{(\mu_1 - \mu_2)^2 + 2(\sigma_2^2 - \sigma_1^2) \left[\log(\frac{p1}{p2}) + \log(\frac{\sigma_2}{\sigma_1}) \right]}}{\sigma_1^2 - \sigma_2^2} \right) \tag{33}$$

Again, sign of term A and order of the roots yield two several cases, depending whether σ_1 is greater or not than σ_2. Both situations with unbalanced classes are illustrated in panel B and D on Figure 4:

- When $\sigma_1 < \sigma_2$, then $x_2 < x_1$ and $p_1 \varphi(x, \mu_1, \sigma_1) < p_2 \varphi(x, \mu_2, \sigma_2)$ on interval $[x_2, x_1]$. Hence, total overlap is given by Equation (34):

$$\text{OVL}(1,2) = p_1 \left(\Phi(x_2, \mu_1, \sigma_1) + 1 - \Phi(x_1, \mu_1, \sigma_1) \right) + p_2 \left(\Phi(x_1, \mu_2, \sigma_2) - \Phi(x_2, \mu_2, \sigma_2) \right) \tag{34}$$

- When $\sigma_1 > \sigma_2$, then $x_1 < x_2$ and $p_1 \varphi(x, \mu_1, \sigma_1) < p_2 \varphi(x, \mu_2, \sigma_2)$ on interval $[x_1, x_2]$. Hence, total overlap is given by Equation (35):

$$\text{OVL}(1,2) = p_2 \left(\Phi(x_1, \mu_2, \sigma_2) + 1 - \Phi(x_2, \mu_2, \sigma_2) \right) + p_1 \left(\Phi(x_2, \mu_1, \sigma_1) - \Phi(x_1, \mu_1, \sigma_1) \right) \tag{35}$$

An interesting result is obtained with the homoscedascity and balanced classes' assumptions of

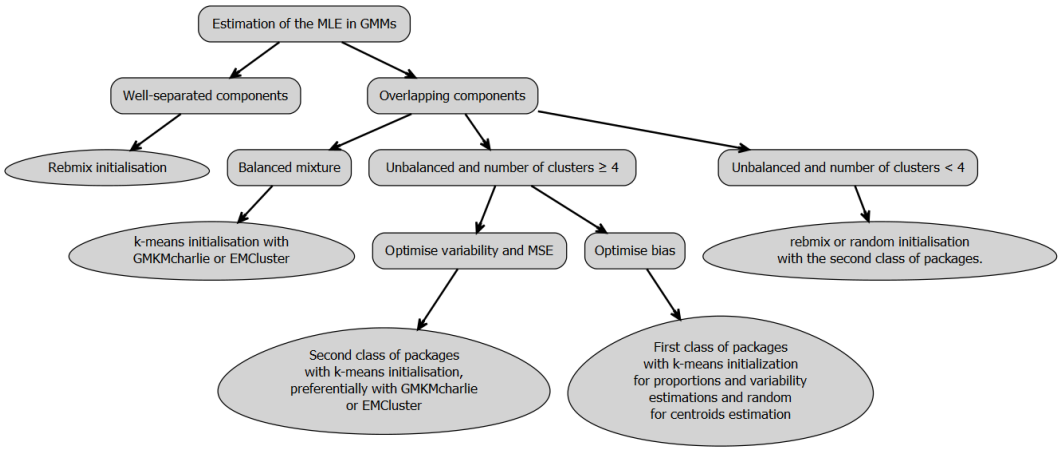


Figure 3: A decision tree to select the best combination of package and initialisation method with respect to the main characteristics of the mixture. It's worth pointing that in both univariate and low dimension multivariate settings, the recommandations are similar.

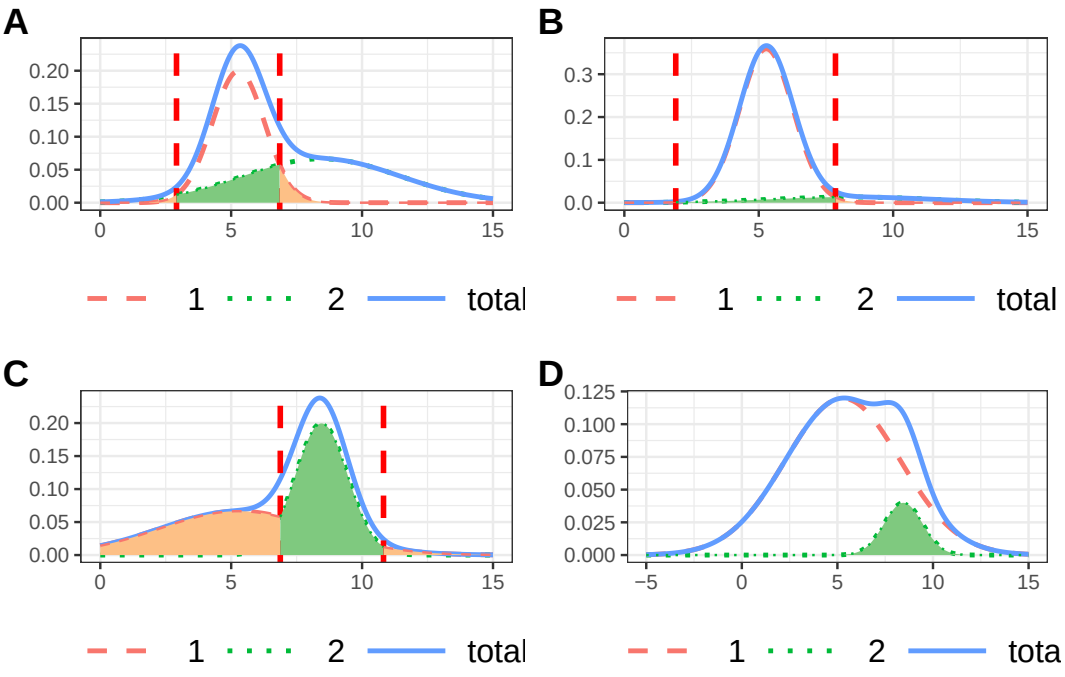


Figure 4: Illustration of the overlaps between a two-components GMM. Density function of component 1 is given by the red line, its of component 2 by the green line, and total density function $f_\theta(X)$ is represented in blue. The total overlap is given by the sum of the green and red areas.

the k-means algorithm. There is only one intersection point in that case: $x_c = \frac{\mu_1+\mu_2}{2}$, that is simply the centre of the segment bounded by the means of the two components. The overlap is simply then $\text{OVL}(1,2) = 2\Phi(-\frac{|\mu_1-\mu_2|}{2\sigma})$.

To our knowledge, no closed formula has been determined returning the overlap generalised to more than two components (combinatorial set of inequations to solve), in the unconstrained multivariate setting (cubic equation to solve in bidimensional space). Indeed, even restraining the study to the bivariate setting (the calculation of the OVL then amounts to estimating the zone of intersection between two ellipses), the exact computation of the OVL involves multiple integration and the algebraic resolution of a quartic equation. A first step is provided by [54], stating algebraic conditions for the existence of an intersection region and computing where applicable a closed formula of the OVL between two coplanar ellipses.

Accordingly, only stochastic approximations, relying on randomised algorithms, such as the Monte-Carlo integration with a rejection technique (knowing that the total area under the curve is normalised to one, we randomly simulate observations and the ratio of the number of observations falling in the intersection area is then used as a proxy of the overlap), are available so far [55]–[57].

Variants in the EM algorithm ecosystem

Two main alternatives were developed in parallel to the EM algorithm and are implemented in some of the reviewed packages: the CEM and the SEM algorithm. However, they do not have its theoretical properties, especially guarantee of the consistency of the algorithm.

The M-step of the *classification EM* (CEM) algorithm [58] maximises a function where each observation was assigned to the maximum a posteriori (MAP) estimate Equation (3). It generalises the well-known k-means algorithm making no assumption of homoscedascity or equibalanced clusters. Its main drawback is to not take into account uncertainty of the cluster assignment, inducing *inconsistency* of the algorithm [13]. EM*, referred in [59] and implemented in the **DCEM** package, is a faster implementation of the CEM algorithm, with roughly a twice smaller complexity. To do so, only the posterior distributions associated to the lower half of the most uncertainly assigned observations are re-computed in the E-step of the EM-algorithm. This normally avoids to recompute data that is unlikely to change of cluster attribution from an iteration to another. However, the higher speed of this algorithm has not been theoretically proven, as the gain of running time per iteration of the algorithm may be alleviated by a greater number of steps to reach the convergence.

The *Stochastic EM* (SEM) replaces the MAP value for S in the E-step of the CEM algorithm by a random draw (or N of them in the N- variant of the algorithm) of the posterior distribution $\mathbb{P}_\theta(S|X)$. As this algorithm does not converge to a unique solution, but rather oscillates around a local maximum, the estimation is usually performed by averaging the late estimated values while ignoring the first estimates from the *burn-in phase*. A theoretical description of these algorithms, with discussion on their convergence properties, is detailed in [11]. SEM algorithm has also a relatively faster convergence than EM algorithm but it is more proned to be trapped in a local maximum or to remove a component. Increasing the number of draws N may alleviate this issue, but at the extent of computational performances.

A wide variety of fast algorithms derived from the EM algorithm have been developed. cwEMM (component-wise EM algorithm), described in [60], is a variation of the EM algorithm aiming at speeding up its convergence. The M-step at each iteration is only performed for one of the components $\theta_j = (p_j, \mu_j, \sigma_j)$, implying that the parameters of a given component are estimated sequentially rather than simultaneously. The theory behind relies on a *Gauss-Seidel* scheme and was first used by the SAGE algorithm. However, the constraints on the proportions set in Equation (2) are only guaranteed if the algorithm converges. Additionally, faster convergence is not theoretically proven for any situation. A list of general acceleration methods for the EM algorithm, not specific to GMMs, is available on **turboEM** [61].

Other EM-inspired algorithms focus on counterbalancing the main limitations of the EM algorithm. The *Variational Bayesian EM* (VBEM) algorithm performs a Bayesian estimation of the parameters. Indeed, the large space of all possible parameter estimates Θ can be hard to explore and the usual initialisation methods are uninformative, not taking into account expert recommendations. VBEM uses these prior assumptions on the parameters' distribution $\mathbb{P}(\theta)$ to optimise the posterior distribution $\mathbb{P}(\theta|X)$, based on Bayes' rule. Direct determination of the Bayesian posterior law of the parameters is generally an intractable problem, hence Variational Bayes only maximises an approximation of the true posterior, assuming that the parameters can be partitioned in independent distributions. This hypothesis is known as *mean-field approximation* [62].

The minimum message length (MML) EM algorithm, implemented in the **GMKMcharlie** package, is a completely unsupervised algorithm as it does not require any prior selection of the number of

components [63] , by dealing explicitly with the possibility of discarding a component during the iteration. To do so, the selection criteria for the number of components is directly included in the optimisation procedure. However, its implementation is close from a Bayesian estimation of the parameters of the model, setting a non-informative Dirichlet prior distribution on the ratios and the higher expected performances of the algorithm are not demonstrated on real use cases [63].

The Expectation/Conditional Maximisation (ECM) [64] belongs to the super-family of GEM (general EM) algorithms, generally used when the maximisation of the auxiliary function yields a non-closed form to solve. To do so, the ECM algorithm replaces the intractable M-step of the EM algorithm by a number of computationally simpler conditional maximization (CM) steps (instead of inferring all parameters at once, the conditional step retrieves a set of optimal parameters, conditioned by the current value of the others). ECM is for instance used with GMMs including an additional linear constraint on the means of the components, as provided by the **mixtools** package. As documented in Table 2, **EMMIXmfa** implements an extension of the ECM, termed alternating expectation–conditional maximization (AECM) algorithm [65], and which can be used to reduce the computational burden of estimating the parameters of mixtures of factor analysers. The AECM algorithm is an extension of the ECM algorithm that allows the complete data used for estimation to differ on each CM-step (generally, in order to speed the computation, by selecting a subset of the most leveraged observations). GEM algorithms share the same asymptotic theoretical properties of the EM algorithm, especially the local consistency of the estimates returned.

2 Appendix B: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms

General workflow

Table 6 lists the terms used in the search, the number of packages returned by the search, the number of packages excluded from review after the search, and the names of the packages ultimately selected for review. Indeed, the CRAN and Bioconductor platforms are the two most popular repositories for R packages, with a constraining review before publication.

Most packages we excluded from review did not focus on the GMM model, but on supplying tools for visualising and asserting the quality of a given clustering. For instance, the search term "cluster" returned many packages implementing other unsupervised clustering methods, such as *k*-means, KNN or graph clustering, were specifically dedicated to specific data, such as single cell analyses. The search term "mixture" returned either packages dealing only with non-Gaussian components, such as **fitmix** with log-normal distributions or were dedicated to chemical mixture designs.

Table 6: Meta-analysis summary about the selection of packages implementing the estimation of GMMs, on CRAN and Bioconductor.

