Journal of Statistical Software

July 2019, Volume 90, Issue 5. *doi: 10.18637/jss.v090.i05*

Weighted Distance-Based Models for Ranking Data Using the R Package rankdist

Zhaozhi Qian
The University of Hong Kong

Philip L. H. Yu
The University of Hong Kong

Abstract

rankdist is a recently developed R package which implements various distance-based ranking models. These models capture the occurring probability of rankings based on the distances between them. The package provides a framework for fitting and evaluating finite mixture of distance-based models. This paper also presents a new probability model for ranking data based on a new notion of weighted Kendall distance. The new model is flexible and more interpretable than the existing models. We show that the new model has an analytic form of the probability mass function and the maximum likelihood estimates of the model parameters can be obtained efficiently even for ranking involving a large number of objects.

Keywords: ranking data, distance-based models, Kendall distance, mixtures models, rank aggregation, R.

1. Introduction

Ranking data occur when raters are asked to rank order a set of objects. Examples of ranking data arise in elections, movie rankings, shopping preferences and so on. By analyzing ranking data, we may want to understand the patterns of rank-order preferences of raters and to find the most representative ranking generally consented by the raters. Ranking data analysis thus has important applications in market research, recommendation systems and political sciences.

In a survey paper, Critchlow (1986) broadly categorized probability models for ranking data into four classes: (1) order statistics models, (2) paired comparison models, (3) distance-based models, and (4) multistage models. For more details, please refer to the monograph by Alvo and Yu (2014). Among them, the order statistics models have the longest history. Typical examples of these include independent order statistics models (Thurstone 1927; Luce 1959) and multivariate order statistics models (Yu 2000; Joe 2001). The basic idea behind these models is that each rater assigns a latent score or utility to each object according to his/her perception of the object and the ordering of these utility scores then determines the rater's ranking of the objects. While this approach works well when the number of objects is small, its parameter estimation often becomes computationally demanding when a large number of objects are ranked. Both the paired comparison models (Smith 1950; Mallows 1957) and the multistage models (Fligner and Verducci 1988) try to decompose the ranking process into a set of independent decisions. The assumed independence between decisions greatly simplifies the computation. A common critique of these two approaches is that the underlying cognitive model of assigning rankings may not always be true. Distance-based models (Diaconis 1988) rely on distance metrics between two rankings. They were not as popular as the above three models until recently because they were thought to be too inflexible. This paper introduces the R package **rankdist** (Qian 2019) which provides a unified way to fit different distance-based ranking models. This paper also presents an original model which has an elegant analytic form of the probability mass function.

The paper is organized as follows. In Section 2, we review the general formulation of distance-based models and several well-known examples. We then introduce the weighted Kendall distance and formulate the weighted Kendall distance model in Section 3. Section 4 shows possible ways to extend distance-based models. We review existing software for ranking model and give details on the package architecture in Section 5. In Section 6, we give concrete examples of using package **rankdist** to model complete and top-q rankings. In Section 7, we first validate the parameter estimation procedure with a comprehensive simulation study. We then present two real-data applications of weighted Kendall distance model using **rankdist**. We compare the results with other ranking models in terms of goodness of fit and interpretability. Finally, we discuss the strengths of our new model and directions for future development of the **rankdist** package.

Before we introduce the model in full details, it is helpful for us to review some important conventions of permutation notations. In the rest of this paper, permutations are denoted as lower-case Greek letters π, σ, τ etc. $\pi(i)$ denotes the rank given to object i, and $\pi^{-1}(i)$ denotes the object assigned the rank i. The total number of objects in the ranking is t. For any two permutations π and σ, the product $\tau = \pi\sigma$ is defined by the equation $\tau(i) = \pi(\sigma(i))$, $i = 1, 2, \ldots, t$.

2. Distance-based ranking models

2.1. Overview of distance-based models

Sometimes it is reasonable to assume that there exists a modal ranking π_0 which has the highest probability to occur and most observed rankings rm are close to π_0. Thus we should assign the highest probability to π_0 and for any other ranking the probability should be negatively correlated with its distance from π_0. According to this framework, Diaconis (1988) proposed a family of distance-based model

$$\mathsf{P}(\pi \mid \lambda, \pi_0) = \frac{e^{-\lambda D(\pi, \pi_0)}}{C(\lambda)}, \tag{1}$$

where π_0 denotes the modal ranking, $D(\pi, \pi_0)$ is a distance function between π and π_0, $\lambda \geq 0$ is the dispersion parameter and $C(\lambda)$ is the normalization constant. The common properties of $D(\pi, \pi_0)$ are: (1) reflexivity, $d(\pi, \pi) = 0$; (2) positivity, $d(\pi, \sigma) > 0$ if $\pi \neq \sigma$; and (3) symmetry, $d(\pi, \sigma) = d(\sigma, \pi)$. For ranking data, the distance should also satisfy an additional property: (4) right-invariance, $d(\pi, \sigma) = d(\pi\nu, \sigma\nu)$ for any permutation π, σ and ν. This ensures that a relabeling of the objects has no effect on the distance. If a distance satisfies the triangular inequality: (5) $d(\pi, \nu) \leq d(\pi, \sigma) + d(\sigma, \nu)$, the distance function is said to be a metric. The following are some commonly used distance metrics. Each of these distances can be formulated into a ranking model as in Equation 1.

- Kendall distance:

  $$D_K(\pi, \sigma) = \sum_{i<j} I\left\{[\pi(i) - \pi(j)]\,[\sigma(i) - \sigma(j)] < 0\right\},$$

  where $I\{\cdot\}$ is the indicator function taking values 1 or 0 depending on whether the statement in brackets holds or not.

- Spearman distance:

  $$D_S(\pi, \sigma) = \frac{1}{2}\sum_{i=1}^{t} [\pi(i) - \sigma(i)]^2$$

- Hamming distance:

  $$D_H(\pi, \sigma) = \sum_{i=1}^{t} I\left\{\pi(i) \neq \sigma(i)\right\}$$

- Footrule distance:

  $$D_F(\pi, \sigma) = \sum_{i=1}^{t} |\pi(i) - \sigma(i)|$$

- Cayley distance: D_C is defined as the minimum number of transpositions needed to transform one ranking into the other.

The existing distance-based models often have one or more of the following limitations.

- The normalization constant $C(\lambda)$ is intractable. The general way of finding $C(\lambda)$ involves the summation of $t!$ terms, which is not computationally feasible when the number of items is large (>10). As a result, the log-likelihood will also be intractable, which poses challenge for inference and parameter estimation.

- There is only one parameter λ controlling the dispersion of the model, which is too restrictive in some cases.

- The distance function is not a metric, which makes the model unintuitive and less interpretable.

- It might be hard to generate samples from the model. The general way of drawing samples is to calculate probability for all $t!$ rankings and sample from the corresponding categorical distribution. This approach is not computationally feasible for $t > 10$ even if the normalization constant is tractable because it involves calculating and storing $t!$ probabilities.

In this paper, we present the weighted Kendall model which addresses all four issues mentioned above. The model is based on a proper distance metric, the weighted Kendall distance. The dispersion parameters are directly related to positions of the ranked list, which is flexible and interpretable. Evaluating the normalization constant has time complexity $O(t)$, and obtaining samples takes $O(t^2)$. Both are huge improvements over the $O(t!)$ complexity.

We will first review several existing distance-based models in order to motivate the weighted Kendall model.

2.2. Mallows' ϕ model

Among the distance functions listed in the previous section, the Kendall distance is the most popular one. Its distance-based model, also called the Mallows' ϕ model (Mallows 1957), has a closed-form normalization constant:

$$C(\lambda) = \prod_{i=1}^{t-1} \frac{1 - e^{-(t-i+1)\lambda}}{1 - e^{-\lambda}}.$$

Kendall distance is a metric and it can also be interpreted as the minimum number of adjacent transpositions required to transform π^{-1} to σ^{-1}.

The Mallows' ϕ-model has only one dispersion parameter λ. As a results, the transpositions between any two items at any position in the list will have the same effect to the probability, which might seem inflexible in some cases. For example, swapping items near the beginning of the list should have a larger effect than swapping items at the end of the list when the rater pays more attention to the top items. Another example will be swapping radically different items might have a larger effect than swapping very similar items even if the adjacent transposition takes place at the same position. As a final example, some items might carry higher importance than others and swapping items with higher importance should have a larger effect to the probability. The Mallows' ϕ-model can capture none of these subtleties. Kumar and Vassilvitskii (2010) presents more examples and several generalized distance measures. Efforts have also been taken to introduce more richness into distance-based models, and we will describe two such examples below.

2.3. The ϕ-component model

The ϕ-component model proposed by Fligner and Verducci (1986) is a generalization of Mallows' ϕ-model. Its basic idea is that the Kendall distance can be decomposed into a sum of $t-1$ scores $V_i(\pi, \sigma), i = 1, \ldots, t-1$ obtained in a ranking process:

$$D_K(\pi, \sigma) = \sum_{i=1}^{t-1} V_i(\pi, \sigma),$$

where

$$V_i(\pi, \sigma) = \sum_{j=i+1}^{t} I\{[\pi(\sigma^{-1}(i)) - \pi(\sigma^{-1}(j))] > 0\}. \tag{2}$$

Here $V_1(\pi, \sigma)$ represents the number of adjacent transpositions on π required to place the object $\sigma^{-1}(1)$ (the object ranked first in σ) in the first position in ranking π. In the ith stage $(2 \le i \le t-1)$, $\pi^{-1}(j) = \sigma^{-1}(j)$ for $j = 1, \ldots, i-1$, and $V_i(\pi, \sigma)$ is the number of adjacent transpositions needed to place the object $\sigma^{-1}(i)$ in the ith position in ranking π. Therefore, the ranking can be described as $t-1$ stages where $V_i(\pi,\sigma)$ can be interpreted as the number of mistakes made in assigning rank to object $\sigma^{-1}(i)$ after $(i-1)$ stages and it takes value $0, 1, \ldots, (t-i)$.

