Statistics and Computing (2020) 30:485–506
https://doi.org/10.1007/s11222-019-09891-z

Clustering multivariate data using factor analytic Bayesian mixtures with an unknown number of components

Panagiotis Papastamoulis[1]

Received: 15 December 2018 / Accepted: 16 August 2019 / Published online: 27 August 2019

Abstract
Recent work on overfitting Bayesian mixtures of distributions offers a powerful framework for clustering multivariate data using a latent Gaussian model which resembles the factor analysis model. The flexibility provided by overfitting mixture models yields a simple and efficient way in order to estimate the unknown number of clusters and model parameters by Markov chain Monte Carlo sampling. The present study extends this approach by considering a set of eight parameterizations, giving rise to parsimonious representations of the covariance matrix per cluster. A Gibbs sampler combined with a prior parallel tempering scheme is implemented in order to approximately sample from the posterior distribution of the overfitting mixture. The parameterization and number of factors are selected according to the Bayesian information criterion. Identifiability issues related to label switching are dealt by post-processing the simulated output with the Equivalence Classes Representatives algorithm. The contributed method and software are demonstrated and compared to similar models estimated using the expectation–maximization algorithm on simulated and real datasets. The software is available online at https://CRAN.R-project.org/package=fabMix.

Keywords Mixture model · Factor analysis · MCMC · R package

1 Introduction

Factor analysis (FA) explains relationships among a set of observed variables using a set of latent variables. This is typically achieved by expressing the observed multivariate data as a linear combination of a smaller set of unobserved and uncorrelated variables known as factors. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ denote a random sample of p-dimensional observations with $\boldsymbol{x}_i \in \mathbb{R}^p$; $i = 1, \ldots, n$. Let $\mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denote the p-dimensional normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ and also denote by $\mathbf{I}_p$ the $p \times p$ identity matrix. The following equations summarize the typical FA model.

$$\boldsymbol{x}_i = \boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{y}_i + \boldsymbol{\varepsilon}_i, \quad i = 1, \ldots, n \tag{1}$$
$$(\boldsymbol{y}_i, \boldsymbol{\varepsilon}_i) \sim \mathcal{N}_q(\mathbf{0}, \mathbf{I}_q)\mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma}), \quad \text{iid for } i = 1, \ldots, n \tag{2}$$
$$\boldsymbol{\Sigma} = \text{diag}(\sigma_1^2, \ldots, \sigma_p^2) \tag{3}$$
$$\boldsymbol{x}_i | \boldsymbol{y}_i \sim \mathcal{N}_p(\boldsymbol{\mu} + \boldsymbol{\Lambda}\boldsymbol{y}_i, \boldsymbol{\Sigma}), \quad \text{ind. for } i = 1, \ldots, n \tag{4}$$

✉ Panagiotis Papastamoulis
papastamoulis@aueb.gr

[1] Department of Statistics, Athens University of Economics and Business, 76 Patission Street, 10434 Athens, Greece

Before proceeding, note that we are not differentiating the notation between random variables and their corresponding realizations. Bold uppercase letters are used for matrices; bold lowercase letters are used for vectors and normal text for scalars.

In Eq. (1), we assume that $\boldsymbol{x}_i$ is expressed as a linear combination of a latent vector of factors $\boldsymbol{y}_i \in \mathbb{R}^q$. The $p \times q$ dimensional matrix $\boldsymbol{\Lambda} = (\lambda_{rj})$ contains the factor loadings, while $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_p)$ contains the marginal mean of $\boldsymbol{x}_i$. The unobserved vector $\boldsymbol{y}_i$ lies on a lower-dimensional space, that is, $q < p$, and it consists of uncorrelated features $y_{i1}, \ldots, y_{iq}$ as shown in Eq. (2), where $\mathbf{0}$ denotes a vector of zeros. Note that the error terms $\boldsymbol{\varepsilon}_i$ are independent from $\boldsymbol{y}_i$. Furthermore, the errors are consisting of independent random variables $\varepsilon_{i1}, \ldots, \varepsilon_{ip}$, as implied by the diagonal covariance matrix $\boldsymbol{\Sigma}$ in Eq. (3). As shown in Eq. (4), the knowledge of the missing data $(\boldsymbol{y}_i)$ implies that the conditional distribution of $\boldsymbol{x}_i$ has a diagonal covariance matrix. The previous assumptions lead to

$$\boldsymbol{x}_i \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\Sigma}), \quad \text{iid for } i = 1, \ldots, n. \tag{5}$$

According to Eq. (5), the covariance matrix of the marginal distribution of $\boldsymbol{x}_i$ is equal to $\boldsymbol{\Lambda}\boldsymbol{\Lambda}^T + \boldsymbol{\Sigma}$. This is the crucial characteristic of factor analytic models, where they aim to explain high-dimensional dependencies using a set of lower-dimensional uncorrelated factors (Kim and Mueller 1978; Bartholomew et al. 2011).

Mixtures of Factor Analyzers (MFA) are generalizations of the typical FA model, by assuming that Eq. (5) becomes

$$\boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Sigma}_k), \text{ iid } i = 1, \ldots, n \qquad (6)$$

where K denotes the number of mixture components. The vector of mixing proportions $\boldsymbol{w} := (w_1, \ldots, w_K)$ contains the weight of each component, with $0 \leqslant w_k \leqslant 1$; $k = 1, \ldots, K$ and $\sum_{k=1}^{K} w_k = 1$. Note that the mixture components are characterized by different parameters $\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k, \boldsymbol{\Sigma}_k, k = 1, \ldots, K$. Thus, MFAs are particularly useful when the observed data exhibit unusual characteristics such as heterogeneity. That being said, this approach aims to capture the behavior of each cluster within a component of the mixture model. A comprehensive perspective on the history and development of MFA models is given in Chapter 3 of the monograph by McNicholas (2016).

Early works applying the expectation–maximization (EM) algorithm (Dempster et al. 1977) for estimating MFA are the ones from Ghahramani et al. (1996), Tipping and Bishop (1999), McLachlan and Peel (2000). McNicholas and Murphy (2008), McNicholas and Murphy (2010) introduced the family of parsimonious Gaussian mixture models (PGMMs) by considering the case where the factor loadings and/or error variance may be shared or not between the mixture components. These models are estimated by the alternating expectation–conditional maximization algorithm (Meng and Van Dyk 1997) and have superior performance compared to other approaches (McNicholas and Murphy 2008). Under a Bayesian setup, Fokoué and Titterington (2003) estimate the number of mixture components and factors by simulating a continuous-time stochastic birth–death point process using a birth–death MCMC algorithm (Stephens 2000). More recently, Papastamoulis (2018b) estimated Bayesian MFA models with an unknown number of components using overfitting mixtures.

In recent years, there is a growing progress on the usage of overfitting mixture models in Bayesian analysis (Rousseau and Mengersen 2011; van Havre et al. 2015; Malsiner Walli et al. 2016, 2017; Frühwirth-Schnatter and Malsiner-Walli 2019). An overfitting mixture model consists of a number of components which is much larger than its true (and unknown) value. Under suitable prior assumptions (see "Appendix A") introduced by Rousseau and Mengersen (2011), it has been shown that asymptotically the redundant components will have zero posterior weight and force the posterior distribution to put all its mass in the sparsest way to approximate the true density. Therefore, the inference on the number of mixture components can be based on the posterior distribution of the "alive" components of the overfitted model, that is, the components which contain at least one allocated observation.

Other Bayesian approaches to estimate the number of components in a mixture model include the reversible jump MCMC (RJMCMC) (Green 1995; Richardson and Green 1997; Dellaportas and Papageorgiou 2006; Papastamoulis and Iliopoulos 2009), birth–death MCMC (BDMCMC) (Stephens 2000) and allocation sampling (Nobile and Fearnside 2007; Papastamoulis and Rattray 2017) algorithms. However, overfitting mixture models are straightforward to implement, while the rest of the approaches require either careful design of various move types that bridge models with different number of clusters, or analytical integration of parameters.

The overall message is that there is a need for developing an efficient Bayesian method that will combine the previously mentioned frequentist advances on parsimonious representations of MFAs and the flexibility provided by the Bayesian viewpoint. This study aims at filling this gap by extending the Bayesian method of Papastamoulis (2018b) to the family of parsimonious Gaussian mixtures of McNicholas and Murphy (2008). Furthermore, we illustrate the proposed method using the `R` (Ihaka and Gentleman 1996; R Core Team 2016) package `fabMix` (Papastamoulis 2018a) available as a contributed package from the Comprehensive R Archive Network at https://CRAN.R-project.org/package=fabMix. The proposed method efficiently deals with many inferential problems (see, e.g., Celeux et al. (2000a)) related to mixture posterior distributions, such as (i) inferring the number of non-empty clusters using overfitting models, (ii) efficient exploration of the posterior surface by running parallel heated chains and (iii) incorporating advanced techniques that successfully deal with the label switching issue (Papastamoulis 2016).

The rest of the paper is organized as follows: Section 2 reviews the basic concepts of parsimonious MFAs. Identifiability problems and corresponding treatments are detailed in Sect. 2.1. The Bayesian model is introduced in Sect. 2.2. Section 3 presents the full conditional posterior distributions of the model. The MCMC algorithm is described in Sect. 3.2. A detailed presentation of the main function of the contributed `R` package is given in Sect. 4. Our method is illustrated and compared to similar models estimated by the EM algorithm in Sects. 5.1 and 5.2 using an extended simulation study and four publicly available datasets, respectively. We conclude in Sect. 6 with a summary of our findings and directions for further research. Appendix contains further discussion on overfitting mixture models ("Appendix A"), details of the

MCMC sampler ("Appendix B") and additional simulation results ("Appendix C").

2 Parsimonious mixtures of factor analyzers

Consider the latent allocation variables z_i which assign observation $\boldsymbol{x}_i$ to a component $k = 1, \ldots, K$ for $i = 1, \ldots, n$. A priori each observation is generated from component k with probability equal to w_k, that is,

$$\mathrm{P}(z_i = k) = w_k, \quad k = 1, \ldots, K, \tag{7}$$

independent for $i = 1, \ldots, n$. Note that the allocation vector $z := (z_1, \ldots, z_n)$ is not observed, so it should be treated as missing data. We assume that $\boldsymbol{z}_i$ and $\boldsymbol{y}_i$ are independent; thus, Eq. (2) is now written as:

$$(\boldsymbol{y}_i, \boldsymbol{\varepsilon}_i | z_i = k) \sim \mathcal{N}_q(\mathbf{0}, \mathbf{I}_q)\mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma}_k), \tag{8}$$

and conditional on the cluster membership and latent factors, we obtain that

$$(\boldsymbol{x}_i | z_i = k, \boldsymbol{y}_i) \sim \mathcal{N}_p(\boldsymbol{\mu}_k + \boldsymbol{\Lambda}_k \boldsymbol{y}_i, \boldsymbol{\Sigma}_k). \tag{9}$$

Consequently,

$$(\boldsymbol{x}_i | z_i = k) \sim \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Sigma}_k), \tag{10}$$

independent for $i = 1, \ldots, n$. From Eqs. (7) and (10), we derive that the marginal distribution of $\boldsymbol{x}_i$ is the finite mixture model in Eq. (6).

Following McNicholas and Murphy (2008), the factor loadings and error variance per component may be common or not among the K components in Eq. (6). If the factor is constrained, then:

$$\boldsymbol{\Lambda}_1 = \cdots = \boldsymbol{\Lambda}_K = \boldsymbol{\Lambda}. \tag{11}$$

If the error variance is constrained, then:

$$\boldsymbol{\Sigma}_1 = \cdots = \boldsymbol{\Sigma}_K = \boldsymbol{\Sigma}. \tag{12}$$

Furthermore, the error variance may be isotropic (i.e., proportional to the identity matrix) or not and depending on whether constraint (12) is disabled or enabled:

$$\boldsymbol{\Sigma}_k = \sigma_k^2 \mathbf{I}_p; k = 1, \ldots, K \quad \text{or} \tag{13}$$
$$\boldsymbol{\Sigma}_k = \sigma^2 \mathbf{I}_p; k = 1, \ldots, K. \tag{14}$$

We note that under constraint (13), the model is referred to as a mixture of probabilistic principal component analyzers (Tipping and Bishop 1999).

Depending on whether a particular constraint is present or not, the following set of eight parameterizations arises.

$$\text{UUU: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Sigma}_k)$$

$$\text{UCU: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Sigma})$$

$$\text{UUC: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \sigma_k^2 \mathbf{I}_p)$$

$$\text{UCC: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \sigma^2 \mathbf{I}_p)$$

$$\text{CUU: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda} \boldsymbol{\Lambda}^T + \boldsymbol{\Sigma}_k)$$

$$\text{CCU: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda} \boldsymbol{\Lambda}^T + \boldsymbol{\Sigma})$$

$$\text{CUC: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda} \boldsymbol{\Lambda}^T + \sigma_k^2 \mathbf{I}_p)$$

$$\text{CCC: } \boldsymbol{x}_i \sim \sum_{k=1}^{K} w_k \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Lambda} \boldsymbol{\Lambda}^T + \sigma^2 \mathbf{I}_p)$$

independent for $i = 1, \ldots, n$. Following the `pgmm` nomenclature (McNicholas and Murphy 2008): the first, second and third letters denote whether $\boldsymbol{\Lambda}_k$, $\boldsymbol{\Sigma}_k = \mathrm{diag}(\sigma_{k1}^2, \ldots, \sigma_{kp}^2)$ and σ_{kj}^2, $k = 1, \ldots, K$; $j = 1, \ldots, p$, are constrained (C) or unconstrained (U), respectively. A novelty of the present study is to offer a Bayesian framework for estimating the whole family of the previous parameterizations (note that Papastamoulis (2018b) estimated the UUU and UCU parameterizations).

