BayesBinMix: an R Package for Model Based Clustering of Multivariate Binary Data

by Panagiotis Papastamoulis and Magnus Rattray

Abstract The **BayesBinMix** package offers a Bayesian framework for clustering binary data with or without missing values by fitting mixtures of multivariate Bernoulli distributions with an unknown number of components. It allows the joint estimation of the number of clusters and model parameters using Markov chain Monte Carlo sampling. Heated chains are run in parallel and accelerate the convergence to the target posterior distribution. Identifiability issues are addressed by implementing label switching algorithms. The package is demonstrated and benchmarked against the Expectation-Maximization algorithm using a simulation study as well as a real dataset.

Introduction

Clustering data is a fundamental task in a wide range of applications and finite mixture models are widely used for this purpose (McLachlan and Peel, 2000; Marin et al., 2005; Frühwirth-Schnatter, 2006). In this paper our attention is focused on clustering binary datasets. A variety of studies aims at identifying patterns in binary data including, but not limited to, voting data (Ilin, 2012), text classification (Juan and Vidal, 2002), handwritten digit recognition (Al-Ohali et al., 2003), medical research (Sun et al., 2007), animal classification Li (2005) and genetics (Abel et al., 1993).

Throughout this paper the term *cluster* is used as a synonym of *mixture component*. Finite mixture models can be estimated under a frequentist approach using the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). However, the likelihood surface of a mixture model can exhibit many local maxima and it is well known that the EM algorithm may fail to converge to the main mode if it is initialized from a point close to a minor mode. Moreover, under a frequentist approach, the selection of the number of clusters is not straightforward: a mixture model for each possible value of number of clusters is fitted and then the optimal one is selected according to penalized likelihood criteria such as the Bayesian information criterion (Schwarz, 1978) or the integrated complete likelihood criterion (Biernacki et al., 2000). The reader is also referred to Lindsay (1995); Böhning (2000) for non-parametric likelihood estimation of a mixture model.

On the other hand, the Bayesian framework allows to put a prior distribution on both the number of clusters as well as the model parameters and then (approximately) sample from the joint posterior distribution using Markov chain Monte Carlo (MCMC) algorithms (Richardson and Green, 1997; Stephens, 2000a; Nobile and Fearnside, 2007; White et al., 2016). However this does not mean that the Bayesian approach is not problematic. In general, vanilla MCMC algorithms may require a very large number of iterations to discover the high posterior density areas and/or sufficiently explore the posterior surface due to the existence of minor modes. Second, identifiability issues arise due to the label switching phenomenon (Redner and Walker, 1984) which complicate the inference procedure.

The **BayesBinMix** package explicitly takes care of the previously mentioned problems for the problem of clustering multivariate binary data:

1. Allows missing values in the observed data,
2. Performs MCMC sampling for estimating the posterior distribution of the number of clusters and model parameters,
3. Produces a rapidly mixing MCMC sample by running parallel heated chains which can switch states,
4. Post-processes the generated MCMC sample and produces meaningful posterior mean estimates using state of the art algorithms to deal with label switching.

The rest of the paper is organised as follows. The mixture model is presented in Section 2.2. Its prior assumptions and the corresponding hierarchical model is introduced in Section 2.3. The basic MCMC scheme is detailed in Section 2.4.1. Section 2.4.2 deals with post-processing the generated MCMC sample in order to overcome identifiability issues due to the label switching problem. Finally, the basic sampler is embedded in a Metropolis-coupled MCMC algorithm as described in Section 2.4.3. The main function of the package is described in Section 2.5. Simulated and real datasets are analyzed in Sections 2.6.1 and 2.6.2, respectively.

Model

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ denote a random sample of multivariate binary data, where $x_i = (x_{i1}, \ldots, x_{id})$; $d > 1$, for $i = 1, \ldots, n$. Assume that the observed data has been generated from a mixture of independent Bernoulli distributions, that is,

$$\begin{aligned} x_i &\sim \sum_{k=1}^{K} p_k \prod_{j=1}^{d} f\left(x_{ij}; \theta_{kj}\right) \\ &= \sum_{k=1}^{K} p_k \prod_{j=1}^{d} \theta_{kj}^{x_{ij}} \left(1-\theta_{kj}\right)^{1-x_{ij}} \mathbb{I}_{\{0,1\}}(x_{ij}), \end{aligned} \tag{1}$$

independently for $i = 1, \ldots, n$, where $\theta_{kj} \in \Theta = (0,1)$ denotes the probability of success for the k-th cluster and j-th response for $k = 1, \ldots, K$; $j = 1, \ldots, d$, $\boldsymbol{p} = (p_1, \ldots, p_K) \in \mathcal{P}_{K-1} = \{p_k; k = 1, \ldots, K-1 : 0 \leqslant p_k \leqslant 1; 0 \leqslant p_K = 1 - \sum_{k=1}^{K-1} p_k\}$ corresponds to the vector of mixture weights and $\mathbb{I}_A(\cdot)$ denotes the indicator function of a (measurable) subset A.

It is straightforward to prove that the variance-covariance matrix of a mixture of independent Bernoulli distributions is not diagonal (see e.g. Bishop (2006)), which is the case for a collection of independent Bernoulli distributions. Therefore, the mixture model exhibits richer covariance structure thus it can prove useful to discover correlations in heterogeneous multivariate binary data.

The observed likelihood of the model is written as

$$L_K(\boldsymbol{p}, \boldsymbol{\theta}; \boldsymbol{x}) = \prod_{i=1}^{n} \sum_{k=1}^{K} p_k \prod_{j=1}^{d} \theta_{kj}^{x_{ij}} \left(1-\theta_{kj}\right)^{1-x_{ij}}, \quad (\boldsymbol{p}, \boldsymbol{\theta}) \in \mathcal{P}_{K-1} \times \Theta^{Kd} \tag{2}$$

where $\boldsymbol{x} \in \mathcal{X}^n = \{0,1\}^{nd}$. For any fixed value of K, Equation (2) can be further decomposed by considering that observation i has been generated from the z_i-th mixture component, that is,

$$x_i | z_i = k \sim \prod_{j=1}^{d} f\left(x_{ij}; \theta_{kj}\right), \quad \text{independent for } i = 1, \ldots, n. \tag{3}$$

Note that the allocation variables $z_i \in \mathcal{Z}_K = \{1, \ldots, K\}$; $i = 1, \ldots, n$ are unobserved, so they are treated as missing data. Assume that

$$P\left(z_i = k | \boldsymbol{p}, K\right) = p_k, \quad k = 1, \ldots, K \tag{4}$$

and furthermore that (x_i, z_i) are independent for $i = 1, \ldots, n$. Data augmentation (Tanner and Wong, 1987) considers jointly the *complete data* $\{(x_i, z_i); i = 1, \ldots, n\}$ and it is a standard technique exploited both by the Expectation-Maximization algorithm (Dempster et al., 1977) as well as the Gibbs sampler (Gelfand and Smith, 1990). The *complete likelihood* is defined as

$$\begin{aligned} L_K^c\left(\boldsymbol{p}, \boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{z}\right) &= \prod_{i=1}^{n} p_{z_i} \prod_{j=1}^{d} \theta_{z_i j}^{x_{ij}} \left(1-\theta_{z_i j}\right)^{1-x_{ij}} \\ &= \prod_{k=1}^{K} p_k^{n_k} \prod_{j=1}^{d} \theta_{kj}^{s_{kj}} \left(1-\theta_{kj}\right)^{n_k - s_{kj}}, \quad (\boldsymbol{p}, \boldsymbol{\theta}) \in \mathcal{P}_{K-1} \times \Theta^{Kd} \end{aligned} \tag{5}$$

where $n_k = \sum_{i=1}^{n} \mathbb{I}(z_i = k)$ and $s_{kj} = \sum_{i=1}^{n} \mathbb{I}(z_i = k) x_{ij}$, $k = 1, \ldots, K$; $j = 1, \ldots, d$, for a given $(\boldsymbol{x}, \boldsymbol{z}) \in \mathcal{X}^n \times \mathcal{Z}_K^n$.

Prior assumptions

Note that the quantities $\boldsymbol{p}, \boldsymbol{\theta}, \boldsymbol{z}$ are defined conditionally on K. For convenience we will assume that $K \in \mathcal{K} = \{1, \ldots, K_{\max}\}$, where $K_{\max}$ denotes an upper bound on the number of clusters. Hence, under a model-based clustering point of view, the vector

$$(K, \boldsymbol{p}, \boldsymbol{\theta}, \boldsymbol{z}) \in \mathcal{A} := \mathcal{K} \times \mathcal{P}_{K-1} \times \Theta^{Kd} \times \mathcal{Z}_K^n$$

summarizes all unknown parameters that we wish to infer.

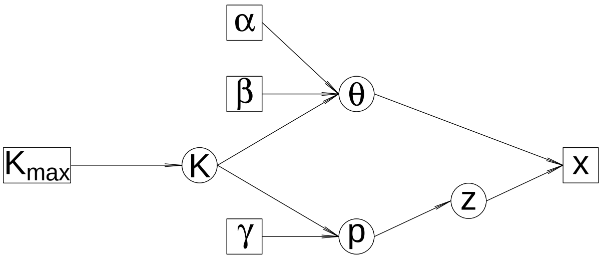


Figure 1: Representation of the hierarchical model (9) as a directed acyclic graph. Squares and circles denote observed/fixed and unknown variables, respectively.

