Model-Based Learning from Preference Data

Eric Frey, Sam Minors, Margherita Philipp, Davis Thomas

March 12, 2023

Contents

1 Introduction

Ranking data is distinct from other forms of data because it only provides ordinal rather than cardinal information. It is easy to convert cardinal numbers into an ordinal ranking: given the turnover figures of several companies we can sort them in descending order and make statements about how much bigger the highest ranking firm's turnover was. However, when someone tells us their top three movie choices are Toy Story, Finding Nemo and Ice Age, we only know that Toy Story is preferred to the others - we do not know by how much.

There are two main tasks we might want to achieve when given ranking data. The first is to aggregate people's preferences into an overall ranking. The second is to extrapolate from an individual's ranking over a subset of options what their ranking would be over the remaining set. The latter task for example relates to recommender algorithms that come up with suitable Netflix titles based on a user's viewing history. Such a viewing history is a form of ranking data from revealed preference: we choose one film over the others, and we either did or didn't finish it.

For this project we have focused on the context of ranking aggregation. This can be applied in the context of vote aggregation. For example, New York City held its first Ranked Choice Voting election in early 2021, allowing voters to rank their top five candidates. Aggregation has also been of interest for grading student assignments - both by teachers and by peers. Fundamentally, being able to aggregate preferences allows us to make statements of what ranks highly overall. This could then lead to a recommendation of who should become mayor or how to best improve a powerpoint slide.

In practice, aggregation of ranking data is performed both on explicit and implicit data. Explicit data is based on assessors generating ranking directly or through pairwise comparisons. For implicit data, assessors' preferences are expressed indirectly through actions such as clicking, buying, or watching among several available items.

1.1 Notation

A full ranking of n assigns a unique rank to each item in a list. This could be a ranking based on how heavy an item is perceived to be, or how much an assessor likes it. A rank is an integer value between 1 and n, with lower values indicating a higher preference. A full ranking is represented as a mapping $R : A \to P_n$, where A is the set of items to be ranked and P_n is the $n!$ space of permutations of n elements. This is denoted as $X = (X_1, X_2, \ldots, X_n)$, where X_i is the i-th item in the ordering.

For example, consider the following full ranking of 10 items:

Item	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
Full ranking	1	7	8	2	10	4	6	9	3	5

The corresponding ordering vector is $X = (A_1, A_4, A_9, A_6, A_{10}, A_7, A_2, A_3, A_8, A_5)$.

In ranking aggregation problems the term of assessors is used to refer to the sources of ranking data. While lower case n refers to the number of items to rank, upper case N refers to the number of assessors who have provided a ranking.

1.2 Development of probabilistic ranking models

From the literature we gained an overview of how the approach to dealing ranking data has evolved over time. There are three main strands or approaches to a probabilistic aggregation of preferences: Thurstonian models, Mallows models and Plackett-Luce models.

In Thurstonian models, latent variables describe the mapping of a continuous scale onto ordered categories. This means that the probability of one item being preferred to another can be modeled as a function of the difference in their underlying latent utilities. Thurstone's model assumes that the difference in the latent utilities is normally distributed, and it estimates the parameters of the distribution using the rank data. While the models described below mainly focus on generating an aggregate, the underlying structure of the Thurstone model allows for the estimation of individual-level utilities. This can be useful in some applications, e.g. when also wanting to give bespoke recommendations.

The Mallows model is the one we focus on in this paper. It assumes that the ranking data can be explained by a central 'consensus' ranking (sometimes also called 'ideal') and a measure of individual-level noise or error. Mallows' model estimates the consensus ranking and the amount of noise using the rank data. An advantage of the Mallows model (that we however do not explore further in this paper) is that it can situations in which an assessor gives the same rank to two items. Namely, the model assumes that the probability of a respondent ranking one item above another depends on the distance between the items in the consensus ranking and the level of noise in the data.