Platforms	Searched terms	Number of returned packages	Number of packages implementing GMMs	Packages implementing GMMs	Packages kept
Bioconductor	mixture	15	3	epigenomix, fmrs, semisup	∅
Bioconductor	cluster	69	1	Melissa	∅
CRAN	mixture	179	44	AdaptGauss, bgmm bmixture, bpgmm, CAMAN, ClusterR, deepgmm DPP, dppmix, EMCluster, EMMIXgene EMMIXmfa, fabMix, flexmix, fmerPack, GMKMcharlie, IMIX, ManlyMix mclust, MGMM, mixAK, MixAll, mixdist, mixR mixreg, mixsmsn, mixtools, mixture MixtureInf, MMDvariance, nor1mix pcensmix, pgmm, pmclust, polySegratioMM rebmix, Rmixmod, RMixtComp RobMixReg, RPMM, SAGMM, sensory, SMNCensReg	bgmm, EMCluster flexmix, GMKMcharlie mclust, mixtools, Rmixmod
CRAN	cluster	418	16	ClusterR, clustMD, DCEM, EMCluster, HDclassif ManlyMix, mclust, mixAK, MixAll mixture, oclust, otrimle, pmclust, rebmix Rmixmod, tclust	EMCluster mclust, Rmixmod

At this stage, too many packages for a tractable benchmark remained. We hence perform stricter selection of them, based on the following criteria:

- Some of the packages did not implement the unconstrained GMM (no constraint of homoscedasticity or equibalanced proportions). Hence, **epigenomix** [66] , **EMMIXgene** [67] , **pcensmix** [68] , **mixAK** [69] (homoscedastic components), **mixture** [70] (multi-dimension only), **AdaptGauss** [71] and **MMDvariance** [72] add constraints on the number of components, on the standard deviation of each component or on mean values of each population, leaving no choice to the user to remove such assumptions. **semisup** [73] restrains on mixtures with two components, for

which a part of the observations are labelled. Additionally, it is designed for GWAS or differential analyses. Other packages were designed to deal with high-dimensional data, projecting the data on a smaller subspace using a factor analysis model. Hence, these packages can not learn a GMM for an univariate distribution, as we can not project on a smaller space than the unidimensional space. This led to the exclusion of **HDclassif**, **fabMix** [74] , **EMMIXmfa** and **pgmm** [75] packages. The **sensory** [76] package both imputes missing data and performs factor regression on a subspace up to 3 dimensions at most, but requires the user to provide its own initial estimates. Alternatively, **clustvarsel** [77] discards the least informative variables, in an attempt to find a locally optimal subset of variables that best discriminate clusters.

- We assume that our original data is continuous. However, some packages are dedicated to deal with discrete data, for instance binned size distributions of medical patients. This led to the exclusion of **mixdist** [78].
- We restrained our review to packages that use the classic EM algorithm, using maximum likelihood estimation to retrieve the parameters of GMMs. For instance, some packages offer a Bayesian estimation of the parameters of the model using MCMC methods, such as **bmixture** [79]], **bpgmm** [80], **DPP** [81] , **dppmix** [82], **BayesCR** [83] and **Melissa** [84]. **polySegratioMM** [85] uses the Bayesian framework JAGS's interface in R. Alternatively, other algorithms focusing on maximising the likelihood do exist, but rely on different statistical methods, such as **RPMM** [86] which implements a recursive algorithm, and **SAGMM** [87] offering a stochastic approximation.

We then removed the packages in which the MLE estimation of the unconstrained GMM model was an ancillary task:

- We removed the packages that focus on learning mixture of Gaussian regressions such as **fmrs** [88] , **mixreg** [89] or **fmerPack** [90] , an extension of the **flexmix** package with an additional feature selection using the lasso method. **nlsmsn** [91] implements regression of skewed Gaussian mixtures, but in unidimensional space only. **RobMixReg** [92] performs robust regression of Gaussian mixtures using five several methods: CTLERob, a component-wise adaptive trimming likelihood estimation; mixbi, bi-square estimation; mixL, Laplacian distribution; mixt, t-distribution; TLE, trimmed likelihood estimation, and flexmix which only performs flexmix regressions with multiple random starts.
- Some packages were built to deal with highly specific tasks. **RMixtComp** [93] and **clustMD** [94] deal with mixed data (continuous + discrete). The **deepgmm** [95] package learns deep Gaussian mixture models, generalising the classical GMM with multiple layers. **IMIX** [96] focuses on clustering multi-omic data that is learnt with the **mclust** package, and **coseq** [97] implements RNA-Seq transcriptome clustering using the **Rmixmod** package.
- Some extend the EM algorithm on Gaussian distributions and overcome its main limitations. The **MGMM** [98] package deal with missing data, which is not relevant in unique dimension. The **mixsmsn** package estimates skewed GMMs. **SMNCensReg** [99] fit univariate right, left or interval censored data. Some packages offer a robust implementation of the algorithm, automatically trimming possible outliers. **otrimle** models the presence of outliers by an extra component following an improper uniform distribution, while **tclust** and **oclust** automatically removes possible outliers before the estimation step.
- We also removed packages that were limited in their functionalities or complex to install. Indeed, **ClusterR** [100] (*k*-means), **rebmix** (REBMIX), **nor1mix** (univariate dimension only, wrong initialisation process), **MixAll** [101] (random and small EM) do not allow to perform the EM algorithm with its own initial estimates. The function to provide its own initial estimates for the \pkg{DCEM] package is only internal, and not supposed to be available for the common user. **pmclust** [102] depends on the availability of the OpenMPI framework for its parallelised implementation of the EM algorithm.
- We also removed the **MixtureInf** [103] , **mixR** [104] and **CAMAN** [105] packages which have not been updated in the last two years or still in beta version.

The popularity of the selected packages varies largely, as shown in Figure 5. Among them, **mclust** and **flexmix** are largely the most popular, followed by **mixtools** and increasingly popular **Rmixmod** package.

Only the packages dedicated to high-dimensionality, listed in our first bullet point, are relevant to benchmark their performance as a function of the number of dimensions. Indeed, although some packages computing mixtures of regressions do implement features allowing to to handle high-dimensional datasets, such as **RobMixReg** and **fmerPack**, they all assume a diagonal covariance structure, and accordingly independent covariates.

The two existing strategies are then limited to projection to a smaller subspace, usually within the theoretical framework of factor analysis or to perform a feature selection strategy. We quickly discarded **fabMix**, since it only retrieves the parameters of GMMs within a Bayesian framework, while we focused on strategies retrieving the MLE via the EM algorithm. The core function `pgmmEM` in the **pgmm** package unfortunately includes a seed for the the algorithm's initialisation that cannot be disabled. Such a feature is generally not recommended for reproducibility, since by defining the seed internally in the function, we were not able to independently generate new reproducible datasets in our benchmark (instead, it is recommended to set the seed value once and for all at the beginning of the virtual simulation). Additionally, while implementing the same AECM variant of the EM algorithm as **EMMIXmfa**, as detailed in Appendix B: the meta-analysis workflow for the final selection of CRAN and Bioconductor platforms, its convergence criteria differs from the other benchmarked packages. Indeed, instead of considering a limiting number of iterations along with a prior threshold, either *absolute* or *relative*, it examines only the difference between the current value of the log-likelihood and a corresponding asymptotic estimate, based on the Aitken acceleration [106]. In brief, the asymptotic value of the log-likelihood is the limiting sum of a geometric series, whose common ratio, the so-called Aitken acceleration, is the relative fraction of the log-likelihood gain of the current iteration. Therefore, the use of a different termination criterion precludes any further fair comparison with the other benchmarked packages, as there is no direct equivalence between the two methods.

Finally, **clustvarsel** is not really tailored for datasets with a large number of dimensions, but rather for datasets with a small number of observations. Indeed, by performing a sequential search in the model space in a forward-backward process (namely by adding variables to the null model till we recover the full model, with all features), the algorithm requires intensive computational resources (for instance, there are already $2^{10} = 1024$ models to be tested in dimension 10). Instead of a greedy strategy, it would have been interesting to implement a in independent and highly parallelizable feature selection process by sampling the model space. To do so, [107], [108] suggests a stochastic and greedy feature selection strategy, using notably the *eBIC* criteria in order to have an equal chance to draw a model of any dimension[14]. Such a strategy is commonly used in *ensemble learning*.

Practical details for the implementation of our benchmark

First, the number of observations ($n = 200$ and $n = 500$ respectively in the univariate and bivariate setting) was chosen enough high to both lower the probability of generating a sample without drawing any observation from one of the components in case of highly-unbalanced clusters and decreases the *margin of error* related to the random sampling error. Specifically, the probability of generating at least one simulation among the N generated fo which less than two observations proceed from component j (the minimal number of elements required to estimate both the mean and the variance of the corresponding cluster), with a two-components mixture of n observations, is given by the following formula (Equation (36)):

$$1 - \left(1 - (1 - p_j)^n - n \times (1 - p_j)^{n-1} \times p_j\right)^N \tag{36}$$

Interestingly, the probability of generating a sample among the N repetitions increases exponentially as the level of imbalance increases. For instance, considering $N = 100$ repetitions, $n = 200$ observations per sample and proportion for the minor component $p_j = 0.1$, the probability of generating a degenerate simulation is insignificant: 1.63×10^{-6} while the risk considerably increases, keeping the same general simulation parameters and setting minor proportion to $p_j = 0.05$, with a probability of 0.04. We have focused on one of the impacts of high dimensionality, namely that related to the homogenisation and convergence of any distance norm and the increase in sparsity in relation with the number of features added. We deliberately do not consider the case where the number of dimensions exceeds the number of observations (namely, when $D > n$), since in this configuration, the covariance matrix is no longer of full rank and invertible, implying that the corresponding probability distribution does spans completely over a smaller subspace. However, with so few observations, ($n = 200$ in scenarios identified as a), we reveal the impact in terms of the quality of the estimation when the number of observations is closed to the number of free parameters required to parametrise the full GMM model (with $k = 2$ clusters and $D = 10$ dimensions, $k \times \frac{D(D+1)}{2} + kD + 1 = 131$ are needed.).

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

[14] Indeed, by simply uniformly sampling among the 2^D models available, the probability of getting models with $D/2$ features is much higher than drawing models at the boundaries, displaying either few or close to $|D|$ covariates.

0.05 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, as no restrictive and difficult-to-test assumptions are required.

If all the seven reviewed packages accept initial estimates provided by the user, both the input and the output format differ between them, requiring an intensive processing to standardise both the initial estimates input, and the output estimates. Notably, a well-known issue with the mixture models is that they identifiable up to a permutation of the components (alternatively, changing the index of the labels do not change the likelihood of the model). Assigning one component of the mixture to a specific index is generally immaterial, as the main objective is to return the estimates. However, when it comes to compare the estimated parameters with the true estimates, we must associate unequivocally each component to a specific index. To do so, we set a partial ordering, sorting the components by increasing order of their mean components. Actually, if the ratio or the covariance estimates can be equal for all the components, it is generally not the case for the centroids, as this would result into a degenerate distribution. The consequence and some illustrations of the non-identifiability of the mixture distributions are discussed in section Identifiability of finite mixture models, in [109] and in Book [110].

We detail below some additional functions we implement to both homogenise input and output of the packages and ease the user's task when comparing the performance of these packages:

- The input observations, mean and covariance matrices have to be transposed compared to the conventional format in packages **bgmm**, **EMCluster**, **GMKMcharlie** and **Rmixmod**, namely $D \times k$ mean matrix and $D^2 \times k$ covariance array (D^2 matrix to store each component variance).
- To save some storage, the **EMCluster** package reshapes the covariance matrix, benefiting from its symmetry. Hence, instead of a three-dimensional array, **EMCluster** expects a compressed $k \times \frac{D(D+1)}{2}$ matrix, each line storing the upper triangular part of the covariance. The memory gain is yet controversial, as decreasing only by a factor two the total space required for the computation. To switch from one format to another, we developed specifically two functions: *trig_mat_to_array()* and *array_to_trig_mat()* in our GitHub package *RGMMBench*, partly inspiring from *vec2sym* function Handy R functions.
- Instead of the covariance matrix, the **mclust** package requires the lower triangular matrix resulting from its Cholesky decomposition. One of the main advantages of this input, in addition to save storage space, is that it ensures that the covariance matrix is indeed positive-definite, as the Cholesky factorisation is only defined if this condition is respected Cholesky decomposition.
- **flexmix** starts by the M-step of the EM algorithm instead of the E-step. Hence, it expects the posterior probabilities assigned to each cluster j for each observation i, $\eta_i(j)$ (Equation (3)), instead of the initial estimates. Both approaches are, however, equivalent.

On the contrary, none of the packages we evaluated that were dedicated to high-dimensional datasets allow the user to provide its own estimates. Thus, when any of the benchmarked initialisation methods listed in Table 3, was internally available in the package, we use it with the same hyperparameters described in Initialisation of the EM algorithm. If not, we provide instead a vector containing the MAP assignments inferred by the native initialisation method, in a process similar to that used used with hierarchical clustering.

In addition to the plots displaying the bootstrap parameter estimations associated to Scenarios in Tables 7 and 12, we have computed summary statistics to compare the performances of the reviewed packages:

- The *bias* measures the deviation between the sample mean value of the estimate and the true parameter: $\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta$.
- The Mean Squared Error (MSE) summarises both the variability of the estimator and its bias: $\text{MSE}(\hat{\theta}) = \mathbb{E}\left[(\hat{\theta} - \theta)^2\right] = \text{var}(\hat{\theta}) + \text{Bias}(\hat{\theta})^2$, where $\text{var}(\hat{\theta})$ is the empiric variance of each estimator given by the diagonal terms of the empiric covariance matrix.
- We enumerate the number of successes (either the package or the initial method returns an error, or fails in returning a set of parameters enforcing standard constraints of multivariate GMMs, namely the unit simplex constraint over the ratios, positive-definite covariance matrices and in general no missing or infinite value).
- For each scenario, we measured independently the running times taken by the initialisation step and by the estimation of the parameters by the EM algorithm. To do so, the **microbenchmark** package [111] was used for its higher accuracy and flexibility for the computation of the running times in place of `System.time`.

The main differences across packages as well as performance results obtained across packages in each univariate, bivariate and high-dimensional simulation scenario are thoroughly described in the next section.