The ϕ-component model introduces dispersion parameter λ_i for each $V_i(\pi, \pi_0)$, and takes the form of:

$$\mathsf{P}(\pi \mid \boldsymbol{\lambda}, \pi_0) = \frac{e^{-\sum_{i=1}^{t-1} \lambda_i V_i(\pi,\pi_0)}}{C(\boldsymbol{\lambda})}, \tag{3}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{t-1})'$ and $C(\boldsymbol{\lambda})$ is the normalization constant with a closed-form, which equals

$$C(\boldsymbol{\lambda}) = \prod_{i=1}^{t-1} \frac{1 - e^{-(t-i+1)\lambda_i}}{1 - e^{-\lambda_i}} \equiv \prod_{i=1}^{t-1} C(\lambda_i).$$

Thus Equation 3 can be simplified as

$$\mathsf{P}(\pi \mid \boldsymbol{\lambda}, \pi_0) = \prod_{i=1}^{t-1} \frac{e^{-\lambda_i V_i(\pi,\pi_0)}}{C(\lambda_i)},$$

which is a joint probability mass function of $t-1$ statistically independent variables $V_1, \ldots, V_{t-1}$.

The term $\sum_{i=1}^{t-1} \lambda_i V_i$ in the ϕ-component model extends the notion of Kendall distance between two rankings and it degenerates to Kendall distance if all λ_i's are the same. Decomposing the ranking process into statistically independent variables also opens up new interpretations of model parameters.

However, ϕ-component model violates the symmetric property of distance metric as shown in Lee and Yu (2012). For example, we have ranking $\pi^{-1} = A \mid C \mid B$ and $\sigma^{-1} = C \mid B \mid A$, then $D(\pi,\sigma) = \lambda_1 + \lambda_2$ while $D(\sigma,\pi) = 2\lambda_1$. It's clear that they are not equal when $\lambda_1 \neq \lambda_2$. The violation of symmetry poses an immediate issue: the distance between modal ranking π_0 and one observation will not be the same as the distance between the observation and π_0. Hence, it's hard to interpret π_0 as a "central" ranking in a geometric sense.

2.4. Weighted-tau distance model

Lee and Yu (2012) proposed new distance-based models by using weighted distance measures to allow different weights for different ranks. The properties (1)-(4) of a distance function are preserved. The weighted version of Kendall distance between two rankings π and σ with the modal ranking π_0 and weights $\boldsymbol{w} = (w_1, \ldots, w_t)$ under this model is defined as

$$D_{wt}(\pi, \sigma \mid \pi_0, \boldsymbol{w}) = \sum_{i<j} w_{\pi_0(i)} w_{\pi_0(j)} I \left\{ [\pi(i) - \pi(j)] \, [\sigma(i) - \sigma(j)] < 0 \right\}.$$

Lee and Yu (2012) called this distance as weighted Kendall tau distance or simply weighted tau distance. Note that the weighted tau distance is not equivalent to any of the generalized distances proposed in Kumar and Vassilvitskii (2010). One obvious difference is that the weighted tau distance is parameterized by a set of weights as well as the central ranking π_0 while the distances in Kumar and Vassilvitskii (2010) do not take into account π_0.

The probability of observing a ranking π under the weighted tau distance-based ranking model is

$$\mathsf{P}(\pi \mid \boldsymbol{w}, \pi_0) = \frac{e^{-D_{wt}(\pi,\pi_0|\pi_0,\boldsymbol{w})}}{C_{wt}(\boldsymbol{w})},$$

where the weight $w_i w_j$ represents the loss in the disagreement in the ranking of the two objects $\pi_0^{-1}(i)$ and $\pi_0^{-1}(j)$ between π and π_0.

The shortcoming of this model is that the normalization constant $C_{wt}(\boldsymbol{w})$ does not have a closed form. The exact probability and likelihood become easily intractable as the time complexity of evaluating them is $O(t!)$.

3. Weighted Kendall distance nodel

3.1. Motivation and definition of weighted Kendall distance

The new model presented in this paper is motivated by another generalization of Kendall distance first introduced in Kumar and Vassilvitskii (2010) and later axiomized by Farnoud and Milenkovic (2014). The distance has been shown to be a distance metric, and we will refer to it as the weighted Kendall distance.

As a motivating example, consider three orderings, $\sigma^{-1} = A \mid B \mid C \mid D$, $\pi_1^{-1} = A \mid B \mid D \mid C$ and, $\pi_2^{-1} = B \mid A \mid C \mid D$. In terms of the original Kendall distance: $D(\pi_1, \sigma) = D(\pi_2, \sigma) = 1$, but the difference between π_1 and σ occurs at the bottom of the list while the difference between π_2 and σ occurs in the first two positions. In reality, raters may care more about the top positions than the bottom positions. A concrete example will be people's perception of the outcome of the Olympic games: in general the public care more about who is the winner than who has ended up in the fourth place. In this case, $D(\pi_1, \sigma)$ should be smaller than $D(\pi_2, \sigma)$. The weighted Kendall distance is developed to capture this phenomenon.

Several additional definitions are needed for a clear presentation of the distance. A transposition $\tau_{a,b}$, $a, b < t$ is defined as a permutation such that $\tau(a) = b$ and $\tau(b) = a$ and $\tau(c) = c$ for all $c \neq a$,b. In particular, $\tau_{a,a+1}$ is referred to as an adjacent transposition, and will be simply denoted as τ_a.

We associate a non-negative weight w_i with each adjacent transposition $\tau_i, i = 1, \ldots, t-1$. Define the set

$$A(\pi, \sigma) = \{\langle \tau_{(1)}, \tau_{(2)}, \ldots \rangle \mid \sigma = \pi \tau_{(1)} \tau_{(2)} \ldots\}$$

to be the set of all ordered sequences of adjacent transpositions that transform π to σ. In the above definition $\tau_{(i)}$ is the adjacent transposition taking place in the ith step. Note that the sequences in $A(\pi, \sigma)$ might have different lengths. The weighted Kendall distance between two rankings π and σ is defined as

$$D_{wK}(\pi, \sigma) = \min_{\langle \tau_{(1)}, \tau_{(2)}, \ldots \rangle \in A(\pi,\sigma)} \sum_i w_{(i)}, \tag{4}$$

where $w_{(i)}$ is the weight associated with the ith adjacent transposition $\tau_{(i)}$ applied to σ in the sequence. In a nutshell, the weighted Kendall distance corresponds to the best way to transform ranking π to σ via adjacent transformation.

The weights w_j's have the natural interpretation as the "difficulty" of transposing adjacent objects at certain rank positions in the ranking. If we define $Q_j(\pi, \sigma)$ to be the total number of adjacent transposition τ_j applied to σ in the minimizer, the distance can be explicitly represented as a linear combination of weights:

$$D_{wK}(\pi, \sigma) = \sum_{j=1}^{t-1} w_j Q_j(\pi, \sigma). \tag{5}$$

3.2. Weighted Kendall distance as a graphic distance

Similar to the original Kendall distance, the weighted Kendall distance is also a graphic distance, which means the distance between π and σ is the length of the shortest path between vertices π and σ on a certain graph. The vertices of such a graph are all possible rankings π's (or orderings π^{-1}'s) of t objects and two vertices are linked by an edge if the corresponding orderings of the objects are different only by one adjacent transposition. In the context of weighted Kendall distance, if the edge corresponds to adjacent transposition τ_i this edge has a weight w_i. By definition, the shortest path minimizes the right hand side of Equation 4. It follows that when all the w_i's equal 1, the weighted Kendall distance degenerates to the original Kendall distance.

Figure 1 provides an illustration of paths used in weighted Kendall distance. The label on each vertex is an ordering. The color of an edge represents the weight associated with that edge. A blue edge represents an transposition between the top two objects and is associated with weight w_1. Likewise, a green edge is associated with w_2, and red edge w_3. Suppose we are interested in finding the weighed Kendall distance between the two orderings A|B|C|D and D|C|A|B. The shortest path $\langle \tau_{(1)}, \tau_{(2)}, \ldots \rangle$ is not unique in this case and is marked with arrows. The Q_j in Equation 5 corresponds to the number of edges with a certain color on the shortest path. There is one blue edge, two green edges and two red edges on the path, so $Q_1 = 1, Q_2 = 2, Q_3 = 2$. Although there could be more than one shortest paths, the value of Q_j's are the same for all these paths and hence the distance is uniquely specified as expected.

3.3. Computing weighted Kendall distance

Applying shortest path algorithms on the graph introduced in Section 3.2 to compute distances is not computationally feasible in general since the graph has $t!$ vertices. Farnoud and Milenkovic (2014) showed that if the weights are decreasing, the distance computation between two rankings of t objects can be performed efficiently with time complexity $O(t^2)$. Surprisingly, the sequence of adjacent transpositions used to iteratively find V_i in the ϕ-component model in Equation 2 is the minimizer of Equation 4 (Farnoud and Milenkovic 2014).

3.4. The normalization constant

The weighted Kendall distance introduced in Section 3.3 can be formulated into a probability model. For the ranking of t objects, the model includes $t - 1$ decreasing weights as well as a modal ranking π_0. We refer to this new model as the weighted Kendall model. The

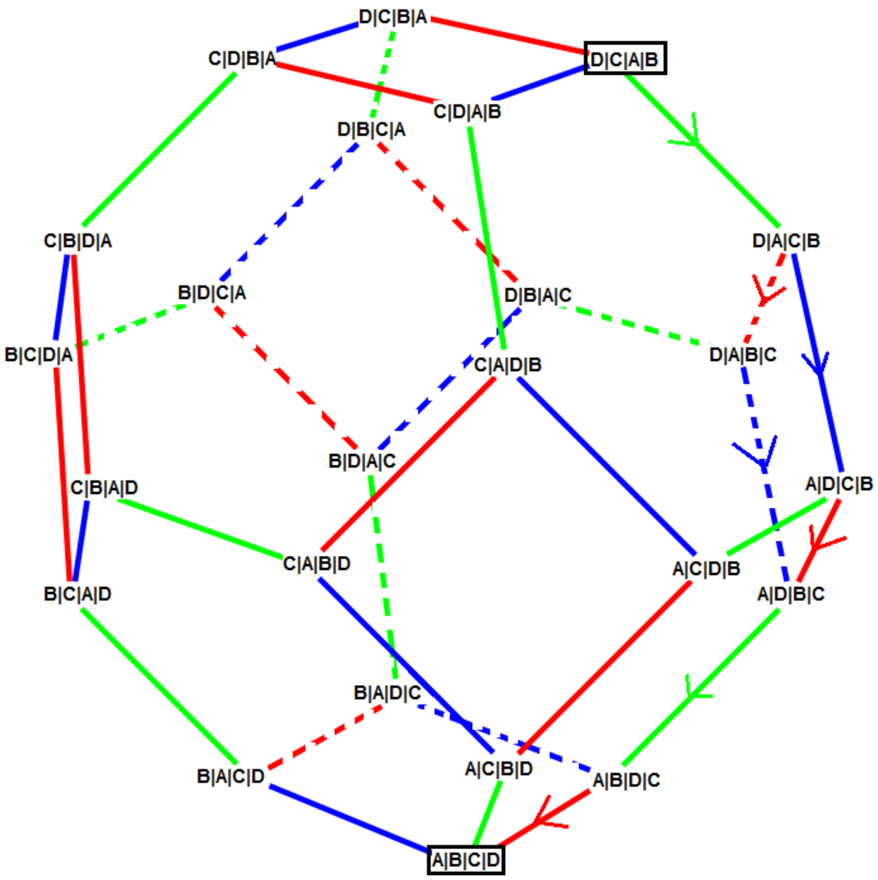

Figure 1: Graphical representation of weighted Kendall distance between four objects.

