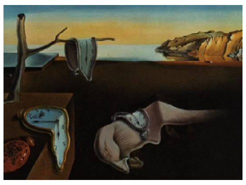

Journal of Statistical Software

November 2021, Volume 100, Issue 16. doi: 10.18637/jss.v100.i16

A Bayesian Approach for Model-Based Clustering of Several Binary Dissimilarity Matrices: The dmbc Package in R

Sergio Venturini
Università Cattolica del Sacro Cuore

Raffaella Piccarreta
Università Commerciale L. Bocconi

Abstract

We introduce the new package **dmbc** that implements a Bayesian algorithm for clustering a set of binary dissimilarity matrices within a model-based framework. Specifically, we consider the case when S matrices are available, each describing the dissimilarities among the same n objects, possibly expressed by S subjects (judges), or measured under different experimental conditions, or with reference to different characteristics of the objects themselves. In particular, we focus on binary dissimilarities, taking values 0 or 1 depending on whether or not two objects are deemed as dissimilar. We are interested in analyzing such data using multidimensional scaling (MDS). Differently from standard MDS algorithms, our goal is to cluster the dissimilarity matrices and, simultaneously, to extract an MDS configuration specific for each cluster. To this end, we develop a fully Bayesian three-way MDS approach, where the elements of each dissimilarity matrix are modeled as a mixture of Bernoulli random vectors. The parameter estimates and the MDS configurations are derived using a hybrid Metropolis-Gibbs Markov Chain Monte Carlo algorithm. We also propose a BIC-like criterion for jointly selecting the optimal number of clusters and latent space dimensions. We illustrate our approach referring both to synthetic data and to a publicly available data set taken from the literature. For the sake of efficiency, the core computations in the package are implemented in C/C++. The package also allows the simulation of multiple chains through the support of the **parallel** package.

Keywords: Bayesian data analysis, dissimilarity matrices, information criteria, multidimensional scaling, MCMC, MDS, mixture models, model-based clustering, three-way MDS.

1. Introduction

Data consisting of proximity measures, that is measurements of the pairwise similarity or dissimilarity between n objects, abound in many fields, ranging from psychology, sociology and market research, to ecology, demography, economics, as well as genomics and linguistics. A dissimilarity matrix, $\mathbf{D}$ – whose (i, h)-th entry, d_{ih}, represents the dissimilarity between objects i and h – is typically analyzed using multidimensional scaling (MDS; for a comprehensive overview see Borg and Groenen 2005). MDS is a factorial technique that aims at finding a set of latent factors (the so-called MDS configuration) in a low-dimensional Euclidean space, such that the distances between objects in this space resemble as much as possible the original d_{ih}'s. In its basic formulation, MDS involves a *single* dissimilarity matrix.

Different MDS variants have been introduced in the literature. The classical or metric MDS (CMDS; Torgerson 1952, 1958) assumes that the d_{ih}'s are Euclidean distances plus a random error. Nonmetric MDS (Shepard 1962; Kruskal 1964) handles ordinal data, assuming that only the ordering of the dissimilarities is relevant in the determination of the MDS configuration. The ALSCAL algorithm (Takane, Young, and De Leeuw 1977) combines metric and nonmetric solutions and derives the MDS configuration by fitting a function (whose features depend on the type of data) of the d_{ih}'s with errors added (see Krantz, Luce, Suppes, and Tversky 1971). Also, maximum likelihood (metric) MDS approaches have been proposed (Ramsay 1977, 1978, 1982; Takane 1978b,a, 1981; Takane and Carroll 1981; Takane and Sergent 1983; Takane and Shibayama 1986, 1992), which allow for the selection of the number of MDS factors using likelihood ratio tests or Akaike's criterion (Takane 2007). Adopting a Bayesian approach, Oh and Raftery (2001) introduced a model where the d_{ih}'s follow a truncated normal distribution whose expected value is the Euclidean distance based upon the MDS factors. The authors rely on Markov Chain Monte Carlo (MCMC) methods for estimation and also introduce a criterion to select the number of dimensions.

A further major development in the literature on MDS is the so-called *three-way* MDS, where *many* dissimilarity matrices, say $\mathbf{D}_1, \ldots, \mathbf{D}_S$, are taken into account simultaneously (Young 1987; Borg and Groenen 2005; Giguère 2006). This situation typically arises when the pairwise dissimilarities among n objects are judgments expressed by different subjects (usually playing the role of judges or raters), or when they are measured under different experimental conditions, or with reference to different characteristics of the objects themselves. For the sake of exposition, and without loss of generality, throughout the paper we will refer to the first situation, thus assuming that the dissimilarity matrices describe the judgments expressed by S subjects.

When dealing with multiple dissimilarity matrices, two opposite situations may be considered, leading to the so-called replicated MDS (RMDS) and weighted MDS (WMDS) (see for example Young 1987; Borg and Groenen 2005). The RMDS approach relies on the assumption that all the dissimilarities derive from the same latent factors, and that the possible differences across the S dissimilarity matrices are due to random measurement errors. Hence, RMDS aims at identifying a consensus MDS configuration minimizing the sum of the squared prediction errors for all the $\mathbf{D}_j$'s. Adopting an opposite perspective, one could regard each $\mathbf{D}_j$ as the emanation of a subject-specific set of factors. Within this context, the WMDS approach assumes that a common MDS configuration exists, but that each subject assigns a different importance to the common underlying factors. Thus, the individual $\mathbf{D}_j$'s can be recovered by applying subject-specific weights to the MDS factors, and the individual differences in judgments can be assessed by comparing the corresponding weights. A notable example in this context is the well-known INDSCAL model (Carroll and Chang 1970; Borg and Groenen 2005, Chapter 21). This approach postulates the existence of a *group space*, that represents what the subjects have in common, and a set of *individual spaces*, which are uniquely generated by stretching or compressing the group space along its dimensions. Both RMDS and WMDS can pose some problems in applications. Indeed, the two approaches postulate that a unique configuration exists, but such hypothesis cannot be tested. Moreover, with regard to WMDS, comparing the subjects based on the differences in the estimated weights can be challenging. Also, in some applications groups of subjects sharing the same system of weights can be observed, and it could be of interest to identify such groups using a proper clustering approach. This would allow unveiling the differences in the cognitive processes related to judgments similarity. In the same way, when dissimilarities are measured under different experimental conditions (for example, different time periods), clustering such conditions permits to properly characterize the scenarios that significantly impact on the dissimilarities between objects. An interesting contribution in this direction is the K-INDSCAL model developed by Bocci and Vichi (2011), that assumes the existence of K homogeneous groups of subjects each having a different group space, while also allowing for class-conditional individual dimensional weighting. More formally, the K-INDSCAL approach is defined as a finite mixture of INDSCAL models, for which estimation is performed by alternating least-squares. A Bayesian version of the K-INDSCAL model was proposed by Okada and Lee (2016), who extended the model introduced Oh and Raftery (2001) for continuous dissimilarities. Notably, while offering a structured inferential framework, the complexity of their model did not allow the definition of a formal model selection tool.

The most popular commercial statistical software all provide a module that implements at least some of the MDS variants described above. For example, SAS (SAS Institute Inc. 2013) includes the `MDS` procedure, which allows to fit two- and three-way, metric and nonmetric MDS models. Similarly, the `PROXSCAL` algorithm in SPSS (IBM Corporation 2021) fits all the traditional MDS models[1]. Finally, Stata (StataCorp 2021) provides the `mds` and `mdsmat` commands to perform classic metric MDS as well as modern metric and nonmetric two-way MDS. Turning to open source implementations, only R offers a relatively large number of packages to apply different types of MDS (for a comprehensive list, refer to the psychometric methods task view, Mair 2021). Among them, it is worth to mention the **smacof** package (De Leeuw and Mair 2009), the most general in our opinion, which implements different MDS approaches based on the minimization of a stress function using an iterative majorization algorithm[2]. However, to the best of our knowledge, there is no package currently available that includes functions for fitting whatever form of Bayesian MDS[3].

Here, we introduce the **dmbc** package, which implements an extension to three-way binary data of the fully Bayesian MDS approach by Oh and Raftery (2007). In our model, the subjects' dissimilarity matrices are partitioned into G clusters and a different p-dimensional MDS configuration is derived for each cluster, with G and p a priori unknown. Thus, the matrices clustered together share a common latent configuration space. Even if our proposal can in

[1]SPSS also includes `PREFSCAL`, an algorithm that implements *unfolding*, which is a further MDS variant used to model preferential choice data (for an overview see Borg and Groenen 2005, Chapter 14).

[2]The package name derives from the name of the algorithm itself, SMACOF, which is the acronym of "scaling by majorizing a complicated function". The algorithm has been introduced by De Leeuw (1977), but for an accessible overview we suggest to see Borg and Groenen (2005), Chapter 8.

[3]The only traces we found on the web about software for Bayesian MDS are: 1) an R function called `BMDS` wrapping a **JAGS** (Plummer 2003) model available at `https://github.com/davidaarmstrong/asmcjr` (see Armstrong II, Bakker, Carroll, Hare, Poole, and Rosenthal 2014) , 2) the original Fortran routines that implement the model in Oh and Raftery (2001), 3) a set of R functions introduced by Okada and Shigemasu (2009), but no longer available on the web, and 4) the Stan (Carpenter *et al.* 2017) code for the Bayesian K-INDSCAL model in Okada and Lee (2016).

principle be applied to any type of dissimilarities, here we focus on binary dissimilarities, taking only values 0 or 1 depending on whether two objects are deemed as dissimilar or not. We concentrate on binary dissimilarities because such type of data are often analyzed using simplistic approaches. Indeed, MDS is frequently applied to a consensus dissimilarity matrix obtained by summing or averaging the individual ones. In other cases, groups of dissimilarity matrices are formed a priori (based on experimental conditions or on the subjects' characteristics, such as gender or age), and three-way MDS (typically, either RMDS or WMDS) is performed on the consensus matrices formed for each group. Clearly, in the latter case, the groups are identified a priori, without evaluating whether they are actually different and/or internally homogeneous. While this can be reasonable when the analysis is confirmatory, an exploratory approach would instead surely benefit from a data-driven procedure aiming at identifying the possible clusters and their specific MDS configurations. As already mentioned, this allows to ignore the possibly negligible differences across subjects, to detect groups of subjects sharing similar opinions, and to better shape and identify their most distinctive traits. Additionally, this simplifies subsequent analyses, which can be based on a reduced number of groups rather than the original set of subjects. The model implemented in the **dmbc** package provides a possible solution to achieve these goals. Embedding the model in a Bayesian framework is particularly convenient because it permits a full inferential assessment of the obtained results. Furthermore, we build on the work of Oh and Raftery (2007), and introduce a BIC-like criterion for simultaneously selecting the optimal number of latent dimensions, p, and number of clusters, G.

A distinctive feature of our approach compared to classic RMDS or WMDS is that in the latter a *single* MDS configuration is extracted, while we obtain a different configuration for each cluster of subjects. Under this perspective, our proposal can be regarded as an extension of RMDS, because we assume that there are groups of subjects – not known a priori – whose dissimilarities derive from the same cluster-specific MDS configuration. Instead, our method differs from WMDS, because we assume that all the subjects placed in the same cluster share the same configuration. Note that the more complex Bayesian model introduced by Okada and Lee (2016) – assuming that subjects in the same cluster assign different weights to the common cluster-specific MDS factors – is a full integration of RMDS and of WMDS. The introduction of individual weights allows capturing differences across subjects, even when placed in the same cluster. Our model instead leads to a compromise solution cluster-wise, which can nonetheless be more representative and easier to interpret. In addition, our simpler model has the advantage of allowing the definition of a criterion for model selection.

Before proceeding, it is worth mentioning that our procedure shares some features with the approach introduced by Hoff, Raftery, and Handcock (2002) and extended by Handcock, Raftery, and Tantrum (2007) to model social networks based on the assumption of the existence of a latent (unique) "social space". Specifically, Hoff *et al.* (2002) refer to a class of random graph models where the nodes represent n social actors, the edge between two nodes signals a specified relation between them, and the probability of observing a relation depends on the distance between the nodes in a latent space. Handcock *et al.* (2007) extend the model to account for the possible clustering of n *actors* based on their position in the unique latent space.[4] Instead, our goal is to cluster the S subjects according to their judgments about the similarity of the n *objects*, thus partitioning configurations rather than positions.

[4]These models can be fit using the **latentnet** package (Krivitsky, Handcock, Raftery, and Hoff 2009; Krivitsky and Handcock 2008).

More specifically, we model the joint distribution of each subject's dissimilarities as a finite mixture of Bernoulli random vectors, with parameters depending on the cluster-specific MDS factors and estimate all the unknown quantities (i.e., the parameters and the latent positions) using a hybrid Metropolis-Gibbs algorithm. Our package also allows to run simultaneously multiple MCMC chains by exploiting the capabilities of the **parallel** package (R Core Team 2021). To make the computations more efficient, most of the routines included in the package are written in C/C++ through the support of the **Rcpp** package (Eddelbuettel and François 2011).

The **dmbc** package is available from the Comprehensive R Archive Network (CRAN) at `http://CRAN.R-project.org/package=dmbc` and it can be installed directly using, for example, `install.packages("dmbc")`. The development version of the package can also be retrieved from `https://github.com/sergioventurini/dmbc`.

The paper first briefly reviews the model and then moves to fully present the package features. More specifically, Section 2 introduces our model-based approach for clustering a set of binary dissimilarity matrices, together with the details of the MCMC algorithm for its estimation. Section 3 illustrates the proposed criterion to simultaneously select the number of latent dimensions and the number of clusters. Section 4 provides a description of the features available in the **dmbc** package and shows how to apply them on simulated data. In Section 5 we apply the package's functionalities to a data set that is available in the literature. Finally, Section 6 summarizes and provides some closing remarks.

