Supplementary Notes on Gaussian Mixture Models in R

Bastien Chassagnol* Antoine Bichat† Cheïma Boudjeniba‡ Pierre-Henri Wuillemin§
Mickaël Guedj¶ David Gohel‖ Gregory Nuel** Etienne Becht††

For the sake of readibility, we display in Table 1 the general configuration used to run all the benchmarks tested.

Table 1: 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

Furthermore, the code snippets, data, and figure subfolders required for replicating the figures documented in this supplementary material are readily accessible through the following public GitHub repository: GMM_appendix.

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 (Walsh 1975).

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 (1)):

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

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 (2)[1]:

*LPSM, Sorbonne Université, bastien\protect_chassagnol@laposte.net
†Les Laboratoires Servier, IRIS
‡Les Laboratoires Servier, IDRS
§LIP6, Sorbonne Université
¶Les Laboratoires Servier, IRIS
‖ArData
**LPSM, Sorbonne Université
††Les Laboratoires Servier, IRIS, etienne.becht@polytechnique.edu

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

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \zeta_j} = \sum_{i=1}^{n} \eta_i(j) \frac{\partial \log(f_{\zeta_j}(X_i|S_i = j))}{\partial \zeta_j} \tag{2}$$

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 (3)):

$$Q(\theta|\hat{\theta}_{q-1}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_i(j) \left(\log(p_j) - \log(\sigma_j) - \frac{(X_i - \mu_j)^2}{2\sigma_j^2} \right) + K \tag{3}$$

and Equation (4) for the multivariate GMM:

$$Q(\theta|\hat{\theta}_{q-1}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \eta_i(j) \left[\log(p_j) - \frac{1}{2} \left(\log(\det(\boldsymbol{\Sigma}_j)) + (x_i - \boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1} (x_i - \boldsymbol{\mu}_j) \right) \right] + K \tag{4}$$

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 (5) - (6)):

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \mu_j} = 0 \Leftrightarrow \mu_j = \frac{\sum_{i=1}^{n} \eta_i(j) X_i}{\sum_{i=1}^{n} \eta_i(j)} \tag{5}$$

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \sigma_j} = 0 \Leftrightarrow \sigma_j^2 = \frac{\sum_{i=1}^{n} \eta_i(j)(x_i - \mu_j)^2}{\sum_{i=1}^{n} \eta_i(j)} \tag{6}$$

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, $\boldsymbol{A}^\top = \boldsymbol{A}$

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* (Petersen and Pedersen 2008).

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 (7) - (8)):

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \boldsymbol{\mu}_j} = \sum_{i=1}^{n} \eta_i(j)\boldsymbol{\Sigma}_j^{-1}(x_i - \boldsymbol{\mu}_j) = 0 \Leftrightarrow \mu_j = \frac{\sum_{i=1}^{n} \eta_i(j)\boldsymbol{x}_i}{\sum_{i=1}^{n} \eta_i(j)} \tag{7}$$

$$\frac{\partial Q(\theta|\hat{\theta}_{q-1})}{\partial \boldsymbol{\Sigma}_j^{-1}} = \frac{1}{2}\sum_{i=1}^{n} \eta_i(j)\left[\boldsymbol{\Sigma}_j - (x_i - \boldsymbol{\mu}_j)(x_i - \boldsymbol{\mu}_j)^\top\right] = 0 \Leftrightarrow \boldsymbol{\Sigma}_j = \frac{\sum_{i=1}^{n} \eta_i(j)(x_i - \boldsymbol{\mu}_j)(x_i - \boldsymbol{\mu}_j)^\top}{\sum_{i=1}^{n} \eta_i(j)} \tag{8}$$

Explicitly optimising the equations ((3)-(4)) yield the following MLE parameters in both the univariate and multivariate settings (Table 2), as detailed in (Leytham 1984; Redner and Walker 1984):

Table 2: 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[2]. 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 (9):

$$\begin{aligned} \ell(\theta|x) &= \log(\sum_{j=1}^{k} p_j f\zeta_j(x)) \\ &= \log\left(\exp\left[\log(p_j) + \log(f\zeta_j(x))\right]\right) \end{aligned} \tag{9}$$

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 (10):

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

$$\begin{aligned}\log\left(\sum_{j=1}^{k} e^{a_j}\right) &= \log\left(\exp(a'_j) \times \left[1 + \sum_{j \neq j'} \exp\left(\frac{a_j}{a_{j'}}\right)\right]\right) \\ &= a_{j'} + \text{log1p}\left(\sum_{j \neq j'} \exp\left(\frac{a_j}{a_{j'}}\right)\right), \text{ with } j' = \arg\max_{\forall j \in \{1,\ldots,k\}} (e^{a_j})\end{aligned} \tag{10}$$

with `log1p` the R function dedicated for this Taylor's development. The posterior probabilities are then given by Equation (11):

$$\log\left(\mathbb{P}_\theta(S = j|X = x)\right) = \log(p_j) + \log(f_{\zeta_j}) - \ell(\theta|x) \tag{11}$$

- 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 (12)):

$$\boldsymbol{\Sigma}_j = \lambda_j \boldsymbol{Q}_j \boldsymbol{D}_j \boldsymbol{Q}_j^\top \tag{12}$$

with $\lambda_j = \det(\boldsymbol{\Sigma}_j)^{\frac{1}{D}}$ a scalar proportional to the total volume of the ellipsoid (or area in bi-dimensional setting), $\boldsymbol{D}_j$ a diagonal matrix storing the eigenvalues normalised such that $|\boldsymbol{D}_j| = 1$[3] 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:

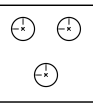
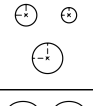
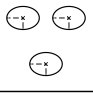
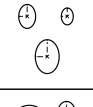
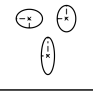
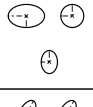
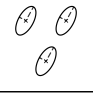
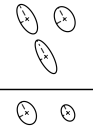
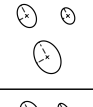
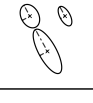
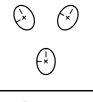
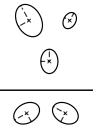
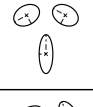
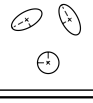
- two models with the *spherical family*, for which only λ_j is used to control the *isotropic* (same radius in any dimension) volume of each component of the corresponding distribution structure
- four models with the *diagonal family*, using λ_j with possibly distinct diagonal elements and $\boldsymbol{D}_j$ 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 (3):

- 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

[3] Langrognet et al. (2021) enforces an additional but, in our opinion, superfluous constraint that the eigen values are sorted by decreasing order

Table 3: The 14 canonical parametrisations of the within-group covariance matrix $\boldsymbol{\Sigma_j}$ 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	$[\lambda Q D Q^\top]$	Ellipsoidal	CF	$\alpha + \beta$	
EVE	$[\lambda Q D_j Q^\top]$	Ellipsoidal	IP	$\alpha + \beta$	
VEE	$[\lambda_j Q D Q^\top]$	Ellipsoidal	IP	$\alpha + \beta + (k-1)(d-1)$	
VVE	$[\lambda_j Q D_j Q^\top]$	Ellipsoidal	IP	$\alpha + \beta + d(k-1)$	
EEV	$[\lambda Q_j D Q_j^\top]$	Ellipsoidal	CF	$\alpha + k\beta - d(k-1)$	
VEV	$[\lambda_j Q_j D Q_j^\top]$	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	$\alpha + k\beta$	

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).

- 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 (12) 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** (Scrucca et al. 2016) and **Rmixmod** (Langrognet et al. 2021) vignettes for a general introduction to GMMs and to (Banfield and Raftery 1993; Celeux and Govaert 1992; Browne and McNicholas 2014) for the closed formulas of the models.

Parameters estimation in a high-dimensional context

However, while parsimonious representations can largely reduce the computational burden, none of them in the general family is able to handle degenerate cases where the number of features, D, exceeds the number of observations n. Likewise situations, when the number of features is consequent, are referred to as high-dimensional, raising the well-known issue of the "curse of the dimensionality". Two distinct approaches have been developed in the literature to handle these degenerate cases:

- The most naive approach aims to eliminate the least informative variables by applying a strong Lasso-type penalty on the parameters to be estimated. We only came across such an approach twice among the reviewed R packages, in the specific context of regressions of mixtures (see**RobMixReg** and **fmerPack** packages).
- The second category includes a larger diversity of methods, all inspired from the *factor analysis* approach whose paradigm is to consider that all the D features used to describe the observations can be spanned in a smaller subspace without lose of information. Precisely, the factor analysis theory describes the variability among observed and correlated variables by a substantial lower number of unobserved variables called *factors* or *latent variables*. In practice, for a given component j, the diagonal matrix storing the eigenvalues is decomposed into two-blocks. The first upper-right diagonal block, assumed generally of dimension $d_j \ll D$, stores the largest d_j eigenvalues and model the variance of the actual data of component j while the lower-left diagonal block, of dimension $D - d_j$, stores an unique parameter that can be interpreted as the variance of the residual error terms, constrained to be strictly inferior to the lowest variability of the informative variables. The dimension d_j can be considered as the intrinsic dimension of the latent subspace of cluster j spanned by the first d_j eigenvectors of $\boldsymbol{Q}_j$[4].

[4]Starting from eigen-decomposition described in (Equation (12)), this approach is equivalent to consider only the d_j largest eigenvalues resulting from the decomposition and sets the others to null.

When the sub dimension d_j is known, a closed version is generally available for the M-step of the EM algorithm, however d_j is itself an hyperparameter to estimate. Though, (Bouveyron, Celeux, and Girard 2011) has shown that a classical Cattell's scree-test could be used to asymptotically estimate the intrinsic dimension of each cluster. Compared to the previous approach, this method has a strong theoretical background and strong impact on the running times performance.

Taking a concrete use case from the help documentation of the package **HDclassif**, it enabled to cluster a dataset of 10 classes with 130 observations overall and described in a 1024-dimensional space (consider the famous machine-learning digit recognition problem). Variants of these approaches have been developed in the following packages: **HDclassif**, **fabMix**, **EMMIXmfa** and **pgmm**. We refer the interested reader to the educational vignette of **HDclassif**: HDclassif and papers (Paul David McNicholas and Murphy 2008; P. D. McNicholas et al. 2010; Paul D. McNicholas and Murphy 2010).

Historically, the first mention of a probabilistic framework with an application to dimension reduction in the context of finite mixture models goes back to Tipping and Bishop (1999), based on principal component analysis. G. J. McLachlan, Peel, and Bean (2003) and Mclachlan and Peel (2000) extend this original model by postulating that the distribution of the data within any latent class could be described using the tools of the factor analysis field[5] Finally, building on the parsimonious parametrisations already theorised for GMMs (see previous section) , Paul David McNicholas and Murphy (2008), P. D. McNicholas et al. (2010) and Bouveyron, Girard, and SCHMID (2007) proposed a variety of constraints, but this time directly defined on the projected subspace. Since all methods based on factor analysis provide a transition matrix, using the two or three most informative eigen values and their associated eigen vectors in order to project the dataset on a smaller subspace provides a simple visualisation tool for representing high dimensional datasets. However, this method may is not suitable for unravelling the clustering structure. Instead, *the GMMDR method*, first proposed by Scrucca (2010) and implemented in the `MclustDR` function, from **mclust** package, aims at recovering the subspace that best captures the underlying latent clustering structure (we notably expect invariance of the global overlap in the sampling space and the corresponding projected subspace). More precisely, the main objective of the GMMDR technique is to infer the global *change-of-basis matrix* $\boldsymbol{Q}$ that minimises the differences in the a posteriori probabilities of assigning each observation i to a given cluster s_i, knowing the value of the vector of observed covariates $\boldsymbol{x}_i$. Namely, we are looking for the orientation matrix $\boldsymbol{Q}$ that maximally ensures the following objective (Eq. (13)):

$$\hat{\boldsymbol{Q}} = \arg\max_{\boldsymbol{Q}} \left(\mathbb{P}_\theta(S_i = j|\boldsymbol{X} = \boldsymbol{x}_i) = \mathbb{P}_\theta(S_i = j|\boldsymbol{X}\boldsymbol{Q})\right) \text{ such that } S \perp \boldsymbol{X}|\boldsymbol{X}\boldsymbol{Q} \tag{13}$$

This procedure itself derives from the *sliced inverse regression* algorithm (K.-C. Li 1991), but instead of conditioning on the known response variable, GMMDR conditions on the estimated MAP cluster assignments. Since the solution returned by the following optimization problem is not unique, we generally constrain the projection matrix to be orthonormal (any of the vectors forming the basis are pairwise orthogonal, and individually of norm 1).

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 (14):

$$\text{GIC}(\theta) = \underbrace{p(\theta)}_{\text{penalty term}} - \underbrace{2\ell(X|\theta)}_{\text{log-likelihood of the model}} \tag{14}$$

[5]Although principal component analysis and factor analysis are closely related, we can differentiate both approaches by their differing objective: while PCA seeks to capture the overall variability of the dataset, factor analysis focuses on describing the intra-variability between covariates. In practice, the differences between the two approaches are minor, we can notably show that the output of PCA is one of the solutions suggested by standard factor analysis.

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) (Schwarz 1978) is a *minimax-rate optimal* (score that minimises the risk in the worst case) but inconsistent metric (Yang 2005) , 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 Fonseca (2008), Celeux, Fruewirth-Schnatter, and Robert (2018).

In the second category, the most commonly implemented is the *ICL (integrated complete-data likelihood)*, a BIC criterion with an additional entropy penalty (G. McLachlan and Peel 2000). 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 (Bacci, Pandolfi, and Pennoni 2012). The several metrics implemented by the reviewed packages are listed in Table 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 (G. McLachlan and Peel 2000). To counterbalance it, bootstrap inference (G. McLachlan and Peel 2000) 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 (G. McLachlan and Peel 2000). Hence, most authors recommend to use bootstrap methods for the generation of confidence intervals, as suggested in (Efron and Tibshirani 1993; Basford et al. 1997).

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 Jaki et al. (2018), 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: $\hat{\theta}_{1:N} = (\hat{\theta}^1, \dots, \hat{\theta}^N)$. 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 (15):

$$\mathbb{E}[\hat{\theta}_t] \pm \frac{1}{\sqrt{n}} z_{1-\frac{\alpha}{2}} \sqrt{\text{var}(\hat{\theta}_t}, \quad \forall t \in \{1, \ldots, 3k\} \tag{15}$$

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 (16):

$$[\mathcal{I}_{\text{exp}}(\theta)]_{1 \leq i \leq 3k, 1 \leq j \leq 3k} = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i \partial\theta_j} \ell(\theta|X)\right] \tag{16}$$

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 Louis (1982) and for the multivariate case in Oakes (1999). Yet, it has to be noted that the expected FIM generally outperforms the observed FIM in estimating the covariance matrix of the MLE (X. Cao and Spall 2012).

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, L. Meng (2016) and Delattre and Kuhn (2019) proposed an accelerated algorithm requiring only computation of first order derivatives. A similar alternative is implemented in the **mixsmsn** package (Prates, Lachos, and Cabral 2021): `mixsmsn::im.smsn`, in which the Hessian matrix is approximated by the cross-product of the gradient of the log-likelihood Equation (17):

$$\mathcal{I}_{\text{obs}}(\theta) \approx -\frac{\partial \log(\ell(\theta|X))}{\partial\theta} \frac{\partial \log(\ell(\theta|X))}{\partial\theta}^T \tag{17}$$

according to an idea developed in paper Basford et al. (1997). 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* Celeux and Govaert (1992).

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 (18):

$$\text{OVL}(X) = 1 - \int_{\mathbb{R}} \max_j (p_j \varphi_{\zeta_j}(x)) dx \tag{18}$$

The 1 in Equation (18) 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) (Marandon et al. 2022).

Equation (18) simplifies for a two component mixture distribution to Equation (19):

$$\text{OVL}(X) = \int_{\mathbb{R}} \min\left(p_1 \varphi_{\zeta_1}(x), p_2 \varphi_{\zeta_2} dx(x)\right) \tag{19}$$

From a probabilistic point of view, we can rewrite Equation (19) 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 (20):

$$\begin{aligned}\text{OVL}(1,2) &= \text{OVL}(1|2) + \text{OVL}(2|1) \\ &= \mathbb{P}\left(p_1\varphi(X,\mu_1,\sigma_1) \leq p_2\varphi(X,\mu_2,\sigma_2)\right) + \mathbb{P}\left(p_2\varphi(X,\mu_2,\sigma_2) \leq p_1\varphi(X,\mu_1,\sigma_1)\right) \\ &= \int_{\mathbb{R}} p_1\varphi_{\zeta_1}(x)1_{p_1\varphi_{\zeta_1} \leq p_2\varphi_{\zeta_2}}dx + \int_{\mathbb{R}} p_2\varphi_{\zeta_2}(x)1_{p_2\varphi_{\zeta_2} \leq p_1\varphi_{\zeta_1}}dx\end{aligned} \tag{20}$$

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 1. 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 (21), all observations issued from class 2 are wrongly assigned to class 1.

$$\eta_i(j) := \mathbb{P}_\theta(S_i = j|X_i = x_i) \tag{21}$$

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 1.

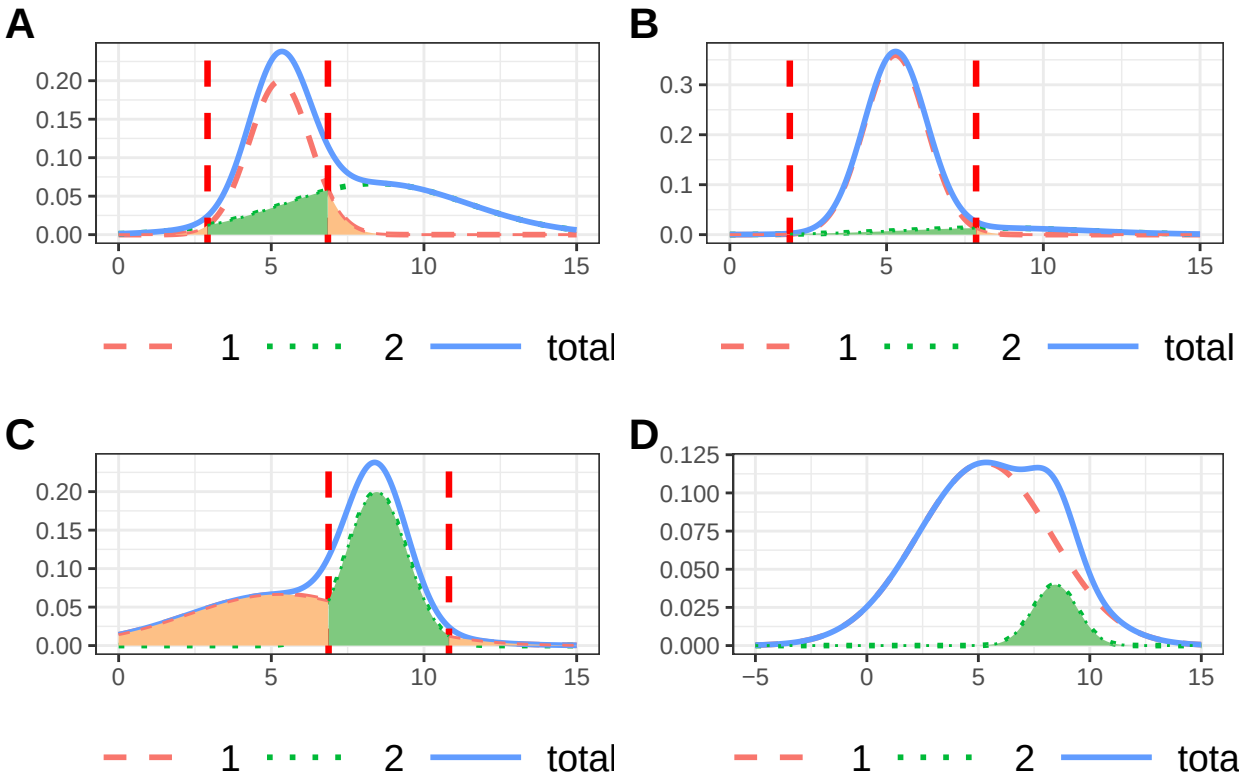


Figure 1: Illustration of the overlaps between a two-components GMM. Density function of component 1 is given by the red line, its of component 2 by the green line, and total density function $f_\theta(X)$ is represented in blue. The total overlap is given by the sum of the green and red areas.