probability of observing a ranking π is

$$\mathsf{P}(\pi \mid \boldsymbol{w}, \pi_0) = \frac{e^{-D_{wK}(\pi,\pi_0)}}{C_{wK}(\boldsymbol{w})},$$

where $C_{wK}(\boldsymbol{w})$ is the normalization constant. Although the distance term in the model can be computed efficiently in $O(t^2)$, the exact probability would still be intractable if the normalization constant does not have a closed form. Fortunately due to the special structure of the model we show that $C_{wK}(\boldsymbol{w})$ has a closed form and is given as follows.

$$C_{wK}(\boldsymbol{w}) = \prod_{i=1}^{t-1}\left[1 + \sum_{j=1}^{i} e^{-\sum_{k=j}^{i} w_{t-k}}\right]. \tag{6}$$

The derivation can be found in Appendix A. Despite having multiple summations and products, the normalization constant can be evaluated very efficiently in $O(t)$ complexity via a recursion. The derivation is also given in Appendix A.

By the composition rule of convex functions one can verify that $\log(C_{wK}(\boldsymbol{w}))$ is a convex function of w. This property has important implications when we estimate w with maximum likelihood method.

Algorithm 1 Estimating central ranking with known weights.

```
Inputs: true weights $w$ and initial ranking $\pi_{00}$

  1. Initialize candidate ranking to be $\pi_{00}$
  2. Evaluate the log-likelihood with $\pi_0$ set to the candidate ranking
  3. Evaluate the log-likelihoods with $\pi_0$ set to candidate ranking's neighbors.
     (rankings which are different from the candidate ranking by one adjacent transposition)
  4. If the candidate ranking achieves highest log-likelihood among its neighbors,
     return the candidate ranking. Otherwise set the candidate ranking to be the
     neighbor that has highest log-likelihood and repeat 2–4.
```

3.5. Parameter estimation

Estimating central ranking π_0

Given a set of weights w and n observed rankings $\sigma_1, \ldots, \sigma_n$, finding the maximum likelihood estimate of central ranking π_0 is equivalent to minimizing the total distance between π_0 and the observed rankings Equation 7.

$$\pi_0 = \arg\min \left\{ \sum_{i=1}^{n} D_{wK}(\pi_0, \sigma_i) \right\} \tag{7}$$

Dwork, Kumar, Naor, and Sivakumar (2001) studied this problem where the distance metric is the original Kendall distance, i.e., the weights are all equal. They showed that even when $n = 4$, the problem is NP-complete. We have seen little evidence suggesting that introducing different weights will reduce the complexity of this problem. Therefore, we have to resort to an optimization algorithm that finds a local optimum. A ranking π is a local optimal of problem in Equation 7 if there is no ranking π' that can be obtained from π by performing a single adjacent transposition and having a smaller total distance.

We propose a heuristic algorithm to find local optimums (Algorithm 1). The algorithm is provided with an initial value of π_0, denoted as π_{00}. This initial value could be a frequently observed ranking or the result of a rank aggregation algorithm such as the Borda count algorithm (Marden 1995). The algorithm repeatedly checks the neighbors of the candidate ranking until the log-likelihood can no longer be improved. Since in each step the objective function in Equation 7 always decreases and there are only a limited number of π_0's to choose from, the algorithm will stop eventually. In Section 7.1 we evaluate this heuristic algorithm with simulation studies. The simulation result shows that when the observed rankings are generated by weighted Kendall model, the algorithm is likely to find the true ranking even when the number of objects are large (~40) and the sample size is small (~200).

Estimating weights w

The weights in the weighted Kendall distance are closely related to the "dispersion" of rankings. They also indicate the relative importance of locations in the ranked list.

Recall that the log-likelihood function of the weighted Kendall model is

$$\ell(\pi_1, \pi_2, \ldots, \pi_n) = -\sum_{i=1}^{n}\sum_{j=1}^{t-1} w_j Q_j(\pi_i, \pi_0) - n\log(C_{wK}(\boldsymbol{w})),$$

where $Q_j(\pi_i, \pi_0)$ are defined as in Equation 5, and they are constants if π_0 is given. In Section 3.4 we showed that the logarithm of the normalizing constant $C_{wK}(\boldsymbol{w})$ is a convex function in w. It follows that the log-likelihood function is a concave function in w. Since the non-decreasing constraint on w is a linear inequality constraint, the problem of finding maximum likelihood estimate of w is a convex optimization problem, which has a global optimal solution.

In practice, we reparameterized w_j as $w_j = \sum_{i=j}^{t-1} \phi_i$, where $\phi_i \geq 0$ for all i, and transform the non-increasing constraint on w into a box constraint on ϕ. The log-likelihood function is still concave and it can be rewritten in terms of ϕ as:

$$\ell(\pi_1, \pi_2, \ldots, \pi_n) = -\sum_{i=1}^{n}\sum_{j=1}^{t-1} \phi_j \left[\sum_{k=1}^{j} Q_k(\pi_i, \pi_0)\right] - n\log(C_{wK}(\boldsymbol{w}(\boldsymbol{\phi}))).$$

With the simplified constraint, we can apply a general convex optimization solver to estimate w. The L-BFGS-B method in the R package **optimx** (Nash 2014) is used for the optimization in **rankdist**. The simulation studies in Section 7.1 confirms that the optimization procedure is reliable.

Estimating central ranking π_0 and weights w jointly

In most cases, neither π_0 nor w is known to us and both need to be estimated from the data. It is hard to estimate π_0 and w simultaneously because π_0 is a ranking while w is a vector of real numbers. Instead, we apply a stage-wise method that iteratively searches for the two parameters (Algorithm 2).

The algorithm is provided with an initial value of π_0, denoted as π_{00}, which can be the result of any rank aggregation algorithm. Similar to Algorithm 1, the algorithm repeatedly check the neighbors of current candidate ranking until log-likelihood can no longer be improved. The algorithm will stop eventually because there are only a limited number of candidate rankings.

We evaluate the joint optimization algorithm in Section 7.1 with simulation studies. The simulation result shows that when the observed rankings are generated by weighted Kendall model, the algorithm is likely to find the true ranking and the true weights even when the number of objects are large (~40) and the sample size is relatively small (~500).

4. Extension of distance-based models

4.1. Mixture of distance-based models

Introducing additional parameters into the distance-based ranking models increases the flexibility of the models. However, all the models presented in the previous sections are strongly uni-modal. Fortunately these models can be easily extended to a finite mixture of several

Algorithm 2 Estimating central ranking and weights.

```
Inputs: initial ranking $\pi_{00}$
    1. Initialize candidate ranking to be $\pi_{00}$
    2. Estimate the weights and evaluate the log-likelihood with $\pi_0$ set to the candidate ranking
    3. Estimate the weights and evaluate the log-likelihoods with $\pi_0$ set to candidate ranking's neighbors. (rankings which are different from the candidate ranking by one adjacent transposition)
    4. If the candidate ranking and weights achieves highest log-likelihood among its neighbors, the algorithm stops. Otherwise set the candidate ranking to be the neighbor that has highest log-likelihood and repeat 2-4.
```

Algorithm 3 EM-algorithm for fitting mixture models.

```
    1. E-Step: compute the value of $z_g$ for each observation $i$:
           $\hat{z}_g^{(i)} = \frac{p_g \mathsf{P}(\pi^{(i)} \mid \boldsymbol{w}_g, \pi_{0,g})}{\sum_{g'=1}^{G} p_{g'} \mathsf{P}(\pi^{(i)} \mid \boldsymbol{w}_{g'}, \pi_{0,g'})}$
    2. M-Step: for each component $g$, estimate $\pi_{0,g}$ and $w_g$ in the same way as for the case of single component except that each observation $i$ has a 'discounted' frequency $\hat{z}_g^{(i)}$.
    3. Repeat the above steps until convergence.
```

distance-based component models in order to cater for the heterogeneity of the rank-order preferences among the raters. Each component model may represent a different group of raters with their own favorite/modal ranking ($\pi_{0,g}$) and weights ($\boldsymbol{w}_g$) used in the distance function. The mixture model of G components can be defined as:

$$\mathsf{P}(\pi_i) = \sum_{g=1}^{G} p_g \mathsf{P}(\pi_i \mid \boldsymbol{w}_g, \pi_{0,g}),$$

where p_g is proportion of raters in the gth component.

Murphy and Martin (2003) first applied the EM-algorithm to efficiently fit such mixture models. We introduce latent variables z to record the component membership of each observation. The latent (membership) variable $z = (z_1, z_2, \dots, z_G)$ is defined such that $z_g = 1$ if the observation certainly belongs to component g and $z_g = 0$ otherwise. The EM-algorithm is summarized in Algorithm 3.

4.2. Top-q rankings

If the raters evaluate all objects but only report the rankings of the best q objects then we obtain top-q ranking data. Top-q ranking data is very common when the number of objects to be ranked is large. The top-q ranking can be viewed as a missing data problem, where additional assumptions about the structure of missing data is needed.