2. Model specification and estimation

We consider S binary and symmetric dissimilarity matrices, $\mathbf{D}_1, \mathbf{D}_2, \ldots, \mathbf{D}_S$, so that the (i,h)-th element of $\mathbf{D}_j$, $d_{ih,j}$, can only take values 0 or 1, with $d_{ih,j} = d_{hi,j}$ if $i \neq h$, and 0 otherwise. We assume that the S subjects are partitioned into G clusters, with G a priori unknown. To represent the subjects' group memberships we refer to a set of latent variables, x_j, such that $\mathsf{P}(x_j = g) = \lambda_g$, with $\lambda_g \geq 0$ and $\sum_{g=1}^{G} \lambda_g = 1$. Matrices assigned to the same cluster are assumed to share a common p-dimensional latent configuration space for the n objects, with p also unknown. Therefore, for the i-th object, a latent vector exists, $\boldsymbol{z}_i^g := (\boldsymbol{z}_i \mid x_j = g) = (z_{i1}^g, \ldots, z_{ip}^g)$, reflecting the factors scores possibly underlying the dissimilarity judgments expressed by the subjects in the g-th cluster. Thus, the true dissimilarity between objects i and h for the subjects in the g-th cluster, δ_{ih}^g, can be expressed as the Euclidean distance between $\boldsymbol{z}_i^g$ and $\boldsymbol{z}_h^g$, $\delta_{ih}^g = \sqrt{\sum_{k=1}^{p} \left(z_{ik}^g - z_{hk}^g\right)^2}$. Conditionally on the cluster, we assume that the observed dissimilarity is a Bernoulli random variable, $(d_{ih} \mid x_j = g) \sim \mathcal{B}er(\pi_{ih}^g)$, with π_{ih}^g depending on δ_{ih}^g through

$$\operatorname{logit}(\pi_{ih}^g) = \alpha_g + \delta_{ih}^g, \tag{1}$$

or equivalently,

$$\pi_{ih}^g = \frac{\exp(\alpha_g + \delta_{ih}^g)}{1 + \exp(\alpha_g + \delta_{ih}^g)} \equiv \operatorname{expit}(\alpha_g + \delta_{ih}^g). \tag{2}$$

Further, the j-th subject's dissimilarities are assumed to be conditionally independent given the latent positions in the configuration space, so that

$$(\boldsymbol{d}_j \mid x_j = g) \sim f(\boldsymbol{d}_j \mid \boldsymbol{\pi}^g) = \prod_{i<h} \left(\pi_{ih}^g\right)^{d_{ih,j}} \left(1 - \pi_{ih}^g\right)^{1-d_{ih,j}}, \tag{3}$$

where $\boldsymbol{d}_j$ denotes the vector whose elements are the $m = n(n-1)/2$ elements in the lower triangular part of $\mathbf{D}_j$, and $\boldsymbol{\pi}^g$ denotes the corresponding vector of π's. Hence, the unconditional distribution of $\boldsymbol{d}_j$ turns out to be the mixture

$$\boldsymbol{d}_j \sim \sum_{g=1}^{G} \lambda_g f(\boldsymbol{d}_j \mid \boldsymbol{\pi}^g). \tag{4}$$

We aim at finding the G clusters and at estimating the latent p-dimensional configuration for each cluster. Note that p is assumed to be the same for all the clusters. This may appear as a strong assumption, possibly reducing the flexibility of the model. Nonetheless, if the unknown dimensions of the clusters' configurations, $(p_1, \ldots, p_G)$, are actually different and vary between, say, p_{min} and p_{max}, we can expect that a dimension $p = p_{max}$ will be selected, and that for a cluster with $p_g < p$, $(p - p_g)$ factors will turn out to be irrelevant. Therefore, the assumption of a constant number of latent dimensions in the different clusters should not have a large impact on the results.

2.1. Model estimation

The adoption of a fully Bayesian approach requires the specification of prior distributions for all the unknown parameters. For the latent positions we assume a p-variate normal distribution, $\boldsymbol{z}_i^g \sim \mathcal{N}_p(\mathbf{0}, \eta_g \mathbf{I}_p)$. The assumption of spherical covariance matrix allows avoiding a too large number of parameters, but in principle it could be relaxed. For η_g we use an inverse gamma distribution with mode $b_g/(a_g + 1)$, $\eta_g \sim \mathcal{IG}(a_g, b_g)$. For the logit cluster-specific intercept in Equation 1, we assume that $\alpha_g \sim \mathcal{N}(0, \sigma_g^2)$, with $\sigma_g^2 \sim \mathcal{IG}(a_0, b_0)$. Finally, we use a G-dimensional Dirichlet distribution for the vector of mixing probabilities, $\boldsymbol{\lambda} \sim \mathcal{D}_G(\nu_1, \ldots, \nu_G)$. All the parameters are assumed to be a priori independent.

Based on such specifications, the full conditional posterior distributions of the model parameters are:

$$(\boldsymbol{z}_i^g \mid \text{else}) \sim \phi_p(\boldsymbol{z}_i^g \mid \mathbf{0}, \eta_g \mathbf{I}_p)\, \tilde{\ell}(\boldsymbol{Z}^g, \alpha_g, \boldsymbol{x}), \tag{5}$$

$$(\eta_g \mid \text{else}) \sim \mathcal{IG}\left(a_g + \frac{np}{2}, b_g + \frac{1}{2}\sum_{k=1}^{p}\sum_{i=1}^{n}(z_{ik}^g)^2\right), \tag{6}$$

$$(\alpha_g \mid \text{else}) \sim \phi(\alpha_g \mid 0, \sigma_g^2)\, \tilde{\ell}(\boldsymbol{Z}^g, \alpha_g, \boldsymbol{x}), \tag{7}$$

$$(\sigma_g^2 \mid \text{else}) \sim \mathcal{IG}\left(a_0 + 1, b_0 + \frac{\alpha_g^2}{2}\right), \tag{8}$$

$$\mathsf{P}(x_j = g \mid \text{else}) = \frac{\lambda_g f\left(\boldsymbol{d}_j \mid \boldsymbol{\pi}^g\right)}{\sum_{l=1}^{G} \lambda_l f\left(\boldsymbol{d}_j \mid \boldsymbol{\pi}^l\right)}, \tag{9}$$

$$(\boldsymbol{\lambda} \mid \text{else}) \sim \mathcal{D}_G(\nu_1 + S_1, \ldots, \nu_G + S_G), \tag{10}$$

where $\boldsymbol{x} = (x_1, \ldots, x_S)$, $\boldsymbol{Z}^g = (\boldsymbol{z}_1^g, \ldots, \boldsymbol{z}_n^g)$, $\phi_p(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ indicates the p-dimensional multivariate normal density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and $\tilde{\ell}(\boldsymbol{Z}^g, \alpha_g, \boldsymbol{x})$ represents the contribution of the g-th cluster to the overall likelihood:

$$\tilde{\ell}(\boldsymbol{Z}^g, \alpha_g, \boldsymbol{x}) = \prod_{i<h} \operatorname{expit}(\alpha_g + \delta_{ih}^g)^{d_{ih,+}^g}(1 - \operatorname{expit}(\alpha_g + \delta_{ih}^g))^{S_g - d_{ih,+}^g}, \tag{11}$$

with $d^g_{ih,+} = \sum_{j=1}^S d^g_{ih,j}$, and $S_g = \sum_{j=1}^S \mathbb{I}(x_j = g)$. The notation $(\cdot \mid \text{else})$ in Equations 5-10 indicates conditioning on all the other unknowns in the model.

To simulate from the posterior distributions, a MCMC algorithm is used, which iterates over the model parameters, latent positions and cluster memberships by updating them in turn. We use a Gibbs sampler for the parameters whose full conditional posterior distribution is available, while we adopt a Metropolis-Hastings step otherwise.

By cycling over the clusters, that is for $g = 1, \ldots, G$, we use a Metropolis-Hastings step to update the $(\boldsymbol{Z}^g)^{(t)}$'s, by sampling the n objects in a random order. For the i-th sampled object, a normal proposal density is used, and a candidate $(\boldsymbol{z}^g_i)^* \sim \mathcal{N}_p((\boldsymbol{z}^g_i)^{(t)}, \gamma^2_{\boldsymbol{Z}}\mathbf{I}_p)$ is accepted with probability

$$\frac{\tilde{\ell}\Big((\boldsymbol{Z}^g)^*, \alpha_g^{(t)}, \boldsymbol{x}^{(t)}\Big)\, \phi_p\Big((\boldsymbol{z}^g_i)^* \mid \mathbf{0}, \eta_g^{(t)}\boldsymbol{I}_p\Big)}{\tilde{\ell}\Big((\boldsymbol{Z}^g)^{(t)}, \alpha_g^{(t)}, \boldsymbol{x}^{(t)}\Big)\, \phi_p\Big((\boldsymbol{z}^g_i)^{(t)} \mid \mathbf{0}, \eta_g^{(t)}\boldsymbol{I}_p\Big)},$$

where $\tilde{\ell}\left((\boldsymbol{Z}^g)^*, \alpha_g^{(t)}, \boldsymbol{x}^{(t)}\right)$ is defined as in Equation 11. Note that we abused a little the notation since the $\boldsymbol{z}^g_i$ are updated one at a time. Indeed, in the expression above it is $(\boldsymbol{Z}^g)^* = (\boldsymbol{z}^g_1, \ldots, (\boldsymbol{z}^g_i)^*, \ldots, \boldsymbol{z}^g_n)$, with only one object's vector of latent positions, the i-th, updated.

A further Metropolis-Hastings step is used to update $\alpha_g^{(t)}$. A candidate $\alpha^*_g \sim \mathcal{N}(\alpha_g^{(t)}, \gamma^2_\alpha)$, is accepted with probability

$$\frac{\tilde{\ell}\Big((\boldsymbol{Z}^g)^{(t+1)}, \alpha^*_g, \boldsymbol{x}^{(t)}\Big)\, \phi\Big(\alpha^*_g \mid \mathbf{0}, \sigma_g^{2(t)}\Big)}{\tilde{\ell}\Big((\boldsymbol{Z}^g)^{(t+1)}, \alpha_g^{(t)}, \boldsymbol{x}^{(t)}\Big)\, \phi\Big(\alpha_g^{(t)} \mid \mathbf{0}, \sigma_g^{2(t)}\Big)}.$$

The variances of the proposal distributions, $\gamma_{\boldsymbol{Z}}$ and γ_α, are chosen to achieve a good performance of the algorithm in terms of acceptance rate. Preliminary testing suggested that setting $\gamma_{\boldsymbol{Z}} = 1.5$ and $\gamma_\alpha = 0.75$ provides satisfactory results in terms of acceptance rates (i.e., between 20% and 30%) in most cases.

Every MCMC iteration for the cluster-specific parameters is completed by updating η_g, σ^2_g, $\boldsymbol{x}$, and $\boldsymbol{\lambda}$, using their full conditionals, shown in Equation 6, Equation 8, Equation 9 and Equation 10, respectively.

We compute the parameter estimates as the posterior means from the corresponding MCMC chain after deleting the burn-in iterations.

2.2. Initialization

To initialize the MCMC simulation, starting values of all the model's parameters must be provided. As for the initial partition $\boldsymbol{x}^0$, in the package three different approaches are available (see Section 4 for the corresponding syntax specification):

1. Select the initial partition randomly.
2. Partition the S subjects into G groups by applying a Ward hierarchical clustering algorithm to the vectors of observed dissimilarities $\boldsymbol{d}_j$, $j = 1, \ldots, S$ using a specified distance measure (Everitt, Landau, Leese, and Stahl 2011).
3. Provide a user-defined partition, possibly obtained using alternative approaches.

In our experience, the choice of the initial partition does not seem to systematically affect the final results of the analysis. In any case, in all the examples reported in this paper, we set the initial partition randomly, to let the algorithm learn from the data without being influenced by any specific initial choice.

No matter how the initial partition is chosen, the corresponding vector of clusters' proportions, $\boldsymbol{\lambda}^0 = (n_1/S, \ldots, n_G/S)$, is used to initialize $\boldsymbol{\lambda}$. Given the initial partition, the starting p-dimensional MDS configuration for the g-th cluster, $(\boldsymbol{Z}^g)^{(0)}$, is derived through a classical metric MDS from a consensus matrix $\mathbf{D}^g$ obtained by summing the dissimilarity matrices of the subjects assigned to cluster g.

To initialize η_g we use the average sample variance computed over the p coordinates of $(\boldsymbol{Z}^g)^{(0)}$, $\bar{s}_g^{(0)}$. For α_g we rely on the means of the elements of the $\boldsymbol{d}_j$'s in the g-th cluster, $\bar{\boldsymbol{d}}^g$, and to the overall mean of the dissimilarities in the g-th cluster, $\bar{\bar{d}}^g$. For each pair of objects, we define the artificial binary dissimilarity $\tilde{d}_{ih}^g$ taking value 1 or 0 depending on whether or not $\bar{d}_{ih}^g > \bar{\bar{d}}^g$. We fit the logistic regression model (Equation 1) relating the $\tilde{d}_{ih}^g$'s to the distances computed using the initial configuration $(\boldsymbol{Z}^g)^{(0)}$, and take $\alpha_g^{(0)}$ as the intercept estimate of this model and the squared standard error of the estimate itself as a starting value for σ_g^2, the variance of α_g.

The hyper-parameters $(a_0, b_0, a_1, b_1, \ldots, a_G, b_G, \nu_1, \ldots, \nu_G)$ need to be specified by the user. As for η_g, we set $a_g = 3/2$ and $b_g = \frac{1}{2}\bar{s}_g^{(0)}$, so that the prior mean for η_g is fixed at $\bar{s}_g^{(0)}$. We use a noninformative prior for σ_g^2, by choosing $a_0 = b_0 = 10^{-1}$. Finally, for the prior distribution of $\boldsymbol{\lambda}$ we follow the standard approach in the literature on Bayesian finite mixtures (Robert 1996) and set $\nu_g = 1$ for $g = 1, \ldots, G$. Tests on the effect of changing the values of the hyper-parameters showed a little impact on the final numerical results, and almost no effect on the qualitative conclusions for all the data sets we analyzed.

2.3. Identifiability and post-processing

Since the Euclidean distance is invariant under translation, rotation, and reflection, the likelihood will also be invariant. To solve the consequent identifiability issues, we follow a standard approach (see for example Oh and Raftery 2001; Hoff *et al.* 2002), and post-process the sampled latent positions by applying a Procrustes transformation (Gower and Dijksterhuis 2004; Borg and Groenen 2005). Thus, the resulting configuration $\boldsymbol{Z}^g$, $g = 1, \ldots, G$, is rotated toward maximum similarity with a reference configuration, that we set equal to the configuration at the last iteration of the MCMC algorithm.

The likelihood is also invariant under a permutation of the clusters' labels (Stephens 2000). To fix this label switching issue, we choose the relabeling approach proposed by Celeux, Hurn, and Robert (2000)[5].