There are two intersection points, x_1 and x_2 , with $\mu_1 < \mu_2$ when solving equation Equation (22):

$$p_1\varphi(x,\mu_1,\sigma_1) = p_2\varphi(x,\mu_2,\sigma_2) \tag{22}$$

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 (23):

$$(x_1, x_2) = \left(\frac{\sigma_1^2\mu_2 - \sigma_2^2\mu_1 \pm \sigma_1\sigma_2\sqrt{(\mu_1-\mu_2)^2 + 2(\sigma_2^2-\sigma_1^2)\left[\log(\frac{p1}{p2}) + \log(\frac{\sigma_2}{\sigma_1})\right]}}{\sigma_1^2 - \sigma_2^2} \right) \tag{23}$$

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 1:

- 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 (24):

$$\text{OVL}(1,2) = p_1\left(\Phi(x_2, \mu_1, \sigma_1) + 1 - \Phi(x_1, \mu_1, \sigma_1)\right) + p_2\left(\Phi(x_1, \mu_2, \sigma_2) - \Phi(x_2, \mu_2, \sigma_2)\right) \tag{24}$$

- 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 (25):

$$\text{OVL}(1,2) = p_2\left(\Phi(x_1, \mu_2, \sigma_2) + 1 - \Phi(x_2, \mu_2, \sigma_2)\right) + p_1\left(\Phi(x_2, \mu_1, \sigma_1) - \Phi(x_1, \mu_1, \sigma_1)\right) \tag{25}$$

An interesting result is obtained with the homoscedascity and balanced classes' assumptions of 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 bi-dimensional 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 (Alberich-Carramiñana, Elizalde, and Thomas 2017), 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 (Maitra and Melnykov 2010; Pastore and Calcagnì 2019; Nowakowska, Koronacki, and Lipovetsky 2014).

Appendix B: Extensions of the EM algorithm to overcome its limitations

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 (Biernacki, Celeux, and Govaert 2000) maximises a function where each observation was assigned to the maximum a posteriori (MAP) estimate Equation (21). 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 (G. McLachlan and Peel 2000). EM*, referred in Kurban, Jenne, and Dalkilic (2017) 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 Celeux and Govaert (1992). 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 Celeux, Chrétien, and Forbes (2012), 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** (Bobb and Varadhan 2021).

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* (Murphy 2012).

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 (Figueiredo and Jain 2002) , 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 (Figueiredo and Jain 2002).

The Expectation/Conditional Maximisation (ECM) (MENG and Rubin 1993) 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 (X.-L. Meng and Van Dyk 1997), 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.

A small simulation to evaluate the impact of outliers

Classical methods used for the parameters' estimation, especially the maximum likelihood estimation (MLE), are sensitive to the presence of outliers. A naive solution consists in assigning null weights to observations suspected to be outliers, so that they do not contribute [6]. Trimming aberrant observations from the distribution is justified theoretically by the principle of the *spurious outlier model* (Gallegos and Ritter 2005). However, this method is quite stringent, requiring human expertise or the use of general outlier detection tools not necessarily adapted to GMM estimation.

Two general approaches for dealing with outliers with a well-defined theoretical background are the *outliers mixture modelling* and the *trimming approach*. *Outliers mixture modelling* integrate an additional component accounting for the outliers in the distribution. Notably, the **mclust** (Fraley, Raftery, and Scrucca 2022) and **otrimle** (Coretto and Hennig 2021) packages use an improper uniform distribution to model the distribution of outliers. Unlike **mclust**, the **otrimle** package does not require the user to set in advance the proportion of outliers in the mixture (Coretto and Hennig 2016). As opposed, in the *trimming approach*, outliers are first removed before the complete estimation of parameters. Such methods are implemented in **tclust** (Iscar, Escudero, and Fritz 2022) and **oclust** (Clark and McNicholas 2019) packages.

tclust (Iscar, Escudero, and Fritz 2022) uses a robust constrained clustering method, where the user has to set an upper threshold to the ratio between the highest and the lowest variability among all components and a trimming ratio α. It extends the work of García-Escudero et al. (2008), with released constraints on the Gaussian distribution. First, the maximal degree of affinity, defined in Equation (26):

$$D(x_i|\theta) = \max_j \left(p_j \varphi_{\zeta_j}(x_i)\right) \tag{26}$$

is computed for each observation x_i, and corresponds for each point to the maximum probability to observe it in the distribution, given parameter θ. Then, α observations the least likely to be observed are trimmed for the estimation of the parameters. When we reach convergence of the estimated parameter and there is no change in the outliers identification from one iteration to another, the iterative algorithm stops. The use of constraints is an additional feature that avoids building over-dispersed or unbalanced clusters, the highest constraint of a ratio of 1 yielding clusters with equal sizes. However, the identification of an observation as aberrant is highly dependant on the variability constraint and the determination of these two hyperparameters is complex and highly dependant on the shape of the distribution. Additionally, a CEM algorithm is used to retrieve the parameters and the proportion of outliers, for which the MLE, in contrast to the EM algorithm, is not asymptotically consistent nor efficient.

Unlike **tclust**, **oclust** (Clark and McNicholas 2019) both retrieves the proportion of outliers and identifies them. To do so, it compares the complete log-likelihood of the mixture $\ell(\theta|X)$ with its value removing one observation $\ell(\theta|X \setminus X_i)$, for all observations. Observations are iteratively removed, based on the assumption that the Kullback-Leibler divergence between the original log-likelihood and the trimmed log-likelihood $\mathrm{KL}\left(\ell(\theta|X)||\ell(\theta|X \setminus X_i)\right)$ follows a Beta distribution. At each step, the observation that maximises the Kullback-Leibler divergence at a statistically significant threshold is removed. The algorithm stops trimming outliers, when this measure is not anymore statistically significant. However, the assumption of a Beta distribution only holds asymptotically and with non-overlapping clusters.

To integrate the impact of outliers in the estimation, we simulated a two-components GMM with well-separated and balanced clusters. The outliers distribution, corresponding to the additional noise component, was retrieved by randomly selecting *prop.outliers* points out of the total number of observations and drew their values from

[6] The use of weighted distributions has more general applications. It can be used to deal with a component distribution that does not fit exactly a Gaussian shape. For instance, to deal with heavy tail distributions, more weight can be given to central components and less weight to the tails.

an uniform distribution bounded by an interval five times as big as the 0.05 and 0.95 quantiles of $f_\theta(X)$. All estimates were obtained comparing the five reviewed initialisation methods, except with **otrimle** which has its own hierarchical clustering initialisation method.

The slowest package is **otrimle**, most of the time being taken by the initialisation step where proportion and identification of the outliers is performed. Running times of the other packages are generally not impacted by the presence of outliers.

Most of the reviewed packages, except the **bgmm** package, are not impacted by the choice of initialisation method. Additionally, the proportions are rather correctly estimated (related to the choice of an uniform distribution to model outliers), but the reviewed packages tend to overestimate the true variability of each component, with the worst results obtained with **rebmix** initialisation. **bgmm** sets apart from the others by its reduced bias on the means and standard deviations estimated, a feature left undocumented. However, increasing the number of outliers (Figure 2, panel C) lead also to biased estimations for **bgmm**, while **otrimle**, a dedicated package, is still able to correctly estimate the individual parameters of the components' distributions with a high proportion of outliers. Yet, analysing the code used to implement the **bgmm** reveals that there is no dedicated feature to remove outliers but rather a specific method used to deal with numerical underflow that artificially increases the probability of observing outlying distributions (EM-implementation differences across reviewed packages).

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

General workflow

Table 4 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 4: 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 homoscedascity or equibalanced proportions). Hence, **epigenomix** (Klein and Schaefer 2022) , **EMMIXgene** (Andrew Thomas Jones 2020) , **pcensmix** (Fallah and Hinde 2017) , **mixAK** (Komárek 2022) (homoscedastic components), **mixture** (Pocuca, Browne, and McNicholas 2022) (multi-dimension only), **AdaptGauss** (Thrun, Hansen-Goos, and Ultsch 2020) and **MMDvariance** (X. Li et al. 2018) 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** (Rauschenberger 2022) 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** (Papastamoulis 2020) , **EMMIXmfa** and **pgmm** (Paul D. McNicholas et al. 2022) packages. The **sensory** (Franczak, Browne, and McNicholas 2016) 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** (Dean, Raftery, and Scrucca 2020) 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** (Macdonald and Juan Du 2018).

- 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** Mohammadi (2021)], **bpgmm** (Lu, Li, and Love 2022), **DPP** (Avila, May, and Ross-Ibarra 2018) , **dppmix** (Xu et al. 2020), **BayesCR** (Garay et al. 2017) and **Melissa** (Kapourani 2022). **polySegratioMM** (Baker 2018) 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** (Houseman et al. 2017) which implements a recursive algorithm, and **SAGMM** (Andrew T. Jones and Nguyen 2019) 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** (Shokoohi 2022) , **mixreg** (Turner 2021) or **fmerPack** (Y. Li and Chen 2021) , an extension of the **flexmix** package with an additional feature selection using the lasso method. **nlsmsn** (Prates, Lachos, and Garay 2021) implements regression of skewed Gaussian mixtures, but in unidimensional space only. **RobMixReg** (S. Cao, Chang, and Zhang 2020) 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** (Kubicki, Biernacki, and Grimonprez 2021) and **clustMD** (McParland and Gormley 2017) deal with mixed data (continuous + discrete). The **deepgmm** (Viroli and McLachlan 2020) package learns deep Gaussian mixture models, generalising the classical GMM with multiple layers. **IMIX** (Wang 2022) focuses on clustering multi-omic data that is learnt with the **mclust** package, and **coseq** (Rau 2022) implements RNA-Seq transcriptome clustering using the **Rmixmod** package.

- Some extend the EM algorithm on Gaussian distributions and overcome its main limitations. The **MGMM** (McCaw 2021) package deal with missing data, which is not relevant in unique dimension. The **mixsmsn**

package estimates skewed GMMs. **SMNCensReg** (Garay, Massuia, and Lachos 2022) 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 (A small simulation to evaluate the impact of outliers).

- We also removed packages that were limited in their functionalities or complex to install. Indeed, **ClusterR** (Mouselimis 2022) (k-means), **rebmix** (REBMIX), **nor1mix** (univariate dimension only, wrong initialisation process), **MixAll** (Iovleff 2019) (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** (W.-C. Chen and Ostrouchov 2021) depends on the availability of the OpenMPI framework for its parallelised implementation of the EM algorithm.
- We also removed the **mixR** (Yu 2021) and **CAMAN** (Schlattmann, Hoehne, and Verba 2022) packages which have not been updated in the last two years or are still in beta version.

The popularity of the selected packages varies largely, as illustrated in Figure 3. Among them, **mclust** and **flexmix** are the most popular, followed by **mixtools** and **Rmixmod** packages. We used the **cranlogs** (Csárdi 2019) package to retrieve the daily number of downloads for each of the benchmarked packages, between the 30$^{\text{st}}$ of January, 2023 and the 30$^{\text{th}}$ of April, 2023.

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 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: Extensions of the EM algorithm to overcome its limitations, 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 (Lindsay 1995). 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). In addition, rather than employing a sequential and greedy strategy, an independent and parallelisable feature selection procedure, through the model space, would have sped up by several orders of magnitude the estimation. To that end, (J. Chen and Chen 2008; He and Chen 2016) 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[7]. Such a strategy is commonly used in *ensemble learning*.

[7]Indeed, by simply uniformly sampling among the 2^D models available, the probability of getting models with $D/2$ features is

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 (27)):

$$1 - \Big(1 - (1 - p_j)^n - n \times (1 - p_j)^{n-1} \times p_j\Big)^N \tag{27}$$

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 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.

Additionally, as stated in Parameters estimation in a high-dimensional context, the intrinsic dimension d_j for each cluster j is a hyperparameter, which is generally inferred independently from the GMM estimation itself. While a variety of methods from the field of factor analysis, enumerated in Factor criteria selection, have been developed to estimate the intrinsic dimension, to our knowledge, only two of them have been implemented in CRAN packages: the *Cattell's scree-test* (Cattell 1966) or the dimension selection graph using one of the *penalty metric* discussed in the appendix Model selection (Bergé, Bouveyron, and Girard 2012). However, while **HDclassif** natively implements a performance criterion method for determining the dimension of the spanning space, performed under the hood by function `mixsmsn::hdcc`, none of the other packages evaluated implemented a dimension selection feature. Instead, we infer it for each of the packages dedicated to high-dimensionality with HDclassif, using using the so-called model "AkjBkQkD", for which the intrinsic dimension is common to all components but the characteristics unique for each component Finally, we use among all supplied parametrisations, the least constrained one. Namely, we used the model "AkjBkQkDk" with HDclassif, in which not only the individual features of the covariance matrix but also the spanning dimension are unique for each cluster, and function `mcfa` of the **EMMIXmfa** package, in which the *transition matrix* is common to all components (referred to as the orientation matrix in Appendix Parameters estimation in a high-dimensional context.

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

much higher than drawing models at the boundaries, displaying either few or close to $|D|$ covariates.

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 Dai and Mukherjea (2001) and in Book Robert and Casella (2010).

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 (21)), 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 1, was internally available in the package, we use it with the same hyperparameters described in main paper, section *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 5, 10 and 15, 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 (Mersmann

2021) 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.

Appendix D: comprehensive report from the univariate and multivariate benchmark

Packages used to generate the reports and visualisations

To compute the summary metrics and generate the corresponding boxplots of the bootstrapped parameters, we made extensive use of the facilities provided with the **tidyverse** (Wickham et al. 2019) packages, including:

- **tibble** (Müller and Wickham 2023) to visually and uniformly store the many datasets generated by our benchmark. We then used **readr** (Wickham, Hester, and Bryan 2023) to save and import in a readable format the summary metrics associated with each scenario and the tables listing the main functionalities implemented in the packages studied, **dplyr** (Wickham, François, et al. 2023) to manipulate the data stored in the tibbles and **purrr** (Wickham and Henry 2023) to manipulate the nested tibbles and perform functional programming.
- **ggplot2** (Wickham, Chang, et al. 2023; Wickham 2016) for data visualization, including generating density graphs in the univariate and bivariate framework, and factorial projection for the high-dimensional framework.
- **stringr** (Wickham 2022), for strings, and **forcats** (Wickham 2023), for factors, were particularly useful for customising and ordering the packages in our graphical representations, in order to highlight differences between them.

In addition to the array of packages within the *tidyverse* ecosystem, we utilized the **flextable** (Gohel and Skintzos 2023) and **kableExtra** (Zhu 2021) packages to facilitate the generation of summary reports in HTML and PDF formats.

Furthermore, we would like to express our gratitude for the contributions of **knitr** (Xie 2023, 2015), **rmarkdown** (Xie, Allaire, and Grolemund 2018; Allaire et al. 2023), and the associated wrapper package **rjtools**(O'Hara-Wild et al. 2023), which greatly streamlined the process of creating these HTML and PDF reports.

In a more specialised context, we harnessed the features offered by these packages:

- **ComplexHeatmap** (Gu, Eils, and Schlesner 2016; Gu 2022) to generate the heatmaps of the correlation matrices. This was complemented by **RColorBrewer** (Neuwirth 2022) for effective management of the R colour palette.
- **cowplot** (Wilke 2020), **grid** (R Core Team 2023), and **gridExtra** (Auguie 2017) were used for aggregating and merging multiple plots.
- **ggtext** (Wilke and Wiernik 2022), **glue** (Hester and Bryan 2022), and **scales** (Wickham and Seidel 2022) were employed to enhance the clarity and readability of both x and y ticks and labels.

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 9). 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[8].
   - 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 (28), Equation 3 p.5 Benaglia et al. (2009)), but without exploiting the tip of Taylor's development over $\log(1+x)$:

$$\eta_i(j) = \frac{p_j\,\varphi_{\zeta_j}(x)}{\sum_{j=1}^{k} p_j\,\varphi_{\zeta_j}(x)} = \frac{\frac{p_j\,\varphi_{\zeta_j}(x)}{p_{j_{\min}}\,\varphi_{\zeta_{j_{\min}}}(x)}}{1 + \frac{\sum_{j\neq j_{\min}} p_j\,\varphi_{\zeta_j}(x)}{p_{j_{\min}}\,\varphi_{\zeta_{j_{\min}}}(x)}} \tag{28}$$

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)[9], 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 (12))[10].

We enumerate below some additional features supplied by the packages:

- 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[11]. 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).

[8] Additionally, **bgmm** does not update the estimated variances if any newly computed variance is below the criterion stop. A remarkable side-effect of these features, as shown in Figure 2, is that the **bgmm** package is less sensitive to the presence of outliers.

[9] Indeed, at least one of the component was removed in 80% of our estimations in the unbalanced and overlapping case (scenario U9 in 5) and in 20% of the simulations in the unbalanced and well-separated case (scenario U3 in 5).

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

[11] 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

- **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[12]. However, unlike **flexmix**, the algorithm stops instead of re parametrising 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.

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

Supplementary Figures and Tables in the univariate simulation

Table below (5) lists the complete set of parameters used to simulate the univariate Gaussian mixture distribution in our benchmark:

Table 5: 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 4-Figure 8 each summarise the benchmarking results associated with one of the scenarios listed in Table 5.

Summary tables 6- 9 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.

Table 6: MSE and Bias associated to scenario U1, in Table 5 (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 4 to Figure 8, 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 5.

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 5), 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, 5, illustrated in Figure 7), compared

Table 7: MSE and Bias associated to scenario U7, in Table 5 (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

to the simplest scenario (see U1, 5, shown in Figure 4). 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[13]:

- With well-separated components (Scenarios U1 and U7 in Table 5), the class of packages (namely the choice of the convergence criterion) has a negligible impact compared to the 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

[13]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 8: MSE and Bias associated to scenario U3, in Table 5 (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

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 5), 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

Table 9: MSE and Bias associated to scenario U9, in Table 5 (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

mixtools, are on average a hundred times slower.

Supplementary Figures and Tables in the bivariate simulation

Table below (10) lists the complete set of parameters used to simulate the multivariate Gaussian mixture distribution in our benchmark:

Table 10: 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

Figures 11- 14 are associated to scenarios B11 - B15 of Table 10. Summary tables 11-14 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 10 and isodensity plot in panel A, Figure 13), whereas the highest OVL is obtained with a negative correlation (scenario 11, Table 10and isodensity plot in panel A, Figure 11). 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 B11, in Table 10 (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

Table 12: MSE and Bias associated to scenario B12, in Table 10 (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

Table 13: MSE and Bias associated to scenario B14, in Table 10 (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

Table 14: MSE and Bias associated to scenario B15, in Table 10 (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

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 15, 16 and 17 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.