2.1 Label switching and other identifiability problems

Let $L(\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\phi} | \boldsymbol{x}) = \prod_{i=1}^{n} \sum_{k=1}^{K} w_k f(x_i | \theta_k, \boldsymbol{\phi})$, $(\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\phi}) \in \mathcal{P}_{K-1} \times \Theta^K \times \Phi$ denote the likelihood function of a mixture of K densities, where $\mathcal{P}_{K-1}$ denotes the parameter space of the mixing proportions $\boldsymbol{w}$, $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$ are the component-specific parameters and $\boldsymbol{\phi}$ denotes a (possibly empty) collection of parameters that are common between all components. For instance, consider the UCU parameterization where $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k)$ for $k = 1, \ldots, K$ and $\boldsymbol{\phi} = \boldsymbol{\Sigma}$. For any permutation $\tau = (\tau_1, \ldots, \tau_K)$ of the set $\{1, \ldots, K\}$, the likelihood of mixture models is invariant to permutations of the component labels: $L(\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\phi} | \boldsymbol{x}) = L(\tau \boldsymbol{w}, \tau \boldsymbol{\theta}, \boldsymbol{\phi} | \boldsymbol{x})$. Thus, the likelihood surface of a mixture model with K components will exhibit $K!$ symmetric areas. If $(\boldsymbol{w}^*, \boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$

corresponds to a mode of the likelihood, the same will hold for any permutation $(\tau \boldsymbol{w}^*, \tau \boldsymbol{\theta}^*, \boldsymbol{\phi}^*)$.

Label switching (Redner and Walker 1984) is the commonly used term to describe this phenomenon. Under a Bayesian point of view, in the case that the prior distribution is also invariant to permutations (which is typically the case, see, e.g., Marin et al. (2005), Papastamoulis and Iliopoulos (2013)), the same invariance property will also hold for the posterior distribution $f(\boldsymbol{w}, \boldsymbol{\theta}, \boldsymbol{\phi}|\boldsymbol{x})$. Consequently, the marginal posterior distributions of mixing proportions and component-specific parameters will be coinciding, i.e., $f(w_1|\boldsymbol{x}) = \cdots = f(w_K|\boldsymbol{x})$ and $f(\theta_1|\boldsymbol{x}) = \cdots = f(\theta_K|\boldsymbol{x})$. Thus, when approximating the posterior distribution via MCMC sampling, the standard practice of ergodic averages for estimating quantities of interest (such as the mean of the marginal posterior distribution for each parameter) becomes meaningless. In order to deal with this identifiability problem, we post-process the simulated MCMC output using a deterministic relabeling algorithm, that is the Equivalence Classes Representatives (ECR) algorithm (Papastamoulis and Iliopoulos 2010; Papastamoulis 2014), as implemented in the `R` package `label.switching` (Papastamoulis 2016).

A second source of identifiability problems is related to orthogonal transformations of the matrix of factor loadings. A popular practice (Geweke and Zhou 1996; Fokoué and Titterington 2003; Mavridis and Ntzoufras 2014; Papastamoulis 2018b) to overcome this issue is to pre-assign values to some entries of $\boldsymbol{\Lambda}$; in particular, we set the entries of the upper diagonal of the first $q \times q$ block matrix of $\boldsymbol{\Lambda}$ equal to zero:

$$\boldsymbol{\Lambda} = \begin{pmatrix} \lambda_{11} & 0 & \cdots & 0 \\ \lambda_{21} & \lambda_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{q1} & \lambda_{q2} & \cdots & \lambda_{qq} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{p1} & \lambda_{p2} & \cdots & \lambda_{pq} \end{pmatrix}.$$

Another problem is related to the so-called sign switching phenomenon, see, e.g., Conti et al. (2014). Simultaneously switching the signs of a given row r of $\boldsymbol{\Lambda}$; $r = 1, \ldots, p$ and $\boldsymbol{y}_i$ does not alter the likelihood. Thus, $\boldsymbol{\Lambda}$ and $\boldsymbol{y}_i$; $i = 1, \ldots, n$ are not marginally identifiable due to sign switching across the MCMC trace. However, this is not a problem in our implementation, since all parameters of the marginal density of $\boldsymbol{x}_i$ in (6) are identified (see also the discussion for sign-invariant parametric functions in Papastamoulis (2018b)).

Parameter-expanded approaches are preferred in the recent literature (Bhattacharya and Dunson 2011; McParland et al. 2017), because the mixing of the MCMC sampler is improved. In our implementation, we are able to obtain excellent mixing using the popular approach of restricting elements of $\boldsymbol{\Lambda}$: the reader is referred to Figure 2 of Papastamoulis (2018b), where it is obvious that our MCMC sampler has the ability to rapidly move between the multiple modes of the target posterior distribution of $\boldsymbol{\Lambda}$ (more details on convergence diagnostics are also presented in "Appendix A.4" of Papastamoulis (2018b)).

2.2 Prior assumptions

We assume that the number of mixture components (K) has a sufficiently large value so that it overestimates the "true" number of clusters. Unless otherwise stated, the default choice is $K = 20$. All prior assumptions of the overfitting mixture models are discussed in detail in Papastamoulis (2018b). For ease of presentation, we repeat them in this section. Let $\mathcal{D}(\cdots)$ denote the Dirichlet distribution, and $\mathcal{G}(\alpha, \beta)$ denote the Gamma distribution with mean α/β. Let also $\boldsymbol{\Lambda}_{kr\cdot}$ denote the r-th row of the matrix of factor loadings $\boldsymbol{\Lambda}_k$; $k = 1, \ldots, K$; $r = 1, \ldots, p$. The following prior assumptions are imposed on the model parameters:

$$\boldsymbol{w} \sim \mathcal{D}(\gamma, \ldots, \gamma), \quad \gamma = \frac{1}{K} \tag{15}$$

$$\boldsymbol{\mu}_k \sim \mathcal{N}_p(\boldsymbol{\xi}, \boldsymbol{\Psi}), \quad \text{iid for } k = 1, \ldots, K \tag{16}$$

$$\boldsymbol{\Lambda}_{kr\cdot} \sim \mathcal{N}_{\nu_r}(\boldsymbol{0}, \boldsymbol{\Omega}), \quad \text{iid. for } r = 1, \ldots, p \tag{17}$$

$$\sigma_{kr}^{-2} \sim \mathcal{G}(\alpha, \beta), \quad \text{iid for } k = 1, \ldots, K; r = 1, \ldots, p \tag{18}$$

$$\omega_\ell^{-2} \sim \mathcal{G}(g, h), \quad \text{iid for } \ell = 1, \ldots, q \tag{19}$$

where all variables are assumed mutually independent and $\nu_r = \min\{r, q\}; r = 1, \ldots, p; \ell = 1, \ldots, q; j = 1, \ldots, K$. In Eq. (17), $\boldsymbol{\Omega} = \text{diag}(\omega_1^2, \ldots, \omega_q^2)$ denotes a $q \times q$ diagonal matrix, where the diagonal entries are distributed independently according to Eq. (19). A graphical representation of the hierarchical model is given in Figure 1 of Papastamoulis (2018b). The default values of the remaining fixed hyperparameters are given in "Appendix B".

The previous assumptions refer to the case of the unconstrained parameter space, that is the UUU parameterization. Clearly, they should be modified accordingly when a constrained model is used. Under constraint (11), the prior distribution in Eq. (17) becomes $\boldsymbol{\Lambda}_{r\cdot} \sim \mathcal{N}_{\nu_r}(\boldsymbol{0}, \boldsymbol{\Omega})$, independent for $r = 1, \ldots, p$. Under constraints (12) and (13), the prior distribution in Eq. (18) becomes $\sigma_r^{-2} \sim \mathcal{G}(\alpha, \beta)$, independent for $r = 1, \ldots, p$. Finally, under constraints (12) and (14), the prior distribution in Eq. (18) becomes $\sigma^{-2} \sim \mathcal{G}(\alpha, \beta)$.

3 Inference

This section describes the full conditional posterior distributions of model parameters and the corresponding MCMC

sampler. Due to conjugacy, all full conditional posterior distributions are available in closed forms.

3.1 Full conditional posterior distributions

Let us define the following quantities:

$$n_k = \sum_{k=1}^{K} I(z_i = k)$$

$$\boldsymbol{A}_k = n_k \boldsymbol{\Sigma}_k^{-1} + \boldsymbol{\Psi}^{-1}$$

$$\boldsymbol{B}_k = \boldsymbol{\Sigma}_k^{-1} \sum_{k=1}^{K} I(z_i = k) \left(\boldsymbol{x}_i - \boldsymbol{\Lambda}_k \boldsymbol{y}_i\right) + \boldsymbol{\xi}\boldsymbol{\Psi}^{-1}$$

$$\boldsymbol{\tau}_{kr} = \frac{\sum_{i=1}^{n} I(z_i = k)(x_{ir} - \mu_{kr})\boldsymbol{y}_i^T}{\sigma_{kr}^2}$$

$$\boldsymbol{\Delta}_{kr} = \frac{\sum_{i=1}^{n} I(z_i = k)\boldsymbol{y}_i \boldsymbol{y}_i^T}{\boldsymbol{\sigma}_{kr}^2}$$

$$s_{kr} = \sum_{i=1}^{n} I(z_i = k) \left(x_{ir} - \mu_{kr} - \boldsymbol{\Lambda}_{kr\cdot} \boldsymbol{y}_i\right)^2$$

$$\boldsymbol{T} = \sum_{k=1}^{K} \sum_{r=1}^{p} \boldsymbol{\Lambda}_{kr\cdot} \boldsymbol{\Lambda}_{kr\cdot}^T$$

$$\boldsymbol{M}_k = \mathbf{I}_q + \boldsymbol{\Lambda}_k^T \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\Lambda}_k^T$$

for $k = 1, \ldots, K$; $r = 1, \ldots, p$. For a generic sequence of the form $\{G_{rc}; r \in \mathcal{R}, c \in \mathcal{C}\}$, we also define $G_{\bullet c} = \sum_r G_{rc}$ and $G_{r\bullet} = \sum_c G_{rc}$. Finally, $(x|\cdots)$ denotes the conditional distribution of x given the value of all remaining variables.

From Eqs. (6) and (7), it immediately follows that for $k = 1, \ldots, K$

$$\mathrm{P}(z_i = k|\cdots) \propto w_k f\left(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Sigma}_k\right), \tag{20}$$

independent for $i = 1, \ldots, n$, where $f(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the probability density function of the multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. Note that in order to compute the right-hand side of the last equation, inversion of the $p \times p$ matrix $\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_k^T + \boldsymbol{\Sigma}_k$ is required. Using the Sherman–Morrison–Woodbury formula (see, e.g., Hager (1989)), the inverse matrix is equal to $\boldsymbol{\Sigma}_k^{-1} - \boldsymbol{\Sigma}_k^{-1} \boldsymbol{\Lambda}_k \boldsymbol{M}_k^{-1} \boldsymbol{\Lambda}_k^T \boldsymbol{\Sigma}_k^{-1}$, for $k = 1, \ldots, K$. The full conditional posterior distribution of mixing proportions is a Dirichlet distribution with parameters

$$\boldsymbol{w}|\cdots \sim \mathcal{D}(\gamma + n_1, \ldots, \gamma + n_K). \tag{21}$$

The full conditional posterior distribution of the marginal mean per component is

$$\boldsymbol{\mu}_k|\cdots \sim \mathcal{N}_p\left(\boldsymbol{A}_k^{-1} \boldsymbol{B}_k, \boldsymbol{A}_k^{-1}\right), \tag{22}$$

independent for $k = 1 \ldots, K$.

The full conditional posterior distribution of the factor loadings without any restriction is

$$\boldsymbol{\Lambda}_{kr\cdot}|\cdots \sim \mathcal{N}_{\nu_r}\left(\left[\boldsymbol{\Omega}^{-1} + \boldsymbol{\Delta}_{kr}\right]^{-1} \boldsymbol{\tau}_{kr}, \left[\boldsymbol{\Omega}^{-1} + \boldsymbol{\Delta}_{kr}\right]^{-1}\right), \tag{23}$$

independent for $k = 1, \ldots, K$; $r = 1, \ldots, p$. Under constraint (11), we obtain that

$$\boldsymbol{\Lambda}_{r\cdot}|\cdots \sim \mathcal{N}_{\nu_r}\left(\left[\boldsymbol{\Omega}^{-1} + \boldsymbol{\Delta}_{\bullet r}\right]^{-1} \boldsymbol{\tau}_{\bullet r}, \left[\boldsymbol{\Omega}^{-1} + \boldsymbol{\Delta}_{\bullet r}\right]^{-1}\right), \tag{24}$$

independent for $r = 1, \ldots, p$.

The full conditional distribution of error variance without any restriction is

$$\sigma_{kr}^{-2}|\cdots \sim \mathcal{G}\left(\alpha + n_k/2, \beta + s_{kr}/2\right), \tag{25}$$

independent for $k = 1, \ldots, K$; $r = 1, \ldots, p$. Under constraint (12), we obtain that

$$\sigma_r^{-2}|\cdots \sim \mathcal{G}(\alpha + n/2, \beta + s_{\bullet r}/2), \tag{26}$$

independent for $r = 1, \ldots, p$. Under constraints (12) and (13), we obtain that

$$\sigma_k^{-2}|\cdots \sim \mathcal{G}(\alpha + n_k p/2, \beta + s_{k\bullet}/2), \tag{27}$$

independent for $k = 1, \ldots, K$. Under constraints (12) and (14), we obtain that

$$\sigma^{-2}|\cdots \sim \mathcal{G}(\alpha + np/2, \beta + s_{\bullet\bullet}/2). \tag{28}$$

The full conditional distribution of latent factors is given by

$$\boldsymbol{y}_i|\cdots \sim \mathcal{N}_q\left(\boldsymbol{M}_{z_i}^{-1} \boldsymbol{\Lambda}_{z_i}^T \boldsymbol{\Sigma}_{z_i}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_{z_i}), \boldsymbol{M}_{z_i}^{-1}\right), \tag{29}$$

independent for $i = 1, \ldots, n$. Finally, the full conditional distribution for ω_ℓ is

$$\omega_\ell^{-2}|\cdots \sim \mathcal{G}\left(g + Kp/2, h + T_{\ell\ell}/2\right), \tag{30}$$

while under constraint (11) we obtain that

$$\omega_\ell^{-2}|\cdots \sim \mathcal{G}\left(g + p/2, h + T_{\ell\ell}/2K\right), \tag{31}$$

independent for $\ell = 1, \ldots, q$.