[5]An alternative to the application of the relabeling algorithm we included in the package consists in skipping the post-processing step by setting `procrustes = FALSE` and `relabel = FALSE`, and then applying another user-defined algorithm to the parameter chains, such as one of those implemented in the **label.switching** package (Papastamoulis 2016).

3. Model choice

In the previous section we described the inferential procedure for given G, the number of clusters, and p, the dimension of the latent spaces. Actually, the proper choice of p and G is a relevant issue in MDS and in cluster analysis respectively. Embedding the problem in a Bayesian framework allows to develop selection criteria. Specifically, we extend to our case the approach proposed in Oh and Raftery (2007) to simultaneously choose p and G, where they take into account the situation of a unique numerical dissimilarity matrix and the aim is to cluster the objects.

Following Oh and Raftery (2007), we propose to select (p, G) based on $p(\boldsymbol{Z}_{pG},\ p,\ G \mid \boldsymbol{D})$, the posterior density function of $(\boldsymbol{Z}, p, G)$ given the data, $\boldsymbol{D}$, evaluated at $\boldsymbol{Z} = \boldsymbol{Z}_{pG}$, the estimated posterior mean of the latent configuration given (p, G). Note that

$$p(\boldsymbol{Z}_{pG}, p, G \mid \boldsymbol{D}) = c \cdot f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G)\, p(\boldsymbol{Z}_{pG}, p, G), \tag{12}$$

where c is a constant, and $f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G)$ and $p(\boldsymbol{Z}_{pG}, p, G)$ are respectively the marginal likelihood and the marginal prior:

$$f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G) = \int f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G, \boldsymbol{\alpha}, \boldsymbol{\lambda}) \prod_{g=1}^{G} p(\alpha_g \mid \sigma_g^2) p(\sigma_g^2) p(\lambda_g) d\alpha_g d\sigma_g^2 d\lambda_g$$
$$p(\boldsymbol{Z}_{pG}, p, G) = \int p(\boldsymbol{Z}_{pG}, p, G, \boldsymbol{\eta}) d\boldsymbol{\eta}.$$

Assuming equal prior probabilities for all $p \in (p_{min}, p_{max})$ and for all $G \in (G_{min}, G_{max})$, Equation 12 can be written as

$$p(\boldsymbol{Z}_{pG}, p, G \mid \boldsymbol{D}) \propto f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G)\, p(\boldsymbol{Z}_{pG} \mid p, G). \tag{13}$$

The marginal prior of $\boldsymbol{Z}$ for given p and G is

$$\begin{aligned} p(\boldsymbol{Z}_{pG} \mid p, G) &= \int p(\boldsymbol{Z}_{pG} \mid p, G, \boldsymbol{\eta})\, p(\boldsymbol{\eta} \mid p, G) d\boldsymbol{\eta} \\ &= (2\pi)^{-npG/2} \times \prod_{g=1}^{G} \frac{\Gamma(a_g + np/2)}{\Gamma(a_g)} \frac{b_g^{a_g}}{\left(b_g + \frac{1}{2}\sum_{k=1}^{p}\sum_{i=1}^{n}(z_{ik}^{g})^2\right)^{(a_g+np/2)}}. \end{aligned} \tag{14}$$

The marginal likelihood is not available in closed form, but it can be approximated using the Laplace-Metropolis method (Lewis and Raftery 1997). More specifically, by defining the $3G$-dimensional vector $\boldsymbol{\theta} = (\boldsymbol{\alpha}, \boldsymbol{\sigma}^2, \boldsymbol{\lambda})$, we obtain

$$f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G) \approx (2\pi)^{3G/2} |\boldsymbol{H}^*|^{1/2} f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G, \boldsymbol{\theta}^*)\, p(\boldsymbol{\theta}^*), \tag{15}$$

where $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} h(\boldsymbol{\theta})$, with $h(\boldsymbol{\theta}) \equiv \log\{f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G, \boldsymbol{\theta})\, p(\boldsymbol{\theta})\}$, and $\boldsymbol{H}^*$ is minus the inverse Hessian of $h(\cdot)$ evaluated at $\boldsymbol{\theta}^*$. Note that when $G = 1$, it is $\boldsymbol{\theta} = (\alpha, \sigma^2)$.

As suggested by Lewis and Raftery (1997), we compute $\boldsymbol{\theta}^*$ as the multivariate median of the posterior sample $\{\boldsymbol{\theta}^{(\ell)} : \ell = 1, \ldots, L\}$, defined as $\boldsymbol{\theta}^* = \arg\min_{\mathbf{q}} \sum_{\ell=1}^{L} |\boldsymbol{\theta}^{(\ell)} - \mathbf{q}|$. As for $\boldsymbol{H}^*$, note that asymptotically it equals the (marginal) posterior covariance matrix. To limit the impact of possible outliers, we use a robust estimator of the posterior covariance matrix, and precisely the minimum covariance determinant estimator (see Rousseeuw 1984; Rousseeuw and Van Driessen 1999).

As pointed out in Oh and Raftery (2001, 2007), a criterion based on $p(\boldsymbol{Z}_{pG}, p, G \mid \boldsymbol{D})$ would favor larger p. Indeed, $p(\boldsymbol{Z}_{pG} \mid p, G)$ is positively related to the scale of $\boldsymbol{Z}$, which tends to decrease as p increases, even without improvement of the fit. To avoid this shrinking effect, two configurations should be compared using the same number of dimensions. Therefore, to compare dimensions p and $(p-1)$ we refer to $\boldsymbol{Z}_p$ and to $\boldsymbol{Z}^*_p = (\boldsymbol{Z}_{p-1} : \boldsymbol{0})$, a p-dimensional configuration obtained by adding to $\boldsymbol{Z}_{p-1}$ a factor with all the coordinates equal to zero. Note that $\boldsymbol{Z}^*_p$ returns the same likelihood as $\boldsymbol{Z}_{p-1}$, and that if $(p-1)$ is the correct dimension, then the optimal solution in the p dimensional space should be (close to) $\boldsymbol{Z}^*_p$.

When $G = 1$, we define $\boldsymbol{Z}_{p1} \equiv \boldsymbol{Z}_p$. In this case, the marginal likelihood reduces to $f(\boldsymbol{D} \mid \boldsymbol{Z}_p, p) = \int f(\boldsymbol{D} \mid \boldsymbol{Z}_{p1}, p, \alpha) p(\alpha \mid \sigma^2) p(\sigma^2) d\alpha d\sigma^2$, and the comparison between configurations with dimensions p and $(p-1)$ can be based on

$$\begin{aligned}
\frac{p(\boldsymbol{Z}_p, p \mid \boldsymbol{D})}{p(\boldsymbol{Z}^*_p, p \mid \boldsymbol{D})} &= \frac{f(\boldsymbol{D} \mid \boldsymbol{Z}_p, p)\, p(\boldsymbol{Z}_p \mid p)}{f(\boldsymbol{D} \mid \boldsymbol{Z}^*_p, p)\, p(\boldsymbol{Z}^*_p \mid p)} \\
&= \frac{f(\boldsymbol{D} \mid \boldsymbol{Z}_p, p)\, p(\boldsymbol{Z}_p \mid p)}{f(\boldsymbol{D} \mid \boldsymbol{Z}_{(p-1)}, p-1)\, p(\boldsymbol{Z}^*_p \mid p)} \\
&= \frac{f(\boldsymbol{D} \mid \boldsymbol{Z}_p, p)}{f(\boldsymbol{D} \mid \boldsymbol{Z}_{(p-1)}, p-1)} \frac{p(\boldsymbol{Z}_p \mid p)}{p(\boldsymbol{Z}_{(p-1)} \mid p-1)} \frac{p(\boldsymbol{Z}_{(p-1)} \mid p-1)}{p(\boldsymbol{Z}^*_p \mid p)} \\
&= \frac{p(\boldsymbol{Z}_p, p \mid \boldsymbol{D})}{p(\boldsymbol{Z}_{(p-1)}, p-1 \mid \boldsymbol{D})} \frac{p(\boldsymbol{Z}_{(p-1)} \mid p-1)}{p(\boldsymbol{Z}^*_p \mid p)} = \frac{p(\boldsymbol{Z}_p, p \mid \boldsymbol{D})}{p(\boldsymbol{Z}_{(p-1)}, p-1 \mid \boldsymbol{D})} A_p\,.
\end{aligned} \tag{16}$$

where A_p can be regarded as a correction factor to the posterior density ratio of $\boldsymbol{Z}_p$ and $\boldsymbol{Z}_{(p-1)}$ for the shrinking effect. Using Equation 14 one can show that

$$-2\log A_p = -n\log(2\pi) - 2\log\left(\frac{\Gamma(a + n(p-1)/2)}{\Gamma(a + np/2)}\right) - n\log\left(b + \frac{1}{2}\sum_{k=1}^{p-1}\sum_{i=1}^{n}\left(z_{ik}^{(p-1)}\right)^2\right), \tag{17}$$

where $a \equiv a_1$, $b \equiv b_1$, and $z_{ik}^{(p-1)}$ is the estimate of the latent space coordinate (for the i-th object and the k-th factor) in the $(p-1)$-dimensional configuration corresponding to $G = 1$. Since the shrinking effect regards only p and not G, we use A_p for all values of G given the same value of p, as suggested in Oh and Raftery (2007). Hence, to choose simultaneously p and G, we propose the following information criterion:

$$\begin{aligned}
DCIC_{1G} &= -2\log f(\boldsymbol{D} \mid \boldsymbol{Z}_{1G}, 1, G) - 2\log p(\boldsymbol{Z}_{1G} \mid 1, G) \\
DCIC_{pG} &= DCIC_{1G} + \sum_{r=2}^{p} -2\log\frac{p(\boldsymbol{Z}_{rG}, r, G \mid \boldsymbol{D})}{p(\boldsymbol{Z}^*_{rG}, r, G \mid \boldsymbol{D})} \\
&= DCIC_{1G} + \sum_{r=2}^{p}\left\{-2\log\frac{f(\boldsymbol{D} \mid \boldsymbol{Z}_{rG}, r, G)}{f(\boldsymbol{D} \mid \boldsymbol{Z}_{(r-1)G}, r-1, G)} \frac{p(\boldsymbol{Z}_{rG} \mid r, G)}{p(\boldsymbol{Z}_{(r-1)G} \mid r-1, G)} - 2\log A_r\right\} \\
&= -2\log f(\boldsymbol{D} \mid \boldsymbol{Z}_{pG}, p, G) - 2\log p(\boldsymbol{Z}_{pG} \mid p, G) - 2\sum_{r=2}^{p}\log A_r,
\end{aligned} \tag{18}$$

where Equation 16 has been used to derive the final expression. The values of p and G corresponding to the minimum $DCIC_{pG}$ should be selected as the best solution. The encouraging results obtained in a small simulation study (not reported here) suggest the reliability of the proposed criterion for the selection of the correct model.

4. The dmbc package

The **dmbc** package allows to fit the model described above and to compute the $DCIC_{pG}$ index for the sequence of models with latent space dimension up to p and number of clusters up to G. Additionally, it includes functionalities for summarizing and plotting all the results. In this section, we provide a detailed description of the package features.

4.1. Object classes

The package uses the S4 system and it is built around six classes whose features and slots are described below.

'`dmbc_data`': Defines instances to represent the input data of a dissimilarity model-based clustering (DMBC) analysis. The classes' slots are: `diss`, a list containing the subjects' dissimilarity matrices, that must be objects with class `dist`; `n`, the number of objects assessed by each subject; `S`, the number of subjects. A further class member, `family`, specifies the distribution of the dissimilarities, but it is not used in the current version of the package and, since we limit attention to binary dissimilarities, at the moment it is internally constrained to `"binomial"`.

'`dmbc_model`': Defines objects providing the main elements of a DMBC model, namely the dimension of the latent space, `p`, the number of clusters, `G`, and `family` having the same content as described above.

'`dmbc_fit`': Defines objects containing the results of a DMBC model fitted using a single MCMC chain. It includes the simulated chains for the model's parameters[6], the Metropolis-Hastings acceptance rates, the list of the observed dissimilarity matrices (`diss`), a list (`dens`) with the log-likelihood, log-prior and log-posterior values at each iteration, the control parameters, the prior hyper-parameters choices, the proposal distribution parameters for the latent positions, and the cluster-specific random-effects α_g's, a list (`dim`) containing the model's dimensions (i.e., n, p, G and S) and the model characteristics (i.e., those described above for the '`dmbc_model`' class).

'`dmbc_fit_list`': Contains a unique member, `results`, that consists of a list of '`dmbc_fit`' objects corresponding to the independent MCMC simulated chains.

'`dmbc_config`': Provides the final estimates: `Z.est` and `Z.sd` respectively contain the estimates of the latent configurations for the n objects in each cluster and the corresponding posterior standard deviation, `cluster` provides the estimate of the subjects' cluster membership, and `est` indicates the estimate type (either posterior means, medians, or modes, or estimates corresponding to the maximum likelihood value attained during the simulations); for convenience, also general information about the fitted model (n, p, etc.) is included. Finally, `chain` is the ID number of the chain used to compute the estimates, and `labels` is an optional character vector of labels to tag the objects in graphical visualizations.

'`dmbc_ic`': Provides information about the comparison of DMBC models based on different values of p and/or G, and specifically the results of the computations discussed in Section 3. In particular, the class slots correspond to the different components of the DCIC index (i.e., `logprior`, `logmlik`, `logcorrfact` and `DCIC` defined in Equation 14, Equation (15), Equation (17) and Equation (18) respectively), the posterior estimates of the compared models

[6]For the latent positions chains, both the original simulations (`z.chain`) and those post-processed using a Procrustes analysis (`z.chain.p`) are provided.

(`post.est`), the estimate type (`est`), and a list (`res_last_p`) of '`dmbc_fit_list`' objects with the results of the fitted DMBC models corresponding to the maximum value of p. The last slot should be used as a starting point in case one wants to extend the comparison to models with higher dimensions. Indeed, as discussed above, the DCIC index is computed incrementally with respect to the value of p.

For all these classes, the package provides specific methods for printing, summarizing and plotting their contents, which are illustrated in more details in the following.

4.2. Model fitting

The package's main function is `dmbc()`, whose primary aim is to set up the problem, verifying the coherency and consistency of the inputs, and possibly filling in with default values all the inputs not specified in the function call. In addition, when more MCMC chains are simulated, the function dispatches the call to perform the requested type of parallel operation.