Supplementary Figures and Tables in the HD simulation

Table below (15) lists the complete set of parameters used to simulate Gaussian distributions in the high dimensional benchmark:

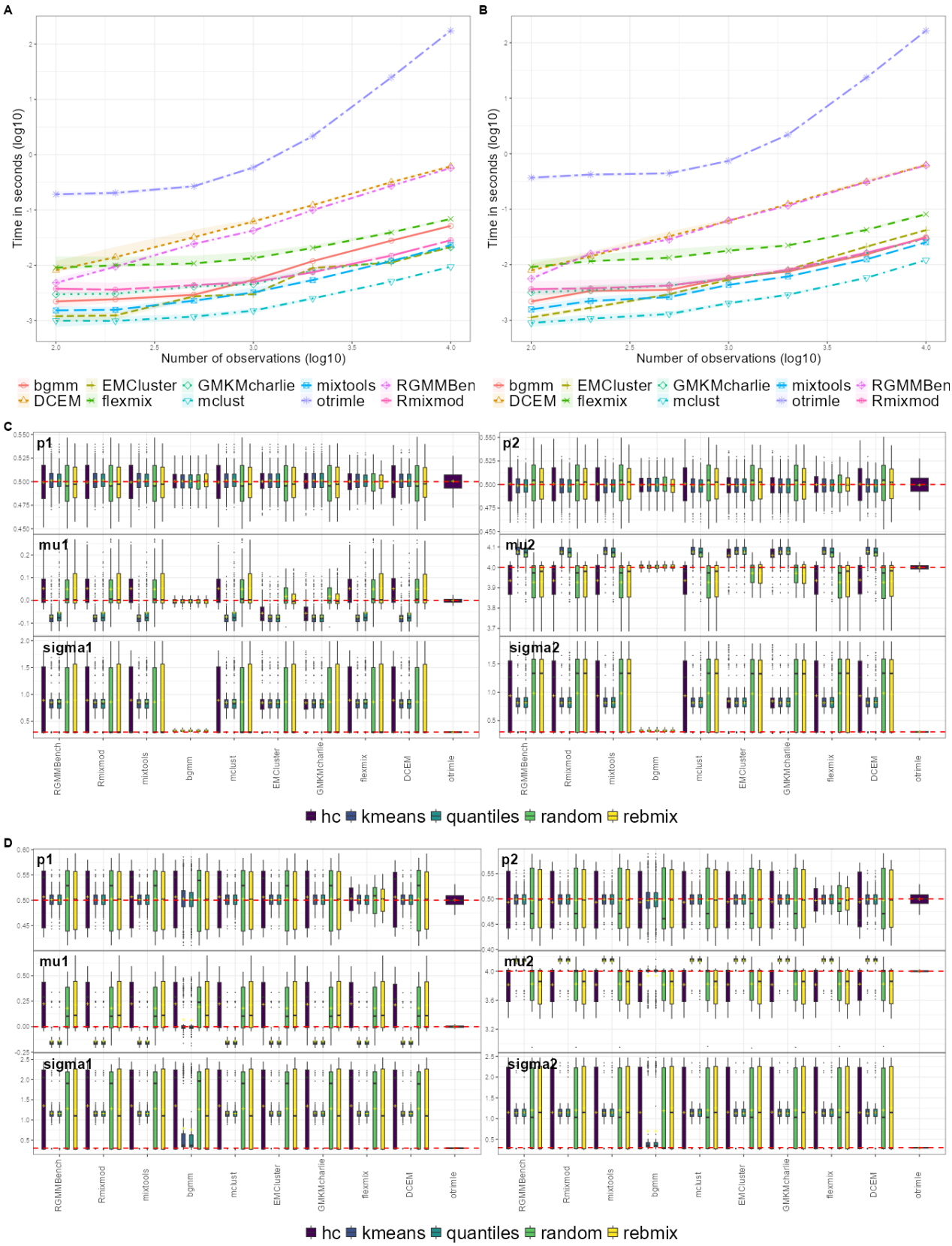


Figure 2: A) Execution times for the nine reviewed packages using hierarchical clustering initialisation, with on the left 2% of outliers in proportion and on the right, 4% of outliers. B) and C) Boxplots of the estimated parameters with $N = 200$ repetitions, $n = 2000$ observations and respectively 2% and 4% of outliers. The red dashed horizontal line corresponds to the true value of the parameters.

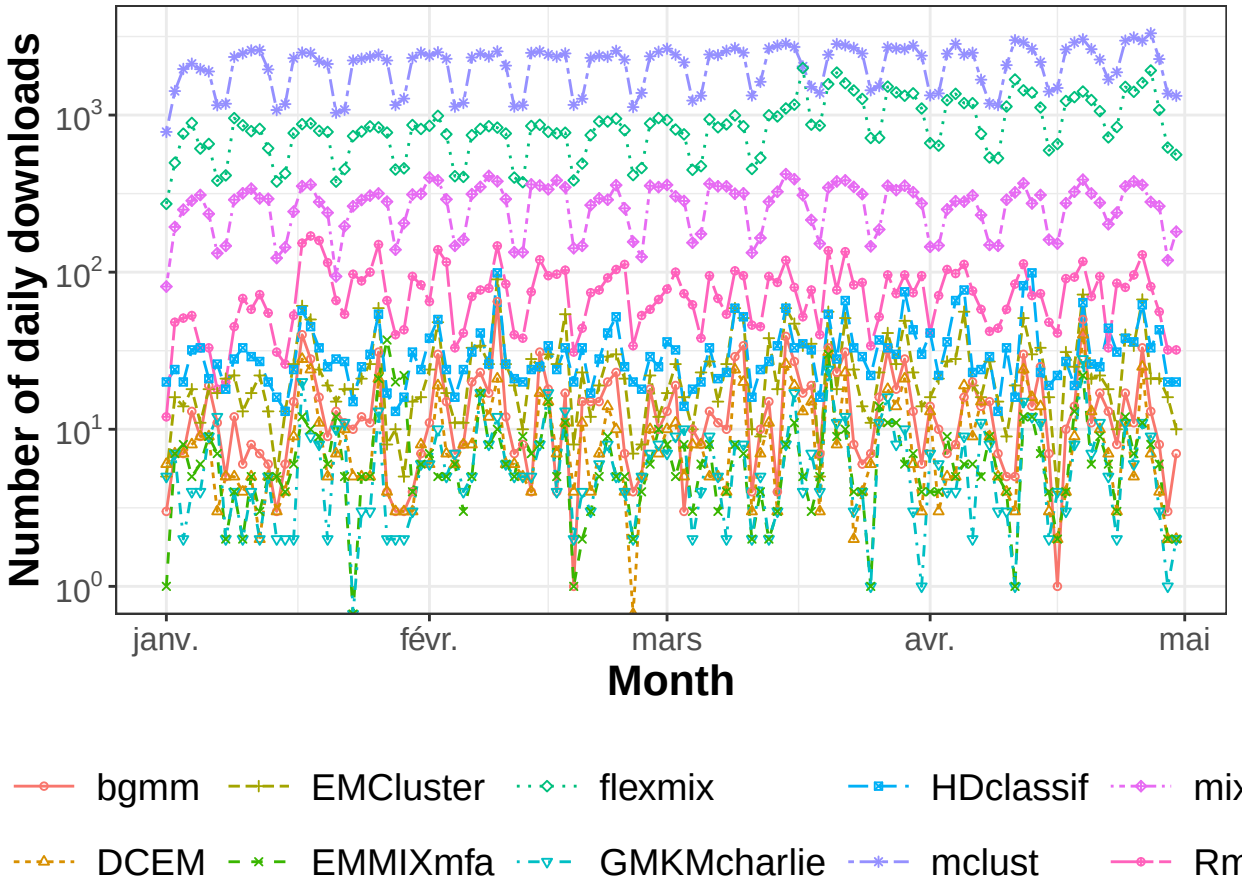


Figure 3: Number of daily downloads (logarithmic scale) from the CRAN mirror from the 1st of January to the 30th April 2022 for the seven R packages reviewed.

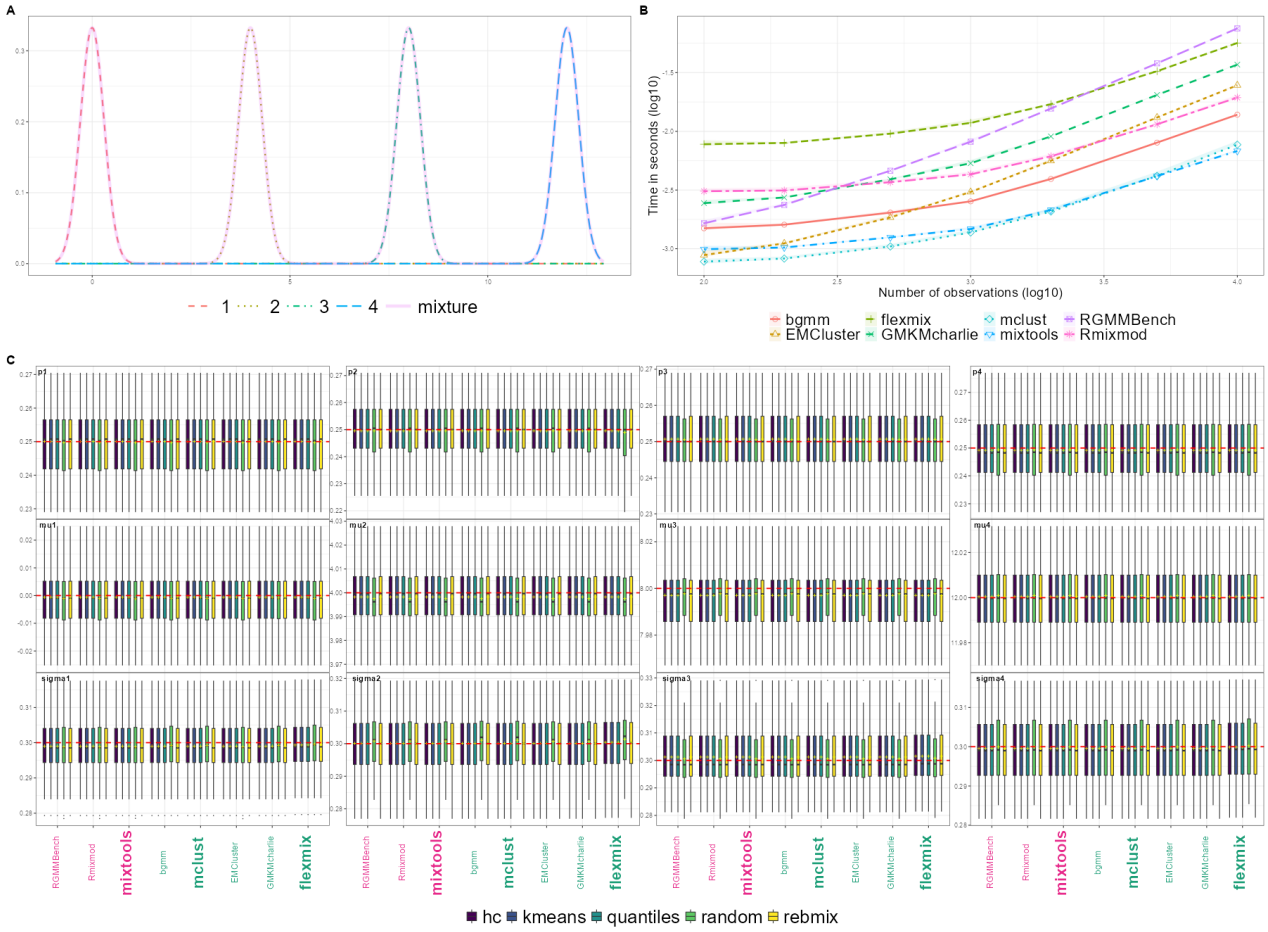


Figure 4: Benchmark summary plots of scenario U1 in Table 5 (balanced and well-separated components), organised as such: The panel A displays the distribution of the global mixture distribution $f_\theta(X)$ (pink solid line) and of each of its constitutive components scaled by their respective proportions (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.

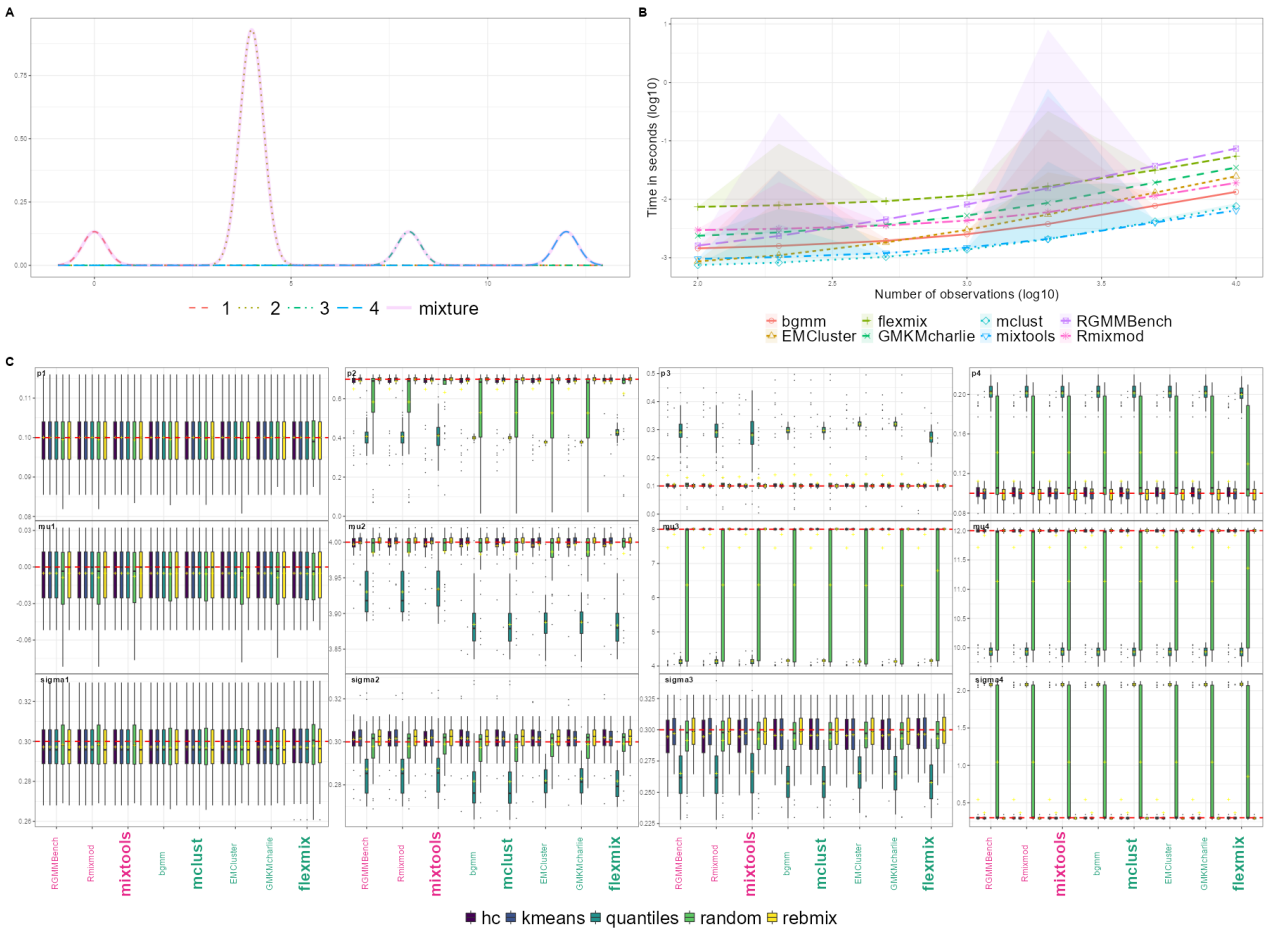


Figure 5: Benchmark summary plots of scenario U7 in Table 5 (unbalanced and well-separated components), with same layout as in Figure 4.

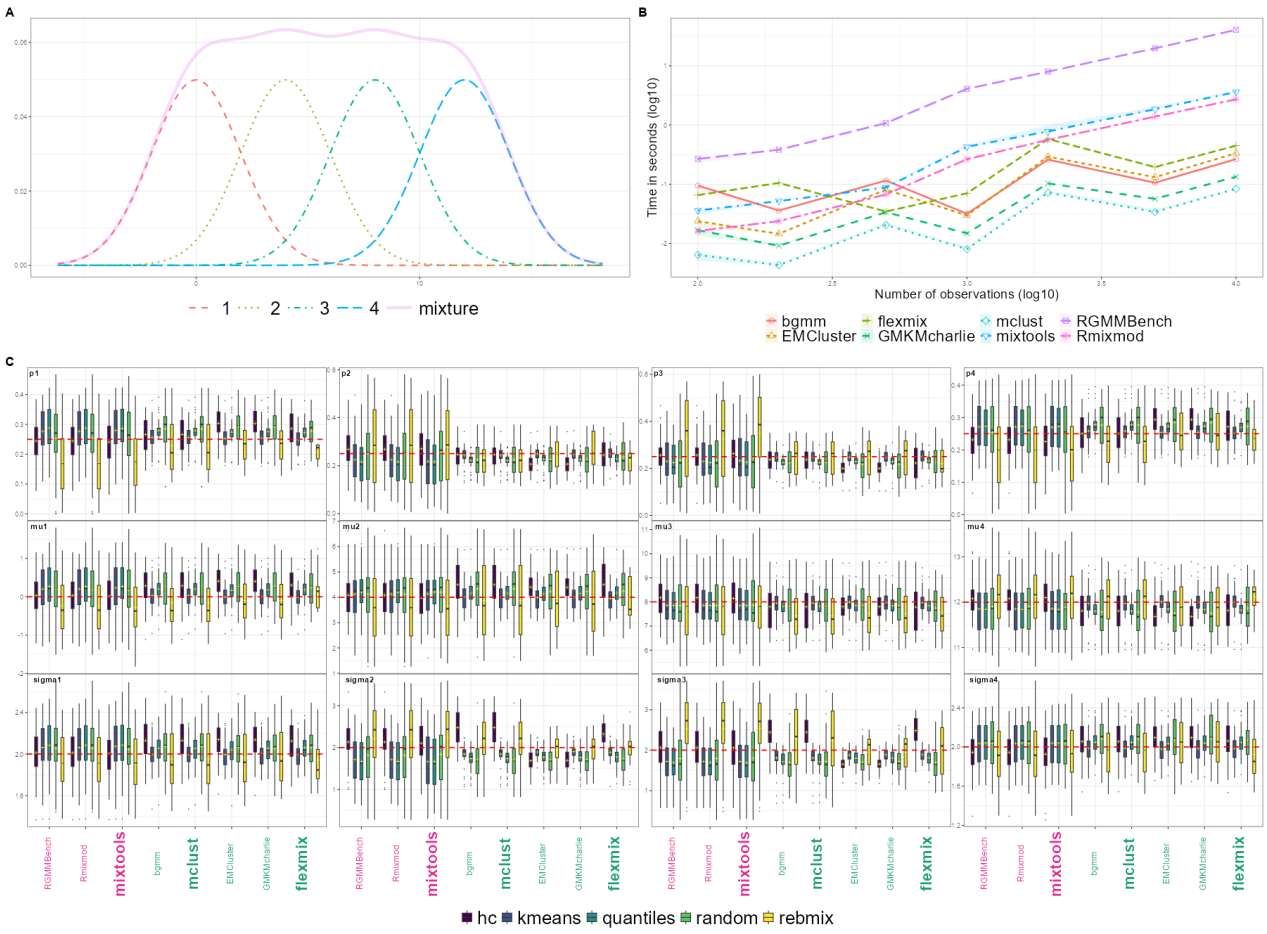


Figure 6: Benchmark summary plots of scenario U3 in Table 5 (balanced and overlapping components), with same layout as in Figure 4.