The following prior assumptions are imposed

$$K \sim \text{Discrete}\{1,\ldots,K_{\max}\} \tag{6}$$
$$\boldsymbol{p}|K \sim \text{Dirichlet}(\gamma_1,\ldots,\gamma_K) \tag{7}$$
$$\boldsymbol{\theta}_{kj}|K \sim \text{Beta}(\alpha,\beta), \tag{8}$$

independent for $k = 1,\ldots,K; j = 1,\ldots,d$. The discrete distribution in Equation (6) can be either a Uniform or a Poisson distribution with mean $\lambda = 1$ truncated on the set $\{1,\ldots,K_{\max}\}$. Equations (7) and (8) correspond to typical prior distributions for the mixture weights and success probabilities, that furthermore enjoy conjugacy properties. Typically, we set $\gamma_1 = \ldots = \gamma_K = \gamma > 0$ so that the prior assumptions do not impose any particular information that separates the mixture components between them, which is also a recommended practice in mixture modelling.

According to Equations (4), (5), (6), (7) and (8), the joint probability density function of the model is

$$f(\boldsymbol{x}, K, \boldsymbol{z}, \boldsymbol{p}, \boldsymbol{\theta}) = f(\boldsymbol{x}|K,\boldsymbol{z},\boldsymbol{\theta})\, f(\boldsymbol{z}|K,\boldsymbol{p})\, f(\boldsymbol{p}|K)\, f(\boldsymbol{\theta}|K)\, f(K), \tag{9}$$

and its graphical representation is shown in Figure 1.

Inference

Allocation sampler

Let $C_K = \frac{\Gamma(\sum_{k=1}^{K}\gamma_k)}{\prod_{k=1}^{K}\Gamma(\gamma_k)}\left\{\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\right\}^{Kd}$. From Equation (9) the joint posterior distribution of $(K,\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{z})$ can be expressed as:

$$f(\boldsymbol{\theta},\boldsymbol{z},\boldsymbol{p},K|\boldsymbol{x}) \propto C_K f(K) \prod_{k=1}^{K}\left\{p_k^{n_k+\gamma_k-1}\prod_{j=1}^{d}\theta_{kj}^{\alpha+s_{kj}-1}(1-\theta_{kj})^{\beta+n_k-s_{kj}-1}\right\}\mathbb{I}_{\mathcal{A}}(K,\boldsymbol{p},\boldsymbol{\theta},\boldsymbol{z}). \tag{10}$$

From the last expression it is straightforward to derive the full conditional distributions of the component specific parameters and latent allocation variables as follows:

$$\boldsymbol{p}|K,\boldsymbol{z} \sim \text{Dirichlet}\left(\gamma_1+n_1,\ldots,\gamma_K+n_K\right) \tag{11}$$
$$\theta_{kj}|K,\boldsymbol{z},\boldsymbol{x} \sim \text{Beta}\left(\alpha+s_{kj},\beta+n_k-s_{kj}\right) \tag{12}$$
$$\mathrm{P}\left(z_i = k|K,\boldsymbol{x}_i,\boldsymbol{p},\boldsymbol{\theta}\right) \propto p_k\prod_{j=1}^{d}\theta_{kj}^{x_{ij}}\left(1-\theta_{kj}\right)^{1-x_{ij}}, \quad k=1,\ldots,K,$$

independent for $i = 1,\ldots,n; k = 1,\ldots,K; j = 1,\ldots,d$.

A general framework for updating the number of mixture components (K) is given by trans-dimensional MCMC approaches, such as the reversible jump MCMC (Green, 1995; Richardson and Green, 1997; Papastamoulis and Iliopoulos, 2009) or the Birth-Death MCMC (Stephens, 2000a) methodologies. However, the conjugate prior assumptions used for the component specific parameters $(\boldsymbol{p},\boldsymbol{\theta})$ allow us to use simpler techniques by integrating those parameters out from the model and perform collapsed sampling (Liu, 1994) on the space of $(K,\boldsymbol{z})$. We use the allocation sampler (Nobile and Fearnside, 2007) which introduced this sampling scheme for parametric families such that conjugate prior distributions exist and also applied it in the specific context of mixtures of normal distributions. This approach was recently followed by White et al. (2016) which also allowed for variable selection.

Let $\mathcal{A}_0 = \Theta^{Kd} \times \mathcal{P}_{K-1}$ and $\mathcal{A}_1 = \mathcal{K} \times \{1,\ldots,K\}^n$. Integrating out $(\boldsymbol{\theta}, \boldsymbol{p})$ from (10) we obtain

$$
\begin{aligned}
f(K, \boldsymbol{z}|\boldsymbol{x}) &= \int\limits_{\mathcal{A}_0} f(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{p}, K|\boldsymbol{x}) \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{p} \\
&\propto C_K f(K) \mathbb{I}_{\mathcal{A}_1}(K, \boldsymbol{z}) \int\limits_{\mathcal{A}_0} \prod_{k=1}^{K} p_k^{n_k+\gamma_k-1} \prod_{j=1}^{d} \theta_{kj}^{\alpha+s_{kj}-1} \left(1-\theta_{kj}\right)^{\beta+n_k-s_{kj}-1} \mathrm{d}\boldsymbol{\theta}\mathrm{d}\boldsymbol{p} \\
&\propto C_K f(K) \frac{\prod_{k=1}^{K} \Gamma(n_k+\gamma_k)}{\Gamma\left(n+\sum_{k=1}^{K}\gamma_k\right)} \prod_{k=1}^{K}\prod_{j=1}^{d} \frac{\Gamma\left(\alpha+s_{kj}\right)\Gamma\left(\beta+n_k-s_{kj}\right)}{\Gamma(\alpha+\beta+n_k)} \mathbb{I}_{\mathcal{A}_1}(K, \boldsymbol{z}). \qquad (13)
\end{aligned}
$$

Let now $\boldsymbol{z}_{[-i]} = \{z_1, \ldots, z_{i-1}, z_{i+1}, \ldots, z_n\}$ and also define the following quantities, for $i = 1, \ldots, n$:

$$
\begin{aligned}
n_k^{[i]} &= \sum_{h \neq i} \mathbb{I}(z_h = k), k = 1, \ldots, K \\
s_{kj}^{[i]} &= \sum_{h \neq i} \mathbb{I}(z_h = k) x_{hj}, k = 1, \ldots, K; j = 1, \ldots, d \\
A_1^{[i]} &= \{j = 1, \ldots, d : x_{ij} = 1\} \\
A_0^{[i]} &= \{j = 1, \ldots, d : x_{ij} = 0\}.
\end{aligned}
$$

From Equation (13), the (collapsed) conditional posterior distribution of z_i is

$$
\mathrm{P}\left(z_i = k|\boldsymbol{z}_{[-i]}, K, \boldsymbol{x}\right) \propto \frac{n_k^{[i]} + \gamma_k}{\left(\alpha+\beta+n_k^{[i]}\right)^d} \prod_{j \in A_1^{[i]}} \left(\alpha + s_{kj}^{[i]}\right) \prod_{j \in A_0^{[i]}} \left(\beta + n_k - s_{kj}^{[i]}\right), \qquad (14)
$$

$k = 1, \ldots, K; i = 1, \ldots, n$.

It is well known that draws from the conditional distributions in Equation (14) exhibit strong serial correlation, slowing down the convergence of the MCMC sampler. The mixing can be improved by proposing simultaneous updates of blocks of $\boldsymbol{z}|K$, by incorporating proper Metropolis-Hastings moves on $\boldsymbol{z}|K$. Following Nobile and Fearnside (2007), we also propose jumps to configurations that massively update the allocation vector as follows:

1. **Move 1:** select two mixture components and propose a random reallocation of the assigned observations.
2. **Move 2:** select two mixture components and propose to move a randomly selected subset of observations from the 1st to the 2nd one.
3. **Move 3:** select two mixture components and propose a reallocation of the assigned observations according to the full conditional probabilities given the already processed ones.

Each move is accepted according to the corresponding Metropolis-Hastings acceptance probability, see Nobile and Fearnside (2007) for details.

The final step of the allocation sampler is to update the number of clusters (K). According to Nobile and Fearnside (2007), this is achieved by performing a Metropolis-Hastings type move, namely a pair of absorption/ejection moves which decrease/increase K, respectively. Assume that the current state of chain is $\{K, \boldsymbol{z}\}$. The following pseudocode describes the Absorption/Ejection step:

```
1. Attemp ejection with probability $p_K^e$, where $p_K^e = 1/2$, $K = 2, \ldots, K_{\max} - 1$, $p_1^e = 1$ and $p_{K_{\max}}^e = 0$. Otherwise, an absorption move is attempted.
2. Suppose that an ejection is attempted. The candidate state is $\{K', \boldsymbol{z}'\}$ with $K' = K + 1$.
    (a) Propose reallocation of observations assigned to the ejecting component between itself and the ejected component according to the Beta$(\tilde{\alpha}, \tilde{\alpha})$ distribution.
    (b) Accept the candidate state with probability $\min\{1, R\}$ where
        $R = R(\tilde{\alpha}) = \frac{f(K', \boldsymbol{z}'|\boldsymbol{x})}{f(K, \boldsymbol{z}|\boldsymbol{x})} \frac{\mathrm{P}(\{K', \boldsymbol{z}'\} \to \{K, \boldsymbol{z}\})}{\mathrm{P}(\{K, \boldsymbol{z}\} \to \{K', \boldsymbol{z}'\})}$   (15)
3. If an absorption is attempted:
    (a) all observations allocated to the absorbed component are reallocated to the absorbing component.
```

(b) the candidate state is accepted with probability $\min\{1, 1/R(\tilde{\alpha})\}$.