3.2 MCMC sampler

Given the number of factors (q) and a model parameterization, a Gibbs sampler (Geman and Geman 1984; Gelfand and Smith 1990) coupled with a prior parallel tempering scheme (Geyer 1991; Geyer and Thompson 1995; Altekar et al. 2004) is used in order to produce a MCMC sample from the joint posterior distribution. Each heated chain ($j = 1, \ldots,$ `nChains`) corresponds to a model with identical likelihood as the original, but with a different prior distribution. Although the prior tempering can be imposed on any subset of parameters, it is only applied to the Dirichlet prior distribution of mixing proportions (van Havre et al. 2015). The inference is based on the output of the first chain ($j = 1$) of the prior parallel tempering scheme (van Havre et al. 2015). The number of factors and model parameterization is selected according to the Bayesian information criterion (BIC) (Schwarz 1978), conditional on the most probable number of alive clusters per model (see Papastamoulis (2018b) for a detailed comparison of BIC with other alternatives).

Let $\mathcal{M}$ and $\mathcal{Q}$ denote the set of model parameterizations and number of factors. In the following pseudocode, $x \leftarrow [y|z]$ denotes that x is updated from a draw from the distribution $f(y|z)$ and $\theta_j^{(t)}$ denotes the value of θ at the t-th iteration of the j-th chain.

1. For $(m, q) \in \mathcal{M} \times \mathcal{Q}$
   (a) Obtain initial values ($\boldsymbol{\Omega}_j^{(0)}$, $\boldsymbol{\Lambda}_{m;j}^{(0)}$, $\boldsymbol{\mu}_j^{(0)}$, $\boldsymbol{z}_j^{(0)}$, $\boldsymbol{\Sigma}_{m;j}^{(0)}$, $\boldsymbol{w}_j^{(0)}$, $\boldsymbol{y}_j^{(0)}$) by running the overfitting initialization scheme, for $j = 1, \ldots,$ `nChains`.
   (b) For MCMC iteration $t = 1, 2, \ldots$ update
      i. For chain $j = 1, \ldots,$ `nChains`
         A. $\boldsymbol{\Omega}_j^{(t)} \leftarrow \left[\boldsymbol{\Omega}|\boldsymbol{\Lambda}_{mj}^{(t-1)}\right]$.

            If $m \in$ {UUU,UCU,UUC, UCC} use (30)

            else use (31).
         B. $\boldsymbol{\Lambda}_{m;j}^{(t)} \leftarrow \left[\boldsymbol{\Lambda}|\boldsymbol{\Omega}_j^{(t)}, \boldsymbol{\mu}_j^{(t-1)}, \boldsymbol{\Sigma}_{m;j}^{(t-1)}, \boldsymbol{x}, \boldsymbol{y}_j^{(t-1)}, \boldsymbol{z}_j^{(t-1)}\right]$

            If $m \in$ {UUU,UCU,UUC, UCC} use (23)

            else use (24).
         C. $\boldsymbol{\mu}_j^{(t)} \leftarrow \left[\boldsymbol{\mu}|\boldsymbol{\Lambda}_m^{(t)}, \boldsymbol{\Sigma}_m^{(t-1)}, \boldsymbol{x}, \boldsymbol{y}^{(t-1)}, \boldsymbol{z}_j^{(t-1)}\right]$ according to (22).
         D. $\boldsymbol{z}_j^{(t)} \leftarrow \left[\boldsymbol{z}|\boldsymbol{w}_j^{(t-1)}, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Lambda}_{m;j}^{(t)}, \boldsymbol{\Sigma}_{m;j}^{(t-1)}, \boldsymbol{x}\right]$ according to (20).
         E. $\boldsymbol{w}_j^{(t)} \leftarrow \left[\boldsymbol{w}|\boldsymbol{z}_j^{(t)}\right]$ according to (21) with prior parameter $\gamma = \gamma_{(j)}$.
         F. $\boldsymbol{\Sigma}_{m;j}^{(t)} \leftarrow \left[\boldsymbol{\Sigma}|\boldsymbol{x}, \boldsymbol{z}_j^{(t)}, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Lambda}_{m;j}^{(t)}, \boldsymbol{y}_j^{(t-1)}\right]$

            If $m \in$ {UUU,CUU} use (25)

            else if $m \in$ {UCU,CCU} use (26)

            else if $m \in$ {UUC,CUC} use (27)

            else use (28).
         G. $\boldsymbol{y}_j^{(t)} \leftarrow \left[\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}_j^{(t)}, \boldsymbol{\mu}_j^{(t)}, \boldsymbol{\Sigma}_{m;j}^{(t)}, \boldsymbol{\Lambda}_{m;j}^{(t)}\right]$ according to (29).
      ii. Select randomly $1 \leqslant j^* \leqslant$ `nChains` $- 1$ and propose to swap the states of chains j^* and j^*+1.
   (c) For chain $j = 1$ compute BIC conditionally on the most probable number of alive clusters.
2. Select the best (m, q) model corresponding to chain $j = 1$ according to BIC and reorder the simulated output of the selected model according to ECR algorithm, conditional on the most probable number of alive clusters.

The MCMC algorithm is initialized using random starting values arising from the "overfitting initialization" procedure introduced by Papastamoulis (2018b). For further details on steps 1.(a) (MCMC initialization) and 1.(b).ii (prior parallel tempering scheme), the reader is referred to "Appendix B" (see also Sections 2.6, 2.7 and 2.9 of Papastamoulis (2018b)).

4 Using the `fabMix` package

The main function of the `fabMix` package is `fabMix()`, with its arguments shown in Table 1. This function takes as input a matrix `rawData` of observed data where rows and columns correspond to observations and variables of the dataset, respectively. The parameters of the Dirichlet prior distribution ($\gamma_{(j)}$; $j = 1, \ldots,$ nChains) of the mixing proportions are controlled by `dirPriorAlphas`. The range for the number of factors is specified in the `q` argument. Valid input for `q` is any positive integer vector between 1 and the Ledermann bound (Ledermann 1937) implied by the number of variables in the dataset. By default, all eight parameterizations are fitted; however, the user can specify in `model` any non-empty subset of them.

The `fabMix()` function simulates a total number of `nChains` × `length(models)` × `length(q)` MCMC chains. For each parameterization and number of factors, the (`nChains`) heated chains are processed in parallel, while swaps between pairs of chains are proposed. Parallelization is possible in the parameterization level as well, using the argument `parallelModels`. This means that `parallelModels` are running in parallel where each one of them runs `nChains` chains in parallel, provided

Table 1 Arguments of the `fabMix()` function

Argument	Description
`model`	Any non-empty subset of `c("UUU", "CUU", "UCU", "CCU", "UCC", "UUC", "CUC", "CCC")`, indicating the fitted models. By default, all models are fitted
`Kmax`	Number of components in the overfitted mixture (integer, at least equal to two). Default: 20
`nChains`	Number of parallel (heated) chains. When `dirPriorAlphas` is supplied, this argument can be ignored
`dirPriorAlphas`	vector of length `nChains` in the form of an increasing sequence of positive scalars. Each entry contains the (common) prior Dirichlet parameter for the corresponding chain. Default: `dirPriorAlphas = c(1, 1 + dN*(2:nChains - 1))/Kmax`, where `dN = 1`, for `nChains > 1`. Otherwise, `dirPriorAlphas = 1/Kmax`
`rawData`	The observed data in the form of an $n \times p$ matrix. Clustering is performed on the rows of the matrix
`outDir`	Name of the output folder. An error is thrown if the directory already exists inside the current working directory. Note: it should not correspond to an absolute path, e.g., `outDir = 'example'` is acceptable, but `outDir = 'C:\User\Documents\example'` is not
`mCycles`	Number of MCMC cycles. Each cycle consists of `nIterPerCycle` MCMC iterations. At the end of each cycle a swap of the state of two randomly chosen adjacent chains is attempted
`burnCycles`	Number of cycles that will be discarded as burn-in period
`g`	Prior parameter g. Default value: `g = 0.5`
`h`	Prior parameter h. Default value: `g = 0.5`
`alpha_sigma`	Prior parameter α. Default value: `alpha_sigma = 0.5`
`beta_sigma`	Prior parameter β. Default value: `beta_sigma = 0.5`
`q`	A vector of strictly positive integers, containing the number of factors to be fitted
`normalize`	Logical value indicating whether the observed data will be normalized. Default value: TRUE (recommended)
`nIterPerCycle`	Number of iterations per MCMC cycle. Default value: 10
`warm_up_overfitting`	Number of iterations for the overfitting initialization scheme. Default value: 500
`warm_up`	Number of iterations that will be used to initialize the models before starting proposing switchings. Default value: 5000
`overfittingInitialization`	Logical value indicating whether the chains are initialized via the overfitting initialization scheme. Default: TRUE (recommended)
`rmDir`	Logical value indicating whether to delete the `outDir` directory. Default: TRUE
`parallelModels`	Model-level parallelization: An optional integer specifying the number of cores that will be used in order to fit in parallel each member of `model`. Default: `NULL` (no model-level parallelization)

that the number of available threads is at least equal to `nChains` × `parallelModels`. In order to parallelize our code, the `doParallel` (Revolution Analytics and Steve Weston 2015), `foreach` (Revolution Analytics and Steve Weston 2014) and `doRNG` (Gaujoux 2018) packages are imported.

The prior parameters g, h, α, β in Eqs. (18) and (19) correspond to `g`, `h`,`alpha_sigma` and `beta_sigma`, respectively, with a (common) default value equal to 0.5. It is suggested to run the algorithm using `normalize = TRUE`, in order to standardize the data before running the MCMC sampler. The default behavior of our method is to normalize the data; thus, all reported estimates refer to the standardized dataset. In the case that the most probable number of mixture components is larger than 1, the ECR algorithm is applied in order to undo the label switching problem. Otherwise, the output is post-processed so that the generated parameters of the (single) alive component are switched to the first component of the overfitting mixture.

The sampler will first run for `warm_up` iterations before starting to propose swaps between pairs of chain. By default, this stage consists of 5000 iterations. After that, each chain will run for a series of `mCycles` MCMC cycles, each one consisting of `nIterPerCycle` MCMC iterations (steps A, B, . . ., G of the pseudocode). The updates of factor loadings according to (23) and (24) at step B of the pseudocode are implemented using object-oriented programming using the `Rcpp` and `RcppArmadillo` libraries (Eddelbuettel and François 2011; Eddelbuettel and Sanderson 2014). At the end of each cycle, a swap between a pair of chains is proposed.

Obviously, the total number of MCMC iterations is equal to `warm_up` + `mCycles` × `nIterPerCycle`, and the first `warm_up` + `burnCycles` × `nIterPerCycle` iterations are discarded as burn-in. Given the default val-

Table 2 Output returned to the user of the `fabMix()` function

Object	Description
`bic`	Bayesian information criterion per model and number of factors
`class`	The estimated single best clustering of the observations according to the selected model
`n_Clusters_per_model`	The most probable number of clusters (number of non-empty components of the overfitting mixture) per model and number of factors
`posterior_probability`	The posterior probability of the estimated allocations according to the selected model
`covariance_matrix`	The estimated posterior mean of the covariance matrix per cluster according to the selected model
`mu`	The estimated posterior mean of the mean per cluster according to the selected model
`weights`	The estimated posterior mean of the mixing proportions according to the selected model
`mcmc`	A list containing the MCMC draws for the parameters of the selected model
`Kmap_prob`	The posterior probability of the Maximum A Posteriori number of alive clusters for each parameterization and factor level

ues of `nIterPerCycle`, `warm_up` and `overfitting Initialization`, choices between $50 \leqslant$ `burnCycles` $\leqslant 500 <$ `mCycles` $\leqslant 1500$ are typical in our implementation (see also the convergence analysis in Papastamoulis (2018b)).

While the function runs, some basic information is printed either on the screen (if `parallelModels` is not enabled) or in separate text files inside the output folder (in the opposite case), such as the progress of the sampler as well as the acceptance rate of proposed swaps between chains. The output which is returned to the user is detailed in Table 2. The full MCMC output of the selected model is returned as a list (named as `mcmc`) consisting of `mcmc` objects, a class imported from the `coda` package (Plummer et al. 2006). In particular, `mcmc` consists of the following:

- `y`: object of class `mcmc` containing the simulated factors.
- `w`: object of class `mcmc` containing the simulated mixing proportions of the alive components, reordered according to ECR algorithm.
- `Lambda`: list containing objects of class `mcmc` with the simulated factor loadings of the alive components, reordered according to ECR algorithm. Note that this particular parameter is not identifiable due to sign switching across the MCMC trace.
- `mu`: list containing objects of class `mcmc` with the simulated marginal means of the alive components, reordered according to ECR algorithm.
- `z`: matrix of the simulated latent allocation variables of the mixture model, reordered according to ECR algorithm.
- `Sigma`: list containing objects of class `mcmc` with the simulated variance of errors of the alive components, reordered according to ECR algorithm.
- `K_all_chains`: matrix of the simulated values of the number of alive components per chain.

The user can call the `print`, `summary` and `plot` methods of the package in order to easily retrieve and visualize various summaries of the output, as exemplified in the next section.

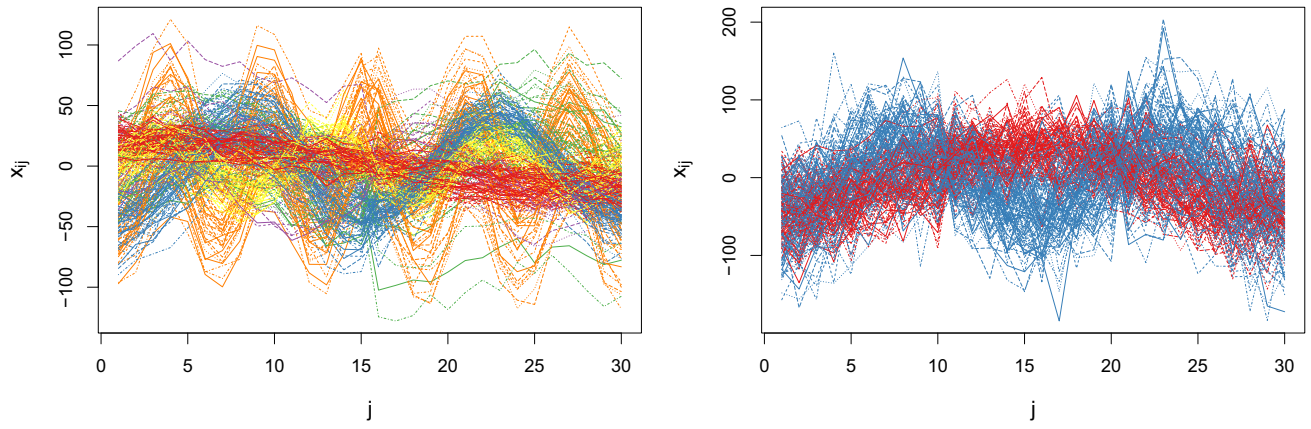
5 Examples

This section illustrates our method. At first, we demonstrate a typical implementation on two single simulated datasets and explain in detail the workflow. Then we perform an extensive simulation study for assessing the ability of the proposed method to recover the correct clustering and compare our findings to the `pgmm` package (McNicholas and Murphy 2008, 2010; McNicholas et al. 2010, 2015). Application to four publicly available datasets is provided next.