EM-implementation differences across reviewed packages

Most of the distinct behaviours between the packages result from additional choices external to the EM algorithm itself, aiming at partly overcoming its main limitations (Panel B, Figure 11). We detail below their differences ranked by decreasing order of their leverage effect on the final estimate:

1. Most of the differences between the two classes of packages (Figure 2) are related to the either relative or absolute choice for the termination criterion of the EM algorithm. Given an user-defined threshold, the *absolute method* early stops the estimation by comparing the difference between two consecutive log-likelihoods, $|\ell(\hat{\theta}_q|X) - \ell(\hat{\theta}_{q-1}|X)|$, while the *relative method* examines the variation rate, $\left|\frac{\ell(\hat{\theta}_q|X)-\ell(\hat{\theta}_{q-1}|X)}{\ell(\hat{\theta}_{q-1}|X)}\right|$.
2. Several methods can be used to deal with numerical underflow, mostly happening with highly unlikely observations, distant from any centroid.
    - The least elaborate feature is from **Rmixmod**, returning an error when either any of the posterior probabilities or any of the estimated parameters goes below to the precision threshold of the machine (2.22×10^{-16} for most OS).
    - If the maximal value of any posterior probability is null, **bgmm** subtracts the minimal logarithm posterior probability to any log-computed probability. This method avoids numerical underflow by preventing computation of null ratios but the correctness of the estimates is no longer enforced[15].
    - The remaining packages handled numeric underflow in a more convincing manner as they guarantee to return the MLE estimate. The **flexmix**, **GMKMcharlie** and **EMCluster** packages use the same log-rescaling tip detailed in (Application of the EM algorithm to GMMs). The **mixtools** and **mclust** packages use a variant of this trick, taking profit of the factorisation by the greatest element (Equation (37), Equation 3 p.5 [112]), but without exploiting the tip of Taylor's development over $\log(1+x)$:

$$\eta_i(j) = \frac{p_j\,\varphi_{\zeta_j}(x)}{\sum_{j=1}^{k} p_j\,\varphi_{\zeta_j}(x)} = \frac{\frac{p_j\,\varphi_{\zeta_j}(x)}{p_{j_{\min}}\,\varphi_{\zeta_{j_{\min}}}(x)}}{1+\frac{\sum_{j\neq j_{\min}} p_j\,\varphi_{\zeta_j}(x)}{p_{j_{\min}}\,\varphi_{\zeta_{j_{\min}}}(x)}} \tag{37}$$

In both cases, the computation of the smallest posterior probability, the most proned to be assigned a null value, is avoided, avoiding inconsistent ratios of type $0/0$.

- The previous two items deal with specific numeric limitations, but do not directly address one of the main theoretical limitation of the EM algorithm, namely the risk of falling into a suboptimal maximum, plateau or getting trapped on the boundary space (occurs when the proportion of one of the component converges to zero). Some packages specifically handle the case of a vanishing component during the EM optimization: the **mixtools** package performs random re-initialisations in case one of the computed variance goes below a user-defined threshold (default 10^{-8}). **flexmix** and **GMKMcharlie** deal explicitly with the removal of a component, by updating the corresponding MLE parameters. **flexmix** removes any component whose associated weight is by default below 0.05 (such a stringent limitation tends to an underestimation of the true number of components in highly unbalanced mixtures)[16], while GMKMcharlie both implements a lower limit on the proportions of the components and an upper threshold over the ratio of the maximum and minimal eigenvalue resulting from the factorisation of the covariance matrix (Equation (24))[17].

We enumerate below some additional features supplied by the packages:

[15] Additionally, **bgmm** does not update the estimated variances if any newly computed variance is below the criterion stop. One remarkable side-effect of these features is the reduced sensitivity of the **bgmm** package to the presence of outliers, as illustrated in Appendix *Gaussian mixture distributions with outliers*.

[16] Indeed, at least one of the component was removed in 80% of our estimations in the unbalanced and overlapping case (scenario U9 in 7) and in 20% of the simulations in the unbalanced and well-separated case (scenario U3 in 7).

[17] These options are set respectively to 0 and $+\infty$ by default, thus they did not impact our simulations

- In addition to log rescaling, **GMKMcharlie** includes an additional argument, *embedNoise*, to avoid degenerate GMMs by adding a small constant to any diagonal term (by default 10^{-6}). Besides, instead of controlling whether there was a relative change of the log-likelihood, the EM implementation of **GMKMcharlie** controls instead that there was no significant relative difference in the estimated parameters in the ten previous optimisations[18]. Finally, since **GMKMcharlie** has implemented a parallelised version of the algorithm, it ensures using a a time limit that the algorithm indeed terminates (by default, set to one hour).
- **flexmix** performs an unbiased estimate of the covariance matrix, instead of the corresponding ML covariance estimate (divides by a factor $n-1$ instead of the number of observations n). Such a choice does not affect the results in our simulations, but may have a stronger impact when fitting models to a small number of observations.
- Similarly to flexmix, the **HDclassif** package implements some constraints to preserve numerical stability. The `min.individuals` attribute, like the `minprior` attribute of `flexmix` function, discards any cluster having fewer observations[19]. However, unlike **flexmix**, the algorithm stops instead of reparametrising the mixture problem with a smaller number of components. Coupled with the *Cattell's scree-test*, the `noise.ctrl` attribute is the minimum threshold of a feature's contribution to the overall variance, computed as the corresponding normalised eigenvalue, in order to be included in the mixture of factor analysers. This additional constraint ensures a parsimonious dimension selection process, so that the number of selected intrinsic dimensions cannot be greater than or equal to the order of the discarded eigenvalues.

3 Appendix C: comprehensive report from the univariate and bivariate benchmark

Supplementary Figures and Tables in the univariate simulation

Table below (7) lists the complete set of parameters used to simulate the univariate Gaussian mixture distribution in our benchmark:

Table 7: 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

Figure 6-Figure 10 each summarise the benchmarking results associated with one of the scenarios listed in Table 7.

Summary tables 8- 11 display the average performance for each package of the benchmark with each initialisation method. The best performing pair (lowest bias or MSE) is highlighted in green, and the worst performing in red. The MSE and bias columns were derived by summing respectively the estimated proportions, means and standard deviations associated with the individual components.

[18] In our simulation, the behaviour of the **GMKMcharlie** did not differ significantly from the remaining packages of the second class. However, the use of an Euclidean distance criterion may be problematic when parameters are not on the same order of magnitude, requiring their prior normalisation

[19] by default, set to two, i.e. the minimum number of replications to derive an unbiased estimate of the empirical variance of a sample

Table 8: MSE and Bias associated to scenario U1, in Table 7 (balanced and well-separated components)

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	kmeans	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	quantiles	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	random	0.0061	1.90000	0.19000	0.0290	0.5300	0.1100
	rebmix	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
flexmix	hc	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	kmeans	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	quantiles	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	random	0.0064	1.80000	0.19000	0.0290	0.5300	0.1100
	rebmix	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
mclust / bgmm	hc	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	kmeans	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	quantiles	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	random	0.0062	1.80000	0.19000	0.0290	0.5300	0.1100
	rebmix	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
mixtools	hc	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	kmeans	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	quantiles	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	random	0.0064	1.90000	0.19000	0.0290	0.5300	0.1100
	rebmix	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
Rmixmod / RGMMBench	hc	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	kmeans	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	quantiles	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028
	random	0.0064	1.90000	0.19000	0.0290	0.5300	0.1100
	rebmix	0.0004	0.00077	0.00037	0.0025	0.0068	0.0028

The panels indexed by the B letter, from Figure 6 to Figure 10, display the 0.05, 0.5 and 0.95 quantiles of the distribution of the operating times taken for parameter estimation, for the scenarios listed in Table 7.

First, we note that the execution time grows asymptotically linearly with the number of observations, confirming empirically the expected linear complexity of the EM algorithm. The most important factor playing on the differences observed is related to the complexity of the distribution, and especially the degree of overlap between the components:

- On the one hand hand, when components are well-separated (scenarios 1 and 3 in Table 7), the estimation of the parameters is simple, leading to a reduced number of iterations required to reach the convergence and shorter running times.
- On the other hand, the time taken by the slowest package for the estimation of the parameters increases by a hundred factor with the most complex scenario (see scenario U9, 7, illustrated in Figure 9), compared to the simplest scenario (see U1, 7, shown in Figure 6). Indeed, the average running time for a complete run of the EM algorithm increases from 0.215 seconds to 10.8 seconds.

To better understand the running times' differences observed between the packages for a given scenario, we perform a three-way anova, taking into account the choice of initialisation method, the programming language and the class of packages[20]:

- With well-separated components (Scenarios U1 and U7 in Table 7), the class of packages (namely the choice of the convergence criterion) has a negligible impact compared to the

[20]To compare whether differences between mean running times or estimation performances differ across packages, we used the between-subjects Anova test `rstatix::anova_test()` to generate the p-values and `rstatix::partial_eta_squared()` to compute the corresponding effect sizes.

Table 9: MSE and Bias associated to scenario U7, in Table 7 (unbalanced and well-separated components)

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.02900	2.8000	0.45000	0.1000	0.840	0.250
	kmeans	0.00730	0.7900	0.13000	0.0260	0.240	0.075
	quantiles	0.16000	19.0000	3.20000	0.6400	6.100	1.800
	random	0.17000	10.5000	1.40000	0.3600	3.100	0.780
	rebmix	0.00027	0.0015	0.00077	0.0025	0.014	0.014
flexmix	hc	0.05500	2.8000	0.45000	0.1100	0.850	0.250
	kmeans	0.00760	0.7800	0.13000	0.0260	0.240	0.075
	quantiles	0.11000	19.0000	3.20000	0.5400	6.000	1.900
	random	0.15000	8.4000	1.00000	0.3000	2.500	0.580
	rebmix	0.00027	0.0015	0.00076	0.0025	0.014	0.011
mclust / bgmm	hc	0.03200	2.8000	0.45000	0.1000	0.850	0.250
	kmeans	0.00740	0.7800	0.13000	0.0260	0.240	0.075
	quantiles	0.14000	19.0000	3.20000	0.6000	6.000	1.900
	random	0.18000	10.4000	1.40000	0.3600	3.100	0.800
	rebmix	0.00027	0.0015	0.00077	0.0025	0.014	0.014
mixtools	hc	0.03200	2.8000	0.45000	0.1000	0.850	0.250
	kmeans	0.00620	0.7600	0.13000	0.0170	0.230	0.079
	quantiles	0.15000	19.0000	3.20000	0.5800	6.000	1.800
	random	0.18000	10.3000	1.40000	0.3600	3.100	0.800
	rebmix	0.00027	0.0015	0.00077	0.0025	0.014	0.014
Rmixmod / RGMMBench	hc	0.02900	2.8000	0.45000	0.1000	0.850	0.250
	kmeans	0.00540	0.7700	0.13000	0.0190	0.230	0.078
	quantiles	0.14000	19.0000	3.20000	0.5900	6.000	1.800
	random	0.17000	10.4000	1.40000	0.3600	3.100	0.800
	rebmix	0.00027	0.0015	0.00077	0.0025	0.014	0.014

choice of initialisation algorithm or the programming language. The effect sizes associated to the programming language and the initialisation method are respectively 1.688×10^{-2} (p-value of 3×10^{-60}) and 13×10^{-5} (p-value of 3×10^{-60}), while the choice of the termination criteria did not significantly impact the execution time, with an effect size of 8.119×10^{-4}(p-value of 0.35). Faster running times with packages natively encoded in Fortran or C compared to those encoded in R only were expected, as R is a high-level programming language known to be slower. Indeed, the **flexmix** package is the slowest, preceded by our baseline R implementation. Additionally, **mclust**, followed by **mixtools**, **Rmixmod** and **bgmm** are the fastest.

- On the other hand, with overlapping components (Scenarios U3 and U9 in Table 7), the package class and the programming language have a statistically significant impact on the average running times (the effect sizes associated to the choice of the termination criteria and the programming language are respectively 0.111 (numerical null p-value) and 0.0852 (p-value of 8×10^{-307})) while the initialisation method has no substantial impact (effect size of 2.967×10^{-4} and p-value of 0.32). In the context of highly overlapping mixture, the fastest ones are **mclust** and **GMKMcharlie**, benefiting from both using relative ratios and a fast programming language, while our baseline implementation `emnmix`, preceded by **Rmixmod** and **mixtools**, are on average a hundred times slower.