The parameter $\tilde{\alpha}$ is chosen in a way that ensures that the probability of ejecting an empty component is sufficiently large. For full details the reader is referred to Nobile and Fearnside (2007).

The allocation sampler for mixtures of multivariate Bernoulli distributions is summarized in the following algorithm.

Algorithm 1 (Allocation sampler for Bernoulli mixtures) *Given an initial state $\{K^{(0)}, \boldsymbol{z}^{(0)}\} \in \mathcal{A}_1$ iterate the following steps for $t = 1, 2, \ldots$*

1. *For $i = 1, \ldots, n$*
   - (a) *Compute $n_k^{[i]} = \sum_{h \neq i} \mathbb{I}(z_h = k)$, $s_{kj}^{[i]} = \sum_{h \neq i} \mathbb{I}(z_h = k) x_{hj}$, $k = 1, \ldots, K^{(t)}$; $j = 1, \ldots, d$.*
   - (b) *Update $z_i^{(t)} | \boldsymbol{z}_{[-i]}, \cdots$ according to Equation (14).*
2. *Propose Metropolis-Hastings moves M_1, M_2 and M_3 to update $\boldsymbol{z}^{(t)}$.*
3. *Propose an Absorption/Ejection move to update $\{K^{(t)}, \boldsymbol{z}^{(t)}\}$.*

Note in step *1.(a)*:

$$z_h = \begin{cases} z_h^{(t)}, & h < i \\ z_h^{(t-1)}, & h > i. \end{cases}$$

Finally, we mention that after the last step of Algorithm 1 we can also simulate the component-specific parameters $\boldsymbol{p}$ and $\boldsymbol{\theta}$ from their full conditional posterior distributions given in (11) and (12), respectively. Although this is not demanded in case that the user is only interested in inferring $K, \boldsymbol{z}|\boldsymbol{x}$, it will produce an (approximate) MCMC sample from the full posterior distribution of $K, \boldsymbol{p}, \boldsymbol{\theta}, \boldsymbol{z}|\boldsymbol{x}$. If the observed data contains missing entries an extra step is implemented in order to simulate the corresponding values. For this purpose we use the full conditional distribution derived from Equation (3), taking only into account the subset of $\{1, \ldots, d\}$ that contains missing values for a given $i = 1, \ldots, n$.

Label switching issue and identifiability

Label switching (Redner and Walker, 1984) is a well known identifiability problem occurring in MCMC outputs of mixture models, arising from the symmetry of the likelihood with respect to permutations of components' labels. A set of sufficient conditions under a general framework of missing data models that lead to label switching and its consequences is given in Papastamoulis and Iliopoulos (2013). If an MCMC sample exhibits label switching, the standard practice of estimating the posterior means and other parametric functions by ergodic averages becomes meaningless. In order to deal with this identifiability problem we have considered two versions of ECR algorithm (Papastamoulis and Iliopoulos, 2010; Papastamoulis, 2014; Rodríguez and Walker, 2014) as well as the KL algorithm (Stephens, 2000b). These algorithms are quite efficient and in most cases exhibit almost identical results, but ECR is significantly faster and computationally lightweight compared to KL. The implementation was performed in the R package **label.switching** (Papastamoulis, 2016).

Note here that in the case that $d = 1$, Equation (1) collapses to a single Bernoulli distribution. Hence, there are two types of identifiability issues in mixture models: the first one is related to the fact that the model is identifiable only up to a permutation of the parameters (label switching). The second one is strict non-identifiability which relates to the fact that for a mixture of discrete distributions (such as the multivariate Bernoulli) totally different parameter values can correspond to the same distribution. We are not dealing with this second source of identifiability problems since it has been empirically demonstrated that estimation can still produce meaningful results in practice (Carreira-Perpiñán and Renals, 2000). In addition, Allman et al. (2009) showed that finite mixtures of Bernoulli products are in fact generically identifiable despite their lack of strict identifiability.

Metropolis-coupled MCMC sampler

There are various strategies for improving MCMC sampling, see e.g. chapter 6 in Gilks et al. (1996). In this study, the Metropolis-coupled MCMC (MC^3) (Geyer, 1991; Geyer and Thompson, 1995; Altekar et al., 2004) strategy is adopted. An MC^3 sampler runs m chains with different posterior distributions $f_i(\xi); i = 1, \ldots, m$. The target posterior distribution corresponds to $i = 1$, that is, $f_1(\xi) = f(\xi)$, while the rest of them are chosen in a way that the mixing is improved. This is typically achieved by considering *heated* versions of the original target, that is, $f_i(\xi) = f(\xi)^{h_i}$ where $h_1 = 1$ and $0 < h_i < 1$

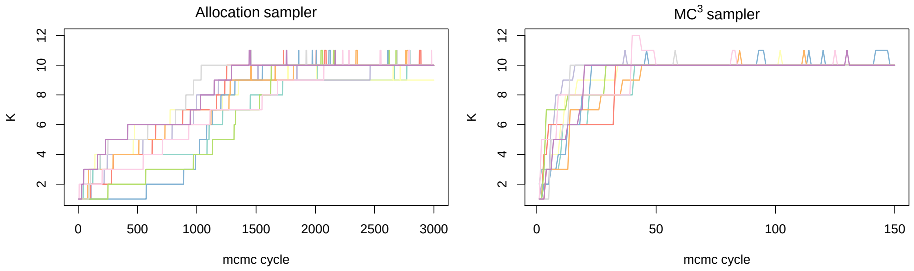


Figure 2: Trace of sampled values of the number of clusters (K) for 10 different runs, using one of the synthetic datasets of Section 2.6.1 with $K_{\text{true}} = 10$. Each run was initialized from $K = 1$ and every 10th iteration is displayed (labeled as MCMC cycle on the x axis) until most chains start to explore $K = 10$. Left: standard allocation sampler , right: MC^3 sampler (using 4 heated chains).

for $i = 2, \ldots, m$ represents the heat value of the chain. Note that when raising the posterior distribution to a power $0 < h_i < 1$ makes the modified posterior surface flatter, thus, easier to explore compared to $f(\xi)$. Only the chain that corresponds to the target posterior distribution is used for the posterior inference, however, after each iteration a proposal attempts to swap the states of two randomly chosen chains. This improves the mixing of the chain since it is possible that an accepted swap between the cold and a heated chain will make the former move to another mode.

Let $\xi_i^{(t)}$ denote the state of chain i at iteration t and that a swap between chains i and j is proposed. Note that in our setup $\xi = (K, \boldsymbol{z})$ and f is given in (13) (up to a normalizing constant). The proposed move is accepted with probability

$$\min\left\{1, \frac{f_i\left(\xi_j^{(t)}\right) f_j\left(\xi_i^{(t)}\right)}{f_i\left(\xi_i^{(t)}\right) f_j\left(\xi_j^{(t)}\right)}\right\} = \min\left\{1, \frac{f\left(\xi_j^{(t)}\right)^{h_i} f\left(\xi_i^{(t)}\right)^{h_j}}{f\left(\xi_i^{(t)}\right)^{h_i} f\left(\xi_j^{(t)}\right)^{h_j}}\right\}.$$

Figure 2 sketches the difference in convergence speed between the standard allocation sampler and an MC^3 sampler which are used to infer the same posterior distribution of the number of clusters. Although a single MCMC cycle of MC^3 is more expensive than a cycle of the allocation sampler, it is evident that the MC^3 sampler can recover the true number of clusters ($K_{\text{true}} = 10$) in a remarkably smaller number of iterations than the standard allocation sampler. In addition, the standard allocation sampler rarely switches between the symmetric modes of the posterior distribution, a fact which typically indicates poor mixing of MCMC samplers in mixture models (Marin et al., 2005). On the contrary, the MC^3 sampler produces a chain where label switching occurs in a rate proportional to the swap acceptance rate.

In order to take full advantage of computing power in modern-day computers, our MC^3 sampler utilizes parallel computing in multiple cores. This is achieved by running each chain in parallel using the R packages **foreach** (Revolution Analytics and Weston, 2014) and **doParallel** (Revolution Analytics and Weston, 2015). Every 10-th iteration a swap is proposed between a pair of chains.

Using package BayesBinMix

The main function of the **BayesBinMix** package is `coupledMetropolis`, with its arguments shown in Table 1. This function takes as input a binary data array (possibly containing missing values) and runs the allocation sampler for a series of heated chains which run in parallel while swaps between pairs of chains are proposed. In the case that the most probable number of mixture components is larger than 1, the label switching algorithms are applied.

