UiO : University of Oslo

Qinghua Liu

Bayesian Preference Learning with the Mallows Model

Thesis submitted for the degree of Philosophiae Doctor

Department of Mathematics

Faculty of Mathematics and Natural Sciences

2021

Series of dissertations submitted to the

Faculty of Mathematics and Natural Sciences, University of Oslo

No. 2452

ISSN 1501-7710

Cover: Hanne Baadsgaard Utigard.

Print production: Reprosentralen, University of Oslo.

Acknowledgements

The past four years or so as a PhD student has been the most intellectually challenging, life-changing and fulfilling experience in my life so far. Looking back at this journey, from the day I took my first step into this "kingdom far far away" (that is Norway) with less than abundance of knowledge in the field of statistics, to where I am now - right upon the completion of this thesis, I am feeling accomplished, and grateful for the many people that I have the pleasure of working with and learning from.

First and foremost, I would like to express my deepest gratitude towards my supervisors Ida Scheel and Arnoldo Frigessi. I have learned and benefited so much from their rich knowledge and research experiences. This PhD would not be possible without their guidance and advice. Their support for me goes well beyond just academics, and I am immensely thankful that I have received so much encouragement and kindness from them during the challenging times.

My sincere gratitude also goes to my co-authors: Valeria Vitelli, Andrew Henry Reiner, Marta Crispino, Carlo Manino and Øystein Sørensen. I have truly enjoyed our fruitful collaborations and we should be proud of what we have achieved together! I would also like to thank Elja Arjas for all the interesting discussions.

To my colleagues at the math department: I have had many fond memories that we have shared within and beyond the university buildings. My special thanks goes to Yinzhi Wang - I will not forget the fun we had during lunch times, our spontaneous day trip and so much more! I would also like to thank the Data Science @ UiO innovation cluster and BigInsight for generously supporting my PhD journey.

Last but not least, I am forever grateful to my family: my parents, brother and my loving husband, for always supporting my pursues, and sending me their love unconditionally.

Qinghua Liu
Oslo, October 2021

List of Papers

Paper I

Qinghua Liu, Marta Crispino, Ida Scheel, Valeria Vitelli, and Arnoldo Frigessi (2019). "Model-based Learning from Preference data". *Annual review of statistics and its application.* 6, s 329- 354. DOI: 10.1146/annurev-statistics-031017-100213.

Paper II

Qinghua Liu, Andrew Henry Reiner, Arnoldo Frigessi and Ida Scheel (2019). "Diverse personalized recommendations with uncertainty from implicit preference data with the Bayesian Mallows Model". *Knowledge-Based Systems.* 186, s 1-12. DOI: 10.1016/j.knosys.2019.104960

Paper III

Qinghua Liu, Valeria Vitelli, Arnoldo Frigessi and Ida Scheel (2021). "Pseudo-Mallows for Efficient Preference Learning". *Manuscript.*

Paper IV

Øystein Sørensen, Marta Crispino, Qinghua Liu and Valeria Vitelli (2020). "BayesMallows: an R Package for the Bayesian Mallows Model". *The R Journal.* 12(1), s 324- 342 DOI: 10.32614/RJ-2020-026

Contents

Chapter 1

Introduction

In our modern world where the choices of goods and services seem endless, every day we are faced with this question: how do we pick out the best products for ourselves? Luckily, with the help of *recommender systems*, we can be guided to make better choices. Based on information about the customer's past interactions with the products, recommender systems learn the personal preferences of each customer, and then produce a list of recommendation for each of them, ranked according to the probabilities that the customer will favor them.

Naturally, the personalized recommendation problem is a ranking problem. First of all, the ultimate goal of a recommender system is to produce a ranking; second, many types of user-product interaction data exist in the form of ranking, or can be easily adapted to rankings. Common commercial recommender systems often adopt collaborative filtering (CF) [37] to produce personalized recommendations. CF, which relies on the matrix factorization technique, has gained great popularity due to its high recommendation accuracy. However, using intuitive ranking models to learn individual preferences and make personalized recommendation remains relative unexplored. In this thesis, we will embark on a journey to explore many aspects of ranking models. We focus mainly on the Bayesian Mallows model, and how it can be applied and improved in the recommender system context.

The first theme of this thesis is preference learning models. Learning personal preference is an important first step to achieve personalized recommendations. Most models that we examine in this thesis are probabilistic models, such as the Mallows ranking model [43], the Plackett-Luce (PL) model [41][52] and the Bradley-Terry (BT) model [11]. We also examine CF although it is not probabilistic, due to its importance in the field of personalized recommendations. The Mallows ranking model uses a permutation of all n item for each of the user, and defines its probabilistic density on the space of permutation of n items; the PL model on the other hand, utilizes a continuous n-dimensional vector. The BT model approaches the problem using paired comparisons, while CF considers a reduced-dimension vector for each user and each item. These different constructions certainly lead to different characteristics, and we will examine these characteristics in detail.

The second focus of this work is learning from clicking data. Clicking data exists in great abundance. Unlike ratings and other explicit user feedback, clicking data is especially valuable as it arises naturally while a user interacts with the products without the need of any solicitation. Consequently, it is relatively

little affected by selection bias. We therefore focus on developing methods to learn personalized preferences and make recommendations from clicking data. Although many forms of preference data exist, such as full rankings, partial rankings, ratings, paired-comparisons, they can all be easily adapted to be clicking data-like, which has a binary form.

The last theme of this thesis is Bayesian methods and computation. The Bayesian framework makes inference from a posterior distribution, which naturally quantifies uncertainties in model parameters by means of probability. Such probabilistic quantification is easy to interpret, and provides valuable information to our analysis. Many Bayesian models, including our Bayesian Mallows Model, are complex and difficult to make inference from, and Markov Chain Monte Carlo (MCMC) is commonly used to obtain samples of the target posterior distribution. On the flip side, MCMC methods often suffer from slow convergence and are hard to scale to larger datasets. Many methods are developed to improve MCMC, and variational inference is a recent framework that has proven to be powerful and scalable. We will investigate both the MCMC option and variational inference to make inference on our Bayesian model.

Overall, in this work, I aim at exploring the theme of personalized recommendation holistically - from modelling to algorithm implementation. My wish is that the readers can be introduced to the world of recommender systems and learn a new way of achieving personalized recommendations.

The rest of this thesis is organized as follows. Chapter 2 contains a brief introduction to recommender systems and the different forms of preference data. In Chapter 3 we study the basics of two important and relevant preference learning methods: CF and the Bayesian Mallows Model, while Chapter 4 contains the popular techniques to learn from Bayesian models. High level summaries of the publications are provided in Chapter 5, and finally, Chapter 6 contains some reflections and final thoughts.

Chapter 2

A primer on recommender systems and preference data

2.1 Recommender systems

Recommender systems help users discover the items that are most relevant to them. It is crucial for online services such as e-commerce websites and streaming platforms where the product catalogue is huge, however, only a small subset of the collection can be displayed to the user due to space constraint. User engagement can be significantly enhanced when the content is personalized for them such that they do not need to go through the overwhelming procedure of searching for the relevant items.

The first major task for a recommender systems is to predict users' preferences on previously unseen items. *Content-based* recommender systems assumes that users have similar preferences on similar items. Therefore, content-based recommender systems try to construct item profiles utilizing information such as descriptions and category information, and then establish user profiles based on the items that the user has interacted with before. Items that best match the user profile will therefore be recommended. More details about content-based recommender systems can be found in [39, 50]. Content-based recommender systems are especially useful when a new user or a new item is introduced to the system, where there is no sufficient historical data about the item or the user available.

As a contrast, *Collaborative filtering* recommender systems make personalized recommendations using user-item interaction data. The word "collaborative" refers to the fact that recommendations are made for a user based on other user's preference data. Collaborative filtering methods can be user-based, which first discover similar users (neighbor users), and make recommendation based on what the neighbor-users has favored. Item-based collaborative filtering on the other hand, is a transpose of user-based collaborative filtering, where a prediction on an item is made based on similar items [56]. One of the most important types of recommender systems is the latent factor-based recommender systems [37], which will be introduced in more details in Chapter 3.

After the recommender system has learned each user's preferences on the items, a list of top-k items that best matches the user's preferences is produced. The recommendations' success should be assessed in terms of both accuracy and diversity [23, 27]. Achieving a good balance between recommendation accuracy and diversity has proven to be challenging, and this issue will be addressed in

Paper II.

2.2 Preference data

In order to learn user preferences, especially through collaborative filtering approaches, preference data is an important resource. Users' preference data is ubiquitous. Imagine a scenario where a user visits an online streaming service platform. After browsing the movie catalogue, the user clicks on a movie, watches it, and then gives it a big thumbs up. In this scenario, (at least) two types of preference data has occurred. The user's click on an item *implicit*ly indicates that she has a likelihood to favor the movie, and then the big thumbs up *explicit*ly expresses her fondness of the movie. From this example, based on the purpose of the data, we can divide preference data into two categories: explicit and implicit.

Explicit data refers to the situation where the data is given in order for a user to express her preferences of the items. Some examples of explicit feedback data are ratings, pairwise comparisons and rankings. Rankings specifically, can also be further categorized as full rankings - where a user has given her ranking of all the items that are available, and partial rankings, where only a subset of items are ranked.

Implicit data on the other hand, can exist in many different types of observable and measurable parameters that arise when the user interacts with the system [33]. To some extent, they carry preference information, however, implicit feedback's original purpose is not to indicate the user's preference. Some examples of implicit data include clicking, browsing history and search history.

One of the biggest advantages of explicit data is the amount of information it carries. A rating for example, not only clearly indicates a user's attitude (positive or negative) towards an item, it also expresses the extent of the preference. Implicit feedback on the other hand, is much more ambiguous. A clicking action can surely indicate a positive preference, however, it can happen by accident too. Another big challenge of implicit data is the lack of direct negative feedback - a non-click can be due to a dislike, or that the user simply did not discover the item. There are strategies available to help discover negative implicit feedback [51], however, the interpretation of implicit data remains a challenge.

Nevertheless, recent research and applications that utilize preference data have gradually shifted away from focusing (solely) on explicit data. One of the obvious reasons is the sheer volume and availability of implicit data, while explicit data requires solicitation. Besides, explicit data also exhibits many types of biases. First, different users have very different scales - some rate a 5/5 for almost everything, while "stricter" users might rate 5/5 only to indicate exceptional excellence. Using such information without normalization will inevitably lead to inaccuracy. Another problematic issue about explicit data is that a rating often indicates the "quality" of the item perceived by the user, which does not

necessarily represent the user's "enjoyment" of the item, or the item's relevance to the user. [32] contains an in depth comparison of explicit vs. implicit feedback.

Despite the differences however, many different forms of preference data can be adapted to each other under some assumptions. For example, ratings can be easily adapted to rankings and/or paired comparisons with the simple assumption that a higher rated item is more preferred, and should therefore be ranked higher. Clicking data can also be converted to paired comparisons and rankings that are consistent with the clicking behaviour if we assume that the items clicked by the user are the user's favorite items. These assumptions are used in all the papers in this thesis, where we learn from different forms of preference data with the Mallows model, which originally is defined based on rankings.

Chapter 3

Two important preference learning methods

3.1 Collaborative filtering (CF)

On a broad context, CF refers to the process of evaluating and selecting items based on other people's opinion [57]. However, due to the popularization of the Netflix Prize [7], the use of the term CF is almost synonymous with matrix factorization-based methods. In this thesis, we use the term CF to exclusively refer to the matrix factorization-based methods.

In this section, we begin by introducing the CF method for explicit user feedback [37], an award-winning method developed for rating prediction, which to some extent, laid the foundation for recommender systems research and development. As methods based on explicit ratings start to show limitations due to the difficulty of soliciting feedback information and selection bias, many recommender systems shift to methods that are based on implicit feedback, such as clicking data. We focus on the CF for implicit feedback method developed by Hu, Koren, and Volinsky [30], which has been, til this day, widely adopted in commercial applications.

3.1.1 CF for explicit user feedback

Given there are N users and n items. User j's rating of item i is denoted as x_{ji}. x_{ji} is usually non-negative and continuous, for example, a 1-5 scale. x_{ji} is set to 0 or NA if user j has not rated item i. Hence, the user-item rating matrix can be represented by a sparse rating matrix $\mathbb{X}^{N\times n}$. We denote the set of all user-item (j, i) pairs that are available as a set κ, and the aim of this model is to predict the ratings for all user-item pairs that are currently not available, i.e., the user-item pairs that are not in κ.

The matrix-factorization method proposed by Koren, Bell, and Volinsky[37] assumes that each user j is associated with a f-dimensional real-valued latent vector $\boldsymbol{p}_j \in \mathbb{R}^f$, whereas each item i can also be represented with a f-dimensional latent vector $\boldsymbol{q}_i \in \mathbb{R}^f$. The dot product $\hat{x}_{ji} = \boldsymbol{p}_j^T \boldsymbol{q}_i$ represents a score assigned to the user-item pair, and intuitively, $\hat{x}_{ji}$ should approximate the rating x_{ji} if it is available. If the latent vectors for all items and all users are learned, we can populate the user-item pairs that are currently not available using $\hat{x}_{ji} = \boldsymbol{q}_i^T \boldsymbol{p}_j$ $\forall (j, i) \notin \kappa$ to estimate user j's preference on item i. Thus, the main task of this model is translated to obtaining the latent vectors $\boldsymbol{p}_j$ and $\boldsymbol{q}_i$, for $j = 1, ..., N$

and $i = 1, ..., n$, and it can be achieved through an optimization process:

$$\min_{\boldsymbol{p},\boldsymbol{q}} \sum_{(j,i)\in\kappa} (x_{ji} - \boldsymbol{p}_j^T \boldsymbol{q}_i)^2 + \lambda(||\boldsymbol{p}_j||^2 + \boldsymbol{q}_i||^2) \tag{3.1}$$

The first term in Equation (3.1) is the squared error between the approximated rating $\hat{x}_{ji}$ and the rating x_{ji} given by the user. The second term is an L2 penalization term to avoid the model from being over-fitted to the given data. The most popular algorithms for solving this function are the Stochastic Gradient Descent (SGD) algorithm [22] and the Alternating Least Square (ALS) algorithm. ALS later becomes the de facto method due to its potential to be parallelized, and its flexibility to handle implicit feedback, which will also be discuss in a following section.

3.1.2 The alternating least squares (ALS) algorithm

ALS [1, 66] is an iterative optimization algorithm. While solving for one of the unknowns, the algorithm keeps the other ones fixed. If we denote the $N \times f$ user sub-matrix as $\boldsymbol{P}$ and the $n \times f$ item sub-matrix as $\boldsymbol{Q}$, the algorithm can be summarized as the following steps:

1. Initialize matrix $\boldsymbol{Q}$. Zhou et al.[66] suggests setting the first row of each item with the average rating of that item, and randomly initialize the rest of the item matrix.
2. While keeping $\boldsymbol{Q}$ fixed, solve for $\boldsymbol{P}$ by minimizing the least square problem
3. While keeping $\boldsymbol{P}$ fixed, solve for $\boldsymbol{Q}$ by minimizing the least square problem
4. Repeat step 2 and 3 until a convergence criterion, such as an RMSE threshold, is met

We denote the loss function (3.1) as $\mathcal{L}$, and the minimization process at step 2 and 3 can be derived as follows:

At each iteration, the updated $\boldsymbol{p}^j$ is obtained by:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{p}_j} &= -2\sum_i (x_{ji} - \boldsymbol{p}_j^T \boldsymbol{q}_i)\boldsymbol{q}_i^T + 2\lambda \boldsymbol{p}_j^T \\
0 &= -\sum_i (x_{ji} - \boldsymbol{p}_j^T \boldsymbol{q}_i)\boldsymbol{q}_i^T + \lambda \boldsymbol{p}_j^T \\
0 &= -(\boldsymbol{x}_j - \boldsymbol{p}_j^T \boldsymbol{Q}^T)\boldsymbol{Q} + \lambda \boldsymbol{p}_j^T \\
\boldsymbol{x}_j \boldsymbol{Q} &= \boldsymbol{p}_j^T (\boldsymbol{Q}^T \boldsymbol{Q} + \lambda \boldsymbol{I}) \\
\boldsymbol{p}_j^T &= \boldsymbol{x}_j \boldsymbol{Q} (\boldsymbol{Q}^T \boldsymbol{Q} + \lambda \boldsymbol{I})^{-1}
\end{aligned} \tag{3.2}$$

Similarly, for $\boldsymbol{q}_i$:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}_i} &= -2\sum_j (x_{ji} - \boldsymbol{p}_j^T \boldsymbol{q}_i)\boldsymbol{p}_j^T + 2\lambda \boldsymbol{q}_i^T \\
0 &= -\sum_j (x_{ji} - \boldsymbol{q}_i^T \boldsymbol{p}_j)\boldsymbol{p}_j^T + \lambda \boldsymbol{q}_i^T \\
0 &= -(\boldsymbol{x}_i - \boldsymbol{q}_i^T \boldsymbol{P}^T)\boldsymbol{P} + \lambda \boldsymbol{Q}_i^T \\
\boldsymbol{x}_i \boldsymbol{P} &= \boldsymbol{q}_i^T (\boldsymbol{P}^T \boldsymbol{P} + \lambda \boldsymbol{I}) \\
\boldsymbol{q}_i^T &= \boldsymbol{x}_i \boldsymbol{P} (\boldsymbol{P}^T \boldsymbol{P} + \lambda \boldsymbol{I})^{-1}
\end{aligned} \tag{3.3}$$

However, it is noteworthy to mention that apart from solving for the sub - matrices, namely, $\boldsymbol{P}$ and $\boldsymbol{Q}$, there are other hyper-parameters involved in this CF model, such as the dimension of the latent matrices f and the penalization parameter λ. These parameters are usually chosen using k-fold cross validation.

3.1.3 CF for implicit feedback

Hu, Koren, and Volinsky[30] extended [37] to be applied to implicit user feedback, such as clicking data.

We continue with our notations from Section 3.1.1, and denote user-item interactions as x_{ji}. However, x_{ji} in this case is not a rating, but a kind of utility score based on the implicit feedback. For example, x_{ji} can be the number of times user j has clicked on item i. A set of binary interaction variable is then introduced: if a user j has interacted with item i, we set their interaction $b_{ji} = 1$, and 0 otherwise. In other words, we define

$$b_{ji} = \begin{cases} 1, & \text{if } x_{ji} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Similar to the CF method in the previous section, we assume that each user j can be represented by a f-dimensional vector $\boldsymbol{p}_j \in \mathbb{R}^f$, and each item a f-dimensional vector $\boldsymbol{q}_i \in \mathbb{R}^f$. Their dot product $\hat{b}_{ji} = \boldsymbol{p}_j^T \boldsymbol{q}_i$ should be an approximation of b_{ji}.

However, not all user-item interactions are equal. For example, if a user clicks on one item multiple times, it can indicate a stronger preference compared to an item that is clicked on only once. Another set of "confidence" variable c_{ji} is hence introduced. One possible construction is:

$$c_{ji} = 1 + \theta x_{ji}, \theta > 0$$

The objective function for the minimization is as follows:

$$\min_{\boldsymbol{p},\boldsymbol{q}} \sum_{(j,i)} c_{ji}(b_{ji} - \boldsymbol{p}_j^T \boldsymbol{q}_i)^2 + \lambda(\sum_j ||\boldsymbol{p}_j||^2 + \sum_i \boldsymbol{q}_i||^2). \tag{3.4}$$

$\boldsymbol{p}_j$ and $\boldsymbol{q}_i$ are once again obtained using ALS. More specifically, at each iteration, we update

$$\begin{aligned} \boldsymbol{p}_j &\leftarrow (\boldsymbol{Q}^T \boldsymbol{c}_j \boldsymbol{Q} + \lambda \boldsymbol{I})^{-1} \boldsymbol{Q}^T \boldsymbol{c}_j \boldsymbol{b}_j \\ \boldsymbol{q}_i &\leftarrow (\boldsymbol{P}^T \boldsymbol{c}_i \boldsymbol{P} + \lambda \boldsymbol{I})^{-1} \boldsymbol{P}^T \boldsymbol{c}_i \boldsymbol{b}_i, \end{aligned} \tag{3.5}$$

and the hyper-parameters λ, θ and f are tuned using cross validation.

The influence of MF and ALS is profound as it has inspired many newer recommender methods. For example, context-aware recommender systems [35, 53, 54], which consider not only user-item interactions, but also a context such as time or location, often utilizes tensor factorization techniques [18, 61], which is a generalization of MF. Another recent trend in the development of recommender systems is the incorporation of deep learning into CF [38, 64]. Deep learning techniques can learn the latent representations of both the users and the items from auxiliary information, enriching the information available for CF beyond just user-item interactions.

3.2 The Mallows Ranking method

Many applications of preference learning, such as recommender systems and search engines' ultimate goal is to produce a ranked list of items. In fact, many forms of preference data exist naturally in the form of ranking or can be conveniently converted, it is therefore intuitive to model user preferences directly as rankings. Ranking models often assume a set of utility scores for the items, these scores can be continuous, such as the Plackett-Luce model, or discrete values, such as the Mallows ranking model. We focus on the Mallows ranking model in this thesis, but more details on various model-based preference learning methods are discussed in Paper I.

3.2.1 Mallows ranking model for complete rankings

Given a set of n items $\mathcal{A} = \{A_1, ..., A_n\}$. A user expresses her preferences of these n items by assigning a ranking to these n items, denoted as $\boldsymbol{r} = \{r_1, ..., r_n\} \in \mathcal{P}_n$. $r_i = 1$ indicates that item i is the user's favorite item, $r_k = n$ indicates that item k is the user's least favorite, and $\mathcal{P}_n$ denotes the space of permutation of n items.

The Mallows model [43] is a probabilistic distribution of rankings, defined directly on $\mathcal{P}_n$. Given a ranking $\boldsymbol{r} \in \mathcal{P}_n$, the Mallows distribution has the form

$$P(\boldsymbol{r}|\alpha, \boldsymbol{\rho}) = Z_n(\alpha, \boldsymbol{\rho})^{-1} \exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \boldsymbol{\rho})\}, \tag{3.6}$$

where $\boldsymbol{\rho} \in \mathcal{P}_n$ is the consensus parameter, $\alpha > 0$ is the scale parameter, $d(\boldsymbol{r}, \boldsymbol{\rho})$ is a distance between the ranking and the consensus, and $Z_n(\alpha, \boldsymbol{\rho}) = \sum_{\boldsymbol{\rho} \in \mathcal{P}_n} \exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \boldsymbol{\rho})\}$ is the normalizing constant.

To understand the model, we can analogize the Mallows distribution as a Gaussian equivalent in the space of permutation [14]. The consensus parameter $\boldsymbol{\rho}$ is similar to the mean where the permutations are centered on, while the scale parameter α, or more precisely, $\frac{1}{\alpha}$ acts in a similar way as the variance, which represents the dispersion of the rankings from the consensus.

The choice of the distance function $d()$ has an impact on the computation of the normalizing constant. Usually, a right-invariant distance [20], meaning that the distance is unaffected by the re-labelling of items, is preferred, since this makes the normalizing constant independent of the consensus parameter $\boldsymbol{\rho}$, and the normalizing constant can hence be rewritten as $Z_n(\alpha) = \sum_{\boldsymbol{r} \in \mathcal{P}_n} \exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \mathbf{1}_n)\}$, where $\mathbf{1}_n = \{1, ..., n\}$. Some possible right-invariant distances include the Kendall distance, the Hamming distance, the Cayley distance, the footrule distance (L1) and the Spearman's distance (L2) [44]. Fligner and Verducci[21] provides the close form of $Z_n(\alpha)$ for the Kendall's distance, and many literature concerning the Mallows method have employed the Kendall distance [34, 40, 45]. The footrule and Spearman's distances are important and natural right-invariant distance choices, but the normalizing constant can only be computed analytically for up to 50 and 14 items, respectively. To enable the use of the footrule and Spearman's distance in the Mallows model for larger n's, Vitelli et al.[62] introduced an important sampling scheme, which can approximates $Z_n(\alpha)$ with the footrule or Spearman's distance for up to a few hundred items within a reasonable time frame. To compute for very large n, Mukherjee et al.[47] introduced an iterative algorithm which gives an asymptotic estimation of $Z_n(\alpha)$ for $n \to \infty$.

Given N users' full rankings $\boldsymbol{R}^1, ..., \boldsymbol{R}^N$, assuming all users are conditionally independent given the Mallows parameters α and $\boldsymbol{\rho}$, the likelihood is:

$$P(\boldsymbol{R}^1, ..., \boldsymbol{R}^N | \alpha, \boldsymbol{\rho}) = Z_n(\alpha)^{-N} \exp\{-\frac{\alpha}{n} \sum_{j=1}^{N} d(\boldsymbol{R}^j, \boldsymbol{\rho})\}, \tag{3.7}$$

and the maximum likelihood estimator for $\boldsymbol{\rho}$ for a given α can be obtained by computing

$$\arg\min_{\boldsymbol{\rho} \in \mathcal{P}_n} \sum_{j=1}^{N} d(\boldsymbol{R}^j, \boldsymbol{\rho}). \tag{3.8}$$

From Equation (3.8) it can be observed that this computation is not tractable when n is large, as the minimization is over the space of permutation.

3.2.2 Bayesian Mallows model(BMM)

Vitelli et al.[62] introduced a Bayesian version of the Mallows model. A uniform prior distribution is chosen for the consensus parameter $\boldsymbol{\rho}$, i.e., $\pi(\boldsymbol{\rho}) = \frac{1}{n!}$, and an exponential prior distribution for α: $\pi(\alpha|\lambda) = \lambda e^{-\lambda\alpha}$. λ is a hyperparameter, which typically has little influence on the inference of $\boldsymbol{\rho}$ in practice, and can be fixed to a value close to 0. With the chosen prior distributions, the posterior distribution can be written as follows:

$$P(\boldsymbol{\rho}, \alpha|\boldsymbol{R}^1, ..., \boldsymbol{R}^N) \propto \pi(\alpha)\pi(\boldsymbol{\rho})Z_n(\alpha)^{-N} \exp\{-\frac{\alpha}{n}\sum_{j=1}^{N} d(\boldsymbol{R}^j, \boldsymbol{\rho})\}. \tag{3.9}$$

We can thus make inference on this posterior distribution through a Markov Chain Monte Carlo (MCMC). More details regarding MCMC will be covered in Section 4.1.

3.2.3 Bayesian Mallows model's extension to partial data

When full ranking data is not readily available, the BMM can be extended to learn both the group consensus and individual rankings from partial data by means of data augmentation.

We denote the partial data given by user j by $\boldsymbol{D}^j$. This partial data can include partial rankings, pairwise comparisons or clicking data. First, we assume that there is a set of rankings $\mathcal{S}^j = \{\tilde{\boldsymbol{R}}^j\}$ that is compatible with the partial data. For example, if $\boldsymbol{D}^j$ is a top-k ranking, a compatible ranking $\tilde{\boldsymbol{R}}^j$ is one that contains all the top-k rankings, but with the missing rankings filled in with values larger than k. If $\boldsymbol{D}^j$ is in the form of paired comparisons, a compatible $\tilde{\boldsymbol{R}}^j$ is a ranking that satisfies all direct and induced paired comparisons contained in $\boldsymbol{D}^j$.