The Plackett-Luce is a multistage ranking model. It assumes that the ranking data can be explained by a probabilistic model in which each item has a fixed probability of being chosen first, and the probability of an item being chosen second (or third, etc.) depends on the items that were chosen first. Plackett-Luce's model estimates the probabilities of each item being chosen first using the ranking data. One advantage of the Plackett-Luce model is that it can handle large sets of items, which can be computationally challenging for some other methods.

2 The Mallows Model

The Mallows model works on the assumption that there is a true underlying consensus. This latent ranking is labelled ρ and also referred to as the location parameter, telling us which end of the ranking order we would expect most assessors' evaluation of each item to be. Of course ρ needs to be within the space of n-dimensional possible permutations P_n.

The other parameter of the model is α. It is a scale parameter that tells us how 'noisy' the data is. When $\alpha = 0$, all assessors have the exact same ranking over the items. A larger α implies that there is more disagreement between assessors. The probability of any one assessor j's ($j \in 1, ..., N$ ranking is

$$P(\boldsymbol{R} \| \alpha, \boldsymbol{\rho}) = \frac{1}{Z_n(\alpha)} \exp\left[-\frac{\alpha}{n} d(\boldsymbol{R}, \boldsymbol{\rho})\right]$$

A feature to choose is the distance measure d. It should be a right-invariant measure, such that the the distance between any two rankings does not change if the alternatives are relabeled. From the equation above we see that as the distance between the observed ranking R and the consensus ρ increases, the absolute value of the the negative exponent increases, implying that the probability of that ranking being observed for a given ρ and α is lower. Possible options for d include the

Spearman's, footrule, Cayley, Hamming and Ulam distance measures (Liu et al, 2019). For this project we used Kendall's tau. An important reason why the Kendall distance is popular in this context is because of the partition function. $Z_n(\alpha)$ can be thought of as a normalising constant, and it exists in closed form for the Kendall distance. In the practical exercise we compare several different distance measures to each other.

To obtain the probability of a set of rankings by N assessors we multiply probabilities of all the individuals rankings in the set:

$$P(\boldsymbol{R}^{(\mathsf{N})}|\alpha, \boldsymbol{\rho}) = \frac{1}{[Z_n(\alpha)]^N} \exp\left[-\frac{\alpha}{n}\sum_{j=1}^{N} d(\boldsymbol{R}_j, \boldsymbol{\rho})\right]$$

The standard Mallows model uses maximum likelihood estimation to estimate the parameters of the model. However, in practice this would be cumbersome as all $n!$ permutations of the rankings would have to be summed over. We would also only ever be able to obtain a maximum likelihood of ρ for a given value of α. Thus we turn to a Bayesian approach that allows us to use a Markov Chain Monte Carlo (MCMC) algorithm to sample from the posterior distribution.

2.1 Adding Bayesian Inference and MCMC Sampling

The Bayesian Mallows model is a variant of the Mallows model that incorporates Bayesian inference methods. In contrast to the standard Mallows model, the Bayesian Mallows model assigns prior probabilities to the parameters and uses Bayesian methods to estimate the posterior probabilities.

In the Bayesian Mallows model (BMM) introduced by Vitelli et al. (2018), the consensus ranking and the dispersion parameter are assigned prior distributions $\pi(\rho)$ and $\pi(\alpha)$, which reflect the researcher's beliefs about their values before observing the data. The posterior distribution of the parameters is then updated using Bayes' theorem to incorporate the information contained in the observed rankings. The posterior is proportional to the product of the priors and the likelihood of the observed rankings given the initialised values of α and ρ.

$$\pi(\boldsymbol{\rho}, \alpha|\boldsymbol{R}^{(N)}) \propto \frac{\pi(\boldsymbol{\rho})\pi(\alpha)}{[Z_n(\alpha)]^N} \exp\left[-\frac{\alpha}{n}\sum_{j=1}^{N} d(\boldsymbol{R}_j, \boldsymbol{\rho})\right]$$

For ρ it makes sense to just pick a prior from a random uniform distribution of the possible permutations. Meanwhile Vitelli et al (2018) proposes that for α, the prior should be a truncated exponential distribution. The truncation ensures that the value of alpha is always positive.