As the function runs it prints some basic information on the screen 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 mainly consists of `"mcmc"` objects, a class imported from the **coda** package (Plummer et al., 2006). More specifically, the `coupledMetropolis()` function returns the objects detailed in Table 2. We note that this is just a subset of the full output of the sampler which consists of several additional quantities, such as the raw MCMC values corresponding to the whole set of generated values of K. Usually this information is not necessary to the average user, thus, it is saved to a separate set of files in the folder specified by `outPrefix`.

Argument	Description
Kmax	Maximum number of clusters (integer, at least equal to two).
nChains	Number of parallel (heated) chains.
heats	nChains-dimensional vector specifying the temperature of each chain: the 1st entry should always be equal to 1 and the rest of them lie on the set:$(0, 1]$.
binaryData	The observed binary data (array). Missing values are allowed as long as the corresponding entries are denoted as NA.
outPrefix	The name of the produced output folder. An error is thrown if the directory exists.
ClusterPrior	Character string specifying the prior distribution of the number of clusters. Available options: poisson or uniform. It defaults to the (truncated) Poisson distribution.
m	The number of MCMC cycles. At the end of each cycle a swap between a pair of heated chains is attempted. Each cycle consists of 10 iterations.
alpha	First shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.
beta	Second shape parameter of the Beta prior distribution (strictly positive). Defaults to 1.
gamma	Kmax-dimensional vector (positive) corresponding to the parameters of the Dirichlet prior of the mixture weights. Default value: rep(1,Kmax).
z.true	An optional vector of cluster assignments considered as the ground-truth clustering of the observations. It is only used to obtain a final permutation of the labels (after the label switching algorithms) in order to maximise the similarity between the resulting estimates and the real cluster assignments. Useful for simulations.
ejectionAlpha	Probability of ejecting an empty component. Defaults to 0.2.
burn	Optional integer denoting the number of MCMC cycles that will be discarded as burn-in period.

Table 1: Arguments of the coupledMetropolis() function.

Examples

In this section the usage of **BayesBinMix** package is described and various benchmarks are presented. At first we demonstrate a typical implementation on a single simulated dataset and inspect the simulated parameter values and estimates. Then we perform an extensive study on the number of estimated clusters and compare our findings to the **FlexMix** package (Leisch, 2004; Grün and Leisch, 2007, 2008). An application to a real dataset is provided next.

Simulation study

At first, a single simulated dataset is used in order to give a brief overview of the implementation. We simulated $n = 200$ observations from the multivariate Bernoulli mixture model (1). The true number of clusters is set to $K = 6$ and the dimensionality of the multivariate distribution is equal to $d = 100$. The mixture weights are drawn from a Dirichlet $\mathcal{D}(1, 1, 1, 1, 1, 1)$ distribution resulting in $(50, 46, 30, 36, 12, 26)$ generated observations from each cluster. For each cluster, true values for the probability of success were generated from a Uniform distribution, that is, $\theta_{kj} \sim \mathcal{U}(0, 1)$, independently for $k = 1, \ldots, K; j = 1, \ldots, d$. Furthermore, we introduce some missing values to the generated data: each row is allowed to contain missing values with probability 0.2: for such a row the total number of missing entries is drawn from the binomial distribution $B(100, 0.3)$. Finally, the observed data is saved to the 200×100 array x which contains a total of 1038 missing values corresponding to 34 rows.

We will run 4 parallel chains with the following temperatures: $(1, 0.8, 0.6, 0.4)$. Observe that the first chain should correspond to the actual posterior distribution, so its temperature equals to 1. Now apply the coupledMetropolis() function as follows.

```
> library('BayesBinMix')
> nChains <- 4
> heats <- seq(1, 0.4, length = nChains)

# using the truncated Poisson prior distribution on the number of clusters
> cm1 < - coupledMetropolis(Kmax = 20, nChains = nChains, heats = heats,
                binaryData = x, outPrefix = "bbm-poisson", ClusterPrior = "poisson",
```

Object	Description
`K.mcmc`	Object of class `"mcmc"` (see **coda** package) containing the simulated values (after burn-in) of the number of clusters for the cold chain.
`parameters.ecr.mcmc`	Object of class `"mcmc"` containing the simulated values (after burn-in) of θ_{kj} (probability of success per cluster k and feature j) and π_k (weight of cluster k) for $k = 1, \ldots, K_{\text{map}}; j = 1, \ldots, d$, where K_{map} denotes the most probable number of clusters. The output is reordered according to ECR algorithm.
`allocations.ecr.mcmc`	Object of class `"mcmc"` containing the simulated values (after burn-in) of z_i (allocation variables) for $i = 1, \ldots, n$, given $K = K_{\text{map}}$. The output is reordered according to ECR algorithm.
`classificationProbabilities.ecr`	Data frame of the reordered classification probabilities per observation after reordering the most probable number of clusters with the ECR algorithm.
`clusterMembershipPerMethod`	Data frame of the most probable allocation of each observation after reordering the MCMC sample which corresponds to the most probable number of clusters according to ECR, STEPHENS and ECR-ITERATIVE-1 methods.
`K.allChains`	`m`×`nChains` matrix containing the simulated values of the number of clusters (K) per chain.
`chainInfo`	Number of parallel chains, cycles, burn-in period and acceptance rate of swap moves.

Table 2: Output returned to the user of the `coupledMetropolis()` function.

```
                m = 1100, z.true = z.true, burn = 100)

# using the uniform prior distribution on the number of clusters
> cm2 <- coupledMetropolis(Kmax = 20, nChains = nChains, heats = heats,
                binaryData = x, outPrefix = "bbm-uniform", ClusterPrior = "uniform",
                m = 1100, z.true = z.true, burn = 100)
```

Note that we have called the function twice using either the truncated Poisson or the Uniform prior on the set $\{1, \ldots, 20\}$. The total number of MCMC cycles corresponds to $m = 1100$ and the first 100 cycles will be discarded as burn-in period. Recall that each cycle contains 10 usual MCMC iterations, so this is equivalent to keeping every 10th iteration of a chain with 11,000 iterations. Since we are interested to compare against the true values used to generate the data, we also supply `z.true` which contains the true allocation of each observation. It is only used for making the inferred clusters agree to the labelling of the true values and it has no impact on the MCMC or label switching algorithms.

Printing, summarizing and plotting the output

In this section we illustrate summaries of the basic output returned to the user, using only the run which corresponds to the Poisson prior distribution (`cm1`). The `print()` method of the package returns a basic summary of the fitted model:

```
> print(cm1)

* Run information:
      Number of parallel heated chains: 4
      Swap acceptance rate: 63.5%
      Total number of iterations: 11000
      Burn-in period: 1000
      Thinning: 10.

* Estimated posterior distribution of the number of clusters:

    6     7     8
0.971 0.026 0.003

* Most probable model: K = 6 with P(K = 6|data) = 0.971
* Estimated number of observations per cluster conditionally on K = 6 (3 label switching
  algorithms):
  STEPHENS ECR ECR.ITERATIVE.1
1       50  50              50
2       46  46              46
3       30  30              30
4       36  36              36
5       12  12              12
6       26  26              26

* Posterior mean of probability of success per feature and cluster (ECR algorithm):
        cluster_1 cluster_2 cluster_3 cluster_4  cluster_5 cluster_6
theta_1 0.33364058 0.8465393 0.7023264 0.3340989 0.08364937 0.8933767
theta_2 0.71919239 0.6653526 0.3227822 0.3982836 0.22369486 0.5936094
theta_3 0.49869339 0.2285653 0.3605507 0.3570447 0.07206039 0.1883581
theta_4 0.22360156 0.9148123 0.3359406 0.7889224 0.15476900 0.5924109
theta_5 0.01867034 0.8296381 0.8107050 0.1121773 0.78051586 0.1442368
   <+ 95 more rows>
```

Next we present summaries of the marginal posterior distributions of the (reordered) MCMC sample of parameters conditionally on the selected number of clusters. The reordered MCMC sample of θ_{kj} and p_k (after burn-in) is returned to the "mcmc" object parameters.ecr.mcmc. Hence we can use the summary() method of the **coda** package, which prints empirical means, standard deviations, as well the quantiles for each variable. This is done with the following command.

```
> summary(cm1$parameters.ecr.mcmc)

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

               Mean      SD  Naive SE Time-series SE
theta.1.1   0.33364 0.06504 0.0020874      0.0020874
...         ...     ...     ...            ...
theta.6.1   0.89338 0.05552 0.0017816      0.0017816
   <+ 99 blocks of 6 rows>
p.1         0.24663 0.02869 0.0009208      0.0009208
...         ...     ...     ...            ...
p.6         0.13270 0.02276 0.0007304      0.0007304

2. Quantiles for each variable:

                 2.5%      25%     50%     75%   97.5%
theta.1.1   0.2115064 0.289971 0.32896 0.37502 0.46542
...         ...       ...      ...     ...     ...
theta.6.1   0.7676282 0.859599 0.90175 0.93405 0.97556
   <+ 99 blocks of 6 rows>
p.1         0.1950401 0.225938 0.24436 0.26611 0.31012
...         ...       ...      ...     ...     ...
p.6         0.0905194 0.117203 0.13206 0.14795 0.17993
```

The user can also visualize the output with a trace of the sampled values and a density estimate for each variable in the chain using the plot() method of the **coda** package. For illustration, the following example plots the trace and histogram for θ_{kj} and p_k for cluster $k = 2$ and feature $j = 1$. The produced plot is shown in Figure 3.

```
mat <- matrix(c(1:4), byrow = TRUE, ncol = 2)
layout(mat, widths = rep(c(2, 1), 2), heights = rep(1, 4))
mcmcSubset <- cm1$parameters.ecr.mcmc[ , c("theta.2.1", "p.2")]
plot(mcmcSubset, auto.layout = FALSE, ask = FALSE, col = "gray40")
```

The reader is also referred to the **coda** package which provides various other functions for calculating and plotting MCMC diagnostics.