The target posterior distribution is hence

$$P(\boldsymbol{\rho}, \alpha|\boldsymbol{D}^1, ..., \boldsymbol{D}^N) = \sum_{\tilde{\boldsymbol{R}}^1 \in \mathcal{S}^1} ... \sum_{\tilde{\boldsymbol{R}}^N \in \mathcal{S}^N} P(\boldsymbol{\rho}, \alpha, \tilde{\boldsymbol{R}}^1, ..., \tilde{\boldsymbol{R}}^N|\boldsymbol{D}^1, ..., \boldsymbol{D}^N). \tag{3.10}$$

3.2.4 Bayesian Mallows model's extensions to multiple groups

Given large number of users, it can be expected that there exists multiple groups of users, and within each group, the users' rankings follow a Mallows distribution. Assume there exists C clusters, each cluster c has its own consensus parameter $\boldsymbol{\rho}_c$ and scale parameter α_c. We denote the cluster assignment of each user as $z^j \in \{1, ..., C\}$ and assume the clusters are mutually independent. With the observed full ranking data $\{\boldsymbol{R}^1, ..., \boldsymbol{R}^N\}$, the likelihood function can be expressed as

$$P(\boldsymbol{R}^1, ..., \boldsymbol{R}^N|\alpha, \boldsymbol{\rho}, z^1, ..., z^N) = \prod_{j=1}^{N} Z_n(\alpha_{z^j})^{-1} \exp\{-\frac{\alpha_{z^j}}{n} d(\boldsymbol{R}^j, \boldsymbol{\rho}_{z^j})\}, \tag{3.11}$$

The prior distributions for $\{\alpha_c, \boldsymbol{\rho}_c\}$ for each cluster c is the same as described in Section 3.2.2. The prior distribution for the cluster assignment variables $\{z^1, ..., z^N\}$ is chosen as follows:

$$\begin{aligned} P(z^1, ..., z^N | \tau^1, ..., \tau^C) &= \prod_{j=1}^{N} \tau_{z^j} \\ \pi(\tau_1, ..., \tau_C) &= \Gamma(\psi C)\Gamma(\psi)^{-C} \prod_{c=1}^{C} \tau_c^{\psi-1}, \end{aligned} \tag{3.12}$$

where τ_c indicates the probability for a user to belong to cluster c, intuitively, $\sum_{c=1}^{C} \tau_c = 1$.

This construction to handle multiple clusters is also applicable to the partial data case with the augmentation scheme embedded in the inference process.

There are also many extensions of the BMM. For example, Asfaw et al. [4] considers the time-varying effect of preferences, which enables the BMM to predict future rankings; Crispino et al.[17] extends the paired-comparison construction of the BMM to enable it to handle the situation where users provide conflicting information. Overall, the Bayesian Mallows method is flexible to handle preference data that exists in many different forms by augmenting them into compatible rankings, and it can subsequently perform rank aggregation as well as individual preference learning with uncertainty. Conveniently, the Bayesian Mallows method is publicly available through the "BayesMallows" R package [59], and can be easily used.

Chapter 4

Bayesian computation and approximation methods

4.1 Markov Chain Monte Carlo (MCMC)

Bayesian modelling, especially hierarchical models [15, 24], often results in complex posterior distributions with intractable integrals. MCMC methods have enabled making inference from Bayesian models of virtually unlimited complexity [12]. The essence of MCMC is to construct a Markov process [31], with the target posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{x})$ as its stationary distribution. As the simulation is run sufficiently long, the samples drawn should approximate the target stationary distribution [24].

4.1.1 Metropolis-Hasting algorithm

The Metropolis-Hasting(M-H) algorithm [26] is closely related to acceptance-rejection sampling [13]. At each iteration t, the M-H algorithm allows a new sample of the parameter θ^* to be drawn from an arbitrary proposal distribution that is convenient to sample from, and then the proposed sample θ^* is either accepted or rejected according to an acceptance probability. In more details, the algorithm can be summarized in the following steps:

1. Initialize $\theta = \theta^0$ s.t. $p(\theta^0|\boldsymbol{x}) > 0$

2. For iteration t= 1, ..., T
   (a) Draw a new sample θ^* from the proposal distribution $J(\theta^*|\theta^{t-1})$
   (b) Compute the acceptance probability
   $$r = \frac{P(\theta^*|\boldsymbol{x})/J(\theta^*|\theta^{t-1})}{P(\theta^{t-1}|\boldsymbol{x})/J(\theta^{t-1}|\theta^*)} \tag{4.1}$$
   (c) Set $\theta^t = \theta^*$ with probability $\min(1, r)$, otherwise, keep $\theta^t = \theta^{t-1}$

Clearly, the choice of proposal distribution $J(\theta^*|\theta^{t-1})$ has an impact on the M-H algorithm. First, if the proposal distribution is symmetric, i.e., $J(\theta^*|\theta^{t-1}) = J(\theta^{t-1}|\theta^*)$, the acceptance probability in Equation (4.1) can be reduced to
$$r = \frac{P(\theta^*|\boldsymbol{x})}{P(\theta^{t-1}|\boldsymbol{x})}, \tag{4.2}$$

resulting in less computation at each iteration. In fact, when the proposal distribution is symmetric, the algorithm is referred to as the Metropolis algorithm [46].

Second, the behaviour of the proposal distribution can affect the efficiency of the algorithm - if the "jumps" between the samples are too small, the samples obtained at consecutive iterations will be highly correlated to each other, and it can take a very long chain before sufficient number of effectively independent samples can be obtained. On the flip side, if the jumps are too far apart, it can potentially lead to too many samples being rejected, which in turn, also leads to inefficiency. In practice, the acceptance rate, i.e., the percentage of proposed samples that are accepted (not to be confused with the acceptance probability) is also tracked to help tuning the proposal distribution. Gelman et al.[24] suggest that when many parameters are updated, an optimal acceptance rate is approximately 0.23.

4.1.2 A special case of M-H: Gibbs sampler

Gibbs sampler, or conditional sampling, is a special case of Metropolis-Hasting algorithm where the acceptance probability is always 1. Suppose $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$ is the parameter vector, for which we want to obtain samples of the posterior distribution. At each iteration t, Gibbs sampler draws a sample of each parameter $\theta_i, i = 1, ..., d$ conditioning on the current values of all other parameters. That is to say, at each iteration t, for each parameter θ_i, a sample is drawn from the full conditional distribution

$$p(\theta_i^t | \boldsymbol{\theta}_{-i}^{t-1}, \boldsymbol{x}), \tag{4.3}$$

where $\boldsymbol{\theta}_{-i}^{t-1}$ represents the current values all other parameters in $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{-i}^{t-1} = \{\theta_1^t, ..., \theta_{i-1}^t, \theta_{i+1}^{t-1}, ..., \theta_d^{t_1}\}. \tag{4.4}$$

Gibbs sampler requires that all the full conditional distributions in Equation (4.3) are standard statistical distributions so that we can conveniently obtain samples for the parameters of interest, and compute summary statistics such as posterior mean, variance and maximum a posteriori (MAP) estimates based on the samples. However, when one or more of these full conditional distributions are non - standard, we need to resort to the Metropolis-Hastings algorithm as described in Section 4.1.1. in order to obtain samples for these non-standard distributions.

The combination of Gibbs sampling and M-H algorithm has enabled inference on complex hierarchical Bayesian statistical models. In Algorithm1, we showcase an MCMC example to make inference from the Bayesian Mallows model for partial data when multiple user groups are present, as described in Section 3.2. For details of the "leap & shift" proposal distribution, we refer to [62].

Algorithm 1: Bayesian Mallows MCMC algorithm for Partial data with C clusters

Input: n_iters, C, ψ, λ, σ_α, $Z_n(\alpha)$, $\{\mathcal{S}^1, ..., \mathcal{S}^N\}$
Output: posterior samples of $\{\boldsymbol{\rho}_1, ..., \boldsymbol{\rho}_C\}$, $\{\alpha_1, ..., \alpha_C\}$, $\{\tilde{\boldsymbol{R}}^1, ..., \tilde{\boldsymbol{R}}^N\}$, $\{\tau_1, ..., \tau_C\}$, $\{z^1, ..., z^N\}$
Initialization: randomly generate $\{\boldsymbol{\rho}_1^0, ..., \boldsymbol{\rho}_C^0\}$, $\{\alpha_1^0, ..., \alpha_C^0\}$,$\{\tau_1^0, ..., \tau_C^0\}$; initialize $\tilde{\boldsymbol{R}}^{j,0} \in \mathcal{S}^j$ for $j = 1, ..., N$
for $t \leftarrow 1$ *to* *n_iters* **do**
  Gibbs sampler update $\tau_1, ..., \tau_C$;
  compute $n_c^t = \sum_{j=1}^{N} \mathbf{1}_{\{z^{j,t-1}=c\}}$ for $c = 1, ..., C$;
  sample $\{\tau_1^t, ..., \tau_C^t\} \sim \mathcal{D}(\psi + n_1^t, ..., \psi + n_C^t)$;
  for $c \leftarrow 1$ *to* C **do**
    M-H update $\boldsymbol{\rho}_c$;
    sample $\boldsymbol{\rho}^* \sim L\&S(\boldsymbol{\rho}^{t-1})$;
    sample $u \sim \mathcal{U}(0, 1)$;
    compute
    $$r_{\boldsymbol{\rho}} = \min\left\{1, \frac{P_{L\&s}(\boldsymbol{\rho}^{t-1}|\boldsymbol{\rho}^*)\pi(\boldsymbol{\rho}^*)}{P_{L\&s}(\boldsymbol{\rho}^*|\boldsymbol{\rho}^{t-1})\pi(\boldsymbol{\rho}^{t-1})} \exp\left[-\frac{\alpha_c^{t-1}}{n}\sum_{j=1}^{n_c^t}\{d(\tilde{\boldsymbol{R}}^{j,t-1}, \boldsymbol{\rho}^*) - d(\tilde{\boldsymbol{R}}^{j,t-1}, \boldsymbol{\rho}^{t-1})\}\right]\right\};$$
    if $u < r_{\boldsymbol{\rho}}$ **then**
      $\boldsymbol{\rho}_c^t \leftarrow \boldsymbol{\rho}^*$
    else
      $\boldsymbol{\rho}_c^t \leftarrow \boldsymbol{\rho}^{t-1}$
    end
    M-H update α_c ;
    compute
    $$r_\alpha = \min\left\{1, \frac{Z_n(\alpha^{t-1})^{n_c^t}\pi(\alpha^*)\alpha^*}{Z_n(\alpha^*)^{n_c^t}\pi(\alpha^{t-1})\alpha^{t-1}} \exp\left[-\frac{(\alpha^* - \alpha^{t-1})}{n}\sum_{j=1}^{n_c^t} d(\tilde{\boldsymbol{R}}^{j,t-1}, \boldsymbol{\rho}^t)\right]\right\};$$
    if $u < r_\alpha$ **then**
      $\alpha_c^t \leftarrow \alpha^*$
    else
      $\alpha_c^t \leftarrow \alpha^{t-1}$
    end
    Gibbs sampler update $z_1, ..., z_N$;
    for $j \leftarrow N$ **do**
      compute $p_{c^j} = \frac{\tau_c^t}{Z_n(\alpha_c^t)} \exp[-\frac{\alpha_c^t}{n} d(\tilde{\boldsymbol{R}}^{j,t-1}, \boldsymbol{\rho}_m^t)]$ for $c = 1, ..., C$;
      sample $z^{j,t} \sim \mathcal{M}(p_{1^j}, ..., p_{C^j})$
    end
    M-H update $\tilde{\boldsymbol{R}}^1, ..., \tilde{\boldsymbol{R}}^N$;
    for $j \leftarrow 1$ *to* N **do**
      sample $\tilde{\boldsymbol{R}}^{j*} \in \mathcal{S}^j$ from L&S($\tilde{\boldsymbol{R}}^{j,t-1}$) ;
      sample $u \sim \mathcal{U}(0, 1)$;
      compute
      $$r_{\tilde{\boldsymbol{R}}} = \min\left\{\exp\left\{-\frac{\alpha_{z_j^t}^t}{n}\left[d(\tilde{\boldsymbol{R}}^{j,*}, \boldsymbol{\rho}_{z_j^t}^t - \tilde{\boldsymbol{R}}^{j,t-1}, \boldsymbol{\rho}_{z_j^t}^t)\right\}\right\};$$
      if $u < r_{\tilde{\boldsymbol{R}}}$ **then**
        $\tilde{\boldsymbol{R}}^{j,t} \leftarrow \tilde{\boldsymbol{R}}^{j,*}$
      else
        $\tilde{\boldsymbol{R}}^{j,t} \leftarrow \tilde{\boldsymbol{R}}^{j,t-1}$
      end
    end
  end
end

4.2 Variational Inference (VI)

MCMC methods, albeit flexible and powerful, are often limited by its slow speed of convergence due to its random walk nature. Moreover, consecutive Markov Chain samples are often highly correlated, large amounts of samples are therefore needed before the target posterior distribution can be effectively approximated. The inefficiency is sometimes exacerbated by the fact that MCMC chains can be stuck in a subset of the state space while failing to explore other modes [55].

Many efforts have been put in to alleviate these limitations in order to achieve faster and more efficient MCMCs. Hamiltonian Monte Carlo (HMC) utilizes an auxiliary variable scheme to suppress the random walk behaviour, and Hoffman and Gelman[28] introduced the "No-U-Turn"(NUTS) algorithm to automate the selection of the step size and number of steps parameters required by HMC. Another group of methods take a "divide-and-conquer" approach, which try to partition a large scale dataset into several subsets, run MCMC on each partition independently, and then combine the results to achieve better mixing [5, 49]. Last but not least, parallelism can help increase the computing speed of MCMC [16, 48].

Variational inference on the other hand, approaches Bayesian inference not through directly sampling from the target posterior distribution, but by positing a family of distribution, i.e., variational distribution which is typically simpler than the target posterior distribution. The member of the variational distribution family that best approximates the target posterior distribution is then used as a replacement of the posterior distribution from which inference is made.

4.2.1 Mean-field VI

Supposed $\boldsymbol{x} = \{x_1, ..., x_n\}$ is a set of observations and $\boldsymbol{z} = \{z_1, ..., z_d\}$ is a set of latent parameters. The target posterior distribution can be expressed as

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{p(\boldsymbol{z}, \boldsymbol{x})}{p(\boldsymbol{x})}, \tag{4.5}$$

where the denominator in Equation (4.5), sometimes referred to as *evidence*, can be further expressed as

$$p(\boldsymbol{x}) = \int p(\boldsymbol{z}, \boldsymbol{x}) d\boldsymbol{z} \tag{4.6}$$

Instead of approaching the posterior distribution Equation (4.5) directly, VI posits a family of distributions $\mathcal{Q}$, and the task is then translated to finding the member $q(\boldsymbol{z}) \in \mathcal{Q}$ that best approximate $p(\boldsymbol{z}|\boldsymbol{x})$, using measured by the Kullback-Leibler (KL) divergence, i.e.,

$$q^*(\boldsymbol{z}) = \underset{q(\boldsymbol{z})\in\mathcal{Q}}{\arg\min}\, KL(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})), \tag{4.7}$$

where the KL-divergence between $q(\boldsymbol{z})$ and $p(\boldsymbol{z}|\boldsymbol{x})$ is defined as

$$\begin{aligned} KL(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) &= \int_{-\infty}^{\infty} q(\boldsymbol{z}) log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})} d\boldsymbol{z} \\ &= \mathbb{E}_q[log \frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}] \\ &= \mathbb{E}_q[logq(\boldsymbol{z})] - \mathbb{E}_q[logp(\boldsymbol{z}|\boldsymbol{x})] \\ &= \mathbb{E}_q[logq(\boldsymbol{z})] - \mathbb{E}_q[logp(\boldsymbol{z}, \boldsymbol{x})] + logp(\boldsymbol{x}). \end{aligned} \tag{4.8}$$

Observing the last line in Equation (4.8), we can see that the last term involves $p(\boldsymbol{x})$, which is usually not computationally feasible. Therefore, instead of optimizing the KL-divergence directly, an alternative function, namely the Evidence Lower Bound (ELBO) is often used as the objective function, defined as

$$\begin{aligned} \mathcal{L}_{ELBO} &= \mathbb{E}_q[logp(\boldsymbol{z}, \boldsymbol{x})] - \mathbb{E}_q[logq(\boldsymbol{z})] \\ &= -KL(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) + logp(\boldsymbol{x}). \end{aligned} \tag{4.9}$$

It can be observed from Equation (4.9) that minimizing the KL divergence is equivalent to maximizing the ELBO, as $logp(\boldsymbol{x})$ is a constant with respect to $q(\boldsymbol{z})$. So the objective function for the optimization problem is

$$q^*(\boldsymbol{z}) = \arg\max_{q(\boldsymbol{z}) \in \mathcal{Q}} \mathbb{E}_q[logp(\boldsymbol{z}, \boldsymbol{x})] - \mathbb{E}_q[logq(\boldsymbol{z})]. \tag{4.10}$$

The most popular choice to construct the variational distribution is the mean-field variational family. Simply put, mean-field VI assumes that all variational parameters are mutually independent, and each parameter is governed by its own variational density. i.e.,

$$q(\boldsymbol{z}) = \prod_{j=1}^{d} q_j(z_j). \tag{4.11}$$

With this assumption, we can re-write Equation (4.9) as

$$\begin{aligned}
\mathcal{L}_{ELBO} &= \mathbb{E}_q[logp(\boldsymbol{z},\boldsymbol{x})] - \mathbb{E}_q[logq(\boldsymbol{z})] \\
&= \mathbb{E}_q[logp(\boldsymbol{x}) \prod_{j=1}^{d} p(z_j|\boldsymbol{x})] - \mathbb{E}_q[\prod_{j=1}^{d} logq_j(z_j)] \\
&= \mathbb{E}_q[logp(\boldsymbol{x}) \prod_{j=1}^{d} p(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})] - \mathbb{E}_q[\prod_{j=1}^{d} logq_j(z_j)] \\
&= logp(\boldsymbol{x}) + \sum_{j=1}^{d} \mathbb{E}_q[logp(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})] - \sum_{j=1}^{d} \mathbb{E}_{q_j}[logq_j(z_j)] \\
&= logp(\boldsymbol{x}) + \sum_{j=1}^{d} \mathbb{E}_{q_j}\mathbb{E}_{q_{-j}}[logp(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})] - \sum_{j=1}^{d} \mathbb{E}_{q_j}[logq_j(z_j)]
\end{aligned} \tag{4.12}$$

Technically, we can take the liberty to choose any parametric form for $q_j(z_j)$, however, there are many models that determines the optimal form of the variational construction [9], and exponential family distributions are often preferred due to their unique properties.

4.2.2 Coordinate Ascent Variational Inference (CAVI)

The concept of the CAVI algorithm was introduced in [8]. While we are not directly utilizing CAVI in the remaining of this thesis, it is important to introduce it so that we can have a realization of how variational inference is implemented. The main idea of CAVI is to iteratively update for each component in the mean-field density while holding the others fixed. The update rule for the j-th component, i.e., $q_j(z_j)$ can be derived as follows. We first write out the portion of the loss function (4.12) that is attributed to the j-th component:

$$\begin{aligned}
\mathcal{L}_j &= \mathbb{E}_{q_j}\mathbb{E}_{q_{-j}}[logp(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})] - \mathbb{E}_{q_j}[logq_j(z_j)] \\
&= \int_{-\infty}^{\infty} q_j(z_j)\mathbb{E}_{q_{-j}}[logp(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})]dz_j - \int_{-\infty}^{\infty} q_j(z_j)logq_j(z_j)dz_j;
\end{aligned} \tag{4.13}$$

Now, we take the derivative with respect to the j-th component:

$$\frac{\partial \mathcal{L}_j}{\partial q_j} = \mathbb{E}_{q_{-j}}[logp(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})]dz_j - logq_j(z_j) - 1; \tag{4.14}$$

Setting the above derivative to be 0, we can obtain that the update rule for the j-th component is as follows:

$$q_j^*(z_j) \propto \exp\{\mathbb{E}_{q_{-j}}[logp(z_j|\boldsymbol{z}_{-j},\boldsymbol{x})]\}. \tag{4.15}$$

CAVI will update for each component iteratively until some criteria, for example, a threshold of the ELBO are met. CAVI does not guarantee that the global

optimum can be found as the function of ELBO is not always convex, however, a local optimum can be guaranteed [9]. One major limitation of CAVI is that it does not scale well due to the fact that all data points are considered when updating for the variational parameter at each step. To counter this, Hoffman et al.[29] introduced the Stochastic Variational Inference (SVI) algorithm, which enables VI to be scaled up significantly.

Variational inference has recently drawn a lot of attention due to its incorporation into deep neural networks [25], with prominent applications such as variational auto-encoder [36]. However, its theoretical development has been limited. Most applications of VI has been limited to using only conjugate exponential family distributions, usually under the mean-field assumption. Variational approximations for very complex distributions which go beyond the exponential family, discrete distributions whose gradients are not conveniently derived, and combinatorial problems where the mean-field assumption no longer applies, albeit attemps by Bouchard-Côté and Jordan[10] and Wand et al.[63], remain relatively uncharted territory. In Paper III, we explore the framework of VI to construct an approximation to the Bayesian Mallows method, to alleviate the scaling and "stickiness" limitation commonly associated with MCMC.

Chapter 5

Summary of papers

Paper I

Qinghua Liu, Marta Crispino, Ida Scheel, Valeria Vitelli, and Arnoldo Frigessi (2019). Model-based Learning from Preference data, ***Annual review of statistics and its application, 6, s 329-354.***

In Paper I, we explore many different tasks in the theme of preference learning, and provide a systematic overview of some of the most popular model-based methods that cater to these tasks. We categorize preference data as full rankings, partial rankings and pairwise comparisons, since other forms of preference data typically inherits one of these three data types in nature. Given a group of homogeneous users, we divide the main objectives of preference learning into two groups: (a) to learn the shared group consensus, or in other words, to perform rank aggregation; (b) to learn the individual user's personal preferences. When heterogeneous groups of users are present, we also need to (c) perform clustering to allocate users into their respective groups. The Bayesian Mallows model (BMM) is a versatile probabilistic method that is capable of performing rank aggregation, learning individual rankings and performing clustering based on all the fore-mentioned data types. We illustrate and compare the BMM with other popular methods through the real-life "potato ranking" dataset.

Both the BMM and the Plackett-Luce Model(PL) are capable of performing rank aggregation from both full and partial ranking data. Both methods characterize users' shared preferences on items as set of utility scores. While the BMM represents the consensus as a ranking $\boldsymbol{\rho} \in \mathcal{P}_n$, the PL model uses a continuous real-value vector $\{\mu^1, ..., \mu^n\} \in \mathbb{R}^n$. Through the "potato experiment", we discover that both methods' estimates of the group consensus reflect the true consensus among the users. However, while the choices of distance functions in the BMM have some impact on the resulted posterior distributions, the BMM's distributions are typically more peaked compared to the PL due to its discrete nature. When performing rank aggregations from paired comparisons, the Bradley-Terry Model (BT) is a strong contender. Similar to the PL, it also characterizes the preferences of items as a set of continuous real-valued utility scores, the BT's likelihood is constructed directly based on paired comparisons. Compared to the BMM, its estimate of the consensus is less accurate when the amount of paired comparisons is scarce, but the two methods' estimates are very similar as more information is available.

The models that are capable of learning individual preferences are BMM, the

Hierarchical Bradley-Terry Model (HBT), and collaborative filtering (CF). The HBT is an extension of the BT that allows for the utility scores to vary among the users. Through our experiment, we discover that consistent with the BT, the HBT is more susceptible to data sparsity compared to the BMM, and its posterior distributions are less peaked compared to the BMM due to its continuous nature. CF on the other hand, does not explicitly assume a shared consensus across the users, therefore cannot perform rank aggregation. CF performs matrix factorization directly on the sparse user-item utility matrix, and use the dot product of the factor matrices to infer each individual's preference. It can accurately predict each user's next top-k items, however, does not quantify the uncertainty, and suffers from low explainability.

When heterogeneous groups of users exist, we need to assign users to clusters. The clustering process can be a stand-alone pre-processing procedure, or be embedded in the preference learning procedure. The PL assumes a mixture model where each cluster has a weight. The BMM adds one more layer to its hierarchy to assign each individual to a cluster probabilistically.

Through the experiments, the BMM has showcased its versatility in handling various forms of preference data, and its strong ability to capture information when the data is sparse. This has revealed its potential to be applied to scenarios such as clicking data.

Paper II

Qinghua Liu, Andrew Henry Reiner, Arnoldo Frigessi and Ida Scheel (2019). Diverse personalized recommendations with uncertaity from implicit preference data with the Bayesian Mallows Model, ***Knowledge-Based Systems, Vol. 186, 104960.***

In Paper II, we adapt the Bayesian Mallows Method to learn personal preferences from clicking data and make diverse and accurate personalized recommendations.

Collaborative filtering (CF) is one of the most commonly used personalized recommendation algorithms due to its superior ability to achieve accurate recommendations for users. However, it has also been shown that CF often overlooks one other important aspect of successful recommendations - diversity. While post-processing strategies can often be used in combination with CF to increase its diversity performance, a laborious tuning process is often involved, and the accuracy-diversity trade-off seems inevitable. Furthermore, there is no interpretable uncertainty being estimated using CF.

We propose the Bayesian Mallows for Clicking Data (BMCD) to offer a new method to make personalized recommendations, which holistically considers accuracy, diversity and uncertainty estimation. We first assume that each user has a latent full ranking on her mind, and all of the items that she has clicked

on are top-ranked, while all the unclicked items are bottom-ranked in this latent ranking. We enforce this restriction regardless of each item's popularity among other users to ensure that the uniqueness of each user is captured. This construction is later proven to be effective at achieving diverse recommendations. We then assume there exists K clusters, within which we assume a Mallows distribution. An MCMC scheme is used to obtain samples of each user's full latent ranking as well as cluster assignment. We make k recommendations per user by calculating the posterior probability of each item to be among the user's next-top-k, and recommend the k items with the highest such posterior probabilities. These posterior probabilities also serve as the uncertainty estimates associated with the top-k recommendations.

We design a simulation study to compare the BMCD's recommendation accuracy and diversity performance with that of CF's, and the results clearly indicate that BMCD outperforms CF in both aspects. We also conduct a case study based on a real-life clicking dataset to compare the BMCD's accuracy-diversity performances with CF, and two post-processing strategies. Our experiment results show that while achieving a comparable recommendation accuracy as CF, BMCD can achieve more superior diversity performance without the need of tuning, nor at the cost of reducing accuracy. BCMD's diversity performance is especially exceptional in terms of coverage, i.e., the percentage of distinct items in the item catalogue ever recommended to the users. Additionally, both the simulation and the case study give strong evidence that the BMCD's uncertainty estimates are well calibrated and interpretable, and this enables us to identify which recommendations are more reliable than others.