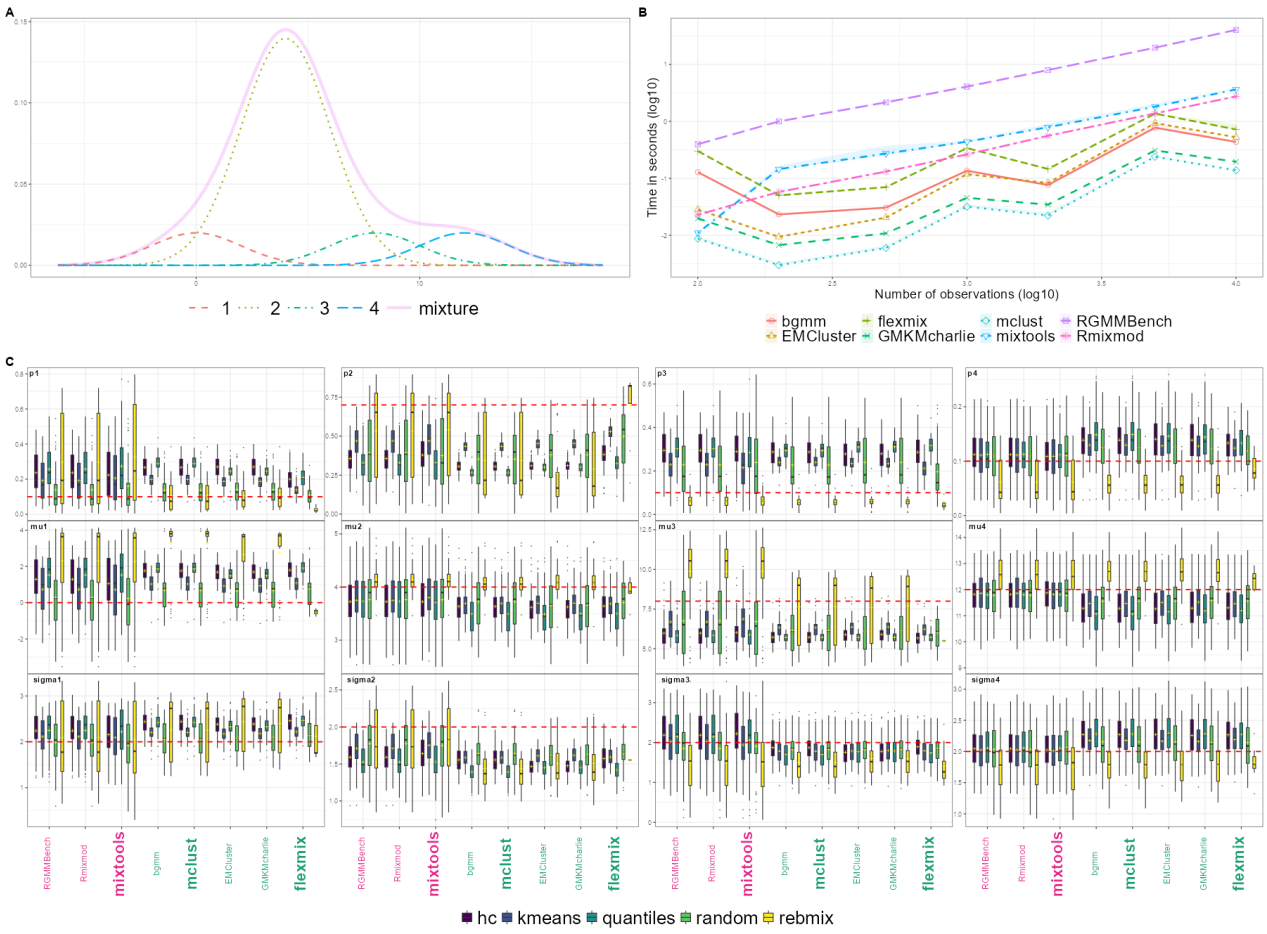


Figure 7: Benchmark summary plots of scenario U9 in Table 5 (unbalanced and overlapping components), with same layout as in Figure 4.

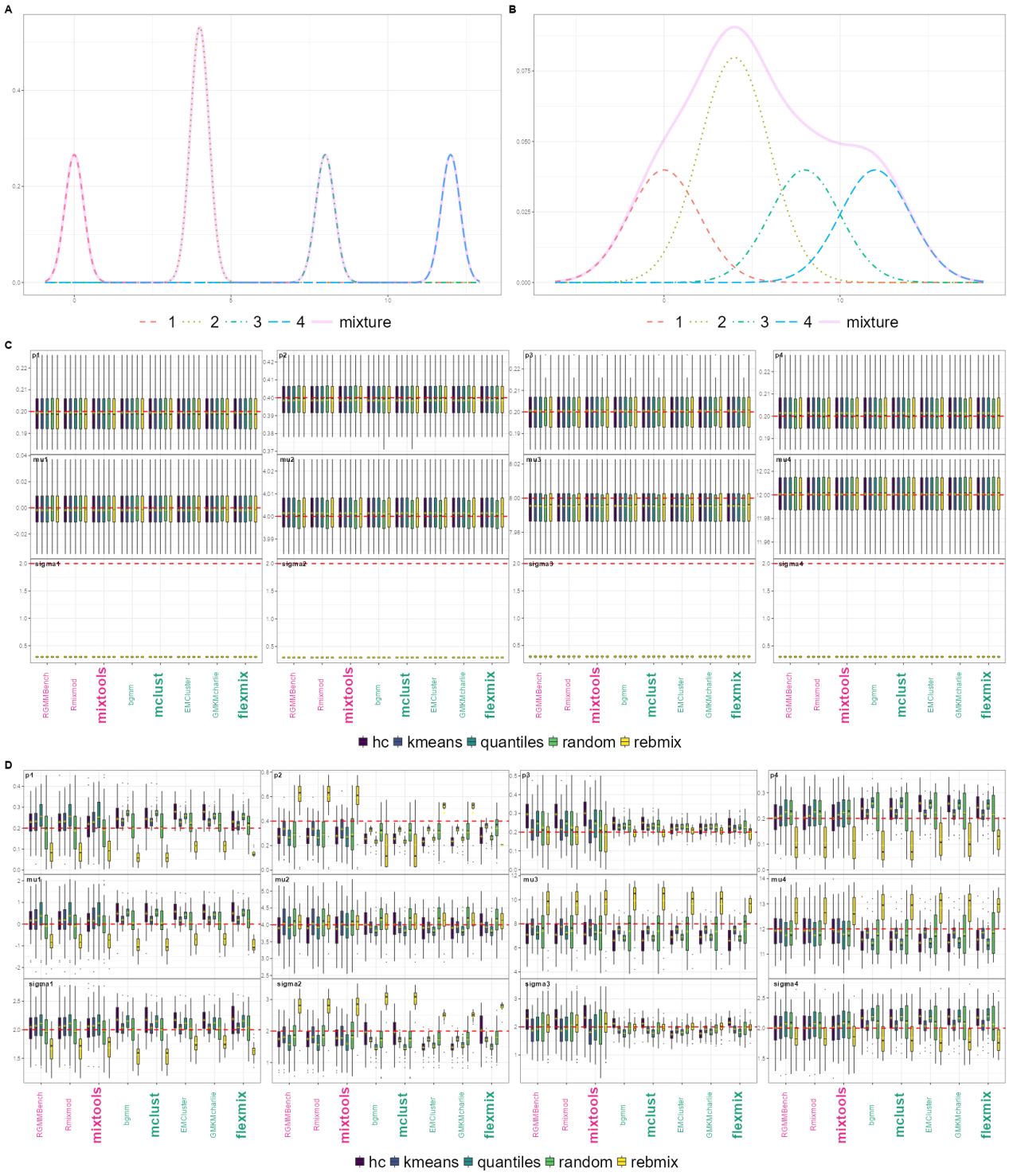


Figure 8: Benchmark summary plots of scenarios U4 and U6 in Table 5 (small unbalance, with additional overlap in scenario U6). Panel A and B display the univariate GMM distributions of respectively scenarios U4 and U6, and Panel C and D the benchmarked distributions of respectively scenarios U4 and U6, built as Panel C of Figure 4.

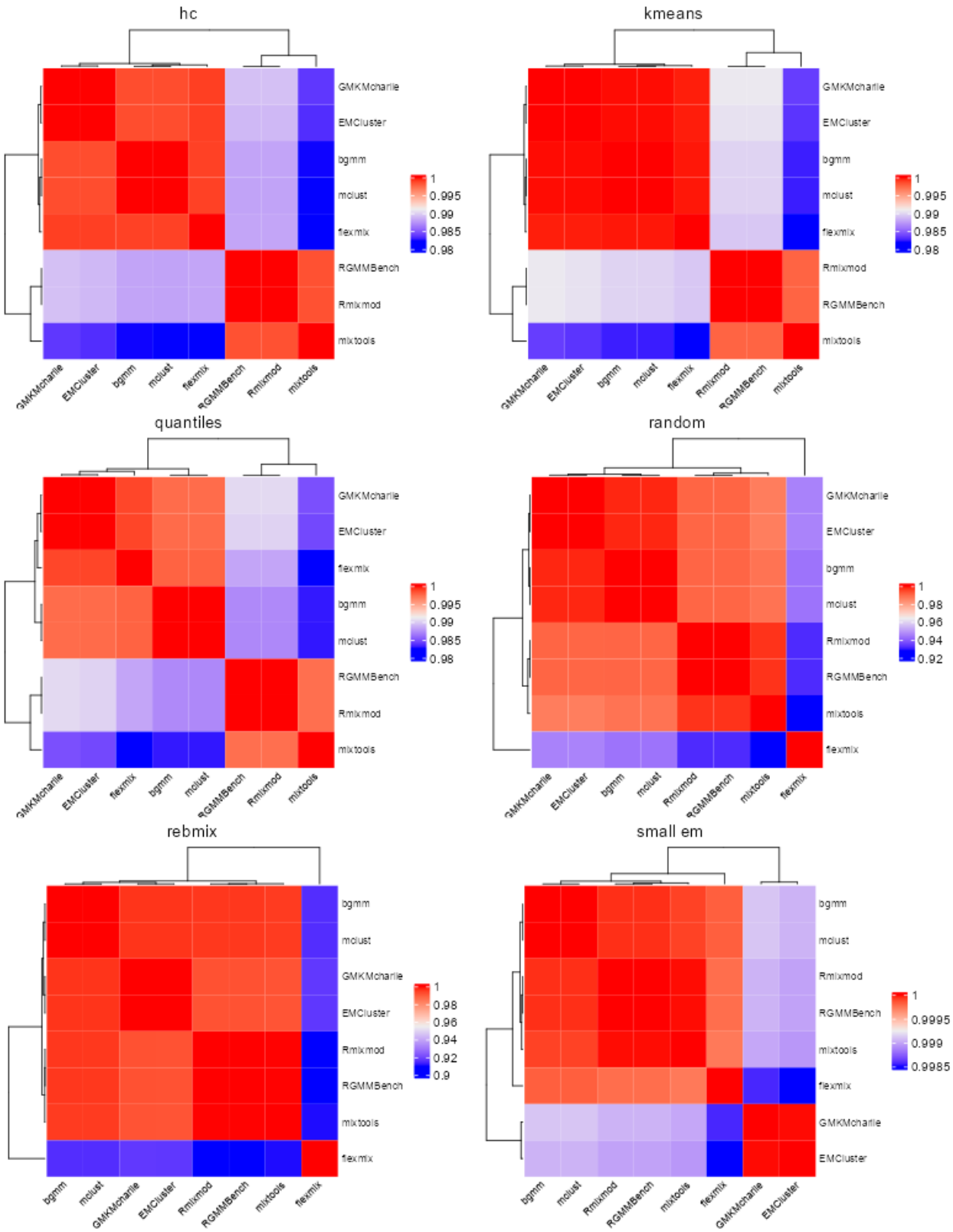


Figure 9: Correlation heatmaps of the estimated parameters extended to the four initialisation methods benchmarked, using the same configuration described in Figure (2), in the bivariate setting.

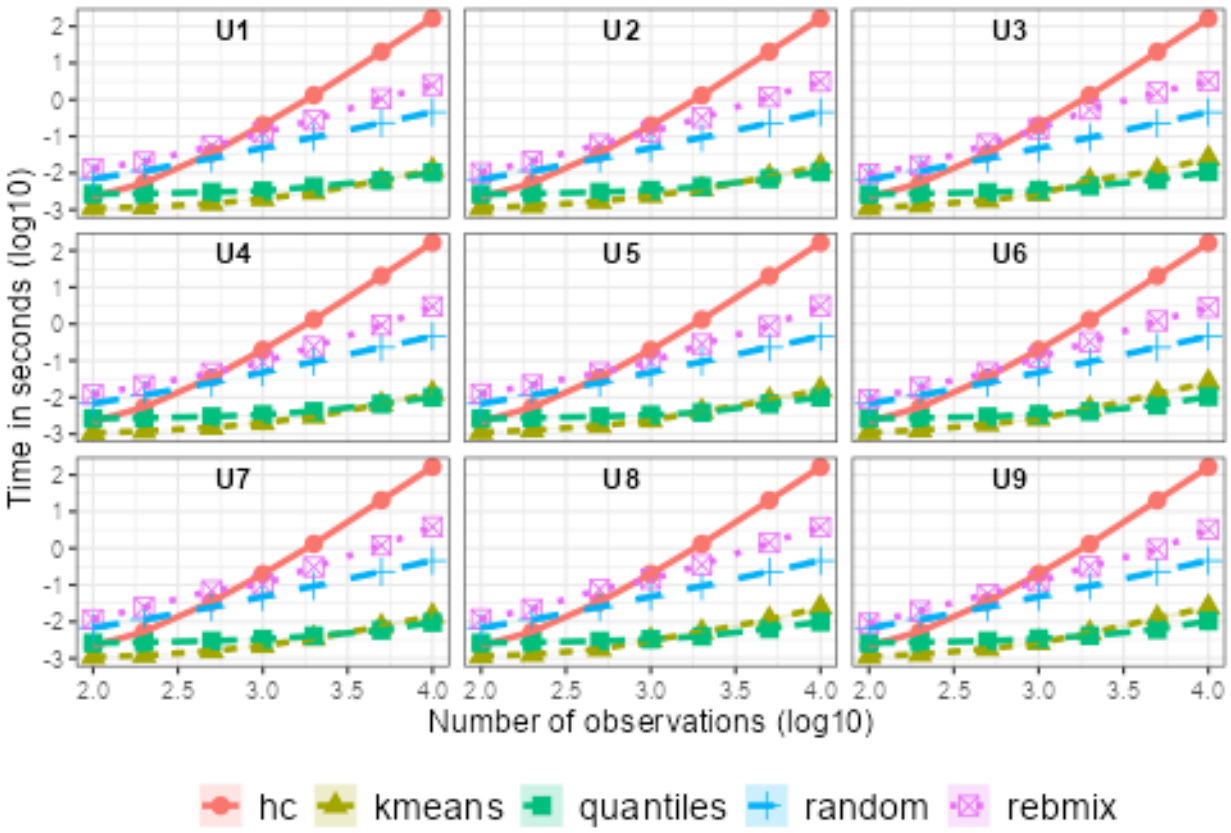


Figure 10: Distribution of the running times taken by each initialisation algorithm enumerated in Table 1, across all scenarios listed in Table 5, sorted by increasing ID number in the lexicographical order.

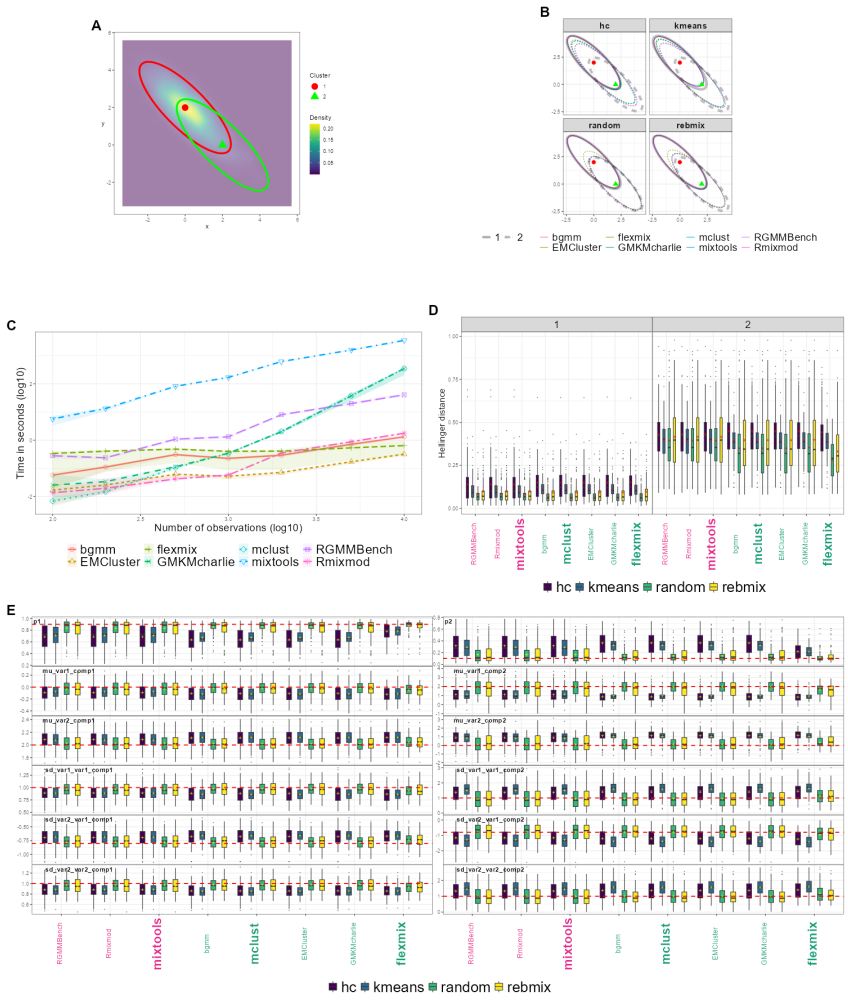


Figure 11: Results of scenario B11 in Table 10 (unbalanced, overlapping and negative correlated components), organised as such: The panel A displays the bivariate contour maps associated to the two-components multivariate Gaussian distribution corresponding to the parametrisation described by the scenario, 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 coloured bands represent the 5^{th} and 95^{th} percentiles of the running time. The distributions of the Hellinger distances 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 each column panel associated to the parameters of one 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 fourth 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.