5.1 Simulation study

We simulated synthetic data of $p = 30$ variables consisting of $n = 300$ observations and $K = 6$ clusters (dataset 1) and $n = 200$, $K = 2$ (dataset 2), as shown in Fig. 1. Both of them were generated using MFA models with $q = 2$ (dataset 1) and $q = 3$ (dataset 2) factors. The two datasets exhibit different characteristics: The variance of errors per cluster ($\boldsymbol{\Sigma}_k$) is significantly larger in dataset 2 compared to dataset 1. In addition, the selection of factor loadings in dataset 2 results in more complex covariance structure. The generating mechanism, described in detail in Papastamoulis (2018b), is available in the `fabMix` package via the `simData()` and `simData2()` functions, as shown below.

```
> library('fabMix')
#  dataset 1
> set.seed(1)
> n = sample(100*(1:10), 1) # sample
size
```



(a) Dataset 1 **(b)** Dataset 2

Fig. 1 Simulated datasets of $p = 30$ variables consisting of $n = 300$ observations and $K = 6$ clusters (dataset 1) and $n = 200$, $K = 2$ (dataset 2). The colors display the ground-truth classification of the data. (Color figure online)

```
> q = sample(1:3, 1)  # number of
factors
> K = sample(1:10, 1) # number of
clusters
# results to n = 300, q = 2, K = 6
> p = 30    # number of variables
# inverse variance of errors
> sINV <- array(data = NA, dim = c(K,p))
> for(k in 1:K){sINV[k,]
<- 1/(1+20*log(k+1))}
> dataset1 <- simData(sameSigma=FALSE,
K.true=K,
+ n = n, q = q, p = p, sINV_values=sINV)
# synthetic dataset 2
> set.seed(30)
> n = 200; q = 3; K = 2; p = 30
# inverse variance of errors
> sINV <- array(data = NA, dim = c(K,p))
> for(k in 1:K){sINV[k,]
<- 1/(1+1000*log(k+1))}
> dataset2 <- simData2(sameSigma=FALSE,
K.true=K,
+ n = n, q = q, p = p, sINV_values
= sINV)
```

Next we estimate the eight overfitting Bayesian MFA models with $K_{\max} = 20$ mixture components, assuming that the number of factors ranges in the set $1 \leqslant q \leqslant 5$. The MCMC sampler runs `nChains = 4` heated chains, each one consisting of `mCycles = 700` cycles, while the first `burnCycles = 100` are discarded. Recall that each MCMC cycle consists of `nIterPerCycle = 10` usual MCMC iterations and that there is an additional warm-up period of the MCMC sampler (before starting to propose chain swaps) corresponding to 5000 usual MCMC iterations.

```
> Kmax <- 20    # number of components
> nChains <- 4  # number of chains
> qRange <- 1:5 # number of factors
#       Run fabMix() for dataset 1
set.seed(1)
> fm1 <- fabMix(nChains = nChains,
+ rawData = dataset1$data, outDir
= "tmp1",
+ Kmax = Kmax, mCycles = 700, burnCycles
= 100,
+ q = qRange, parallelModels = 4)
#       Run fabMix() for dataset 2
set.seed(1)
> fm2 <- fabMix(nChains = nChains,
+ rawData = dataset2$data, outDir
= "tmp2",
+ Kmax = Kmax, mCycles = 700, burnCycles
= 100,
+ q = qRange, parallelModels = 4)
```

The argument `parallelModels = 4` implies that four parameterizations will be processed in parallel. In addition, each model will use `nChains = 4` threads to run in parallel the specified number of chains. Our job script used 16 threads so in this case the `parallelModels` × `nChains` = 16 jobs are efficiently allocated.

5.1.1 Methods for printing, summarizing and plotting the output

The `print` method for a `fabMix.object` displays some basic information for a given run of the `fabMix` function. The following output corresponds to the first dataset.

```
> print(fm1)
* Run information:
```

```
      Number of fitted models:
      (5 factor levels) x
      (8 parameterizations)
       = 40 models.
      Selected model: UUC model with
      K = 6
      clusters and q = 2 factors.

* Maximum A Posteriori (MAP) number of
'alive' clusters and selected number of
factors (BIC) per model:
  model K_MAP K_MAP_prob q  BIC_q
  chain_swap
1   UUU     4   1.00 3 4295.0      8.43%
2   CUU     5   0.70 3 4248.9     13.29%
3   UCU     7   0.71 2 3345.8     25.86%
4   CCU    13   0.70 2 3410.5     93.71%
5   UCC     7   0.55 2 3270.7     22.71%
6   UUC     6   1.00 2 2274.5     19.43%
7   CUC    10   0.41 3 2868.3     78.43%
8   CCC    13   0.52 2 3378.3     92.43%

* Estimated number of observations per
cluster  (selected model):
label
 4  7 13 14 15 17
60 55 41 72 50 22
```

The following output corresponds to the `print` method for the `fabMix` function for the second dataset.

```
> print(fm2)
* Run information:
      Number of fitted models:
      (5 factor levels) x
      (8 parameterizations)
       = 40 models.
      Selected model: UUC model with
      K = 2 clusters and q = 2 factors.

* Maximum A Posteriori (MAP) number of
'alive' clusters and selected number of
factors (BIC) per model:
  model K_MAP K_MAP_prob q  BIC_q chain_
                            swaps
1   UUU  2   1.00 2 14776.6  4.86%
2   CUU  2   1.00 2 14676.0  3%
3   UCU  2   0.64 3 15043.1  4.57%
4   CCU  3   0.48 3 14836.2  12.71%
5   UCC  3   0.64 2 15141.9  4.29%
6   UUC  2   0.95 2 14558.5  3.14%
7   CUC  3   0.85 3 14598.7  4.14%
8   CCC  4   0.44 3 14851.6  11.14%

* Estimated number of observations per
cluster  (selected model):
label
  6  20
113  87
```

We conclude that the selected models correspond to the UUC parameterization with $K = 6$ clusters and $q = 2$ factors for dataset 1 and $K = 2$, $q = 2$ for dataset 2. The selected number of clusters and factors for the whole range of eight models is displayed next, along with the estimated posterior probability of the number of alive clusters per model (`K_MAP_prob`), the value of the BIC for the selected number of factors (`BIC_q`) as well as the proportion of the accepted swaps between the heated MCMC chains in the last column. The frequency table of the estimated single best clustering of the datasets is displayed in the last field. We note that the labels of the frequency table correspond to the labels of the alive components of the overfitting mixture model, that is, components 4, 7, 13, 14, 15, and 17 for dataset 1 and components 6 and 20 for dataset 2. Clearly, these labels can be renamed to 1, 2, 3, 4, 5, 6 and 1, 2 respectively, but we prefer to retain the raw output of the sampler as a reminder of the fact that it corresponds to the alive components of the overfitted mixture model.

The `summary` method of the `fabMix` package summarizes the MCMC output for the selected model by calculating posterior means and quantiles for the mixing proportions, marginal means and the covariance matrix per (alive) cluster. A snippet of the output for dataset 2 is shown below.

```
> s <- summary(fm2)
* 'Alive' cluster labels:
[1] "6"  "20"

* Posterior mean of the mixing
proportions:
   6   20
0.58 0.42

* Posterior mean of the marginal means:
         Cluster label
Variable     6    20
     V1  -0.06  0.08
     V2  -0.02  0.02
     ..............
     V30 -0.01  0.00

* Posterior mean of the covariance
matrix:

   Covariance matrix for cluster '6':
       V1    V2  ...   V30
V1   1.12  0.53  ... -0.18
V2   0.53  1.11  ... -0.14
.........................
V30 -0.18 -0.14  ...  1.27
```

```
   Covariance matrix for cluster '20':
       V1    V2   ...   V30
V1    0.66  0.39  ... -0.05
V2    0.39  0.88  ... -0.03
.........................
V30 -0.05 -0.03  ...  0.57

Quantiles for each parameter:
                quantile
parameter        2.5%   25%   50%  75%  97.5%
  weight_6       0.51  0.55  0.58  0.51  0.65
  weight_20      0.35  0.40  0.42  0.45  0.50
  mean_6_V1     -0.27 -0.13 -0.06  0.01  0.13
  mean_20_V1    -0.09  0.03  0.08  0.14  0.25
  mean_6_V2     -0.22 -0.09 -0.02  0.04  0.15
  mean_20_V2    -0.17 -0.05  0.02  0.08  0.19
  ..........................................
  mean_6_V30    -0.21 -0.08 -0.01  0.07  0.20
  mean_20_V30   -0.16 -0.06  0.00  0.06  0.17
  cov_6_V1_V1    0.89  1.03  1.10  1.19  1.46
  cov_20_V1_V1   0.50  0.59  0.64  0.72  0.85
  cov_6_V1_V2    0.37  0.46  0.52  0.59  0.78
  cov_20_V1_V2   0.24  0.32  0.38  0.44  0.56
  ..........................................
  cov_6_V1_V30  -0.37 -0.24 -0.18 -0.12  0.01
  cov_20_V1_V30 -0.16 -0.08 -0.05 -0.01  0.06
  ..........................................
  cov_6_V2_V30  -0.34 -0.21 -0.13 -0.08  0.04
  cov_20_V2_V30 -0.17 -0.07 -0.03  0.03  0.12
  ..........................................
  cov_6_V30_V30  1.03  1.18  1.26  1.37  1.62
  cov_20_V30_V30 0.45  0.51  0.56  0.61  0.73
```

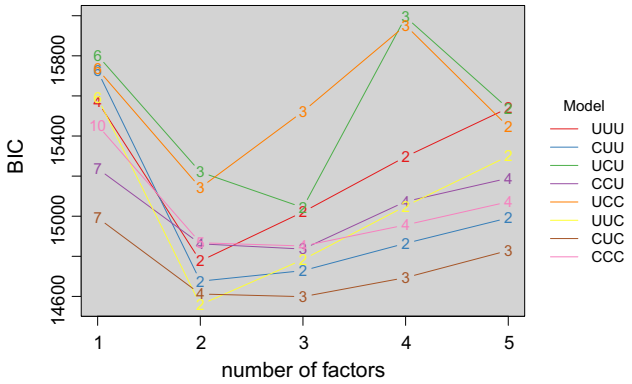
The printed output is also returned to the user via `s$posterior_means` and `s$quantiles`.

The `plot()` method of the package generates the following types of graphics output:

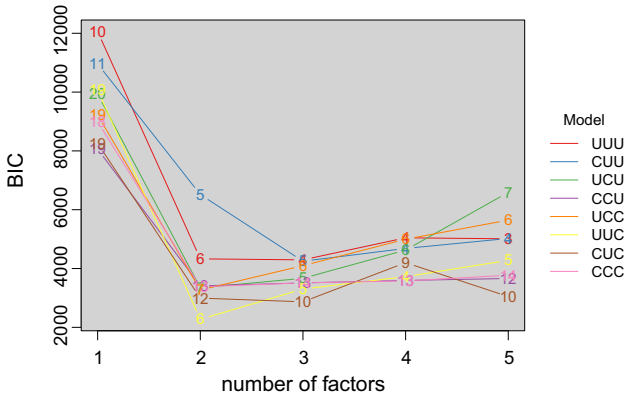
(1) Plot of the BIC values per factor level and parameterization.
(2) Plot of the posterior means of marginal means ($\boldsymbol{\mu}_k$) per (alive) cluster and highest density intervals of the corresponding normal distribution along with its assigned data.
(3) The coordinate projection plot of the `mclust` package (Fraley and Raftery 2002; Scrucca et al. 2017), that is, a scatterplot of the assigned data per cluster for each pair of variables.
(4) Visualization of the posterior mean of the correlation matrix per cluster using the `corrplot` package.
(5) The MAP estimate of the factor loadings ($\boldsymbol{\Lambda}_k$) per (alive) cluster.

The following commands produce plot (1) for datasets 1 and 2.

```
> plot(fm1, what = 'BIC')
> plot(fm2, what = 'BIC')
```



(a) Dataset 1



(b) Dataset 2

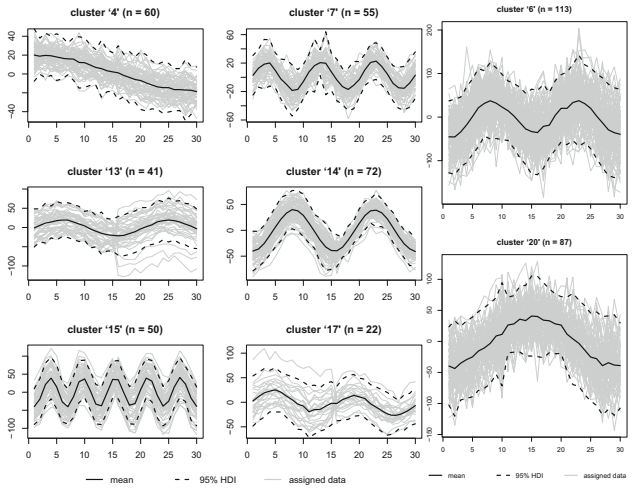
Fig. 2 BIC values per parameterization and factor level using the `plot(fabMix.object)` method

The produced plots are shown in Fig. 2. Note that each point in the plot is labeled by an integer, which corresponds to the MAP number of alive components for the specific combination of factors and parameterization.

The following commands produce plot (2) for datasets 1 and 2.