Paper III

Qinghua Liu, Valeria Vitelli, Arnoldo Frigessi and Ida Scheel (2021). Pseudo-Mallows for Efficient Preference Learning

MCMC as an inference method for hierarchical Bayesian model is often limited by low speed of convergence. The BMCD method as introduced in Paper II, being a discrete model defined on the space of permutation, exacerbates this effect. Variational inference on the other hand, is a new framework to approach Bayesian inference, and in recent years, it has demonstrated its advantage of achieving faster inference from Bayesian models. In this work, we construct an approximation to the BMCD using variational inference, namely the Pseudo-Mallows method and demonstrate that the Pseudo-Mallows method can achieve good approximation to the BMCD with much fewer samples drawn, thus leading to much shorter computing time.

We first introduce the construction of the Pseudo-Mallows distribution, which exists in the form of a product of n factors, and each i-th factor depends on all previous $i-1$ factors. Therefore, the sequence of the factors matters, and this sequence $\{i_1, ..., i_n\}$ is our variational parameter to be optimized. Through

conjectures and an empirical study combined with some theoretical support, we propose that the most optimal sequence of $\{i_1, ..., i_n\}$ is closely related to the consensus parameter $\boldsymbol{\rho}^0$ of the data, and that the middle-ranked items in $\boldsymbol{\rho}^0$ should be top-ranked in $\{i_1, ..., i_n\}$ while the top- and bottom-ranked items in $\boldsymbol{\rho}^0$ should be bottom-ranked in $\{i_1, ..., i_n\}$, in order to achieve the best approximation to the BMCD. We name these optimal sequences the "V"-rankings, and we also prove that the "V"-rankings can be inferred from the data. We also provide a method to quickly obtain an estimate of the scale parameter α. We then extend the Pseudo-Mallows model to handle partial data and clicking data using data augmentation, in order to make personalized recommendation.

Through a systematic simulation study, we demonstrate that the Pseudo-Mallows construction can achieve good approximation to the BMCD MCMC in much less computing time since all samples drawn are independent, and much fewer samples are needed. With a real-life clicking dataset, we also showcase that the pseudo-Mallows can achieve accurate personalized recommendation in a timely manner, and the uncertainty estimation associated with each recommendation is well calibrated.

Paper IV

Øystein Sørensen, Marta Crispino, Qinghua Liu and Valeria Vitelli (2020). BayesMallows: an R Package for the Bayesian Mallows Model, ***The R Journal, Vol.12(1), s 324- 342***

In this work, we illustrate the methodological backgrounds and practical computational strategies of our own brain child: the *BayesMallows* R package. The *BayesMallows* R package is the most comprehensive R package that implements the Bayesian Mallows model to perform rank aggregation, predictions and clustering with uncertainty, while allowing for many choices of distance functions. It also provides convenient tools for its users to assess convergence and visualize the results.

We first recap the Bayesian Mallows Model's set up, and then discuss the difference between the *BayesMallows* and a few other R packages that also implement the Mallows model. We then explore the usage of the various functions in the *BayesMallows* package, and demonstrate how they can be applied to various data formats such as complete ranking data and paired comparisons through examples of code snippets. We focus not only on showing the readers the general usage of the package, but also on how to properly choose appropriate hyper parameters, and how to assess convergence using the visualisation tools to ensure that the MCMC estimation is efficient and accurate.

Chapter 6

Discussions

In this thesis, we have explored many facets of personalized preferences learning with the Mallows model. We have compared the Mallows model's characteristics with other popular preference ranking models (Paper I); extended and applied the Bayesian Mallows Method to clicking data (Paper II), and explored the variational approximation for faster inference (Paper III). Through this work, we have holistically explored the theme of preference learning and recommendations through a combination of methodological development, practical application and implementation. Yet, there are a few area that remain to be investigated.

In this thesis, particularly in Paper II, we have addressed the issue of recommendation diversity from a modelling perspective. Our BMCD method naturally incorporates diversity into considerations. However, after learning users' preferences on items, our recommendation strategy is the simple assumption that the items with the highest utility scores, in our cases, the posterior probabilities of being among the users' top- k, are to be recommended. In fact, the post-processing step of choosing the right items to recommend based on the utility scores is an interesting problem on its own. In Paper II, two post-processing methods, proposed by Ziegler et al.[67] and Adomavicius and Kwon[2] were tested out in combination of CF to enhance CF's diversity. Furthermore, there are many other post-processing methods available to better select items for recommendations, such as the graph-based method suggested by Antikacioglu and Ravi[3], the explanation-based diversification method [65], and the re-ranking method introduced by Steck[60]. However, we did not explore the possibility of post-processing due to the following reasons. First, the focus of our work is on the preference learning methodology, and by incorporating diversity into modelling, our BMCD method has achieved a good balance between accuracy and diversity. Second, users' true behaviours online often differ from offline testing results. Although post-processing can improve the offline diversity performance, this effect may not be reflected online, and online testing is actually the ultimate measurement of the quality of the recommendations. In reality, the sole process of making recommendations based on the learned personal preference is more intricate than just mathematical problems: which items should be recommended to the user also depends highly on the business domain and the overall business objective. Learning each user's preference on all the items is necessary, but not sufficient to determine "what should be recommended to whom".

Throughout this thesis, clicking data has been formulated as binary data, with the straightforward assumption that if a user has clicked on an item, it is marked as a 1 and 0 otherwise. This assumption is simple and intuitive, however, has

its limitation. First, it is sometimes unclear whether a non-click is due to the fact that the user dislikes the item or the user simply has not discovered the item. Assuming all clicked items are ranked higher than unclicked items is not a perfect solution. CF alleviates this problem by introducing a confidence variable. We did not investigate this issue further since this construction, although results in some un-clicked items being ranked too low, the recommendation quality is not serioiusly affected, since such items will appear on top among the unclicked items, and we only consider unclicked items for recommendation. Second, given the application's context, clicking data contains much more information than just binary information. For example, in an e-commerce setting, we can construct the user-item matrix based on the number of times an item is purchased by a user. On a streaming platform, we can encode the different kinds of user-item interactions, for example, 0 for no interaction, 0.5 for starting a movie, 1 for completing a movie. This way, we can learn from much more information. Our assumption that all clicked items are ranked higher than all non-clicked items still stands, but at the same time, we also gain information within the clicked item group about the relative rankings among them. Therefore, when applying the BMCD or the Pseudo-Mallows in a real life application, we can go beyond the binary user-item matrix and introduce more information based on the nature of the implicit data and the application's use case.

Another theme that is not addressed in this thesis is the "cold-start" problem [58] of recommender systems. The term "cold-start" refers to the situation where there is scarce or no data at all for certain users or items, and learning for these users or items is extremely challenging. The cold-start problem frequently occurs when new items and/or users are introduced to the system, and it is specifically relevant for applications such as news websites. Content-based recommender systems, which leverage user or item information such as age, gender, genre, descriptions, etc are often used during the cold-start stage. Such information, or covariates, can also be incorporated into our Mallows methods to alleviate the cold-start problem. One simplest way of utilizing covariates is by introducing them as a pre-processing step: we can use covariates to identify similar users or items that already exist in the system, and then treat the new items or users as an aggregation of their nearest neighbors in the user-item matrix. Neural networks [6] can also be used to identify nearest neighbors from covariates. Another option is to introduce covariates information directly into our model, possibly by adding one extra layer of regression in the hierarchy.

Implementation-wise, both the BMCD and the Pseudo-Mallows have the potential to speed up so that they can scale up to even bigger datasets. One factor that contributes to slow computing speed of both algorithm is that both algorithms contain the term $\sum_{j=1}^{N} d(\boldsymbol{R}^j, \boldsymbol{\rho})$ - which requires summing over all data points. For larger N, this term will obviously be computationally heavy. However, drawing inspiration from stochastic algorithms, we can randomly draw a subset of N', $N' << N$ data points, and then scale up the sum by a factor of N/N'.

Distributed computing framework such as MapReduce [19] can also help speed up the computation of a large sum. Both algorithms' scaling abilities are also limited when n is large. However, for the Pseudo-Mallows specifically, it should be relatively straightforward to reduce the number of items to be estimated. For each user j, when estimating the full individual ranking $\tilde{\boldsymbol{R}}^j$, the Pseudo-Mallows algorithm currently loops through all items. This procedure cannot be parallelized since the sampling of each item is dependent on all previous items. Inevitably, when n is large, the algorithm becomes cumbersome. However, when the aim is to make top - k recommendation, the items that are bottom-ranked are in fact not of interest. Conveniently, in the case of the Pseudo-Mallows, as each item is estimated individually with a tractable normalizing constant, we can skip the estimation process for a large number of items that show strong evidence of not being top ranked by user j. This will give the Pseudo-Mallows the potential to scale up to datasets with a very large number of items.

There are also a few points about the Pseudo-Mallows method introduced in Paper III that remain to be completed. First, our Pseudo-Mallows method handles the situation where only one homogeneous cluster exists, or where users' cluster assignments are available. When a mixture of clusters exists, it is theoretically possible to use a clustering algorithm such as K-means clustering [42] to pre-process the dataset, split the dataset into K single-cluster datasets accordingly, and then run the Pseudo-Mallows algorithm independently for each cluster. In practice however, K-means clustering rarely produces homogeneous clusters where users' rankings follow the Mallows distribution. Making accurate recommendations with these cluster assignments turn out not problematic during our experiments, however, the uncertainty estimation is not as well calibrated. As with the BMCD, we can introduce a clustering layer into our hierarchy to also estimate the cluster assignments probabilistically. However, as we have discovered while experimenting with the BMCD, the clustering of users converges very fast, and most users do not sway between clusters. Therefore, we did not further incorporate clustering into our model, but instead, suggested running the BayesMallows R package for a few hundred iterations to obtain the clustering assignments for each user, and then use the Pseudo-Mallows to learn individual preferences independently on each cluster.

One other possible extension to the Pseudo-Mallows is the estimation of α. In Paper III, we introduced a strategy to obtain a fast estimation of the scale parameter by simulating datasets with a grid of known α values. Overall, our estimation strategy can obtain a reasonable estimate of α in a speedy manner. Yet, unlike the consensus parameter $\boldsymbol{\rho}$ and the individual rankings $\tilde{\boldsymbol{R}}^j$, the scale parameter α is a continuous variable, therefore, it should be possible to derive its gradient, and optimize for α using stochastic gradient descent. We did not explore this option due to the fact that α is generally not a parameter that we draw further inference on; more importantly, the gradient of α is not a convex function, therefore, gradient descent algorithms can only guarantee to find a local optimum.

Bibliography

[1] Aberger, C. R. "Recommender: An analysis of collaborative filtering techniques". In: *Stanford University*. 2014.

[2] Adomavicius, G. and Kwon, Y. "Improving aggregate recommendation diversity using ranking-based techniques". In: *IEEE Transactions on Knowledge and Data Engineering* vol. 24, no. 5 (2011), pp. 896–911.

[3] Antikacioglu, A. and Ravi, R. "Post processing recommender systems for diversity". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 707–716.

[4] Asfaw, D. et al. "Time-varying rankings with the Bayesian Mallows model". In: *Stat* vol. 6, no. 1 (2017), pp. 14–30.

[5] Bai, Y., Craiu, R. V., and Di Narzo, A. F. "Divide and conquer: a mixture-based approach to regional adaptation for MCMC". In: *Journal of Computational and Graphical Statistics* vol. 20, no. 1 (2011), pp. 63–79.

[6] Barkan, O. and Koenigstein, N. "Item2vec: neural item embedding for collaborative filtering". In: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2016, pp. 1–6.

[7] Bennett, J., Lanning, S., et al. "The netflix prize". In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York. 2007, p. 35.

[8] Bishop, C. M. *Pattern recognition and machine learning*. springer, 2006.

[9] Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* vol. 112, no. 518 (2017), pp. 859–877.

[10] Bouchard-Côté, A. and Jordan, M. I. "Variational inference over combinatorial spaces". In: *Advances in Neural Information Processing Systems*. 2010, pp. 280–288.

[11] Bradley, R. A. and Terry, M. E. "Rank analysis of incomplete block designs: I. The method of paired comparisons". In: *Biometrika* vol. 39, no. 3/4 (1952), pp. 324–345.

[12] Carlin, B. P. and Chib, S. "Bayesian model choice via Markov chain Monte Carlo methods". In: *Journal of the Royal Statistical Society: Series B (Methodological)* vol. 57, no. 3 (1995), pp. 473–484.

[13] Chib, S. and Greenberg, E. "Understanding the metropolis-hastings algorithm". In: *The american statistician* vol. 49, no. 4 (1995), pp. 327–335.

[14] Chierichetti, F. et al. "Mallows models for top-k lists". In: *Advances in Neural Information Processing Systems*. 2018, pp. 4382–4392.

[15] Congdon, P. *Applied bayesian modelling*. Vol. 595. John Wiley & Sons, 2014.

[16] Corander, J., Gyllenberg, M., and Koski, T. "Bayesian model learning based on a parallel MCMC strategy". In: *Statistics and computing* vol. 16, no. 4 (2006), pp. 355–362.

[17] Crispino, M. et al. "A Bayesian Mallows approach to nontransitive pair comparison data: How human are sounds?" In: *The Annals of Applied Statistics* vol. 13, no. 1 (2019), pp. 492–519.

[18] De Lathauwer, L., De Moor, B., and Vandewalle, J. "A multilinear singular value decomposition". In: *SIAM journal on Matrix Analysis and Applications* vol. 21, no. 4 (2000), pp. 1253–1278.

[19] Dean, J. and Ghemawat, S. "MapReduce: Simplified data processing on large clusters". In: (2004).

[20] Diaconis, P. "Group representations in probability and statistics". In: *Lecture notes-monograph series* vol. 11 (1988), pp. i–192.

[21] Fligner, M. A. and Verducci, J. S. "Distance based ranking models". In: *Journal of the Royal Statistical Society: Series B (Methodological)* vol. 48, no. 3 (1986), pp. 359–369.

[22] Funk, S. *Netflix update: Try this at home*. 2006.

[23] Ge, M., Delgado-Battenfeld, C., and Jannach, D. "Beyond accuracy: evaluating recommender systems by coverage and serendipity". In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pp. 257–260.

[24] Gelman, A. et al. *Bayesian data analysis*. CRC press, 2013.

[25] Goodfellow, I. et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[26] Hastings, W. K. "Monte Carlo sampling methods using Markov chains and their applications". In: (1970).

[27] Herlocker, J. L. et al. "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* vol. 22, no. 1 (2004), pp. 5–53.

[28] Hoffman, M. D. and Gelman, A. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." In: *J. Mach. Learn. Res.* vol. 15, no. 1 (2014), pp. 1593–1623.

[29] Hoffman, M. D. et al. "Stochastic variational inference". In: *The Journal of Machine Learning Research* vol. 14, no. 1 (2013), pp. 1303–1347.

[30] Hu, Y., Koren, Y., and Volinsky, C. "Collaborative filtering for implicit feedback datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. Ieee. 2008, pp. 263–272.

[31] Ibe, O. *Markov processes for stochastic modeling*. Newnes, 2013.

[32] Jannach, D., Lerche, L., and Zanker, M. "Recommending based on implicit feedback". In: *Social Information Access*. Springer, 2018, pp. 510–569.

[33] Jawaheer, G., Szomszor, M., and Kostkova, P. "Comparison of implicit and explicit feedback from an online music recommendation service". In: *proceedings of the 1st international workshop on information heterogeneity and fusion in recommender systems*. 2010, pp. 47–51.

[34] Jiao, Y. and Vert, J.-P. "The Kendall and Mallows kernels for permutations". In: *International Conference on Machine Learning*. 2015, pp. 1935–1944.

[35] Karatzoglou, A. et al. "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering". In: *Proceedings of the fourth ACM conference on Recommender systems*. 2010, pp. 79–86.

[36] Kingma, D. P. and Welling, M. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[37] Koren, Y., Bell, R., and Volinsky, C. "Matrix factorization techniques for recommender systems". In: *Computer* vol. 42, no. 8 (2009), pp. 30–37.

[38] Li, S., Kawale, J., and Fu, Y. "Deep collaborative filtering via marginalized denoising auto-encoder". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 811–820.

[39] Lops, P., De Gemmis, M., and Semeraro, G. "Content-based recommender systems: State of the art and trends". In: *Recommender systems handbook*. Springer, 2011, pp. 73–105.

[40] Lu, T. and Boutilier, C. "Learning Mallows models with pairwise preferences". In: *ICML*. 2011.

[41] Luce, R. D. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[42] MacQueen, J. et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[43] Mallows, C. L. "Non-null ranking models. I". In: *Biometrika* vol. 44, no. 1/2 (1957), pp. 114–130.

[44] Marden, J. I. *Analyzing and modeling rank data*. CRC Press, 1996.

[45] Meila, M. and Chen, H. "Dirichlet process mixtures of generalized mallows models". In: *arXiv preprint arXiv:1203.3496* (2012).

[46] Metropolis, N. et al. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* vol. 21, no. 6 (1953), pp. 1087–1092.

[47] Mukherjee, S. et al. "Estimation in exponential families on permutations". In: *The Annals of Statistics* vol. 44, no. 2 (2016), pp. 853–875.

[48] Neiswanger, W., Wang, C., and Xing, E. "Asymptotically exact, embarrassingly parallel MCMC". In: *arXiv preprint arXiv:1311.4780* (2013).

[49] Nemeth, C., Sherlock, C., et al. “Merging MCMC subposteriors through Gaussian-process approximations”. In: *Bayesian Analysis* vol. 13, no. 2 (2018), pp. 507–530.

[50] Pazzani, M. J. and Billsus, D. “Content-based recommendation systems”. In: *The adaptive web*. Springer, 2007, pp. 325–341.

[51] Peska, L. and Vojtas, P. “Negative implicit feedback in e-commerce recommender systems”. In: *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics*. 2013, pp. 1–4.

[52] Plackett, R. L. “The analysis of permutations”. In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* vol. 24, no. 2 (1975), pp. 193–202.

[53] Rendle, S. and Schmidt-Thieme, L. “Pairwise interaction tensor factorization for personalized tag recommendation”. In: *Proceedings of the third ACM international conference on Web search and data mining*. 2010, pp. 81–90.

[54] Rendle, S. et al. “Learning optimal ranking with tensor factorization for tag recommendation”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2009, pp. 727–736.

[55] Robert, C. P. et al. “Accelerating MCMC algorithms”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* vol. 10, no. 5 (2018), e1435.

[56] Sarwar, B. et al. “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 285–295.

[57] Schafer, J. B. et al. “Collaborative filtering recommender systems”. In: *The adaptive web*. Springer, 2007, pp. 291–324.

[58] Schein, A. I. et al. “Methods and metrics for cold-start recommendations”. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002, pp. 253–260.

[59] Sorensen, O. et al. *BayesMallows: Bayesian Preference Learning with the Mallows Rank Model*. R package version 0.4.0. 2019.

[60] Steck, H. “Calibrated recommendations”. In: *Proceedings of the 12th ACM conference on recommender systems*. 2018, pp. 154–162.

[61] Tucker, L. R. “Some mathematical notes on three-mode factor analysis”. In: *Psychometrika* vol. 31, no. 3 (1966), pp. 279–311.

[62] Vitelli, V. et al. “Probabilistic preference learning with the Mallows rank model”. In: *The Journal of Machine Learning Research* vol. 18, no. 1 (2017), pp. 5796–5844.

[63] Wand, M. P. et al. “Mean field variational Bayes for elaborate distributions”. In: *Bayesian Analysis* vol. 6, no. 4 (2011), pp. 847–900.

[64] Wang, H., Wang, N., and Yeung, D.-Y. “Collaborative deep learning for recommender systems”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1235–1244.

[65] Yu, C., Lakshmanan, L. V., and Amer-Yahia, S. “Recommendation diversification using explanations”. In: *2009 IEEE 25th International Conference on Data Engineering*. IEEE. 2009, pp. 1299–1302.

[66] Zhou, Y. et al. “Large-scale parallel collaborative filtering for the netflix prize”. In: *International conference on algorithmic applications in management*. Springer. 2008, pp. 337–348.

[67] Ziegler, C.-N. et al. “Improving recommendation lists through topic diversification”. In: *Proceedings of the 14th international conference on World Wide Web*. 2005, pp. 22–32.

Papers

Paper II

Diverse Personalized Recommendations with Uncertainty from Implicit Preference Data with the Bayesian Mallows Model

Qinghua Liu, Andrew Henry Reiner, Arnoldo Frigessi, Ida Scheel

DOI: https://doi.org/10.1016/j.knosys.2019.104960

II

Knowledge-Based Systems 186 (2019) 104960

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Diverse personalized recommendations with uncertainty from implicit preference data with the Bayesian Mallows model☆

Qinghua Liu [a], Andrew Henry Reiner [b], Arnoldo Frigessi [b,c], Ida Scheel [a,*]

[a] Department of Mathematics, University of Oslo, P.O. Box 1053, Blindern, 0316 Oslo, Norway
[b] Oslo Center for Biostatistics and Epidemiology, Oslo University Hospital, Klaus Torgårds vei 3, 0372 Oslo, Norway
[c] Oslo Centre for Biostatistics and Epidemiology, University of Oslo, Sognsvannsveien 9, 0372 Oslo, Norway

ARTICLE INFO

Article history:
Received 11 February 2019
Received in revised form 8 August 2019
Accepted 15 August 2019
Available online 20 August 2019

Keywords:
Preference learning
Collaborative filtering
Clicking data
Probabilistic modeling

ABSTRACT

Clicking data, which exists in abundance and contains objective user preference information, is widely used to produce personalized recommendations in web-based applications. Current popular recommendation algorithms, typically based on matrix factorizations, often focus on achieving high accuracy. While achieving good clickthrough rates, diversity of the recommended items is often overlooked. Moreover, most algorithms do not produce interpretable uncertainty quantifications of the recommendations. In this work, we propose the Bayesian Mallows for Clicking Data (BMCD) method, which simultaneously considers accuracy and diversity. BMCD augments clicking data into compatible full ranking vectors by enforcing all the clicked items clicked by a user to be top-ranked regardless of their rarity. User preferences are learned using a Mallows ranking model. Bayesian inference leads to interpretable uncertainties of each individual recommendation, and we also propose a method to make personalized recommendations based on such uncertainties. With a simulation study and a real life data example, we demonstrate that compared to state-of-the-art matrix factorization, BMCD makes personalized recommendations with similar accuracy, while achieving much higher level of diversity, and producing interpretable and actionable uncertainty estimation.

1. Introduction

Personalized recommendations are widely used to help users and customers sort digital information for their purpose. From online streaming services to e-commerce websites, recommender systems can improve business efficiency, sort search results and enhance user experience.

Accuracy, reliability and diversity are some of the most important objectives for effective recommender systems. To help users find the items they prefer, recommendations should be accurate. At the same time, a recommender system should assess how reliable, or certain, the recommendations are, so that the users will not be disturbed by many irrelevant recommendations. To achieve this, uncertainty quantification of recommendations is essential. Furthermore, recommendations should be diverse. From a user's standpoint, a recommendation list is more interesting if it consists of a variety of items of different categories and genres; from a vendor's point of view, it is more cost efficient for rare and less popular items to have more exposure because the licensing of such items is less costly [1]. The importance of diversity is often overlooked by many recommender systems, and it is challenging to achieve both high accuracy and diversity. There is scarce information about the less popular items, therefore, it is much more risky to consider such items for recommendations. The trade-off between accuracy and diversity is referred to as the "accuracy–diversity dilemma" [2–4].

Personalized recommendations are based on the users' preference data, which can be explicit feedbacks such as ratings, and implicit feedbacks such as click stream data. Clicking data is easy to collect, exists in great abundance, and often better reflects user preferences compared to ratings. However, the interpretation of clicking data can be challenging, as there is no direct negative feedback from users [5], and the data naturally exhibits high sparsity [6].

The state-of-the-art approach using implicit feedback for personalized recommendation is the Collaborative Filtering for Implicit Data method developed by Hu et al. [5]. This method is based on matrix factorization (MF). It is effective and scalable,

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.104960.

* Corresponding author.
E-mail addresses: qinghual@math.uio.no (Q. Liu), a.h.reiner@medisin.uio.no (A.H. Reiner), arnoldo.frigessi@medisin.uio.no (A. Frigessi), idasch@math.uio.no (I. Scheel).

https://doi.org/10.1016/j.knosys.2019.104960

and is commonly adopted by commercial applications [7]. However, there are some drawbacks. First of all, it does not provide interpretable uncertainty quantifications: when an item is recommended to a user, the method does not quantify the reliability of the recommendation. More importantly, the collaborative filtering framework does not consider diversity, and it has a tendency to favor the most popular items. While achieving high accuracy, these recommendations can sometimes be monotonous and lack diversity. Post-processing methods have been proposed to improve the diversity performance of collaborative filtering [8,9]. These methods usually contain a tuning parameter that balances the accuracy–diversity tradeoff. However, it is often impossible to achieve higher diversity without sacrificing the accuracy performance, and a poor choice of the tuning parameter can heavily impair the accuracy performance.

The method we introduce in this paper takes a more holistic approach. Instead of considering accuracy and diversity as separate objectives, our method embeds diversity considerations into the model for learning user preferences. In addition it provides interpretable uncertainty quantifications. Our method is called the Bayesian Mallows for Clicking Data (BMCD) method, and is constructed by further developing the approach introduced by Vitelli et al. [10]. We assume that users prefer clicked items to unclicked items, and individual clicking data is subsequently augmented to ranking vectors by enforcing all the clicked items to be top-ranked. This construction ensures that the uniqueness of each user, demonstrated especially by the rare items that are clicked, can be preserved and subsequently learned by the model. Our method inherently consider diversity in the model, achieving a good accuracy–diversity balance without the use of tuning parameters. Through a simulation study and an offline testing with a real life dataset provided by the Norwegian Broadcasting Corporation (NRK), we illustrate BMCD's effective accuracy–diversity performance, comparing to the state-of-the art Collaborative Filtering for Implicit Data method, and two popular post-processing diversity enhancements methods.

The main contribution of our work is a new method for making personalized recommendation based on implicit data, which inherently balances accuracy and diversity without the need of a tuning parameter. Contrasting with the current accuracy-driven methods such as Collaborative Filtering [5], our method holistically considers both accuracy and diversity. Compared to the diversity-driven post-processing methods [8,9], our method does not involve tuning parameters to achieve the accuracy–diversity balance, and does not sacrifice accuracy for diversity. Moreover, the Bayesian nature of our method ensures that each recommendation made is associated with a calibrated and interpretable uncertainty estimation.

The paper is organized as follows: We start by discussing related work in Section 2. In Section 3 we summarize the Bayesian Mallows Method, and then we introduce BMCD, and show how we can make personalized recommendations based on posterior probabilities. In Section 4, we introduce the evaluation metrics: accuracy and four diversity metrics. In Section 5.1 we explain the simulation study and demonstrate how BMCD makes recommendations with uncertainty quantification. In Section 5.2 we present a detailed comparison of BMCD's accuracy and diversity compared to Collaborative Filtering for Implicit Data for the simulation study. In Sections 5.3.2 and 5.3.4 we apply both methods to the NRK dataset, and compare their accuracy and diversity performances. It is followed by Section 5.4, which is dedicated to the comparison between BMCD and two post-processing methods in combination with Collaborative Filtering, based on the NRK dataset. We compare these methods' performances in terms of the accuracy–diversity balance. Last, a summary and further work are included in Section 6.