The advantage of the Bayesian formulation we have above is that we can sample from it using a MCMC algorithm, and at convergence this will essentially sample from the posterior distribution of interest. For the BMM implementation, the authors use a Metropolis-Hastings MCMC method. In the Metropolis-Hastings algorithm a new parameter value θ is proposed iteratively and either accepted or rejected, depending on whether it makes the observed data more or less likely. The algorithm can be broken down into a loop of four steps:

1. Propose a new sample based on a proposal distribution.
2. Compute the acceptance probability.
3. If acceptance probability is greater than a random number drawn from a uniform distribution in the range (0,1), accept the proposal. If not, reject the proposal.
4. Repeat until convergence.

In the general introduction above we can think of our parameters α and ρ as part of a parameter vector θ. In practice, however, we need to update one parameter at a time. As α is a single number, we might hope to reach convergence more quickly than for ρ with its many permutations. In fact, the implementation of the Mallows model allows us to update α less frequently and instead

focus more iterations on finding ρ. The underlying consensus may also be greater interest than the extent to which assessors are close to the consensus. However, step 1 needs to be adapted when dealing with a list of ranks rather than a single numerical value. We discuss this in the next section.

It should be noted that the interpretation of α is critical: if it is high, then strong disagreement may encourage us to explore whether it makes more sense to assume that the data was generated from clusters around two or more consensus rankings rather than a single central one.

2.2 Proposal Distributions

A key feature of Metropolis-Hastings MCMC algorithm is that is comes up with a new proposed parameter based on the current proposal. This is more challenging for ρ than for α, which is why Vitelli et al (2018) propose the Leap and Shift algorithm.

2.2.1 Proposal Distribution for α

Alpha α is initialized with a value of 1, (see Vitelli, V. (2019)), which obeys the condition that it be positive, and further values are drawn from a proposal distribution that is a log normal distribution where the mean is the previous alpha, and the variance is 0.5.

The forward and backward transitive probabilities in this case are more intuitive, as the probability of drawing the new proposal α^* conditional on the existing value of alpha α_{i-1} is just the probability density of the value of the proposal from the probability distribution of a log normal distribution which has a mean of α_{i-1}. The acceptance ratio is also computed here, and used to decide whether the proposal should be updated or not.

2.2.2 Leap and Shift approach for ρ

As ρ is a list of rankings, we cannot treat it as a single number. We also do not want to update all rankings at the same time, but rather make partial adjustments to come up with a new consensus ranking proposal. The Lean and Shift algorithm effectively re-shuffles a subset of rankings within the list around a randomly chosen index u.

The description of the algorithm from the original paper is shown below. As the notation is quite dense, we provide an illustrative example to demonstrate how this algorithm generates a re-arranged consensus proposal.

Definition 1 *Leap-and-Shift Proposal (L&S). Fix an integer* $L \in \{1, \ldots, \lfloor (n-1)/2 \rfloor\}$ *and draw a random number* $u \sim \mathcal{U}\{1, \ldots, n\}$. *Define, for a given* $\boldsymbol{\rho}$, *the set of integers* $\mathcal{S} = \{\max(1, \rho_u - L), \min(n, \rho_u + L)\} \setminus \{\rho_u\}$, $\mathcal{S} \subseteq \{1, \ldots, n\}$, *and draw a random number* r *uniformly in* $\mathcal{S}$. *Let* $\boldsymbol{\rho}^* \in \{1, 2, ...n\}^n$ *have elements* $\rho^*_u = r$ *and* $\rho^*_i = \rho_i$ *for* $i \in \{1, \ldots, n\} \setminus \{u\}$, *constituting the leap step. Now, define* $\Delta = \rho^*_u - \rho_u$ *and the proposed* $\boldsymbol{\rho}' \in \mathcal{P}_n$ *with elements*