More operationally, fitting a DMBC model requires the three steps described below.

Step 1. Specification, at least partial, of a set of tuning parameters controlling the model's fitting procedure:

- `nsim`: The number of MCMC iterations (after burn-in).
- `burnin`: The number of burn-in iterations.
- `thin`: The number of iterations between consecutive draws.
- `nchains`: The number of MCMC chains to run.
- `threads`: The number of cores to use in the computations.
- `seed`: An optional random seed.
- `parallel`: A string indicating the type of parallel operation to use for running multiple chains simultaneously. Admissible values are `"no"` (i.e., serial computation, one chain at a time), `"multicore"` and `"snow"`. Specifically, **multicore** and **snow** are the two original packages that have been merged into the **parallel** package that we use internally (R Core Team 2021). We offer the possibility to choose between `"multicore"` and `"snow"` because the first option, which exploits the `mclapply()` function from the **parallel** package, is not available on Windows systems.
- `z.prop`: The variance of the normal proposal distribution for the latent positions (denoted as $\gamma_{\boldsymbol{Z}}^2$ in Section 2.1).
- `alpha.prop`: The variance of the normal proposal distribution for the cluster-specific random effects α_g (denoted as γ_α^2 in Section 2.1).
- `random.start`: A Boolean value indicating whether the initial cluster membership should be chosen randomly or not (see the next two arguments for alternatives).
- `partition`: A numeric vector containing the user-defined initial partition.

- `method`: If the initial partition is not chosen randomly (i.e., `random.start = FALSE`), a Ward hierarchical clustering algorithm is applied to the observed dissimilarities using the distance measure specified through this argument. The allowed distances are those available in the `dist()` function, that is either `euclidean`, `maximum`, `manhattan`, `canberra`, `binary` or `minkowski`, with the default set to the Manhattan distance.
- `procrustes`: A Boolean value indicating whether to post-process the simulated MCMC chains by applying a Procrustes transformation.
- `relabel`: A Boolean value indicating whether to post-process the simulated MCMC chains by running the relabeling algorithm by Celeux *et al.* (2000). Note that turning off the relabeling allows to dramatically reduce the computational burden of the whole algorithm.
- `store.burnin`: A Boolean value indicating whether the burn-in iterations should be stored in the final output[7].
- `verbose`: A Boolean value indicating whether information on the progress of the MCMC algorithm should be printed in the R console[8].

To support the user in setting these parameters, the package includes the `dmbc_control()` auxiliary function which can be used to specify all or only a subset of the above parameters. The parameters that are not explicitly provided will be set to their default values automatically. Section 4.3 provides more details about the meaning of the most relevant tuning parameters and the corresponding default values.

Step 2. Choice of the prior hyper-parameters (see Section 2.1) using a list with elements:

- `eta`: A list of two vectors `a` and `b`, both of size G, containing the hyper-parameters of the Inverse Gamma distributions of the $\boldsymbol{z}_i^g$'s prior variances.
- `sigma2`: A list with two numeric scalars, `a` and `b`, containing the hyper-parameters of the Inverse Gamma distribution of the prior variance of α_g.
- `lambda`: A vector of size G containing the parameters of the Dirichlet prior distribution of $\boldsymbol{\lambda}$.

The package includes an auxiliary function, `dmbc_prior()`, to assist the user in the proper setting of the prior hyper-parameters, which works similarly to the `dmbc_control()` seen above.

Step 3. Call of the `dmbc()` function. After a preliminary check of the validity of the control and prior arguments passed, the function internally generates the model's parameter starting values using the `dmbc_init()` function, and it dispatches the computation using the chosen parallel framework. `dmbc()` internally calls `dmbc_fit()`, the workhorse function that performs all the computations. In particular, for each chain `dmbc_fit()` performs the following substeps.

[7]Hence, the final number of iterations is `floor((nsim + burnin)/thin)` if `store.burnin = TRUE` and `floor(nsim/thin)` otherwise.

[8]Unfortunately, the `verbose` control parameter is ineffective when computations are performed in parallel. In this case, `verbose` produces only limited information regarding the creation and shutting down of a parallel socket cluster.

- *Sub-step 3a*: It runs the MCMC simulation by calling the internal `dmbc_mcmc()` C routine.
- *Sub-step 3b*: It post-processes the latent position chains by applying a Procrustes transformation as described in Section 2.3. This step is performed using the `procrustes()` function from the **MCMCpack** package (Martin, Quinn, and Park 2011).
- *Sub-step 3c*: It relabels all the parameter chains to take care of the label-switching issue. This is implemented internally in the `dmbc_relabel()` C routine.

Note that sub-steps 3b an 3c are performed only if the `procrustes` and `relabel` options in `dmbc_control()` are both set to `TRUE` (default).

After these computations, the control returns to the `dmbc()` function, which may perform an optional final post-processing step (i.e., Procrustes transformation and relabeling) to try making the results of the different chains more consistent in terms of interpretation. This optional final step can be achieved by setting to `TRUE` the `post_all` argument of `dmbc()`.

The output returned by `dmbc()` is an object of class '`dmbc_fit_list`'.

To describe the features of the package, we refer to the simulated data stored in the `simdiss` object[9], which contains $S = 10$ binary dissimilarity matrices. For a preliminary exploration of the data, it is possible to use the `plot()` function, that displays the dissimilarities matrices using a grid of rectangles with colors corresponding to their values (0 or 1):

```
R> data("simdiss", package = "dmbc")
R> plot(simdiss, colors = c("white", "darkgray"), cex.font = 0.75)
```

The picture, reported in Figure 1, highlights that some of the subjects share a common pattern of 0s and 1s. This is particularly evident for subjects from 4 to 8, whose matrices present a top right block of 0s.

We illustrate the estimation process by fitting a model with $p = 2$ dimensions and $G = 3$ clusters, running two independent MCMC chains using the `parallel = "snow"` option, and keeping in the final output one in every 10 iterations:

```
R> p <- 2
R> G <- 3
R> control <- dmbc_control(nsim = 10000, burnin = 20000, thin = 10,
+    nchains = 2, threads = 2, parallel = "snow", seed  = 601,
+    verbose = TRUE)
R> prior <- dmbc_prior(eta = list(a = rep(1, G), b = rep(2, G)))
R> res <- dmbc(simdiss, p = p, G = G, control = control,
+    prior = prior, post_all = TRUE)

--- STARTING PARALLEL SIMULATION OF 2 CHAINS ---
starting worker pid=49949 on localhost:11736 at 16:14:24.946
starting worker pid=49963 on localhost:11736 at 16:14:25.190
--- END OF PARALLEL SIMULATION OF 2 CHAINS ---
```

[9]These data are included in the package. The code for reproducing them is also available as a demo. The demo can be run by typing `demo(simdiss, package = "dmbc")`.

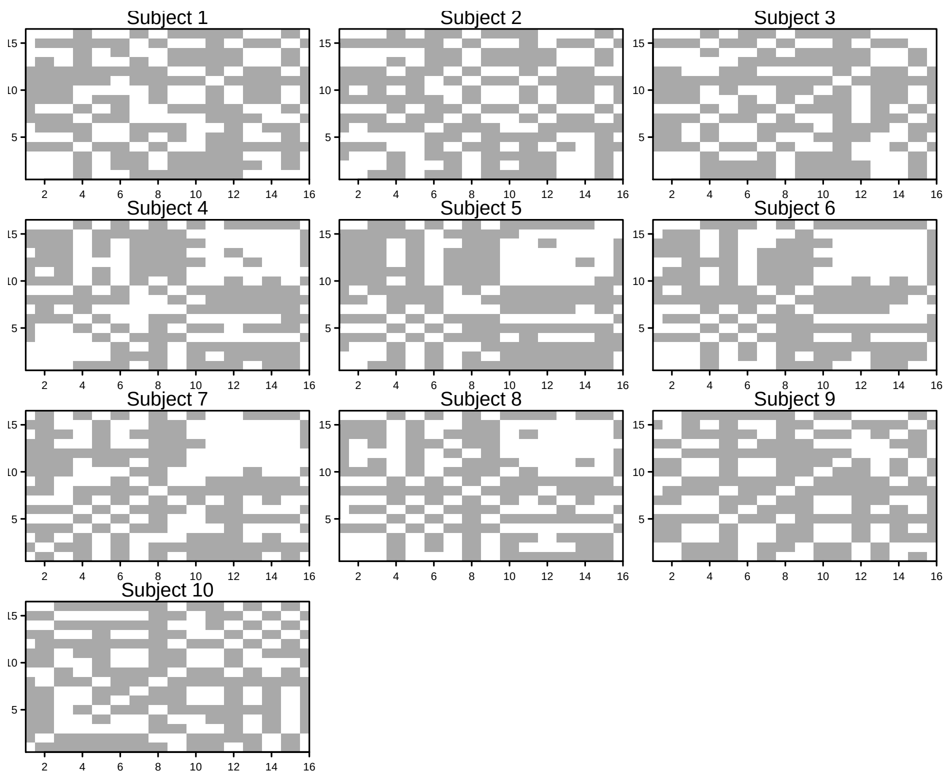

Figure 1: Simulated data. Each panel shows the binary dissimilarity matrix for a subject in the sample. The white color corresponds to the 0s and the dark gray to the 1s in the matrices.

```
Final post-processing of all chains:
   - applying Procrustes transformation...
     0%   10   20   30   40   50   60   70   80   90   100%
     [----|----|----|----|----|----|----|----|----|----]
     |==================================================|
   - relabeling the parameter chain...
     0%   10   20   30   40   50   60   70   80   90   100%
     [----|----|----|----|----|----|----|----|----|----]
     |==================================================|
```

4.3. Diagnostics

The standard approach in the estimation of Bayesian models through MCMC is to let the algorithm run for a prespecified number of iterations to allow the probability distribution of the sampled values converge to the "target" distribution, that is the posterior distribution of the model's parameters. Therefore, a common step in every Bayesian data analysis problem is to inspect the results provided by the MCMC simulation to check whether the algorithm did actually convergence. Even if an in-depth discussion of these procedures goes beyond the scope of the paper, we provide here a brief summary of the most frequently adopted strategies, referring to the classical reference by Gelman and Rubin (1992) and to the recent review by Roy (2020) for a more formal and detailed description.

A very common practice to attain convergence consists in discarding (*burn-in*) the first part of the simulated chains to avoid including in the estimates calculation values of the parameters that belong to regions of low (posterior) density. In addition, to reduce the autocorrelation in the sampled values, one can *thin* the sample, using only one in every k-th iteration.

Based on the theoretical considerations discussed in the references above and on the many experimentation we performed during the development of the package, we chose some default values to support the user in setting these critical tuning parameters. More specifically:

- The default for the length of the burn-in period (`burnin` in our package) was set to 5,000, that is half of the total number of iterations. This is a pretty commonly adopted strategy. Even so, when convergence is slower it is advisable to increase this value.

- The default for the number of iterations after burn-in (`nsim` in the package) was set to 10,000, which has shown to provide good results in most of the cases we considered.

- For thinning (`thin` in the package), the default is set to 1, so that no thinning is applied. Indeed, we think that this choice should be left to the user, because it depends on the complexity of the problem and on possible preliminary results.

A further set of parameters that typically have a strong impact on the "goodness" of the sampled values is represented by the variance of the proposal distributions from which the algorithm advances new potential values. On one hand, too large values of these variances may produce too many rejections, and the algorithm will probably get stuck. On the other hand, too small values will produce too many acceptances, preventing the algorithm to explore regions of the parameter space with potentially higher density. In our algorithm, there are two such variances: the variance of the proposal distribution for the latent coordinates z_{ij}^g, and the variance for the cluster-specific intercepts α_g (see Section 2.1 for details). The default values for these parameters were set respectively to 1.5 and 0.75, which worked pretty smoothly in most examples. However, it is very difficult to suggest proposal variance values that work well for every situation and some trials are always recommendable (at least to confirm that the default values work fine).

Besides and beyond the setting of the tuning parameters, it is always crucially important to monitor the algorithm and to assess whether it converged or not. The simplest, but probably most powerful, graphical tool in this respect is the *trace plot*, reporting the parameter values sampled at each iteration. When the trace plot shows a "stationary" series, that is a sequence of values with no trend and constant variability, one can tentatively conclude that the algorithm achieved convergence.

For the sake of convenience, we linked our package to others developed for MCMC diagnostics. More specifically, the function `dmbc_fit_to_mcmc.list()`, converts the output returned by `dmbc()` into an object of class '`mcmc.list`', which can be inspected using the functions from the **coda** package (Plummer, Best, Cowles, and Vines 2006), the most popular for the analysis of MCMC results. Furthermore, our package allows to produce a variety of graphical visualizations to monitor the simulated chains. These graphs are generated using the functions from the **bayesplot** package (Gabry, Simpson, Vehtari, Betancourt, and Gelman 2019). In the next sections we provide some examples of the visualizations available with `dmbc()`.