Table 10: MSE and Bias associated to scenario U3, in Table 7 (balanced and overlapping components)

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.0170	1.60	0.45	0.1950	1.320	0.93
	kmeans	0.0054	0.81	0.18	0.0125	0.023	0.32
	quantiles	0.0070	0.67	0.30	0.0930	0.590	0.56
	random	0.0440	8.40	1.00	0.0710	0.330	0.63
	rebmix	0.0990	11.00	1.60	0.2000	1.600	0.78
flexmix	hc	0.0260	2.60	0.94	0.1120	1.160	1.22
	kmeans	0.0044	0.67	0.14	0.0036	0.091	0.27
	quantiles	0.0054	0.57	0.27	0.0850	0.670	0.55
	random	0.0420	8.20	1.10	0.0450	0.450	0.68
	rebmix	0.1210	14.30	2.70	0.2700	2.700	1.17
mclust / bgmm	hc	0.0110	2.50	0.84	0.0330	1.160	1.10
	kmeans	0.0068	0.86	0.24	0.0294	0.114	0.36
	quantiles	0.0075	0.70	0.32	0.1110	0.720	0.63
	random	0.0490	9.10	1.20	0.0800	0.320	0.68
	rebmix	0.1410	10.90	2.90	0.2900	1.800	1.47
mixtools	hc	0.0320	2.40	0.80	0.0670	0.360	0.25
	kmeans	0.0415	2.51	1.11	0.1000	0.664	0.74
	quantiles	0.0383	2.40	1.00	0.1170	0.770	0.78
	random	0.0660	9.40	1.80	0.0130	0.340	0.48
	rebmix	0.1090	9.60	2.50	0.2600	1.800	1.33
Rmixmod / RGMMBench	hc	0.0220	2.00	0.67	0.0490	0.370	0.25
	kmeans	0.0318	2.31	0.85	0.0952	0.602	0.67
	quantiles	0.0297	2.19	0.80	0.1210	0.770	0.76
	random	0.0620	9.40	1.70	0.0160	0.310	0.50
	rebmix	0.1140	10.30	2.60	0.2600	1.900	1.31

Supplementary Figures and Tables in the bivariate simulation

Table below (12) lists the complete set of parameters used to simulate the multivariate Gaussian mixture distribution in our benchmark:

Figures 13- 16 are associated to scenarios B11 - B15 of Table 12. Summary tables 13-16 show the average performance for each combination of a benchmarked package and initialisation method, with the same conventions as discussed in Supplementary Figures and Tables in the univariate simulation.

First, we can directly observe that the OVL increases as the individual variance of each component, the proximity of the centroids of the clusters and the level of imbalance is increased. We demonstrate this statement formally in section An analytic formula of the overlap for univariate Gaussian mixtures. Nonetheless, the influence of the correlation between the x and the y-axis (the off-diagonal term of the covariance matrix) is not immediate, notably the assumption of independent features does not automatically entail a lower OVL or simpler estimation.

From our experiments, we deduce that the highest OVL is obtained when the main axis of the two respective components aligns with the line joining the two centroids. For instance, in our scenario, the lowest OVL is obtained when the correlation term is positive for both clusters (scenario 14, Table 12 and isodensity plot in panel A, Figure 15), whereas the highest OVL is obtained with a negative correlation (scenario 11, Table 12and isodensity plot in panel A, Figure 13). Recall that the slope joining the two centroids of the two components in all our simulated distributions is indeed negative.

Table 11: MSE and Bias associated to scenario U9, in Table 7 (unbalanced and overlapping components)

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.230	9.3	0.94	0.78	4.9	1.33
	kmeans	0.094	5.1	0.57	0.50	3.4	0.89
	quantiles	0.230	9.9	1.00	0.80	5.1	1.34
	random	0.270	11.5	0.90	0.63	2.8	0.85
	rebmix	0.330	20.0	2.20	0.53	5.3	0.84
flexmix	hc	0.170	10.5	0.88	0.64	5.2	1.14
	kmeans	0.051	5.6	0.61	0.34	3.6	0.94
	quantiles	0.210	11.3	1.20	0.75	5.6	1.53
	random	0.180	9.5	0.77	0.43	2.7	0.86
	rebmix	0.110	10.0	1.70	0.15	2.3	1.48
mclust / bgmm	hc	0.230	10.2	0.84	0.79	5.1	1.20
	kmeans	0.107	5.5	0.62	0.53	3.6	0.96
	quantiles	0.270	11.4	1.20	0.87	5.6	1.59
	random	0.300	12.2	1.06	0.66	2.9	0.84
	rebmix	0.270	21.0	2.50	0.46	5.2	1.13
mixtools	hc	0.200	9.7	1.19	0.64	3.4	0.69
	kmeans	0.135	7.7	1.16	0.46	2.1	0.48
	quantiles	0.280	11.2	1.60	0.74	4.2	0.72
	random	0.350	15.7	1.62	0.65	2.1	0.64
	rebmix	0.240	22.0	2.70	0.47	5.1	1.18
Rmixmod / RGMMBench	hc	0.210	9.5	1.07	0.69	3.8	0.79
	kmeans	0.113	6.5	0.90	0.46	2.4	0.43
	quantiles	0.240	10.1	1.30	0.74	4.2	0.81
	random	0.320	14.6	1.45	0.61	2.1	0.58
	rebmix	0.250	22.0	2.70	0.49	5.2	1.18

In contrast to the univariate setting (Supplementary Figures and Tables in the univariate simulation), the fastest packages are **bgmm**, **EMCluster**, **flexmix**, and **Rmixmod**, and the slowest ones **mclust**, **GMKMcharlie** and **mixtools**, independently from the difficulty of the simulation.

Finally, Figures 17, 18 and 19 represent in a synthetic way less interesting scenarios benchmarked with to the left, the contour maps and to the right the corresponding Hellinger boxplots, with one scenario being illustrated per row.

Table 12: 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

4 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

Table 13: MSE and Bias associated to scenario B11, in Table 12 (unbalanced, overlapping and negative correlated components).

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.230	3.90	1.8	0.550	2.30	1.200
	kmeans	0.136	2.80	1.9	0.450	2.30	2.200
	random	0.028	1.27	1.1	0.084	0.12	0.140
	rebmix	0.071	2.20	1.4	0.170	0.66	0.111
flexmix	hc	0.260	3.90	1.9	0.480	2.40	1.300
	kmeans	0.077	2.80	1.9	0.270	2.40	2.300
	random	0.028	0.96	1.0	0.064	0.77	0.720
	rebmix	0.087	1.90	1.0	0.170	1.02	0.468
mclust / bgmm	hc	0.230	3.90	1.8	0.550	2.30	1.200
	kmeans	0.136	2.80	1.9	0.450	2.30	2.200
	random	0.028	1.27	1.1	0.084	0.12	0.140
	rebmix	0.071	2.20	1.4	0.170	0.66	0.111
mixtools	hc	0.210	3.30	1.8	0.470	1.80	1.100
	kmeans	0.131	2.60	1.8	0.380	1.80	1.800
	random	0.051	1.61	1.1	0.129	0.20	0.180
	rebmix	0.093	2.40	1.4	0.210	0.60	0.063
Rmixmod / RGMMBench	hc	0.210	3.30	1.8	0.470	1.80	1.100
	kmeans	0.131	2.60	1.8	0.380	1.80	1.800
	random	0.051	1.61	1.1	0.129	0.20	0.180
	rebmix	0.093	2.40	1.4	0.210	0.60	0.063

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

Table 14: MSE and Bias associated to scenario B12, in Table 12 (unbalanced, overlapping and opposite correlated components).

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.00076	0.049	0.16	0.0063	0.056	0.131
	kmeans	0.00076	0.049	0.16	0.0063	0.056	0.131
	random	0.00075	0.049	0.16	0.0057	0.055	0.123
	rebmix	0.00087	0.066	0.17	0.0070	0.063	0.190
flexmix	hc	0.00144	0.049	0.16	0.0101	0.055	0.071
	kmeans	0.00144	0.049	0.16	0.0101	0.055	0.071
	random	0.00144	0.050	0.16	0.0099	0.054	0.067
	rebmix	0.00145	0.048	0.16	0.0142	0.047	0.110
mclust / bgmm	hc	0.00076	0.049	0.16	0.0063	0.056	0.131
	kmeans	0.00076	0.049	0.16	0.0063	0.056	0.131
	random	0.00075	0.049	0.16	0.0057	0.055	0.124
	rebmix	0.00087	0.066	0.17	0.0070	0.063	0.190
mixtools	hc	0.00075	0.050	0.16	0.0049	0.054	0.112
	kmeans	0.00075	0.050	0.16	0.0049	0.054	0.112
	random	0.00075	0.050	0.16	0.0049	0.054	0.112
	rebmix	0.00086	0.066	0.17	0.0061	0.062	0.170
Rmixmod / RGMMBench	hc	0.00075	0.050	0.16	0.0049	0.054	0.112
	kmeans	0.00075	0.050	0.16	0.0049	0.054	0.112
	random	0.00075	0.050	0.16	0.0049	0.054	0.112
	rebmix	0.00086	0.066	0.17	0.0061	0.062	0.170

Etienne Becht
Les Laboratoires Servier
50 Rue Carnot
92150, Suresnes, France
ORCiD: 0000-0003-1859-9202
etienne.becht@servier.com

Table 15: MSE and Bias associated to scenario B14, in Table 12 (unbalanced, overlapping and positive correlated components).

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.00043	0.044	0.13	0.00081	0.044	0.060
	kmeans	0.00043	0.044	0.13	0.00081	0.044	0.060
	random	0.00043	0.044	0.13	0.00080	0.044	0.060
	rebmix	0.00040	0.044	0.13	0.00120	0.047	0.053
flexmix	hc	0.00043	0.044	0.13	0.00072	0.043	0.035
	kmeans	0.00043	0.044	0.13	0.00072	0.043	0.035
	random	0.00043	0.044	0.13	0.00072	0.044	0.035
	rebmix	0.00040	0.044	0.14	0.00110	0.047	0.044
mclust / bgmm	hc	0.00043	0.044	0.13	0.00081	0.044	0.060
	kmeans	0.00043	0.044	0.13	0.00081	0.044	0.060
	random	0.00043	0.044	0.13	0.00080	0.044	0.060
	rebmix	0.00040	0.044	0.13	0.00120	0.047	0.053
mixtools	hc	0.00043	0.044	0.13	0.00078	0.044	0.060
	kmeans	0.00043	0.044	0.13	0.00078	0.044	0.060
	random	0.00043	0.044	0.13	0.00078	0.044	0.060
	rebmix	0.00040	0.044	0.13	0.00110	0.047	0.053
Rmixmod / RGMMBench	hc	0.00043	0.044	0.13	0.00078	0.044	0.060
	kmeans	0.00043	0.044	0.13	0.00078	0.044	0.060
	random	0.00043	0.044	0.13	0.00078	0.044	0.060
	rebmix	0.00040	0.044	0.13	0.00110	0.047	0.053

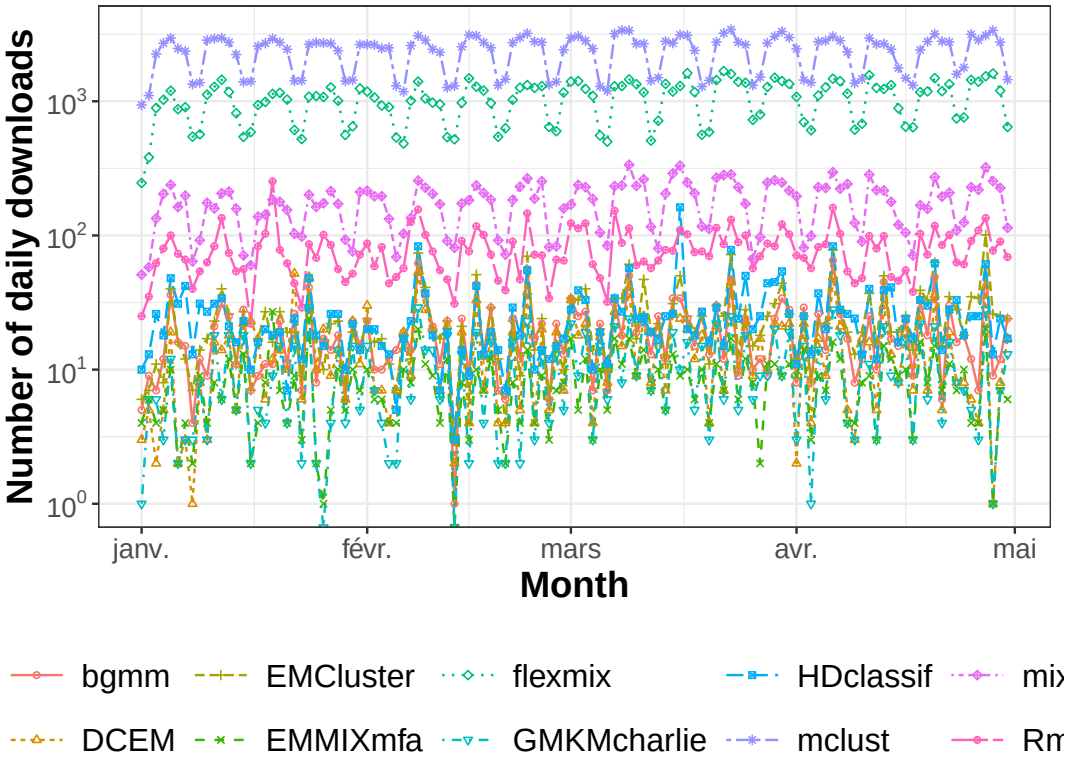


Figure 5: Number of daily downloads (logarithmic scale) from the CRAN mirror from the 1st of January to the 30th April 2022 for the seven R packages reviewed.