A common assumption roots in the maximum entropy principle. According to this principle, all complete rankings that are compatible with the observed top-q ranking have the same probability to occur. We will refer to this assumption as the equal-probability assumption. It follows that in terms of likelihood function, observing a top-q ranking $\pi(q)$ is equivalent to assigning weight $\frac{1}{(t-q)!}$ to all complete rankings compatible with $\pi(q)$. In this way the top-q rankings are transformed into complete rankings and any model that works for complete rankings can be applied. Note that this approach works for missing ranking in general, i.e., the missing rankings do not need to occur at the bottom of the list. However, this approach has issues with scalability. For a top-q ranking, the number of all its compatible rankings is $(t-q)!$. Evaluating all of them has huge computational cost when $t-q$ is big.

The weighted Kendall model can avoid such problem if an additional assumption is imposed. This assumption is that the raters consider all unreported objects to be equally bad, or in other words, those unreported objects have tied rank $q+1$. We will refer to this assumption as the tied-rank assumption. Under this assumption we are able to assign $w_j = 0$ for all $j > q$, which means swapping objects with rank larger than q does not affect the likelihood (since they are tied). It follows from the definition of the weighted Kendall distance that if $w_k = 0$ for all $k > q$,

$$D_{wK}(\pi_i, \sigma) = D_{wK}(\pi_j, \sigma)$$

Here σ can be any complete ranking and π_i π_j are two complete rankings compatible with $\pi(q)$. Therefore, with $w_k = 0$ for all $k > q$ we can define $D_{wK}(\pi(q), \sigma) = D_{wK}(\pi, \sigma)$ where π is any complete ranking compatible with $\pi(q)$.

The algorithm for evaluating distance between top-q rankings is presented in the box "Algorithm 4". Note that the distance computation can be optimized to have time complexity $O(q^2)$ and the computational cost is no longer related to the number of objects (t), which is an extremely desirable property in applications where a large number of objects are compared. We will show in Appendix B that the normalization constant $C_q(\boldsymbol{w})$ for top-q rankings is proportional to the normalization constant $C_{wK}(\boldsymbol{w})$ for complete rankings. In particular $C_q(\boldsymbol{w}) = \frac{C_{wK}(\boldsymbol{w})}{(t-q)!}$.

Note that the tied-rank assumption is stronger than the equal-probability assumption. In some applications the tied-rank assumption may not be valid and the resulting model would not be as good. However this assumption greatly simplifies computation when $t-q$ is big.

If a data set contains both complete and top-q rankings for m distinct values of q, the model can still be adapted to such heterogeneity. We can write the likelihood of the whole data set as follows:

$$\begin{aligned} \ell(\vec{\pi}(q_1), \vec{\pi}(q_2), \ldots, \vec{\pi}(q_m)) &= \log \prod_{i=1}^{m} \mathsf{P}(\vec{\pi}(q_i)) \\ &= \sum_{i=1}^{m} \left[-\sum_{k=1}^{n_{q_i}} D_{wK}(\pi_k(q_i), \pi_0) - n_{q_i} \log(C_{q_i}) \right] \end{aligned} \tag{8}$$

where $\vec{\pi}(q_i)$ is the sample of all the top-q_i rankings in the data set and n_{q_i} is the sample size of $\vec{\pi}(q_i)$.

Algorithm 4 Computing weighted Kendall distance between top-q rankings.

```
for r in 1 to q:
    obj = the object with rank r in pi_1
    pos = UNFOUND
    for r2 in 1 to q:
     if obj has rank r2 in pi_2:
         pos = r2
    if pos == UNFOUND:
     increment each weight
     increment the rank of each remaining object in pi_2
    else:
     for j in r to (pos-1):
         increment weight j
         increment rank j in pi_2
return weight
```

5. Package architecture and implementation

The **rankdist** package is available from the Comprehensive R Archive Network (CRAN) at https://CRAN.R-project.otg/package=rankdist. The latest development version can be found at https://github.com/ZhaozhiQIAN/rankdist. The software is written in R (R Core Team 2019) and C++.

5.1. Existing software for modeling ranking data

At present, there are several alternative packages for modeling ranking data in R including **PerMallows** (Irurozki, Calvo, and Lozano 2016), **Rankcluster** (Grimonprez and Jacques 2014), **pmr** (Lee and Yu 2015) and **mlogit** (Croissant 2019).

Package **PerMallows** implements six common distance-based models. The key features that distinguish **rankdist** from **PerMallows** are (1) **rankdist** supports mixture models; (2) the more compact representation of ranking data set; and (3) extendability. The components in **rankdist** are modularized so that minimum effort is needed to implement a new model. Hence, **rankdist** is very suitable for developing and testing new ranking models. Package **PerMallows** has two functionality that are currently unavailable in **rankdist**. It contains abundant high-performance utility functions to perform calculations related to permutations and it includes different ways to draw samples from the fitted model.

To our knowledge, **Rankcluster** and **rankdist** are the only two well-maintained R packages that provide functionality to fit mixture model for ranking data. Package **Rankcluster** implements one ranking model, the ISR model proposed in Biernacki and Jacques (2013). Evaluating the exact probability of a ranking in ISR model involves summing up $t!$ terms, which is intractable when the number of items is greater than twenty. Biernacki and Jacques (2013) proposed a MCMC (Markov chain Monte Carlo) based algorithm to approximate the likelihood in large-t situation. The algorithm is implemented in **Rankcluster** as the default inference method. The user needs to tune and specify the sample size, sampling iteration, burn-in period and several other parameters to control the behavior of MCMC sampling. Package **Rankcluster** is able to model partial rankings in general while **rankdist** only supports top-q rankings. The rankings of unrated items are treated as missing variables in **Rankcluster**. They are imputed by reusing the MCMC samples drawn in the inference step. In Section 7.2, we will compare the performance of **rankdist** and **Rankcluster** on a well-studied real data set.

Package **pmr** includes four distance-based ranking models and their weighted version. It currently does not support mixture models. Package **mlogit** implements the mixed logit model and multinomial probit model, which belong to the order-statistics models (Alvo and Yu 2014).

5.2. Package architecture

The package **rankdist** is designed with efficiency and extendability in mind. Where possible, C++ code is used to perform computationally intensive steps. The R code is optimized to reduced memory footprints. The package is carefully modularized. Making modifications on one part of the software is unlikely to affect the other parts. Unit tests are also implemented in order to avoid code degradation.

Three S4 classes serve as the backbone of the package. The '`RankData`' class keeps all the information about the ranking data set. The '`RankInit`' class specifies initial parameter settings and indicates whether we want to fit a mixture model. The '`RankControl`' class is a void class and it has to be derived into a '`RankControl[model]`' class. This class specifies which model we want to fit and the control parameters used in the optimization procedure.

Three input objects are provided to the model generator function `RankDistanceModel`. This function will apply the EM solver if the '`RankInit`' object contains initial values for more than one component. As discussed in Section 4.1, each component model in the mixture is fitted to a ranking data set with "discounted" observations. In fact the EM solver will prepare those data sets and make multiple calls to `SearchPi0`, which fits a single component model. The function `SearchPi0` then searches the optimum modal rankings. The details of this algorithm is described in Section 3.5. Checked candidates are cached in a hash table to prevent double checking. For each candidate modal ranking, the function `SingleClusterModel` is called to estimate parameter weightings. It is implemented as a generic function and the internal method dispatch mechanism will apply suitable solver according to the signature of the '`RankControl[model]`' object.

The object-oriented approach brings two immediate benefits. First of all, the ranking data stored in the data structure are better protected against user mistakes such as confusing rankings with orderings. The data integrity benefits as a result. Secondly, with a higher level of abstraction implementing a new ranking model reduces to extending the '`RankControl`' class and implementing the two methods (`SingleClusterModel` and `FindProb`), and the other parts of the software do not need to be affected. Such infrastructure can be used in other packages which deal with ranking data. We hope the package would make experiments with new ranking models easier.

6. Using the rankdist package

6.1. Preprocessing

The ranking data have two equivalent representations: ranking and ordering. The ranking representation records the ranks of objects in the form $(\pi(1), \pi(2), \ldots)$ where $\pi(i)$ is the rank of object i while the ordering representation records the objects in the rank order $(\pi^{-1}(1), \pi^{-1}(2), \ldots)$ where $\pi^{-1}(i)$ is the object ranked the ith. Since both representations are common, it is very important for us to make a clear distinction and avoid any confusion. The **rankdist** package stores ranking data in a S4 class 'RankData'. The user only needs to initialize the object once using either representation and the software would handle the data storage problem internally. The details of the initialize method of 'RankData' class can be found in the full documentation that is available as part of the software.

In the following example, we illustrate how to initialize a 'RankData' object from different representations. We will use a helping function GenerateExample to generate two simple data sets. The representation of generated data depends on the argument ranking = TRUE. In the example, the data set gen1 is encoded in ranking representation, and the data set gen2 is encoded in ordering representation. Note that the ranking or ordering matrix do not have duplicated rows, and the number of observations for each ranking is specified in a vector named count. The 'RankData' objects dat1 and dat2 are then initialized from the two raw data sets by supplying the constructor with appropriate arguments.

```
R> library("rankdist")
R> gen1 <- GenerateExample(ranking = TRUE)
R> tail(gen1$ranking)

       [,1] [,2] [,3] [,4] [,5]
[115,]    5    4    1    2    3
[116,]    5    4    1    3    2
[117,]    5    4    2    1    3
[118,]    5    4    2    3    1
[119,]    5    4    3    1    2
[120,]    5    4    3    2    1

R> dat1 <- new("RankData", ranking = gen1$ranking, count = gen1$count)
R> gen2 <- GenerateExample(ranking = FALSE)
R> tail(gen2$ordering)

       [,1] [,2] [,3] [,4] [,5]
[115,]    3    4    5    2    1
[116,]    3    5    4    2    1
[117,]    4    3    5    2    1
[118,]    5    3    4    2    1
[119,]    4    5    3    2    1
[120,]    5    4    3    2    1

R> dat2 <- new("RankData", ordering = gen2$ordering, count = gen2$count)
```

The user can also store a data set that contains top-q rankings as illustrated in the next example. We use another helping function GenerateExampleTopQ to generate a simple data set containing top-three rankings of five objects. Two additional arguments need to be provided to the constructor. The argument nobj represents the total number of objects, which is typically the number of columns of the ranking matrix. The argument topq specifies the value of q, the position where exact rankings are available.

```
R> genq <- GenerateExampleTopQ()
R> head(genq$ranking)

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    4
[2,]    1    2    4    3    4
[3,]    1    2    4    4    3
[4,]    1    3    2    4    4
[5,]    1    3    4    2    4
[6,]    1    3    4    4    2

R> datq <- new("RankData", ranking = genq$ranking, count = genq$count,
+    nobj = ncol(genq$ranking), topq = max(genq$ranking) - 1)
```

6.2. Model generation

In this section, we illustrate how to fit models in **rankdist** after we have obtained '`RankData`' objects.