4.4. Exploration and visualization of results

For a summary of the obtained results, one can apply the `summary()` function to the object of class '`dmbc_fit_list`' returned by `dmbc()`. The function first converts the output into an `mcmc.list` object and then applies the corresponding `summary()` method. The summary method for '`dmbc_fit_list`' allows specifying two arguments. The first, `include.burnin`, controls whether the burn-in iterations (assuming they have been stored in the original call) should be used or not when computing the summaries. The second, `summary.Z`, controls the printout of the summary for the latent position estimates. For the simulated data example, one gets the following results:

```
R> summary(res, include.burnin = FALSE, summary.Z = FALSE)

Iterations = 20001:29991
Thinning interval = 10
Number of chains = 2
Sample size per chain = 1000

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

             Mean       SD Naive SE Time-series SE
alpha[1]  -3.8594   0.5164 0.011546       0.036506
alpha[2]  -3.3207   0.3528 0.007889       0.024965
alpha[3]  -3.8348   0.7032 0.015724       0.057758
eta[1]     7.1986   2.4715 0.055264       0.112231
eta[2]     4.9174   1.5434 0.034510       0.080073
eta[3]     7.9105   3.2217 0.072040       0.197908
sigma2[1] 38.7963 363.8506 8.135946       8.137515
sigma2[2] 23.9627 165.7760 3.706864       3.706896
sigma2[3] 42.6845 249.5785 5.580745       7.818240
lambda[1]  0.3086   0.1250 0.002795       0.002796
lambda[2]  0.4657   0.1338 0.002992       0.002992
lambda[3]  0.2257   0.1099 0.002457       0.002457

2. Quantiles for each variable:

              2.5%     25%     50%     75%    97.5%
alpha[1]  -4.97023 -4.1876 -3.8319 -3.5002  -2.8988
alpha[2]  -4.03252 -3.5614 -3.3101 -3.0655  -2.6773
alpha[3]  -5.32564 -4.2902 -3.7899 -3.3448  -2.5625
eta[1]     3.58702  5.4571  6.7880  8.4850  13.0449
eta[2]     2.58887  3.8333  4.6855  5.7719   8.7478
eta[3]     3.44995  5.7142  7.3069  9.4781  16.0688
sigma2[1]  1.75988  4.8764  9.6093 21.8536 190.0575
sigma2[2]  1.40564  3.6178  7.1333 15.9046 128.8550
sigma2[3]  1.54723  4.5953  9.5105 24.3741 231.8324
lambda[1]  0.09830  0.2173  0.2984  0.3905   0.5765
```

```
lambda[2]  0.21370  0.3707  0.4607  0.5635   0.7283
lambda[3]  0.05238  0.1423  0.2129  0.2975   0.4621
```

The model's estimates of the objects' latent positions and of the subjects' cluster membership can be obtained by calling the `dmbc_get_configuration()` function, which returns an object of class '`dmbc_config`' (see Section 4.1). By default, the `dmbc_get_configuration()` function uses the results of the first simulated chain to compute the summaries. It is possible to use another chain by setting the `chain` argument accordingly[10]. For the simulated data example we obtain:

```
R> est <- dmbc_get_configuration(res)
R> summary(est)

Dissimilarity Model Based Clustering -- Latent configuration estimates
Number of objects (n): 16
Number of latent dimensions (p): 2
Number of subjects (S): 10
Number of clusters (G): 3
Family: binomial
Chain ID: 1
Estimate type: posterior mean
Cluster sizes:
             Size
Cluster 1       3
Cluster 2       5
Cluster 3       2
Subjects assigned to each cluster:
Cluster1: 1, 2, 3
Cluster2: 4, 5, 6, 7, 8
Cluster3: 9, 10
Latent configuration estimates (Z):
--------------------------------- Cluster 1 ---------------------------------
        Dimension 1  Dimension 2
i = 1        2.5827       1.1961
i = 2        2.4143       0.5661
...
i = 16       2.1879       0.4982
--------------------------------- Cluster 2 ---------------------------------
        Dimension 1  Dimension 2
i = 1        2.2937       0.4323
i = 2        2.7216       0.5056
...
i = 16       2.7855       0.5391
--------------------------------- Cluster 3 ---------------------------------
```

[10]The `dmbc_get_configuration()` function does not use the results from all available chains because it may be that the cluster memberships are not consistent across independent runs of the algorithm (i.e., different MCMC chains).

```
        Dimension 1  Dimension 2
i = 1        0.6337      -4.1890
i = 2       -0.3493      -3.5771
...
i = 16       0.9984      -4.0258
```

To access the cluster memberships more explicitly, one can use the `clusters()` generic method, which is imported from the **modeltools** package (Hothorn, Leisch, and Zeileis 2020):

```
R> clusters(est)
```

```
[1] 1 1 1 2 2 2 2 2 2 3 3
```

Note that subjects from 4 to 8, who exhibited common dissimilarities patterns (see Figure 1), are correctly clustered together.

The package includes a method for plotting the objects in the estimated latent space, possibly identifying the points in the diagram with the labels provided in the call to function `dmbc_get_configuration()`. If not explicitly given, points in the diagram are labeled from 1 to n:

```
R> set.seed(101)   # needed to replicate the label positioning
R> library("bayesplot")
R> color_scheme_set(rep("black", 6))
R> graph <- plot(est, size = 2, size_lbl = 3, label_objects = TRUE)
R> graph + panel_bg(fill = "white", color = NA)
```

The plot of the latent configuration estimates shows that the subjects clustered in different groups have different perceptions about the similarities/dissimilarities among the objects they evaluated.

The results of a fitted DMBC model can be represented graphically in many different ways. In particular, as we already mentioned in Section 4.3, the **dmbc** package exploits the capabilities of the **bayesplot**, a wrapper of **ggplot2** (Wickham 2016), that is focused on the visualization of MCMC simulations. To visualize the results of a fitted DMBC model, one needs to use the `plot()` function specifying the type of the desired plot in the `what` argument. All the possibilities offered by **bayesplot** have been mirrored in **dmbc**[11]. In Figures 3–5 we provide some examples of the available plot types, which are commonly used to graphically assess the algorithm convergence:

```
R> color_scheme_set("brightblue")
R> plot(res, what = "areas_ridges", regex_pars = "lambda",
+    include.burnin = TRUE)
R> color_scheme_set("mix-blue-red")
```

[11]Admissible values for the `what` argument are `acf`, `areas`, `dens`, `hex`, `hist`, `intervals`, `neff`, `pairs`, `parcoord`, `recover`, `rhat`, `scatter`, `trace`, `violin` and `combo`. For more details on these visualization we suggest to see the corresponding **bayesplot** documentation.

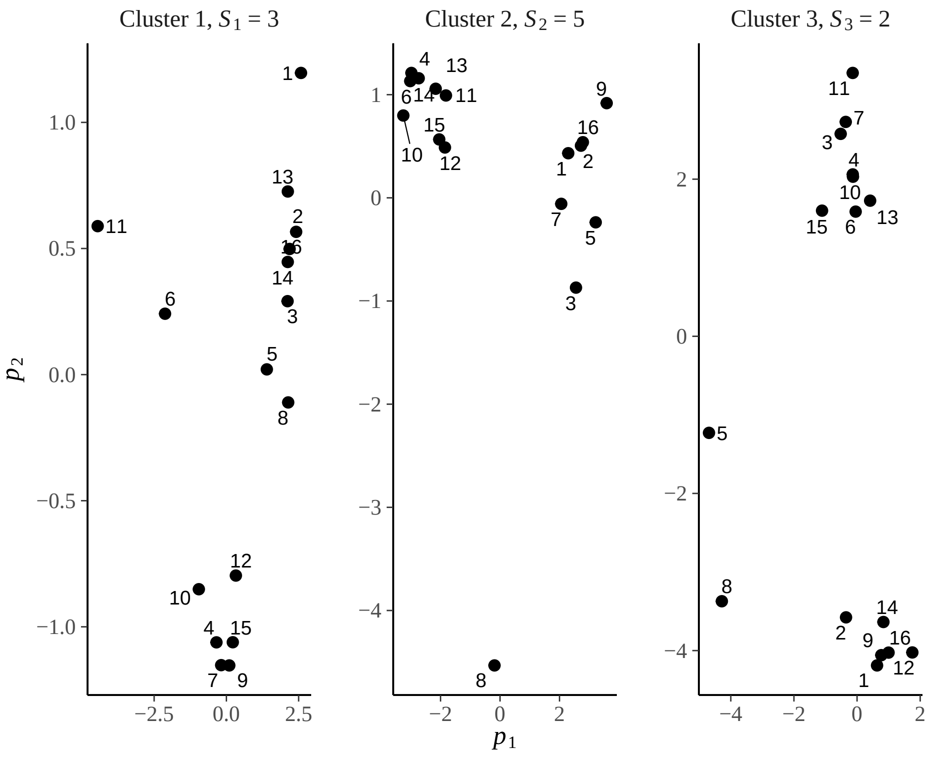

Figure 2: Simulated data. Plot of the estimated 2-dimensional configuration in the $G = 3$ clusters. The titles of the plots report the number of subjects assigned to each cluster; the points in the plot are identified by the corresponding object's label.

```
R> plot(res, what = "trace", regex_pars = "eta", include.burnin = TRUE,
+    transformation = "log",
+    facet_args = list(labeller = ggplot2::label_parsed))
R> color_scheme_set("viridisA")
R> plot(res, what = "pairs", regex_pars = "alpha", diag_fun = "hist",
+    off_diag_fun = "hex", include.burnin = TRUE)
```

In particular, Figure 3 shows the density plots of the λ_g parameters. A unimodal density for the values from a MCMC simulation, like those represented in the figure, typically provides evidence in favor of convergence. Figure 4 reports the trace plots for the log-transformed η_g parameters; the logarithmic transformation is particularly useful for variances, that typically exhibit right-skewed distributions. The plots show no trend or change in the variability, suggesting that the algorithm reached convergence also for these parameters. Finally, Figure 5 displays the histograms for the simulated α_g parameter values, together with the graphical representation of their pairwise joint distributions. Again, no unexpected patterns are visible in these diagrams, and this further corroborates the conclusion that the algorithm converged.

4.5. Model selection

The **dmbc** package provides the `dmbc_IC()` function that implements the approach for model

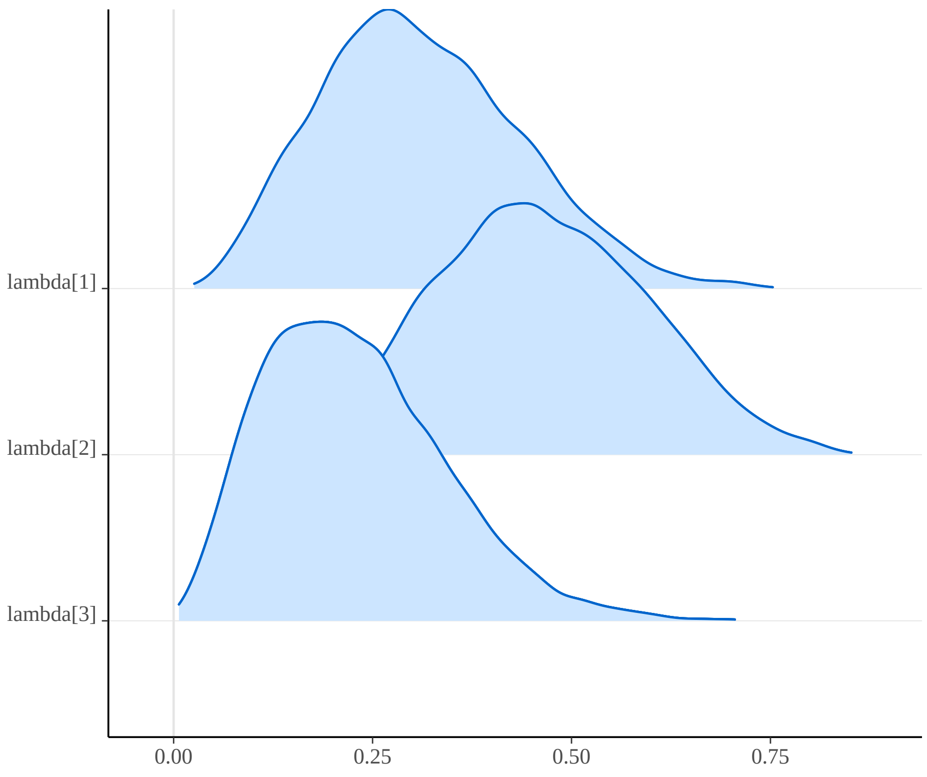

Figure 3: Simulated data. Plot of overlapping ridge-lines for the simulated $\boldsymbol{\lambda}$ parameters of a fitted DMBC model with $p = 2$ latent dimensions and $G = 3$ clusters.

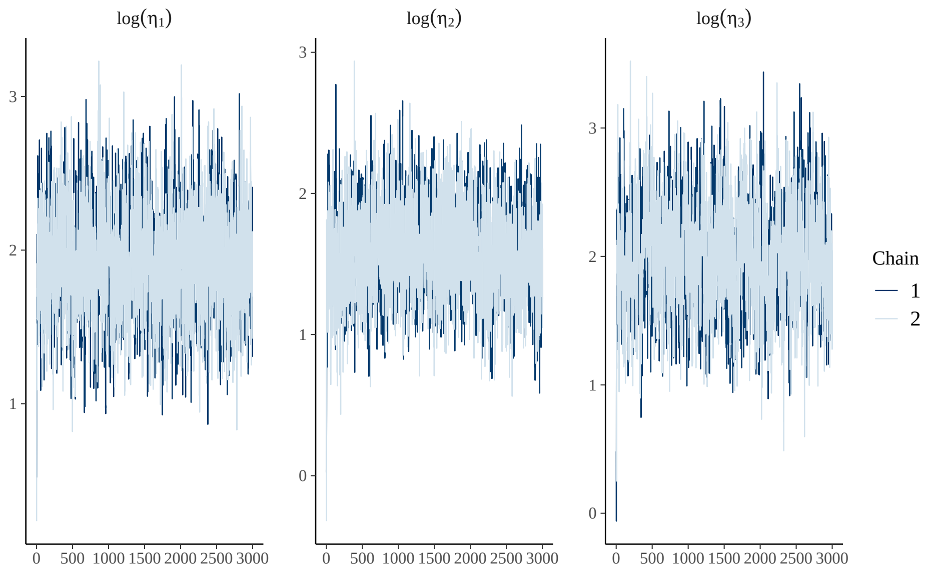

Figure 4: Simulated data. Trace plots for the logarithm of the simulated $\boldsymbol{\eta}$ parameters of a fitted DMBC model with $p = 2$ latent dimensions and $G = 3$ clusters.