$$\rho'_i = \begin{cases} \rho^*_u & \text{if } \rho_i = \rho_u \\ \rho_i - 1 & \text{if } \rho_u < \rho_i \leq \rho^*_u \text{ and } \Delta > 0 \\ \rho_i + 1 & \text{if } \rho_u > \rho_i \geq \rho^*_u \text{ and } \Delta < 0 \\ \rho_i & \text{else ,} \end{cases}$$

for $i = 1, \ldots, n$, *constituting the shift step.*

As a first step, we randomly pick index u such that u=2. We choose a small window of only one step to the left and right (L=1), which defines the set of neighbours, excluding the value in position ρ_u. We now determine r randomly take on the value of one of these neighbours: $r \in 1, 2$, e.g. r = 1. We also record the difference between the rank-value in position ρ_u and the randomly chosen neighbour r : $\Delta = r - \rho_u = 1 - 3 = -2$. Once these values have been established, we can go on to reassign the rankings in the list according to the rule above.

We first note that $\rho^*_u = r$, so we can update ρ'_u (in our case ρ'_2) to take on the value of r (in our case 1). The we note that in our example, $\Delta < 0$, so for positions ρ'_1 and ρ'_3 we each add 1 to the ranking values previously attributed to the items in those positions. Items outside the window keep the same rank as before. Thus a full new proposal for the consensus has been generated, with all rankings still in the list and only a subset of them reshuffled.

(a) Previous consensus proposal

(b) New consensus proposal

Figure 1: Illustration of the Leap and Shift algorithm proposed by Vitelli et al (2018).

A probability mass function P_L can be computed associated to this transition between the newly proposed ρ' and the old one. This is then used in the full equation for the acceptance ratio as outlined in the next section.

2.3 Computing the acceptance ratio

The acceptance ration for the Metrolopolis Hastings algorithm can be generalized using the following equation.

$$\alpha(x, y) = min\{1, \frac{p(y)q(x|y)}{p(x)q(y|x)}\}$$

Where:

- $\alpha(x, y)$ is the probability of accepting a proposed state y given the current state x
- $p(x)$ is the probability density function of the current state x
- $p(y)$ is the probability density function of the proposed state y
- $q(x|y)$ is the conditional probability of proposing x given y
- $q(y|x)$ is the conditional probability of proposing y given x
- $\min\{a, b\}$ returns the minimum value between a and b.

For the ρ proposal step, the proposed ρ is accepted with the following probability, where the acceptance ratio is the the term inside the interval.

$$\min\left\{1, \frac{P_L\left(\rho \mid \rho'\right)\pi\left(\rho'\right)}{P_L\left(\rho' \mid \rho\right)\pi(\rho)} \exp\left[-\frac{\alpha}{n}\sum_{j=1}^{N}\left\{d\left(\mathbf{R}_j, \rho'\right) - d\left(\mathbf{R}_j, \rho\right)\right\}\right]\right\}$$

For the α proposal step, the proposed α is accepted with the following probability, where the acceptance ratio for the α parameter is the term within the interval.

$$\min\left\{1, \frac{Z_n(\alpha)^N \pi\left(\alpha'\right)\alpha'}{Z_n\left(\alpha'\right)^N \pi(\alpha)\alpha} \exp\left[-\frac{(\alpha' - \alpha)}{n}\sum_{j=1}^{N} d\left(\mathbf{R}_j, \rho\right)\right]\right\}$$

We accept the proposals sampled with these probabilities, and intuitively, this can be considered as accepting samples from regions of our posterior where the probability of getting the proposed parameter value is higher. Essentially, we will be sampling from the true posterior distribution at convergence.

3 Application to data

The R markdown file contains the full application of the Bayesian Mallows model. It consists of the following parts:

1. **Synthetic data generation:** To check our understanding of the model, we generated data on the basis of a true consensus over a set of 8 film characters. To generate a single observation consisting of an assessor's ranked items, we sample from a multinomial distribution without replacement until all items have been sampled. The order of the ordered probabilities of this multinomial distribution represent the true consensus ranking. We repeat this sampling process for 50 assessors, thereby generating synthetic data based on a true consensus ranking.