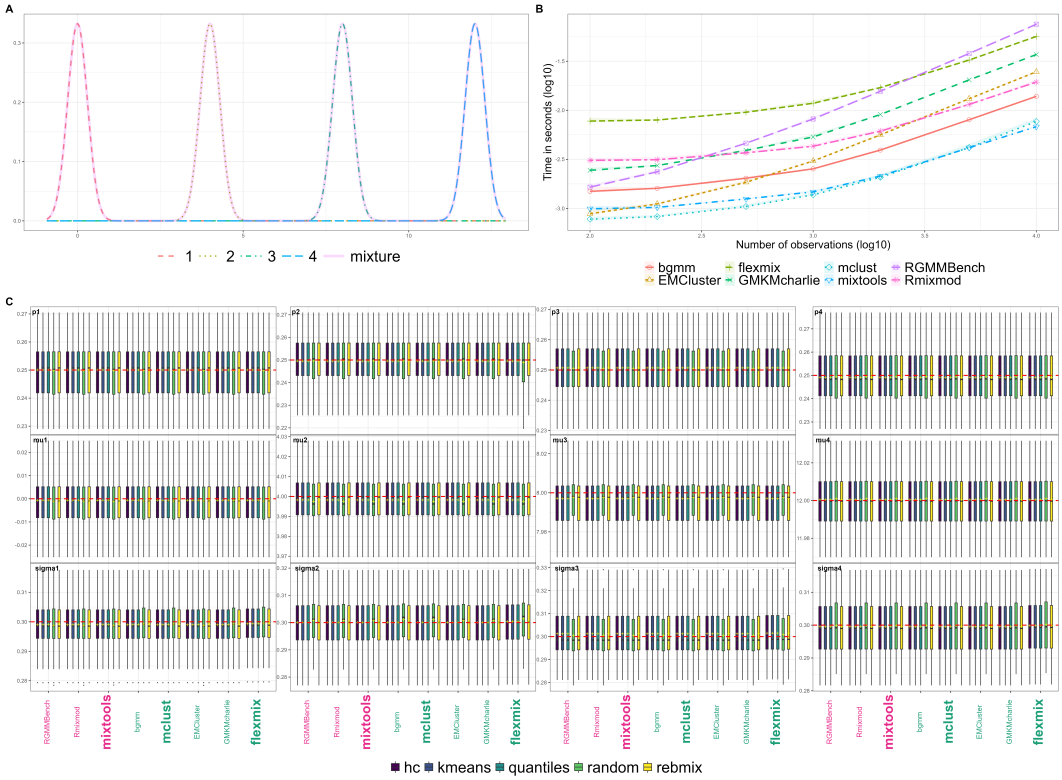


Figure 6: Benchmark summary plots of scenario U1 in Table 7 (balanced and well-separated components), organised as such: The panel A displays the distribution of the global mixture distribution $f_\theta(X)$ (pink solid line) and of each of its constitutive components scaled by their respective proportions (dotted lines). Running times are displayed in Panel B with the k-means initialisation. The number of observations (x-axis) and the running time (y-axis) is in $\log(10)$ scale, implying that any linear relationship between the running time and the number of observations is represented by a slope of 1. The points represent median running time. The coloured bands represent the 5th and 95th percentiles of the running time. In panel C are represented the boxplots associated with the distribution of the estimates, with one box per pair of package and initialisation method. The median is displayed with bold black line, the mean with a yellow cross and the 0.25 and 0.95 quantiles match the edges of the rectangular band. Solid black lines extending past the box boundaries represent the 1.5 IQR, estimates above these limits considered as outliers and omitted from the plot. Finally, the true value of the parameter is represented as a dashed red line. The bold black writing in the upper right-hand corner refers to the parameter whose distribution is shown in the corresponding facet. The first, second and third rows are the distributions of the ratios, means and variances of each component, identified by the column index.

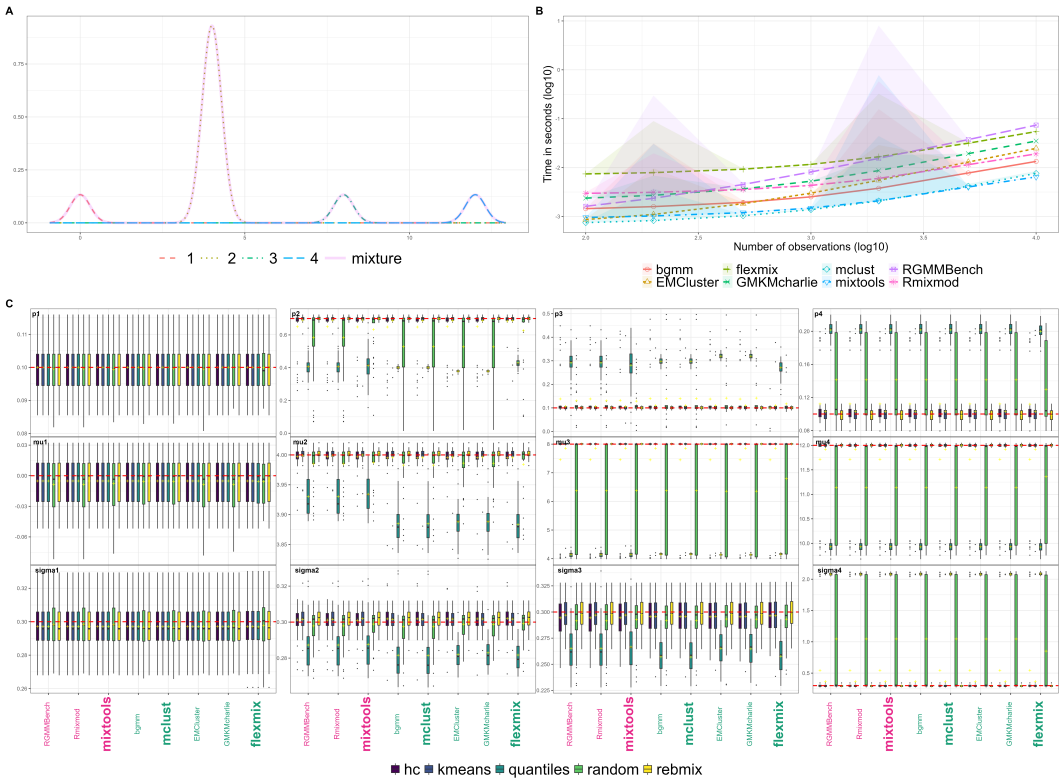


Figure 7: Benchmark summary plots of scenario U7 in Table 7 (unbalanced and well-separated components), with same layout as in Figure 6.

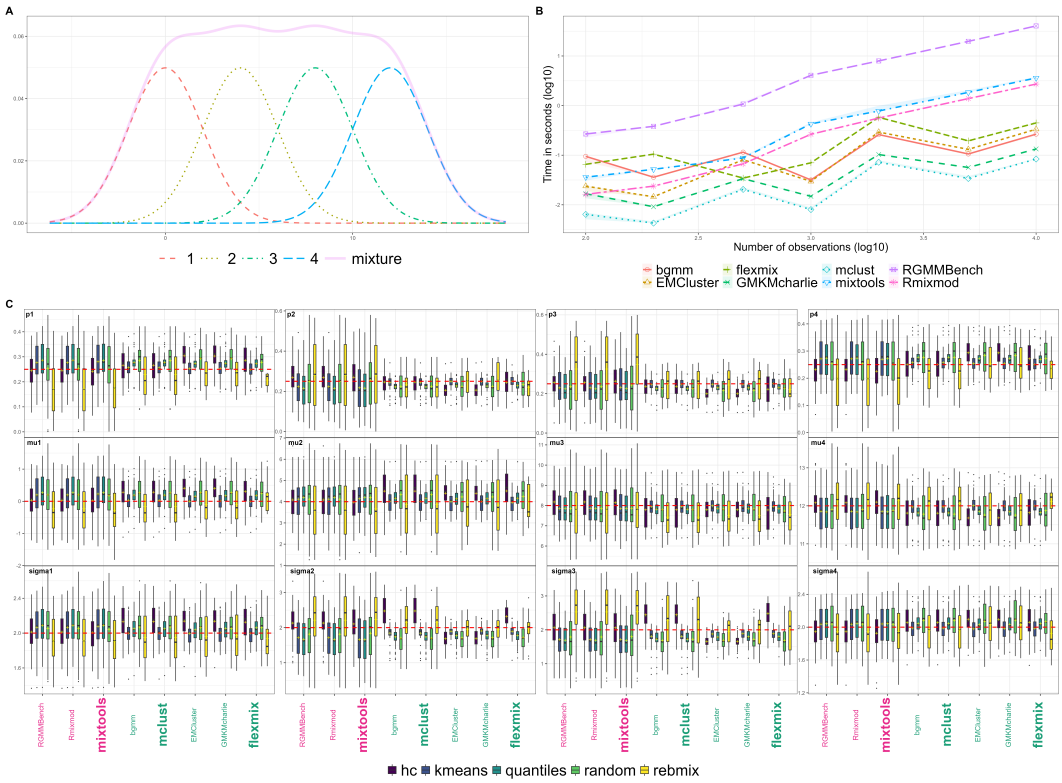


Figure 8: Benchmark summary plots of scenario U3 in Table 7 (balanced and overlapping components), with same layout as in Figure 6.

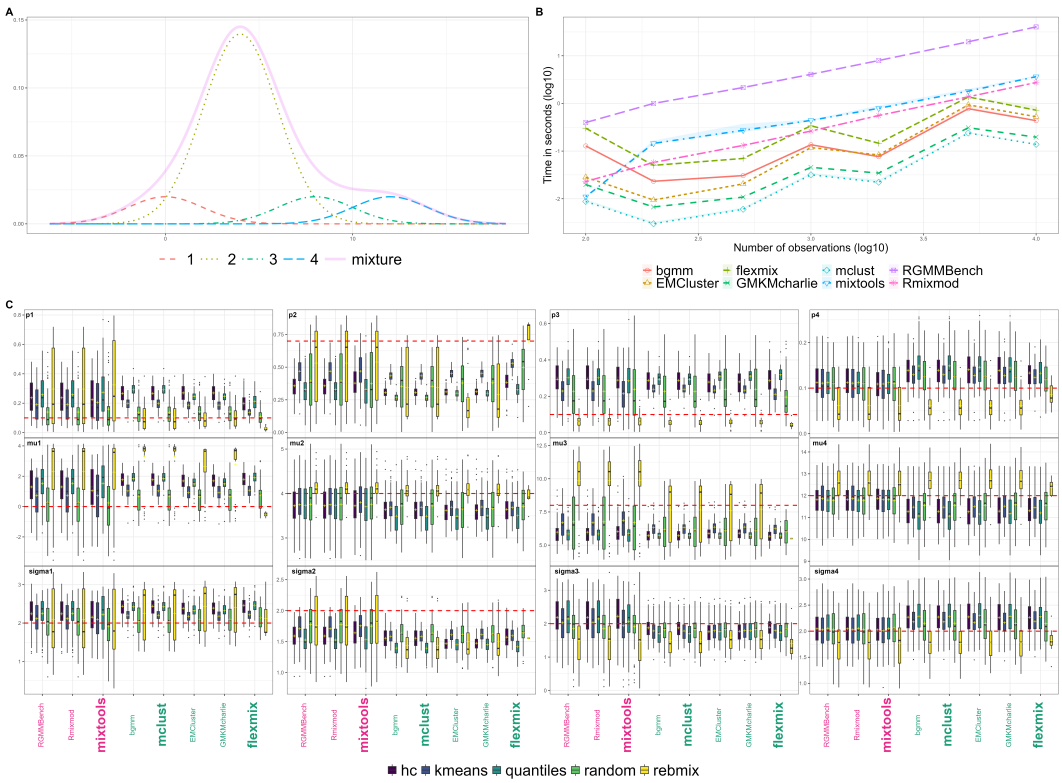


Figure 9: Benchmark summary plots of scenario U9 in Table 7 (unbalanced and overlapping components), with same layout as in Figure 6.

Table 16: MSE and Bias associated to scenario B15, in Table 12 (unbalanced, overlapping and uncorrelated components).

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ
EMCluster / GMKMcharlie	hc	0.1110	2.30	1.30	0.280	1.40	0.90
	kmeans	0.0500	1.50	1.30	0.200	1.05	1.06
	random	0.0290	0.71	0.63	0.070	0.28	0.19
	rebmix	0.0163	0.69	0.78	0.074	0.37	0.44
flexmix	hc	0.1330	2.40	1.40	0.240	1.50	1.05
	kmeans	0.0320	1.60	1.40	0.110	1.21	1.26
	random	0.0370	0.71	0.64	0.048	0.35	0.29
	rebmix	0.0058	0.70	0.84	0.028	0.49	0.62
mclust / bgmm	hc	0.1110	2.30	1.30	0.280	1.40	0.90
	kmeans	0.0500	1.50	1.30	0.200	1.05	1.06
	random	0.0290	0.71	0.63	0.070	0.28	0.19
	rebmix	0.0163	0.69	0.78	0.074	0.37	0.44
mixtools	hc	0.0860	1.90	1.20	0.220	1.10	0.75
	kmeans	0.0470	1.30	1.10	0.170	0.79	0.78
	random	0.0230	0.67	0.66	0.065	0.24	0.19
	rebmix	0.0158	0.69	0.77	0.068	0.30	0.37
Rmixmod / RGMMBench	hc	0.0860	1.90	1.20	0.220	1.10	0.75
	kmeans	0.0470	1.30	1.10	0.170	0.79	0.78
	random	0.0230	0.67	0.66	0.065	0.24	0.19
	rebmix	0.0158	0.69	0.77	0.068	0.30	0.37