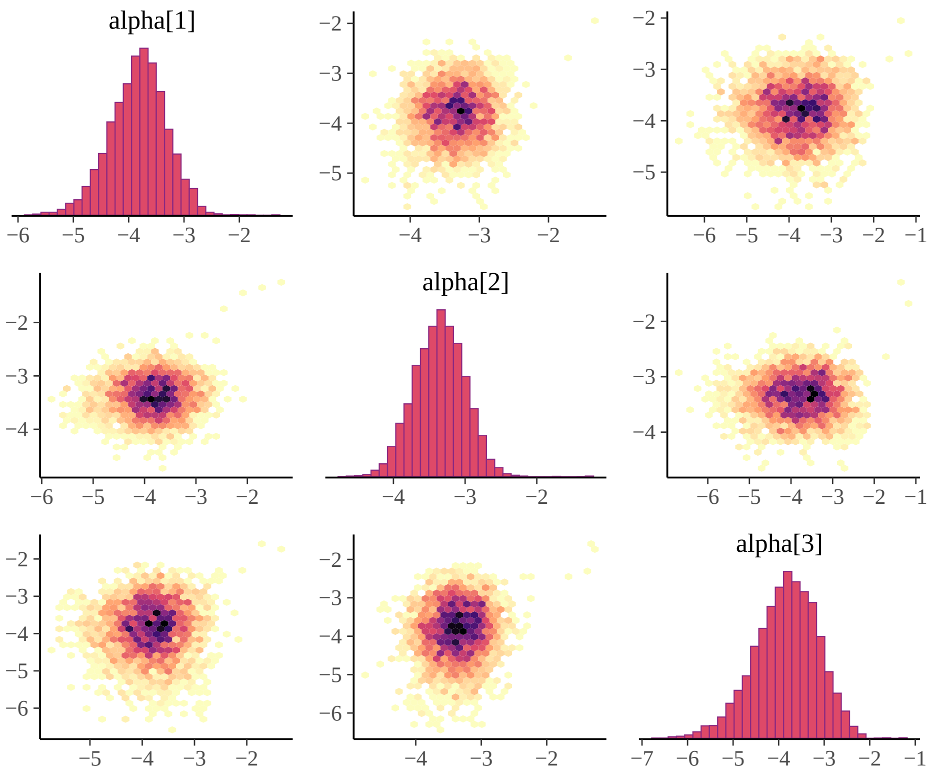

Figure 5: Simulated data. Hexagonal heat maps of bin counts, with corresponding histograms on the diagonal panels for the simulated $\boldsymbol{\alpha}$ parameters of a fitted DMBC model with $p = 2$ latent dimensions and $G = 3$ clusters.

selection described in Section 3. Together with the data, control and prior choices as for `dmbc()`, this function also requires to specify `pmax`, the maximum value of the latent space dimension one wants to consider, and `Gmax`, the maximum number of cluster solutions to compare. In addition, one may choose the estimate type to use in the computations. By default, the posterior means are used, but other available possibilities are posterior medians or modes, or the estimates corresponding to the maximum likelihood value attained during the simulation. `dmbc_IC()` returns an object of class '`dmbc_ic`' containing all the results, and in particular the values of the DCIC index corresponding to the requested models.

As an illustration, we complete the analysis of the simulated data by comparing the DMBC models with p from 1 to 3 and with G from 1 to 4, with priors set to the default values. To speed up the process, we choose a small number of iterations (for brevity, we omit the output):

```
R> pmax <- 3
R> Gmax <- 4
R> control <- list(burnin = 7000, nsim = 3000, thin = 10,
+    z.prop = 1.5, alpha.prop = 0.75, random.start = TRUE,
+    verbose = TRUE, store.burnin = TRUE, seed = 301)
R> ic <- dmbc_IC(data = simdiss, pmax = pmax, Gmax = Gmax,
+    control = control)
```

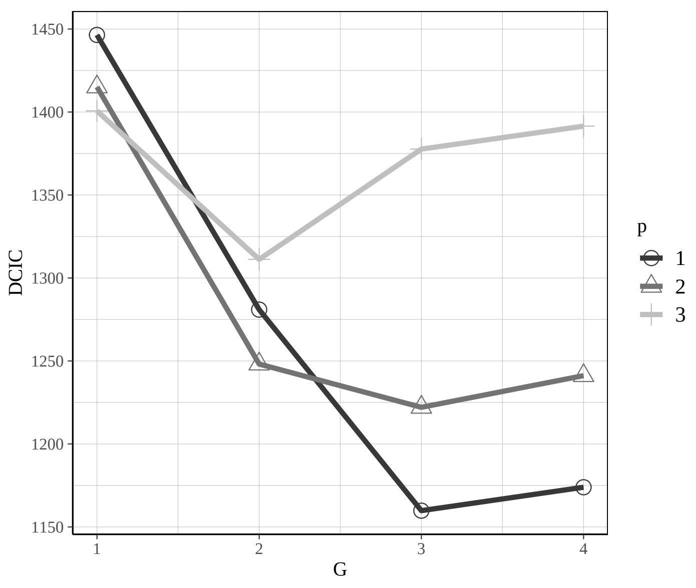

Figure 6: Simulated data. Graphical representation of the DCIC index values for a set of DMBC models with p between 1 and 3 and G between 1 and 4. The best model, corresponding to the minimum DCIC value, is the one with $p = 1$ and $G = 3$.

The values of the DCIC index for the requested models can be obtained by printing the object returned by `dmbc_IC()`:

```
R> print(ic)

Dissimilarity Model Based Clustering -- Model choice
Latent dimensions (p) included: from 1 to 3
Number of clusters (G) included: from 1 to 4
Family: binomial
Dissimilarity Model Based Clustering Information Criterion (DCIC):
          G = 1     G = 2     G = 3     G = 4
p = 1   1446.45   1280.94   1159.81   1173.91
p = 2   1415.12   1248.14   1222.07   1241.16
p = 3   1400.68   1311.20   1377.67   1391.48
```

The best model seems to be the one with $p = 1$ and $G = 3$, which is characterized by the minimum DCIC. To ease the comparison of the DCIC values, one can also visualize the results using the plot method for '`dmbc_ic`' objects (see Figure 6):

```
R> color_scheme_set(c("black", "white", "darkgray", "white",
+    "gray", "white"))
R> plot(ic, size = c(4, 1.5)) +
+    theme(panel.grid.major = element_line(colour = "gray", size = .05),
+      panel.grid.minor = element_line(colour = "gray", size = .05),
+      panel.border = element_rect(colour = "black", fill = NA),
+      panel.background = element_blank())
```

The package also includes a convenient procedure for updating the evaluation of p and G by increasing their maximum values. This can be done using the `update` method for '`dmbc_ic`' objects imported from the **stats4** package (R Core Team 2021). For example, the following code allows to add the DCIC indices for models with $p = 4$ to those previously obtained, while maintaining the same range of values for G (output not reported here):

```
R> new.pmax <- pmax + 1
R> new.ic <- update(ic, pmax = new.pmax, Gmax = Gmax)
R> plot(ic, size = c(4, 1.5))
```

5. Empirical application

We refer to data presented in Takane, Jung, and Takane (2009) relative to judgments on the similarity between $n = 18$ animals expressed by $S = 20$ subjects. Each subject was asked to divide the animals into as many groups as needed, based on their similarity. In their work, Takane *et al.* (2009) build a summary dissimilarity matrix whose (i, j)-th entry contains the number of subjects who did not place the i-th and the j-th animal into the same group. A standard (nonmetric) MDS with the Euclidean distance was used to analyze the data, and a three-dimensional solution was chosen "primarily for ease of presentation" (Takane *et al.* 2009, p. 233).

This approach is based on the implicit assumption that the subjects' dissimilarity matrices are homogeneous, so that a consensus matrix effectively summarizes the data. However, one could be interested in investigating whether such assumption is supported by data, and to study more in depth the possible differences in the subjects' perceived dissimilarities, by identifying clusters of subjects with similar perceptions. Therefore, for each subject we build a dissimilarity matrix whose entries signal for each pair of animals whether they were placed in the same group (0) or not (1). The resulting binary matrices, available in the `animals` object within the package, can be visualized as in Figure 7:

```
R> data("animals", package = "dmbc")
R> plot(animals, colors = c("white", "darkgray"), cex.font = 0.75)
```

We first apply the procedure to simultaneously select p and G based on the DCIC index. Thus, for each combination of (p, G), with $1 \leq p \leq 5$ and $1 \leq G \leq 5$, we apply our algorithm using 20,000 MCMC iterations, with the first 10,000 used for warming-up (the ranges for p and G were chosen based on a preliminary analysis using a reduced number of iterations). The prior and proposal variance parameters are set as illustrated in Section 2.1.

```
R> pmax <- 5
R> Gmax <- 5
R> prm.prop <- list(z = 2.5, alpha = 1)
R> control <- list(burnin = 10000, nsim = 10000, z.prop = prm.prop$z,
+    alpha.prop = prm.prop$alpha, verbose = TRUE, seed = 201)
R> animals_ic <- dmbc_IC(animals, pmax = pmax, Gmax = Gmax,
+    control = control)
R> animals_ic
```

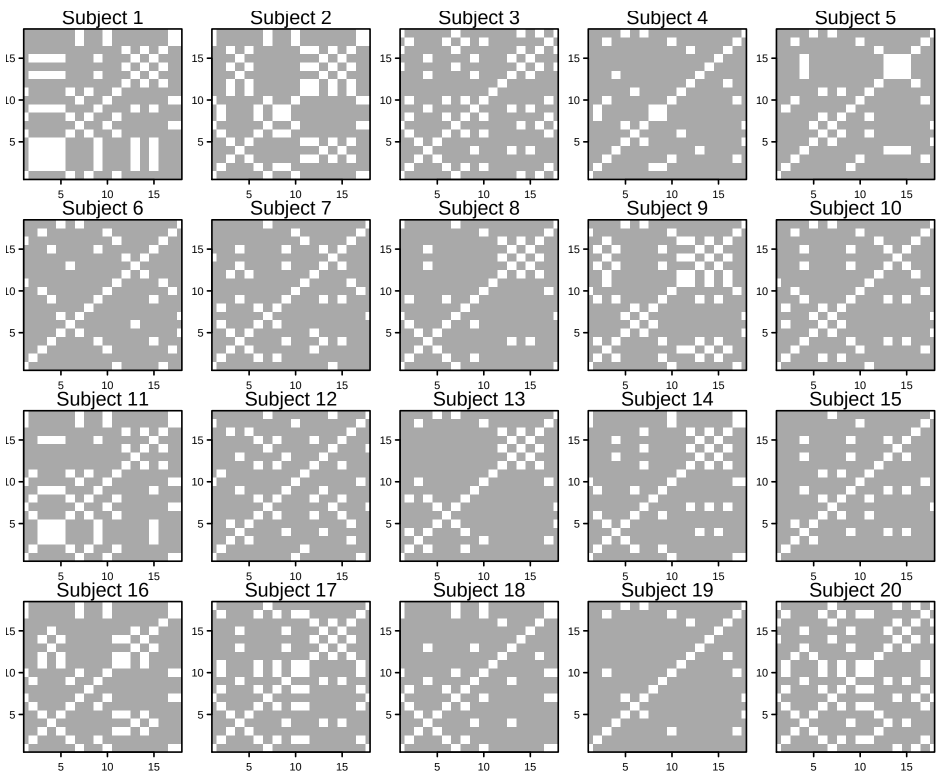

Figure 7: Animals data. Each panel shows the binary dissimilarity matrix for a subject in the sample. The matrices have been obtained using the data in Takane *et al.* (2009), by setting their elements equal to 0 or 1 depending on whether a pair of animals is placed or not in the same group by the subject. The white color corresponds to the 0s and the dark gray to the 1s in the matrices.

```
Dissimilarity Model Based Clustering -- Model choice
Latent dimensions (p) included: from 1 to 5
Number of clusters (G) included: from 1 to 5
Family: binomial
Dissimilarity Model Based Clustering Information Criterion (DCIC):
          G = 1     G = 2     G = 3     G = 4     G = 5
p = 1   2129.01   1723.20   1688.98   1672.74   1895.29
p = 2   1773.61   1810.01   2121.70   1885.02   2564.19
p = 3   1682.60   2058.95   2224.82   2416.28   2707.07
p = 4   1682.79   1919.54   2346.64   2527.38   2827.89
p = 5   1707.99   2104.38   2139.97   2642.78   3062.09
```

These results suggest $(p, G) = (1, 4)$ as the best solution. Therefore, we refit the selected model using a 100,000 iterations for burn-in followed by further 50,000 iterations. To avoid strong dependence along the parameter chains, we choose to retain only 1 in every 10 iterations. Finally, to allow for a better mixing, we set the proposal distribution variances to relatively large values, namely $\gamma_{\boldsymbol{Z}} = 3$ and $\gamma_\alpha = 1$. Below is the code to simultaneously fitting two chains using the `snow` protocol:

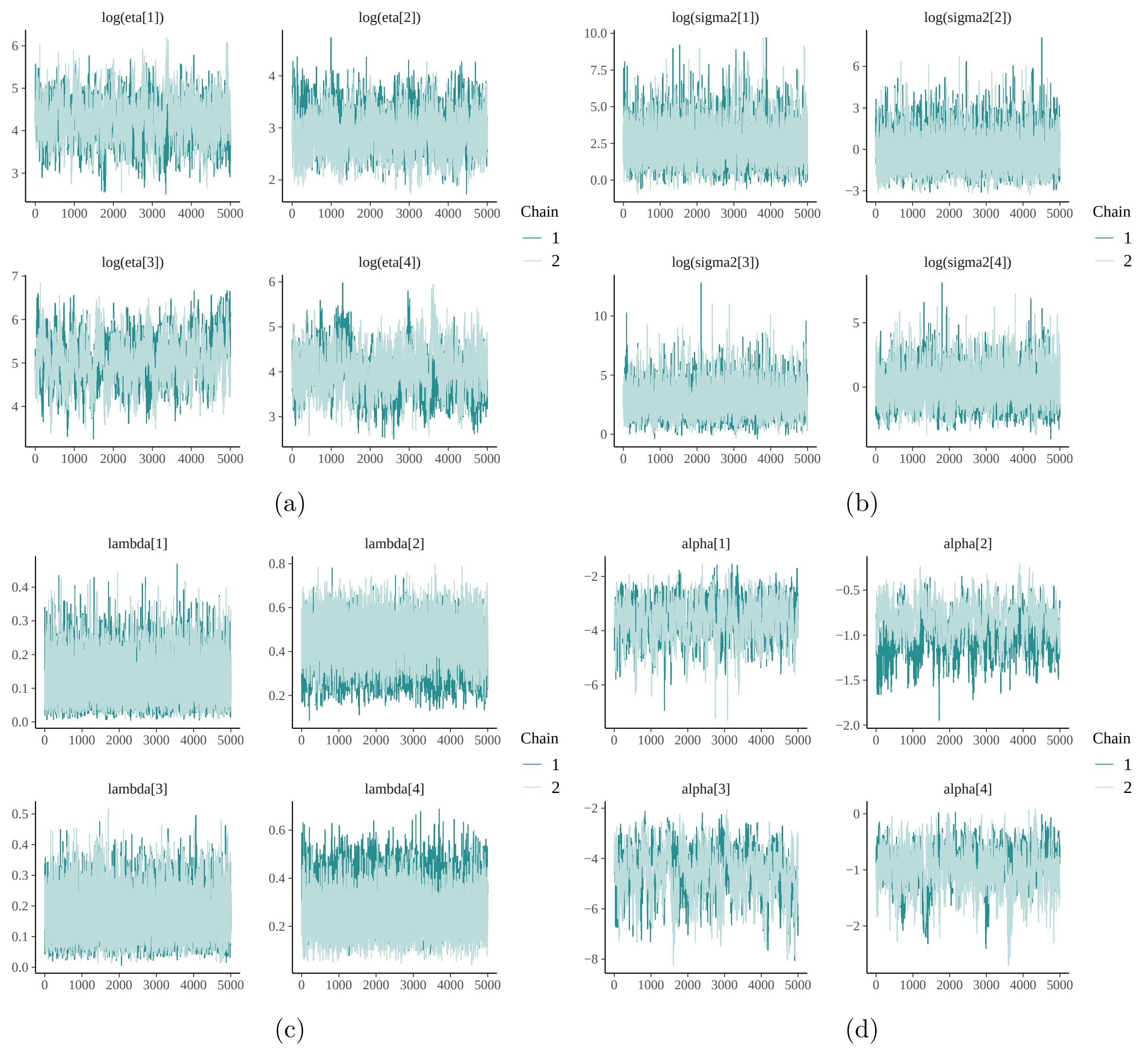

Figure 8: Animals data. Trace plots for the fitted DMBC model with $(p, G) = (1, 4)$ for the following parameters (a) (log-transformed) η_g's. (b) (log-transformed) σ_g^2's. (c) λ_g's. (d) α_g's.

```
R> prm.prop <- list(z = 3, alpha = 1)
R> control <- list(burnin = 100000, nsim = 50000, thin = 10,
+    z.prop = prm.prop$z, alpha.prop = prm.prop$alpha,
+    verbose = TRUE, seed = 1406, nchains = 2, threads = 2,
+    parallel = "snow")
R> animals_1_4 <- dmbc(animals, p = 1, G = 4, control = control,
+    post_all = TRUE)
```