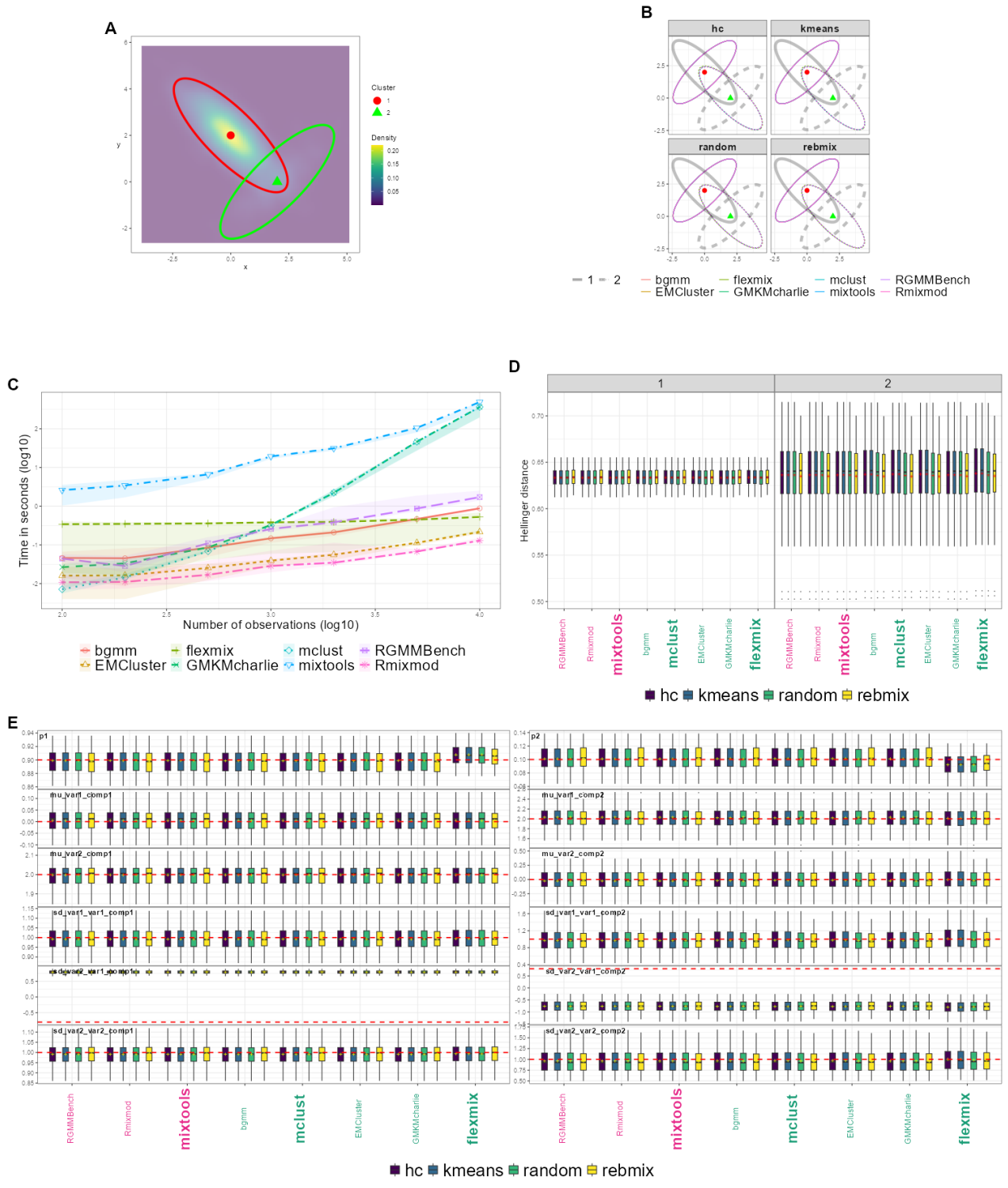


Figure 12: Results of scenario B12 in Table 10 (unbalanced, overlapping and opposite correlated components), with the same layout as Figure 11.

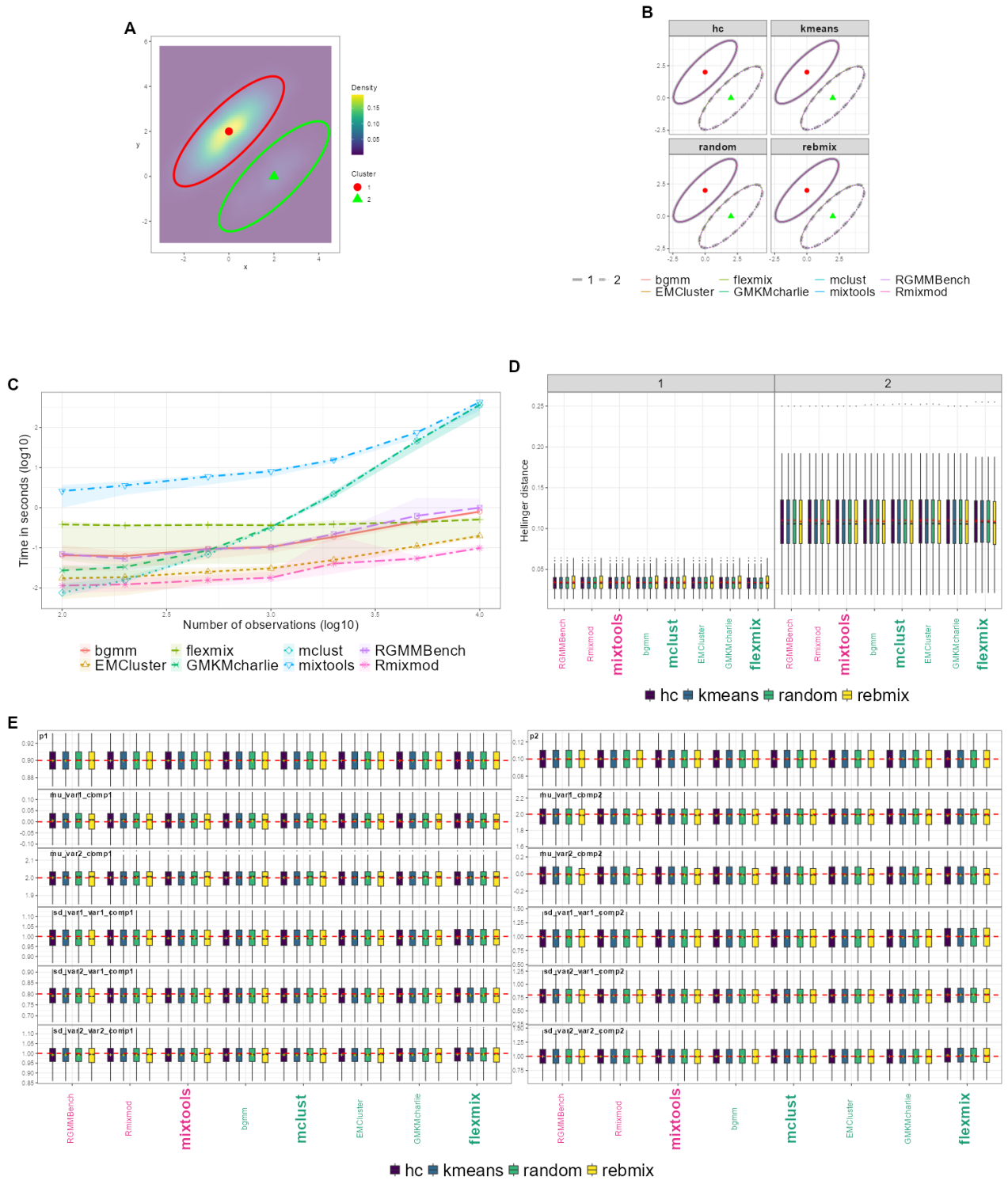


Figure 13: Results of scenario B14 in Table 10 (unbalanced, overlapping and positive correlated components), with the same layout as Figure 11.

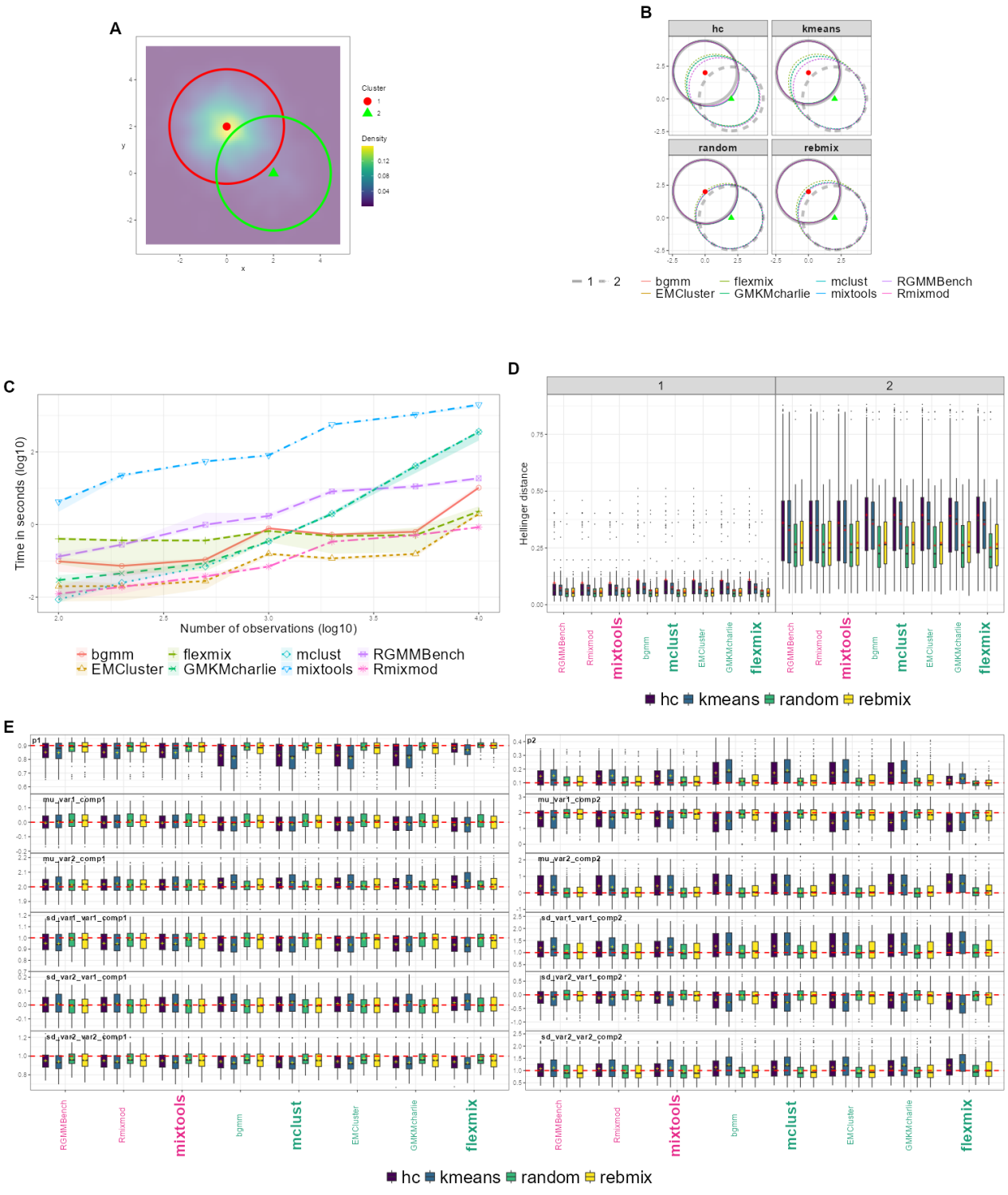


Figure 14: Results of scenario B15 in Table 10 (unbalanced, overlapping and uncorrelated components), with the same layout as Figure 11.

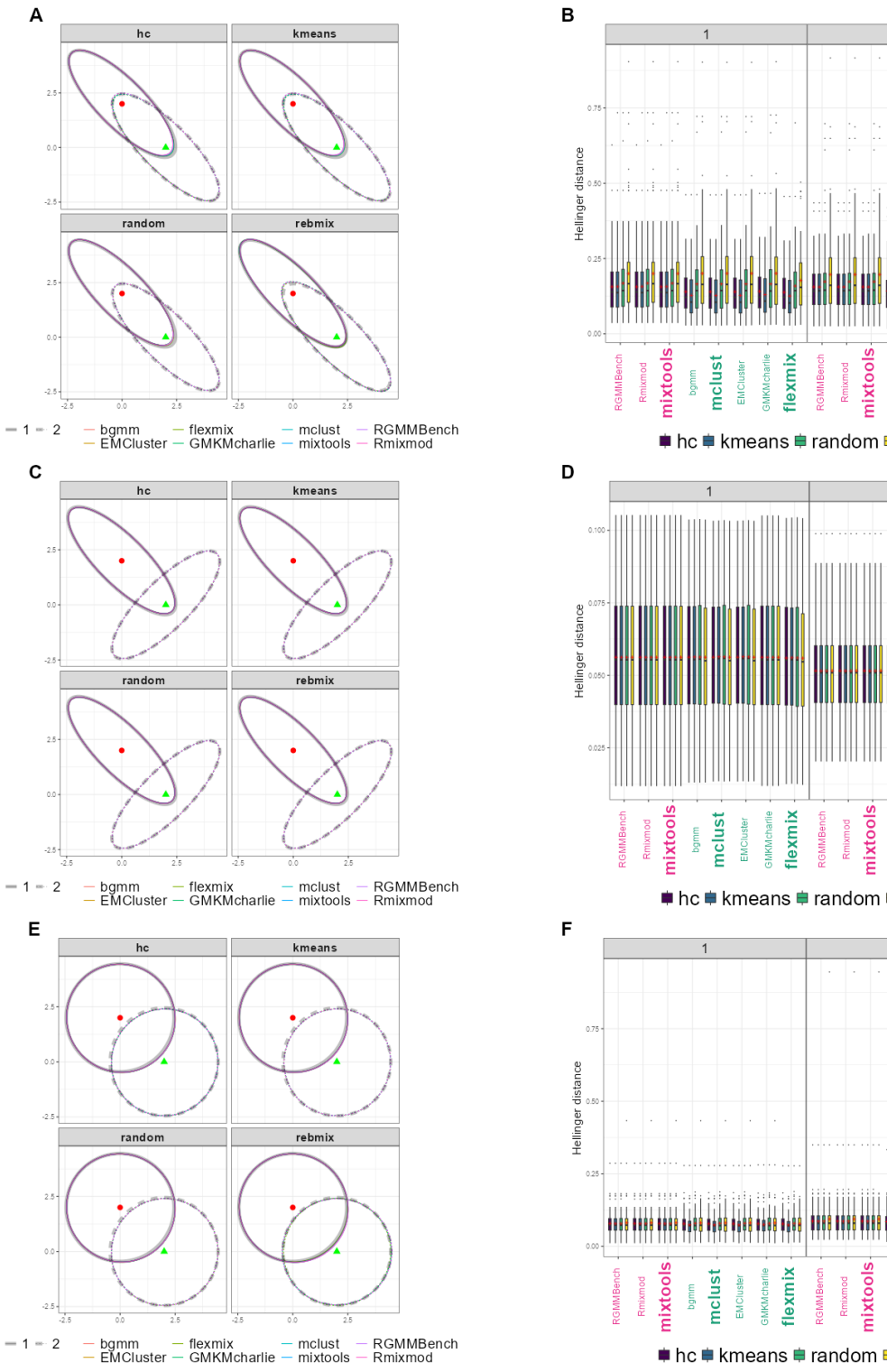


Figure 15: Benchmark summary plots of respectively scenarios B1, B2 and B5 in Table 10 featuring balanced and overlapping clusters. Summary plots of B1, B2 and B5 are represented in this order on each row, with the left column displaying the 95% confidence ellipsoidal regions associated to the mean estimated parameters across each package and the right column the distribution of the Hellinger distances.

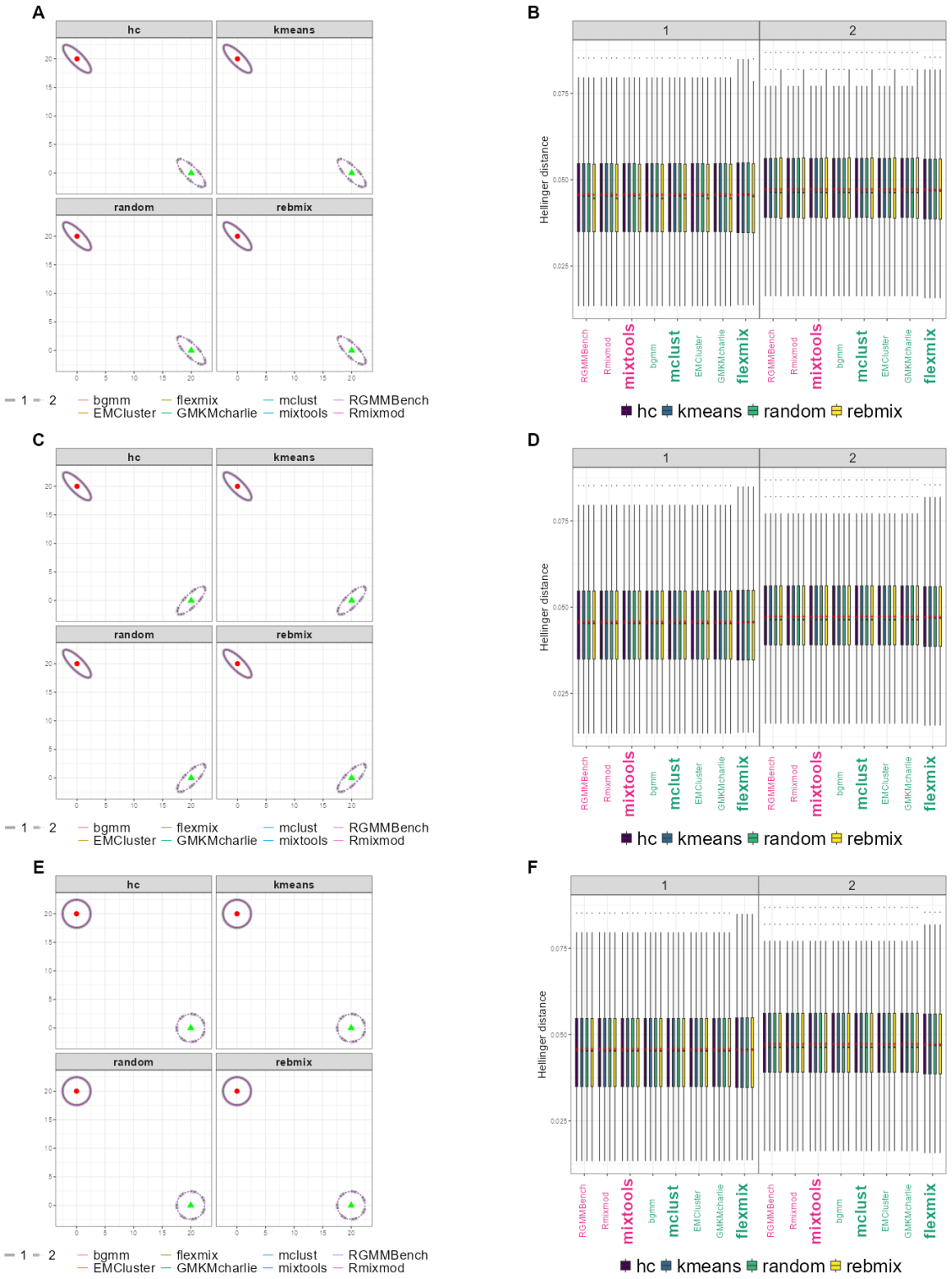


Figure 16: Benchmark summary plots of respectively scenarios B6, B7 and B10 in Table 10 featuring balanced and well-separated clusters, with the same layout as Figure 15.