2. Related work

Collaborative filtering [11] is a framework utilizing user–item interaction data to make personalized recommendations through borrowing strength across the pool of users and items.

User-based collaborative filtering is an early method. For a particular user, the basic idea is first to discover other users who have similar preferences, often measured by cosine similarities or Pearson's correlation coefficient. After such neighbors are identified, recommendations are made based on an aggregation of the neighbors' preferences. User-based collaborative filtering is intuitive and easy to implement, however, it is often limited by the sparsity of the data as well as scalability. Instead, Sarwar et al. [12] proposed an item-based collaborative filtering algorithm. For a given user, her preference of an unknown item is predicted based on the users' past preferences of the k most similar items.

Matrix Factorization (MF)-based collaborative filtering methods are among the most successful [13]. The MF method proposed by Koren et al. [14] is developed for a user–item rating matrix. The data matrix $\mathbb{X}$ has dimensions $N \times n$, where N is the number of users and n is the number of items. Each entry x_{ij} is the rating given by a user j to an item i, or is empty. Assume that each user j has rated $\leq n$ items. MF obtains two reduced-dimension matrices $\boldsymbol{U}^{N\times L}$ and $\boldsymbol{V}^{n\times L}$, with $L < n$, so that their product will be a full matrix $\hat{\mathbb{X}}$ that approximates the original rating matrix $\mathbb{X}$. $\hat{\mathbb{X}}$ predicts, for each user, the ratings of the items that the user has not rated. Hu et al. [5] extended the method to implicit data. In this paper, BMCD is compared with the method in [5] since this is the widely adopted, state-of-the-art method. For more details on collaborative filtering, see [15].

Hu et al. [5] introduced the Collaborative Filtering for Implicit Data method (CF), which extends the classic matrix factorization method. It can be applied to datasets based on implicit user feedbacks, such as clicking data. We now denote the implicit user–item matrix as $\mathbb{X}$. The content of x_{ij} depends on the use case, for example, it can represent the number of times user i has clicked on item j. First, a binary matrix $\mathbb{W}$ is introduced by binarizing $\mathbb{X}$ such that w_{ij} is set to 1 if $x_{ij} > 0$, and 0 otherwise, i.e., w_{ij} is set to 1 if user j has clicked item i, and 0 otherwise. Second, a set of "confidence" variables c_{ij} is introduced. The rationale behind this variable is that different interactions indicate different levels of certainty that an item is preferred by the user. One choice for c_{ij} is: $c_{ij} = 1 + \beta x_{ij}$, $\beta \geq 0$. Finally, the factor matrices are obtained through minimizing the penalized loss function $\min_{U,V} \sum_{j,i} c_{ij}(w_{ij} - \boldsymbol{u}_j^T \boldsymbol{v}_i) + \theta(\sum_j \|\boldsymbol{u}_j\|^2 + \sum_i \|\boldsymbol{v}_i\|^2)$, where both $\boldsymbol{u}_j$ and $\boldsymbol{v}_i$ are L-dimensional column vectors. The last term in the loss function is a regularization term and is added to reduce overfitting. The parameters β, θ, and the reduced dimension of the factor matrices L are determined by cross-validation, while the minimization process is often achieved using algorithms such as alternating least square (ALS) [14]. In the later sections, the term "CF" refers exclusively to the method proposed by Hu et al. [5], and we use its implementation in Apache Spark [16]. BMCD will later be compared with this CF method, in terms of recommendation performances.

To address the accuracy–diversity dilemma, Zhou et al. [2] proposed a graph-based method inspired by the heat diffusion process. One accuracy-driven model and one diversity-driven model are combined with linear weighted average to balance accuracy and diversity. Karakaya and Tevfik [17] introduced a modification of Koren et al. [14]'s MF model for explicit feedback by penalizing popular items to improve diversity. However, the method has not been extended to implicit datasets.

Post-processing of recommendations can help enhance diversity. Antikacioglu et al. [18] proposed a bipartite graph-based

post-processing method. After a recommendation model is fitted, a score for each user–item pair is obtained, and then these scores serve as weights for the user–item edges. The recommendation process is modeled as a maximum-weight bipartite graph matching problem, and diversity is achieved by imposing diversity-related constraints to the optimization, which can be solved using algorithms for minimum cost flow problems. This method could post-process both BMCD and the Collaborative Filtering for Implicit Data Method, but we do not pursue this any further.

The most popular post-processing methods are proposed by Adomavicius et al. [9] and Ziegler et al. [8]. Both methods are used in combination with a model for recommendation, such as CF, which predicts a score for each user–item pair. To make k recommendations for each user, instead of using the traditional recommendation method, which simply recommends the k items with the highest such scores for each user, these methods impose special rules based on the predicted user–item scores. Adomavicius et al.'s [9] approach introduces a threshold: for each user, if there exists $l \geq k$ items whose scores have surpassed the threshold, the k least popular items among them are recommended to the user. If there exists $0 < l < k$ items whose scores surpassed the threshold, all the l items are recommended first, and the rest of the $k - l$ items are recommended using the traditional method. In the case where no item has surpassed the threshold, the traditional recommendation method is used to recommend the top k items to the user. The threshold is the tuning parameter that balances accuracy and diversity — a high threshold makes the recommendations more accuracy driven, while a low threshold results in more diverse but likely less accurate recommendations. Ziegler et al. [8] introduced an iterative approach instead. First, for each user, the top -1 item is added to the recommendation list using the traditional method. For each item that is not in the recommendation list, a similarity score is calculated between the item and the current recommendation list, and these items are ranked based on the similarity scores — from the least similar to the most similar. We denote this ranking as $rank_{sim}$. At the same time, another ranking is produced based on the ranking of the scores in descending order, denoted as $rank_{scores}$. The two rankings are combined in a linear weighted average, i.e. $rank_{combined} = \alpha\ rank_{sim} + (1-\alpha) rank_{score}$, $0 \leq \alpha \leq 1$, and the item with the lowest combined ranking (top-ranked) is added to the recommendation list. This procedure is repeated until k recommendations are reached. Clearly, α is the tuning parameter, for which a high value closer to 1 makes the algorithm more diversity-driven. We will use these two post-processing methods in combination with CF, and compare their accuracy–diversity performances with BMCD in Section 5.4.

3. Bayesian Mallows for Clicking Data (BMCD)

Consider a dataset of N users and n items $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$. Suppose first that each user j indicates her preferences with a ranking of all n items, $\boldsymbol{R}_j = \{R_{1j}, R_{2j}, \ldots, R_{nj}\}$, where R_{ij} is the rank assigned to item i by user j, $i = 1, \ldots, n, j = 1, \ldots, N$. The Mallows model is a probabilistic model on the space of permutations of n items $\mathcal{P}_n$. In the simplest case, assuming that all users share a common latent consensus $\boldsymbol{\rho} \in \mathcal{P}_n$, it has the form of $P(\boldsymbol{R}_j = \boldsymbol{r}|\alpha, \boldsymbol{\rho}) = \frac{1}{Z_n(\alpha,\boldsymbol{\rho})}\exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \boldsymbol{\rho})\}$, where α is a scale parameter, and $d(\boldsymbol{r}, \boldsymbol{\rho})$ is a distance between $\boldsymbol{r}$ and $\boldsymbol{\rho}$. Possible choices of distance functions include the footrule distance, the Spearman distance, and the Kendall distance. In this paper, we choose the footrule distance, defined as $d(\boldsymbol{r}, \boldsymbol{\rho}) = \sum_{i=1}^{n} |r_i - \rho_i|$, because of its effectiveness [19]. Other distances can also be used. Lastly, $Z_n(\alpha, \boldsymbol{\rho}) = \sum_{\boldsymbol{r}\in\mathcal{P}_n} \exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \boldsymbol{\rho})\}$ is the normalizing function. As the footrule distance is a right-invariant distance function, the partition function Z_n is independent of $\boldsymbol{\rho}$, and only depends on α, hence we denote it as $Z_n(\alpha)$. For $n < 50$, $Z_n(\alpha)$ has been computed [10], but is otherwise not analytically computable. When $n \geq 50$, the asymptotic approach introduced by Mukherjee et al. [20] and the importance sampling scheme introduced in [10] are available.

Realistically however, it is uncommon that all users are homogeneous. Assume that the N users are grouped in C clusters, and within each cluster, users share a common latent consensus. For each of the homogeneous clusters, we assume a Mallows distribution with parameters α_c, $\boldsymbol{\rho_c}$, $c = 1, \ldots, C$. The random variable denoted by $z_j \in \{1, \ldots, C\}$ assigns user j to cluster z_j. Assuming that users' preferences are conditionally independent given the Mallows parameters and their cluster assignments z_j, the likelihood function is hence

$$P(\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N | \{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N) = \prod_{j=1}^{N} [Z_n(\alpha_{z_j})]^{-1} \exp\{-\frac{\alpha_{z_j}}{n} d(\boldsymbol{R}_j, \boldsymbol{\rho}_{z_j})\}. \quad (1)$$

Vitelli et al. [10] introduce a Bayesian version of this model. The Mallows parameters $\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}$ are assumed a priori mutually independent. An exponential prior with hyperparameter λ is chosen for $\alpha_c, c = 1, \ldots, C$, i.e., $\pi(\alpha_1, .., \alpha_c|\lambda) = \lambda^C \exp\{-\lambda \sum_{c=1}^{C} \alpha_c\}$. For $\boldsymbol{\rho}_c, c = 1, \ldots, C$, the noninformative uniform prior $\pi(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_C) = n!^{-C}$ is chosen. The prior for the cluster assignments $z_j, j = 1, \ldots, N$ is $p(z_1, \ldots, z_N|\tau_1, \ldots, \tau_C) = \prod_{j=1}^{N} \tau_{z_j}$, where the probabilities $\tau_1, \ldots, \tau_C$ follow a Dirichlet prior $\pi(\tau_1, \ldots, \tau_C) = \Gamma(\psi C)\Gamma(\psi^{-C}) \prod_{c=1}^{C} \tau_c^{\psi-1}$, $\psi > 0$. Hyperparameters ψ and λ are assumed to be fixed, see [10] for guidelines.

The posterior distribution of $\{\{\alpha_c, \boldsymbol{\rho_c}\}_{c=1,\ldots,C}, z_1, \ldots, z_N\}$ is therefore

$$P(\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N | \boldsymbol{R}_1, \ldots, \boldsymbol{R}_N) \propto \pi(\alpha_1, \ldots, \alpha_C|\lambda)\pi(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_C)p(z_1, \ldots, z_N|\tau_1, \ldots, \tau_C) \cdot \pi(\tau_1, \ldots, \tau_C)P(\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N|\{\alpha_c, \boldsymbol{\rho}_c\}, z_1, \ldots, z_N). \quad (2)$$

We will now extend the Bayesian Mallows model to clicking data. For clicking data, the full ranking of the n items is not available and needs to be inferred from the clicking data. We denote the latent, full individual ranking vector for user j as $\tilde{\boldsymbol{R}}_j$. Suppose that each user j has clicked on a subset of the items $\mathcal{A}_j \subseteq \mathcal{A}$, with the number of clicks $|\mathcal{A}_j| = c_j$. It is common to assume that a clicked item is preferred by the user to any other un-clicked item [21]. This assumption also ensures that the uniqueness of each user is enforced, as all clicked items are forced to be more preferred to non-clicked items, regardless of the items' popularities. For each user j, the set of rankings compatible with this assumption is $\mathcal{S}_j(\mathcal{A}_j) = \{\tilde{\boldsymbol{R}}_j \in \mathcal{P}_n \text{ s.t. } \tilde{R}_{ij} < \tilde{R}_{kj} \text{ if } A_i \in \mathcal{A}_j \text{ and } A_k \in \mathcal{A}_j^c, \forall i, k, i \neq k\}$.

Given the clicking data, the goal is hence, to sample from the posterior distribution

$$P(\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N | \mathcal{A}_1, \ldots, \mathcal{A}_N) = \sum_{\tilde{\boldsymbol{R}}_1 \in \mathcal{S}_1(\mathcal{A}_1)} \cdots \sum_{\tilde{\boldsymbol{R}}_N \in \mathcal{S}_N(\mathcal{A}_N)} P(\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N, \tilde{\boldsymbol{R}}_1, .. \tilde{\boldsymbol{R}}_N | \mathcal{A}_1, \ldots, \mathcal{A}_N) \quad (3)$$

To make inference, we follow a Markov Chain Monte Carlo (MCMC) scheme similar to the one in [10]. Each iteration of the algorithm consists of three major steps:

(i) Update the parameters $\{\alpha_c, \boldsymbol{\rho}_c\}$ within each cluster $c = 1, .., C$, given the current values of the individual rankings $\tilde{\boldsymbol{R}}_j$, and the cluster assignments $z_j, j = 1, \ldots, N$

(ii) Re-assign users to clusters based on the current values of the parameters $\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}$ and the individual ranking vectors $\tilde{\boldsymbol{R}}_j$, $j = 1, \ldots, N$

(iii) Update $\tilde{\boldsymbol{R}}_j$ for each user j given the current values of $z_1, \ldots, z_N$ and $\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}$

As in [10], we use a Metropolis–Hasting algorithm for step (i), and a Gibbs sampler for step (ii). For step (iii), we sample from $P(\tilde{\boldsymbol{R}}_1, \ldots, \tilde{\boldsymbol{R}}_N | \{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N, \mathcal{A}_1, \ldots, \mathcal{A}_N)$. Given the cluster assignments and the Mallows parameters, the individual rankings are conditionally independent. Therefore, for each user j, we can independently sample from the posterior $P(\tilde{\boldsymbol{R}}_j | \alpha_{z_j}, \boldsymbol{\rho}_{z_j}, z_j, \mathcal{A}_j)$ using a Metropolis–Hasting algorithm, where a new $\tilde{\boldsymbol{R}}_j'$ for each user must be proposed. One convenient way is to choose two items i, k such that $\{i, k\} \in \mathcal{A}_j$ or $\{i, k\} \in \mathcal{A}_j^c$, and then swap the rankings of the two items for each user j. This proposal is symmetric, and each proposed latent full individual ranking vectors $\tilde{\boldsymbol{R}}_j'$ is accepted with probability $\min\{1, \exp[-\frac{\alpha_{z_j}}{n}(d(\tilde{\boldsymbol{R}}_j', \rho_{z_j}) - d(\tilde{\boldsymbol{R}}_j, \rho_{z_j}))]\}$. Another way of proposing a new $\tilde{\boldsymbol{R}}_j'$ is to treat each $\tilde{\boldsymbol{R}}_j$ as two parts: the clicked part and the un-clicked part. The "leap-and-shift" algorithm in [10] can then be used separately for the two parts.

To make personal recommendations, the variables of interest are the latent augmented full ranking $\tilde{\boldsymbol{R}}_j$ for each user. For a given user j that has clicked on c_j items, the objective of making $k \geq 1$ recommendations is equivalent to inferring which items are to be ranked as the user's $c_j + 1$th, …, $c_j + k$th items. We therefore calculate for each user j and each item i the posterior probability to be ranked between $c_j + 1, \ldots, c_j + k$, which we refer to as the "next top - k" items. That is, we estimate for each user j and each item i

$$\begin{aligned} P_{ij} &= P(c_j + 1 \leq \tilde{R}_{ij} \leq c_j + k | \mathcal{A}_1, \ldots, \mathcal{A}_N) \\ &= P(\tilde{R}_{ij} \leq c_j + k | \mathcal{A}_1, \ldots, \mathcal{A}_N). \end{aligned} \tag{4}$$

Once estimated, these posterior probabilities are later ranked for each user j in descending order, and the k items with the highest such probabilities are recommended to the user. The estimated top-k probabilities are referred to as the top posterior probabilities (TPP), and the set of k recommended items for user j is denoted as $Rec_{j,k}$. Section 6 in the supplement contains a structured description of our algorithms.

4. Recommendation evaluation—accuracy and diversity

In this section, we introduce the assessment metrics that will be used in order to assess the accuracy and diversity performances of the recommendation methods.

To assess recommendation accuracy, the next $k \geq 1$ recommendations are made for each user. In simulations, the recommendations are later compared with the truth. In offline experiments, based on a train-test split of the dataset, accuracy is measured as the percentage of the recommended items that are clicked in the test set. For online experiments, the truth is obtained by experimentation. The drawback of offline experiments compared to online experiments is that the recommendations are not actually given to the users, and hence the truth defined by the test set is not a response to the recommendations. This might be problematic for the recommendation of less popular items, since the users might not even be aware of these items, and hence could not have clicked them in the test set. Offline training-test experimentation is often the only and best alternative for assessing accuracy.

Despite being an important measure of performance, accuracy is not the only factor that defines successful recommendations [22,23]. User experience can be greatly enhanced when recommendations are diverse, and hence has the potential to be novel and surprising. To assess the diversity of recommendations, we adopt the following four metrics: coverage [24], correct coverage, intra-list similarity [8,25] and novelty [2].

4.1. Coverage

Ge et al. [24] introduced the metric "coverage", defined as

$$\text{coverage} = \frac{\#\text{ distinct items recommended to users}}{\#\text{ distinct items eligible for recommendation}},$$

the percentage of the distinct items ever recommended to the users. A recommender system with a high coverage has exploited its pool of items more efficiently, and their users, collectively, are exposed to a wider spectrum of items.

This coverage metric has one major limitation. For a highly inaccurate recommender system, in the extreme case, when recommendations are made randomly, the coverage can be very high while the recommendation accuracy is extremely low. Therefore, we also introduce the "correct coverage" metric, defined as:

$$\text{correct coverage} = \frac{\#\text{ distinct items recommended and clicked by at least one user}}{\#\text{ distinct items eligible for recommendation}}$$

4.2. Intra-list similarity

Ziegler et al. [8] introduced the "Intra-list similarity" metric to assess diversity on an individual level. The rationale behind this metric is that, on an individual level, each user tends to prefer recommendations from various categories. A recommendation list that contains items from only one or a few specific categories (for example, a list of only Harry Potter movies), are far less exciting compared to a good mixture of very different items (even for Harry Potter fans). Similarity between two items a, b is measured by binary cosine similarity [25] based on the training data, defined as

$$CosSim(a, b) = \frac{\#\text{ users clicked both a and b}}{\sqrt{\#\text{ users that clicked } a} \times \sqrt{\#\text{ users that clicked } b}},$$

and the intra-list similarity metric is hence defined as

$$\text{Intra-list similarity} = \frac{1}{N}\sum_{j=1}^{N} \sum_{a,b \in Rec_{j,k}, b<a} CosSim(a, b).$$

It is desirable for a recommender to have a low intra-list similarity.

4.3. Novelty

The novelty measure, introduced by Zhou et al. [2], assesses the recommender's ability to recommend items less explored by the users. It is defined as

$$\text{novelty} = \frac{1}{N}\sum_{j=1}^{N} \sum_{i \in Rec_{j,k}} \frac{|\log_2 pop_i|}{k},$$

in which pop_i refers to the popularity of item i, in this case, the fraction of all clicks attributed to item i in the training data. A recommender that recommends rare and less popular items, and hence makes novel recommendations, will have a high novelty score. It is desirable to make novel recommendations because a recommendation list consisting of only the most popular items lacks personalization. In addition, the less popular items are challenging to be recommended due to lack of data [23], and are often valuable to the business [1].

5. Experiment and results

In this section, we conducted a detailed comparison between the recommendation performances by BMCD and the Apache Spark [16] implementation of CF through a simulation study, as well as an offline case study with a dataset provided by the Norwegian Broadcasting Corporation (NRK). Each method will be assessed in terms of recommendation accuracy as well as diversity. Using the NRK dataset, we also compare the accuracy–diversity performances between BMCD and the post-processed versions of CF in Section 5.4.

5.1. Simulation study design

In this simulation study, we consider a group of $N = 3000$ users and $n = 50$ items. The users are partitioned in $C = 3$ equally sized and distinct clusters. The users in each cluster are given full ranking vectors $\boldsymbol{R}_j$ sampled from the Mallows model using the sampler in [10]. For each cluster c, the parameters are chosen to be $(\alpha = 3, \boldsymbol{\rho_c})$, with $\boldsymbol{\rho_1} = \{1, 2, .., 50\}$, $\boldsymbol{\rho_2} = \{50, 49, \ldots, 1\}$, and $\boldsymbol{\rho_3} = \{39, 36, 11, 1, 13, 12, 8, 48, 20, 49, 29, 32, 22, 28, 19, 5, 42, 18, 15, 7, 6, 27, 24, 16, 46, 4, 21, 26, 34, 44, 25, 43, 41, 38, 35, 37, 45, 2, 14, 50, 40, 47, 9, 23, 30, 31, 3, 10, 33\}$. The 3 consensuses are chosen since they will produce 3 distinct clusters that separate well. Hence, a dense $N \times n$ ranking dataset is obtained, which will later serve as the ground truth for checking recommendation accuracy, and from which we will build the incomplete clicking dataset.

To simulate clicking data, the full ranking dataset is converted to a binary dataset in the following way. For each user $j = 1, \ldots, N$, we draw the number of clicks c_j from a truncated Poisson distribution with parameter $\lambda = 5$, truncated to a minimum of 1. Thereafter, the top ranked c_j items are considered "clicked", while the rest of the items considered "unclicked". In other words, for each user j, we obtain $\mathcal{A}_j = \{A_i : R_{ij} \leq c_j\}$.

We generate independently 20 such datasets, and use both CF and BMCD to recommend $k = 5$ and $k = 10$ items for each user, i.e., to predict for each user j which items are ranked among $c_j + 1, \ldots, c_j + k$. The parameters for CF are determined through 10-fold cross validation. Although the ground truth dataset is generated from a Mallows model, which can impose some bias towards BMCD in terms of accuracy checking, the dataset is converted to a binary clicking dataset for model fitting. The binarization adds great sparsity to the dataset, and converts ranking vectors into binary vectors, which the Mallows model is not defined for. BMCD's advantages in inference are considerably reduced due to the binarization.

To use BMCD, the number of clusters C needs to be determined first. We run Algorithm 1 and 2 in the supplementary material with random initialization and varying numbers of clusters $C = 2, 3, \ldots, 8$. For each value of C, we estimate the posterior mean of the sum of within cluster footrule distances (MWCD), defined as

$$\text{MWCD} = \mathbb{E}[\sum_{c=1}^{C}\sum_{j=1}^{N} d(\tilde{\boldsymbol{R}}_j, \boldsymbol{\rho}_{z_j})|\mathcal{A}_1, \ldots, \mathcal{A}_N],$$

by the natural Monte Carlo mean. For each of the 20 simulated datasets, the number of clusters C with the smallest MWCD is chosen. It turns out that $C = 3$ is chosen for all runs, except for run numbers 5, 8, and 20, for which $C = 4$. Fig. 1 in the supplementary material Section 1 shows the boxplots of the posterior sum of within cluster distances for 3 selected runs (1, 5, 10).

The MCMC is run for 1 million iterations, with the first 500 000 iterations discarded as burn-in. Similar to the set up in [10], parameters $\{\alpha_1, \ldots \alpha_C\}$ are only proposed for update every 10 iterations. The trace plots of α_c for run 1, $c = 1, 2, 3$, after the burn-in period is included in the supplementary material Section 3. Convergence was checked by using multiple starting points of the MCMC chains. Recommendations of the Bayesian Mallows method are made based on TPP. The recommendation procedure is described in Section 3 in the supplementary material.

5.2. Simulation results and discussion

5.2.1. Recommendation accuracy

After recommendations are made, we refer to the ground truth full ranking vectors $\boldsymbol{R}_j$ to check whether the recommended items are truly among each user's next-k items.

Table 1 shows the recommendation accuracy using CF and BMCD to predict each user's next $k = 5$ and $k = 10$ items. It can be observed that BMCD makes slightly more accurate recommendations compared to CF in predicting both the next 5 and next 10 items. The accuracy advantage over CF is more significant in the next 10 case. In addition, BMCD's accuracy performance is less varied than that of CF's. BMCD's high accuracy demonstrates that the Bayesian Mallows model is a good fit for the recommendation problem, and that the recommendation process, which is based on posterior probability estimations, can accurately reflect users' true preferences.

We have also discovered that the number of clusters C chosen has little effect on the overall recommendation accuracy, as long as the number of clusters chosen is not too small. As shown in Fig. 1, the overall recommendation accuracy stabilizes after $C \geq 3$. It is apparent that if the number of clusters C is too small, users that do not share a consensus, i.e., users of very different tastes, are grouped together. Hence, the estimated ρ will not be able to represent all the cluster members' tastes, which will negatively affect the estimation accuracy. On the other hand, when a too larger C value is chosen, some of the clusters break down into smaller clusters. As long as the cluster size is not too small, the estimated consensus parameter still fairly represents a consensus of the cluster members' tastes. Hence, we can effectively learn each individual's preferences, borrowing strength from other users within the cluster. In our simulation, given that $C \geq 3$, the accuracy performance is quite stable as long as each cluster has roughly more than 60 members, as shown in the supplementary material.

5.2.2. Recommendation uncertainty quantification

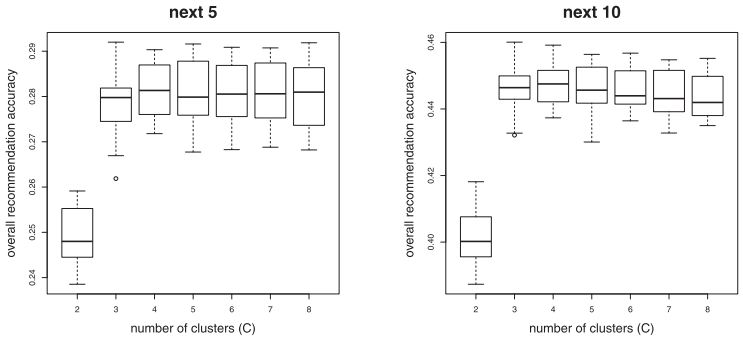
BMCD also estimates the uncertainty associated with each recommendation through the TPP. Such uncertainties can help assess the reliability of the recommendations by predicting the actual "hit rates" of the recommended items. For each user j and each recommended item i, we can use the binary indicator t to indicate whether the recommended item i is truly among user j's next top-k: $t_{ij} = 1$ if $R_{ij} \leq c_j + k$, and 0 otherwise.

At the same time, we can bin the TPPs by putting them into M intervals of equal width. In this case, we choose 0.01 as the bin size. For all TPPs that belong to interval m, the associated indicators t are averaged to $\bar{t}_m$, indicating the average "hit rate" of the recommended items associated with the corresponding level of certainty.