The trace plots for the fitted model (see Figure 8) are obtained as follows:

```
R> color_scheme_set("teal")
```

```
R> plot(animals_1_4, what = "trace", regex_pars = "eta",
+     transformation = "log")
R> plot(animals_1_4, what = "trace", regex_pars = "sigma2",
+     transformation = "log")
R> plot(animals_1_4, what = "trace", regex_pars = "lambda")
R> plot(animals_1_4, what = "trace", regex_pars = "alpha")
```

These diagrams allow to conclude that the simulated chains behave properly.

Figure 9 reports the estimated posterior means of the MDS configurations for the $G = 4$ identified clusters obtained with the `dmbc_get_configuration()` function[12]:

```
R> animals_lbl <- attr(animals@diss[[1]], "Labels")
R> z <- dmbc_get_configuration(animals_1_4, labels = animals_lbl)
R> color_scheme_set(rep("black", 6))
R> z_p <- plot(z, label_objects = TRUE, nudge_x = 1.5)
R> z_p + panel_bg(fill = "white", color = NA)
R> drop(z@Z.est)
```

```
         g = 1      g = 2      g = 3      g = 4
be   7.8912752  6.5106375 -21.455187   8.224499
cm   0.2380378 -3.8663022   2.605586   3.829071
ct  -3.4236000  1.8068985  -7.849494 -13.674802
cw  -3.4194108 -6.6235717  15.044190   1.428815
dg  -3.4104635  2.0627046  -7.820329  -8.948322
el   4.1662027 -3.4291387   2.599000   5.570218
gf  11.5683549  5.3200457 -21.472301  -8.852644
fx   4.1953358 -2.7289153   2.606050   4.723642
hs  -3.4355190 -5.0130349  15.045028   2.329998
li  11.5609187  6.8156971   2.617145 -13.787044
mk   4.1867203  0.3508692   2.732718   6.421884
ms -12.9864275  1.2205536 -12.737961  -3.632380
pg  -3.9004318 -6.6333716  15.030775   1.089844
rb -12.9862458  2.0959669 -17.909004  -1.743951
sh  -3.4317612 -6.5386677  15.042455   1.068187
sq -13.0303637  1.4428922 -17.886356  -3.491393
tg  11.5594469  6.8165351   2.625455 -13.786998
wf  11.5698326  5.6893051 -21.485886  -8.853681
```

The clusters' configurations are displayed along the 45 degrees line to reduce label cluttering and to better appreciate the arrangement of the objects (animals) along the single extracted factor. Subjects in clusters 1 and 3 are those who arranged the animals in the neatest and simplest way. Subjects in cluster 1 (i.e., subjects 1 and 11) mostly distinguish between farm and house animals, animals living in a wild habitat (divided into two groups – elephant,

[12]Codes for animals: Bear: be; Camel: cm; Cat: ct; Cow: cw; Dog: dg; Elephant: el; Giraffe: gf; Fox: fx; Horse: hs; Lion: li; Monkey: mk; Mouse: ms; Pig: pg; Rabbit: rb; Sheep: sh; Squirrel: sq; Tiger: tg; Wolf: wf.

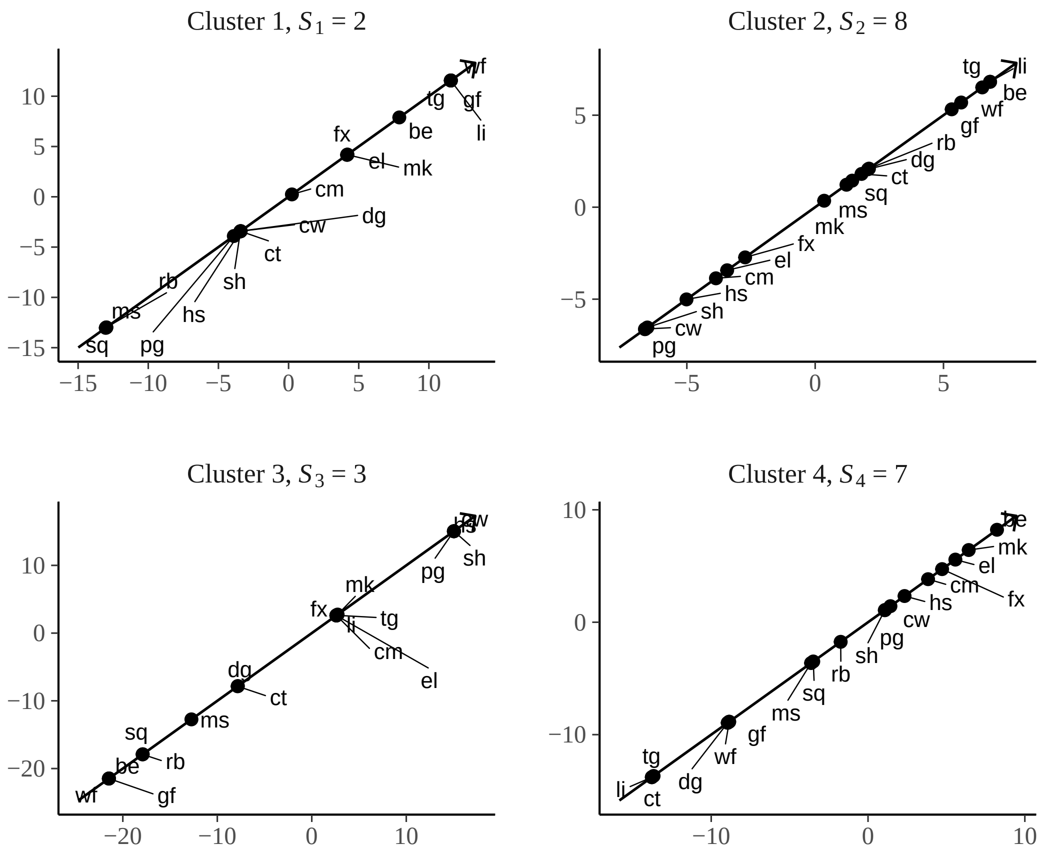

Figure 9: Animals data. Estimated posterior means of the one-dimensional configurations for the $G = 4$ clusters of subjects. To avoid label cluttering, the estimated latent configurations are represented along the 45 degrees line. Clusters compositions (subjects): 1: $(1, 11)$; 2: $(7, 8, 9, 12, 14, 16, 18)$; 3: $(3, 17, 20)$; 4: $(4, 5, 6, 10, 13, 15, 19)$. Codes for animals: Bear: be; Camel: cm; Cat: ct; Cow: cw; Dog: dg; Elephant: el; Giraffe: gf; Fox: fx; Horse: hs; Lion: li; Monkey: mk; Mouse: ms; Pig: pg; Rabbit: rb; Sheep: sh; Squirrel: sq; Tiger: tg; Wolf: wf.

fox and monkey on one side, tiger, lion, giraffe, and wolf on the other), and rodents. The singletons laying along the MDS factor (namely, camel and bear) are projected between two classes because they are differently classified by the two subjects in the cluster. Subjects in cluster 3 (3, 17 and 20) present a similar classification, even if they distinguish between farm and house animals (cat and dog), divide wild animals differently, and quite weirdly perceive squirrel and rabbit as similar to giraffe, bear and wolf. As for cluster 4 (subjects 4, 5, 6, 10, 13, 15 and 19), on one extreme, the farm animals are grouped and placed relatively close to the big four-footed animals (camel, elephant) and to the fox, which are not homogeneously classified by the cluster's subjects. On the opposite side, the placement of felines (lion, tiger and cat) and canines (wolf and dog) can be noted. In the center of the configuration we find the rodents. Further, monkey and bear are projected toward the end of the dimension; their positions can be ascribed to an imperfect fit of these animals with the other identified throngs. Finally, cluster 2 (subjects 2, 7, 8, 9, 12, 14, 16 and 18) includes subjects with the most articulated system of classification.

6. Conclusion

In this work, we introduced **dmbc**, an R package that extends to three-way binary data the Bayesian multidimensional scaling approach by Oh and Raftery (2007). Our procedure provides a partition of the set of dissimilarity matrices into G groups, and simultaneously returns a p-dimensional MDS configuration for each cluster. Embedding the problem in a Bayesian framework has the advantage of allowing the development of a Bayesian criterion to jointly select G and p. Furthermore, it is possible to obtain credibility intervals for all the estimated parameters as well as for the objects' coordinates in the different clusters' configurations.

Our procedure is flexible and includes RMDS and WMDS as particular cases. If the configuration underlying the identified clusters is unique and the individual differences are only due to a different system of weights, then the cluster-specific configurations will be similar except for cluster-specific scaling factors. Clearly, a model based on a unique configuration with a possibly different system of weights is easier to interpret compared to the case when many configurations are obtained. Even so, the comparison of the configurations across the clusters' can be pursued using a Procrustean analysis (see for example Gower and Dijksterhuis 2004) and rotating the estimated configurations to maximum similarity with a reference configuration.

Even if our focus here is very specific, a nice feature of our model-based three-way MDS is its versatility. Indeed, the model and the package can be extended along different directions. First, our approach can be adapted to deal with multinomial and numerical (i.e., non-binary) dissimilarity data by suitably modifying the assumption (Equation 3) on the dissimilarity distribution. Second, it can be enriched by introducing subject-level covariates, such as socio-demographic characteristics, assuming that they influence either the cluster membership, or the linear predictor (Equation 1), or both. Third, it is possible to allow for asymmetric dissimilarity matrices. All these ideas are currently under development.

One limitation of the model regards the assumption that the cluster-specific latent spaces have the same dimension, p. This might lead to overestimate the latent space dimension for some clusters. A simple way to overcome this issue is to apply principal components analysis to the estimated cluster-specific configurations to assess if one or more clusters present redundant dimensions. Although one could envision an extension of the model that allows for a different number of dimensions of the configuration spaces, this would come at the cost of a more complicated procedure for selecting the optimal value for p in the different clusters.

In the MCMC algorithm presented in the paper we use normal proposal densities. This choice is motivated by the difficulty in identifying suitable distributions in a complex multivariate setting like the one analyzed here. An alternative strategy, which has been largely discussed in the literature, is the surrogate proposal approach, used for example in Gormley and Murphy (2010), that approximates a target distribution by simplifying its terms through a quadratic multivariate Taylor expansion. This allows constructing proposal distributions which better approximate problematic full conditional distributions from which sampling is required, and allows the Metropolis-Hastings algorithm to explore more efficiently the parameters' space.

A further limitation of our proposal concerns the DCIC criterion. For given values of p and G, the computation of the DCIC requires the estimation of all models with dimensions from 1 to $(p-1)$, and this does not permit to focus on a restricted set of (hypothesized) promising dimensions. The estimation of the model over a wide range of dimensionality can therefore be very time consuming. One possibility to overcome this issue is to adapt to our context the approach introduced by Oh (2012) for choosing the MDS dimensionality in the case of a single dissimilarity matrix. In this procedure, for each factor in an MDS configuration, a prior is assumed which is a mixture of a point mass at 0 and a continuous distribution for the rest of the parameter space. This prior specification leads to a marginal posterior distribution which is a mixture of the same form of the prior, where the mixing weight of the continuous distribution can be regarded as a measure of significance for the factor itself. Hence, in such framework the optimal dimension of the configuration coincides with the number of factors having high posterior probabilities of being non-zero.

Our methodology does not account for uncertainty in the number of clusters G and the latent space dimension p, which are both assumed to be unknown but fixed quantities. This clearly represents a further direction of research, which can be carried on using a trans-dimensional MCMC approach, such as the reversible jump sampling (Green 1995; Richardson and Green 1997; Brooks, Giudici, and Roberts 2003), in which the dimension of the parameter space can change from one iteration to the next. Alternatively, one can achieve a higher degree of flexibility by adopting a nonparametric perspective, for example using Dirichlet process mixtures (see for example Ferguson 1973; Escobar and West 1995; MacEachern and Müller 1998; Kyung, Gill, and Casella 2010; Müller, Quintana, Jara, and Hanson 2015 provides a comprehensive survey of these models).

Still with reference to the selection of the best model, we note that, up to a certain extent, the DCIC index tends to favor an increase in the number of clusters over an increase in the number of dimensions of the MDS configuration. As a consequence, when the sample contains a large number of subjects with very peculiar perceptions, one may get a DCIC profile showing a very slow monotonic decay with respect to the number of clusters. In these cases, model selection becomes an issue, since one may find a "global" optimum only selecting a very high value for G. When this occurs, we suggest to adopt a more heuristic procedure that mimics the strategy used to identify the dimension of factorial configurations through the well-known scree plot. Thus, in these situations we suggest to select as a reasonable value of G the point where the slope of the DCIC curve levels off showing the classical "elbow".