We first introduce the '`RankControl`' class, which specifies the type of model and controls the behavior of parameter estimation. In the following example, we create two '`RankControl`' objects, `ctrl1` and `ctrlq`. They specify the desired model class to be the Mallows' ϕ model and the weighted Kendall model respectively. We provide an additional argument, `SearchPi0_show_message = FALSE`, to suppress the message output when searching for the central ranking. All other available arguments are detailed in the software documentation.

```
R> ctrl1 <- new("RankControlKendall", SearchPi0_show_message = FALSE)
R> ctrlq <- new("RankControlWeightedKendall",
+    SearchPi0_show_message = FALSE)
```

We use the '`RankInit`' object to specify initial values of parameters as well as properties of the mixture model. The central ranking can be initialized with any rank aggregation procedure. The initialization of weight parameters is less important as we have shown in Section 3.5 that weights estimation is a convex optimization problem which has global optimal solution. Function `MomentsEst` creates feasible initial values for weights. The number of mixture components needs to be specified in the `clu` argument. If `clu > 1`, the user needs to provide initial π_0 and weights for each component.

In the following example, we first create object `init1` to fit a single cluster model for complete rankings. The central ranking is estimated using Borda count method (Marden 1995) and the weights are given by the helper function `MomentsEst`. Next, we create object `init1c` to fit a two-cluster mixture model for complete rankings. For simplicity, the central rankings are initialized randomly for each cluster. Finally, we create object `initq` to fit a single cluster model for top-q rankings. Again, rank initialization is done by Borda count method. Note that the algorithm will treat all rankings greater than q properly. The weights are initialized by an arbitrary value in the feasible region (0.5).

```
R> str1 <- MomentsEst(dat1, 500)
R> avg_rank <- dat1@count %*% dat1@ranking
```

```
R> modal_ranking.init <- OrderingToRanking(order(avg_rank))
R> modal_ranking.init.c1 <- sample(dat1@nobj, dat1@nobj)
R> modal_ranking.init.c2 <- sample(dat1@nobj, dat1@nobj)
R> init1 <- new("RankInit", param.init = list(str1),
+    modal_ranking.init = list(modal_ranking.init), clu = 1L)
R> init1c <- new("RankInit", param.init = list(str1, str1),
+    modal_ranking.init = list(modal_ranking.init.c1, modal_ranking.init.c2),
+    clu = 2L)
R> initq <- new("RankInit", param.init = list(rep(0.5, datq@topq)),
+    modal_ranking.init = list(modal_ranking.init), clu = 1L)
```

With three objects ‘RankData’, ‘RankControl’, and ‘RankInit’, model fitting is simply done by calling the generic function RankDistanceModel as illustrated as follows.

```
R> model1 <- RankDistanceModel(dat1, init1, ctrl1)
R> model1c <- RankDistanceModel(dat1, init1c, ctrl1)
R> modelq <- RankDistanceModel(datq, initq, ctrlq)
```

The function ModelSummary provides a summary of the fitted model. Two common approaches to assess the goodness of fit of a particular model is the BIC (Bayesian information criterion) value and the sum of squares of Pearson residuals (SSR).

$$\mathrm{BIC} = -2 \cdot \ell(\pi_1, \pi_2, \ldots, \pi_n) + \mathrm{dof} \cdot \log(n),$$
$$\mathrm{SSR} = \sum_{i=1}^{t!} \frac{(O_i - E_i)^2}{E_i},$$

where dof is the number of free parameters in the model and O_i and E_i are the observed and expected frequencies of the ith ranking, respectively. A lower value in BIC and SSR indicates a better fit.

```
R> ModelSummary(model1)
```

```
Summary of model1
================
Goodness of Fit
SSR:          109.426
BIC:          18832.76
dof:          1
Parameter Estimation
Cluster  A  B  C  D  E  p  Parameters
1        1  2  3  4  5  1  0.2
```

7. Experimental study

7.1. Simulation study

The aim of the simulation study is to verify that the estimation procedure produces reasonable parameter estimations when the sample rankings are generated by the weighted Kendall model. We are especially interested in cases when the number of objects t is large (more than 20). For simulations involving large t, the first problem is to generate samples from the predefined weighted Kendall distribution. The naive way is to calculate the probability of each ranking and take samples from the corresponding categorical distribution. The naive method does not scale up as it involves calculating and storing $t!$ probabilities. The typical alternatives are MCMC type algorithms such as the Metropolis-Hasting algorithm. These algorithms are computational and memory efficient but the generated samples will not be independent. In the next section, we introduce a way to efficiently obtain independent samples from weighted Kendall model. Then we proceed to the simulation results.

Generating samples from the weighted Kendall model

A sample ranking σ can be recursively generated from weighted Kendall model with π_0 and $\boldsymbol{w}$. In the first stage, the algorithm samples the item to be ranked first, i.e., $\sigma^{-1}(1)$, from t available items. The probability for choosing item k as the first item is given by

$$\mathsf{P}\left(\sigma^{-1}(1) = k\right) \propto \exp\left(-\sum_{i=1}^{\pi_0(k)-1} w_i\right).$$

The summation inside represents the weights accumulated when item k is moved to the top of π_0 via adjacent transpositions.

After $\sigma^{-1}(1)$ is sampled, the item will be removed from π_0. The weights should also be updated by removing w_1 and keeping the remaining ones. The same procedure is then used again with updated π_0 and $\boldsymbol{w}$. The algorithm chooses one item from the remaining $t-1$ items to be $\sigma^{-1}(2)$.

After t steps, σ will be a complete ranking and the recursion will end. The complexity of obtaining one sample is $O(t^2)$, a big improvement over the naive method.

Estimating π_0 with given w

This simulation study tests the heuristic algorithm presented in Section 3.5. A data set of n sample rankings of t objects is generated from weighted Kendall model with a certain π_0 and $\boldsymbol{w}$. We choose t to be 40 to capture the large-t scenario and test the scalability of the algorithm. The choice of π_0 does not affect the result of this simulation because the distance metric is invariant under relabeling of items. The weight $\boldsymbol{w}$ is decreasing and $w_i = \log(41-i)/5$. The weight gives the right amount of dispersion so that the sample rankings are highly likely to be unique.

The algorithm is provided with the sample data, the true weights $\boldsymbol{w}$, and an initial ranking π_{00} estimated using the Borda count method (Marden 1995). We record the Kendall distance between the central ranking found by the algorithm and the true π_0. This procedure is then

n	Distance	0	1	2	3
200	initial	263	190	42	5
	final	354	128	16	2
500	initial	422	76	2	0
	final	450	49	1	0

Table 1: Distribution of Kendall distance between the true π_0 and the estimated ones. The results are for two simulations with sample sizes 200 and 500. Type "initial" represents the distance between the initial ranking (Borda count method) and true π_0, and type "final" represents the distance between the ranking estimated by the algorithm and true π_0. The number in the cell represents the number of runs that the distance is observed. The numbers in each row sum up to 500 because the sampling-estimation procedure is repeated for 500 times for each simulation.

repeated for 500 times. On a laptop with a 2.6 GHz CPU each repetition takes less than one second to finish.

In Table 1, we tabulate the distribution of distance for two simulations with sample sizes $n = 200$ and 500 respectively. The distance between the initial ranking (Borda count method) and π_0 is also shown for reference. We observe that in general the rankings produced by the heuristic algorithm is closer to the true π_0 than the initial ranking.

Estimating w with given π_0

This simulation study tests the algorithm for estimating weights $\boldsymbol{w}$ presented in Section 3.5. The data generation procedure is the same as Section 7.1. The choice of π_0 and $\boldsymbol{w}$ also remains the same and the sample size is chosen to be 500.

The weight estimation algorithm is provided with the sample data and the true ranking π_0. We record the estimated weights in each run. This procedure is then repeated for 500 times. Again, each repetition takes less than one second to finish.

Figure 2 illustrates the true weights w_i (red dots) and the distribution of estimated weights in the 500 runs (box plot). In most cases the true weight is very close to the median value of the estimated weights. In all cases, the true weights fall between the quartiles of estimated weights.

Estimating both π_0 and w

This simulation study tests the algorithm that jointly estimates π_0 and w presented in Section 3.5. The data generation procedure is the same as Section 7.1. The choice of π_0 and $\boldsymbol{w}$ also remains the same and the sample size is chosen to be 500.

The algorithm is provided with the sample data and an initial ranking π_{00} estimated using the Borda count method (Marden 1995). We record the Kendall distance between the central ranking found by the algorithm and the true π_0 as well as the estimated weights. This procedure is then repeated for 500 times. Each repetition takes around 15 seconds to finish.

In Table 2, we tabulate the distribution of recorded distance. We observe that in general the rankings estimated by the algorithm is closer to the true π_0 than the initial ranking. Compared with the results in Section 7.1, the estimation of π_0 does not seem to be significantly worse when weights are also unknown.