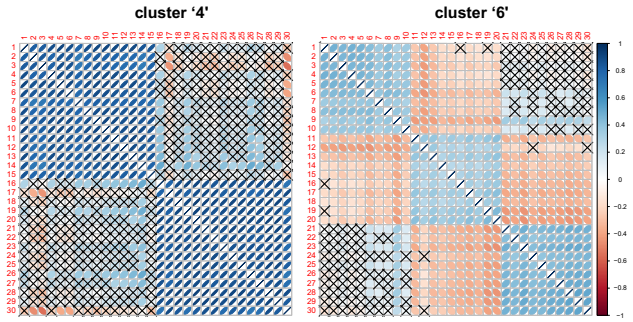
```
> plot(fm1, what = 'classification_matplot',
+ class_mfrow = c(3,2), confidence = 0.95)
> plot(fm2, what = 'classification_matplot',
+ class_mfrow = c(2,1), confidence = 0.95)
```

The created plots are shown in Fig. 3. The `class_mfrow` arguments control the rows and columns of the layout and it should consist of two integers with their product equal to the selected number of (alive) clusters. In addition, a legend is placed on the bottom of the layout. The value(s) in the `confidence` argument draws the highest density interval(s) of the estimated normal distribution. Note that these plots display the original and not the scaled dataset which is used in the MCMC sampler. Therefore, the central curve and confidence limits displayed in the specific plot correspond to the mean and variance (multiplied by the appropriate quantile of the standard Normal distribution) of the random variables



(a) Dataset 1 **(b)** Dataset 2

Fig. 3 Marginal mean with 95% highest density interval and the corresponding assigned data per alive cluster using the `plot(fabMix.object)` method



(a) Dataset 1 **(b)** Dataset 2

Fig. 4 Correlation matrix for the first (alive) cluster of each dataset

arising by applying the inverse of the z-transformation on the MCMC estimates reported by the `fabMix` function.

Figure 4 visualizes the correlation matrix for the first cluster of each dataset, using the `corrplot` package. The argument `sig_correlation` $= \alpha$ is used for marking cases where the equally tailed $(1 - \alpha)$ Bayesian credible interval contains zero. The following commands generate the plots in Fig. 4.

```
> plot(fm1, what = 'correlation',
+ sig_correlation = 0.05)
> plot(fm2, what = 'correlation',
+ sig_correlation = 0.05)
```

5.1.2 Assessing clustering accuracy and comparison with `pgmm`

In this section, we compare our findings against the ground truth in simulated datasets and also compare against the `pgmm` package, considering the same range of clusters and factors per dataset. For each combination of number of factors, components and parameterization, the `pgmmEM()` algorithm was initialized using three random starting values as well as the K-means clustering algorithm, that is, four different starts in total. Note that the number of different starts of the EM algorithm is set equal to number of parallel chains in the MCMC algorithm. The input data are standardized in both algorithms.

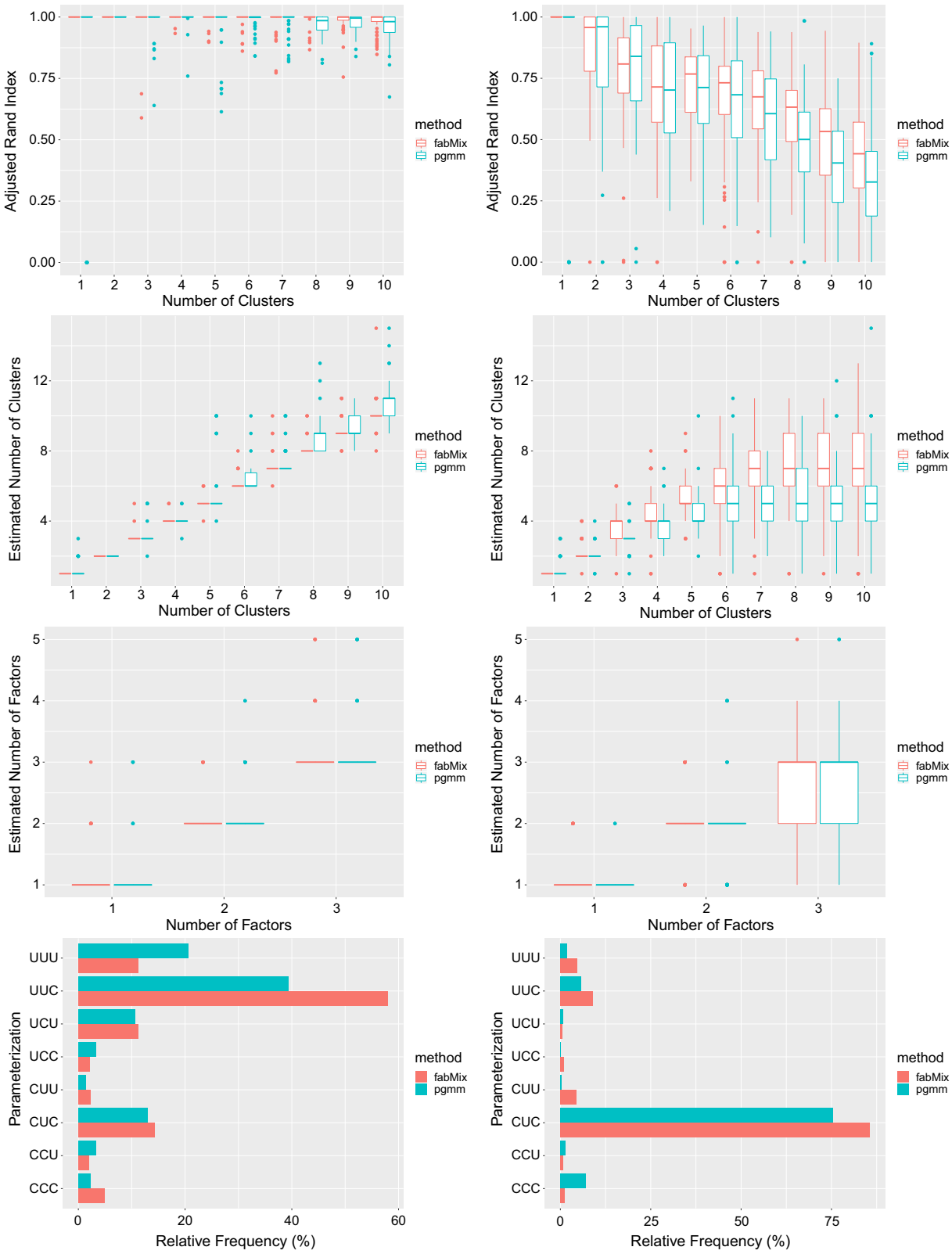
Table 3 Selected number of clusters, factors, parameterization and adjusted Rand index for simulated data 1 and 2

Data (K, q)	`fabMix`				`pgmm`			
	$\widehat{K}$	$\widehat{q}$	Model	ARI	$\widehat{K}$	$\widehat{q}$	Model	ARI
1 (6, 2)	6	2	UUC	1	7	2	UUC	.95
2 (2, 3)	2	2	UUC	.98	2	2	CUC	.88

As shown in Table 3, the adjusted Rand index (ARI) (Rand 1971) between `fabMix` and the ground-truth classification is equal to 1 and 0.98 for simulated datasets 1 and 2, respectively. The corresponding ARI for `pgmm` is equal to 0.98 and 0.88, respectively. In both cases, our method finds the correct number of clusters; however, `pgmm` overestimates K in dataset 1. Both methods select the UUC parameterization in dataset 1, but in dataset 2 different models are selected (UUC by `fabMix` and CUC by `pgmm`).

The selected number of factors equals 2; however, in dataset 2 the "true" number of factors equals 3. The underestimation of the number of factors in dataset 2 remains true for a wide range of similar data: In particular, we generated synthetic datasets with identical parameter values as the ones in dataset 2 but each time the sample size was increasing by 200 observations. We observed that the correct number of factors is returned when $n \geqslant 1600$ for `fabMix` and $n \geqslant 1800$ for `pgmm`.

Next we replicate the two distinct simulation procedures (according to the `simData()` and `simData2()` functions of the package) used to generate the previously described datasets, but considering that $1 \leqslant K \leqslant 10$ (true number of clusters) and $1 \leqslant q \leqslant 3$ (true number of factors). The number of variables remains the same as before, that is $p = 30$, and the sample size is drawn uniformly at random in the set $\{100, 200, \ldots, 1000\}$. We will use the terms 'scenario 1' and 'scenario 2' to label the two different simulation procedures. In scenario 1, the diagonal of the variance of errors is generated as $\sigma^2_{kr} = 1 + 20\log(k+1)$, $r = 1, \ldots, p$, whereas in scenario 2: $\sigma^2_{kr} = 1 + u_r \log(k+1)$, where $u_r \sim \text{Uniform}(500, 1000)$, $r = 1, \ldots, p$; $k = 1, \ldots, K$. In general, scenario 1 generates datasets with well-separated clusters. On the other hand, the amount of error variance in scenario 2 makes the clusters less separated. For a given simulated dataset with K_{true} clusters and q_{true} factors, we



(a) Scenario 1 **(b)** Scenario 2

Fig. 5 Adjusted Rand index (first row), estimated number of clusters (second row), estimated number of factors (third row) and selected parameterization (last row) for various replications of scenarios 1 and 2 with varying number of clusters and factors. In all cases, the sample size is drawn randomly in the set {100, 200, . . . , 1000}

are considering that the total number of components in the overfitting mixture model (`fabMix`) as well as the maximum number of components fitted from `pgmm` is set equal to $K_{\max} = K_{\text{true}} + 6$ and that the number of factors ranges between $1 \leqslant q \leqslant q_{\text{true}} + 2$. These bounds are selected in order to speed up computation time without introducing any bias in the resulting inference (as confirmed by a smaller pilot study). For each scenario, 500 datasets were simulated.

The main findings of the simulation study are illustrated in Fig. 5. Note that in scenario 1 `fabMix` almost always finds the correct clustering structure: The boxplots of the adjusted Rand index are centered at 1, and, on the second row, the boxplots of the estimated number of clusters are centered at the corresponding true value. On the other hand, observe that for $K \geqslant 6$ `pgmm` has the tendency to overestimate the number of clusters. In the more challenging scenario 2, the estimates of the number of cluster exhibit larger variability. However, note that for $K = 8, 9, 10$ the number of clusters selected by `fabMix` is closer to the true value than `pgmm`, a fact which is also reflected in the ARI where `fabMix` tends to have larger values than `pgmm`. For both scenarios, the estimation of the number of factors is in strong agreement between the two methods, as shown in the third row of Fig. 5. In the last row, the selected parameterization is shown. Observe that the results are fairly consistent between the two methods.

Finally, we note that in the presented simulation study, the generated clusters have equal sizes (on average). The reader is referred to "Appendix C" for exploring the performance of the compared methods in the presence of small and large clusters with respect to the size of the available data (n).

5.2 Publicly available datasets

In this section, we analyze four publicly available datasets: a subset of the wave dataset (Breiman et al. 1984; Lichman 2013) available at the `fabMix` package, the wine dataset (Forina et al. 1986) available at the `pgmm` package, the coffee dataset (Streuli 1973) available at the `pgmm` package, and the standardized yeast cell cycle data (Cho et al. 1998) available at http://faculty.washington.edu/kayee/model/. Note that Papastamoulis (2018b) analyzed the first three datasets but only considering the UUU and UCU parameterizations for `fabMix`.

The coffee dataset consists of $n = 43$ coffee samples of $p = 12$ variables, collected from beans corresponding to the Arabica and Robusta species (thus, $K = 2$). The wave dataset consists of a randomly sampled subset of 1500 observations from the wave dataset (Breiman et al. 1984), available from the UCI machine learning repository (Lichman 2013). According to the available ground-truth classification of the dataset, there are three equally weighted underlying classes of 21-dimensional continuous data. The wine dataset (Forina et al. 1986), available at the `pgmm` package (McNicholas et al. 2015), contains $p = 27$ variables measuring chemical and physical properties of $n = 178$ wines, grouped in three types (thus, $K = 3$). The reader is referred to McNicholas and Murphy (2008), Papastamoulis (2018b) for more detailed descriptions of the data.

Table 4 Selected number of clusters, factors, parameterization and adjusted Rand index for the publicly available data

Data (K)	`fabMix`				`pgmm`			
	$\widehat{K}$	$\widehat{q}$	Model	ARI	$\widehat{K}$	$\widehat{q}$	Model	ARI
Coffee (2)	2	1	CUU	1	4	4	CUU	.29
Wave (3)	3	1	UCU	.61	3	1	UCU	.61
Wine (3)	5	4	CUU	.83	3	4	CUU	.97
Yeast (5)	5	6	CUU	.50	20	10	CUC	.20

The yeast cell cycle data (Cho et al. 1998) quantify gene expression levels over two cell cycles (17 time points). The dataset has previously been used for evaluating the effectiveness of model-based clustering techniques (Yeung et al. 2001). We used the standardized subset of the 5-phase criterion, containing $n = 384$ genes measured at $p = 17$ time points. The expression levels of the $n = 384$ genes peak at different time points corresponding to the five phases of cell cycle, so this five-class partition of the data is used as the ground-truth classification.

We applied our method using the eight parameterizations of overfitting mixtures with `Kmax = 20` components for $1 \leqslant q \leqslant q_{\max}$ factors using `nChains = 4` heated chains. We set $q_{\max} = 5$ for the coffee, wave and wine datasets, while $q_{\max} = 10$ for the yeast cell cycle dataset. The number of MCMC cycles was set to `mCycles = 1100`, while the first `burnCycles = 100` were discarded as burn-in. The eight parameterizations are processed in parallel on `parallelModels = 4` cores, while each heated chain of a given parameterization is also running in parallel. All other prior parameters were fixed at their default values.

We have also applied `pgmm` considering the same range of clusters and factors per dataset. For each combination of number of factors, components and parameterization, the EM algorithm was initialized using five random starting values as well as the K-means clustering algorithm, that is, six different starts in total. For the coffee dataset, a larger number of different starts are required as discussed in Papastamoulis (2018b).