As a final remark, for what regards the computational efficiency of the algorithm implementation, we note that the algorithm scales well with respect to the number of subjects (S) and to the number of clusters (G), while it tends to slow down when the number of objects (n) increases.

Acknowledgments

We thank the associate editor and two anonymous reviewers for the useful comments that significantly improved the manuscript.

References

Armstrong II DA, Bakker R, Carroll R, Hare C, Poole KT, Rosenthal H (2014). *Analyzing Spatial Models of Choice and Judgment with R*. Chapman & Hall/CRC. doi:10.1201/b16486.

Bocci L, Vichi M (2011). "The K-INDSCAL Model for Heterogeneous Three-Way Dissimilarity Data." *Psychometrika*, **76**, 691–714. doi:10.1007/s11336-011-9225-5.

Borg I, Groenen PJF (2005). *Modern Multidimensional Scaling*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-1-4757-2711-1.

Brooks SP, Giudici P, Roberts GO (2003). "Efficient Construction of Reversible Jump MCMC Proposal Distributions." *Journal of the Royal Statistical Society B*, **65**, 3–55. doi:10.1111/1467-9868.03711.

Carpenter B, Gelman A, Hoffman MD, Lee D, Goodrich B, Betancourt M, Brubaker M, Guo J, Li P, Riddell A (2017). "Stan: A Probabilistic Programming Language." *Journal of Statistical Software*, **76**(1), 1–32. doi:10.18637/jss.v076.i01.

Carroll JD, Chang JJ (1970). "Analysis of Individual Differences in Multidimensional Scaling via an N-Way Generalization of 'Eckart-Young' Decomposition." *Psychometrika*, **30**. doi:10.1007/bf02310791.

Celeux G, Hurn M, Robert CP (2000). "Computational and Inferential Difficulties with Mixture Posterior Distributions." *Journal of the American Statistical Society*, **95**(451), 957–970. doi:10.1080/01621459.2000.10474285.

De Leeuw J (1977). "Applications of Convex Analysis To Multidimensional Scaling." In JR Barra, F Brodeau, G Romier, B Van Cutsem (eds.), *Recent Developments in Statistics*, pp. 133–145. North Holland.

De Leeuw J, Mair P (2009). "Multidimensional Scaling Using Majorization: SMACOF in R." *Journal of Statistical Software*, **31**(3), 1–30. doi:10.18637/jss.v031.i03.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Escobar MD, West M (1995). "Bayesian Density Estimation and Inference Using Mixtures." *Journal of the American Statistical Association*, **90**(430), 577–588. doi:10.1080/01621459.1995.10476550.

Everitt BS, Landau S, Leese M, Stahl D (2011). *Cluster Analysis*. 5th edition. John Wiley & Sons. doi:10.1002/9780470977811.

Ferguson TS (1973). "A Bayesian Analysis of Some Nonparametric Problems." *The Annals of Statistics*, **1**(2), 209–230. doi:10.1214/aos/1176342360.

Gabry J, Simpson D, Vehtari A, Betancourt M, Gelman A (2019). "Visualization in Bayesian Workflow." *Journal of the Royal Statistical Society A*, **182**, 389–402. doi:10.1111/rssa.12378.

Gelman A, Rubin DB (1992). "Inference From Iterative Simulation Using Multiple Sequences." *Statistical Science*, **7**, 457–472. doi:10.1214/ss/1177011136.

Giguère G (2006). "Collecting and Analyzing Data in Multidimensional Scaling Experiments. A Guide for Psychologists Using SPSS." *Tutorial in Quantitative Methods for Psychology*, **2**, 27–38. doi:10.20982/tqmp.02.1.p026.

Gormley IC, Murphy TB (2010). "A Mixture of Experts Latent Position Cluster Model for Social Network Data." *Statistical Methodology*, **7**, 385–405. doi:10.1016/j.stamet.2010.01.002.

Gower JC, Dijksterhuis GB (2004). *Procrustes Problems*. Oxford University Press, Oxford. doi:10.1093/acprof:oso/9780199574797.003.0001.

Green PJ (1995). "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination." *Biometrika*, **82**, 711–732. doi:10.1093/biomet/82.4.711.

Handcock MS, Raftery AE, Tantrum JM (2007). "Model-Based Clustering for Social Networks." *Journal of the Royal Statistical Society A*, **170**, 301–354. doi:10.1111/j.1467-985x.2007.00471.x.

Hoff PD, Raftery AE, Handcock MS (2002). "Latent Space Approaches To Social Network Analysis." *Journal of the American Statistical Association*, **97**(460), 1090–1098. doi:10.1198/016214502388618906.

Hothorn T, Leisch F, Zeileis A (2020). ***modeltools**: Tools and Classes for Statistical Models*. R package version 0.2-23, URL https://CRAN.R-project.org/package=modeltools.

IBM Corporation (2021). *IBM SPSS Statistics 28*. IBM Corporation, Armonk. URL https://www.ibm.com/analytics/spss-statistics-software.

Krantz DH, Luce RD, Suppes P, Tversky A (1971). *Foundations of Measurement*. Academic Press, New York.

Krivitsky PN, Handcock MS (2008). "Fitting Position Latent Cluster Models for Social Networks with **latentnet**." *Journal of Statistical Software*, **24**(5), 1–23. doi:10.18637/jss.v024.i05.

Krivitsky PN, Handcock MS, Raftery AE, Hoff PD (2009). "Representing Degree Distributions, Clustering, and Homophily in Social Networks with Latent Cluster Random Effects Models." *Social Networks*, **31**, 204–213. doi:10.1016/j.socnet.2009.04.001.

Kruskal JB (1964). "Multidimensional Scaling By Optimizing Goodness of Fit to a Nonmetric Hypothesis." *Psichometrika*, **29**, 1–27. doi:10.1007/bf02289565.

Kyung M, Gill J, Casella G (2010). "Estimation in Dirichlet Random Effects Models." *The Annals of Statistics*, **38**(2), 979–1009. doi:10.1214/09-aos731.

Lewis SM, Raftery AE (1997). "Estimating Bayes Factors Via Posterior Simulation with the Laplace-Metropolis Estimator." *Journal of the American Statistical Association*, **92**, 648–655. doi:10.1080/01621459.1997.10474016.

MacEachern SN, Müller P (1998). "Estimating Mixture of Dirichlet Process Models." *Journal of Computational and Graphical Statistics*, **7**(2), 223–238. doi:10.1080/10618600.1998.10474772.

Mair P (2021). *CRAN Task View: Psychometric Models and Methods*. Version 2021-11-08, URL https://CRAN.R-project.org/view=Psychometrics.

Martin AD, Quinn KM, Park JH (2011). "**MCMCpack**: Markov Chain Monte Carlo in R." *Journal of Statistical Software*, **42**(9), 22. doi:10.18637/jss.v042.i09.

Müller P, Quintana FA, Jara A, Hanson T (2015). *Bayesian Nonparametric Data Analysis*. Springer-Verlag.

Oh MS (2012). "A Simple and Efficient Bayesian Procedure for Selecting Dimensionality in Multidimensional Scaling." *Journal of Multivariate Analysis*, **107**, 200–209. doi:10.1016/j.jmva.2012.01.012.

Oh MS, Raftery AE (2001). "Bayesian Multidimensional Scaling and Choice of Dimension." *Journal of the American Statistical Association*, **96**, 1031–1044. doi:10.1198/016214501753208690.

Oh MS, Raftery AE (2007). "Model-Based Clustering with Dissimilarities: A Bayesian Approach." *Journal of Computational and Graphical Statistics*, **16**, 559–585. doi:10.1198/106186007x236127.

Okada K, Lee MD (2016). "A Bayesian Approach to Modeling Group and Individual Differences in Multidimensional Scaling." *Journal of Mathematical Psychology*, **70**(1), 35–44. doi:10.1016/j.jmp.2015.12.005.

Okada K, Shigemasu K (2009). "BMDS: A Collection of R Functions for Bayesian Multidimensional Scaling." *Applied Psychological Measurement*, **33**(7), 570–571. doi:10.1177/0146621608321761.

Papastamoulis P (2016). "**label.switching**: An R Package for Dealing with the Label Switching Problem in MCMC Outputs." *Journal of Statistical Software*, **69**(1), 1–24. doi:10.18637/jss.v069.c01.

Plummer M (2003). "**JAGS**: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling." In K Hornik, F Leisch, A Zeileis (eds.), *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)*. Technische Universität Wien, Vienna, Austria. URL https://www.R-project.org/conferences/DSC-2003/Proceedings/Plummer.pdf.

Plummer M, Best N, Cowles K, Vines K (2006). "**coda**: Convergence Diagnosis and Output Analysis for MCMC." *R News*, **6**(1), 7–11. URL https://CRAN.R-project.org/doc/Rnews/.

Ramsay JO (1977). "Maximum Likelihood Estimation in Multidimensional Scaling." *Psychometrika*, **42**, 337–360. doi:10.1007/bf02294052.

Ramsay JO (1978). "Confidence Regions for Multidimensional Scaling Analysis." *Psychometrika*, **43**, 145–160. doi:10.1007/bf02293859.

Ramsay JO (1982). "Some Statistical Approaches To Multidimensional Scaling Data." *Journal of the Royal Statistical Society A*, **145**, 285–312. doi:10.2307/2981865.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Richardson S, Green PJ (1997). "On Bayesian Analysis of Mixtures with an Unknown Number of Components." *Journal of the Royal Statistical Society B*, **59**, 731–792. doi:10.1111/1467-9868.00095.

Robert CP (1996). "Mixtures of Distributions: Inference and Estimation." In WR Gilks, S Richardson, DJ Spiegelhalter (eds.), *Markov Chain Monte Carlo in Practice*, pp. 441–464. Chapman & Hall/CRC, New York.

Rousseeuw PJ (1984). "Least Median of Squares Regression." *Journal of the American Statistical Association*, **79**, 871–880. doi:10.1080/01621459.1984.10477105.

Rousseeuw PJ, Van Driessen K (1999). "A Fast Algorithm for the Minimum Covariance Determinant Estimator." *Technometrics*, **41**, 212–223. doi:10.1080/00401706.1999.10485670.

Roy V (2020). "Convergence Diagnostics for Markov Chain Monte Carlo." *Annual Review of Statistics and Its Application*, **7**, 387–412. doi:10.1146/annurev-statistics-031219-041300.

SAS Institute Inc (2013). *The SAS System, Version 9.4*. SAS Institute Inc., Cary. URL https://www.sas.com/.

Shepard RN (1962). "The Analysis of Proximities: Multidimensional Scaling with an Unknown Distance Function. II." *Psychometrika*, **27**, 219–246. doi:10.1007/bf02289621.

StataCorp (2021). *Stata Statistical Software: Release 17*. StataCorp LLC, College Station. URL https://www.stata.com/.

Stephens M (2000). "Bayesian Analysis of Mixture Models with an Unknown Number of Components." *The Annals of Statistics*, **28**, 40–74. doi:10.1214/aos/1016120364.

Takane Y (1978a). "A Maximum Likelihood Method for Nonmetric Multidimensional Scaling: I. The Case in Which All Empirical Pairwise Orderings Are Independent – Evaluations." *Japanese Psychological Research*, **20**, 105–114. doi:10.4992/psycholres1954.20.105.

Takane Y (1978b). "A Maximum Likelihood Method for Nonmetric Multidimensional Scaling: I. The Case in Which All Empirical Pairwise Orderings Are Independent – Theory." *Japanese Psychological Research*, **20**, 7–17. doi:10.4992/psycholres1954.20.7.

Takane Y (1981). "Multidimensional Successive Categories Scaling: A Maximum Likelihood Method." *Psychometrika*, **46**, 9–28. doi:10.1007/bf02293914.

Takane Y (2007). "Application of Multidimensional Scaling in Psychometrics." In CR Rao, S Sinharay (eds.), *Handbook of Statistics 26: Psychometrics*, pp. 359–400. Chapman & Hall/CRC.

Takane Y, Carroll JD (1981). "Nonmetric Maximum Likelihood Multidimensional Scaling From Directional Rankings of Similarities." *Psychometrika*, **46**, 389–405. doi:10.1007/bf02293797.

Takane Y, Jung S, Takane YO (2009). "Multidimensional Scaling." In RE Millsap, A Maydeu-Olivares (eds.), *The Sage Handbook of Quantitative Methods in Psychology*, chapter 10, pp. 219–242,. Sage Publications.

Takane Y, Sergent J (1983). "Multidimensional Scaling Models for Reaction Times and Same-Different Judgments." *Psychometrika*, **48**, 393–423. doi:10.1007/bf02293683.

Takane Y, Shibayama T (1986). "Comparison of Models for the Stimulus Recognition Data." In J De Leeuw, WJ Heiser, J Meulman, F Critchley (eds.), *Multidimensional Data Analysis*, pp. 359–400. DSWO Press.

Takane Y, Shibayama T (1992). "Structures in Stimulus Identification Data." In FG Ashby (ed.), *Probabilistic Multidimensional Models of Perception and Cognition*, pp. 335–362. Earlbaum.

Takane Y, Young FW, De Leeuw J (1977). "Nonmetric Individual Differences Multidimensional Scaling: An Alternating Least Squares Method with Optimal Scaling Features." *Psychometrika*, **42**, 7–67. doi:10.1007/bf02293745.

Torgerson WS (1952). "Multidimensional Scaling: I. Theory and Method." *Psychometrika*, **17**. doi:10.1007/bf02288916.

Torgerson WS (1958). *Theory and Methods of Scaling*. John Wiley & Sons.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. 2nd edition. Springer-Verlag.

Young FW (1987). *Multidimensional Scaling: History, Theory, and Applications*. Lawrence Erlbaum Associates.

Affiliation:

Sergio Venturini
Dipartimento di Scienze Economiche e Sociali
Università Cattolica del Sacro Cuore
Via Bissolati 74, 26100 Cremona, Italy
E-mail: sergio.venturini@unicatt.it

Raffaella Piccarreta
Dipartimento di Scienze delle Decisioni
Università Commerciale L. Bocconi
Via Röntgen 1, 20136 Milano, Italy
E-mail: raffaella.piccarreta@unibocconi.it

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
November 2021, Volume 100, Issue 16 *Submitted:* 2019-07-25
doi:10.18637/jss.v100.i16 *Accepted:* 2021-03-01