In Fig. 2 we plot $\bar{t}_m$ against the binned TPPs for the next $k = 5$ and $k = 10$ cases. Run number 10 is shown here, but other runs demonstrate similar trends. The blue dotted $x = y$ line indicates perfect calibration. The red dotted line in the figure represents the percentage of correct recommendations made by CF for this run. From the next-5 case, we can clearly observe excellent calibration, especially when the TPPs are in the range

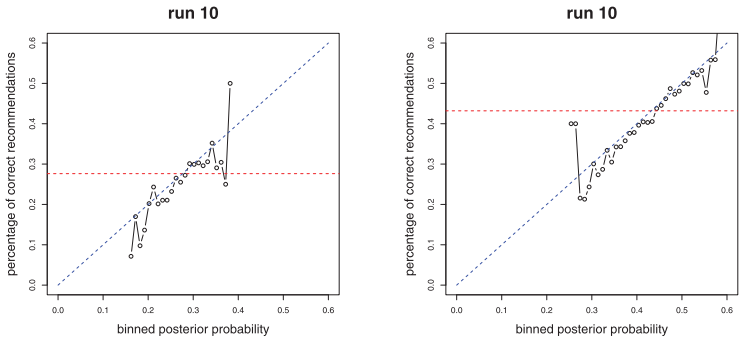
Table 1
Comparison of accuracies of BMCD and CF, summary of 20 runs.

Method	Min	25%	Median	75%	Max	Mean	Std dev
CF Next-5	25.02%	26.59%	27.34%	27.66%	28.65%	27.16%	0.89%
BMCD Next-5	26.69%	27.54%	27.98%	28.26%	29.20%	27.92%	0.69%
CF Next-10	40.54%	41.96%	42.70%	43.64%	44.47%	42.79%	1.05%
BMCD Next-10	43.21%	44.33%	44.64%	45.04%	46.01%	44.67%	0.72%



(a) Recommend next 5 (b) Recommend next 10

Fig. 1. BMCD's recommendation accuracy vs number of clusters chosen.



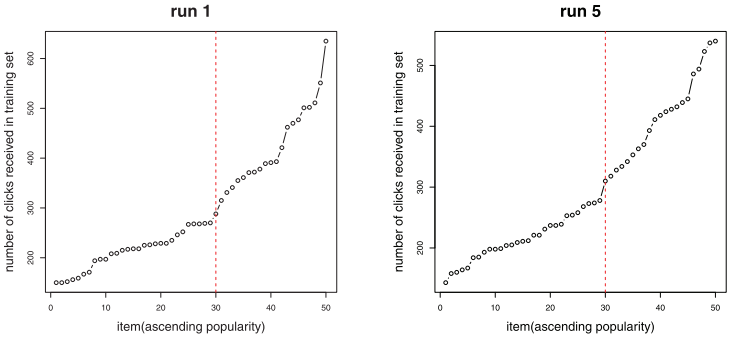
(a) Run 10, next-5 (b) Run 10, next-10

Fig. 2. Percentage of correct recommendations vs. binned TPPs of one selected run. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

between 0.25 and 0.35, where the majority of the recommendations lie within. The uncertainty is not as well calibrated when the TPPs are higher than 0.35 and below 0.25, since there are very few recommendations made with these TPPs. We observe a similar trend in the next-10 case, also in Fig. 2, with overall higher TPPs, and higher accuracy.

The TPP calculations make it possible for BMCD to identify which recommendations are more reliable than others, because the posterior probabilities are precise and interpretable, and hence can be further exploited. CF on the other hand, produces scores useful for ranking the items but are not easily interpretable. One usage of BMCD's TPPs is introducing a nearly calibrated cut off in order to achieve a higher overall recommendation accuracy. That is to say, we can decide to only make recommendations whose posterior probabilities of being in the next top k has surpassed a threshold and can be expected to be at least the threshold value as hit rate. This will inevitably reduce the number of recommendations made to the users, however, overall accuracy can be expected to be higher.

Tables 2 and 3 show how the recommendation accuracies improve when cut off TPPs are used, for the next-5 and next-10 case, respectively. For the next-5 case, it can be observed from Table 2 that all of the recommendations made with BMCD have TPPs above 0.1. Setting a TPP cut off of 0.25 can increase the overall recommendation accuracy by 1.7% points compared to not having a cut off, while retaining more than 70% of the recommendations. Likewise, all TPPs for the next-10 case are above 0.2, but fewer than 100 recommendations have a TPP of 0.6 or higher. When the cut off TPP is set at 0.45, the number of



(a) Run 1 (b) Run 5

Fig. 3. Item popularity of selected runs. The 20 most clicked items are considered as popular.

Table 2
Cut off TPPs and the corresponding recommendation accuracies for predicting next 5 items, summary of 20 runs. CF average accuracy: 27.2%.

Cut off	Number of recommendations	Recommendation accuracy
0.10	15 000 ± 0	28.0 ± 0.3%
0.15	14 997 ± 3.1	28.0 ± 0.3%
0.20	14 136 ± 148.9	28.6 ± 0.3%
0.25	11 196 ± 395.1	29.7 ± 0.3%
0.30	5 425 ± 592.6	31.0 ± 0.4%
0.35	577 ± 211.1	32.4 ± 1.5%
0.40	145 ± 9.3	32.5 ± 8.4%

Table 3
Cut off TPPs and the corresponding recommendation accuracies for predicting next 10 items, summary of 20 runs. CF average: 42.8%.

Cut off	Number of recommendations	Recommendation accuracy
0.20	30 000 ± 0.0	44.6 ± 0.3%
0.25	29 997 ± 1.7	44.6 ± 0.3%
0.30	29 254 ± 144.8	45.0 ± 0.3%
0.35	26 308 ± 460.5	46.4 ± 0.3%
0.40	21 215 ± 694.9	48.0 ± 0.3%
0.45	14 468 ± 762.1	50.0 ± 0.3%
0.50	7 104 ± 800.0	51.0 ± 0.4%
0.55	1 467 ± 438.4	52.0 ± 1.2%
0.60	59 ± 44.0	61.0 ± 8.2%

recommendations are reduced to roughly 50%, while increasing the overall recommendation accuracy by 5.4% points to 50%.

To summarize, BMCD makes recommendations with similar or slightly higher recommendation accuracies compared to CF in this simulation study. Moreover, the posterior probabilities associated with the recommendations are well calibrated and can be further exploited to assess the reliability of the recommendations. Overall recommendation accuracy can be improved by setting a cut off posterior probability.

5.2.3. Diversity

In this section, we assess both CF and BMCD's abilities to fully exploit the item collection, by making novel and diverse recommendations for each user. We will use the four metrics described in Section 4.

Table 4 summarizes the diversity performances of BMCD and CF. It is desirable to have high values of the coverage, correct coverage and novelty metrics, and a low value of intra-list similarity. We see that recommendations made with BMCD are more diverse and novel compared to CF. BMCD outperforms CF especially on

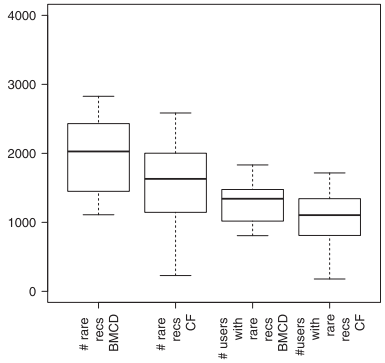


Fig. 4. Comparisons of number of rare item recommendations and number of users with at least 1 recommendations, summary of 20 runs.

the coverage metric, suggesting that BMCD has stronger ability to discover the less popular items.

If we rank all n items according to the number of clicks received by each item (popularity) in the training data in ascending order, and plot the corresponding number of clicks, as shown in Fig. 3, it can be observed that the majority of the clicks are received by a small fraction of items. If we define the 20 most clicked items as "popular", and the rest of the 30 items as less popular, or "rare", we can take a closer look at how often BMCD and CF recommend these "rare" items, and how many users have received at least one such rare recommendation.

From Fig. 4, it can be seen that BMCD recommends many more rare items, and out of $N = 3000$ users, more than 1000 users have received at least one rare recommendation for all runs, outperforming CF in its ability to explore rare items.

Unsurprisingly, BMCD outperforms CF in its diversity performance since the model construction inherently considers diversity by enforcing all clicked items to be top-ranked regardless of how rare these items are. CF on the other hand, is an accuracy-driven method that does not consider diversity as part of its objectives. Instead, the rare items clicked by the users might be sacrificed in the optimization process.

Table 4
Comparison of diversity performances of BMCD and CF.

Metric	Min	25%	Median	75%	Max	Mean
Coverage	CF:0.540 BMCD: 0.680	CF: 0.575 BMCD: 0.720	CF:0.720 BMCD: 0.740	CF: 0.720 BMCD: 0.740	CF:0.760 BMCD: 0.780	CF:0.672 BMCD: 0.731
Corr covg	CF:0.520 BMCD: 0.620	CF: 0.560 BMCD: 0.640	CF:0.660 BMCD: 0.680	CF: 0.680 BMCD: 0.700	CF:0.700 BMCD: 0.720	CF:0.629 BMCD: 0.676
Intra-list similarity	CF:1.92 BMCD: 1.70	CF: 2.05 BMCD: 1.78	CF:2.11 BMCD:1.91	CF: 2.16 BMCD:2.06	CF:2.23 BMCD: 2.11	CF:2.10 BMCD: 1.91
Novelty	CF:5.05 BMCD: 5.14	CF: 5.11 BMCD: 5.15	CF:5.18 BMCD: 5.29	CF: 5.20 BMCD: 5.43	CF:5.23 BMCD: 5.44	CF:5.16 BMCD: 5.29

Table 5
Description of the two NRK training datasets.

Dataset	# of items (n)	# of users (N)	Min clicks	Median clicks	Max clicks	Sparsity
1	200	7872	3	8	93	5.26%
2	200	2143	3	9	83	5.98%

5.3. Case study: A clicking dataset from the Norwegian Broadcasting Corporation (NRK)

In this section, we study a dataset containing anonymous users' clicks on movies, TV-series and news programs that are available on the NRK TV website as well as the apps for mobile phones, tablets and other streaming devices such as AppleTV. The data was collected when no personalized recommendation was implemented. We consider only the 200 most popular items. Here for simplification, a whole season of TV series, or a daily news program (which consists of more than one episode), is considered as one single item. Each user–item click is only recorded once, that is, multiple clicks on one item by one user are treated as one click.

We created two relatively sparse training datasets for model fitting, each with 3 or more clicks per user, as seen in Table 5. They were created by finding the users with 13 (dataset 1) and 23 (dataset 2) or more clicks per user, then randomly removing $k = 10$ (dataset 1) and $k = 20$ clicks per user. These k clicks were hence not part of the training data, but reserved for evaluating the ability of the fitted models to make accurate recommendations. After fitting a model, k recommendations were made by the model, and then these recommendations were compared to the k clicks not part of the training data. The fraction of overlap between the recommendations and the actual k clicks gives the recommendation accuracy. We will also assess both methods' diversity with the metrics introduced in Section 4.

5.3.1. MCMC set up and initialization

First, the number of clusters C needs to be determined. Alternative to the approach shown in Section 5.1, we used K-means clustering on the NRK binary datasets $\{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ with different values of C, and plot the within cluster sum of square against the value of C, see Fig. 5 in the supplement. Combining the elbow method and a preference towards a slightly larger number of clusters, which we showed was important in Section 5.1, $C = 17$ is chosen for dataset 1 and $C = 12$ for dataset 2.

While there are many ways of initializing the MCMC, we use the following procedures in order to achieve faster convergence. To initialize the augmented individual ranking vectors $\tilde{\boldsymbol{R}}_j^0$, we first suppose that all users belong to the same cluster, and estimate very roughly a consensus for all users $\boldsymbol{\rho}^0$ based on item popularity. We obtain $\boldsymbol{\rho}^0$ by ranking the n items according to the number of clicks each of them has received, and randomize the ties if there are any. Next, we initialize the augmented individual ranking vectors $\tilde{\boldsymbol{R}}_j^0$ based on $\boldsymbol{\rho}^0$. First, $\tilde{\boldsymbol{R}}_j^0$ needs to be compatible with the restriction that the clicked items are top-ranked, i.e., $\tilde{R}_{ij}^0 \leq c_j \forall A_i \in \mathcal{A}_j$, and $\tilde{R}_{ij}^0 > c_j \forall A_i \in \mathcal{A}_j^c$. Second, while satisfying this restriction, we want to inherit the pairwise comparisons represented in the group consensus $\boldsymbol{\rho}^0$. That is to say, for each user j,

$$\forall p, q \in \mathcal{A}_j \text{ and } \forall a, b \in \mathcal{A}_j^c$$
$$\tilde{R}_{pj}^0 < \tilde{R}_{qj}^0 \leq c_j, \text{ if } \rho_p^0 < \rho_q^0$$
$$c_j < \tilde{R}_{aj}^0 < \tilde{R}_{bj}^0, \text{ if } \rho_a^0 < \rho_b^0.$$

For example, if we have a 5-item set {A, B, C, D, E}, $\boldsymbol{\rho}^0 = \{1, 2, 3, 4, 5\}$, and user j has clicked on item A, C, and E, the initialization of the augmented vector $\tilde{\boldsymbol{R}}_j^0$ is therefore, {1, 4, 2, 5, 3}. This initialization speeds up the MCMC convergence significantly.

The cluster assignment z_j for $j = 1, \ldots, N$, is initialized randomly. Within each cluster c, $\boldsymbol{\rho}_c^0$ is initialized in a similar manner as $\boldsymbol{\rho}^0$, however, the item popularity is calculated only based on the clicks by the users that belong to cluster c. The parameters $\{\alpha_c^0\}_{c=1,\ldots,C}$ are initialized as $a_1^0 = \cdots = a_C^0 = 3$, other values can also be chosen.

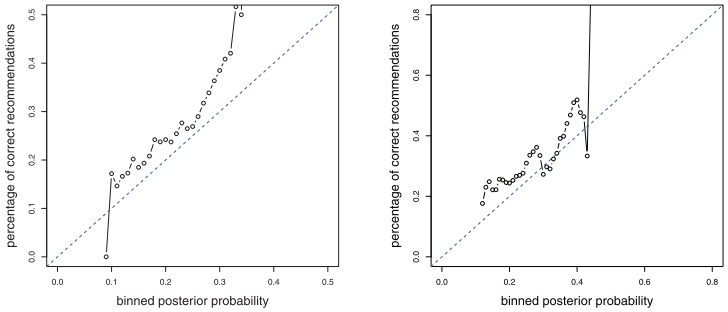
We run the MCMC for 5 million iterations and 7 million iterations, for dataset 1 and dataset 2, respectively. It takes longer for dataset 2 to reach convergence, presumably since there are more users that swing between different clusters. Only the last 1 million iterations are used for subsequent analyses. The MCMC is thinned at every 100 iterations while $\{\alpha_1, \ldots \alpha_C\}$ is proposed every 10 iterations. The trace plots of $\{\alpha_1, \ldots \alpha_C\}$ after the burn-in period are shown in Fig. 4 in the supplement.

Table 6
Comparison of next-k recommendation accuracies for the NRK datasets.

Method	Dataset 1 (next 10)	Dataset 2 (next 20)
BMCD	26.4%	35.0%
CF	29.9%	34.9%

5.3.2. Recommendation accuracy

Table 6 shows the overall recommendation accuracy for predicting the next-k items for both datasets using BMCD and CF respectively. It can be observed that CF in this case outperforms BMCD in terms of accuracy for dataset 1, while for dataset 2, the two methods' accuracies are almost identical. The NRK dataset is collected when no personalized recommendation is rolled out. In this situation, all users' clicks are quite concentrated on the popular items. As will be discussed in more detail in Section 5.3.4, BMCD's tendency to recommend a more diverse set of items and the inclusion of less popular items, compared to CF, presumably contributes to the slightly inferior recommendation accuracy for dataset 1.



(a) Dataset 1 (b) Dataset 2

Fig. 5. Recommendation accuracies vs. binned TPPs for BMCD.

Table 7
Comparisons of coverage and correct coverage.

Dataset	Coverage	Corr coverage
Dataset 1	CF:0.61 (122/200) BMCD: 0.865(173/200)	CF: 0.545 (109/200) BMCD: 0.720(144/200)
Dataset 2	CF: 0.38 (76/200) BMCD: 0.82 (164/200)	CF: 0.280 (56/200) BMCD: 0.685(137/200)

5.3.3. Uncertainty quantification of BMCD recommendations

Similar to Fig. 2, Fig. 5 shows the recommendation accuracy plotted against the binned TPPs. A clearly increasing trend can be observed. BMCD in this case, tends to underestimate the certainty of each recommendation made, or in other words, the TPPs estimated are slightly lower than the actual hit rates of the recommendations as the blue line is slightly above the dotted line. This can be explained by a slight misfit of the Mallows model. We can exploit the uncertainty to identify reliable recommendations as well as introducing cut off TPPs to improve overall accuracy of BMCD.

We see from Fig. 6 that, as the cut off TPP increases, the number of recommendations strictly decreases while the overall recommendation accuracy improves. For dataset 1, when the cut off posterior probability is 0.23 or above, BMCD's overall recommendation accuracy exceeds 30%, making it identical to CF, while retaining 60% of the recommendations.

5.3.4. Diversity

The coverage metric is especially important for NRK. As a national broadcaster, NRK has a large collection of valuable historical contents and non-mainstream programs that may be rarely discovered by its users; however, these programs have high quality and should be promoted.

Table 7 summarizes the comparisons of coverage and correct coverage of BMCD and CF. Both methods cover a broader range of items for dataset 1, as the dataset contains more users, leading to more diverse preferences. Dataset 2 is a more difficult scenario where the users' preferences are more homogeneous, and it is therefore more challenging to make diverse recommendations. It is clear that BMCD outperforms CF in terms of coverage, and the advantage is especially significant for dataset 2. This suggests that, consistent with the simulation, CF tends to recommend more popular items while BMCD has a stronger ability to explore the rare items. In addition, BMCD does not sacrifice much accuracy for diversity, as it also outperforms CF in the correct coverage metric.

Table 8
Comparison of rare items recommendations and number of users with at least 1 rare recommendation.

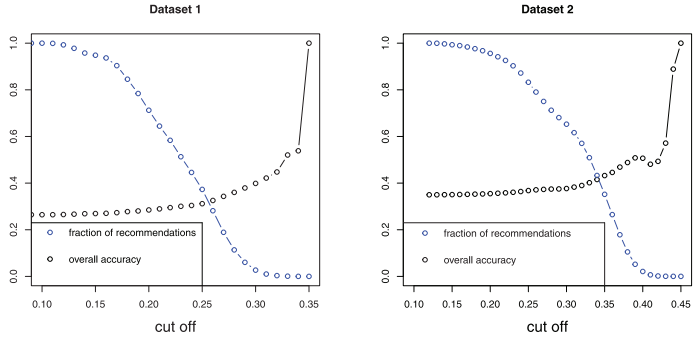
Dataset	# rare recs	# users w. $\geq$1 rare recs
Dataset 1	CF:4388 (5.6%) BMCD: 10071 (12.8%)	CF:929 (11.8%) BMCD: 3454 (43.9%)
Dataset 2	CF: 171 (4.0%) BMCD:8770 (20.5%)	CF: 58 (2.7%) BMCD: 1667 (77.8%)

Table 9
Comparison of intra-list similarity and novelty.

Dataset	Intra-list similarity	Novelty
Dataset 1	CF:11.78 BMCD: 10.75	CF:5.97 BMCD: 6.18
Dataset 2	CF:42.69 BMCD:39.10	CF: 6.31 BMCD: 6.60

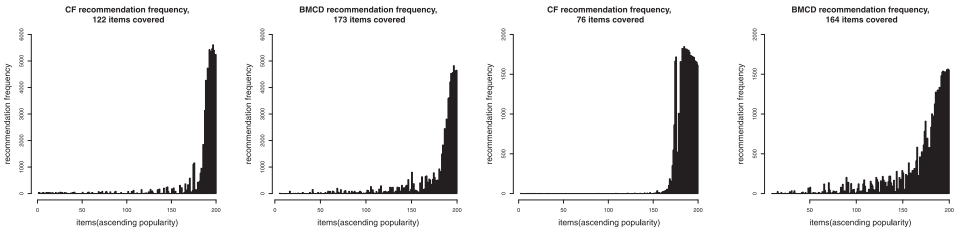
Fig. 7 shows the recommendation frequency of the items being recommended. On the *x*-axis, the items are ranked according to their popularity in ascending order. For both datasets, CF's recommendations are much more concentrated on the more popular items. BMCD in comparison, recommends much fewer popular items compared to CF.

To give a clearer definition of "popular" items, Fig. 8 shows the number of clicks received by each item in the training dataset, with the *x*-axis arranged from the least clicked item to the most clicked item. In both datasets, most of the clicks are attributed to roughly the 40 most popular items, which we define as "popular" items, while the rest we defined as "rare" items. Based on this definition, the number of "rare" items recommended by BMCD and CF, and the number of users receiving at at least one "rare" recommendations are shown in Table 8. It clearly shows that BMCD makes significantly more recommendations that are less popular compared to CF. In particular, for dataset 1, more than 12.8% of all recommendations made with BMCD involves rare items and more than 40% of all users receive at least 1 rare recommendation. For CF, only 5.6% of all recommendations are rare, while 11.8% of the users receive 1 or more rare recommendations. The contrast between CF and BMCD is even more obvious for dataset 2, where only 4% of all recommendations made with CF involves rare items, compared to BMCD's 20.5%. Given that CF's and BMCD's recommendation accuracies are similar in this case, and that BMCD makes more rare recommendations, it follows that BMCD also has a higher recommendation accuracy when recommending popular items compared to CF.



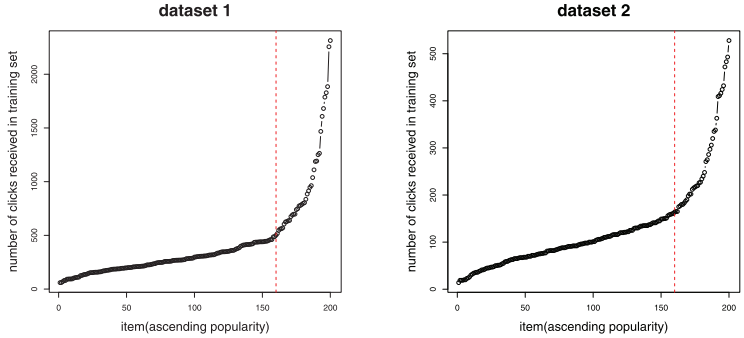
(a) Dataset 1 (b) Dataset 2

Fig. 6. Overall recommendation accuracies and fraction of recommendations performed vs. cut off posterior probabilities, blue line: fraction of recommendations, black line: overall recommendation accuracies. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Dataset 1, CF (b) Dataset 1, BMCD (c) Dataset 2, CF (d) Dataset 2, BMCD

Fig. 7. Histogram of correct recommendations. X-axis: item labels, arranged according to ascending item popularity (measured by the number of clicks received in the training set). Y-axis: recommendation frequency.



(a) Dataset 1 (b) Dataset 2

Fig. 8. Item popularity in the training set. X-axis: item labels, arranged according to ascending item popularity (measured by the number of clicks received in the training set). Y-axis: item popularity.

A comparison of intra-list similarity and novelty is shown in Table 9. Consistent with the simulation, BMCD recommends to each user a list of more diverse items, obtaining a lower intra-list similarity score compared to CF. At the same time, BMCD has a stronger ability to recommend more rare and novel items to the users.

5.4. Accuracy–diversity performances of BMCD and post-processing methods for CF

CF is not intrinsically a diversity-driven method. However, CF's diversity performance can be enhanced through post-processing. It is of interest to see if BMCD's accuracy–diversity performance

is effective compared to CF in combination with such post-processing methods. In this section, we choose two popular post-processing methods to combine with CF, one proposed by Adomavicius et al. [11] and one proposed by Ziegler et al. [8], and compare their accuracy–diversity performances with that of BMCD's. We refer to the two methods as CF+A and CF+Z respectively. We continue with the NRK datasets. Coverage is chosen as the diversity metric in this section, as it is a good indicator of both aggregated diversity of the recommender system, and level of personalization [9].

In Fig. 9, we plot the accuracy–diversity curve for the CF+A (blue) and CF+Z (black) methods, resulting from using a grid of tuning parameters. We can observe a clear trade-off between accuracy and diversity for both post-processing methods. BMCD's accuracy–diversity performance is represented in red and is a single point in the plots. For dataset 1, BMCD clearly outperforms CF+A: while fixing the accuracy level to that of BMCD, illustrated by the vertical red dotted line, BMCD achieves a higher diversity level, and vice versa. BMCD's accuracy–diversity performance is similar to the "elbow" of CF+Z's accuracy–diversity curve, at which point diversity is improved significantly without compromising much accuracy. However, beyond this point, poor choices of the tuning parameter can jeopardize CF+Z's accuracy without making much diversity improvement. BMCD achieves this accuracy–diversity balance without a tuning parameter, while maintaining a slight margin, as shown in the figure that the red point is positioned slightly above the black line. For dataset 2, the users are more similar to each other and therefore, it is a more challenging scenario to achieve diversity. We can clearly observe from Fig. 9(b) that BMCD outperforms both CF+A and CF+Z methods in the combined accuracy–diversity performance.

6. Further discussions and future works

In this paper, we have introduced and applied BMCD to make personalized recommendations based on clicking data. We have also compared the recommendation performances of BMCD with the popular Collaborative Filtering, in terms of accuracy and diversity.

Through a simulation study and an offline testing of a dataset, we have observed that BMCD and CF make recommendations with similar level of accuracy. BMCD, in addition, produces interpretable uncertainty estimation for each recommendation made. We showed that the uncertainty can be further exploited to improve the overall accuracy.

We have also assessed the recommendation diversity of both methods through measures of coverage, correct coverage, intra-list similarity and novelty. We have found that compared to CF, BMCD has a stronger ability to recommend diverse and rare items to the users, and considers more items for recommendations. We have also discovered that BMCD can achieve a similar or higher level of diversity compared to the post-processing methods, when a reasonable level of accuracy is required. BMCD however, does not require tuning, nor does it compromise its accuracy performance for higher diversity.

There are several reasons that explain BMCD's excellent diversity performance. First, BMCD, by construction, follows the restriction that all items clicked by a user, need to be among the user's top-ranked items, regardless of how unusual the clicked items are. This restriction enforces every user's uniqueness, and helps capture and preserve each individual user's "peculiar" behavior. CF on the other hand, often sacrifices the "unusual" items in the matrix factorization process, since the unusual items contribute less to the cost function.