Table 4 summarizes the results for each of the publicly available data. We conclude that `fabMix` performs better than `pgmm` at the coffee and yeast datasets. In the wine dataset, on the other hand, `pgmm` performs better than `fabMix`, but we underline the improved performance of our method compared to the one reported by Papastamoulis (2018b) where only the UUU and UCU parameterizations were fitted. The two methods are in agreement on the wave

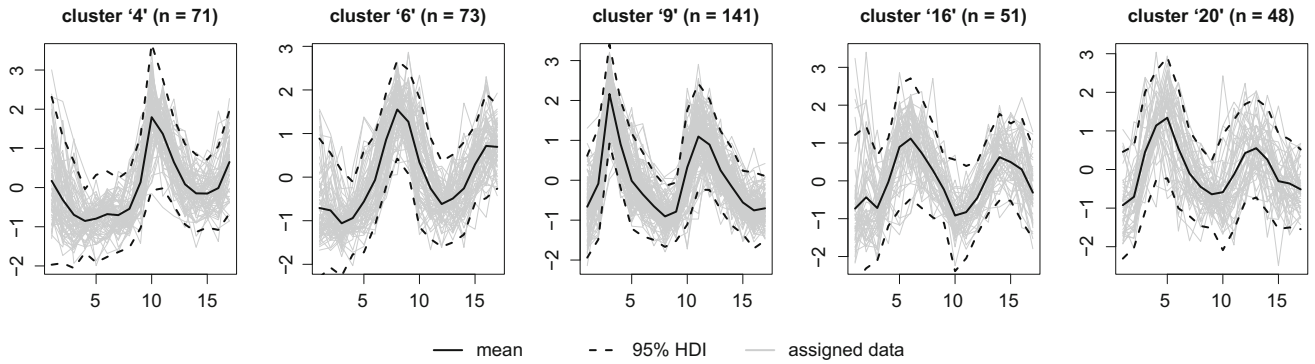


Fig. 6 Marginal mean with 95% highest density interval and the corresponding assigned data per alive cluster for the yeast cell cycle data

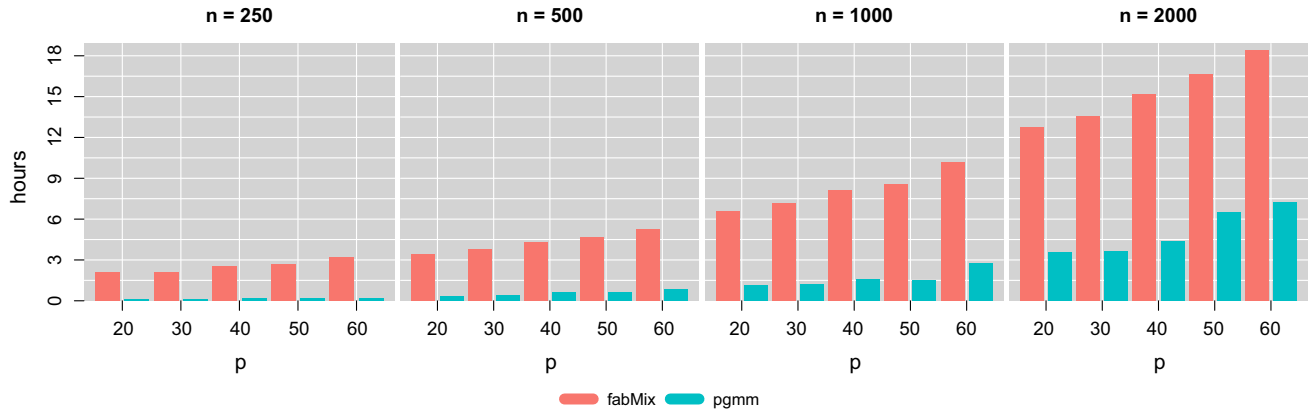


Fig. 7 Total time needed for fitting the eight parameterizations considering $q = 1, \ldots, 5$ (40 models in total) for various levels of sample size (n) and number of variables (p). We considered $K_{\max} = 20$ components in `fabMix` and $1 \leqslant K \leqslant 20$ in `pgmm`. Each parameterization is fitted in parallel using eight threads. No multiple runs (`pgmm`) or parallel chains (`fabMix`) are considered. The MCMC algorithm in `fabMix` ran for 12,000 iterations. The bars display averaged wall clock run times across five replicates

dataset. The `plot` command of the `fabMix` package displays the estimated clusters according to the CUU model with six factors for the yeast dataset, as shown in Fig. 6.

6 Discussion and further remarks

This study offered an efficient Bayesian methodology for model-based clustering of multivariate data using mixtures of factor analyzers. The proposed model extended the ideas of Papastamoulis (2018b) building upon the previously introduced set of parsimonious Gaussian mixture models (McNicholas and Murphy 2008; McNicholas et al. 2010). The additional parameterizations improved the performance of the proposed method compared to Papastamoulis (2018b) where only two out of eight parameterizations were available. Furthermore, our contributed `R` package makes the proposed method available to a wider audience of researchers.

The computational cost of our MCMC method is larger than the EM algorithm, as shown in Fig. 7. But of course, when a point estimate is required, the EM algorithm is the quickest solution. When a point estimate is not sufficient, our method offers an attractive Bayesian treatment of the problem. Clearly, the Bayesian approach does show further advantages (as in the simulated datasets according to Scenario 1, as well as in the coffee and yeast datasets), where the multimodality of the likelihood potentially causes the EM to converge to local maxima.

A direction for future research is to generalize the method in order to automatically detect the number of factors in a fully Bayesian manner. This is possible by, for example, treating the number of factors as a random variable and implementing a reversible jump mechanism in order to update it inside the MCMC sampler. Another possibility would be to incorporate strategies for searching the space of sparse factor loading matrices allowing posterior inference for factor selection (Bhattacharya and Dunson 2011; Mavridis and Ntzoufras 2014; Conti et al. 2014). Recent advances on infinite mixtures of infinite factor models (Murphy et al. 2019) also allow for direct inference of the number of clusters and factors and could boost the flexibility of our modeling approach.

Acknowledgements The author would like to acknowledge the assistance given by IT services and use of the Computational Shared Facility of the University of Manchester. The suggestions of two anonymous reviewers helped to improve the findings of this study.

A Overfitted mixture model

Assume that the observed data have been generated from a mixture model with K_0 components

$$f_{K_0}(\boldsymbol{x}) = \sum_{k=1}^{K_0} w_k f_k(\boldsymbol{x}|\boldsymbol{\theta}_k),$$

where $f_k \in \mathcal{F}_\Theta = \{f(\cdot|\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}; k = 1, \ldots, K_0$ denotes a member of a parametric family of distributions. Consider that an overfitted mixture model $f_K(\boldsymbol{x})$ with $K > K_0$ components is fitted to the data. Rousseau and Mengersen (2011) showed that the asymptotic behavior of the posterior distribution of the $K - K_0$ redundant components depends on the prior distribution of mixing proportions ($\boldsymbol{w}$). Let d denote the dimension of free parameters of the distribution f_k. For the case of a Dirichlet prior distribution,

$$\boldsymbol{w} \sim \mathcal{D}(\gamma_1, \ldots, \gamma_K) \tag{A.32}$$

if

$$\max\{\gamma_k; k = 1, \ldots, K\} < d/2$$

then the posterior weight of the extra components converges to zero (Theorem 1 of Rousseau and Mengersen (2011)).

Let $f_K(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x})$ denote the joint posterior distribution of model parameters and latent allocation variables for a model with K components. When using an overfitted mixture model, the inference on the number of clusters reduces to (a): choosing a sufficiently large value of mixture components (K), (b): running a typical MCMC sampler for drawing samples from the posterior distribution $f_K(\boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x})$ and (c) inferring the number of "alive" mixture components. Note that at MCMC iteration $t = 1, 2, \ldots$ (c) reduces to keeping track of the number of elements in the set $\boldsymbol{K_0}^{(t)} = \{k = 1, \ldots, K : \sum_{i=1}^{n} I(z_i^{(t)} = k) > 0\}$, where $z_i^{(t)}$ denotes the simulated allocation of observation i at iteration t.

In our case, the dimension of free parameters in the k-th mixture component is equal to $d = 2p + pq - \frac{q(q-1)}{2}$. Following Papastamoulis (2018b), we set $\gamma_1 = \cdots = \gamma_K = \frac{\gamma}{K}$; thus, the distribution of mixing proportions in Eq. (A.32) becomes

$$\boldsymbol{w} \sim \mathcal{D}\left(\frac{\gamma}{K}, \ldots, \frac{\gamma}{K}\right) \tag{A.33}$$

where $0 < \gamma < d/2$ denotes a pre-specified positive number. Such a value is chosen for two reasons. At first, it is smaller than $d/2$ so the asymptotic results of Rousseau and Mengersen (2011) ensure that extra components will be emptied as $n \to \infty$. Second, this choice can be related to standard practice when using Bayesian nonparametric clustering methods where the parameters of a mixture are drawn from a Dirichlet process (Ferguson 1973), that is, a Dirichlet process mixture model (Neal 2000).

B Details of the MCMC sampler

Data normalization and prior parameters Before running the sampler, the raw data are standardized by applying the z-transformation

$$\frac{x_{ir} - \bar{x}_r}{\sqrt{s_r^2}}, \quad i = 1, \ldots, n; r = 1, \ldots, p$$

where $\bar{x}_r = \frac{\sum_{i=1}^{n} x_{ir}}{n}$ and $s_r^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ir} - \bar{x}_r)^2$. The main reason for using standardized data is that the sampler mixes better. Furthermore, it is easier to choose prior parameters that are not depending on the observed data, that is, using the data twice. In any other case, one could use empirical prior distributions as reported in Fokoué and Titterington (2003), see also Dellaportas and Papageorgiou (2006). For the case of standardized data, the prior parameters are specified in Table 5. Standardized data are also used as input to `pgmm`.

Prior parallel tempering It is well known that the posterior surface of mixture models can exhibit many local modes (Celeux et al. 2000b; Marin et al. 2005). In such cases, simple MCMC algorithms may become trapped in minor modes and demand a very large number of iterations to sufficiently explore the posterior distribution. In order to produce a well-mixing MCMC sample and improve the convergence of our algorithm, we utilize ideas from parallel tempering schemes Geyer (1991), Geyer and Thompson (1995), Altekar et al. (2004), where different chains are running in parallel and they are allowed to switch states. Each chain corresponds to a different posterior distribution, and usually each one represents a "heated" version of the target posterior distribution. This is achieved by raising the original target to a power T with $0 \leqslant T \leqslant 1$, which flattens the posterior surface and thus easier to explore when using an MCMC sampler.

Table 5 Prior parameter specification for the case of standardized data

	α	β	γ	g	h	$\boldsymbol{\xi} = (\xi_1, \ldots, \xi_p)^T$	$\boldsymbol{\Psi}$
Value	0.5	0.5	1	0.5	0.5	$(0, \ldots, 0)^T$	$\mathbf{I}_p$

In the context of overfitting mixture models, van Havre et al. (2015) introduced a prior parallel tempering scheme, which is also applied by Papastamoulis (2018b). Under this approach, each heated chain corresponds to a model with identical likelihood as the original, but with a different prior distribution. Although the prior tempering can be imposed on any subset of parameters, it is only applied to the Dirichlet prior distribution of mixing proportions (van Havre et al. 2015). Let us denote by $f_i(\boldsymbol{\varphi}|\boldsymbol{x})$ and $f_i(\boldsymbol{\varphi})$; $i = 1, \ldots, J$, the posterior and prior distribution of the i-th chain, respectively. Obviously, $f_i(\boldsymbol{\varphi}|\boldsymbol{x}) \propto f(\boldsymbol{x}|\boldsymbol{\varphi}) f_i(\boldsymbol{\varphi})$. Let $\boldsymbol{\varphi}_i^{(t)}$ denote the state of chain i at iteration t and assume that a swap between chains i and j is proposed. The proposed move is accepted with probability $\min\{1, A\}$ where

$$A = \frac{f_i(\boldsymbol{\varphi}_j^{(t)}|\boldsymbol{x}) f_j(\boldsymbol{\varphi}_i^{(t)}|\boldsymbol{x})}{f_i(\boldsymbol{\varphi}_i^{(t)}|\boldsymbol{x}) f_j(\boldsymbol{\varphi}_j^{(t)}|\boldsymbol{x})} = \frac{f_i(\boldsymbol{\varphi}_j^{(t)}) f_j(\boldsymbol{\varphi}_i^{(t)})}{f_i(\boldsymbol{\varphi}_i^{(t)}) f_j(\boldsymbol{\varphi}_j^{(t)})} = \frac{\widetilde{f}_i(w_j^{(t)}) \widetilde{f}_j(w_i^{(t)})}{\widetilde{f}_i(w_i^{(t)}) \widetilde{f}_j(w_j^{(t)})}, \tag{B.34}$$

and $\widetilde{f}_i(\cdot)$ corresponds to the probability density function of the Dirichlet prior distribution related to chain $i = 1, \ldots, J$. According to Eq. (A.33), this is

$$\boldsymbol{w} \sim \mathcal{D}\left(\frac{\gamma_{(j)}}{K}, \ldots, \frac{\gamma_{(j)}}{K}\right), \tag{B.35}$$

for a pre-specified set of parameters $\gamma_{(j)} > 0$ for $j = 1, \ldots, J$.

In our examples, we used a total of $J = 4$ parallel chains where the prior distribution of mixing proportions for chain j in Eq. (B.35) is selected as

$$\gamma_{(j)} = \gamma + \delta(j-1), \quad j = 1, \ldots, J,$$

where $\delta > 0$. For example, when the overfitting mixture model uses $K = 20$ components and $\gamma = 1$ (the default value shown in Table 5), it follows from Eq. (A.33) that the parameter vector of the Dirichlet prior of mixture weights which corresponds to the target posterior distribution ($j = 1$) is equal to $(0.05, \ldots, 0.05)$. Also in our examples, we have used $\delta = 1$, but in general we strongly suggest to tune this parameter until a reasonable acceptance rate is achieved. Each chain runs in parallel, and every 10 iterations we randomly select two adjacent chains $(j, j+1)$, $j \in \{1, \ldots, J-1\}$ and propose to swap their current states. A proposed swap is accepted with probability A in Eq. (B.34).

"Overfitting initialization" strategy We briefly describe the "overfitting initialization" procedure introduced by Papastamoulis (2018b). We used an initial period of 500 MCMC iterations where each chain is initialized from totally random starting values, but under a Dirichlet prior distribution with large prior parameter values. These values were chosen in a way that the asymptotic results of Rousseau and Mengersen (2011) guarantee that the redundant mixture components will have non-negligible posterior weights. More specifically for chain j, we assume $w \sim \mathcal{D}(\gamma'_j, \ldots, \gamma'_j)$ with $\gamma'_{(j)} = \frac{d}{2} + (j-1)\frac{d}{2(J-1)}$, for $j = 1, \ldots, J$. Then, we initialize the actual model by this state. According to Papastamoulis (2018b), this specific scheme was found to outperform other initialization procedures.