2. **Fitting the generated data:** Using the **BayesMallows** package in R, we fit the Bayes Mallows model seen in section two to the generated data. As this package is authored by the same authors of Vitelli, V. (2019), the methods and algorithms used to estimate the posterior distribution are the same- the package utilizes the Metropolis-Hastings MCMC algorithm, and in order to sample from ρ_m the Leap-Shift approach is implemented.

   Below is the pseudo code for this model:

Algorithm 1: Basic MCMC Algorithm for Complete Rankings

input : $\mathbf{R}_1, \ldots, \mathbf{R}_N$; λ, σ_α, α_{jump}, L, $d(\cdot,\cdot)$, $Z_n(\alpha)$, M.
output: Posterior distributions of $\boldsymbol{\rho}$ and α.
Initialization of the MCMC: randomly generate $\boldsymbol{\rho}_0$ and α_0.

for $m \leftarrow 1$ **to** M **do**
- **M-H step: update $\boldsymbol{\rho}$:**
- sample: $\boldsymbol{\rho}' \sim$ L&S($\boldsymbol{\rho}_{m-1}, L$) and $u \sim \mathcal{U}(0,1)$
- compute: $ratio \leftarrow$ equation (6) with $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}_{m-1}$ and $\alpha \leftarrow \alpha_{m-1}$
- **if** $u < ratio$ **then** $\boldsymbol{\rho}_m \leftarrow \boldsymbol{\rho}'$
- **else** $\boldsymbol{\rho}_m \leftarrow \boldsymbol{\rho}_{m-1}$
- **if** $m \mod \alpha_{jump} = 0$ **then M-H step: update α:**
- sample: $\alpha' \sim \log\mathcal{N}(\alpha_{m-1}, \sigma_\alpha^2)$ and $u \sim \mathcal{U}(0,1)$
- compute: $ratio \leftarrow$ equation (8) with $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho}_m$ and $\alpha \leftarrow \alpha_{m-1}$
- **if** $u < ratio$ **then** $\alpha_m \leftarrow \alpha'$
- **else** $\alpha_m \leftarrow \alpha_{m-1}$

end

Where:

- λ is the rate of the exponetial prior. By default this set to .001
- σ_α is the standard deviation of the lognormal proposal distribution used for α. By default this set to .1.
- α_{jump} refers to how many times to sample ρ between each sampling of α. By default this is set to an integer of 1.
- L is the step size of leap-and-shift proposal for ρ.
- d is the right-invariant distance among rankings. We specify the Kendall distance.
- $Z_n(\alpha)$ is the normalizing constant. This is by default set to Null and gets computed during over iterations of the algorithm.
- M is an integer specifying the number of iterations of the Metropolis-Hastings algorithm to run. By default this is set to an integer of 2000. m is the current iteration.
- equation (6) refers to the ratio between the posterior distributions of $\rho\prime$ and ρ_{m-1} given the data, as per the Metropolis Hastings MCMC algorithm.
- equation (8) refers to the ratio between the posterior distributions of $\alpha\prime$ and α_{m-1} given the data, as per the Metropolis Hastings MCMC algorithm.

3. **Model Assessment:** To examine how the model behaves, we utilize a number of functions within the **BayesMallows** package.

   We use the `assess_convergence` function to plot the value of α over the 2000 iterations. We can see that α converges to a fixed region around 3.75 after about 100 iterations. Based on the shape of Figure 1, when plotting the probability distribution of ρ we set the burn-in value to 100, which means that we remove the first 100 observations of ρ in the plot.