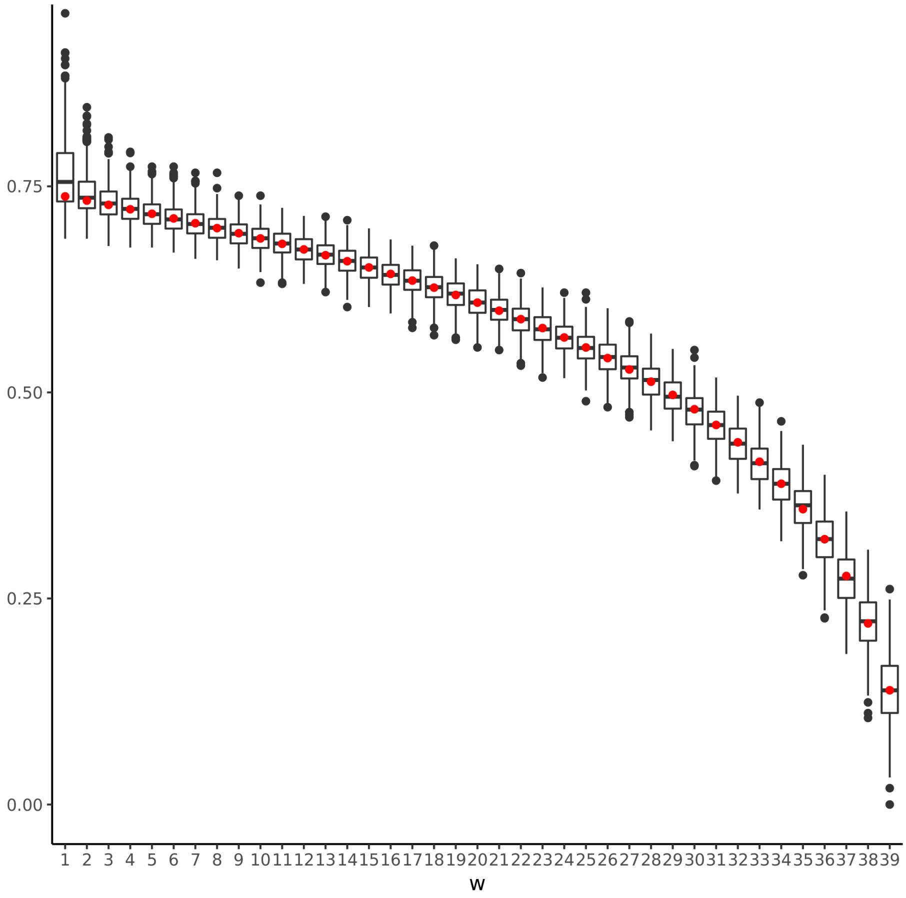

Figure 2: Visualization of estimated weights when the true π_0 is given. In this setting the weights are the only parameters to be estimated. The simulation involves 39 unknown weight parameters, which are plotted across the horizontal axis. The true weights are represented as a red dot. The distribution of the estimated weight in the 500 runs is illustrated via a box plot. The box plot encodes the median, lower and upper quartiles, and outliers.

n	Distance	0	1	2
500	initial	414	84	2
	final	467	33	0

Table 2: Distribution of Kendall distance between the true π_0 and the estimated ones.

Figure 3 illustrates the true weights w_i (red dots) and the distribution of estimated weights in the 500 runs (box plot). Similar to the results in Section 7.1 we found that in most cases the true weight is very close to the median value of the estimated weights. In all cases, the true weights fall between the quartiles of estimated weights.

This simulation suggests that π_0 and w can be estimated reliably by the algorithm.

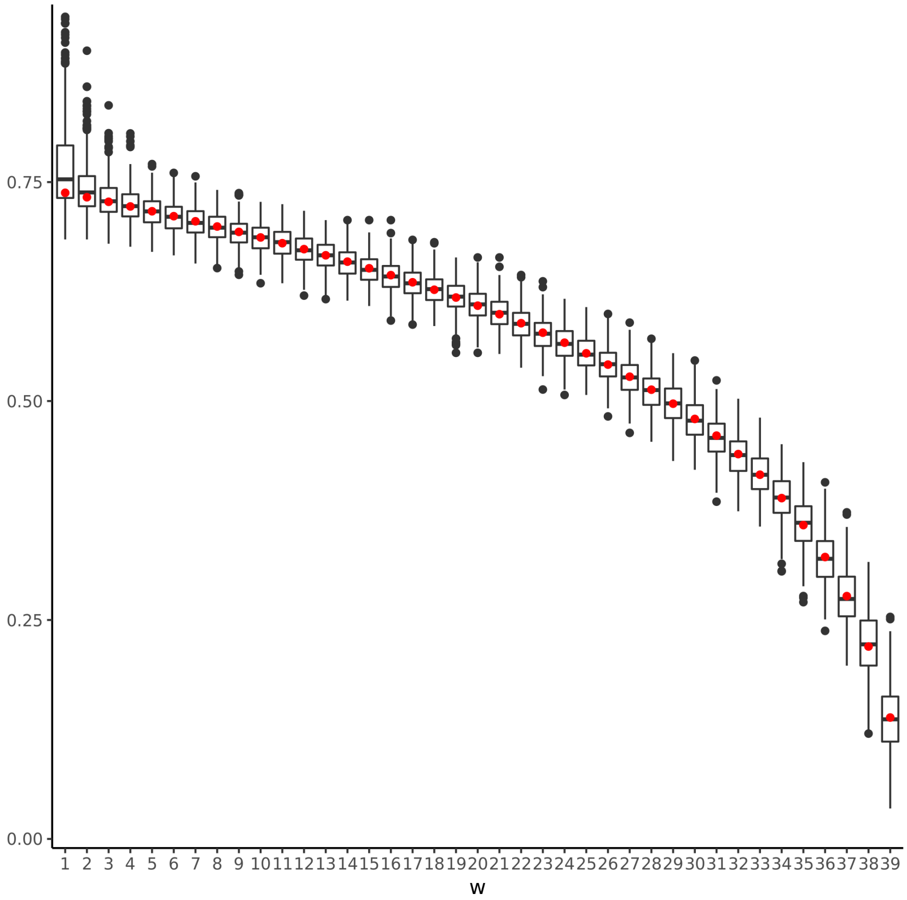

Figure 3: Visualization of estimated weights when the true π_0 is also unknown. Unlike the previous setting, the weights and π_0 have to be estimated jointly here. The simulation involves 39 unknown weight parameters, which are plotted across the horizontal axis. As before, the true weights are represented as a red dot, and the distribution of the estimated weight in the 500 runs is illustrated via a box plot. The box plot encodes the median, lower and upper quartiles, and outliers.

7.2. Modeling ranking probability

Data set and methodology

The purpose of this experiment is two-fold: to show that the weighted Kendall model is able to achieve excellent performance on real data set, and to show that one can easily fit and evaluate different ranking models in package **rankdist**.

In the experiment we use the APA Election data set, which consists of votes in the 1980 American Psychological Association (APA) presidential election (Diaconis 1988). This data set is well studied in the literature and it has been included in the **rankdist** package. It contains $15,449$ rankings of five candidates, of which $5,738$ are complete rankings. The rest are top-q rankings with q ranging from 1 to 3. Note that only one of the five candidates will be elected as the president. Therefore the voters are likely to put more emphasis on the top positions, a scenario which can be captured by weighted Kendall model.

Model	Components	Free parameters	BIC
Weighted Kendall	3	9	53685.4
Mallows' ϕ	5	9	53729.3
Weighted tau	3	17	53785.2
ISR[1]	4	11	54335.8

Table 3: Summary of the fitted models for complete rankings.

Model	Cluster	Modal ranking π_{0g}					w_{1g}	w_{2g}	w_{3g}	w_{4g}	w_{5g}	Proportion
	g	A	B	C	D	E						p_g
Weighted Kendall	1	2	3	1	5	4	1.03	1.03	0.52	0.34	-	37%
	2	3	4	5	1	2	0.48	0.48	0.36	0.36	-	33%
	3	4	2	5	3	1	0.23	0.22	0.22	0.22	-	30%
Mallows' ϕ	1	3	2	5	1	4	0.38	-	-	-	-	27%
	2	2	3	1	5	4	0.83	-	-	-	-	25%
	3	3	4	5	2	1	0.61	-	-	-	-	21%
	4	3	4	2	5	1	0.24	-	-	-	-	20%
	5	2	5	1	3	4	0.72	-	-	-	-	8%
Weighted tau	1	1	5	2	4	3	0.19	0.61	0.16	1.15	0.24	56%
	2	3	2	1	4	5	5.43	5.77	5.46	1.69	0.11	24%
	3	3	4	1	2	5	4.04	0.98	0.58	-0.93	0.28	20%

Table 4: Parameter estimates of the fitted models for complete ranking.

In the experiment, we consider the following distance-based ranking models available in the **rankdist** package:

- Mixture of Mallows' ϕ model (Section 2.2)
- Mixture of weighted tau model (Section 2.4)
- Mixture of weighted Kendall model proposed in this paper (Section 3)

We will also evaluate mixture of ISR model (Jacques and Biernacki 2014) using package **Rankcluster** (Grimonprez and Jacques 2014).

For single-parameter models we fit a mixture of up to eight components and for more complicated models we fit up to three components. The BIC (Bayesian information criterion) is used to select the best number of components in the mixture model by choosing the one with the smallest BIC. Multiple initial value configurations have been tried and the best results are shown below.

Modeling for the complete rankings

We present the model fitting results in Tables 3 and 4. Table 3 displays the best fitted models selected using the BIC criterion while Table 4 shows the parameter estimates of the three fitted models. The weighted Kendall model achieves the best fit in terms of BIC.

There are several well-known facts about the APA election data set. First, candidates A and C are research psychologists, candidates D and E are clinical psychologists and candidate

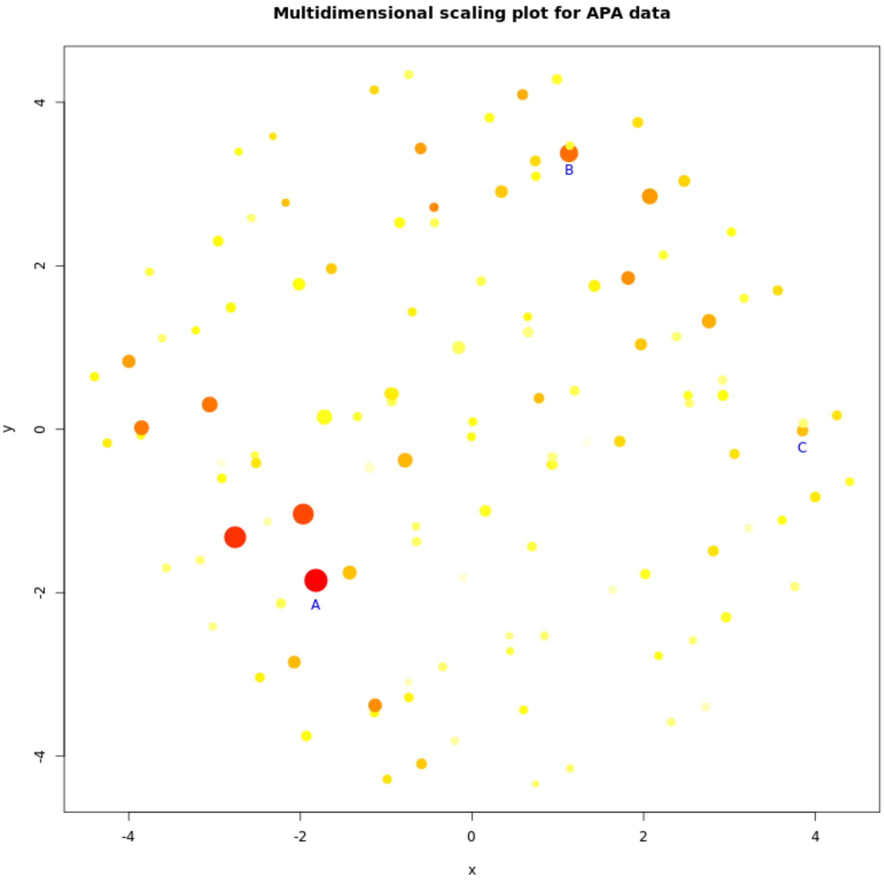

Figure 4: Visualization of APA data with fitted model.