C Additional simulations

In the simulation section of the manuscript, the weights of the simulated datasets have been randomly generated from a Dirichlet distribution with mean equal to $1/K$, conditional on the number of clusters (K). Thus, on average, the true cluster sizes are equal. In this section, we examine the performance of the proposed method in the presence of unequal cluster sizes with respect to the size (n) of the observed data.

We replicate the simulation mechanism for scenarios 1 and 2 presented in the main text, but now we consider unequal (true) cluster sizes, as detailed in Table 6. For each case, the sample size is increasing (as shown in the last column of Table 6) while keeping all others parameters (that is, the true values of marginal means and factor loadings) constant. As shown in Table 6, in scenario 1 there are five clusters and two factors, whereas in scenario 2 there are two clusters and three factors. In total, three different examples per scenario are considered: for a given scenario, the component-specific parameters are different in each example, but the weights are the same. An instance of our three examples (per scenario) using $n = 200$ simulated observations is shown in Fig. 8. Observe that in all cases the "true clusters" are not easily distinguishable, especially in scenario 2 where there is a high degree of cluster overlapping.

We applied `fabMix` and `pgmm` using the same number (4) of parallel chains (for `fabMix`) and different starts (for `pgmm`) as in the simulations presented in the main paper. The results are summarized in Figs. 9 and 10 for scenarios 1 and 2, respectively. The adjusted Rand index is displayed in the first line of each figure, where the horizontal axis denotes the sample size (n) of the synthetic data. The dotted black line corresponds to the adjusted Rand index between the ground truth and the cluster assignments arising when applying the Maximum A Posteriori rule using the true parameter values that generated the data, that is,

$$z_i = \max_{k \in \{1, \ldots, K^*\}} \left\{ \frac{w_k^* f_k(\boldsymbol{x}_i|\theta_k^*)}{\sum_{j=1}^{K^*} w_j^* f_j(\boldsymbol{x}_i|\theta_j^*)} \right\}, \quad i = 1, \ldots, \tag{C.36}$$

Table 6 Setup for our simulation scenarios with unequal cluster sizes. In all cases, the dimensionality of the multivariate data is equal to $p = 30$

	Scenario 1	Scenario 2
True number of clusters (K)	5	2
True number of factors (q)	2	3
True mixing proportions ($\boldsymbol{w}$)	$\left(\frac{1}{15}, \frac{2}{15}, \frac{3}{15}, \frac{4}{15}, \frac{5}{15}\right)$	$\left(\frac{1}{20}, \frac{19}{20}\right)$
Sample size (n)	50, 100, 200, 300, 400, 500	50, 100, 200, 300, . . . , 1500

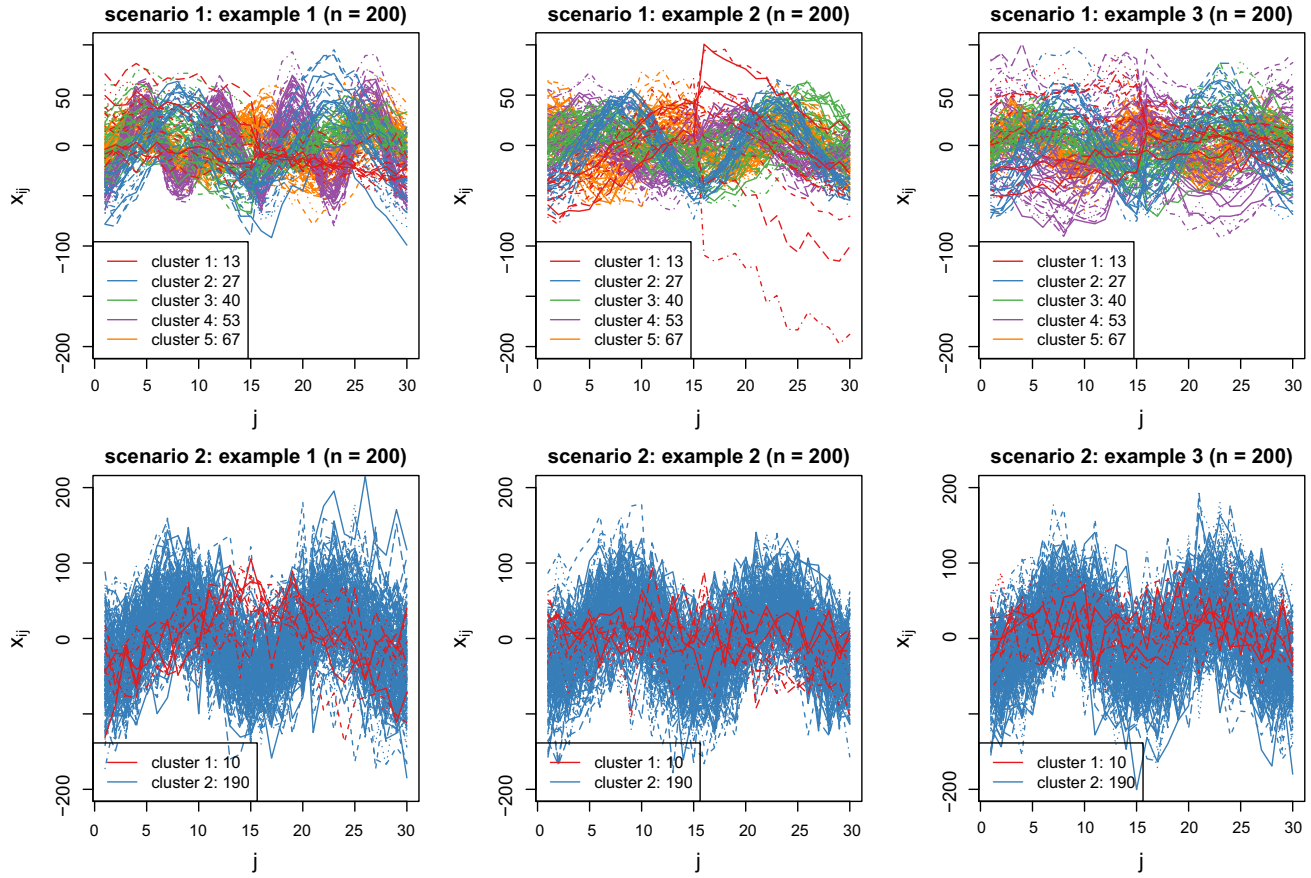


Fig. 8 Examples of simulated datasets with unequal cluster sizes according to scenarios 1 and 2 and $n = 200$. The legend shows the "true" cluster sizes

where K^*, $(w_1^*, \ldots, w_{K^*}^*)$ and $(\theta_1^*, \ldots, \theta_{K^*}^*)$ denote the values of number of components, mixing proportions and parameters of the multivariate normal densities of the mixture model used to generate the data. Observe that in all three examples of scenario 1 the dotted black line is always equal to 1, but this is not the case in the more challenging Scenario 2 due to enhanced levels of cluster overlapping.

The adjusted Rand index between the ground-truth clustering and the estimated cluster assignments arising from `fabMix` and `pgmm` is shown in the first row of Figs. 9 and 10. Clearly, the compared methods have similar performance as the sample size increases, but for smaller values of n the proposed method outperforms the `pgmm` package.

The estimated number of clusters, shown at the second row of Figs. 9 and 10, agrees (in most cases) with the true number of clusters, but note that our method is capable of detecting the right value earlier than `pgmm`. Two exceptions occur at $n = 200$ for example 2 of scenario 1 where `fabMix` (red line at second row of Fig. 9) inferred 6 alive clusters instead of 5, as well as at $n = 800$ for example 1 of scenario 2 where `fabMix` (red line at second row of Fig. 10) inferred 3 alive clusters instead of 2.

Finally, the last row in Figs. 9 and 10 displays the inferred number of factors for scenarios 1 and 2, respec-

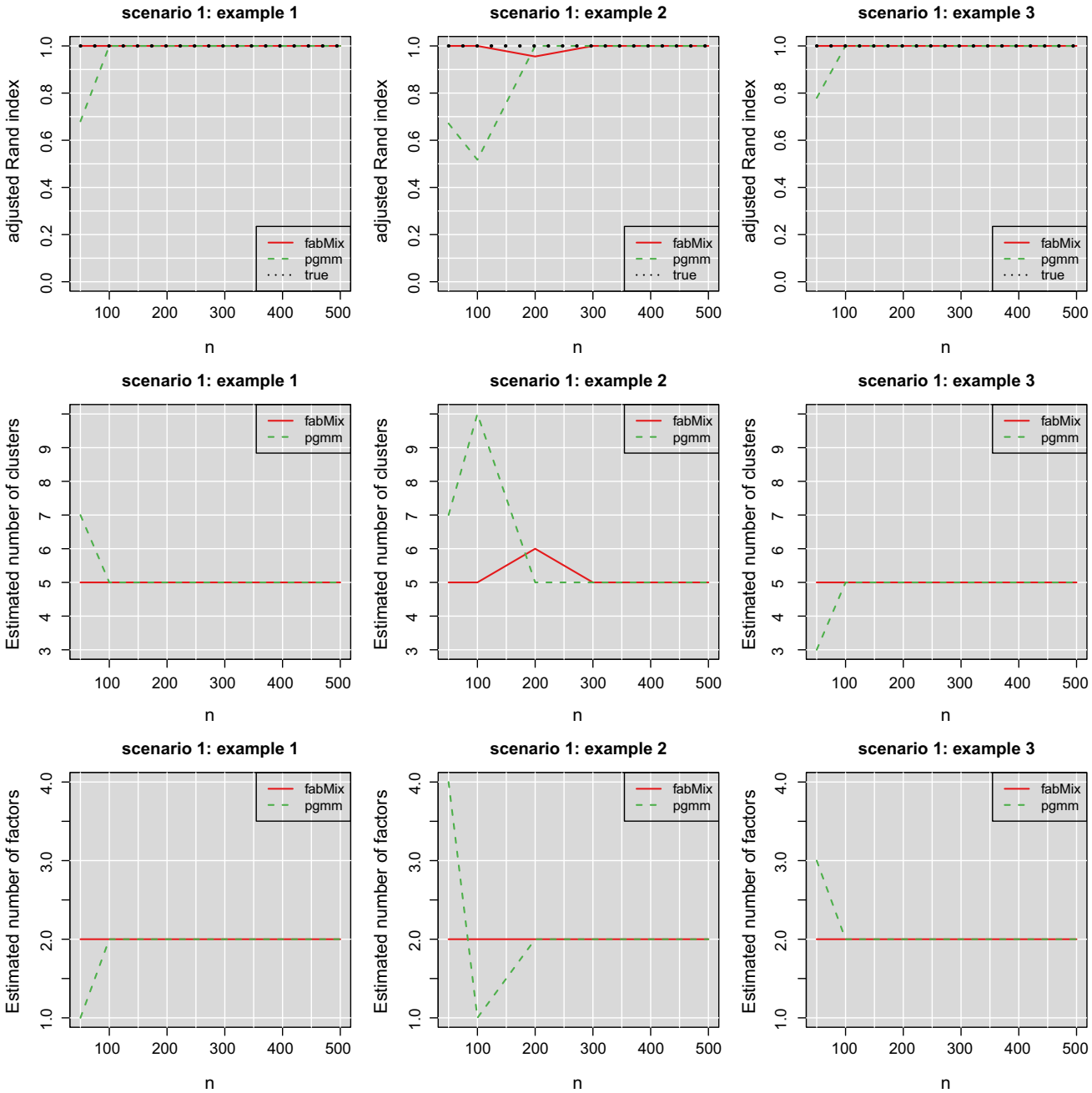


Fig. 9 Adjusted Rand index (first row), estimated number of clusters (second row) and estimated number of factors (third row) for simulated data according to scenario 1 with unequal cluster sizes and increasing sample size. The dotted line in the first row corresponds to the adjusted Rand index between the ground truth and the clustering of the data when applying the Maximum A Posteriori rule using the parameter values that generated the data (C.36). For all examples, the true number of clusters and factors is equal to 5 and 2, respectively

tively. In every single case, the estimate arising from `fabMix` is at least as close as the estimate arising from `pgmm` to the corresponding true value. Note, however, that in example 1 of scenario 2 both methods detect a smaller number of factors (2 instead of 3 factors). In all other cases, we observe that as the sample size increases both methods infer the "true" number of factors.