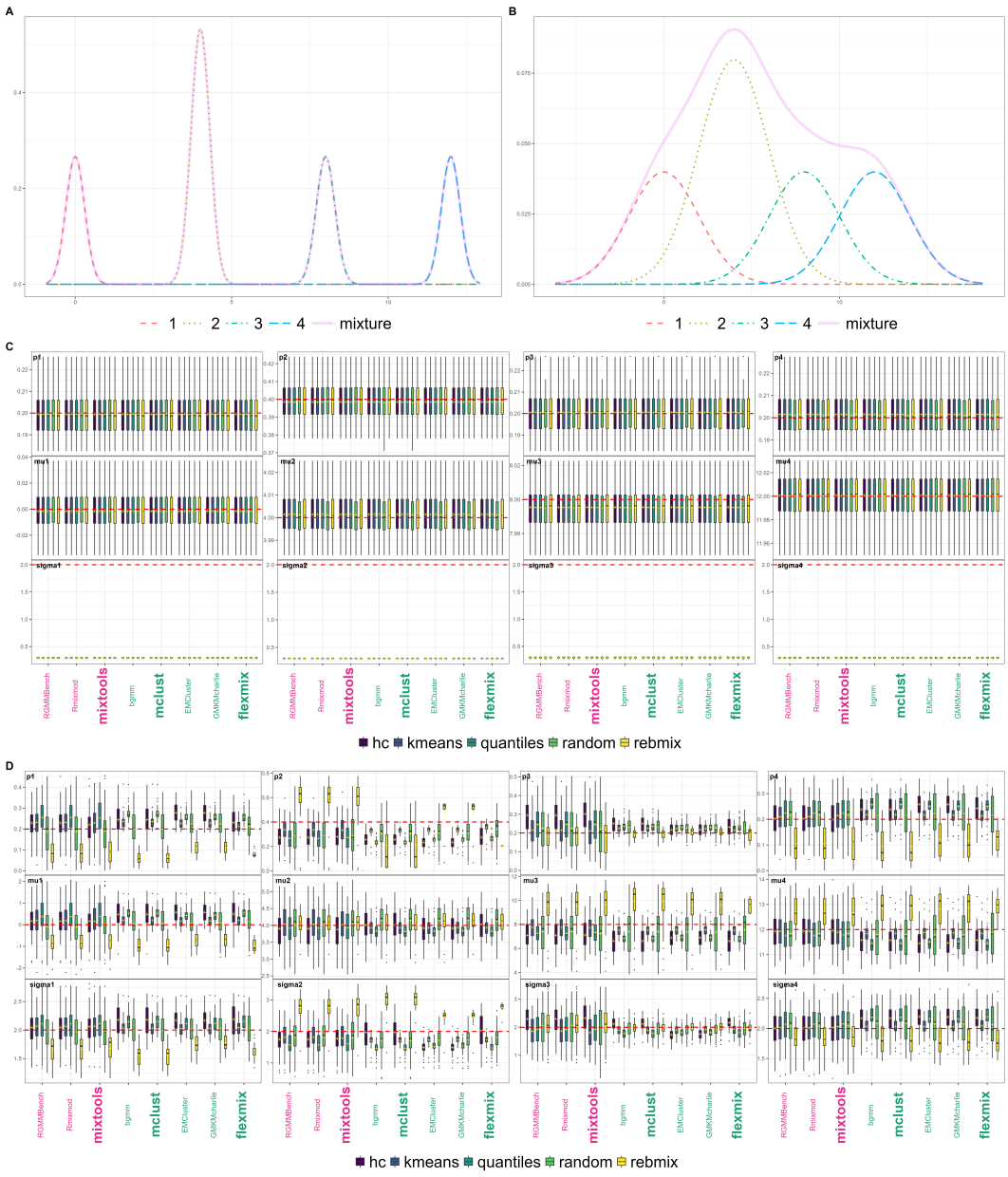


Figure 10: Benchmark summary plots of scenarios U4 and U6 in Table 7 (small unbalance, with additional overlap in scenario U6). Panel A and B display the univariate GMM distributions of respectively scenarios U4 and U6, and Panel C and D the benchmarked distributions of respectively scenarios U4 and U6, built as Panel C of Figure 6.

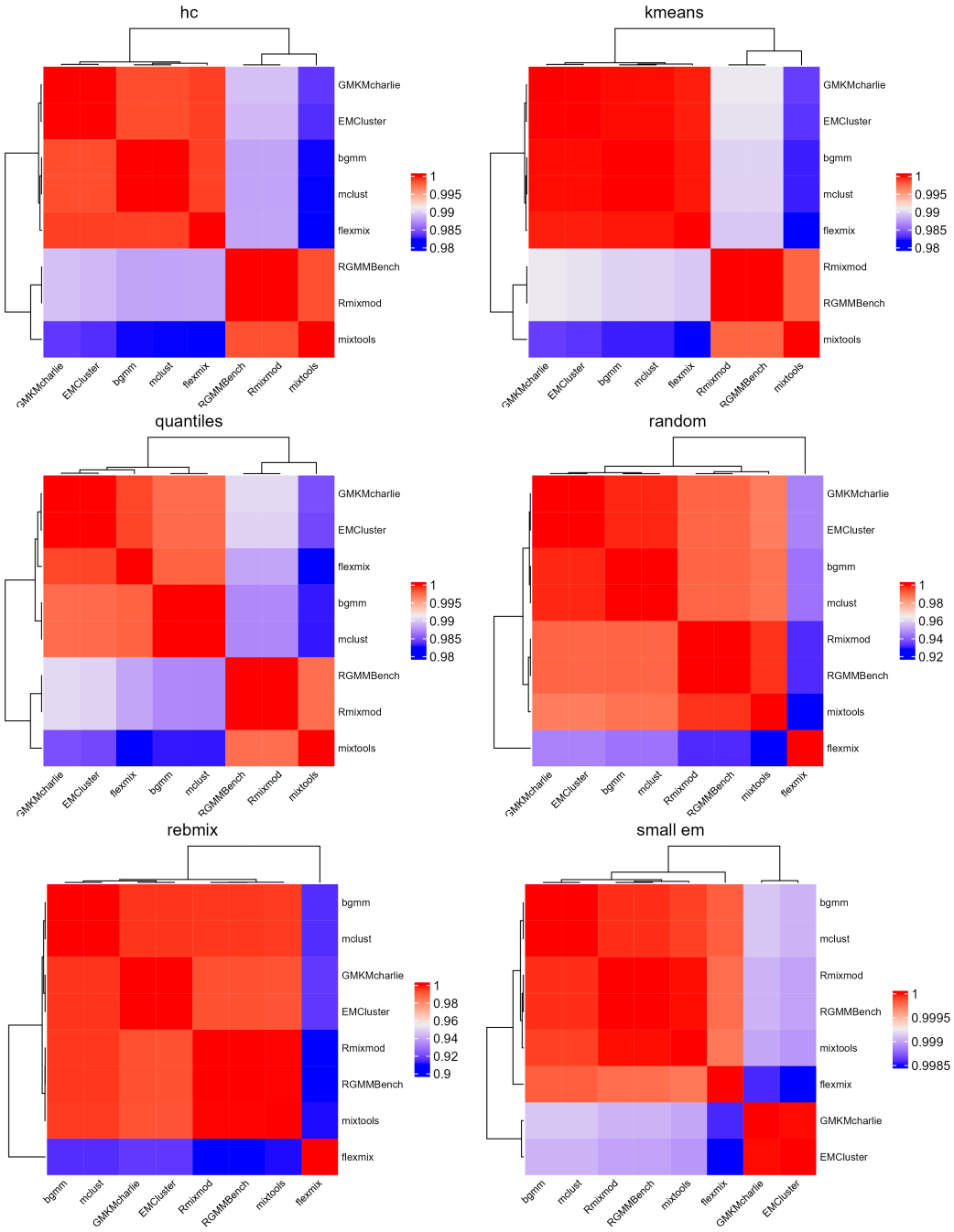


Figure 11: Correlation heatmaps of the estimated parameters extended to the four initialisation methods benchmarked, using the same configuration described in Figure 2, in the bivariate setting.

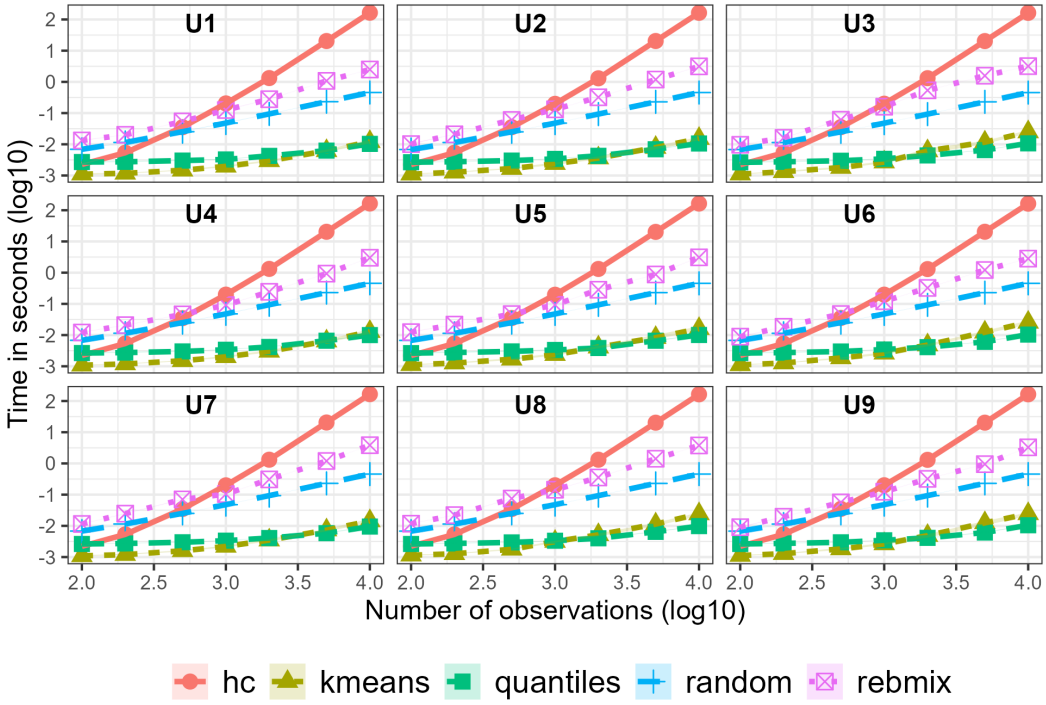


Figure 12: Distribution of the running times taken by each initialisation algorithm enumerated in Table 3, across all scenarios listed in Table 7, sorted by increasing ID number in the lexicographical order.

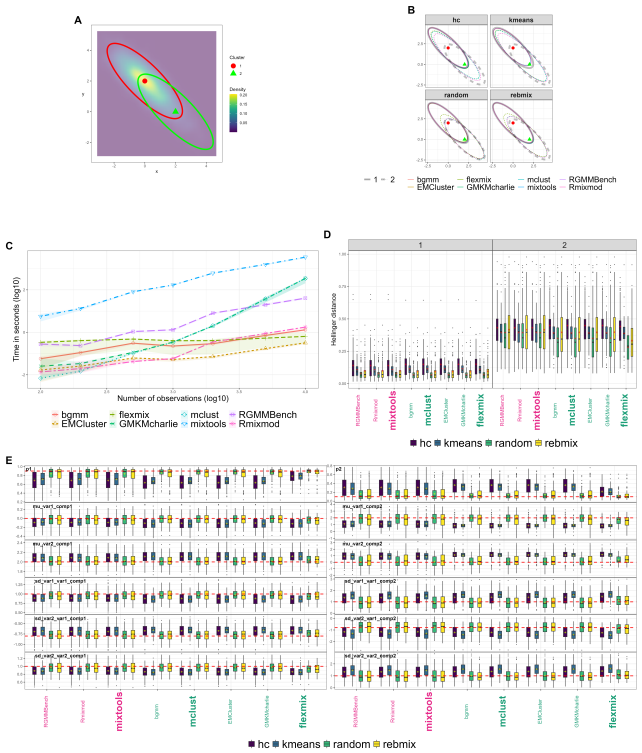


Figure 13: Results of scenario B11 in Table 12 (unbalanced, overlapping and negative correlated components), organised as such: The panel A displays the bivariate contour maps associated to the two-components multivariate Gaussian distribution corresponding to the parametrisation described by the scenario, warmer colours corresponding to regions of higher densities. The two centroids, whose coordinates are given by the mean components' elements, are represented with distinct shaped and coloured point estimates. In both Panels A and B, the ellipsoids correspond to the 95% confidence region associated to each component's distribution. To generate them, we largely inspired from the `mixtools::ellipse()` and website How to draw ellipses. To generate them, we retain for each individual parameter its mean (similar results with the median) over the $N = 100$ sampling experiments, restrained to the random initialisation method. The running times are displayed in Panel C with the *k*-means initialisation. The number of observations (x-axis) and the running time (y-axis) is in $\log(10)$ scale. The points represent median running time. The coloured bands represent the 5^{th} and 95^{th} percentiles of the running time. The distributions of the Hellinger distances (a closed form is only available for the Gaussian multivariate distribution, not the mixture) are computed for each component, each initialisation method and each package with respect to the true Gaussian distribution expected for each component. The more dissimilar are the distributions, the higher is the Hellinger distance, knowing it is normalised between 0 and 1. We represent them using boxplot representations in Panel D. In panel E we represent the boxplots associated with the distribution of the estimates, with one box per pair of package and initialisation method. As the correlation is a symmetric operator, we only represent the distribution of the lower part of the lower matrix. Each column is associated to the parameters of a component. First row represents the distribution of the estimated ratios, second and third respectively the distributions of the mean vector on the x-axis and on the y-axis, third and four the distributions of the individual variances of each feature and finally the fifth row shows the distribution of the correlation between dimension 1 and 2.