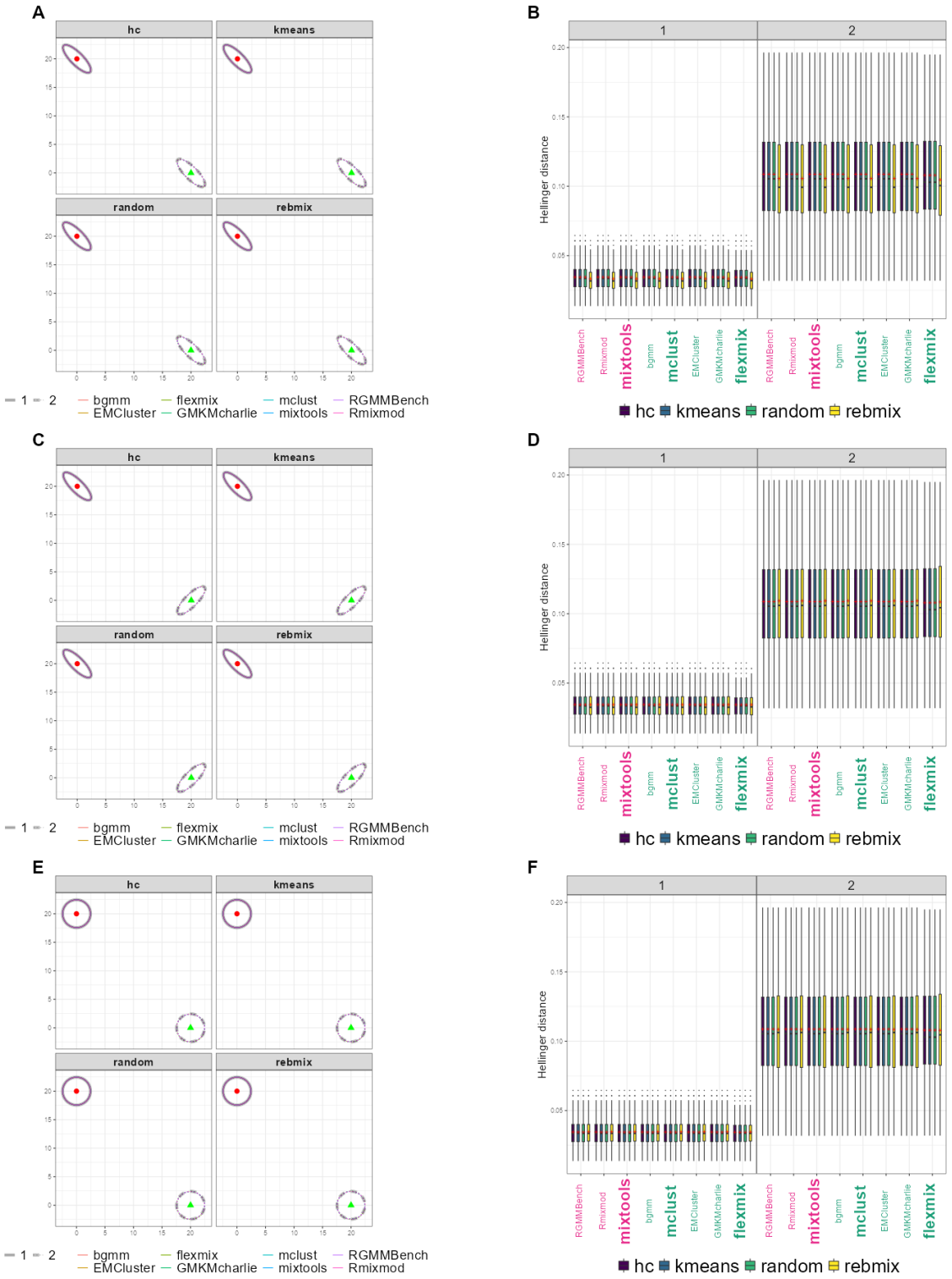


Figure 17: Benchmark summary plots of respectively scenarios B16, B17 and B20 in Table 10 featuring unbalanced and well-separated clusters, with the same layout as Figure 15.

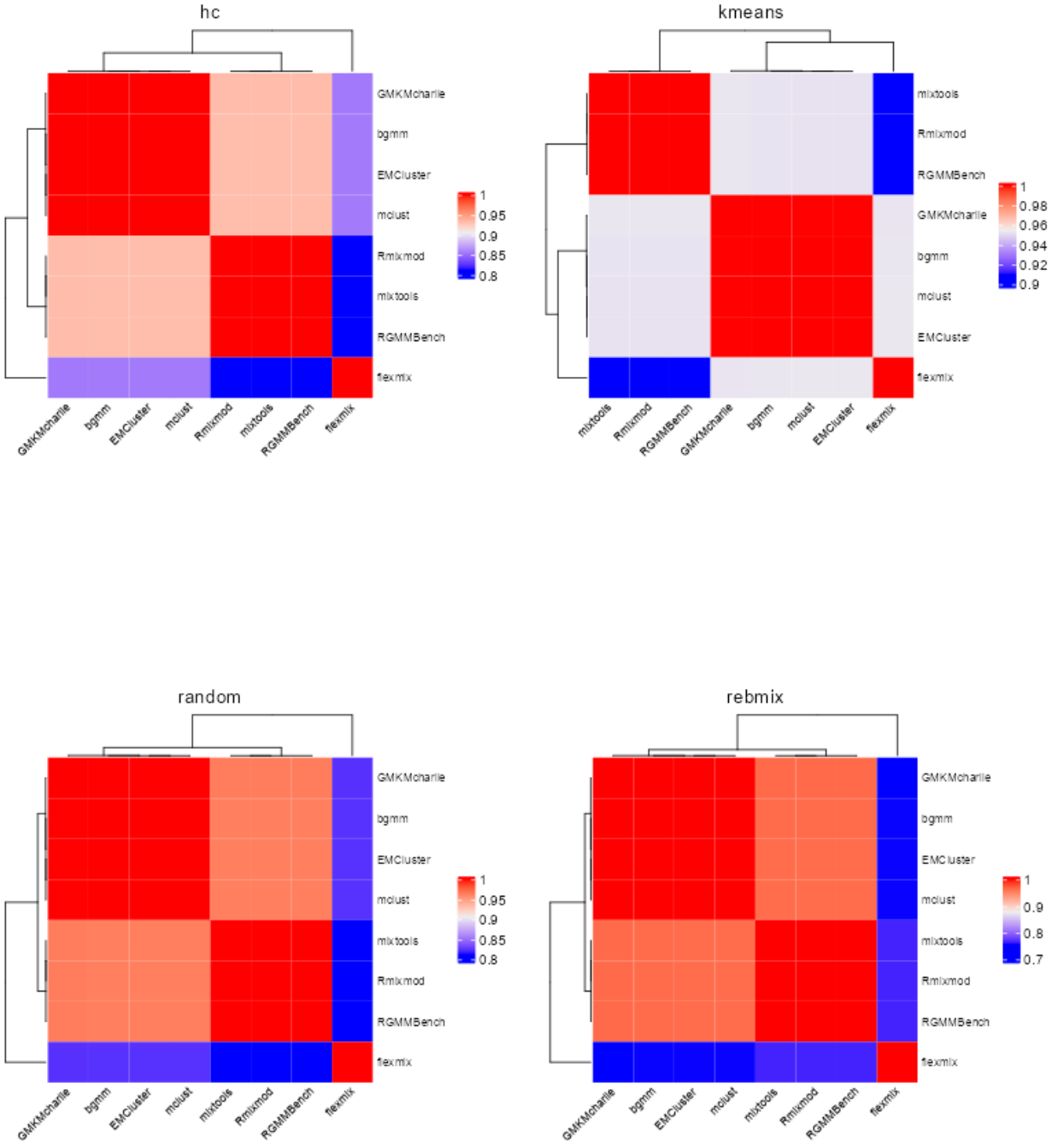


Figure 18: Correlation heatmaps of the estimated parameters in the bivariate setting extended to the four initialisation methods benchmarked, with the most discriminating scenario B11, using the same process described in Figure (2).

Table 15: The 16 parameter configurations tested to generate the samples in a high dimensional context. The first digit of each ID index refers to an unique parameter configuration (identified by its level of overlap, entropy and topological structure, either circular or ellipsoidal, of the covariance matrix, while the lowercase letter depicts the number of observations, a) with $n = 200$ and b) with $n = 2000$.

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

Table 16: MSE and Bias associated to scenario HD4a, in Table 15 (unbalanced, separated and ellipsoidal components).

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ	% Success
mixtools / Rmixmod / RGMMBench	hc	0.0333	0.0212	0.0106	0.0020	0.056	0.097	100
	kmeans	0.0333	0.0212	0.0106	0.0020	0.056	0.097	100
	rebmix	0.3244	0.1980	0.0845	0.0720	0.395	0.535	98
mclust / flexmix / GMKMcharlie	hc	0.0333	0.0212	0.0106	0.0020	0.056	0.097	100
	kmeans	0.0333	0.0212	0.0106	0.0020	0.056	0.097	100
	rebmix	0.2553	0.1444	0.0924	0.0470	0.364	0.596	85
bgmm	hc	0.0337	0.0214	0.0107	0.0070	0.064	0.096	100
	kmeans	0.0338	0.0216	0.0106	0.0074	0.064	0.096	100
	rebmix	0.4818	0.1152	0.3442	0.0320	0.223	2.329	94
EMCluster	hc	0.0333	0.0212	0.0107	0.0023	0.056	0.096	100
	kmeans	0.0334	0.0213	0.0106	0.0018	0.056	0.096	100
	rebmix	1.5983	1.0992	0.3794	0.3100	2.018	2.575	84
HDclassif	hc	8.4062	8.3936	0.0111	0.0020	10.426	0.149	100
	kmeans	7.9407	7.9282	0.0111	0.0019	10.081	0.149	100
	rebmix	7.9803	7.9514	0.0273	0.0044	10.128	0.262	84
EMMIXmfa	hc	4.0605	3.3317	0.3357	0.6500	5.757	2.772	95
	kmeans	3.8790	3.2175	0.3372	0.5400	5.781	2.777	96
	rebmix	4.0127	3.2715	0.3337	0.5700	5.680	2.757	80

Table 17: MSE and Bias associated to scenario HD7a, in Table 15 (balanced, overlapping and ellipsoidal components).

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ	% Success
mixtools / Rmixmod / RGMMBench	hc	5.8544	2.2153	3.5586	0.0450	2.084	6.704	100
	kmeans	5.4773	1.9490	3.4819	0.0086	2.323	6.861	100
	rebmix	6.9243	2.5185	4.2898	0.0620	2.670	7.626	97
mclust / flexmix / GMKMcharlie	hc	6.0584	2.4737	3.5198	0.0180	2.565	7.624	100
	kmeans	5.6388	2.1597	3.4549	0.0140	2.744	8.266	100
	rebmix	6.5397	2.4738	3.9661	0.0700	2.774	7.764	93
bgmm	hc	9.5015	5.1348	4.1086	0.1000	3.720	10.310	100
	kmeans	8.7930	4.7119	3.8693	0.1500	3.932	10.108	100
	rebmix	10.3630	5.6474	4.4026	0.2700	3.798	10.049	97
EMCluster	hc	6.4022	2.8255	3.5124	0.0120	3.141	9.086	100
	kmeans	6.4333	2.8740	3.5523	0.0110	4.210	11.007	100
	rebmix	6.5527	2.9643	3.4862	0.0580	3.051	9.253	93
HDclassif	hc	15.9010	11.5382	4.2950	0.1400	10.846	10.100	100
	kmeans	15.3377	10.9441	4.3716	0.0087	10.990	10.771	100
	rebmix	16.1231	11.1103	4.9113	0.1600	10.761	10.513	93
EMMIXmfa	hc	4.8606	1.6546	3.1856	0.0160	2.030	7.395	15
	kmeans	4.4039	1.4129	2.9701	0.0260	1.734	6.236	21
	rebmix	5.0984	2.0057	3.0689	0.0470	2.314	7.613	16

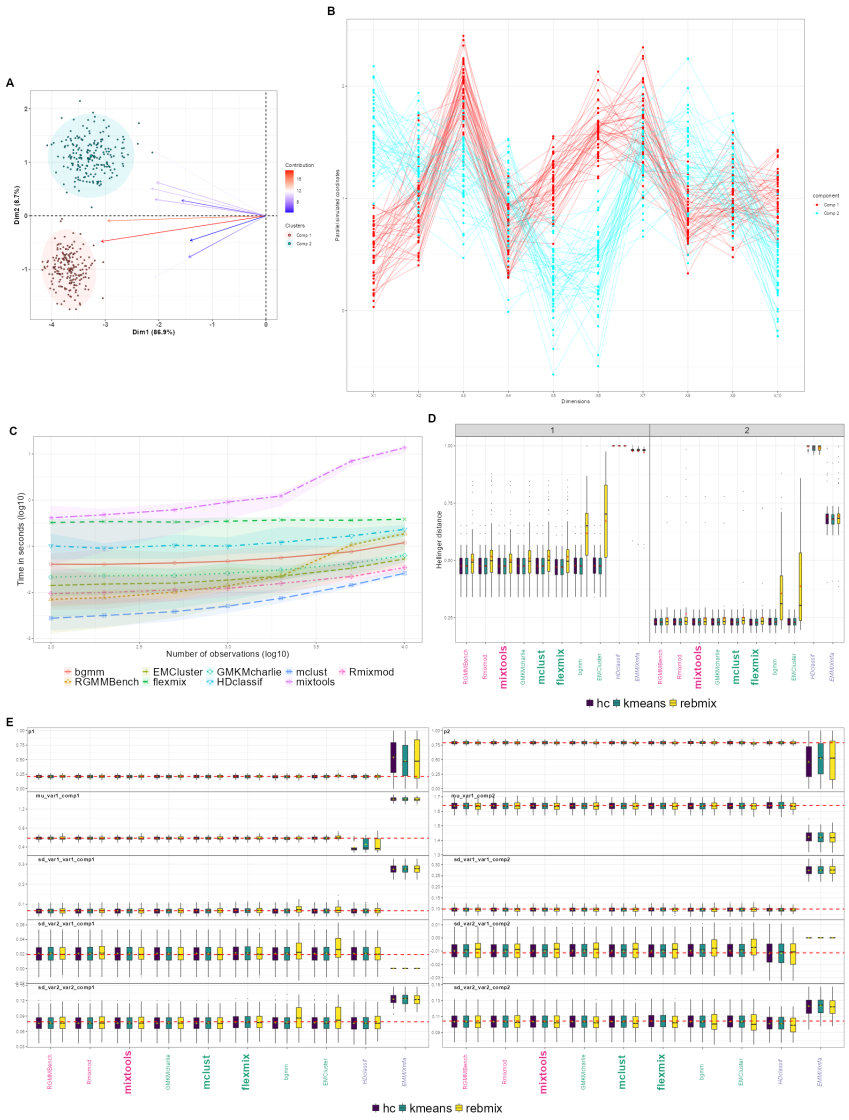


Figure 19: Results of scenario HD4a) in Table 15 (unbalanced, overlapping and negative correlated components), organised as such: The panel A displays the bivariate factorial projection of a random sample drawn from the 10-dimensional multivariate Gaussian distribution parametrised by Table 15. Each component is associated to a specific color, a centroid whose coordinates are given by the mean components' elements in the bivariate projected space and a 95% confidence ellipse. Arrows represent the correlation circle of the dimensional variables. Both panels were displayed respectively using functions `factoextra::fviz_eig` and `factoextra::fviz_pca_biplot` while the underlying computations proceed from the principal component analysis performed by `ade4::dudi.pca` preceded by standard scaling of the sampling dataset. The panel B pictures the *parallel distribution plots* from a random sampling of $n = 100$ observations, generated using `GGally::ggparcoord`, and representing the coordinates of each simulated data point in 10 dimensions. 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 distributions of the Hellinger distances are computed for each component in Panel D, each initialisation method and each package with respect to the true Gaussian distribution expected for each component. In panel E we represent the boxplots associated with the distribution of some of the estimates. Since it was impractical to represent all of the $k + kD + k\frac{D\times(D+1)}{2}$ with $k = 2$ and $D = 10$ parameters, we only represent the first component's mean, two first components' variances and their covariance term.

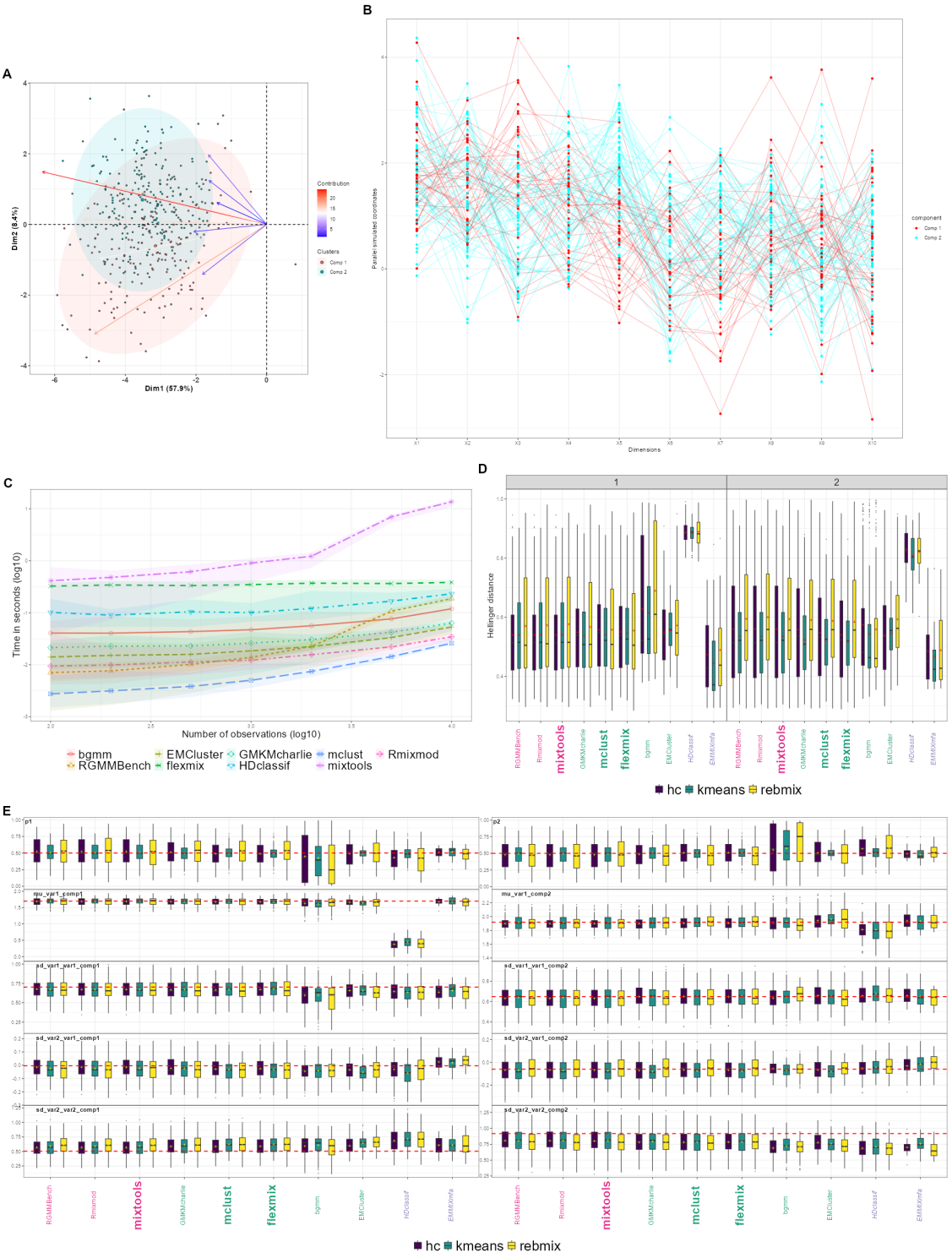


Figure 20: Results of scenario HD7a) in Table 15 (balanced and overlapping components, with full covariance structure), with the same layout as Figure 19.

Table 18: MSE and Bias associated to scenarios HD5a) and HD6a), in Table 15 (overlapping and spherical-distributed components). We delimite each scenario by doubled backslashes with respectively balanced and unbalanced clusters.