   To plot the heatmap of the latent rankings of ρ in Figure 2, we use the `plot.BayesMallows` function. Most of the items' highest probabilities are at their true ranking, while a few have their highest probability at a ranking other than the true one. If we want to examine the confidence with a ranking estimate, we can use the `compute_posterior_intervals` function and specifying the parameter to be "rho", which displays the confidence intervals of the rankings:

4. **Comparison between distance metrics:**

   To compare between distance metrics, we iterate through all possible distances provided by the **BayesMallows** package, and compute performance across different sample sizes from 10 to 500, 20 times for each sample size and distance metric. Figures 4 and 5 display performance with respect to hamming distance and computational time to fit the model, respectively. We

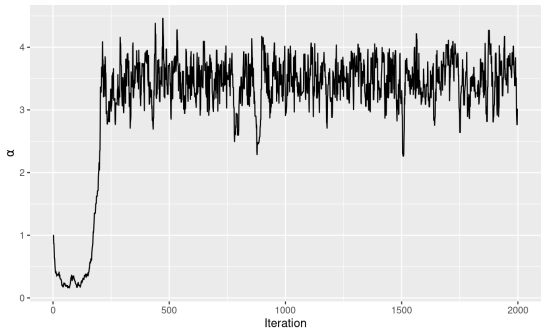


Figure 2: Convergence of alpha over M iterations.

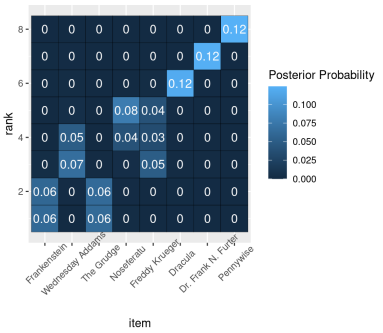
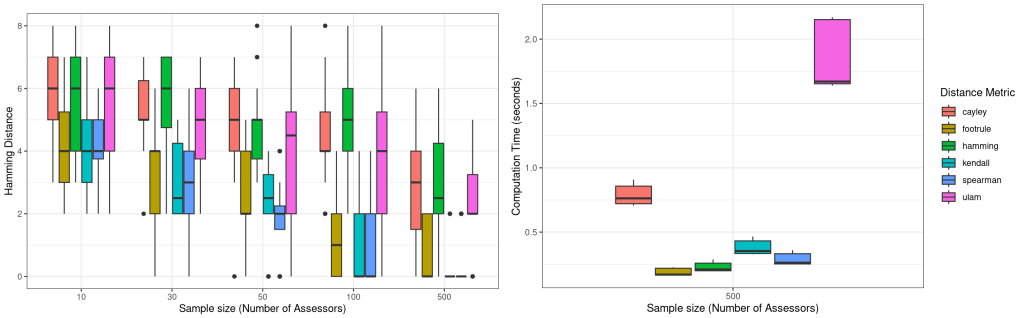


Figure 3: Heatmap of the latent rankings of ρ.

item <fctr>	parameter <chr>	mean <dbl>	median <dbl>	conf_level <chr>	central_interval <chr>
Frankenstein	rho	2	2	95 %	[1,3]
Wednesday Ad...	rho	4	3	95 %	[3,6]
The Grudge	rho	2	2	95 %	[1,2]
Noseferatu	rho	5	5	95 %	[2,5]
Freddy Krueger	rho	4	4	95 %	[3,5]
Dracula	rho	6	6	95 %	[4,6]
Dr. Frank N. F...	rho	7	7	95 %	[7]
Pennywise	rho	8	8	95 %	[8]

Figure 4: The 95% confidence intervals of the ranking estimates

can see that in general footrule, kendall, and spearman distances perform quite well in terms of estimating the true consensus. The ulam and cayle distances perform notably worse in terms of computation time, while the remainder finish in similar times.



(a) Distance from true consensus when increasing the number of assessors.

(b) Computation time for different distance functions.

Figure 5: The effects of changing the sample size on getting close to the true consensus and of different distance metrics on computation time. Each boxplot for (a) is a summary of 20 randomly generated data sets.