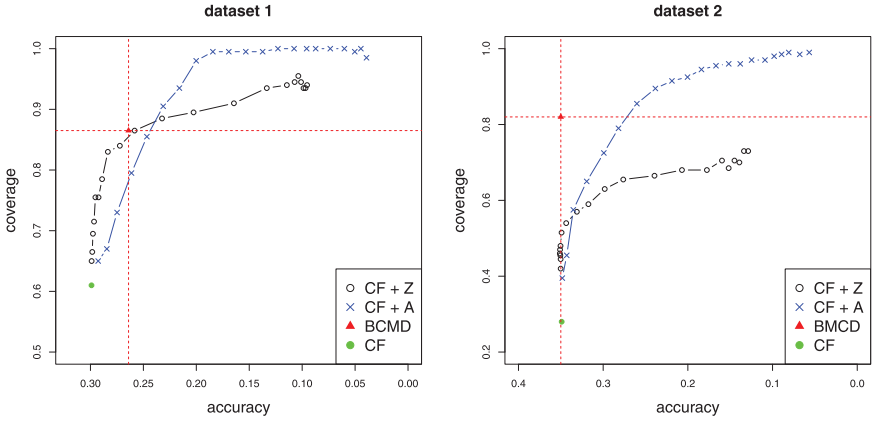
Second, BMCD is sensitive to the clicks in the sparse part of the dataset. BMCD contains the consensus parameter $\boldsymbol{\rho}$, and for the highly sparse part of the dataset, when an item receives a few clicks, these clicks will impact the distribution of the consensus parameter $\boldsymbol{\rho}$, and even more so, certain summary statistics such as the Maximum A Posteriori. The consensus, in turn, has an impact on the distribution and summary statistics of the individual users' latent full ranking vectors $\tilde{\boldsymbol{R}}_j$. However, it can also be a double-edge sword: when the sparse information turns out to be inaccurate or unrepresentative, it can decrease the method's recommendation accuracy.

It is therefore not surprising to observe that recommendations using BMCD are often more diverse, involving more rare items, even when user behaviors are rather homogeneous. BMCD's strong ability to capture the peculiarity of the users and its tendency to recommend less popular items partly explains why it was marginally outperformed in terms of accuracy by CF in the offline testing scenario, where the ground truth is limited by what the users have already seen and clicked. Rare items are often not yet discovered by the users, and it is almost impossible to verify the success of such recommendations in an offline testing. We are currently planning online testing of BMCD.

One of the biggest drawbacks of BMCD, which is based on MCMC, is scaling. BMCD does not scale well due to the huge amount of parameters to be estimated. The computing time required is dependent on the number of users N, the number of clusters C, as well as the number of iterations required to reach convergence. It takes 53 h to compute for 1 million iterations for the NRK dataset 1, and 14 h for dataset 2 using one core of the Intel Xeon e-8890 processor, running at 2.5 GHz. The iterative nature of MCMC also makes efficient parallelization more challenging since the computational overhead is very heavy. The Spark implementation of CF on the other hand, is very efficient. However, it can also be computationally costly if a thorough cross-validation is to be performed. For BMCD, in practice, it often happens that the cluster assignments for each user, $z_1, \ldots, z_N$ converge quite quickly. In the situation that most users do not switch cluster memberships often, after the cluster assignments have converged, we can split the dataset into C different segments, and compute BMCD algorithm without the clustering steps independently and in parallel. The reduction in the number of users, and the number of parameters needed to be estimated, can reduce computing time to at least 1/C of its original required computing time if the C clusters are similar in size. Another way to speed up the computation is by choosing smart starting points for the MCMC such that convergence can be reached in fewer iterations. We suggest, for example, that instead of randomly initializing the cluster assignment for each user, $z_1, \ldots, z_N$ can be initialized based on a K-means clustering. We are currently working on variational Monte Carlo versions of our algorithm, which we expect to reduce computational time very significantly.

Recently, Mao et al. [26] proposed a new approach to multiobjective recommendations using hypergraph rankings, aiming at recommending items that optimize a combination of objective functions, for example, price and quality. Such multiobjective methods could be expanded to a combination of accuracy and diversity. Exploiting social network relations between users, as done in [27], might also be an interesting avenue for further research, for example by giving neighboring users diverse recommendations.

In conclusion, BMCD can be considered as a valid alternative to traditional state-of-the-art collaborative filtering and its post-process enhancements when diversity in the recommendation is an important objective.



(a) Dataset 1 (b) Dataset 2

Fig. 9. Comparison of accuracy–diversity performances of BMCD and post-processing methods combined with CF.

Acknowledgments

We give special thanks to Linn Cecilie Solbergersen and the Norwegian Broadcasting Corporation for generously providing us research data and kind collaborations. We also thank Øystein Sørensen, Elja Arjas and Valeria Vitelli for fruitful discussions.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2019.104960.

References

[1] D.G. Goldstein, D.C. Goldstein, Profiting from the long tail, Harv. Bus. Rev. 84 (6) (2006) 24–28.
[2] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J.R. Wakeling, Y.-C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems, Proc. Natl. Acad. Sci. 107 (10) (2010) 4511–4515.
[3] J.-G. Liu, K. Shi, Q. Guo, Solving the accuracy–diversity dilemma via directed random walks, Phys. Rev. E 85 (1) (2012) 016118.
[4] L. Hou, K. Liu, J. Liu, R. Zhang, Solving the stability–accuracy–diversity dilemma of recommender systems, Physica A 468 (2017) 415–424.
[5] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, IEEE, 2008, pp. 263–272.
[6] Z. Huang, H. Chen, D. Zeng, Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Trans. Inf. Syst. 22 (1) (2004) 116–142.
[7] C.A. Gomez-Uribe, N. Hunt, The netflix recommender system: Algorithms, business value, and innovation, ACM Trans. Manag. Inf. Syst. 6 (4) (2016) 13.
[8] C.-N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International Conference on World Wide Web, ACM, 2005, pp. 22–32.
[9] G. Adomavicius, Y. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 896–911.
[10] V. Vitelli, Ø. Sørensen, M. Crispino, A. Frigessi, E. Arjas, Probabilistic preference learning with the Mallows rank model, J. Mach. Learn. Res. 18 (158) (2018) 1–49.
[11] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. (6) (2005) 734–749.
[12] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285–295.
[13] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 426–434.
[14] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (8) (2009) 30–37.
[15] Y. Koren, R. Bell, Advances in collaborative filtering, in: Recommender Systems Handbook, Springer, 2015, pp. 77–118.
[16] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, J. Mach. Learn. Res. 17 (1) (2016) 1235–1241.
[17] M.Ö. Karakaya, T. Aytekin, Effective methods for increasing aggregate diversity in recommender systems, Knowl. Inf. Syst. 56 (2) (2018) 355–372.
[18] A. Antikacioglu, R. Ravi, Post processing recommender systems for diversity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 707–716.
[19] Q. Liu, M. Crispino, I. Scheel, V. Vitelli, A. Frigessi, Model-based learning from preference data, Annu. Rev. Stat. Appl. (6) (2019).
[20] S. Mukherjee, et al., Estimation in exponential families on permutations, Ann. Statist. 44 (2) (2016) 853–875.
[21] T. Joachims, Optimizing search engines using clickthrough data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 133–142.
[22] S.M. McNee, J. Riedl, J.A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: CHI'06 Extended Abstracts on Human Factors in Computing Systems, ACM, 2006, pp. 1097–1101.
[23] G. Adomavicius, Y. Kwon, Overcoming accuracy–diversity tradeoff in recommender systems: A variance-based approach, in: Proceedings of WITS, Vol. 8, Citeseer, 2008.
[24] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 257–260.
[25] Y.C. Zhang, D.Ó. Séaghdha, D. Quercia, T. Jambor, Auralist: introducing serendipity into music recommendation, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, ACM, 2012, pp. 13–22.
[26] M. Mao, J. Lu, J. Han, G. Zhang, Multiobjective e-commerce recommendations based on hypergraph ranking, Inform. Sci. 471 (2019) 269–287.
[27] M. Mao, J. Lu, G. Zhang, J. Zhang, Multirelational social recommendations via multigraph ranking, IEEE Trans. Cybern. 47 (12) (2016) 4049–4061.

Paper IV

BayesMallows: an R Package for the Bayesian Mallows Model

Øystein Sørensen, Marta Crispino, Qinghua Liu, Valeria Vitelli

DOI: https://doi.org/10.32614/RJ-2020-026

BayesMallows: An R Package for the Bayesian Mallows Model

by Øystein Sørensen, Marta Crispino, Qinghua Liu, and Valeria Vitelli

Abstract **BayesMallows** is an R package for analyzing preference data in the form of rankings with the Mallows rank model, and its finite mixture extension, in a Bayesian framework. The model is grounded on the idea that the probability density of an observed ranking decreases exponentially with the distance to the location parameter. It is the first Bayesian implementation that allows wide choices of distances, and it works well with a large amount of items to be ranked. **BayesMallows** handles non-standard data: partial rankings and pairwise comparisons, even in cases including non-transitive preference patterns. The Bayesian paradigm allows coherent quantification of posterior uncertainties of estimates of any quantity of interest. These posteriors are fully available to the user, and the package comes with convienient tools for summarizing and visualizing the posterior distributions.

Introduction

Preference data are usually collected when individuals are asked to rank a set of items according to a certain preference criterion. The booming of internet-related activities and applications in recent years has led to a rapid increase of the amount of data that have ranks as their natural scale, however often in the form of partial or indirect rankings (pairwise preferences, ratings, clicks). The amount of readily available software handling preference data has consequently increased consistently in the last decade or so, but not many packages are flexible enough to handle all types of data mentioned above. The typical tasks when analyzing preference data are rank aggregation, classification or clustering, and prediction, where the latter task refers to the estimation of the individual rankings of the assessors when completely or partially missing. These tasks can be addressed either via model-based inference or via heuristic machine learning algorithms, with or without uncertainty quantification. However, very few methods allow handling diverse data types, combining several inferential tasks with proper propagation of the uncertainties, while also providing individualized predictions. Our proposal goes exactly in this direction, thus making the scopes of **BayesMallows** broad when it comes to data handling and individual-level inference.

The R package **BayesMallows** is the first software conceived to answer the needs mentioned above in a unified framework: it implements full Bayesian inference for ranking data, and performs all of the tasks above in the framework of the Bayesian Mallows model (BMM) (Mallows, 1957; Vitelli et al., 2018). More specifically, **BayesMallows** allows for data in the forms of complete rankings, partial rankings, as well as pairwise comparisons, including the case where some comparisons are inconsistent. In these situations, it provides all Bayesian inferential tools for rank modeling with the BMM: it performs rank aggregation (estimation of a consensus ranking of the items), it can cluster together the assessors providing similar preferences (estimating both cluster specific model parameters, and individual cluster assignments, with uncertainty), it performs data augmentation for estimating the latent assessor-specific full ranking of the items in all missing data situations (partial rankings, pairwise preferences). The latter in particular, i.e., the possibility of predicting individual preferences for unranked items, enables the model to be used as a probabilistic recommender system. **BayesMallows** also enlarges the pool of distances that can be used in the Mallows model, and it supports the rank distances most used in the literature: Spearman's footrule (henceforth footrule), Spearman's rank correlation (henceforth Spearman), Cayley, Hamming, Kendall, and Ulam distances (we refer to Diaconis, 1988; Marden, 1995, for details on these). Finally, **BayesMallows** implements the Iterative Proportional Fitting Procedure (IPFP) algorithm for computing the partition function for the Mallows model (MM) (Mukherjee, 2016) and the importance sampling algorithm described in Vitelli et al. (2018). In addition to being used in the MM, these functions may be of interest in their own right.

Comparing with other available inferential software, we notice that not many packages allow for such flexibility, very few in combination with full Bayesian inference, and none when using the MM as outlined in Section 4 below. Often machine learning approaches focus on either rank aggregation (i.e., consensus estimation), or individual rank prediction, while **BayesMallows** handles both. Since the BMM is fully Bayesian, all posterior quantities of interest are automatically available from **BayesMallows** for the first time for the MM. In addition, the package also has tools for visualizing posterior distributions, and hence, posterior quantities as well as their associated uncertainties. Uncertainty quantification is often critical in real applications: for recommender systems, the model should not spam the users with very uncertain recommendations; when performing subtype identification for cancer patients, a very uncertain cluster assignment might have serious consequences for the clinical practice, for example in treatment choice. The package also works well with a fairly large number of

items, thanks to computational approximations and efficient programming. In conclusion, **BayesMallows** provides the first fully probabilistic inferential tool for the MM with many different distances. It is flexible in the types of data it handles, and computationally efficient. We therefore think that this package will gain popularity, and prove its usefulness in many practical situations, many of which we probably cannot foresee now.

The paper is organized as follows. The BMM for ranking data is briefly reviewed in Section 2, as well as its model extensions to different data types and to mixtures. Section 3 includes details on the implementation of the inferential procedure. For a thorough discussion of both the model and its implementation we refer interested readers to Vitelli et al. (2018). An overview of existing R packages implementing the Mallows model (MM) is given in Section 4. The use of the **BayesMallows** package is presented, in the form of three case studies, in Sections 5, 6, and 7. Section 8 concludes the paper, also discussing model extensions that will come with new releases of the package.

Background: the Bayesian Mallows model for rankings

In this section we give the background for understanding the functions in the **BayesMallows** package. More details can be found in Vitelli et al. (2018) and Liu et al. (2019). The section is organized as follows: we first clarify the notations that we will use throughout the paper (Section 2.1). We then briefly describe the BMM for complete ranking data (Section 2.2), also focusing on the relevance of the choice of distance (Section 2.3). The last two sections focus on model extensions: partial and pairwise data (Section 2.4), non-transitive pairwise comparisons (Section 2.5), and mixtures (Section 2.6).

Notation

Let us denote with $\mathcal{A} = \{A_1, ..., A_n\}$ the finite set of labeled items to be ranked, and with $\mathcal{P}_n$ the space of n-dimensional permutations. A complete ranking of n items is then a mapping $\boldsymbol{R} : \mathcal{A} \to \mathcal{P}_n$ that attributes a rank $R_i \in \{1, ..., n\}$ to each item, according to some criterion. We here denote a generic complete ranking by $\boldsymbol{R} = (R_1, ..., R_n)$, where R_i is the rank assigned to item A_i. Given a pair of items $\{A_i, A_k\}$, we denote a pairwise preference between them as $(A_i \prec A_k)$, meaning that item A_i is preferred to item A_k. Note the intimate relation that exists between a ranking and pairwise preferences. Given a full ranking $\boldsymbol{R} \in \mathcal{P}_n$, it is immediate to evince all the possible $n(n-1)/2$ pairwise preferences between the items taken in pairs, since the item in the pair having the lower rank is the preferred one:

$$(A_{t_1} \prec A_{t_2}) \iff R_{t_1} < R_{t_2}, \qquad t_1, t_2 = 1, ..., n, \quad t_1 \neq t_2.$$

Conversely, obtaining a full ranking from a set of pairwise preferences is not straightforward. Pairwise preference data are typically incomplete, meaning that not all pairwise preferences necessary to determine each individual ranking are present. They can contain non-transitive patterns, that is, one or more pairwise preferences contradict what is implied by other pairwise preferences. In this package we can handle partial and possibly non-transitive pairwise preferences.

The BMM for Complete Rankings

The MM for ranking data (Mallows, 1957) specifies the probability density for a ranking $\mathbf{r} \in \mathcal{P}_n$ as follows

$$P(\mathbf{r}|\alpha, \boldsymbol{\rho}) = \frac{1}{Z_n(\alpha)} \exp\left[-\frac{\alpha}{n} d(\mathbf{r}, \boldsymbol{\rho})\right] 1_{\mathcal{P}_n}(\mathbf{r}) \tag{1}$$

where $\boldsymbol{\rho} \in \mathcal{P}_n$ is the location parameter representing the consensus ranking, $\alpha \geq 0$ is the scale parameter (precision), $Z_n(\alpha)$ is the normalizing function (or partition function), $d(\cdot, \cdot)$ is a right-invariant distance among rankings (Diaconis, 1988), and $1_{\mathcal{P}_n}(\mathbf{r})$ is an indicator function for the set $\mathcal{P}_n$ which equals one when $\mathbf{r} \in \mathcal{P}_n$ and zero otherwise.

In the complete data case, N assessors have provided complete rankings of the n items in $\mathcal{A}$ according to some criterion, yielding the permutation $\mathbf{R}_j = \left(R_{1j}, \ldots, R_{nj}\right)$ for assessor j, $j = 1, \ldots, N$. The likelihood of the N observed rankings $\mathbf{R}_1, \ldots, \mathbf{R}_N$, assumed conditionally independent given α and $\boldsymbol{\rho}$, is

$$P(\mathbf{R}_1, \ldots, \mathbf{R}_N|\alpha, \boldsymbol{\rho}) = \frac{1}{Z_n(\alpha)^N} \exp\left[-\frac{\alpha}{n} \sum_{j=1}^{N} d\left(\mathbf{R}_j, \boldsymbol{\rho}\right)\right] \prod_{j=1}^{N} 1_{\mathcal{P}_n}\left(\mathbf{R}_j\right). \tag{2}$$

According to the BMM introduced in Vitelli et al. (2018), prior distributions have to be elicited on

every parameter of interest. A truncated exponential prior distribution was specified for α

$$\pi(\alpha|\lambda) = \frac{\lambda \exp(-\lambda\alpha) 1_{[0,\alpha_{\max}]}(\alpha)}{1 - \exp(-\lambda\alpha_{\max})}, \tag{3}$$

where λ is a rate parameter, small enough to ensure good prior dispersion, and $\alpha_{\max}$ is a cutoff, large enough to cover reasonable α values. A uniform prior $\pi(\boldsymbol{\rho})$ on $\mathcal{P}_n$ was assumed for $\boldsymbol{\rho}$. It follows that the posterior distribution for α and $\boldsymbol{\rho}$ is

$$P(\alpha, \boldsymbol{\rho}|\mathbf{R}_1, \ldots, \mathbf{R}_N) \propto \frac{1_{\mathcal{P}_n}(\boldsymbol{\rho})}{Z_n(\alpha)^N} \exp\left[-\frac{\alpha}{n}\sum_{j=1}^{N} d\left(\mathbf{R}_j, \boldsymbol{\rho}\right) - \lambda\alpha\right] 1_{[0,\alpha_{\max}]}(\alpha). \tag{4}$$

Inference on the model parameters is based on a Metropolis-Hastings (M-H) Markov Chain Monte Carlo (MCMC) algorithm, described in Vitelli et al. (2018). Some details relevant for a correct use of this package are also given in Section 3.1.

Distance measures and partition function

The partition function $Z_n(\alpha)$ in (1), (2) and (4) does not depend on the latent consensus ranking $\boldsymbol{\rho}$ when the distance $d(\cdot,\cdot)$ is right-invariant, meaning that it is unaffected by a relabelling of the items (Diaconis, 1988). Right-invariant distances play an important role in the MM, and **BayesMallows** only handles right-invariant distances. The choice of distance affects the model fit to the data and the results of the analysis, and is crucial also because of its role in the partition function computation. Some right-invariant distances allow for analytical computation of the partition function, and for this reason they have become quite popular. In particular, the MM with Kendall (Meilă and Chen, 2010; Lu and Boutilier, 2014), Hamming (Irurozki et al., 2014) and Cayley (Irurozki et al., 2016b) distances have a closed form of $Z_n(\alpha)$ due to Fligner and Verducci (1986). There are however important and natural right-invariant distances for which the computation of the partition function is NP-hard, in particular the footrule (l_1) and the Spearman (l_2) distances. For precise definitions of all distances involved in the MM we refer to Marden (1995). **BayesMallows** handles the footrule, Spearman, Cayley, Hamming, Kendall, and Ulam distances.

Partial rankings and transitive pairwise comparisons

When complete rankings of all items are not readily available, the BMM can still be used by applying data augmentation techniques. Partial rankings can occur because ranks are missing at random, because the assessors have only ranked their top-k items, or because they have been presented with a subset of items. In more complex situations, data do not include ranks at all, but the assessors have only compared pairs of items and given a preference between the two. The Bayesian data augmentation scheme can still be used to handle such pairwise comparison data, thus providing a solution that is fully integrated into the Bayesian inferential framework. The following paragraphs provide a summary of Sections 4.1 and 4.2 of Vitelli et al. (2018), which we refer to for details.

Let us start by considering the case of top-k rankings. Suppose that each assessor j has chosen a set of preferred items $\mathcal{A}_j \subseteq \mathcal{A}$, which were given ranks from 1 to $n_j = |\mathcal{A}_j|$. Now $R_{ij} \in \{1, \ldots, n_j\}$ if $A_i \in \mathcal{A}_j$, while for $A_i \in \mathcal{A}_j^c$, R_{ij} is unknown, except for the constraint $R_{ij} > n_j$, $j = 1, \ldots, N$. The augmented data vectors $\tilde{\mathbf{R}}_1, \ldots, \tilde{\mathbf{R}}_N$ are introduced in the model to include the missing ranks, which are estimated as latent parameters. Let $\mathcal{S}_j = \{\tilde{\mathbf{R}}_j \in \mathcal{P}_n : \tilde{R}_{ij} = R_{ij} \text{ if } A_i \in \mathcal{A}_j\}$, $j = 1, \ldots, N$ be the set of possible augmented random vectors, including the ranks of the observed top-n_j items together with the unobserved ranks, which are assigned a uniform prior on the permutations of $\{n_j + 1, \ldots, n\}$. The goal is to sample from the posterior distribution

$$P(\alpha, \boldsymbol{\rho}|\mathbf{R}_1, \ldots, \mathbf{R}_N) = \sum_{\tilde{\mathbf{R}}_1 \in \mathcal{S}_1} \cdots \sum_{\tilde{\mathbf{R}}_N \in \mathcal{S}_N} P(\alpha, \boldsymbol{\rho}, \tilde{\mathbf{R}}_1, \ldots, \tilde{\mathbf{R}}_N|\mathbf{R}_1, \ldots, \mathbf{R}_N). \tag{5}$$

The augmentation scheme amounts to alternating between sampling α and $\boldsymbol{\rho}$ given the current values of the augmented ranks using the posterior given in (4), and sampling the augmented ranks given the current values of α and $\boldsymbol{\rho}$. For the latter task, once α, $\boldsymbol{\rho}$ and the observed ranks $\mathbf{R}_1, \ldots, \mathbf{R}_N$ are fixed, one can see that $\tilde{\mathbf{R}}_1, \ldots, \tilde{\mathbf{R}}_N$ are conditionally independent, and that each $\tilde{\mathbf{R}}_j$ only depends on the corresponding $\mathbf{R}_j$. As a consequence, the update of new augmented vectors is performed independently, for each $j = 1, \ldots, N$.

The above procedure can also handle more general situations where missing rankings are not necessarily the bottom ones, and where each assessor is asked to provide the mutual ranking of some

possibly random subset $\mathcal{A}_j \subseteq \mathcal{A}$ consisting of $n_j \leq n$ items. Note that the only difference from the previous formulation is that each latent rank vector $\tilde{\mathbf{R}}_j$ takes values in the set

$$\mathcal{S}_j = \left\{ \tilde{\mathbf{R}}_j \in \mathcal{P}_n : \left(R_{i_1 j} < R_{i_2 j} \right) \wedge \left(A_{i_1}, A_{i_2} \in \mathcal{A}_j \right) \Rightarrow \tilde{R}_{i_1 j} < \tilde{R}_{i_2 j} \right\}.$$

Also in this case the prior for $\tilde{\mathbf{R}}_j$ is assumed uniform on $\mathcal{S}_j$.

In the case of pairwise comparison data, let us call $\mathcal{B}_j$ the set of all pairwise preferences stated by assessor j, and let $\mathcal{A}_j$ be the set of items appearing at least once in $\mathcal{B}_j$. Note that the items in $\mathcal{A}_j$ are not necessarily fixed to a given rank but may only be given some partial ordering. For the time being, we assume that the observed pairwise orderings in $\mathcal{B}_j$ are transitive, i.e., mutually compatible, and define by $\text{tc}\left(\mathcal{B}_j\right)$ the transitive closure of $\mathcal{B}_j$, which contains all pairwise orderings of the elements in $\mathcal{A}_j$ induced by $\mathcal{B}_j$. The model formulation remains the same as in the case of partial rankings, with the prior for the augmented data vectors $\tilde{\mathbf{R}}_1, \ldots, \tilde{\mathbf{R}}_N$ being uniform on the set $\mathcal{S}_j$ of rankings that are compatible with the observed data.

Non-transitive pairwise comparisons

It can happen in real applications that individual pairwise comparison data are non-transitive, that is, they may contain a pattern of the form $x \prec y$, $y \prec z$ but $z \prec x$. This is typically the case of data collected from internet user activities, when the pool of items is very large: non-transitive patterns can arise for instance due to assessors' inattentiveness, uncertainty in their preferences, and actual confusion, even when one specific criterion for ranking is used. Another frequent situation is when the number of items is not very large, but the items are perceived as very similar by the assessors. This setting is discussed in Crispino et al. (2019), where the model for transitive pairwise comparisons of Section 2.4 is generalized to handle situations where non-transitivities in the data occur. Note that the kind of non-transitivity that is considered in Crispino et al. (2019) considers only the individual level preferences. A different type of non-transitivity, which we do not consider here, arises when aggregating preferences across assessors, as under Condorcet (Marquis of Condorcet, 1785) or Borda (de Borda, 1781) voting rules.

The key ingredient of this generalization consists of adding one layer of latent variables to the model hierarchy, accounting for the fact that assessors can make mistakes. The main assumption is that the assessor makes pairwise comparisons based on her latent full rankings $\tilde{\boldsymbol{R}}$. A mistake is defined as an inconsistency between one of the assessor's pairwise comparisons and $\tilde{\boldsymbol{R}}$. Suppose each assessor $j = 1, \ldots, N$ has assessed M_j pairwise comparisons, collected in the set $\mathcal{B}_j$, and assume the existence of latent ranking vectors $\tilde{\boldsymbol{R}}_j$, $j = 1, \ldots, N$. Differently from Section 2.4, since $\mathcal{B}_j$ is allowed to contain non-transitive pairwise preferences, the transitive closure of $\mathcal{B}_j$ is not defined, and the posterior density (5) cannot be evaluated. In this case, the posterior takes the form,

$$\begin{aligned} P\left(\alpha, \boldsymbol{\rho} | \mathcal{B}_1, ..., \mathcal{B}_N\right) &= \sum_{\tilde{\boldsymbol{R}}_1 \in \mathcal{P}_n} \cdots \sum_{\tilde{\boldsymbol{R}}_N \in \mathcal{P}_n} P\left(\alpha, \boldsymbol{\rho}, \tilde{\boldsymbol{R}}_1, ..., \tilde{\boldsymbol{R}}_N | \mathcal{B}_1, ..., \mathcal{B}_N\right) = \\ &= \sum_{\tilde{\boldsymbol{R}}_1 \in \mathcal{P}_n} \cdots \sum_{\tilde{\boldsymbol{R}}_N \in \mathcal{P}_n} P(\alpha, \boldsymbol{\rho} | \tilde{\boldsymbol{R}}_1, ..., \tilde{\boldsymbol{R}}_N) P\left(\tilde{\boldsymbol{R}}_1, ..., \tilde{\boldsymbol{R}}_N | \mathcal{B}_1, ..., \mathcal{B}_N\right) \end{aligned} \tag{6}$$

where the term $P\left(\tilde{\boldsymbol{R}}_1, ..., \tilde{\boldsymbol{R}}_N | \mathcal{B}_1, ..., \mathcal{B}_N\right)$ models the presence of mistakes in the data, while in the case of transitive pair comparisons it was implicitly assumed equal to 1 if each augmented ranking $\tilde{\boldsymbol{R}}_j$ was compatible with the partial information contained in $\mathcal{B}_j$, and 0 otherwise.