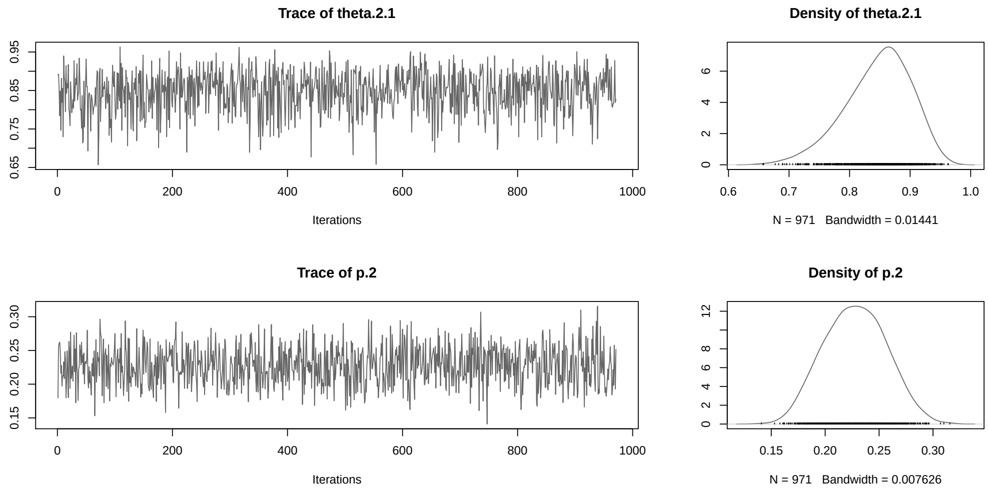


Figure 3: MCMC trace and density estimate for the reordered values of θ_{kj} and p_k for cluster $k = 2$ and feature $j = 1$, conditionally on the selected number of clusters ($K = 6$).

Further inspection of the output

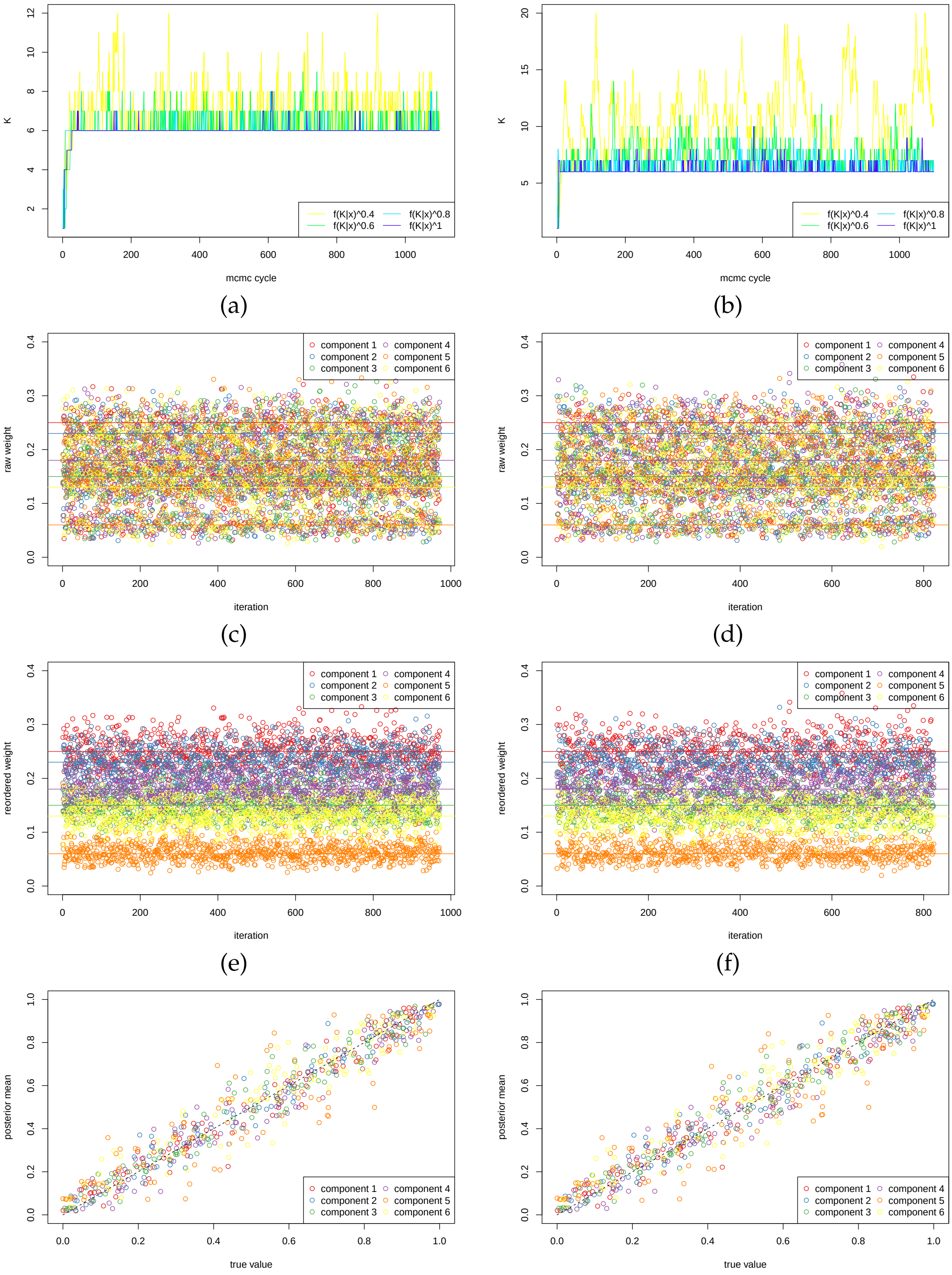
Figures 4.(a) and 4.(b) illustrate the sampled values of K per chain according to the Poisson and uniform prior distribution, respectively. This information is returned to the user as an m×nChains array named K.allChains. The actual posterior distribution corresponds to the blue line. Note that as the temperature increases the posterior distribution of K has larger variability. In both cases, the most probable state corresponds to $K = 6$ clusters, that is, the true value.

Next we inspect the MCMC output conditionally on the event that the number of clusters equals 6 and compare to the true parameter values. At first, we can inspect the raw MCMC output, which is not identifiable due to the label switching problem. Thus, this information is not directly returned to the user, however it is saved to the file 'rawMCMC.mapK.6.txt' in the output directory specified by the output argument. For illustration we plot the raw values of mixture weights. As shown in Figures 4.(c) and 4.(d), the sample is mixing very well to the symmetric posterior areas, since in every iteration labels are changing. The corresponding reordered values (according to the ECR algorithm) are returned to the user as an "mcmc" object named parameters.ecr.mcmc, shown in Figures 4.(e) and 4.(f). Note that the high posterior density areas are quite close to the true values of relative frequencies of generated observations per cluster (indicated by horizontal lines). Finally, Figures 4.(g) and 4.(h) display the posterior mean estimates (arising from the reordered MCMC sample) versus the true values of θ_{kj}, $k = 1, \ldots, 6; j = 1, \ldots, 100$.