B is a community psychologist. Candidates naturally form three competing subgroups, and voters show different preferences towards different subgroups. Estimation result shows that candidates A and C are ranked at the top two in the first cluster while candidates D and E are preferred in the second cluster, and the third cluster assigns the highest rank to candidate E. The competing effects between different candidates are successfully captured. The weighted tau model does not capture this phenomenon although a three-component mixture model is found. The Mallows' ϕ model captures the competing relationship with the first two clusters, but it also includes a few smaller but confounding components. The inflexibility of single-parameter models often requires a larger number of mixing components, which makes interpretation more difficult.

The weighted Kendall model also allows more fine-grained interpretation. The weights for swapping first three positions are higher in the first cluster than the second cluster. This suggests that for those who prefer candidates A and C, they have a stronger tendency to do so. On the contrary, the weights for the third cluster is lower, suggesting that voters in this cluster have more diverse preferences.

[1] Figure 1 of Jacques and Biernacki (2014) shows that mixture of ISR model with four components achieves BIC value around 53000 on the complete rankings of APA data set (the exact value was not given). However, we were not able to reproduce this result with package **Rankcluster** (version 0.94) after trying various MCMC configurations. The BIC value shown in the table is the best result obtained in our experiment.

Model	Components	Free parameters	BIC
Equal probability	3	10	141276.5
Tied rank	3	9	141479.5
ISR[2]	3	8	144774.2

Table 5: Summary for the fitted models for all rankings.

Assumption	Cluster	Modal ranking π_{0g}					w_{1g}	w_{2g}	w_{3g}	w_{4g}	Proportion
	g	A	B	C	D	E					p_g
Equal probability	1	3	4	5	1	2	0.41	0.41	0.24	0.18	42%
	2	2	3	1	5	4	1.08	1.08	0.36	0.22	29%
	3	3	1	5	4	2	0.18	0.18	0.15	0.15	28%
Tied rank	1	3	4	5	2	1	0.52	0.52	0.41	0.04	42%
	2	3	1	5	2	4	0.33	0.33	0.33	0.05	29%
	3	2	3	1	5	4	1.62	1.62	0.27	0.27	28%

Table 6: Parameter estimates of the fitted models for all rankings.

We apply a multidimensional scaling (MDS) to the 120 × 120 distance matrix with each cell being the Kendall distance between two observed rankings. Figure 4 shows a plot of the two-dimensional solution to MDS. Each point on the plot represents a ranking. The distance between points on the plot is approximately proportional to the Kendall distance between two corresponding rankings. The size of the point represents the observed frequency of the ranking, whereas the color represents the expected frequency from the fitted weighted Kendall model.

We observe that larger dots have darker color in general, which indicates the expected frequencies agree with observed ones. The modal rankings of the three clusters are labeled as 1, 2 and 3 on the plot. The first two clusters in the model are clearly reflected on the plot as two distinct regions of larger points whereas the structure of the third cluster looks a bit less clear. This feature is supported by the estimation result which suggests voters in the third cluster have more diversified opinions. It is thus reasonable that there are few dominating rankings in this cluster and the cluster looks more noisy.

Modeling for all rankings

In the APA election data set, there are 5,738 complete rankings and 9,711 incomplete rankings. Analysis based on the data with the incomplete rankings removed may have a significant effect on the conclusion drawn from the data. Here, we fit weighted Kendall model to all rankings. Both the equal probability and tied rank assumptions described in Section 4.2 are considered in the estimation of the model parameters. The results are also compared with mixture of ISR model obtained by **Rankcluster**. Tables 5 and 6 show the best fitted mixture models and the parameter estimates.

BIC suggests a three-component solution for all models considered in this experiment. Both forms of weighted Kendall models outperform the ISR model. The model based on the equal probability assumption has the best fit in terms of BIC.

[2] Figure 1 of Jacques and Biernacki (2014) reports worse BIC value (around 151800, exact value not reported) than we have obtained with package **Rankcluster** (version 0.94) using default arguments.

Data set name	Nobj	Nobs	BetterObs	DistToTruth	BordaToTruth
Nine of Ten Amendments	9	63	0%	0	1
Book Releases	10	78	18%	7	7
Classic Oscar Releases	10	78	9%	4	3
Country Landmasses	10	142	0%	1	1
Country Populations	10	78	18%	11	11
European City Populations	10	78	23%	12	11
Hardness of Materials	10	78	27%	13	11
Movie Releases	10	78	6%	2	2
Recent Oscar Releases	10	78	0%	1	4
River Lengths	10	78	14%	12	11
Super bowl	10	78	4%	10	10
Ten Amendments	10	78	5%	4	6
Ten Commandments	10	78	18%	11	12
US City Populations	10	142	1%	6	2
US Holidays	10	146	1%	1	2
US Presidents	10	142	0%	0	0
US States West to East	10	78	1%	1	3
World City Populations	10	144	4%	10	9
NBA East 2010 Season	15	148	20%	35	36
NBA West 2010 Season	15	135	9%	19	24

Table 7: Rank aggregation results of 20 different data sets.

Note that more than 60% of data are incomplete rankings. After fitting the weighted Kendall models (with equal probability assumption) to all rankings, the results are slightly different from those for complete rankings. Table 6 shows that the largest cluster (44%) becomes the one with candidates D and E ranked at the top two and the one with candidates A and C ranked at the top two is the next largest cluster (29%). This indicates that the voters who gave incomplete rankings tend to prefer candidates D and E more. Finally, the third cluster represents about 27% of voters who have diverse views on their favorable candidates.

7.3. Test of general knowledge

In this section, we empirically validate whether the weighted Kendall distance model gives reasonable estimation on the central ranking. We use 20 different data sets collected by Lee, Steyvers, and Miller (2014) where undergraduates recruited from the human subjects pool at the University of California Irvine were asked to perform 20 ranking tasks. The ranking tasks involve general knowledge of existing ground truths.

As the true rankings of the objects are known in all 20 tasks, we can assess the goodness of the estimated modal ranking by measuring its distance to truth. We also use the aggregated ranking given by Borda count method (Marden 1995) as a reference.

The data sets and the model performance are summarized in Table 7. All data sets involve around 10 objects and fewer than 150 observations. The column "BetterObs" of Table 7 records the percentage of observed rankings which are closer to the true ranking than the aggregated ranking in terms of Kendall distance. The column "DistToTruth" records the

Kendall distance between the true ranking and the aggregated ranking. The last column records the Kendall distance between the true ranking and the ranking obtained from the Borda count method.

The results reveal that in most cases, only a small fraction of observed rankings are closer to truth than the aggregated ranking. Furthermore, in 17 out of 20 cases the ranking obtained by weighted Kendall model is closer or has the same distance to the truth than the one obtained by Borda count method. This indicates that the model performs reasonably well in finding the modal ranking.

8. Conclusions

In this paper, we presented the R package **rankdist** for fitting different distance-based ranking models. We also introduced a new probability model based on weighted Kendall distance whose parameterization are both flexible and interpretable. We showed that the model has a nice analytic form. We also proposed and validated the parameter estimation procedure.

In the experimental study we have shown that the package offers a simple and coherent way to fit a wide variety of ranking models. Model selection and comparison are made easy as a consequence. Moreover the package **rankdist** can be easily modified to incorporate new ranking models. Thus it provides a platform for simulation experiments and development of new ranking models.

We found that the computation time for model-fitting procedure is dominated by the neighbor-checking step in the search of modal ranking. In this step the program checks all neighbors of the current best modal ranking. A possible way to improve the **rankdist** package is therefore to parallelize this step and fully utilize the power of multi-core processors.

Acknowledgments

The research of Philip L. H. Yu was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No.17303515).

References

Alvo M, Yu PLH (2014). *Statistical Methods for Ranking Data*. Springer-Verlag. doi:10.1007/978-1-4939-1471-5.

Biernacki C, Jacques J (2013). "A Generative Model for Rank Data Based on Insertion Sort Algorithm." *Computational Statistics & Data Analysis*, **58**, 162–176. doi:10.1016/j.csda.2012.08.008.

Critchlow D (1986). "A Unified Approach to Constructing Nonparametric Rank Tests." *Technical Report 86-15*, Department of Statistics, Purdue University.

Croissant Y (2019). ***mlogit**: Multinomial Logit Model*. R package version 1.0-1, URL https://CRAN.R-project.org/package=mlogit.

Diaconis P (1988). *Group Representations in Probability and Statistics.* Institute of Mathematical Statistics, Hayward.

Dwork C, Kumar R, Naor M, Sivakumar D (2001). "Rank Aggregation Methods for the Web." In *Proceedings of the 10th International Conference on World Wide Web*, pp. 613–622.

Farnoud F, Milenkovic O (2014). "An Axiomatic Approach to Constructing Distances for Rank Comparison and Aggregation." *IEEE Transactions on Information Theory*, **60**(10), 6417–6439. doi:10.1109/tit.2014.2345760.

Fligner MA, Verducci JS (1986). "Distance-Based Ranking Models." *Journal of the Royal Statistical Society B*, **48**(3), 359–369. doi:10.1111/j.2517-6161.1986.tb01420.x.

Fligner MA, Verducci JS (1988). "Multi-Stage Ranking Models." *Journal of the American Statistical Association*, **83**(403), 892–901. doi:10.2307/2289322.