Two models for (6) are considered in Crispino et al. (2019): the Bernoulli model, which accounts for random mistakes, and the Logistic model, which lets the probability of making a mistake depend on the similarity of the items being compared. The Bernoulli model states that:

$$P\left(\tilde{\boldsymbol{R}}_1, ..., \tilde{\boldsymbol{R}}_N | \theta, \mathcal{B}_1, ..., \mathcal{B}_N\right) \propto \theta^M \left(1 - \theta\right)^{\sum_j M_j - M}, \quad \theta \in [0, 0.5) \tag{7}$$

where M counts the number of times the observed preferences contradict what is implied by the ranking $\tilde{\boldsymbol{R}}_j$, M_j is the number of pairwise comparisons reported by assessor j, and the parameter θ is the probability of making a mistake in a single pairwise preference. θ is *a priori* assigned a truncated Beta distribution on the interval $[0, 0.5)$ with given hyper-parameters κ_1 and κ_2, conjugate to the Bernoulli model (7). The Logistic model is a generalization of (7) where, instead of assigning a constant value θ to the probability of making a mistake, it depends on the distance between the ranks of the two items under comparison. In Crispino et al. (2019) the Logistic model gave results very similar to the Bernoulli model, and currently only the Bernoulli model is available in **BayesMallows**. The sampling scheme is similar to the one used for the case of transitive pairwise preferences, apart from an additional step for updating θ, and the augmentation scheme for $\tilde{\boldsymbol{R}}_j$, which is slightly different. We

refer to Crispino et al. (2019) for details.

Clustering

The assumption, implicit in the discussion so far, that there exists a unique consensus ranking shared by all assessors is unrealistic in most real applications: the BMM thus handles the case where the rankings provided by all assessors can be modeled as a finite mixture of MMs. In the following brief discussion we assume that the data consist of complete rankings, but **BayesMallows** can fit a mixture based on any kinds of preference data described so far. See Section 4.3 of Vitelli et al. (2018) for details.

Let $z_1, \dots, z_N \in \{1, \dots, C\}$ assign each assessor to one of C clusters, and let the rankings within each cluster $c \in \{1, \dots, C\}$ be described by an MM with parameters α_c and $\boldsymbol{\rho}_c$. The likelihood of the observed rankings $\mathbf{R}_1, \dots, \mathbf{R}_N$ is given by

$$P\left(\mathbf{R}_1, \dots, \mathbf{R}_N \middle| \{\boldsymbol{\rho}_c, \alpha_c\}_{c=1,\dots,C}, \left\{z_j\right\}_{j=1,\dots,N}\right) = \prod_{j=1}^{N} \frac{1_{\mathcal{P}_n}\left(\mathbf{R}_j\right)}{Z_n\left(\alpha_{z_j}\right)} \exp\left[-\frac{\alpha_{z_j}}{n} d\left(\mathbf{R}_j, \boldsymbol{\rho}_{z_j}\right)\right],$$

where conditional independence is assumed across the clusters. We also assume independent truncated exponential priors for the scale parameters and independent uniform priors for the consensus rankings. The cluster labels $z_1, \dots, z_N$ are a priori assumed conditionally independent given the clusters mixing parameters $\tau_1, ..., \tau_C$, and are assigned a uniform multinomial. Finally $\tau_1, \dots, \tau_C$ (with $\tau_c \geq 0$, $c = 1, \dots, C$ and $\sum_{c=1}^{C} \tau_c = 1$) are assigned the standard symmetric Dirichlet prior of parameter Ψ, thus implying a conjugate scheme. The posterior density is then given by

$$P\left(\{\boldsymbol{\rho}_c, \alpha_c, \tau_c\}_{c=1}^{C}, \left\{z_j\right\}_{j=1}^{N} \middle| \mathbf{R}_1, \dots, \mathbf{R}_N\right) \propto \left[\prod_{c=1}^{C} e^{-\lambda\alpha_c} \tau_c^{\Psi-1}\right] \left[\prod_{j=1}^{N} \frac{\tau_{z_j} e^{-\frac{\alpha_{z_j}}{n} d\left(\mathbf{R}_j, \boldsymbol{\rho}_{z_j}\right)}}{Z_n\left(\alpha_{z_j}\right)}\right]. \tag{8}$$

Computational considerations

In this section we briefly give some additional details regarding the implementation of the models described in Section 2. The BMM implementation is thoroughly described in Vitelli et al. (2018).

Details on the MCMC procedures

In order to obtain samples from the posterior density of equation (4), **BayesMallows** implements an MCMC scheme iterating between (i) updating $\boldsymbol{\rho}$ and (ii) updating α (Algorithm 1 of Vitelli et al., 2018). The leap-and-shift proposal distribution, which is basically a random local perturbation of width L of a given ranking, is used for updating $\boldsymbol{\rho}$ in step (i). The L parameter of the leap-and-shift proposal controls how far the proposed ranking is from the current one, and it is therefore linked to the acceptance rate. The recommendation given in Vitelli et al. (2018) is to set it to $L = n/5$, which is also the default value in **BayesMallows**, but the user is allowed to choose a different value. For updating α in step (ii), a log-normal density is used as proposal, and its variance σ_α^2 can be tuned to obtain a desired acceptance rate.

As mentioned in Section 2.4, the MCMC procedure for sampling from the posterior densities corresponding to the partial data cases (Algorithm 3 of Vitelli et al., 2018) has an additional M-H step to account for the update of the augmented data rankings $\left\{\tilde{\mathbf{R}}_j\right\}_{j=1}^{N}$. In the case of partial rankings, for updating the augmented data $\tilde{\mathbf{R}}_j$, $j = 1, ..., N$ we use a uniform proposal on the set of rankings compatible with the partial data, $\mathcal{S}_j$. In the case of pairwise preferences, due to the increased sparsity in the data, we instead implemented a modified parameter-free leap-and-shift proposal distribution, which proposes a new augmented ranking by locally permuting the ranks in $\tilde{\mathbf{R}}_j$ within the constraints given by $\mathcal{B}_j$ (Vitelli et al., 2018, Section 4.2). The generalization to non-transitive pairwise comparisons, outlined in Section 4 of Crispino et al. (2019), requires further considerations. First, in the M-H step for updating the augmented data rankings, the modified parameter-free leap-and-shift proposal has to be replaced by a Swap proposal, whose tuning parameter $L^\star$ is the maximum allowed distance between the ranks of the swapped items. Second, the Bernoulli model for mistakes makes it necessary to add a a Gibbs step for the update of θ. The MCMC algorithm for sampling from the mixture model posterior (8) (Algorithm 2 of Vitelli et al., 2018) alternates between updating $\{\boldsymbol{\rho}_c, \alpha_c\}_{c=1}^{C}$ in an M-H step, and $\left\{\tau_c, z_j\right\}_{c=1,j=1}^{C,N}$ in a Gibbs sampling step, in addition to the necessary M-H steps for data

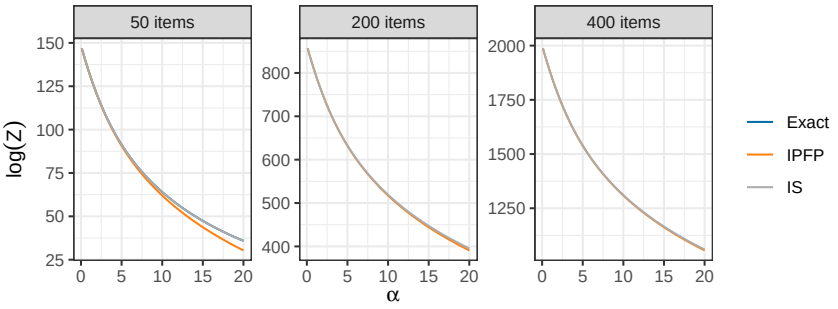


Figure 1: Estimates of the log partition function for the BMM with footrule distance computed using exact calculation (only in the case of 50 items), asymptotic estimation with the IPFP algorithm, and Monte Carlo simulation with the IS algorithm. Exact and IS estimates are perfectly overlapping in the case of 50 items. The bias of the IPFP estimates decreases when the number of items increases.

augmentation or estimation of error models, as outlined above.

Partition Function

When the distance in the BMM is footrule or Spearman, the partition function $Z_n(\cdot)$ does not have a closed form. In these situations **BayesMallows** allows for three different choices, which the user may employ depending on the value of n: (a) exact calculation, (b) Importance Sampling (IS), and (c) the asymptotic approximation due to Mukherjee (2016).

The package contains integer sequences for exact calculation of the partition function with footrule distance for up to $n = 50$ items, and with the Spearman distance for up to $n = 14$ items (see Vitelli et al., 2018, Section 2.1). These sequences are downloaded from the On-Line Encyclopedia of Integer Sequences (Sloane, 2017).

The IS procedure can be used to compute an off-line approximation $\hat{Z}_n(\alpha)$ of $Z_n(\alpha)$ for the specific value of n which is needed in the application at hand. The IS estimate $\hat{Z}_n(\alpha)$ is computed on a grid of α values provided by the user, and then a smooth estimate obtained via a polynomial fit is returned to the user, who can also select the degree of the polynomial function. Finally, the user should set the number K of IS iterations, and we refer to Vitelli et al. (2018) for guidelines on how to select a large enough value for K. The procedure might be time-consuming, depending on K, n, and on how the grid for α is specified. In our experience, values of n larger than approximately 100 might require K to be as large as 10^8 in order for the IS to provide a good estimate, and hence a long computing time.

The IPFP algorithm (Mukherjee, 2016, Theorem 1.8) yields a numeric evaluation of $Z_{\text{lim}}(\cdot)$, the asymptotic approximation to $Z_n(\cdot)$. In this case the user needs to specify two parameters: the number of iterations m to use in the IPFP, and the number of grid points K of the grid approximating the continuous domain where the limit is computed. Values of m and K have been suggested by Mukherjee (2016), and we refer to the Supplementary Material of Vitelli et al. (2018) for more details.

A simulation experiment was conducted comparing the methods for estimating $\log(Z(\alpha))$ with footrule distance for $\alpha = 0.1, 0.2, \ldots, 20$. Let $\log(Z(\alpha))^K$ denote the IS estimate obtained with K iterations, and define the absolute relative difference between two IS estimates obtained with K_2 and K_1 iterations as

$$\rho(K_2, K_1) = \max_\alpha \left\{ \frac{\left|\log(Z(\alpha))^{K_2} - \log(Z(\alpha))^{K_1}\right|}{\left|\log(Z(\alpha))^{K_1}\right|} \right\}.$$

The IS algorithm was run with 10^5, 10^6, and 10^7 iterations, and we obtained $\rho(10^6, 10^5) < 0.3\%$ and $\rho(10^7, 10^6) < 0.15\%$, suggesting that using $K = 10^7$ yields low Monte Carlo variation over this range of α. The IPFP algorithm was used to estimate $\log(Z_{\text{lim}}(\alpha))$, with $K = 10^3$ and $m = 10^3$. The results are summarized in Figure 1, in which also exact computation is included in the case of 50 items. With 50 items, the IS estimate perfectly overlaps the exact estimate, while the asymptotic estimate has a bias that increases with increasing α. As expected, the bias of the asymptotic estimate decreases when the number of items increases, as this brings it closer to the asymptotic limit.

Sampling from the Bayesian Mallows Model

To obtain random samples from the MM with Cayley, Hamming, Kendall, or Ulam distance and fixed α and $\boldsymbol{\rho}$, we suggest using the **PerMallows** (Irurozki et al., 2016a) package, which is optimized for this task. We instead provide a procedure for sampling from the MM with footrule and Spearman. The procedure to generate a random sample of size N from the MM is straightforward, and described in Appendix C of Vitelli et al. (2018). Basically we run a Metropolis-Hastings algorithm with fixed $\boldsymbol{\rho}$ and α and accept/reject the sample based on its acceptance probability. We then take N rankings with a large enough interval between each of them to achieve independence.

Packages implementing the Mallows model

This section gives an overview of existing packages for fitting the MM.

- **PerMallows** is the package that comes closest in functionality to **BayesMallows**. It contains functions for learning and sampling from the frequentist versions of the MM and generalized Mallows model (GMM) (Fligner and Verducci, 1986). Compared to **BayesMallows**, it lacks support for footrule or Spearman distance, it is not Bayesian, and does not compute uncertainty ranges for the estimated parameters. In addition **PerMallows** handles only complete rankings, and does not provide functionality for computation of mixture models. According to Irurozki et al. (2016a, Table 1), computing the maximum likelihood estimates (MLE) of α and ρ using the function `lmm` is possible when $n < 80$ for Kendall, $n < 250$ for Cayley, $n < 90$ for Hamming and $n < 100$ for Ulam. Our experiments suggest that these estimates are conservative, and that even larger numbers of items are fit rapidly. Hence, **PerMallows** seems to be a good choice for modeling with complete data without clusters, when the supported distance measures are appropriate and uncertainty estimates are not sought. **PerMallows** also has very efficient functions for sampling from the MM with Cayley, Hamming, Kendall, and Ulam distances.
- **pmr** (Lee and Yu, 2013) provides summary statistics, visualization, and model fitting tools for complete ranking data in the MM, as well as other models. The function `dbm` returns the MLE of α together with its variance. The MLE of ρ, however, is not returned, but printed to the console, and no uncertainty estimates are given. Internally, `dbm` generates a matrix of size $n! \times n$ containing all possible permutations of the n items. As a result, it quickly runs into memory issues. In our tests, `pmr` was not able to handle a ranking dataset with $n = 10$ items.
- **rankdist** (Qian, 2018) implements distance-based probability models for ranking data as described in Alvo and Yu (2014), returning MLEs for α and ρ, but no uncertainty estimates. The package handles a large number of distances and supports mixture models, but in our experiments a warning was issued when using mixtures with all distances except Kendall. **rankdist** also implements the GMM (Fligner and Verducci, 1986). However, for Cayley, footrule, Hamming, and Spearman distances, it generates an $n! \times n$ matrix internally, causing our R session to crash with $n \geq 10$ items, hence limiting its applicability. For Kendall, on the other hand, **rankdist** appears to work fine both with a large number of items, and with mixtures.

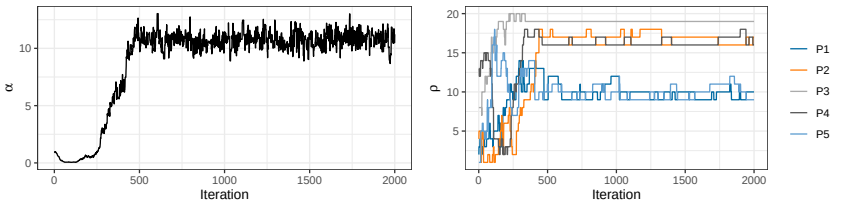
BayesMallows provides many new functionalities not implemented in these packages, as will be illustrated in the use cases of the following three sections.

Analysis of complete rankings with BayesMallows

We illustrate the case of complete rankings with the potato datasets described in Liu et al. (2019, Section 4). In short, a bag of 20 potatoes was bought, and 12 assessors were asked to rank the potatoes by weight, first by visual inspection, and next by holding the potatoes in hand. These datasets are available in **BayesMallows** as matrices with names `potato_weighing` and `potato_visual`, respectively. The true ranking of the potatoes' weights is available in the vector `potato_true_ranking`. In general, `compute_mallows` expects ranking datasets to have one row for each assessor and one column for each item. Each row has to be a proper permutation, possibly with missing values. We are interested in the posterior distribution of both the level of agreement between assessors, as described by α, and in the latent ranking of the potatoes, as described by $\boldsymbol{\rho}$. We refer to the attached replication script for random number seeds for exact reproducibility.

First, we do a test run to check convergence of the MCMC algorithm, and then get trace plots with `assess_convergence`.

```
bmm_test <- compute_mallows(potato_visual)
assess_convergence(bmm_test)
```



(a) Trace of α. **(b)** Trace of ρ.

Figure 2: Trace plots of α and ρ for the MCMC algorithm with the `potato_visual` dataset. The plots indicating good mixing for both parameters after about 500 iterations.

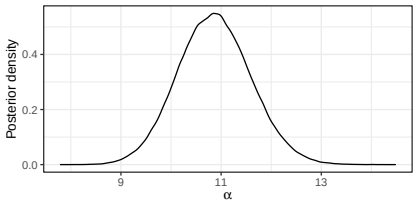


Figure 3: Posterior distribution of α with the `potato_visual` dataset. The posterior mass is symmetrically centered between 9 and 13, with a mean around 11.

By default, `assess_convergence` returns a trace plot for α, shown in Figure 2a. The algorithm seems to be mixing well after around 500 iterations. Next, we study the convergence of $\boldsymbol{\rho}$. To avoid overly complex plots, we pick potatoes $1-5$ by specifying this in the `items` argument.

```
assess_convergence(bmm_test, parameter = "rho", items = 1:5)
```

The corresponding plot is shown in Figure 2b, illustrating that the MCMC algorithm seems to have converged after around 1,000 iterations.

From the trace plots, we decide to discard the first 1,000 MCMC samples as burn-in. We rerun the algorithm to get 500,000 samples after burn-in. The object `bmm_visual` has `S3` class "BayesMallows", so we plot the posterior distribution of α with `plot.BayesMallows`.

```
bmm_visual <- compute_mallows(potato_visual, nmc = 501000)
bmm_visual$burnin <- 1000 # Set burn-in to 1000
plot(bmm_visual) # Use S3 method for plotting
```

The plot is shown in Figure 3. We can also get posterior credible intervals for α using `compute_posterior_intervals`, which returns both highest posterior density intervals (HPDI) and central intervals in a `tibble` (Müller and Wickham, 2018). **BayesMallows** uses `tibbles` rather than `data.frames`, but both are accepted as function inputs. We refer to `tibbles` as dataframes henceforth.

```
compute_posterior_intervals(bmm_visual, decimals = 1L)

# A tibble: 1 x 6
  parameter  mean median conf_level hpdi       central_interval
  <ch>    <dbl>  <dbl> <ch>     <ch>     <chr>
1 alpha      10.9   10.9 95 %       [9.4,12.3] [9.5,12.3]
```

Next, we can go on to study the posterior distribution of $\boldsymbol{\rho}$.

```
plot(bmm_visual, parameter = "rho", items = 1:20)
```

If the `items` argument is not provided, and the number of items exceeds five, five items are picked at random for plotting. To show all potatoes, we explicitly set `items = 1:20`. The corresponding plots are shown in Figure 4.

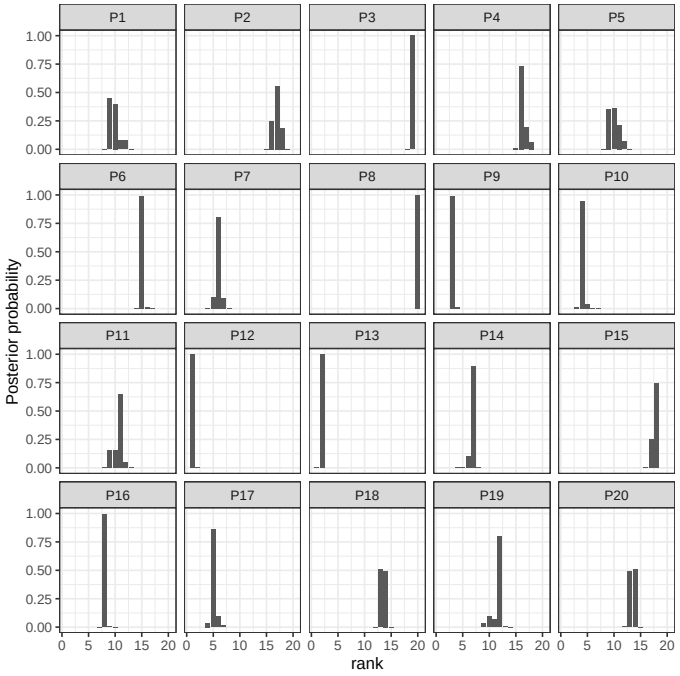


Figure 4: Posterior distribution of latent ranks ρ with the `potato_visual` dataset. Most potatoes have highly peaked posterior distributions, indicating low uncertainty about their ranking.

Jumping over the scale parameter

Updating α in every step of the MCMC algorithm may not be necessary, as the number of posterior samples typically is more than large enough to obtain good estimates of its posterior distribution. With the `alpha_jump` argument, we can tell the MCMC algorithm to update α only every `alpha_jump`-th iteration. To update α every 10th update of ρ, we do

```
bmm <- compute_mallows(potato_visual, nmc = 501000, alpha_jump = 10)
```

On a MacBook Pro 2.2 GHz Intel Core i7 running R version 3.5.1, the above call ran in 2.0 seconds on average over 1,000 replications using **microbenchmark** (Mersmann, 2018), while it took 4.2 seconds using the default value `alpha_jump = 1`, i.e., updating α less frequently more than halved the computing time.

Other distance metrics

By default, `compute_mallows` uses the footrule distance, but the user can also choose to use Cayley, Kendall, Hamming, Spearman, or Ulam distance. Running the same analysis of the potato data with Spearman distance is done with the command

```
bmm <- compute_mallows(potato_visual, metric = "spearman", nmc = 501000)
```

For the particular case of Spearman distance, **BayesMallows** only has integer sequences for computing the exact partition function with 14 or fewer items. In this case a precomputed importance sampling estimate is part of the package, and used instead.

Analysis of preference data with BayesMallows

Unless the argument `error_model` to `compute_mallows` is set, pairwise preference data are assumed to be consistent within each assessor. These data should be provided in a dataframe with the following three columns, with one row per pairwise comparison.

- `assessor` is an identifier for the assessor; either a numeric vector containing the assessor index, or a character vector containing the unique name of the assessor.
- `bottom_item` is a numeric vector containing the index of the item that was disfavored in each pairwise comparison.
- `top_item` is a numeric vector containing the index of the item that was preferred in each pairwise comparison.

A dataframe with this structure can be given in the `preferences` argument to `compute_mallows`. `compute_mallows` will generate the full set of implied rankings for each assessor using the function `generate_transitive_closure`, as well as an initial ranking matrix consistent with the pairwise preferences, using the function `generate_initial_ranking`.

We illustrate with the beach preference data containing stated pairwise preferences between random subsets of 15 images of beaches, by 60 assessors (Vitelli et al., 2018, Section 6.2). This dataset is provided in the dataframe `beach_preferences`.

Transitive closure and initial ranking

We start by generating the transitive closure implied by the pairwise preferences.

```
beach_tc <- generate_transitive_closure(beach_preferences)
```

The dataframe `beach_tc` contains all the pairwise preferences in `beach_preferences`, with all the implied pairwise preferences in addition. The latter are preferences that were not specifically stated by the assessor, but instead are implied by the stated preferences. As a consequence, the dataframe `beach_tc` has 2921 rows, while `beach_preferences` has 1442 rows. Initial rankings, i.e., a set of full rankings for each assessor that are consistent with the implied pairwise preferences are then generated, and we set the column names of the initial ranking matrix to "Beach 1", "Beach 2", ..., "Beach 15" in order have these names appear as labels in plots and output.

```
beach_init_rank <- generate_initial_ranking(beach_tc)
colnames(beach_init_rank) <- paste("Beach", 1:ncol(beach_init_rank))
```

If we had not generated the transitive closure and the initial ranking, `compute_mallows` would do this for us, but when calling `compute_mallows` repeatedly, it may save time to have these precomputed and saved for future re-use. In order to save time in the case of big datasets, the functions for generating transitive closures and initial rankings from transitive closures can all be run in parallel, as shown in the examples to the `compute_mallows` function. The key to the parallelization is that each assessor's preferences can be handled independently of the others, and this can speed up the process considerably with large dataset.

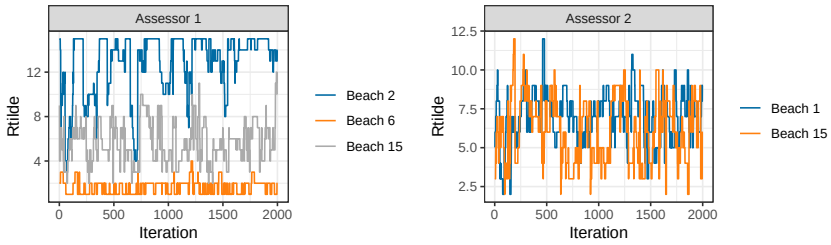
As an example, let us look at all preferences stated by assessor 1 involving beach 2. We use `filter` from **dplyr** (Wickham et al., 2018) to obtain the right set of rows.

```
library("dplyr")
# All preferences stated by assessor 1 involving item 2
filter(beach_preferences, assessor == 1, bottom_item == 2 | top_item == 2)

# A tibble: 1 x 3
  assessor bottom_item top_item
     <dbl>       <dbl>    <dbl>
1        1           2       15
```

Assessor 1 has performed only one direct comparison involving beach 2, in which the assessor stated that beach 15 is preferred to beach 2. The implied orderings, on the other hand, contain two preferences involving beach 2:

```
# All implied orderings for assessor 1 involving item 2
filter(beach_tc, assessor == 1, bottom_item == 2 | top_item == 2)

  assessor bottom_item top_item
1        1           2        6
2        1           2       15
```

In addition to the statement that beach 15 is preferred to beach 2, all the other orderings stated by assessor 1 imply that this assessor prefers beach 6 to beach 2.



(a) Beaches 2, 6, and 15 for assessor 1. The ordering of the traces is always consistent with the constraints set by assessor 1's preferences.

(b) Beaches 1 and 15 for assessor 2. No orderings between beaches 1 and 15 are implied by assessor 1's preferences, and the traces are hence free to cross.

Figure 5: Trace plots of augmented ranks $\tilde{R}$

Convergence diagnostics

As with the potato data, we can do a test run to assess the convergence of the MCMC algorithm. However, this time we provide the initial rankings `beach_init_rank` to the `rankings` argument and the transitive closure `beach_tc` to the `preferences` argument of `compute_mallows`. We also set `save_aug = TRUE` to save the augmented rankings in each MCMC step, hence letting us assess the convergence of the augmented rankings.

```
bmm_test <- compute_mallows(rankings = beach_init_rank,
                            preferences = beach_tc, save_aug = TRUE)
```

Running `assess_convergence` for α and $\boldsymbol{\rho}$ shows good convergence after 1000 iterations (not shown). To check the convergence of the data augmentation scheme, we need to set `parameter = "Rtilde"`, and also specify which items and assessors to plot. Let us start by considering items 2, 6, and 15 for assessor 1, which we studied above.

```
assess_convergence(bmm_test, parameter = "Rtilde",
                   items = c(2, 6, 15), assessors = 1)
```

The resulting plot is shown in Figure 5a. It illustrates how the augmented rankings vary, while also obeying their implied ordering.