Model selection study

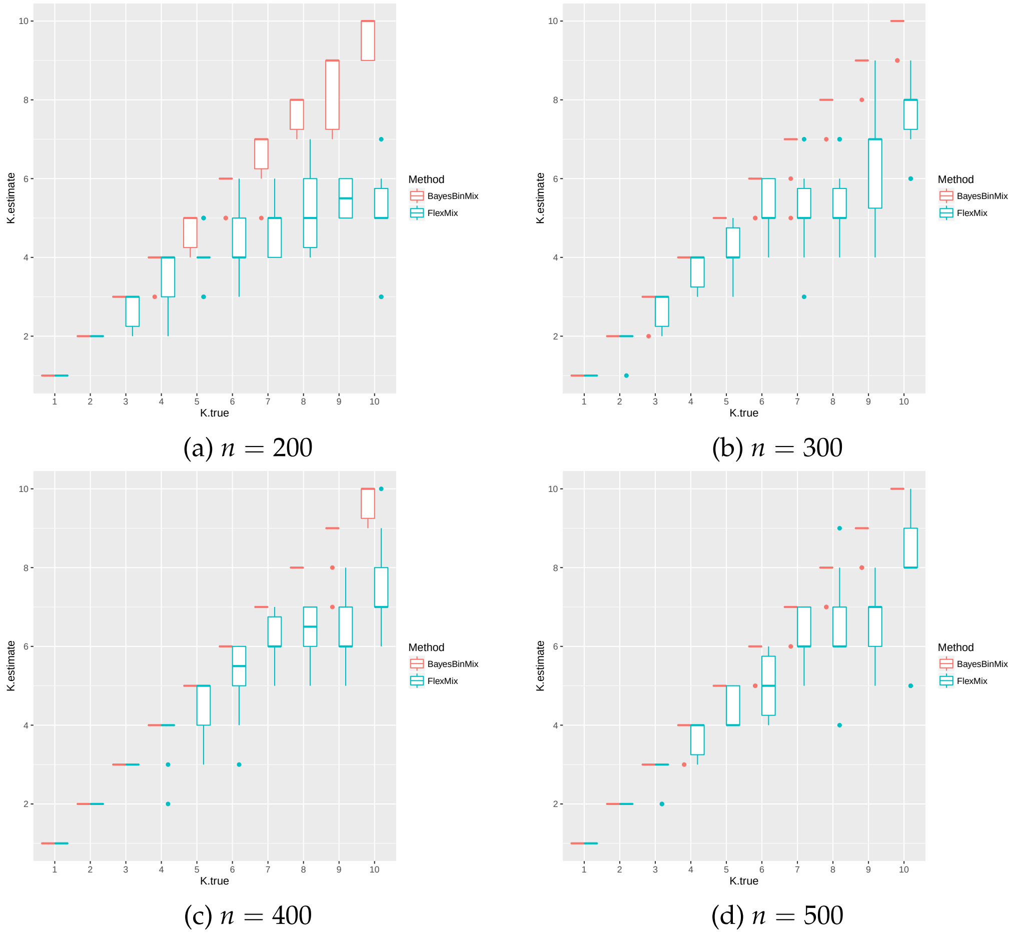
Next we are dealing with model selection issues, that is, selecting the appropriate number of clusters. For this reason we compare **BayesBinMix** with the EM-algorithm implementation provided in **FlexMix**. Under a frequentist framework, the selection of the number of mixture components is feasible using penalized likelihood criteria, such as the BIC (Schwarz, 1978) or ICL (Biernacki et al., 2000), after fitting a mixture model for each possible value of K. We used the ICL criterion since it has been shown to be more robust than BIC, see e.g. Papastamoulis et al. (2016). We considered that the true number of clusters ranges in the set $\{1, 2, \ldots, 10\}$ and for each case we simulated 10 datasets using the same data generation procedure as previously but without introducing any missing values due to the fact that **FlexMix** does not handle missing data. The number of observations varies in the set $n \in \{200, 300, 400, 500\}$. For each simulated data the general call is the following.

```
> library("BayesBinMix")
> library("flexmix")
> nChains <- 8
> heats <- seq(1,0.4,length = nChains)
> cm <- coupledMetropolis(Kmax = 20, nChains = nChains, heats =  heats, binaryData = x,
                          outPrefix = "sampler", ClusterPrior = "poisson", m = 330, burn = 30)
# now run flexmix for binary data clustering
```



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4: Results for simulated data with $K_{\text{true}} = 6$ using the Poisson (left) or uniform (right) prior distribution on the number of clusters (K). (a) and (b): generated values of K per heated chain. (c) and (d): raw output of $p_1, \ldots, p_K$ conditionally on $K = 6$. (e) and (f): reordered sample according to ECR algorithm. Horizontal lines indicate true values of relative number of observations per cluster. (g) and (h): posterior mean estimates of Bernoulli parameters per cluster versus true values.



(a) $n = 200$

(b) $n = 300$

(c) $n = 400$

(d) $n = 500$

Figure 5: Model selection comparison between **BayesBinMix** and **FlexMix**. The x axis corresponds to the true number of clusters and the y axis to the estimated value. Each boxplot corresponds to 10 simulated datasets from a mixture of Bernoulli distributions.

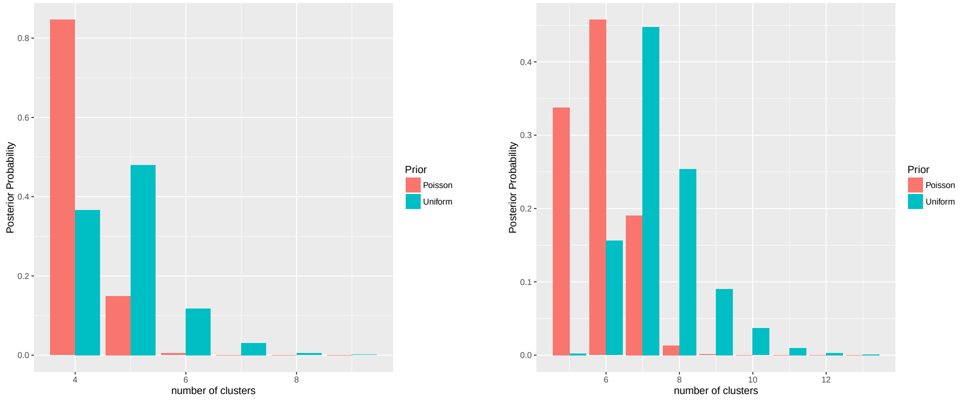
```
> ex <- initFlexmix(x ~ 1, k = 1:20, model = FLXMCmvbinary(),
                    control = list(minprior = 0), nrep = 10)
```

Note that for both algorithms the number of clusters varies in the set $\{1,\ldots,20\}$. Eight heated chains are considered for the MCMC scheme, while each run of the EM algorithm is initialised using `nrep = 10` different starting points in **FlexMix**. Here we used a total of only `m = 330` MCMC cycles in order to show that reliable estimates can be obtained using small number of iterations. Figure 5 displays the most probable number of mixture components estimated by **BayesBinMix** and the selected number of clusters using **FlexMix**, for each possible value of the true number of clusters used to simulate the data. Observe that when the number of clusters is less than 5 both methods are able to estimate the true number of mixture components. However, **FlexMix** tends to underestimate the number of clusters when $K \geqslant 5$, while **BayesBinMix** is able to recover the true value in most cases.

Real data

We consider the zoo database available at the UC Irvine Machine Learning Repository (Lichman, 2013). The database contains 101 animals, each of which has 15 boolean attributes and 1 discrete attribute (`legs`). The partition of animals into a total of 7 classes (mammal, bird, reptile, fish, amphibian, insect and invertebrate) can be considered as the ground-truth clustering of the data, provided in the vector `z.ground_truth`. Following Li (2005), the discrete variable `legs` is transformed into six binary features, which correspond to 0, 2, 4, 5, 6 and 8 legs, respectively. Also we eliminate one of the two entries corresponding to `frog`, as suggested by Li (2005). In total we consider an 100×21 binary array `x` as the input data.

Recall that the Bernoulli mixture in Equation (1) assumes that each cluster consists of a product of independent Bernoulli distributions. Here this assumption is not valid due to the fact that the six new binary variables arising from `legs` are not independent: they should sum to 1. Nevertheless, it is



(a) $\alpha = \beta = 1$ (Uniform)

(b) $\alpha = \beta = 0.5$ (Jeffreys)

Figure 6: Zoo dataset: Estimated posterior distribution of the number of clusters when using different parameters (α, β) on the beta prior of θ_{kj}. Both choices of the prior on the number of clusters are considered: truncated Poisson (red) and uniform (green).

interesting to see how our method performs in cases that the data is not generated by the assumed model.

We test our method considering both prior assumptions on the number of clusters, as well as different hyper-parameters on the prior distribution of θ_{kj} in Equation (8): we consider $\alpha = \beta = 1$ (default values) as well as $\alpha = \beta = 0.5$. Note that the second choice corresponds to the Jeffreys prior (Jeffreys, 1946) for a Bernoulli trial. Figure 6 displays the estimated posterior distribution of the number of clusters K when $K \in \{1, \ldots, 20\}$. This is done with the following commands.

```
# read data
> xOriginal <- read.table("zoo.data", sep = ",")
> x <- xOriginal[ , -c(1, 14, 18)]
> x <- x[-27, ] # delete 2nd frog
# now transform v14 into six binary variables
> v14 <- xOriginal[-27, 14]
> newV14 <- array(data = 0, dim = c(100, 6))
> for(i in 1:100){
+        if( v14[i] == 0 ){ newV14[i,1] = 1 }
+        if( v14[i] == 2 ){ newV14[i,2] = 1 }
+        if( v14[i] == 4 ){ newV14[i,3] = 1 }
+        if( v14[i] == 5 ){ newV14[i,4] = 1 }
+        if( v14[i] == 6 ){ newV14[i,5] = 1 }
+        if( v14[i] == 8 ){ newV14[i,6] = 1 }
+ }
> x <- as.matrix(cbind(x, newV14))

# apply BayesBinMix using 8 heated chains
> library("BayesBinMix")
> nChains <- 8
> heats <- seq(1, 0.6, length = nChains)

# K ~ P{1,...,20}, theta_{kj} ~ Beta(1, 1)
> c1 <- coupledMetropolis(Kmax = 20, nChains = nChains, heats =
                          heats, binaryData = x,
+                          outPrefix = "poisson-uniform", ClusterPrior = "poisson",
+                          m = 4400, burn = 400, z.true = z.ground_truth)

# K ~ U{1,...,20}, theta_{kj} ~ Beta(1, 1)
> c2 <- coupledMetropolis(Kmax = 20, nChains = nChains, heats =  heats, binaryData = x,
+                          outPrefix = "uniform-uniform", ClusterPrior = "uniform",
+                          m = 4400, burn = 400, z.true = z.ground_truth)

# K ~ P{1,...,20}, theta_{kj} ~ Beta(0.5, 0.5)
> c3 <- coupledMetropolis(Kmax = 20, nChains = nChains, heats =  heats, binaryData = x,
```