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ	% Success
mixtools / Rmixmod / RGMMBench	hc	4.2772 // 19.5172	0.9198 // 1.9835	3.3027 // 17.4943	0.017 // 0.069	0.995 // 0.6	3.571 // 4.381	100 // 100
	kmeans	3.9776 // 17.2212	0.8279 // 1.6336	3.1111 // 15.5684	0.072 // 0.069	0.841 // 0.82	3.023 // 5.034	100 // 100
	rebmix	9.3136 // 25.8028	2.7793 // 4.2893	6.4009 // 21.3519	0.15 // 0.22	3.619 // 2.507	9.061 // 11.826	96 // 80
mclust / flexmix / GMKMcharlie	hc	2.9743 // 18.1175	0.5862 // 1.7729	2.3612 // 16.3168	0.024 // 0.057	0.449 // 0.514	2.127 // 4.412	100 // 100
	kmeans	2.5629 // 15.2959	0.4642 // 1.5608	2.0855 // 13.7206	0.085 // 0.086	0.671 // 1.047	1.67 // 5.801	100 // 100
	rebmix	8.2907 // 23.7588	2.6468 // 4.1629	5.5421 // 19.4579	0.12 // 0.22	3.438 // 2.543	8.792 // 11.94	96 // 69
bgmm	hc	2.4088 // 33.8392	0.7261 // 9.0609	1.6153 // 24.6796	0.12 // 0.038	0.652 // 1.986	1.98 // 10.77	100 // 100
	kmeans	2.0912 // 28.5103	0.5899 // 7.5426	1.4577 // 20.8989	0.091 // 0.025	0.566 // 1.45	1.738 // 9.783	100 // 100
	rebmix	4.6278 // 35.9294	1.9526 // 11.0184	2.5372 // 24.6276	0.048 // 0.22	0.632 // 2.023	2.96 // 12.729	98 // 86
EMCluster	hc	2.5152 // 17.7053	0.5087 // 2.1191	1.9849 // 15.5379	0.024 // 0.12	0.321 // 0.929	1.512 // 5.611	100 // 100
	kmeans	1.793 // 12.8799	0.3527 // 1.6839	1.4344 // 11.155	0.062 // 0.24	0.593 // 2.177	2.547 // 9.595	100 // 100
	rebmix	6.9275 // 23.0817	2.7461 // 5.4713	4.0985 // 17.4511	0.044 // 0.32	3.177 // 3.836	8.535 // 15.437	96 // 70
HDclassif	hc	11.4938 // 49.4328	9.1746 // 12.2155	2.2913 // 36.5886	0.027 // 0.91	8.899 // 9.56	1.98 // 19.55	100 // 100
	kmeans	11.1438 // 40.4749	9.0384 // 11.9946	2.0912 // 28.0385	0.096 // 0.7	9.059 // 9.024	1.682 // 16.35	100 // 100
	rebmix	14.6998 // 47.2364	8.7649 // 12.6715	5.8029 // 33.929	0.22 // 0.92	8.135 // 9.145	8.018 // 21.824	96 // 70
EMMIXmfa	hc	5.6809 // 21.1181	3.7272 // 6.1206	1.7452 // 14.9126	0.41 // 0.019	5.772 // 3.645	4.299 // 12.812	96 // 45
	kmeans	5.7063 // 21.3775	3.6759 // 6.589	1.79 // 14.5681	0.39 // 0.17	5.788 // 4.08	4.357 // 13.352	96 // 40
	rebmix	5.8175 // 19.9703	3.8142 // 6.3202	1.7592 // 13.5389	0.35 // 0.033	5.819 // 4.402	4.349 // 13.812	93 // 34

Table 19: Minimal example setting apart MSE and Bias whether it proceeds from diagonal or offset terms of the covariance matrix, for scenarios HD5a) and HD6a), in Table 15 (overlapping and spherical-distributed components). We delimite each scenario by doubled backslashes with respectively balanced and unbalanced clusters.

Package	Initialisation Method	Global MSE diag(Σ)	Global MSE upper.tri(Σ)	Global Bias diag(Σ)	Global Bias upper.tri(Σ)
mixtools / Rmixmod / RGMMBench	hc	1.1 // 5.9	2.2 // 12	0.9194 // 2.3003	2.651 // 2.081
	kmeans	0.99 // 5.6	2.1 // 10	0.8929 // 2.7422	2.13 // 2.292
mclust / flexmix / GMKMcharlie	hc	0.76 // 5.5	1.6 // 11	0.5698 // 2.418	1.557 // 1.994
	kmeans	0.67 // 5.2	1.4 // 8.5	0.6909 // 3.4316	0.979 // 2.37
bgmm	hc	0.67 // 11	0.94 // 14	0.7755 // 6.9204	1.205 // 3.849
	kmeans	0.58 // 9.1	0.88 // 12	0.6004 // 6.124	1.138 // 3.659
EMCluster	hc	0.62 // 6.1	1.4 // 9.5	0.4985 // 3.4685	1.013 // 2.143
	kmeans	0.48 // 7.5	0.95 // 3.6	0.9269 // 7.6352	1.621 // 1.96
HDclassif	hc	0.72 // 26	1.6 // 11	0.5383 // 17.2225	1.441 // 2.328
	kmeans	0.68 // 20	1.4 // 8.5	0.7156 // 13.7249	0.966 // 2.626
EMMIXmfa	hc	1.6 // 10	0.13 // 4.6	3.7632 // 10.0798	0.536 // 2.733
	kmeans	1.6 // 11	0.17 // 3.9	3.7621 // 10.8057	0.594 // 2.546

Table 20: MSE and Bias associated to scenarios HD1a) and HD1b), in Table 15 (well-separated and spherical-distributed components). We delimite by doubled backslashes for each entry of the summary metrics table respectively the scores with $n = 200$ and $n = 2000$ observations.

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ	% Success
mixtools / Rmixmod / RGMMBench	hc	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0097 // 0.00071	0.053 // 0.018	0.139 // 0.04	100 // 100
	kmeans	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0097 // 0.00071	0.053 // 0.018	0.139 // 0.04	100 // 100
	rebmix	0.5611 // 0.0058	0.2364 // 0.0028	0.3035 // 0.0026	0.019 // 0.00071	0.372 // 0.018	0.915 // 0.04	98 // 100
mclust / flexmix / GMKMcharlie	hc	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0095 // 0.00071	0.053 // 0.018	0.14 // 0.04	100 // 100
	kmeans	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0095 // 0.00071	0.053 // 0.018	0.14 // 0.04	100 // 100
	rebmix	0.3134 // 0.0058	0.1305 // 0.0029	0.1729 // 0.0026	0.0039 // 0.0022	0.2 // 0.02	0.537 // 0.044	88 // 81
bgmm	hc	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0098 // 0.00071	0.053 // 0.018	0.139 // 0.041	100 // 100
	kmeans	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0097 // 0.00071	0.053 // 0.018	0.139 // 0.04	100 // 100
	rebmix	0.7437 // 0.1977	0.3409 // 0.0028	0.3895 // 0.1946	0.02 // 0.00034	0.308 // 0.017	1.602 // 0.926	97 // 99
EMCluster	hc	0.0577 // 0.0058	0.0289 // 0.0028	0.0264 // 0.0026	0.0093 // 0.00061	0.054 // 0.018	0.139 // 0.041	100 // 100
	kmeans	0.0577 // 0.0058	0.0288 // 0.0028	0.0264 // 0.0026	0.0092 // 0.00039	0.054 // 0.017	0.139 // 0.04	100 // 100
	rebmix	0.6887 // 0.3391	0.2466 // 0.1276	0.4201 // 0.1969	0.019 // 0.048	0.519 // 0.289	1.721 // 0.875	87 // 81
HDclassif	hc	11.3787 // 11.3322	11.3499 // 11.3293	0.0264 // 0.0026	0.0094 // 0.00071	11.166 // 11.179	0.139 // 0.04	100 // 100
	kmeans	11.3831 // 11.3301	11.3543 // 11.3271	0.0264 // 0.0026	0.0094 // 0.00068	11.162 // 11.181	0.139 // 0.04	100 // 100
	rebmix	11.5949 // 11.6085	11.5167 // 11.6055	0.072 // 0.0026	0.0024 // 0.0022	11.227 // 11.369	0.27 // 0.044	87 // 81
EMMIXmfa	hc	5.9739 // 5.9149	4.0397 // 3.9727	1.8288 // 1.8228	0.32 // 0.47	6.999 // 7.042	4.25 // 4.296	100 // 100
	kmeans	5.972 // 5.8863	4.0431 // 3.9596	1.8283 // 1.8259	0.33 // 0.43	6.997 // 7.051	4.255 // 4.34	100 // 100
	rebmix	5.9835 // 5.9078	4.0477 // 3.9671	1.8257 // 1.8244	0.37 // 0.46	6.994 // 7.045	4.232 // 4.301	87 // 81

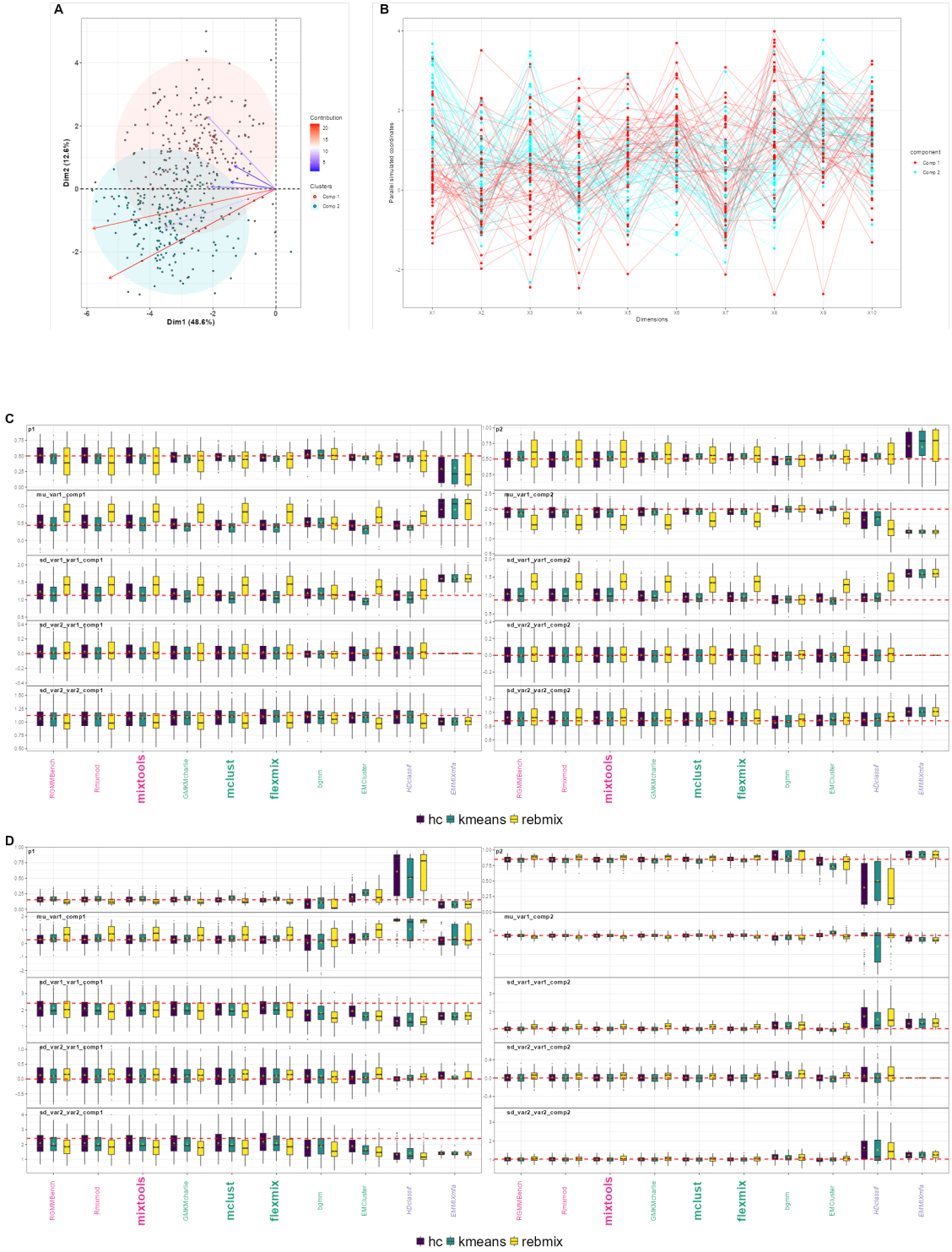


Figure 21: We gathered on the same plot two multivariate benchmark scenarios, in which we consider a strictly spherical structure of the covariance matrix: We represent in Panel A and B, respectively the bivariate projection and parallel distribution plot, associated to scenario HD5a) in Table 15 (balanced and overlapping components, with spherical covariance structure). In Panel C, we display the boxplots associated to scenario HD5a), computing them similarly as in Panel E of Figure 19. In Panel D, we display the boxplots associated to scenario HD6a) (unbalanced and overlapping components, with spherical covariance structure).

Table 21: MSE and Bias associated to scenarios HD8a) and HD8b), in Table 15 (overlapping components with full covariance structure). We delimite by doubled backslashes for each entry of the summary metrics table respectively the scores with $n = 200$ and $n = 2000$ observations.

Package	Initialisation Method	Global MSE p	Global MSE μ	Global MSE σ	Global Bias p	Global Bias μ	Global Bias σ	% Success
mixtools / Rmixmod / RGMMBench	hc	18.6085 // 0.6735	3.566 // 0.0536	14.9495 // 0.6193	0.23 // 0.0017	3.327 // 0.107	14.475 // 0.649	100 // 100
	kmeans	16.7452 // 0.6735	2.9065 // 0.0536	13.7662 // 0.6193	0.2 // 0.0016	2.819 // 0.107	12.552 // 0.649	100 // 100
	rebmix	22.2986 // 0.6738	4.3127 // 0.0536	17.8418 // 0.6196	0.25 // 0.0021	3.768 // 0.108	16.249 // 0.648	95 // 100
mclust / flexmix / GMKMcharlie	hc	20.6916 // 0.728	4.5672 // 0.0696	16.0328 // 0.6557	0.28 // 0.064	4.381 // 0.459	18.656 // 2.07	100 // 100
	kmeans	17.9622 // 0.7169	3.7547 // 0.0671	14.1405 // 0.6474	0.27 // 0.062	3.88 // 0.465	16.802 // 2.049	100 // 100
	rebmix	22.4636 // 0.7553	4.7502 // 0.0678	17.5784 // 0.6853	0.26 // 0.0054	4.165 // 0.158	17.735 // 0.725	94 // 98
bgmm	hc	35.6085 // 13.8411	12.8826 // 3.6502	22.3428 // 10.0718	0.29 // 0.46	6.212 // 5.661	26.812 // 23.753	100 // 100
	kmeans	33.8007 // 12.5545	11.7236 // 3.1419	21.7292 // 9.2934	0.28 // 0.47	6.348 // 5.654	26.546 // 23.141	100 // 100
	rebmix	35.3167 // 13.106	12.2374 // 3.3747	22.6615 // 9.6213	0.37 // 0.42	6.007 // 5.273	26.287 // 23.02	96 // 100
EMCluster	hc	23.4472 // 16.4192	6.2451 // 4.9191	17.0777 // 11.3124	0.35 // 0.51	5.469 // 6.279	23.503 // 22.821	99 // 100
	kmeans	21.0058 // 14.6852	5.9951 // 4.1684	14.9329 // 10.4293	0.38 // 0.42	5.604 // 6.592	24.628 // 24.706	100 // 100
	rebmix	23.0408 // 19.7372	6.4923 // 7	16.3824 // 12.5099	0.36 // 0.35	5.272 // 5.454	23.419 // 23.9	93 // 98
HDclassif	hc	36.5924 // 33.4077	16.108 // 14.4007	20.3809 // 18.8638	0.3 // 0.44	12.706 // 13.085	25.393 // 29.363	100 // 100
	kmeans	34.7935 // 30.1935	15.4329 // 13.78	19.2529 // 16.2816	0.4 // 0.41	12.766 // 12.756	24.988 // 25.151	100 // 100
	rebmix	38.9707 // 24.0327	16.134 // 12.9266	22.6961 // 11.0138	0.25 // 0.21	12.811 // 12.275	25.79 // 15.996	95 // 98

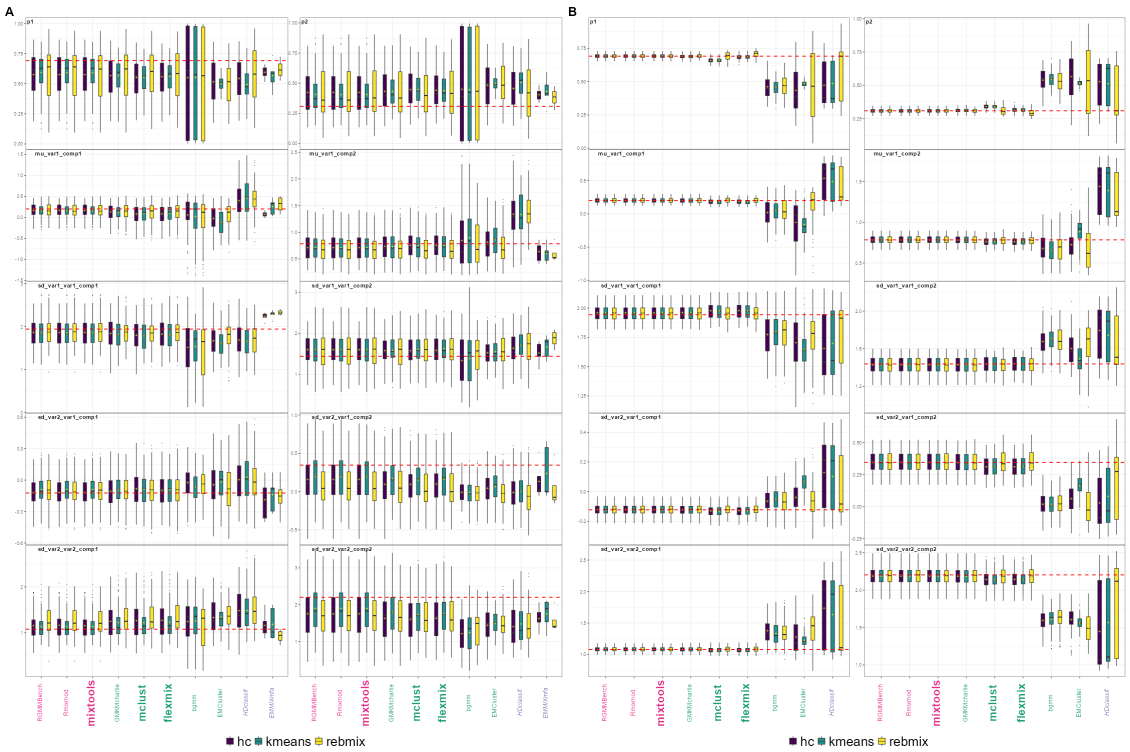


Figure 22: Overview of scenarios HD1 a) and b) and HD8 a) and b) in Table 15comparing the performance of the algorithms in respectively the easiest and most complex scenario. The left-hand column shows box plots of the estimated parameters from simulations with $n = 200$ observations on the left and $n = 2000$ observations on the right.

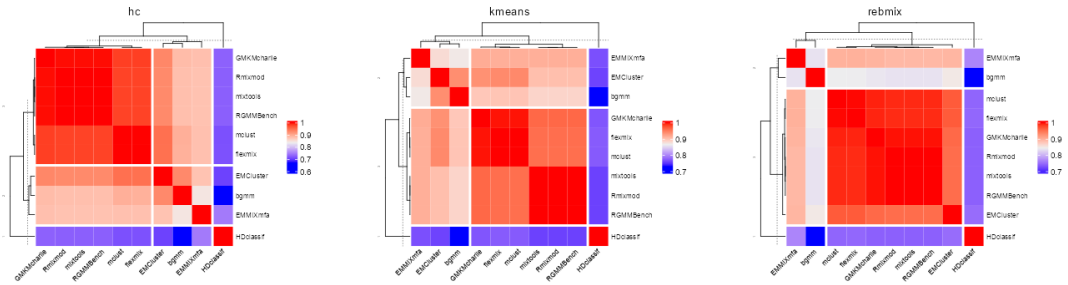


Figure 23: Correlation heatmaps of the estimated parameters in the high dimensional (HD) setting extended to the three initialisation methods benchmarked (respectively hc, k-means and rebmix) in the most discriminating scenario HD8a), using the same process described in Figure (2).

Supplementary bibliography

Alberich-Carramiñana, Maria, Borja Elizalde, and Federico Thomas. 2017. "New Algebraic Conditions for the Identification of the Relative Position of Two Coplanar Ellipses." *Computer Aided Geometric Design*. https://doi.org/10.1016/j.cagd.2017.03.013.