By further investigation of `beach_tc`, we would find that no orderings are implied between beach 1 and beach 15 for assessor 2. With the following command, we create trace plots to confirm this:

```
assess_convergence(bmm_test, parameter = "Rtilde",
                   items = c(1, 15), assessors = 2)
```

The resulting plot is shown in Figure 5b. As expected, the traces of the augmented rankings for beach 1 and 15 for assessor 2 do cross each other, since no ordering is implied between them. Ideally, we should look at trace plots for augmented ranks for more assessors to be sure that the algorithm is close to convergence. We can plot assessors 1-8 by setting `assessors = 1:8`. We also quite arbitrarily pick items 13-15, but the same procedure can be repeated for other items.

```
assess_convergence(bmm_test, parameter = "Rtilde",
                   items = 13:15, assessors = 1:8)
```

The resulting plot is shown in Figure 6, indicating good mixing.

Posterior distributions

Based on the convergence diagnostics, and being fairly conservative, we discard the first 2,000 MCMC iterations as burn-in, and take 100,000 additional samples.

```
bmm_beaches <- compute_mallows(rankings = beach_init_rank, preferences = beach_tc,
                               nmc = 102000, save_aug = TRUE)
bmm_beaches$burnin <- 2000
```

The posterior distributions of α and $\boldsymbol{\rho}$ can be studied as shown in the previous sections. Posterior intervals for the latent rankings of each beach are obtained with `compute_posterior_intervals`:

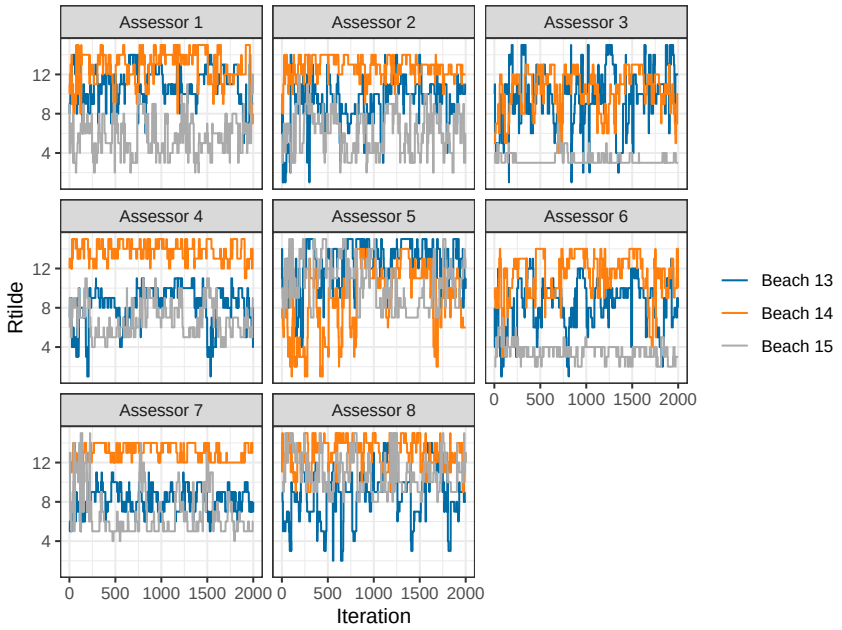


Figure 6: Trace plots of augmented ranks $\tilde{R}$ for beaches 13-15 and assessors 1-8, indicating that the MCMC algorithm obtains good mixing after a low number of iterations.

```
compute_posterior_intervals(bmm_beaches, parameter = "rho")

# A tibble: 15 x 7
   item     parameter  mean median conf_level hpdi    central_interval
   <fct>    <chr>     <dbl>  <dbl> <chr>      <chr>   <chr>
 1 Beach 1  rho           7      7 95 %       [7]     [6,7]
 2 Beach 2  rho          15     15 95 %       [15]    [15]
 3 Beach 3  rho           3      3 95 %       [3,4]   [3,4]
 4 Beach 4  rho          12     12 95 %       [11,13] [11,14]
 5 Beach 5  rho           9      9 95 %       [8,10]  [8,10]
 6 Beach 6  rho           2      2 95 %       [1,2]   [1,2]
 7 Beach 7  rho           9      8 95 %       [8,10]  [8,10]
 8 Beach 8  rho          12     11 95 %       [11,13] [11,14]
 9 Beach 9  rho           1      1 95 %       [1,2]   [1,2]
10 Beach 10 rho           6      6 95 %       [5,6]   [5,7]
11 Beach 11 rho           4      4 95 %       [3,4]   [3,5]
12 Beach 12 rho          13     13 95 %       [12,14] [12,14]
13 Beach 13 rho          10     10 95 %       [8,10]  [8,10]
14 Beach 14 rho          13     14 95 %       [12,14] [11,14]
15 Beach 15 rho           5      5 95 %       [5,6]   [4,6]
```

We can also rank the beaches according to their cumulative probability (CP) consensus (Vitelli et al., 2018, Section 5.1) and their maximum posterior (MAP) rankings. This is done with the function `compute_consensus`, and the following call returns the CP consensus:

```
compute_consensus(bmm_beaches, type = "CP")

# A tibble: 15 x 3
   ranking item     cumprob
     <dbl> <chr>      <dbl>
 1       1 Beach 9    0.896
 2       2 Beach 6    1
 3       3 Beach 3    0.738
```

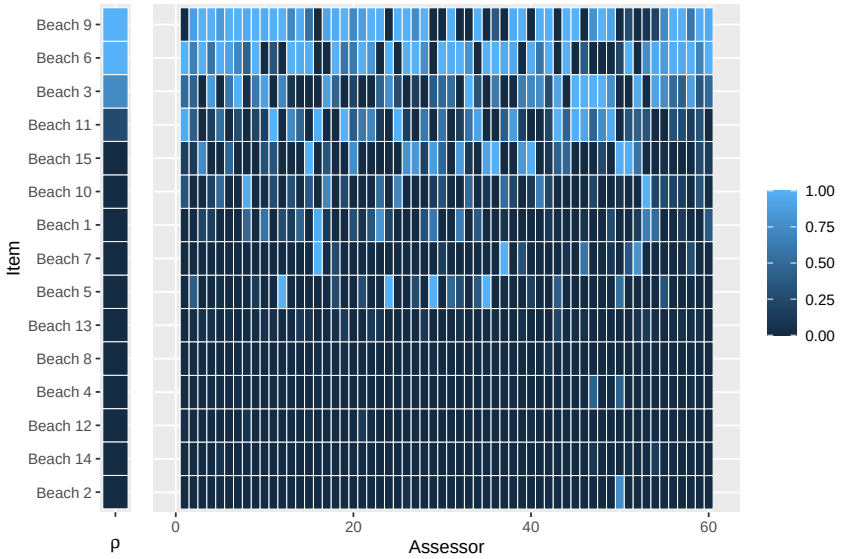


Figure 7: Probability of being ranked top-3 for each beach in the beach preference example (left) and the probability that each assessor ranks the given beach among top-3 (right). Beaches 6 and 9 are most popular overall, but the assessor differ considerably in their preference for these beaches, as can be seen by the varying pattern of light and dark blue.

```
 4       4 Beach 11   0.966
 5       5 Beach 15   0.953
 6       6 Beach 10   0.971
 7       7 Beach 1    1
 8       8 Beach 7    0.528
 9       9 Beach 5    0.887
10      10 Beach 13   1.00
11      11 Beach 8    0.508
12      12 Beach 4    0.717
13      13 Beach 12   0.643
14      14 Beach 14   0.988
15      15 Beach 2    1
```

The column `cumprob` shows the probability of having the given rank or lower. Looking at the second row, for example, this means that beach 6 has probability 1 of having latent ranking 2 or lower. Next, beach 3 has probability 0.738 of having latent rank 3 or lower. This is an example of how the Bayesian framework can be used to not only rank items, but also to give posterior assessments of the uncertainty of the rankings. The MAP consensus is obtained similarly, by setting `type = "MAP"`.

Keeping in mind that the ranking of beaches is based on sparse pairwise preferences, we can also ask: for beach i, what is the probability of being ranked top-k by assessor j, and what is the probability of having latent rank among the top-k. The function `plot_top_k` plots these probabilities. By default, it sets `k = 3`, so a heatplot of the probability of being ranked top-3 is obtained with the call:

```
plot_top_k(bmm_beaches)
```

The plot is shown in Figure 7. The left part of the plot shows the beaches ranked according to their CP consensus, and the probability $P(\rho_i) \leq 3$ for each beach i. The right part of the plot shows, for each beach as indicated on the left axis, the probability that assessor j ranks the beach among top-3. For example, we see that assessor 1 has a very low probability of ranking beach 9 among her top-3, while assessor 3 has a very high probability of doing this. The function `predict_top_k` returns a dataframe with all the underlying probabilities. For example, in order to find all the beaches that are among the top-3 of assessors 1-5 with more than 90 % probability, we would do:

```
predict_top_k(bmm_beaches) %>%
  filter(prob > 0.9, assessor %in% 1:5)
```

```
# A tibble: 6 x 3
# Groups:   assessor [4]
  assessor item      prob
     <dbl> <chr>    <dbl>
1        1 Beach 11 0.955
2        1 Beach 6  0.997
3        3 Beach 6  0.997
4        3 Beach 9  1
5        4 Beach 9  1.00
6        5 Beach 6  0.979
```

Note that assessor 2 does not appear in this table, i.e., there are no beaches for which we are at least 90 % certain that the beach is among assessor 2's top-3.

Clustering with BayesMallows

BayesMallows comes with a set of sushi preference data, in which 5,000 assessors each have ranked a set of 10 types of sushi (Kamishima, 2003). It is interesting to see if we can find subsets of assessors with similar preferences. The sushi dataset was analyzed with the BMM by Vitelli et al. (2018), but the results in that paper differ somewhat from those obtained here, due to a bug in the function that was used to sample cluster probabilities from the Dirichlet distribution.

Computing mixtures of Mallows distributions

The function `compute_mallows_mixtures` computes multiple Mallows models with different numbers of mixture components. It returns a list of models of class `BayesMallowsMixtures`, in which each list element contains a model with a given number of mixture components. Its arguments are `n_clusters`, which specifies the number of mixture components to compute, an optional parameter `cl` which can be set to the return value of the `makeCluster` function in the **parallel** package, and an ellipsis (`...`) for passing on arguments to `compute_mallows`.

Hypothesizing that we may not need more than 10 clusters to find a useful partitioning of the assessors, we start by doing test runs with 1, 4, 7, and 10 mixture components in order to assess convergence. We set the number of Monte Carlo samples to 5,000, and since this is a test run, we do not save cluster assignments nor within-cluster distances from each MCMC iteration and hence set `save_clus = FALSE` and `include_wcd = FALSE`. We also run the computations in parallel on four cores, one for each mixture component.

```
library("parallel")
cl <- makeCluster(4)
bmm <- compute_mallows_mixtures(n_clusters = c(1, 4, 7, 10),
                                rankings = sushi_rankings, nmc = 5000,
                                save_clus = FALSE, include_wcd = FALSE, cl = cl)
stopCluster(cl)
```

Convergence diagnostics

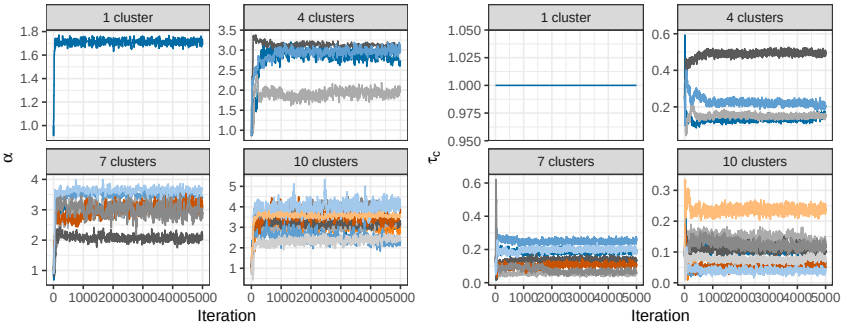
The function `assess_convergence` automatically creates a grid plot when given an object of class `BayesMallowsMixtures`, so we can check the convergence of α with the command

```
assess_convergence(bmm)
```

The resulting plot is given in Figure 8a, showing that all the chains seem to be close to convergence quite quickly. We can also make sure that the posterior distributions of the cluster probabilities τ_c, $(c = 1, \dots, C)$ have converged properly, by setting `parameter = "cluster_probs"`.

```
assess_convergence(bmm, parameter = "cluster_probs")
```

The trace plots for each number of mixture components are shown in Figure 8b. Note that with only one cluster, the cluster probability is fixed at the value 1, while for other number of mixture components, the chains seem to be mixing well.



(a) Trace of α. **(b)** Trace of τ_c.

Figure 8: Trace plot of α and τ_c for the sushi dataset with 1, 4, 7, and 10 mixture components, respectively. Both trace plots indicate good mixing after a few thousand iterations.

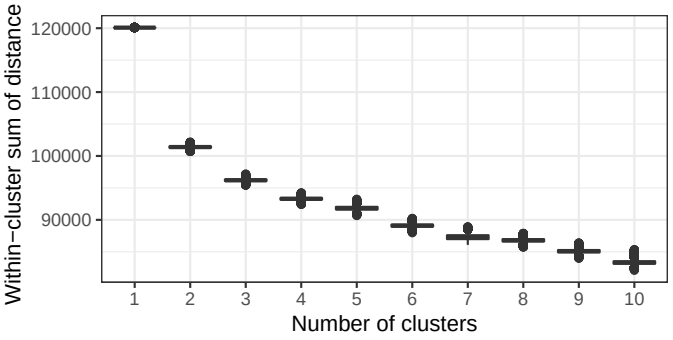


Figure 9: Elbow plot for the sushi mixture models. While it is not entirely clear where the elbow occurs, we choose the mixture distribution with five clusters.

Deciding on the number of mixtures

Given the convergence assessment of the previous section, we are fairly confident that a burn-in of 5,000 is sufficient. We run 95,000 additional iterations, and try from 1 to 10 mixture components. Our goal is now to determine the number of mixture components to use, and in order to create an elbow plot, we set `include_wcd = TRUE` to compute the within-cluster distances in each step of the MCMC algorithm. Since the posterior distributions of ρ_c ($c = 1, \ldots, C$) are highly peaked, we save some memory by only saving every 10th value of $\boldsymbol{\rho}$ by setting `rho_thinning = 10`.

```
cl <- makeCluster(4)
bmm <- compute_mallows_mixtures(n_clusters = 1:10, rankings = sushi_rankings,
                                nmc = 100000, rho_thinning = 10, save_clus = FALSE,
                                include_wcd = TRUE, cl = cl)
stopCluster(cl)
plot_elbow(bmm, burnin = 5000) # Create elbow plot
```

The resulting elbow plot is a notched boxplot (Mcgill et al., 1978; Wickham, 2016) shown in Figure 9, for which the barely visible upper and lower whiskers represent approximate 95 % confidence intervals. Although not clear-cut, we see that the within-cluster sum of distances levels off at around 5 clusters, and hence we choose to use 5 clusters in our model.

Posterior distributions

Having chosen 5 mixture components, we go on to fit a final model, still running 95,000 iterations after burnin. This time we call `compute_mallows` and set `n_clusters = 5`. We also set `save_clus =`

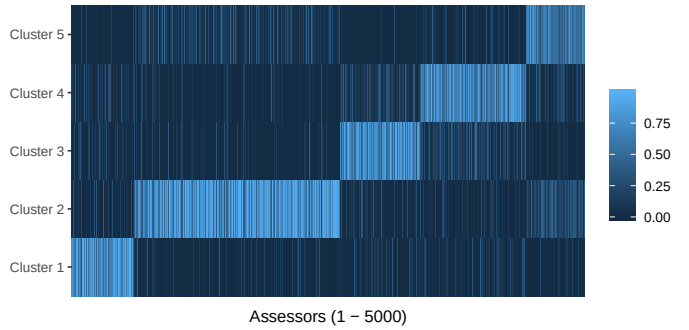


Figure 10: Posterior probabilities of assignment to each cluster for each of the 5000 assessors in the sushi dataset. The scale to the right shows the color coding of probabilities. The blocks of light colors along the anti-diagonal show the clusters to which the assessors were assigned. Darker colors within these blocks indicate assessors whose cluster assignment is uncertain.

`TRUE` and `clus_thin = 10` to save the cluster assignments of each assessor in every 10th iteration, and `rho_thinning = 10` to save the estimated latent rank every 10th iteration.

```
bmm <- compute_mallows(rankings = sushi_rankings, n_clusters = 5, save_clus = TRUE,
                       clus_thin = 10, nmc = 100000, rho_thinning = 10)
bmm$burnin <- 5000
```

We can plot the posterior distributions of α and $\boldsymbol{\rho}$ in each cluster using `plot.BayesMallows` as shown preivously for the potato data. We can also show the posterior distributions of the cluster probabilities, using:

```
plot(bmm, parameter = "cluster_probs")
```

Using the argument `parameter = "cluster_assignment"`, we can visualize the posterior probability for each assessor of belonging to each cluster:

```
plot(bmm, parameter = "cluster_assignment")
```

The resulting plot is shown in Figure 10. The underlying numbers can be obtained using the function `assign_cluster`.

We can find clusterwise consensus rankings using `compute_consensus`. The following call finds the CP consensuses, and then uses `select` from **dplyr** and `spread` from **tidyr** (Wickham and Henry, 2018) to create one column for each cluster. The result is shown in Table 1.

```
library("tidyr")
compute_consensus(bmm) %>%
    select(-cumprob) %>%
    spread(key = cluster, value = item)
```

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
1	shrimp	fatty tuna	fatty tuna	fatty tuna	fatty tuna
2	sea eel	sea urchin	sea eel	tuna	sea urchin
3	egg	salmon roe	tuna	shrimp	shrimp
4	squid	sea eel	shrimp	tuna roll	tuna
5	salmon roe	tuna	tuna roll	squid	salmon roe
6	fatty tuna	shrimp	squid	salmon roe	squid
7	tuna	tuna roll	egg	egg	tuna roll
8	tuna roll	squid	cucumber roll	cucumber roll	sea eel
9	cucumber roll	egg	salmon roe	sea eel	egg
10	sea urchin	cucumber roll	sea urchin	sea urchin	cucumber roll

Table 1: CP consensus for each of the clusters found for sushi data.

Note that for estimating cluster specific parameters, label switching is a potential problem that needs to be handled. **BayesMallows** ignores label switching issues inside the MCMC, because it has been shown that this approach is better for ensuring full convergence of the chain (Jasra et al., 2005; Celeux et al., 2000). MCMC iterations can be re-ordered after convergence is achieved, for example by using the implementation of Stephens' algorithm (Stephens, 2000) provided by the R package **label.switching** (Papastamoulis, 2016). A full example of how to assess label switching after running `compute_mallows` is provided by running the following command:

```
help("label_switching")
```

For the sushi data analyzed in this section, no label switching is detected by Stephen's algorithm.

Discussion

In this paper we discussed the methodological background and computational strategies for the **BayesMallows** package, implementing the inferential framework for the analysis of preference data based on the Bayesian Mallows model, as introduced in Vitelli et al. (2018). The package aims at providing a general probabilistic tool, capable of performing various inferential tasks (estimation, classification, prediction) with a proper uncertainty quantification. Moreover, the package widens the applicability of the Mallows model, by providing reliable algorithms for approximating the associated partition function, which has been the bottleneck for a successful use of this general and flexible model so far. Finally, it handles a variety of preference data types (partial rankings, pairwise preferences), and it could possibly handle many others which can lie in the above mentioned categories (noisy continuous measurements, clicking data, ratings).

One of the most important features of the **BayesMallows** package is that, despite implementing a Bayesian model, and thus relying on MCMC algorithms, its efficient implementation makes it possible to manage large datasets. The package can easily handle up to hundreds of items, and thousands of assessors; an example is the Movielens data analyzed in Section 6.4 of (Vitelli et al., 2018). By using the log-sum-exp trick, the implementation of the importance sampler is able to handle at least ten thousand items without numerical overflow. We believe that all these features make the package a unique resource for fitting the Mallows model to large data, with the benefits of a fully probabilistic interpretation.

Nonetheless, we also recognize that the **BayesMallows** package can open the way for further generalizations. The Bayesian Mallows model for time-varying rankings that has been introduced in Asfaw et al. (2017) will be considered for a future release. Some further extensions which we might consider to implement in the **BayesMallows** in the future include: fitting an infinite mixture of Mallows models for automatically performing model selection; allowing for a non-uniform prior for $\boldsymbol{\rho}$; performing automatic item selection; estimating the assessors' quality as rankers; and finally including covariates, both on the assessors and on the items. In addition, since the data augmentation steps in the MCMC algorithm are independent across assessors, potential speedup in the case of missing data or pairwise preferences can be obtained by updating the augmented data in parallel, and this is likely to be part of a future package update.

Acknowledgments

The authors would like to thank Arnoldo Frigessi and Elja Arjas for fruitful discussions.

Bibliography

M. Alvo and P. L. Yu. *Statistical Methods for Ranking Data*. Frontiers in Probability and the Statistical Sciences. Springer, New York, NY, USA, 2014. URL https://doi.org/10.1007/978-1-4939-1471-5. [p7]

D. Asfaw, V. Vitelli, Ø. Sørensen, E. Arjas, and A. Frigessi. Time-varying rankings with the Bayesian Mallows model. *Stat*, 6(1):14–30, 2017. URL https://doi.org/10.1002/sta4.132. [p17]

G. Celeux, M. Hurn, and C. Robert. Computational and inferential difficulties with mixture posterior distribution. *Journal of the American Statistical Association*, 95(451):957–970, 2000. URL https://doi.org/10.2307/2669477. [p17]

M. Crispino, E. Arjas, V. Vitelli, N. Barrett, and A. Frigessi. A Bayesian Mallows approach to nontransitive pair comparison data: How human are sounds? *The Annals of Applied Statistics*, 13(1):492–519, Mar. 2019. URL https://doi.org/10.1214/18-aoas1203. [p4, 5]

J. C. de Borda. Mémoire sur les élections au scrutin, histoire de l'académie royale des sciences. *Paris, France*, 1781. [p4]

P. Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *Lecture Notes - Monograph Series*. Institute of Mathematical Statistics, Hayward, CA, USA, 1988. [p1, 2, 3]

M. A. Fligner and J. S. Verducci. Distance based ranking models. *Journal of the Royal Statistical Society: Series B (Methodological)*, 48(3):359–369, July 1986. URL https://doi.org/10.1111/j.2517-6161.1986.tb01420.x. [p3, 7]

E. Irurozki, B. Calvo, and A. Lozano. Sampling and learning the Mallows and weighted Mallows models under the Hamming distance. *Technical Report*, 2014. URL https://addi.ehu.es/handle/10810/11240. [p3]

E. Irurozki, B. Calvo, and J. A. Lozano. PerMallows: An R Package for Mallows and Generalized Mallows Models. *Journal of Statistical Software*, 71(12), 2016a. URL https://doi.org/10.18637/jss.v071.i12. [p7]

E. Irurozki, B. Calvo, and J. A. Lozano. Sampling and learning Mallows and generalized Mallows models under the Cayley distance. *Methodology and Computing in Applied Probability*, 20(1):1–35, June 2016b. URL https://doi.org/10.1007/s11009-016-9506-7. [p3]

A. Jasra, C. C. Holmes, and D. A. Stephens. Markov chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. *Statistical Science*, 20(1):50–67, Feb. 2005. URL https://doi.org/10.1214/088342305000000016. [p17]

T. Kamishima. Nantonac collaborative filtering: Recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 583–588, New York, NY, USA, 2003. ACM. ISBN 1-58113-737-0. URL https://doi.org/10.1145/956750.956823. [p14]

P. H. Lee and P. L. Yu. An R package for analyzing and modeling ranking data. *BMC Medical Research Methodology*, 13(1):65, May 2013. ISSN 1471-2288. doi: 10.1186/1471-2288-13-65. URL https://doi.org/10.1186/1471-2288-13-65. [p7]

Q. Liu, M. Crispino, I. Scheel, V. Vitelli, and A. Frigessi. Model-based learning from preference data. *Annual Review of Statistics and Its Application*, 6(1):329–354, 2019. URL https://doi.org/10.1146/annurev-statistics-031017-100213. [p2, 7]

T. Lu and C. Boutilier. Effective sampling and learning for Mallows models with pairwise-preference data. *Journal of Machine Learning Research*, 15:3783–3829, 2014. [p3]

C. L. Mallows. Non-null ranking models. i. *Biometrika*, 44(1-2):114–130, 1957. URL https://doi.org/10.1093/biomet/44.1-2.114. [p1, 2]

J. I. Marden. *Analyzing and Modeling Rank Data*, volume 64 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, Cambridge, MA, USA, 1995. [p1, 3]

M. J. A. N. d. C. Marquis of Condorcet. Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix. *Paris: De l'imprimerie royale*, 1785. [p4]

R. Mcgill, J. W. Tukey, and W. A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978. URL https://doi.org/10.1080/00031305.1978.10479236. [p15]

M. Meilă and H. Chen. Dirichlet process mixtures of generalized Mallows models. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 358–367, Corvallis, OR, USA, 2010. AUAI Press. [p3]

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2018. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-6. [p9]

S. Mukherjee. Estimation in exponential families on permutations. *The Annals of Statistics*, 44(2):853–875, 2016. URL https://doi.org/doi:10.1214/15-AOS1389. [p1, 6]

K. Müller and H. Wickham. *tibble: Simple Data Frames*, 2018. URL https://CRAN.R-project.org/package=tibble. R package version 1.4.2. [p8]

P. Papastamoulis. label.switching: An R package for dealing with the label switching problem in MCMC outputs. *Journal of Statistical Software*, 69, 2016. URL https://doi.org/10.18637/jss.v069.c01. [p17]

Z. Qian. *rankdist: Distance Based Ranking Models*, 2018. URL https://CRAN.R-project.org/package=rankdist. R package version 1.1-3. [p7]

N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences, 2017. URL http://oeis.org. [p6]

M. Stephens. Dealing with label switching in mixture models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):795–809, Nov. 2000. URL https://doi.org/10.1111/1467-9868.00265. [p17]

V. Vitelli, Ø. Sørensen, M. Crispino, A. Frigessi, and E. Arjas. Probabilistic preference learning with the Mallows rank model. *Journal of Machine Learning Research*, 18(1):5796–5844, Jan. 2018. [p1, 2, 3, 5, 6, 7, 10, 12, 14, 17]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL http://ggplot2.org. [p15]

H. Wickham and L. Henry. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*, 2018. URL https://CRAN.R-project.org/package=tidyr. R package version 0.8.1. [p16]

H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2018. URL https://CRAN.R-project.org/package=dplyr. R package version 0.7.7. [p10]

Øystein Sørensen
Center for Lifespan Changes in Brain and Cognition
Department of Psychology
University of Oslo
ORCID: 0000-0003-0724-3542
oystein.sorensen@psykologi.uio.no

Marta Crispino
Univ. Grenoble Alpes, Inria, CNRS, LJK
38000 Grenoble, France
crispino.marta8@gmail.com

Qinghua Liu
Department of Mathematics
University of Oslo
qinghual@math.uio.no

Valeria Vitelli
Oslo Centre for Biostatistics and Epidemiology
Department of Biostatistics
University of Oslo
OORCID: 0000-0002-6746-0453
valeria.vitelli@medisin.uio.no