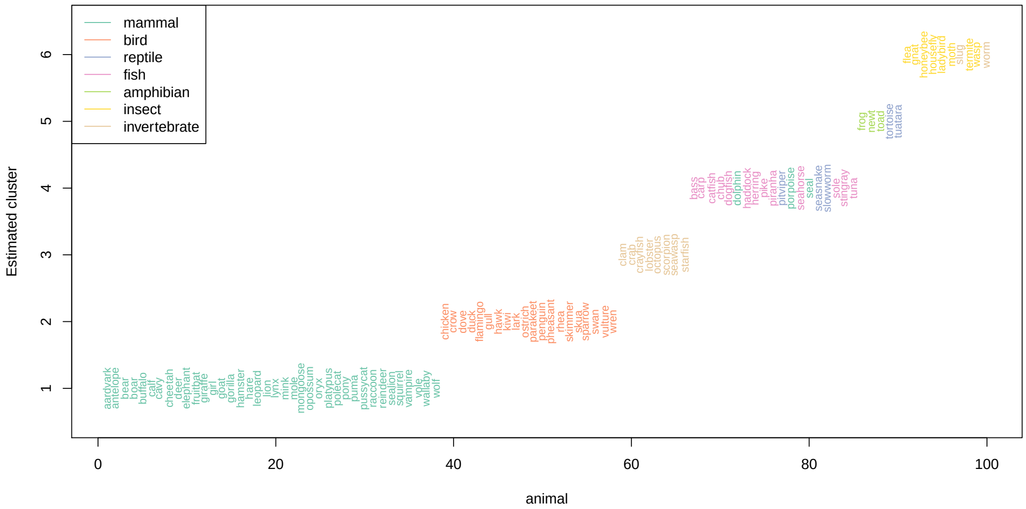


Figure 7: Zoo dataset clustering conditionally on the most probable number of clusters $K = 6$ when using the truncated Poisson prior on K and the Jeffreys prior on θ_{kj}. The ground-truth classification of the animals to seven classes is illustrated using different colour for each class.

```
+                           outPrefix = "poisson-jeffreys", ClusterPrior = "poisson",
+                           m = 4400, burn = 400, z.true = z.ground_truth)

# K ~ U{1,...,20}, theta_{kj} ~ Beta(0.5, 0.5)
> c4 <- coupledMetropolis(Kmax = 20, nChains = nChains, heats =  heats, binaryData = x,
+                           outPrefix = "uniform-jeffreys", ClusterPrior = "uniform",
+                           m = 4400, burn = 400, z.true = z.ground_truth)
```

Next, we compare the estimated clusters (for the most probable value of K) with the classification of the data into 7 classes (given in the vector z.ground_truth). For this reason we provide the rand index (adjusted or not) based on the confusion matrix between the estimated and ground-truth clusters, using the package **flexclust** (Leisch, 2006).

```
> library("flexclust")
> z <- array(data = NA, dim = c(100, 4))
> z[ , 1] <- c1$clusterMembershipPerMethod$ECR
> z[ , 2] <- c2$clusterMembershipPerMethod$ECR
> z[ , 3] <- c3$clusterMembershipPerMethod$ECR
> z[ , 4] <- c4$clusterMembershipPerMethod$ECR
> rand.index <- array(data = NA, dim = c(4, 3))
> rownames(rand.index) <- c("poisson-uniform", "uniform-uniform",
                            "poisson-jeffreys", "uniform-jeffreys")
> colnames(rand.index) <- c("K_map", "rand_index", "adjusted_rand_index")
> findMode <- function(x){ as.numeric( names(sort(-table(x$K.mcmc)))[1] ) }
> rand.index[ , 1] <- c( findMode(c1), findMode(c2), findMode(c3), findMode(c4) )
> for(i in 1:4){
+         rand.index[i, 2] <- randIndex(table(z[ , i], z.ground_truth), correct = FALSE)
+         rand.index[i, 3] <- randIndex(table(z[ , i], z.ground_truth))
+ }
> rand.index
                 K_map rand_index adjusted_rand_index
poisson-uniform      4  0.9230303           0.7959666
uniform-uniform      5  0.9408081           0.8389208
poisson-jeffreys     6  0.9505051           0.8621216
uniform-jeffreys     7  0.9490909           0.8525556
```

Note that both rand indices (raw and adjusted) are larger for z[,3], that is, the six-component mixture model that corresponds to the Poisson prior on K and the Jeffreys prior on θ_{kj}. A detailed view on the estimated clusters for this particular model is shown in Figure 7. We conclude that the estimated groups are characterized by animals belonging to the same taxonomy with very small deviations from the true clusters. Interestingly, in the case that an animal is wrongly assigned to a cluster, notice that

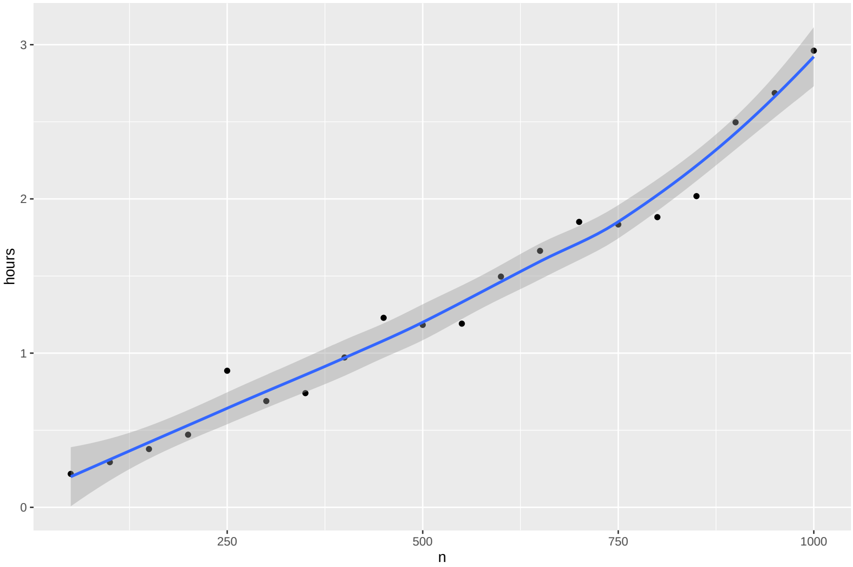


Figure 8: Wall clock time of the `coupledMetropolis()` function when running `nChains = 4` heated chains on the same number of parallel threads for `m = 1100` cycles, with each cycle consisting of 10 MCMC iterations. At most `Kmax = 20` clusters are allowed. For all sample sizes (n), the dimension of the multivariate data equals to $d = 100$.

the estimated grouping might still make sense: e.g. the sea mammals dolphin, porpoise and seal are assigned to the fourth cluster which is mainly occupied by the group “fish”.

Summary and remarks

The **BayesBinMix** package for fitting mixtures of Bernoulli distributions with an unknown number of components has been presented. The pipeline consists of a fully Bayesian treatment for the clustering of multivariate binary data: it allows the joint estimation of the number of clusters and model parameters, deals with identifiability issues as well as produces a rapidly mixing chain. Using a simulation study we concluded that the method outperforms the EM algorithm in terms of estimating the number of clusters and at the same time produces accurate estimates of the underlying model parameters. In the real dataset we explored the flexibility provided by using different prior assumptions and concluded that the estimated clusters are strongly relevant to the natural grouping of the data.

For the prior distribution on the number of clusters our experience suggests that the truncated Poisson distribution performs better than the uniform (see also Nobile and Fearnside (2007)). Regarding the prior distribution on the Bernoulli parameters we recommend to try both the uniform distribution (default choice) as well as the Jeffreys prior, especially when the sample size is small. An important parameter is the number of heated chains which run in parallel, as well as the temperature of each chain. We suggest to run at least `nChains = 4` heated chains. The `heat` parameter for each presented example achieved an acceptance ratio of proposed swaps between pairs of chains between 10% and 70%. The default choice for the temperature vector is `heats = seq(1,0.3,length = nChains)`, however we advise to try different values in case that the swap acceptance ratio is too small (e.g. $< 2\%$) or too large (e.g. $> 90\%$). Finally, we recommend running the algorithm using at least `m = 1100` and `burn = 100` for total number of MCMC cycles and burn-in period, respectively. For these particular values of `nChains` and `m`, Figure 8 displays the wall clock time demanded by the `coupledMetropolis()` function.