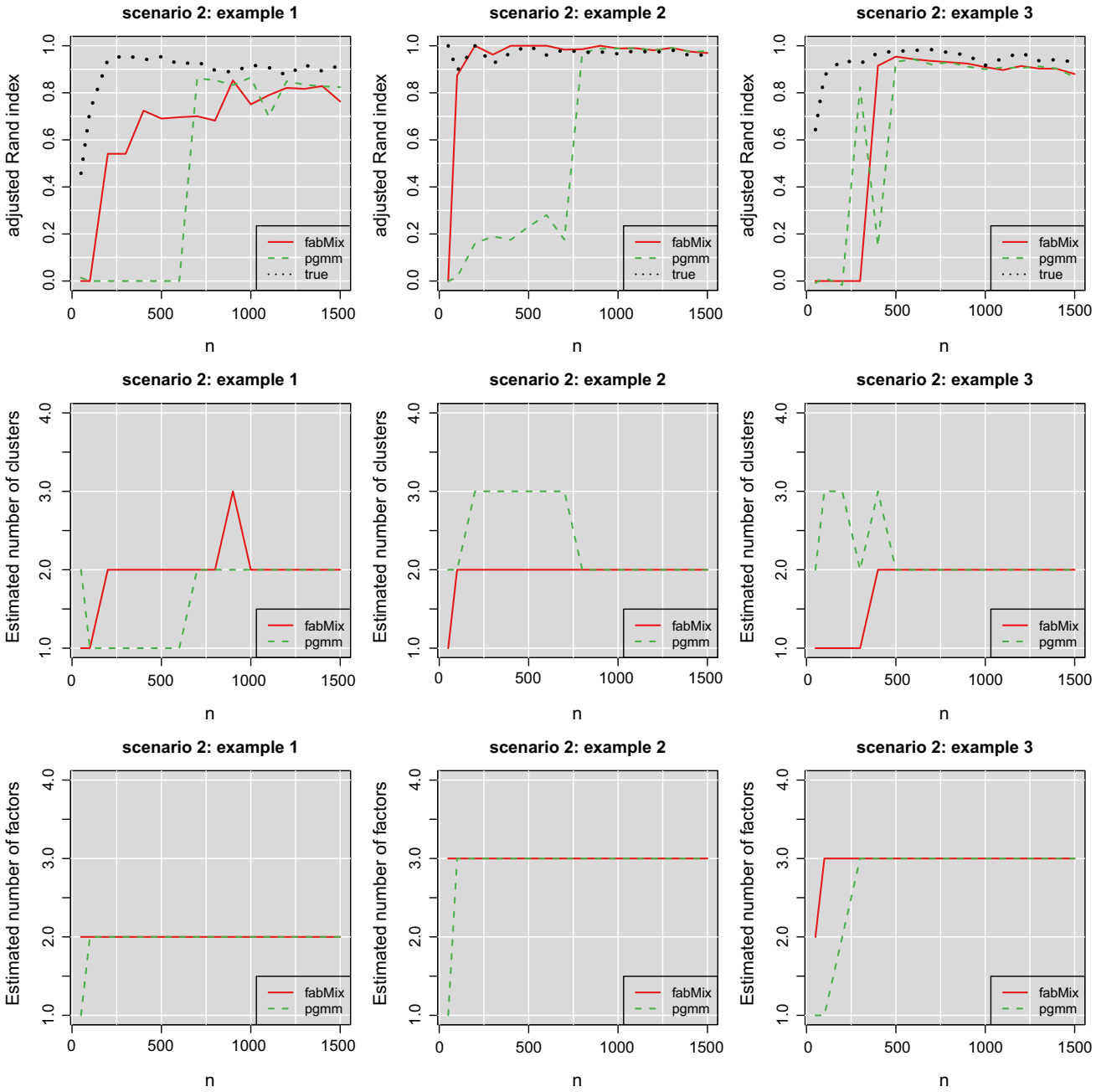


Fig. 10 Adjusted Rand index (1st row), estimated number of clusters (second row) and estimated number of factors (third row) for simulated data according to scenario 2 with unequal cluster sizes and increasing sample size. The dotted line in the first row corresponds to the adjusted Rand index between the ground truth and the clustering of the data when applying the Maximum A Posteriori rule using the parameter values that generated the data (C.36). For all examples, the true number of clusters and factors is equal to 2 and 3, respectively

References

Altekar, G., Dwarkadas, S., Huelsenbeck, J.P., Ronquist, F.: Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. Bioinformatics **20**(3), 407–415 (2004). https://doi.org/10.1093/bioinformatics/btg427

Bartholomew, D.J., Knott, M., Moustaki, I.: Latent Variable Models and Factor Analysis: A Unified Approach, vol. 904. Wiley, Hoboken (2011)

Bhattacharya, A., Dunson, D.B.: Sparse Bayesian infinite factor models. Biometrika **98**(2), 291–306 (2011)

Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth International Group, Belmont, CA (1984)

Celeux, G., Hurn, M., Robert, C.P.: Computational and inferential difficulties with mixture posterior distributions. J. Am. Stat. Assoc. **95**(451), 957–970 (2000a). https://doi.org/10.1080/01621459.2000.10474285

Celeux, G., Hurn, M., Robert, C.P.: Computational and inferential difficulties with mixture posterior distributions. J. Am. Stat. Assoc. **95**(451), 957–970 (2000b)

Cho, R.J., Campbell, M.J., Winzeler, E.A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T.G., Gabrielian, A.E., Landsman, D., Lockhart, D.J., Davis, R.W.: A genome-wide transcriptional analysis of the mitotic cell cycle. Molecular Cell **2**(1), 65–73 (1998). https://doi.org/10.1016/S1097-2765(00)80114-8

Conti, G., Frühwirth-Schnatter, S., Heckman, J.J., Piatek, R.: Bayesian exploratory factor analysis. J. Econom. **183**(1), 31–57 (2014)

Dellaportas, P., Papageorgiou, I.: Multivariate mixtures of normals with unknown number of components. Stat. Comput. **16**(1), 57–68 (2006)

Dempster, A.P., Laird, N.M., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm (with discussion). J. R. Stat. Soc. B **39**, 1–38 (1977)

Eddelbuettel, D., François, R.: Rcpp: seamless R and C++ integration. J. Stat. Softw. **40**(8), 1–18 (2011). https://doi.org/10.18637/jss.v040.i08

Eddelbuettel, D., Sanderson, C.: Rcpparmadillo: accelerating R with high-performance C++ linear algebra. Comput. Stat. Data Anal. **71**, 1054–1063 (2014). https://doi.org/10.1016/j.csda.2013.02.005

Ferguson, T.S.: A Bayesian analysis of some nonparametric problems. Ann. Stat. **1**(2), 209–230 (1973)

Fokoué, E., Titterington, D.: Mixtures of factor analysers. Bayesian estimation and inference by stochastic simulation. Mach. Learn. **50**(1), 73–94 (2003)

Forina, M., Armanino, C., Castino, M., Ubigli, M.: Multivariate data analysis as a discriminating method of the origin of wines. Vitis **25**(3), 189–201 (1986)

Fraley, C., Raftery, A.E.: Model-based clustering, discriminant analysis and density estimation. J. Am. Stat. Assoc. **97**, 611–631 (2002)

Frühwirth-Schnatter, S., Malsiner-Walli, G.: From here to infinity: sparse finite versus Dirichlet process mixtures in model based clustering. Adv. Data Anal. Classif. **13**, 33–64 (2019)

Gaujoux, R.: doRNG: Generic Reproducible Parallel Backend for 'foreach' Loops. https://CRAN.R-project.org/package=doRNG, r package version 1.7.1 (2018)

Gelfand, A., Smith, A.: Sampling-based approaches to calculating marginal densities. J. Am. Stat. Assoc. **85**, 398–409 (1990)

Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **PAMI–6**(6), 721–741 (1984). https://doi.org/10.1109/TPAMI.1984.4767596

Geweke, J., Zhou, G.: Measuring the pricing error of the arbitrage pricing theory. Rev. Financ. Stud. **9**(2), 557–587 (1996). https://doi.org/10.1093/rfs/9.2.557

Geyer, C.J.: Markov chain Monte Carlo maximum likelihood. In: Proceedings of the 23rd symposium on the interface, interface foundation, Fairfax Station, Va, pp. 156–163 (1991)

Geyer, C.J., Thompson, E.A.: Annealing Markov chain Monte Carlo with applications to ancestral inference. J. Am. Stat. Assoc. **90**(431), 909–920 (1995). https://doi.org/10.1080/01621459.1995.10476590

Ghahramani, Z., Hinton, G.E., et al.: The em algorithm for mixtures of factor analyzers. Tech. rep., Technical Report CRG-TR-96-1, University of Toronto (1996)

Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika **82**(4), 711–732 (1995)

Hager, W.W.: Updating the inverse of a matrix. SIAM Rev. **31**(2), 221–239 (1989)

Ihaka, R., Gentleman, R.: R: a language for data analysis and graphics. J. Comput. Graph. Stat. **5**(3), 299–314 (1996). https://doi.org/10.1080/10618600.1996.10474713

Kim, J.O., Mueller, C.W.: Factor Analysis: Statistical Methods and Practical Issues, vol. 14. Sage, Thousand Oaks (1978)

Ledermann, W.: On the rank of the reduced correlational matrix in multiple-factor analysis. Psychometrika **2**(2), 85–93 (1937)

Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml. Accessed 15 Sept 2018

Malsiner Walli, G., Frühwirth-Schnatter, S., Grün, B.: Model-based clustering based on sparse finite Gaussian mixtures. Stat. Comput. **26**, 303–324 (2016)

Malsiner Walli, G., Frühwirth-Schnatter, S., Grün, B.: Identifying mixtures of mixtures using bayesian estimation. J. Comput. Graph. Stat. **26**, 285–295 (2017)

Marin, J., Mengersen, K., Robert, C.: Bayesian modelling and inference on mixtures of distributions. Handb. Stat. **25**(1), 577–590 (2005)

Mavridis, D., Ntzoufras, I.: Stochastic search item selection for factor analytic models. Br. J. Math. Stat. Psychol. **67**(2), 284–303 (2014). https://doi.org/10.1111/bmsp.12019

McLachlan, J., Peel, D.: Finite Mixture Models. Wiley, New York (2000)

McNicholas, P.D., ElSherbiny, A., Jampani, R.K., McDaid, A.F., Murphy, B., Banks, L.: pgmm: Parsimonious Gaussian Mixture Models. http://CRAN.R-project.org/package=pgmm, R package version 1.2.3 (2015)

McNicholas, P.D.: Mixture Model-Based Classification. CRC Press, Boca Raton (2016)

McNicholas, P.D., Murphy, T.B.: Parsimonious Gaussian mixture models. Stat. Comput. **18**(3), 285–296 (2008)

McNicholas, P.D., Murphy, T.B.: Model-based clustering of microarray expression data via latent Gaussian mixture models. Bioinformatics **26**(21), 2705 (2010)

McNicholas, P.D., Murphy, T.B., McDaid, A.F., Frost, D.: Serial and parallel implementations of model-based clustering via parsimonious Gaussian mixture models. Comput. Stat. Data Anal. **54**(3), 711–723 (2010)

McParland, D., Phillips, C.M., Brennan, L., Roche, H.M., Gormley, I.C.: Clustering high-dimensional mixed data to uncover sub-phenotypes: joint analysis of phenotypic and genotypic data. Stat. Med. **36**(28), 4548–4569 (2017)

Meng, X.L., Van Dyk, D.: The EM algorithm—an old folk-song sung to a fast new tune. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **59**(3), 511–567 (1997)

Murphy, K., Gormley, I.C., Viroli, C.: Infinite mixtures of infinite factor analysers (2019). arXiv preprint arXiv:1701.07010

Neal, R.M.: Markov chain sampling methods for Dirichlet process mixture models. J. Comput. Graph. Stat. **9**(2), 249–265 (2000)

Nobile, A., Fearnside, A.T.: Bayesian finite mixtures with an unknown number of components: the allocation sampler. Stat. Comput. **17**(2), 147–162 (2007). https://doi.org/10.1007/s11222-006-9014-7

Papastamoulis, P.: fabMix: Overfitting Bayesian mixtures of factor analyzers with parsimonious covariance and unknown number of components (2018a). http://CRAN.R-project.org/package=fabMix, R package version 4.5

Papastamoulis, P.: Handling the label switching problem in latent class models via the ECR algorithm. Commun. Stat. Simul. Comput. **43**(4), 913–927 (2014)

Papastamoulis, P.: label.switching: an R package for dealing with the label switching problem in MCMC outputs. J. Stat. Softw. **69**(1), 1–24 (2016)

Papastamoulis, P.: Overfitting Bayesian mixtures of factor analyzers with an unknown number of components. Comput. Stat. Data

Anal. **124**, 220–234 (2018b). https://doi.org/10.1016/j.csda.2018.03.007

Papastamoulis, P., Iliopoulos, G.: Reversible jump MCMC in mixtures of normal distributions with the same component means. Comput. Stat. Data Anal. **53**(4), 900–911 (2009)

Papastamoulis, P., Iliopoulos, G.: An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. J. Comput. Graph. Stat. **19**, 313–331 (2010)

Papastamoulis, P., Iliopoulos, G.: On the convergence rate of random permutation sampler and ECR algorithm in missing data models. Methodol. Comput. Appl. Probab. **15**(2), 293–304 (2013). https://doi.org/10.1007/s11009-011-9238-7

Papastamoulis, P., Rattray, M.: BayesBinMix: an R package for model based clustering of multivariate binary data. R J. **9**(1), 403–420 (2017)

Plummer, M., Best, N., Cowles, K., Vines, K.: CODA: convergence diagnosis and output analysis for MCMC. R News **6**(1), 7–11 (2006)

R Core Team (2016) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/, ISBN 3-900051-07-0

Rand, W.M.: Objective criteria for the evaluation of clustering methods. J. Am. Stat. Assoc. **66**(336), 846–850 (1971)

Redner, R.A., Walker, H.F.: Mixture densities, maximum likelihood and the EM algorithm. SIAM Rev. **26**(2), 195–239 (1984)

Revolution Analytics and Steve Weston (2014) foreach: Foreach looping construct for R. http://CRAN.R-project.org/package=foreach, r package version 1.4.2

Revolution Analytics and Steve Weston (2015) doParallel: Foreach Parallel Adaptor for the 'parallel' Package. http://CRAN.R-project.org/package=doParallel, r package version 1.0.10

Richardson, S., Green, P.J.: On Bayesian analysis of mixtures with an unknown number of components. J. R. Stat. Soc. Ser. B **59**(4), 731–758 (1997)

Rousseau, J., Mengersen, K.: Asymptotic behaviour of the posterior distribution in overfitted mixture models. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **73**(5), 689–710 (2011)

Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)

Scrucca, L., Fop, M., Murphy, T.B., Raftery, A.E.: mclust 5: clustering, classification and density estimation using Gaussian finite mixture models. R J. **8**(1), 205–233 (2017)

Stephens, M.: Bayesian analysis of mixture models with an unknown number of components—an alternative to reversible jump methods. Ann. Stat. **28**(1), 40–74 (2000)

Streuli, H.: Der heutige stand der kaffeechemie. In: 6th International Colloquium on Coffee Chemisrty, Association Scientifique International du Cafe, Bogata, Columbia, pp. 61–72 (1973)

Tipping, M.E., Bishop, C.M.: Mixtures of probabilistic principal component analyzers. Neural Comput. **11**(2), 443–482 (1999)

van Havre, Z., White, N., Rousseau, J., Mengersen, K.: Overfitting Bayesian mixture models with an unknown number of components. PLoS ONE **10**(7), 1–27 (2015)

Yeung, K.Y., Fraley, C., Murua, A., Raftery, A.E., Ruzzo, W.L.: Model-based clustering and data transformations for gene expression data. Bioinformatics **17**(10), 977–987 (2001). https://doi.org/10.1093/bioinformatics/17.10.977