Grimonprez Q, Jacques J (2014). ***Rankcluster**: Model-Based Clustering for Multivariate Partial Ranking Data.* R package version 0.92-9, URL https://CRAN.R-project.org/packages=Rankcluster.

Irurozki E, Calvo B, Lozano JA (2016). "**PerMallows**: An R Package for Mallows and Generalized Mallows Models." *Journal of Statistical Software*, **71**(12), 1–30. doi:10.18637/jss.v071.i12.

Jacques J, Biernacki C (2014). "Model-Based Clustering for Multivariate Partial Ranking Data." *Journal of Statistical Planning and Inference*, **149**, 201–217. doi:10.1016/j.jspi.2014.02.011.

Joe H (2001). "Multivariate Extreme Value Distributions and Coverage of Ranking Probabilities." *Journal of Mathematical Psychology*, **45**(1), 180–188. doi:10.1006/jmps.1999.1294.

Kumar R, Vassilvitskii S (2010). "Generalized Distances between Rankings." In *Proceedings of the 19th International Conference on World Wide Web*, pp. 571–580.

Lee MD, Steyvers M, Miller B (2014). "A Cognitive Model for Aggregating People's Rankings." *PloS ONE*, **9**(5), e96431. doi:10.1371/journal.pone.0096431.

Lee PH, Yu PLH (2012). "Mixtures of Weighted Distance-Based Models for Ranking Data with Applications in Political Studies." *Computational Statistics & Data Analysis*, **56**(8), 2486–2500. doi:10.1016/j.csda.2012.02.002.

Lee PH, Yu PLH (2015). ***pmr**: Probability Models for Ranking Data.* R package version 1.2-5, URL https://CRAN.R-project.org/package=pmr.

Luce RD (1959). *Individual Choice Behavior.* John Wiley & Sons, New York.

Mallows CL (1957). "Non-Null Ranking Models. I." *Biometrika*, **44**(1–2), 114–130. doi:10.2307/2333244.

Marden JI (1995). *Analyzing and Modeling Rank Data.* Chapman Hall, New York. doi:10.1201/b16552.

Murphy TB, Martin D (2003). "Mixtures of Distance-Based Models for Ranking Data." *Computational Statistics & Data Analysis*, **41**(3–4), 645–655. `doi:10.1016/s0167-9473(02)00165-2`.

Nash JC (2014). "On Best Practice Optimization Methods in R." *Journal of Statistical Software*, **60**(2), 1–14. `doi:10.18637/jss.v060.i02`.

Qian Z (2019). ***rankdist**: Distance Based Ranking Models*. R package version 1.1.4, URL `https://CRAN.R-project.org/package=rankdist`.

R Core Team (2019). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `https://www.R-project.org/`.

Smith BB (1950). "Discussion of Professor Ross's Paper." *Journal of the Royal Statistical Society B*, **12**(1), 53–56. `doi:10.1111/j.2517-6161.1950.tb00041.x`.

Thurstone LL (1927). "A Law of Comparative Judgement." *Psychological Reviews*, **34**(4), 273–286. `doi:10.1037/h0070288`.

Yu PLH (2000). "Bayesian Analysis of Order-Statistics Models for Ranking Data." *Psychometrika*, **65**(3), 281–299. `doi:10.1007/bf02296147`.

A. Derivation of the normalization constant

Denote S_t be the set of all possible permutations of t objects labeled by integers $\{1, \ldots, t\}$, and π_0 be the modal ranking of interest. For any ranking π in S_t, we can use Equation 2 to construct a vector $\vec{V} = \langle V_1, V_2, \ldots, V_{t-1} \rangle$, where

$$V_i = \sum_{j=i+1}^{t} I\{[\pi(\pi_0^{-1}(i)) - \pi(\pi_0^{-1}(j))] > 0\}.$$

Note that V_i can take value $0, 1, \ldots, t-i$. Let $\Omega = \{\langle V_1, V_2, \ldots, V_{t-1} \rangle | V_i \leq t-i, V_i \in \mathbb{N}_0\}$ be the set of all possible values of the V_i's, where $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$. Fligner and Verducci (1988) showed that the definition of $\vec{V}$ itself provides a bijective mapping $f : S_t \to \Omega$. It follows that if the sets $\Gamma_1, \Gamma_2, \ldots, \Gamma_m$ partition the set Ω, then the sets $f^{-1}(\Gamma_1), f^{-1}(\Gamma_2), \ldots, f^{-1}(\Gamma_m)$ partition the set S_t, and vice versa.

In the first step, we partition Ω into the following two sets:

$$\begin{aligned} \Gamma_{1,1} &= \{\langle V_1, V_2, \ldots, V_{t-2}, 1 \rangle | V_i \leq t-i, V_i \in \mathbb{N}_0\}, \\ \Gamma_{1,0} &= \{\langle V_1, V_2, \ldots, V_{t-2}, 0 \rangle | V_i \leq t-i, V_i \in \mathbb{N}_0\}. \end{aligned}$$

Let $g : \Gamma_{1,0} \to \Gamma_{1,1}$ be a mapping such that

$$g(\langle V_1, V_2, \ldots, V_{t-2}, 0 \rangle) = \langle V_1, V_2, \ldots, V_{t-2}, 1 \rangle.$$

Note that the mapping g is also a bijection. For all $\vec{V} \in \Gamma_{1,0}$,

$$D_{wK}(f^{-1}(\vec{V}), \pi_0) = D_{wK}(f^{-1}(g(\vec{V}), \pi_0) - w_{t-1}.$$

This equation holds because the path between $f^{-1}(g(\vec{V}))$ and π_0 is longer than the path between $f^{-1}(\vec{V})$ and π_0 by one adjacent transposition τ_{t-1}. It follows that

$$C_{wK}(\boldsymbol{w}) = \sum_{\vec{V} \in \Omega} \exp\left[-D_{wK}(f^{-1}(\vec{V}), \pi_0)\right] = \left\{ \sum_{\vec{V} \in \Gamma_{1,0}} \exp\left[-D_{w}(f^{-1}(\vec{V}), \pi_0)\right] \right\} (1 + e^{-w_{t-1}}).$$

The next step is to partition $\Gamma_{1,0}$ into three sets:

$$\begin{aligned} \Gamma_{2,2} &= \{\langle V_1, V_2, \ldots, V_{t-3}, 2, 0 \rangle | V_i \leq t-i, V_i \in \mathbb{N}_0\}, \\ \Gamma_{2,1} &= \{\langle V_1, V_2, \ldots, V_{t-3}, 1, 0 \rangle | V_i \leq t-i, V_i \in \mathbb{N}_0\}, \\ \Gamma_{2,0} &= \{\langle V_1, V_2, \ldots, V_{t-3}, 0, 0 \rangle | V_i \leq t-i, V_i \in \mathbb{N}_0\}. \end{aligned}$$

Using the same method, we can show that

$$\sum_{\vec{V} \in T_{1,0}} \exp\left[-D_{wK}(f^{-1}(\vec{V}), \pi_0)\right] =$$

$$\left\{ \sum_{\vec{V} \in T_{2,0}} \exp\left[-D_{wK}(f^{-1}(\vec{V}), \pi_0)\right] \right\} (1 + e^{-w_{t-2}} + e^{-w_{t-2} - w_{t-1}}).$$

In each step, we form partitions of the target set and reduce the number of summations. In the final step, $\Gamma_{t-1,0} = \{\langle V_1, 0, \ldots, 0, 0, 0\rangle | V_1 \leq t-1, V_1 \in \mathbb{N}_0\}$, and it should be partitioned into t subsets according to the value of V_1. Thus the normalization constant can be written as

$$\begin{aligned} C_{wK}(\boldsymbol{w}) &= (1 + e^{-w_{t-1}}) \cdot \\ &\quad (1 + e^{-w_{t-2}} + e^{-w_{t-2}-w_{t-1}}) \cdot \\ &\quad (1 + e^{-w_{t-3}} + e^{-w_{t-3}-w_{t-2}} + e^{-w_{t-3}-w_{t-2}-w_{t-1}}) \cdots \end{aligned}$$

After reorganization, we obtain Equation 6.

The time complexity of evaluating $C_{wK}(\boldsymbol{w})$ is $O(t)$. To see that, we need to use the recursive structure of the terms in the product. If we denote the ith term in the product as M_i, then the following equation holds

$$M_{i+1} = 1 + w^{t-i-1} \cdot M_i.$$

Hence, it only takes constant time to obtain a new term in the product, and the overall complexity is $O(t)$.

B. Derivation of normalization constant for top-q rankings

Here, we are going to show that under the tied-rank assumption, the normalization constant for top-q rankings is proportional to the normalization constant for complete rankings. The proof is given below.

By definition $\mathsf{P}(\pi(q)) = \sum_\pi \mathsf{P}(\pi)$, where π are compatible complete rankings of the top-q ranking $\pi(q)$. Furthermore, for all pairs of complete rankings π_i and π_j compatible with the same top-q ranking $\mathsf{P}(\pi_i) = \mathsf{P}(\pi_j)$ if and only if $w_k = 0$ if $k > q$. It means that under the assumption $\mathsf{P}(\pi(q)) = (t-q)! \cdot \mathsf{P}(\pi)$. That is,

$$\mathsf{P}(\pi(q)) = \frac{e^{-D_{wK}(\pi(q),\pi_0)}}{C_q(\boldsymbol{w})} = (t-q)! \cdot \frac{e^{-D_{wK}(\pi,\pi_0)}}{C_{wK}(\boldsymbol{w})}.$$

Since $w_k = 0$ if $k > q$, we have $D_{wK}(\pi(q), \pi_0) = D_{wK}(\pi, \pi_0)$. We finally obtain

$$C_q(\boldsymbol{w}) = \frac{C_{wK}(\boldsymbol{w})}{(t-q)!},$$

and the proof is completed.

Affiliation:

Zhaozhi Qian, Philip L.H. Yu
Department of Statistics and Actuarial Science
The University of Hong Kong
Hong Kong, China
E-mail: qianzhaozhi@connect.hku.hk, plhyu@hku.hk

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/

July 2019, Volume 90, Issue 5 *Submitted:* 2015-10-14
doi:10.18637/jss.v090.i05 *Accepted:* 2018-05-03