Acknowledgements

Research was funded by MRC award MR/M02010X/1. The authors would like to thank Rebecca Howard and Dr. Lijing Lin (University of Manchester) for using the software and reporting bugs to earlier versions. We also thank an anonymous reviewer and Roger Bivand, Editor of the R Journal, for their valuable comments and suggestions that considerably improved the package and the presentation of our findings.

Bibliography

L. Abel, J.-L. Golmard, and A. Mallet. An autologistic model for the genetic analysis of familial binary data. *American journal of human genetics*, 53(4):894, 1993. [p403]

Y. Al-Ohali, M. Cheriet, and C. Suen. Databases for recognition of handwritten Arabic cheques. *Pattern Recognition*, 36(1):111–121, 2003. URL `https://doi.org/10.1016/s0031-3203(02)00064-x`. [p403]

E. S. Allman, C. Matias, and J. A. Rhodes. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, pages 3099–3132, 2009. URL `https://doi.org/10.1214/09-aos689`. [p407]

G. Altekar, S. Dwarkadas, J. P. Huelsenbeck, and F. Ronquist. Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 20(3):407–415, 2004. URL `https://doi.org/10.1093/bioinformatics/btg427`. [p407]

C. Biernacki, G. Celeux, and G. Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):719–725, 2000. URL `https://doi.org/10.1109/34.865189`. [p403, 412]

C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006. [p404]

D. Böhning. *Computer-Assisted Analysis of Mixtures and Applications: Meta-Analysis, Disease Mapping and Others*, volume 81. CRC press, 2000. [p403]

M. Á. Carreira-Perpiñán and S. Renals. Practical identifiability of finite mixtures of multivariate Bernoulli distributions. *Neural Computation*, 12(1):141–152, 2000. URL `https://doi.org/10.1162/089976600300015925`. [p407]

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38, 1977. URL `http://www.jstor.org/stable/2984875`. [p403, 404]

S. Frühwirth-Schnatter. *Finite Mixture and Markov Switching Models*. Springer-Verlag, New York, 2006. URL `https://doi.org/10.1007/978-0-387-35768-3`. [p403]

A. E. Gelfand and A. F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of American Statistical Association*, 85:398–409, 1990. URL `https://doi.org/10.2307/2289776`. [p404]

C. J. Geyer. Markov chain Monte Carlo maximum likelihood. In *Proceedings of the 23rd Symposium on the Interface*, pages 156–163. Interface Foundation, Fairfax Station, Va, 1991. [p407]

C. J. Geyer and E. A. Thompson. Annealing Markov chain Monte Carlo with applications to ancestral inference. *Journal of the American Statistical Association*, 90(431):909–920, 1995. URL `https://doi.org/10.1080/01621459.1995.10476590`. [p407]

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman and Hall, London, 1996. [p407]

P. J. Green. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, 1995. [p405]

B. Grün and F. Leisch. Fitting finite mixtures of generalized linear regressions in R. *Computational Statistics & Data Analysis*, 51(11):5247–5252, 2007. URL `https://doi.org/10.1016/j.csda.2006.08.014`. [p409]

B. Grün and F. Leisch. FlexMix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, 28(4):1–35, 2008. URL `https://doi.org/10.18637/jss.v028.i04`. [p409]

R. Ilin. Unsupervised learning of categorical data with competing models. *IEEE transactions on neural networks and learning systems*, 23(11):1726–1737, 2012. URL `https://doi.org/10.1109/tnnls.2012.2213266`. [p403]

H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, 186:453–461, 1946. URL `https://doi.org/10.1098/rspa.1946.0056`. [p415]

A. Juan and E. Vidal. On the use of Bernoulli mixture models for text classification. *Pattern Recognition*, 35(12):2705–2710, 2002. URL `https://doi.org/10.1016/s0031-3203(01)00242-4`. [p403]

F. Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8):1–18, 2004. URL https://doi.org/10.18637/jss.v011.i08. [p409]

F. Leisch. A toolbox for k-centroids cluster analysis. *Computational Statistics & Data Analysis*, 51(2): 526–544, 2006. URL https://doi.org/10.1016/j.csda.2005.10.006. [p416]

T. Li. A general model for clustering binary data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 188–197. ACM, 2005. URL https://doi.org/10.1145/1081870.1081894. [p403, 414]

M. Lichman. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml. [p414]

B. G. Lindsay. Mixture models: Theory, geometry and applications. In *NSF-CBMS Regional Conference Series in Probability and Statistics*, pages i–163. JSTOR, 1995. [p403]

J. S. Liu. The collapsed Gibbs sampler in Bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):958–966, 1994. URL https://doi.org/10.2307/2290921. [p405]

J. M. Marin, K. Mengersen, and C. P. Robert. Bayesian modelling and inference on mixtures of distributions. *Handbook of Statistics*, 25(1):577–590, 2005. URL https://doi.org/10.1016/s0169-7161(05)25016-2. [p403, 408]

J. G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, New York, 2000. URL https://doi.org/10.1002/0471721182. [p403]

A. Nobile and A. T. Fearnside. Bayesian finite mixtures with an unknown number of components: The allocation sampler. *Statistics and Computing*, 17(2):147–162, 2007. URL https://doi.org/10.1007/s11222-006-9014-7. [p403, 405, 406, 407, 417]

P. Papastamoulis. Handling the label switching problem in latent class models via the ECR algorithm. *Communications in Statistics-Simulation and Computation*, 43(4):913–927, 2014. URL https://doi.org/10.1080/03610918.2012.718840. [p407]

P. Papastamoulis. Label.switching: An R package for dealing with the label switching problem in MCMC outputs. *Journal of Statistical Software*, 69(1):1–24, 2016. URL https://doi.org/10.18637/jss.v069.c01. [p407]

P. Papastamoulis and G. Iliopoulos. Reversible jump MCMC in mixtures of normal distributions with the same component means. *Computational Statistics & Data Analysis*, 53(4):900–911, 2009. URL https://doi.org/10.1016/j.csda.2008.10.022. [p405]

P. Papastamoulis and G. Iliopoulos. An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. *Journal of Computational and Graphical Statistics*, 19:313–331, 2010. URL https://doi.org/10.1198/jcgs.2010.09008. [p407]

P. Papastamoulis and G. Iliopoulos. On the convergence rate of random permutation sampler and ECR algorithm in missing data models. *Methodology and Computing in Applied Probability*, 15(2): 293–304, 2013. ISSN 1573-7713. URL https://doi.org/10.1007/s11009-011-9238-7. [p407]

P. Papastamoulis, M.-L. Martin-Magniette, and C. Maugis-Rabusseau. On the estimation of mixtures of Poisson regression models with large number of components. *Computational Statistics & Data Analysis*, 93:97–106, 2016. URL https://doi.org/10.1016/j.csda.2014.07.005. [p412]

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006. URL https://journal.r-project.org/archive/. [p408]

R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM review*, 26(2):195–239, 1984. URL https://doi.org/10.1137/1026034. [p403, 407]

Revolution Analytics and S. Weston. *Foreach: Foreach Looping Construct for R*, 2014. URL http://CRAN.R-project.org/package=foreach. R package version 1.4.2. [p408]

Revolution Analytics and S. Weston. *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*, 2015. URL http://CRAN.R-project.org/package=doParallel. R package version 1.0.10. [p408]

S. Richardson and P. J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society B*, 59(4):731–758, 1997. URL https://doi.org/10.1111/1467-9868.00095. [p403, 405]

C. E. Rodríguez and S. G. Walker. Label switching in Bayesian mixture models: Deterministic relabeling strategies. *Journal of Computational and Graphical Statistics*, 23(1):25–45, 2014. URL https://doi.org/10.1080/10618600.2012.735624. [p407]

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. URL https://doi.org/10.1214/aos/1176344136. [p403, 412]

M. Stephens. Bayesian analysis of mixture models with an unknown number of components – an alternative to reversible jump methods. *The Annals of Statistics*, 28(1):40–74, 2000a. [p403, 405]

M. Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society B*, 62(4):795–809, 2000b. URL https://doi.org/10.1111/1467-9868.00265. [p407]

Z. Sun, O. Rosen, and A. R. Sampson. Multivariate Bernoulli mixture models with application to postmortem tissue studies in schizophrenia. *Biometrics*, 63(3):901–909, 2007. ISSN 1541-0420. URL https://doi.org/10.1111/j.1541-0420.2007.00762.x. [p403]

M. Tanner and W. Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540, 1987. URL https://doi.org/10.2307/2289457. [p404]

A. White, J. Wyse, and T. B. Murphy. Bayesian variable selection for latent class analysis using a collapsed Gibbs sampler. *Statistics and Computing*, 26(1-2):511–527, 2016. URL https://doi.org/10.1007/s11222-014-9542-5. [p403, 405]

Panagiotis Papastamoulis
University of Manchester
Oxford road, Manchester, M13 9PL
United Kingdom
panagiotis.papastamoulis@manchester.ac.uk

Magnus Rattray
University of Manchester
Oxford road, Manchester, M13 9PL
United Kingdom
magnus.rattray@manchester.ac.uk