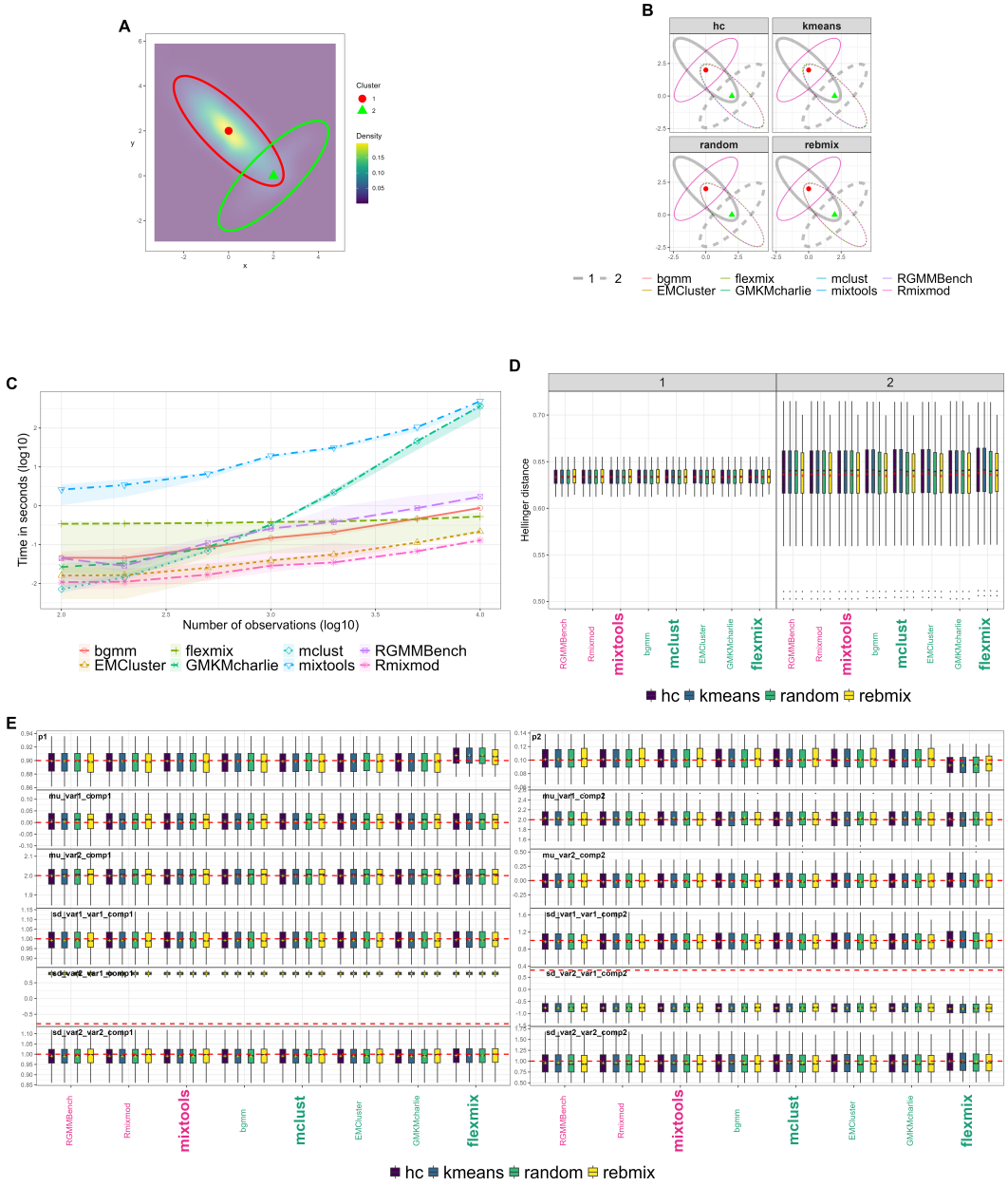


Figure 14: Results of scenario B12 in Table 12 (unbalanced, overlapping and opposite correlated components), with the same layout as Figure 13.

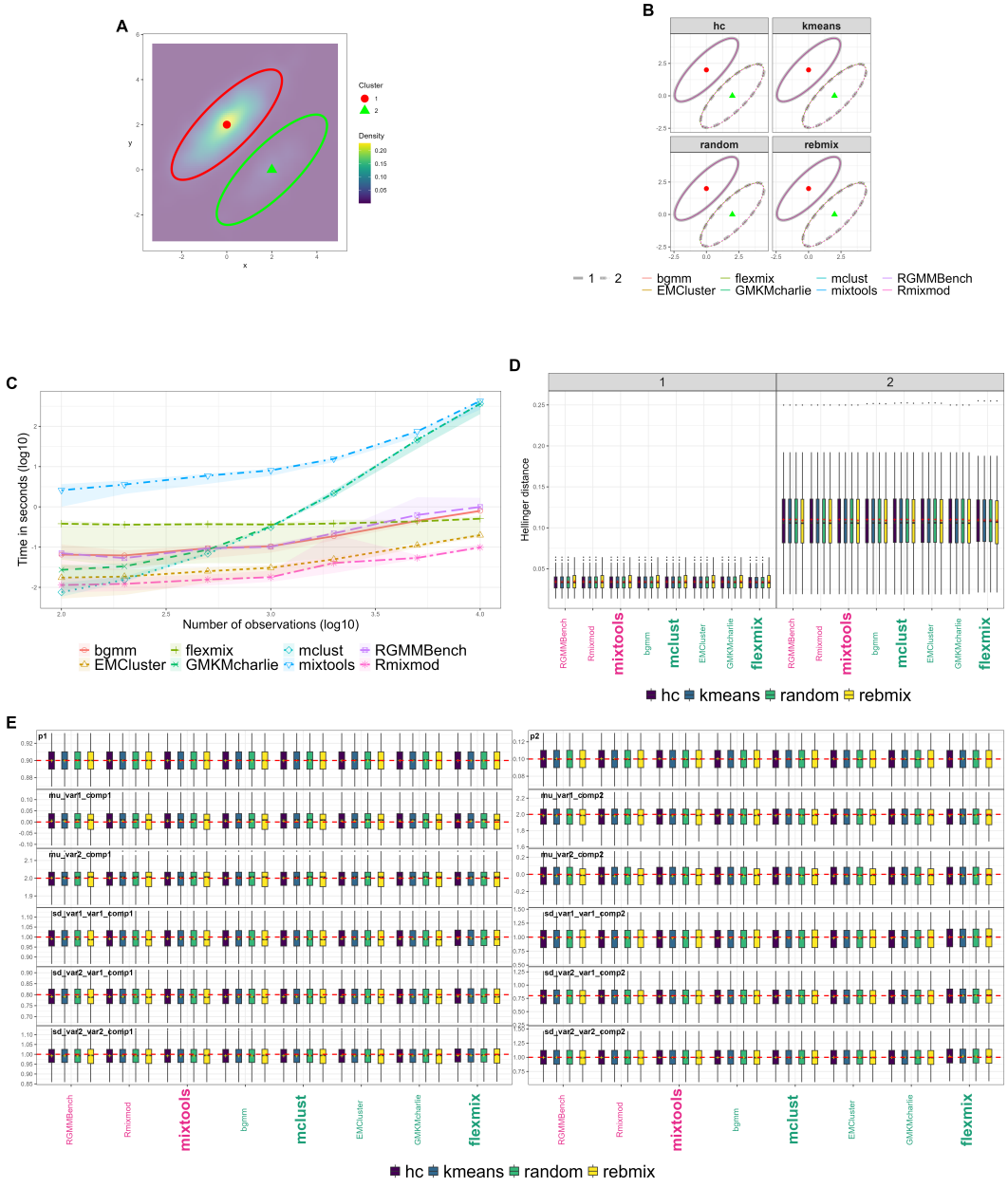


Figure 15: Results of scenario B14 in Table 12 (unbalanced, overlapping and positive correlated components), with the same layout as Figure 13.

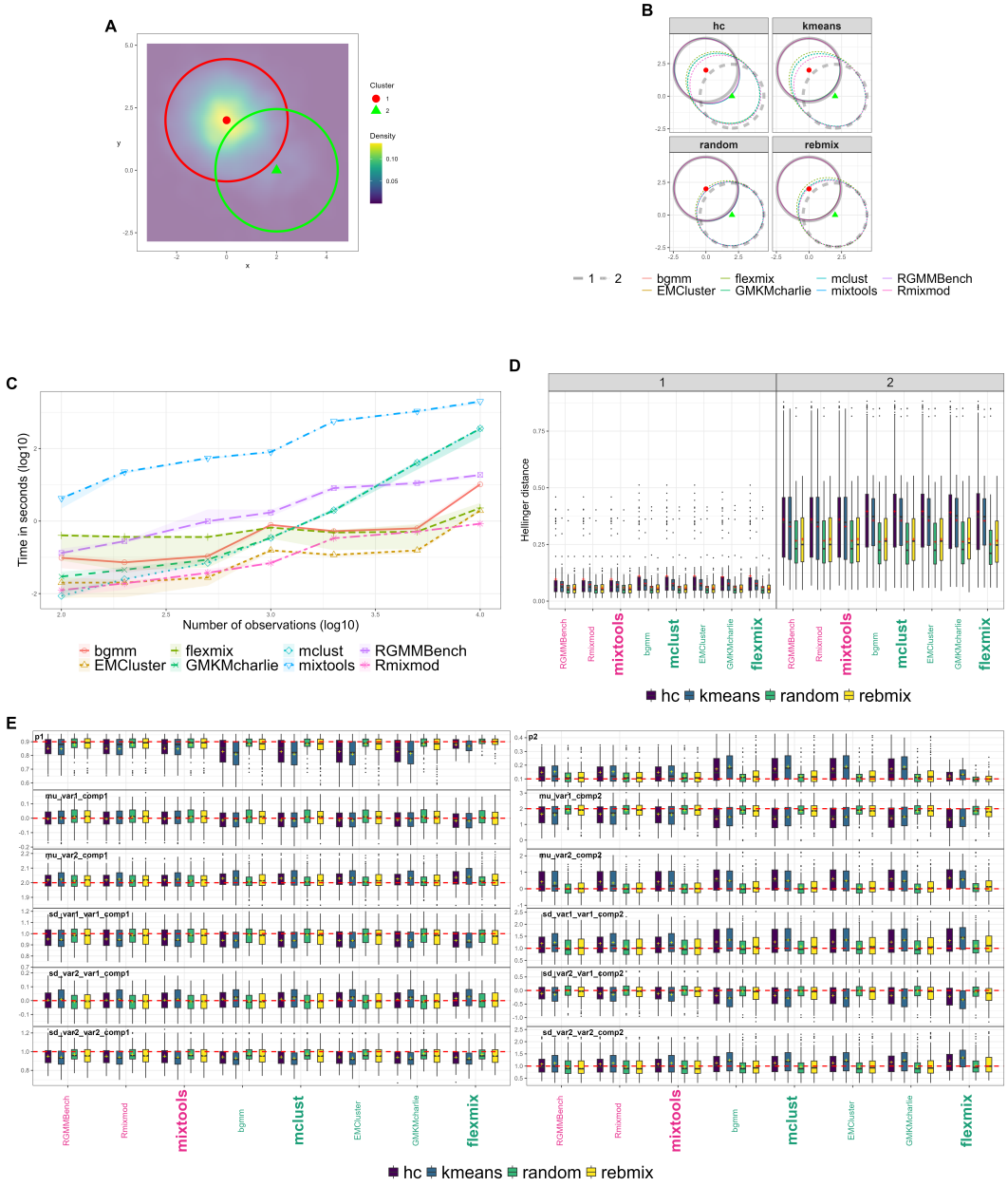


Figure 16: Results of scenario B15 in Table 12 (unbalanced, overlapping and uncorrelated components), with the same layout as Figure 13.

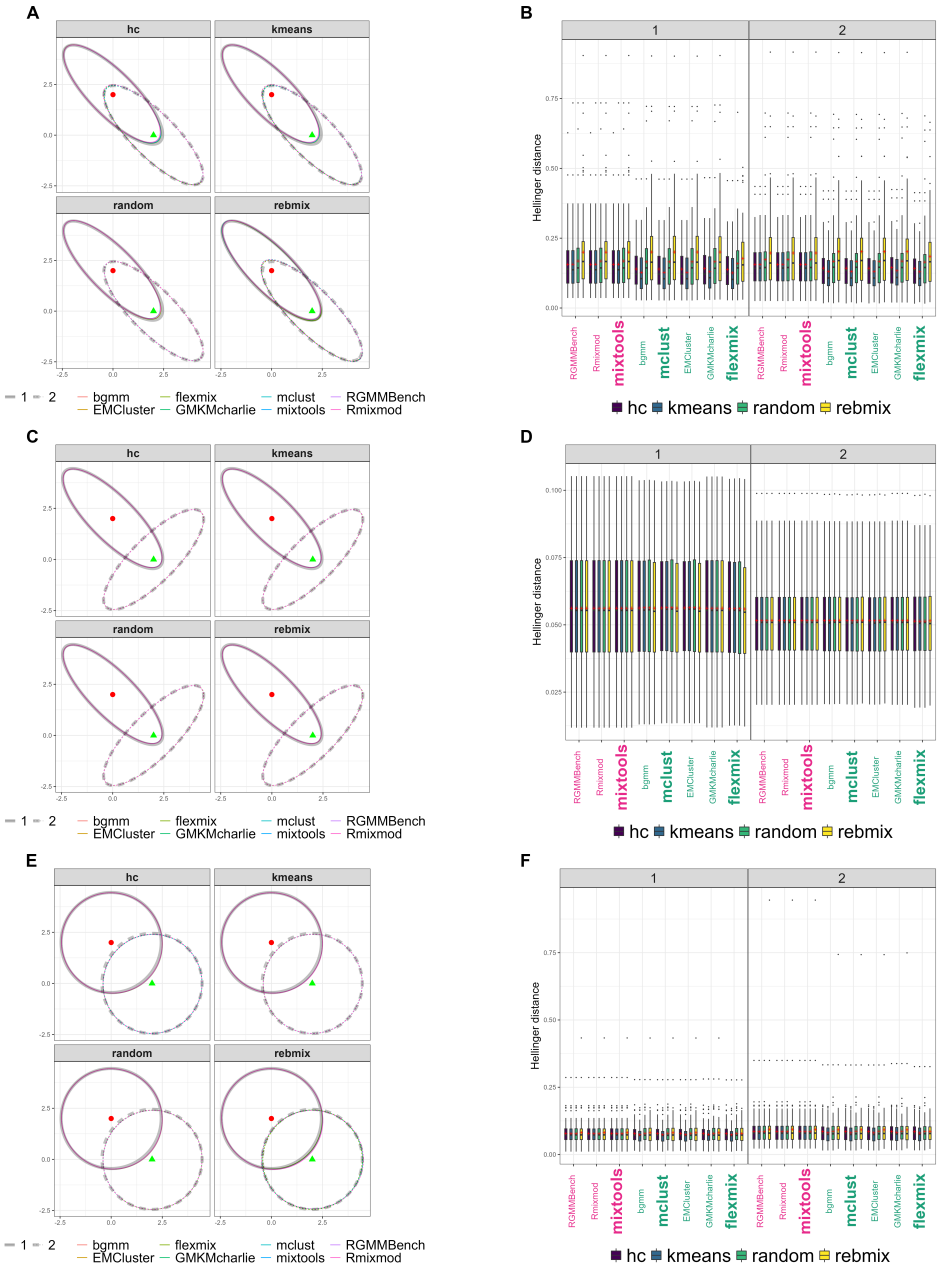


Figure 17: Benchmark summary plots of respectively scenarios B1, B2 and B5 in Table 12 featuring balanced and overlapping clusters. Summary plots of B1, B2 and B5 are represented in this order on each row, with the left column displaying the 95% confidence ellipsoidal regions associated to the mean estimated parameters across each package and the right column the distribution of the Hellinger distances.