4 Conclusion and Reflection

This project has opened up a new area of probabilistic inference for us. We had never studied ranking data before and were able to delve deeper into understanding the Metropolis-Hastings algorithm as an example of an MCMC approach to Bayesian inference. This also helped us reason through a probabilistic framework ranking data aggregation, and has set us up to explore inference from partial rankings, as well as personalised recommender systems in future projects.

There are also many further areas of study on the aggregation of ranking data. Here we mention a few that we found especially interesting. The first is that we may infer from a high α or otherwise

have reason to believe that there is more than one underlying consensus. Figure 6 gives an example where assuming that there is only one underlying consensus would be misguided and would yield unhelpful results. In these situations we need to add a clustering step (e.g. Pérez-Núnez et al (2022)).

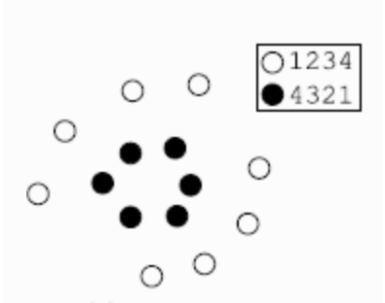


Figure 6: Rankings with more than one underlying consensus.

We may also need to deal with incorrect or incompatible rankings, e.g. if an assessor has somehow given the same rank to two different items. Our understanding is that this could be addressed within the Mallows model. If items A_1 and A_2 have both been given the same rank, we could use a consensus derived from the other assessors' data to determine which one we expect to have the higher rank and thus update the the ranking accordingly. We could even make this update probabilistic - making the update in line with the interim consensus with a higher probability when our estimate for α is low.

There have also been attempts at deriving a consensus with missing data, i.e. partial rankings from assessors. This would be of interest in the previously given example of creating a consensus ranking on lots of exams when different subsets of exams have been marked through pairwise comparisons. In this situation, when calculating the distance from the true consensus, we would need to restrict the central ranking to each corresponding selection set (Fotakis et al (2020)). There will also be a minimum requirement for pair appearance: every pair of alternatives needs to appear is at least some fraction of the ranking sets.

Finally, there are issues with different assessor qualities. This is especially interesting to contrast with the case for clustering. When a subset of assessors disagrees with the others, one possible interpretation is that their rankings are derived from a different consensus. It would be a qualitatively different approach to think of them as 'wrong' - and it may imply different quantitative measures of giving new assessors that are further away from the consensus a lower weight when updating the consensus.

References

Fotakis, D., Kalavasis, A., Stavropoulos, K. (2020). Aggregating Incomplete and Noisy Rankings. , *24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021*. https://doi.org/10.48550/arXiv.2011.00810

Grbovic, M., Djuric, N., Guo, S. et al. (2013). Supervised clustering of label ranking data using label preference information. , *Mach Learn 93, 191{225*. https://doi.org/10.1007/s10994-013-5374-3

Liu, Q., Crispino, M., Scheel, I., Vitelli, V., and Frigessi, A. (2019). Model-based learning from preference data. *Annual Reviews of Statistics and its Application, , 6 (1), pp.329-354.*. https://hal.science/hal-01972948

Øystein, S., Liu, Q., Crispino, M., Scheel, I., Vitelli, V. (2019). BayesMallows: An R Package for the Bayesian Mallows Model. , *Journal of Machine Learning Research. 18. 1-49*. https://doi.org/10.48550/arXiv.1902.08432

Pérez-Núnez, P., Díez, J., Luaces, O. et al. (2022). User encoding for clustering in very sparse recommender systems tasks. , *Multimed Tools Appl 81, 2467{2488* . https://doi.org/10.1007/s11042-021-11564-x

Vitelli, V., Øystein, S., Crispino, M., Frigessi, A., Arjas, E. (2018). Probabilistic preference learning with the Mallows rank model. , *Journal of Machine Learning Research. 18. 1-49*. https://jmlr.org/papers/v18/15-481.html