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2023. *rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Auguie, Baptiste. 2017. *gridExtra: Miscellaneous Functions for "Grid" Graphics*. https://CRAN.R-project.org/package=gridExtra.

Avila, Luis M., Michael R. May, and Jeff Ross-Ibarra. 2018. *DPP: Inference of Parameters of Normal Distributions from a Mixture of Normals*.

Bacci, Silvia, Silvia Pandolfi, and Fulvia Pennoni. 2012. "A Comparison of Some Criteria for States Selection in the Latent Markov Model for Longitudinal Data." http://arxiv.org/abs/1212.0352.

Baker, Peter. 2018. *polySegratioMM: Bayesian Mixture Models for Marker Dosage in Autopolyploids*.

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

Basford, K., D. Greenway, G. Mclachlan, et al. 1997. "Standard Errors of Fitted Component Means of Normal Mixtures." *Computational Statistics*. https://www.researchgate.net/publication/37625647/_Standard/_errors/_of/_fitted/_component/_means/_of/_normal/_mixtures.

Benaglia, Tatiana, Didier Chauveau, David R. Hunter, et al. 2009. "mixtools: An R Package for Analyzing Finite Mixture Models." *Journal of Statistical Software*.

Bergé, Laurent, Charles Bouveyron, and Stéphane Girard. 2012. "HDclassif: An R Package for Model-Based Clustering and Discriminant Analysis of High-Dimensional Data." *Journal of Statistical Software*. https://doi.org/10.18637/jss.v046.i06.

Biernacki, Christophe, Gilles Celeux, and Gerard Govaert. 2000. "Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood." *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. https://doi.org/10.1109/34.865189.

Bobb, Jennifer F., and Ravi Varadhan. 2021. *turboEM: A Suite of Convergence Acceleration Schemes for EM, MM and Other Fixed-Point Algorithms*.

Bouveyron, Charles, Gilles Celeux, and Stéphane Girard. 2011. "Intrinsic Dimension Estimation by Maximum Likelihood in Isotropic Probabilistic PCA." *Pattern Recognition Letters*. https://doi.org/10.1016/j.patrec.2011.07.017.

Bouveyron, Charles, Stéphane Girard, and Cordelia SCHMID. 2007. "High-Dimensional Data Clustering." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/j.csda.2007.02.009.

Browne, Ryan P., and Paul D. McNicholas. 2014. "Estimating Common Principal Components in High Dimensions." *Advances in Data Analysis and Classification*. https://doi.org/10.1007/s11634-013-0139-1.

Cao, Sha, Wennan Chang, and Chi Zhang. 2020. *RobMixReg: Robust Mixture Regression*.

Cao, X., and J. C. Spall. 2012. "Relative Performance of Expected and Observed Fisher Information in Covariance Estimation for Maximum Likelihood Estimates." https://doi.org/10.1109/ACC.2012.6315584.

Cattell, Raymond B. 1966. "The Scree Test For The Number Of Factors." *Multivariate Behavioral Research*. https://doi.org/10.1207/s15327906mbr0102/_10.

Celeux, Gilles, Stéphane Chrétien, and Florence Forbes. 2012. "A Component-wise EM Algorithm for Mixtures."

Celeux, Gilles, Sylvia Fruewirth-Schnatter, and Christian P. Robert. 2018. "Model Selection for Mixture Models - Perspectives and Strategies." arXiv. https://doi.org/10.48550/ARXIV.1812.09885.

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

Chen, J., and Z. Chen. 2008. "Extended Bayesian Information Criteria for Model Selection with Large Model Spaces." *Biometrika*. https://doi.org/10.1093/biomet/asn034.

Chen, Wei-Chen, and George Ostrouchov. 2021. *Pmclust: Parallel Model-Based Clustering Using Expectation-Gathering-Maximization Algorithm for Finite Mixture Gaussian Model*.

Clark, Katharine M., and Paul D. McNicholas. 2019. *Oclust: Gaussian Model-Based Clustering with Outliers*.

Coretto, Pietro, and Christian Hennig. 2016. "Consistency, Breakdown Robustness, and Algorithms for Robust Improper Maximum Likelihood Clustering." *Journal of Machine Learning Research*.

———. 2021. *Otrimle: Robust Model-Based Clustering.*

Csárdi, Gábor. 2019. *Cranlogs: Download Logs from the RStudio 'CRAN' Mirror.* https://CRAN.R-project.org/package=cranlogs.

Dai, Ming, and Arunava Mukherjea. 2001. "Identification of the Parameters of a Multivariate Normal Vector by the Distribution of the Maximum." *Journal of Theoretical Probability.* https://doi.org/10.1023/A:1007889519309.

Dean, Nema, Adrian E. Raftery, and Luca Scrucca. 2020. *Clustvarsel: Variable Selection for Gaussian Model-Based Clustering.*

Delattre, Maud, and Estelle Kuhn. 2019. "Estimating Fisher Information Matrix in Latent Variable Models Based on the Score Function." https://doi.org/https://doi.org/10.48550/arXiv.1909.06094.

Efron, Bradley, and Robert Tibshirani. 1993. *An Introduction to the Bootstrap.*

Fallah, Lida, and John Hinde. 2017. *Pcensmix: Model Fitting to Progressively Censored Mixture Data.*

Figueiredo, M. A. T., and A. K. Jain. 2002. "Unsupervised Learning of Finite Mixture Models." *IEEE Transactions on Pattern Analysis and Machine Intelligence.* https://doi.org/10.1109/34.990138.

Fonseca, Jaime. 2008. "The Application of Mixture Modeling and Information Criteria for Discovering Patterns of Coronary Heart Disease." *Journal of Applied Quantitative Methods.* https://doi.org/10.1109/EMS.2008.36.

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

Franczak, Brian C., Ryan P. Browne, and Paul D. McNicholas. 2016. *Sensory: Simultaneous Model-Based Clustering and Imputation via a Progressive Expectation-Maximization Algorithm.*

Gallegos, María Teresa, and Gunter Ritter. 2005. "A Robust Method for Cluster Analysis." *The Annals of Statistics.* https://doi.org/10.1214/009053604000000940.

Garay, Aldo M., Monique Bettio Massuia, and Victor Lachos. 2022. *SMNCensReg: Fitting Univariate Censored Regression Model Under the Family of Scale Mixture of Normal Distributions.*

Garay, Aldo M., Monique B. Massuia, Victor H. Lachos, et al. 2017. *BayesCR: Bayesian Analysis of Censored Regression Models Under Scale Mixture of Skew Normal Distributions.*

García-Escudero, Luis A., Alfonso Gordaliza, Carlos Matrán, et al. 2008. "A General Trimming Approach to Robust Cluster Analysis." *The Annals of Statistics.* https://doi.org/10.1214/07-AOS515.

Gohel, David, and Panagiotis Skintzos. 2023. *Flextable: Functions for Tabular Reporting.* https://CRAN.R-project.org/package=flextable.

Gu, Zuguang. 2022. "Complex Heatmap Visualization." *iMeta.* https://doi.org/10.1002/imt2.43.

Gu, Zuguang, Roland Eils, and Matthias Schlesner. 2016. "Complex Heatmaps Reveal Patterns and Correlations in Multidimensional Genomic Data." *Bioinformatics.* https://doi.org/10.1093/bioinformatics/btw313.

He, Yawei, and Zehua Chen. 2016. "The EBIC and a Sequential Procedure for Feature Selection in Interactive Linear Models with High-Dimensional Data." *Annals of the Institute of Statistical Mathematics.* https://doi.org/10.1007/s10463-014-0497-2.

Hester, Jim, and Jennifer Bryan. 2022. *glue: Interpreted String Literals.* https://CRAN.R-project.org/package=glue.

Houseman, E. Andres, Sc.D., Devin C. Koestler, et al. 2017. *RPMM: Recursively Partitioned Mixture Model.*

Iovleff, Serge. 2019. *MixAll: Clustering and Classification Using Model-Based Mixture Models.*

Iscar, Agustin Mayo, Luis Angel Garcia Escudero, and Heinrich Fritz. 2022. *Tclust: Robust Trimmed Clustering.*

Jaki, Thomas, Ting-Li Su, Minjung Kim, et al. 2018. "An Evaluation of the Bootstrap for Model Validation in Mixture Models." *Communications in Statistics: Simulation and Computation.* https://doi.org/10.1080/03610918.2017.1303726.

Jones, Andrew Thomas. 2020. *EMMIXgene: A Mixture Model-Based Approach to the Clustering of Microarray Expression Data.*

Jones, Andrew T., and Hien D. Nguyen. 2019. *SAGMM: Clustering via Stochastic Approximation and Gaussian Mixture Models.*

Kapourani, C. A. 2022. *Melissa: Bayesian Clustering and Imputationa of Single Cell Methylomes.*

Klein, Hans-Ulrich, and Martin Schaefer. 2022. *Epigenomix: Epigenetic and Gene Transcription Data Normalization and Integration with Mixture Models.*

Komárek, Arnošt. 2022. *mixAK: Multivariate Normal Mixture Models and Mixtures of Generalized Linear Mixed*

Models Including Model Based Clustering.

Kubicki, Vincent, Christophe Biernacki, and Quentin Grimonprez. 2021. *RMixtComp: Mixture Models with Heterogeneous and (Partially) Missing Data*.

Kurban, Hasan, Mark Jenne, and Mehmet M. Dalkilic. 2017. "Using Data to Build a Better EM: EM∗for Big Data." *International Journal of Data Science and Analytics*. https://doi.org/10.1007/s41060-017-0062-1.

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

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

Li, Ker-Chau. 1991. "Sliced Inverse Regression for Dimension Reduction." *Journal of the American Statistical Association*. https://doi.org/10.2307/2290563.

Li, Xuan, Yuejiao Fu, Xiaogang Wang, et al. 2018. *MMDvariance: Detecting Differentially Variable Genes Using the Mixture of Marginal Distributions*.

Li, Yan, and Kun Chen. 2021. *fmerPack: Tools of Heterogeneity Pursuit via Finite Mixture Effects Model*.

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

Louis, Thomas A. 1982. "Finding the Observed Information Matrix When Using the EM Algorithm." *Journal of the Royal Statistical Society*. https://doi.org/10.1111/j.2517-6161.1982.tb01203.x.

Lu, Xiang, Yaoxiang Li, and Tanzy Love. 2022. *Bpgmm: Bayesian Model Selection Approach for Parsimonious Gaussian Mixture Models*.

Macdonald, Peter, and with contributions from Juan Du. 2018. *Mixdist: Finite Mixture Distribution Models*.

Maitra, Ranjan, and Volodymyr Melnykov. 2010. "Simulating Data to Study Performance of Finite Mixture Modeling and Clustering Algorithms." *Journal of Computational and Graphical Statistics*. https://doi.org/10.1198/jcgs.2009.08054.

Marandon, Ariane, Tabea Rebafka, Etienne Roquain, et al. 2022. "False Clustering Rate Control in Mixture Models." arXiv. https://doi.org/10.48550/ARXIV.2203.02597.

McCaw, Zachary. 2021. *MGMM: Missingness Aware Gaussian Mixture Models*.

McLachlan, G. J., D. Peel, and R. W. Bean. 2003. "Modelling High-Dimensional Data by Mixtures of Factor Analyzers." *Computational Statistics & Data Analysis*, Recent Developments in Mixture Model,. https://doi.org/10.1016/S0167-9473(02)00183-4.

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

Mclachlan, G., and David Peel. 2000. "Mixtures of Factor Analyzers." In. https://doi.org/10.48550/arXiv.1507.02801.

McNicholas, P. D., T. B. Murphy, A. F. McDaid, et al. 2010. "Serial and Parallel Implementations of Model-Based Clustering via Parsimonious Gaussian Mixture Models." *Computational Statistics & Data Analysis*. https://doi.org/10.1016/j.csda.2009.02.011.

McNicholas, Paul David, and Thomas Brendan Murphy. 2008. "Parsimonious Gaussian Mixture Models." *Statistics and Computing*. https://doi.org/10.1007/s11222-008-9056-0.

McNicholas, Paul D., Aisha ElSherbiny, Aaron F. McDaid, et al. 2022. *Pgmm: Parsimonious Gaussian Mixture Models*.

McNicholas, Paul D., and Thomas Brendan Murphy. 2010. "Model-Based Clustering of Microarray Expression Data via Latent Gaussian Mixture Models." *Bioinformatics*. https://doi.org/10.1093/bioinformatics/btq498.

McParland, Damien, and Isobel Claire Gormley. 2017. *clustMD: Model Based Clustering for Mixed Data*.

Meng, Lingyao. 2016. "Method for Computation of the Fisher Information Matrix in the Expectation-Maximization Algorithm." arXiv. https://doi.org/10.48550/ARXIV.1608.01734.

MENG, XIAO-LI, and Donald Rubin. 1993. "Maximum Likelihood Estimation via the ECM Algorithm: A General Framework." *Biometrika*. https://doi.org/10.1093/biomet/80.2.267.

Meng, Xiao-Li, and David Van Dyk. 1997. "The EM Algorithm–an Old Folk-Song Sung to a Fast New Tune." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. https://doi.org/10.1111/1467-9868.00082.

Mersmann, Olaf. 2021. *Microbenchmark: Accurate Timing Functions*.

Mohammadi, Reza. 2021. *Bmixture: Bayesian Estimation for Finite Mixture of Distributions*.

Mouselimis, Lampros. 2022. *ClusterR: Gaussian Mixture Models, k-Means, Mini-Batch-Kmeans, k-Medoids and Affinity Propagation Clustering*.

Müller, Kirill, and Hadley Wickham. 2023. *Tibble: Simple Data Frames*. https://CRAN.R-project.org/package=tibble.

Murphy, Kevin. 2012. *Machine Learning A Probabilistic Perspective*. Adaptive Computation; Machine Learning series. https://doi.org/10.1080/09332480.2014.914768.

Neuwirth, Erich. 2022. *RColorBrewer: ColorBrewer Palettes*. https://CRAN.R-project.org/package=RColorBrewer.

Nowakowska, Ewa, Jacek Koronacki, and Stan Lipovetsky. 2014. "Tractable Measure of Component Overlap for Gaussian Mixture Models." https://doi.org/10.48550/arXiv.1407.7172.

O'Hara-Wild, Mitchell, Stephanie Kobakian, H. Sherry Zhang, Di Cook, Simon Urbanek, and Christophe Dervieux. 2023. *rjtools: Preparing, Checking, and Submitting Articles to the "R Journal"*. https://CRAN.R-project.org/package=rjtools.

Oakes, D. 1999. "Direct Calculation of the Information Matrix via the EM." *Journal of the Royal Statistical Society*. https://doi.org/10.1111/1467-9868.00188.

Papastamoulis, Panagiotis. 2020. *fabMix: Overfitting Bayesian Mixtures of Factor Analyzers with Parsimonious Covariance and Unknown Number of Components*.

Pastore, Massimiliano, and Antonio Calcagnì. 2019. "Measuring Distribution Similarities Between Samples: A Distribution-Free Overlapping Index." *Frontiers in Psychology*. https://doi.org/10.3389/fpsyg.2019.01089.

Petersen, K. B., and M. S. Pedersen. 2008. "The Matrix Cookbook." Technical University of Denmark.

Pocuca, Nik, Ryan P. Browne, and Paul D. McNicholas. 2022. *Mixture: Mixture Models for Clustering and Classification*.

Prates, Marcos, Victor Lachos, and Celso Cabral. 2021. *Mixsmsn: Fitting Finite Mixture of Scale Mixture of Skew-Normal Distributions*.

Prates, Marcos, Victor Lachos, and Aldo Garay. 2021. *Nlsmsn: Fitting Nonlinear Models with Scale Mixture of Skew-Normal Distributions*.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Rau, Andrea. 2022. *Coseq: Co-Expression Analysis of Sequencing Data*.

Rauschenberger, Armin. 2022. *Semisup: Semi-Supervised Mixture Model*.

Redner, Richard A., and Homer F. Walker. 1984. "Mixture Densities, Maximum Likelihood and the Em Algorithm." *SIAM Review*. https://doi.org/239https://doi.org/10.1137/1026034.

Robert, Christian, and George Casella. 2010. *Introducing Monte Carlo Methods with R*. Springer. https://doi.org/10.1007/978-1-4419-1576-4.

Schlattmann, Peter, Johannes Hoehne, and Maryna Verba. 2022. *CAMAN: Finite Mixture Models and Meta-Analysis Tools - Based on c.a.MAN*.

Schwarz, Gideon. 1978. "Estimating the Dimension of a Model." *The Annals of Statistics*. https://doi.org/10.1214/aos/1176344136.

Scrucca, Luca. 2010. "Dimension Reduction for Model-Based Clustering." *Statistics and Computing*. https://doi.org/10.1007/s11222-009-9138-7.

Scrucca, Luca, Michael Fop, T. Brendan Murphy, et al. 2016. "Mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models." *The R Journal*. https://doi.org/10.32614/RJ-2016-021.

Shokoohi, Farhad. 2022. *Fmrs: Variable Selection in Finite Mixture of AFT Regression and FMR Models*.

Thrun, Michael, Onno Hansen-Goos, and Alfred Ultsch. 2020. *AdaptGauss: Gaussian Mixture Models (GMM)*.

Tipping, Michael E., and Christopher M. Bishop. 1999. "Probabilistic Principal Component Analysis." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. https://doi.org/10.1111/1467-9868.00196.

Turner, Rolf. 2021. *Mixreg: Functions to Fit Mixtures of Regressions*.

Viroli, Cinzia, and Geoffrey J. McLachlan. 2020. *Deepgmm: Deep Gaussian Mixture Models*.

Walsh, Gordon Raymond. 1975. *Methods of Optimization*. Wiley.

Wang, Ziqiao. 2022. *IMIX: Gaussian Mixture Model for Multi-Omics Data Integration*.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2022. *Stringr: Simple, Consistent Wrappers for Common String Operations*. https://CRAN.R-project.org/package=stringr.

———. 2023. *Forcats: Tools for Working with Categorical Variables (Factors)*. https://CRAN.R-project.org/package=forcats.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, and Dewey Dunnington. 2023. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://CRAN.R-project.org/package=ggplot2.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, and Lionel Henry. 2023. *Purrr: Functional Programming Tools*. https://CRAN.R-project.org/package=purrr.

Wickham, Hadley, Jim Hester, and Jennifer Bryan. 2023. *Readr: Read Rectangular Text Data*. https://CRAN.R-project.org/package=readr.

Wickham, Hadley, and Dana Seidel. 2022. *scales: Scale Functions for Visualization*. https://CRAN.R-project.org/package=scales.

Wilke, Claus O. 2020. *cowplot: Streamlined Plot Theme and Plot Annotations for "ggplot2"*. https://CRAN.R-project.org/package=cowplot.

Wilke, Claus O., and Brenton M. Wiernik. 2022. *ggtext: Improved Text Rendering Support for "ggplot2"*. https://CRAN.R-project.org/package=ggtext.

Xie, Yihui. 2015. *Dynamic Documents with R and Knitr*. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. https://yihui.org/knitr/.

———. 2023. *knitr: A General-Purpose Package for Dynamic Report Generation in r*. https://yihui.org/knitr/.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xu, Yanxun, Peter Mueller, Donatello Telesca, et al. 2020. *Dppmix: Determinantal Point Process Mixture Models*.

Yang, Yuhong. 2005. "Can the Strengths of AIC and BIC Be Shared? A Conflict Between Model Identification and Regression Estimation." *Biometrika*. https://doi.org/10.1093/biomet/92.4.937.

Yu, Youjiao. 2021. *mixR: Finite Mixture Modeling for Raw and Binned Data*.

Zhu, Hao. 2021. *kableExtra: Construct Complex Table with Kable and Pipe Syntax*. https://CRAN.R-project.org/package=kableExtra.