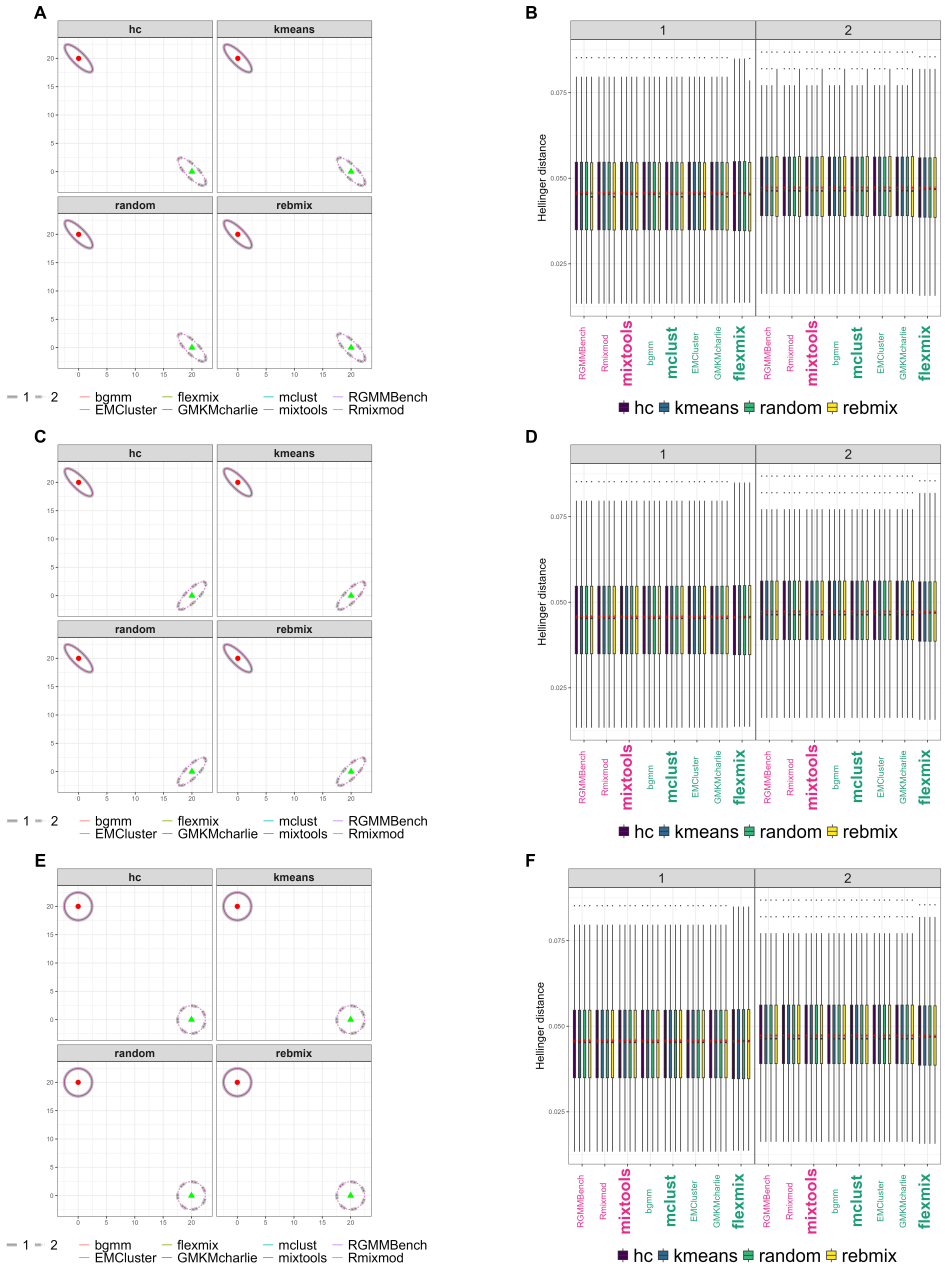


Figure 18: Benchmark summary plots of respectively scenarios B6, B7 and B10 in Table 12 featuring balanced and well-separated clusters, with the same layout as Figure 17.

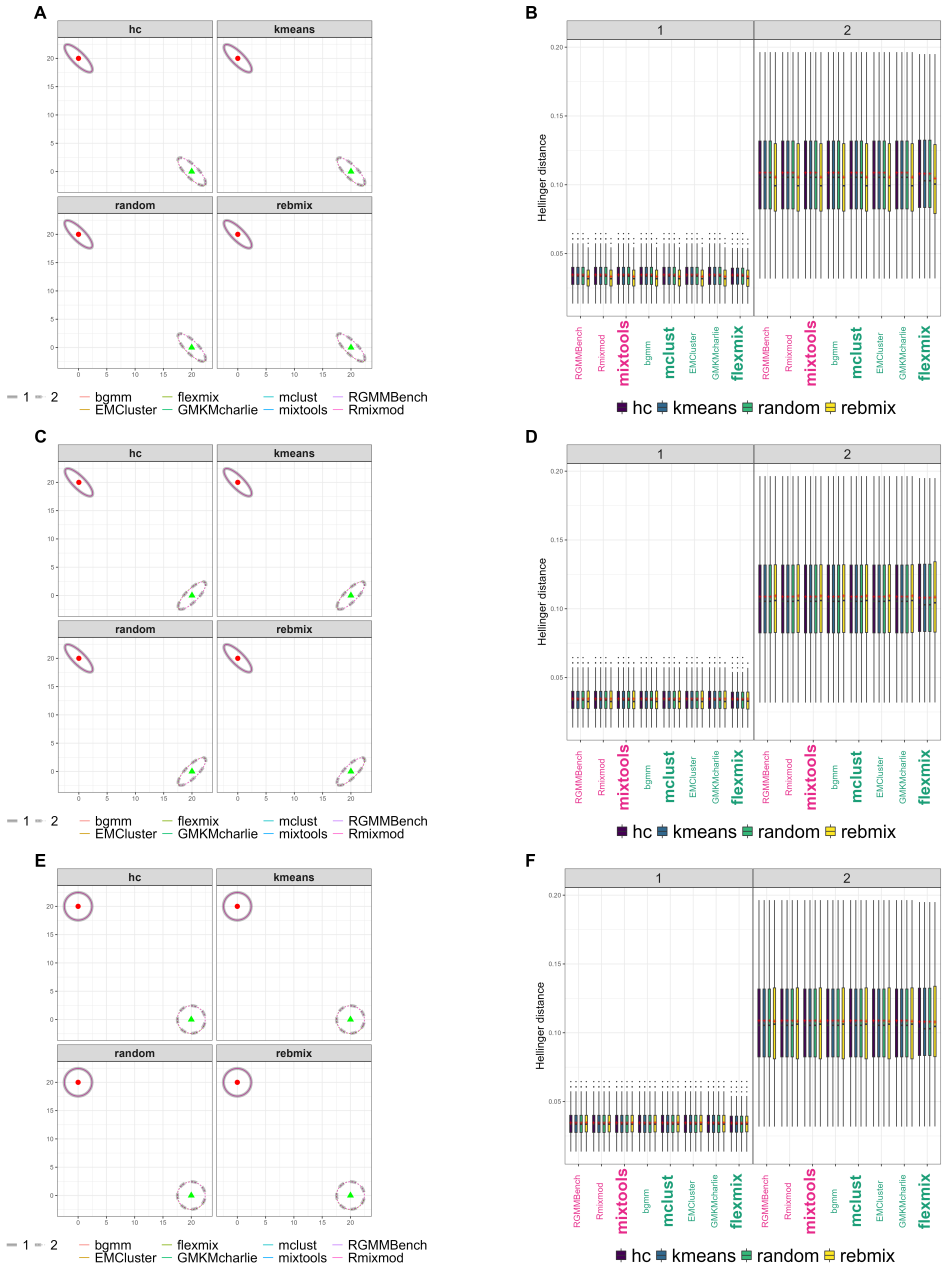


Figure 19: Benchmark summary plots of respectively scenarios B16, B17 and B20 in Table 12 featuring unbalanced and well-separated clusters, with the same layout as Figure 17.

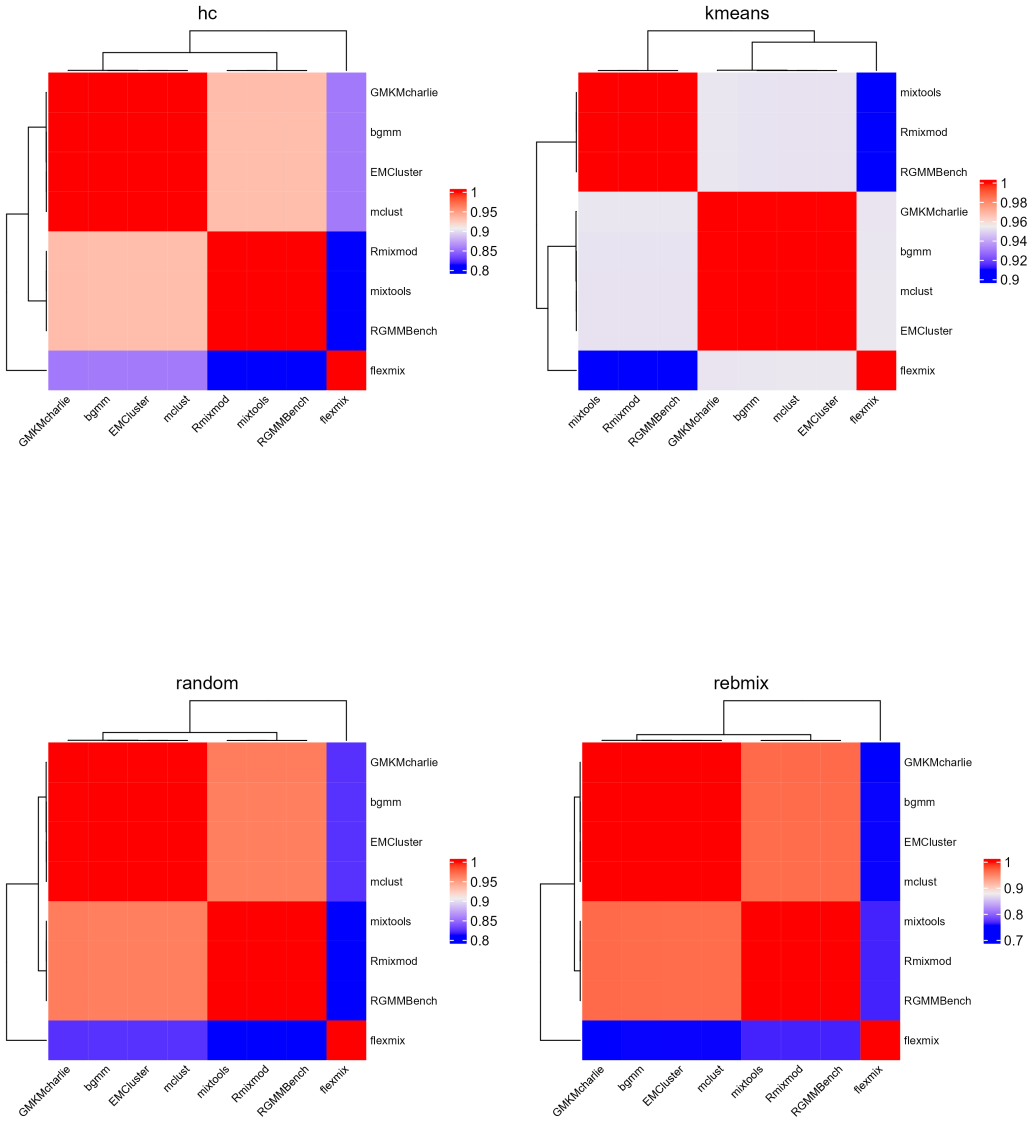


Figure 20: Correlation heatmaps of the estimated parameters in the bivariate setting extended to the four initialisation methods benchmarked, with the most discriminating scenario B11, using the same process described in Figure 2.