*Journal of Statistical Software*

*MMMMMM YYYY, Volume VV, Issue II.* *doi: 10.18637/jss.v000.i00*

# 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. In this paper, we also provide some examples on the usage of the package. The examples also demonstrate the advantage of the new model in different aspects.

*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 where each decision can be to compare a pair of objects considered in the decision or to count the number of "mistakes" generated in the decision. A common critique of these two approaches is that the assumed 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** which provides a unified way to fit different distance-based ranking models. This paper also presents a 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 give details on the package architechture and implementation in Section 5. In Section 6, we illustrate the usage of package **rankdist** with simulated examples. In Section 7, we present the results of applying different models to APA Election data set using package **rankdist** and compare their performance in terms of goodness of fit and interpretability. We also apply our model to rank aggregation tasks, and compare the results to aggregated ranking based on the Borda count. 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 $\pi, \sigma, \tau$ 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 $\pi$ and $\sigma$, the product $\tau = \pi\sigma$ is defined by the equation $\tau(i) = \pi(\sigma(i)), i = 1, 2, \cdots, t$.

# 2. Distance-based ranking models

## 2.1. Overview of distance-based models

Sometimes it is reasonable to assume that there exists a modal ranking $\pi_0$ which has the highest probability to occur and most observed rankings are close to $\pi_0$. Thus we should assign the highest probability to $\pi_0$ and for any other ranking the probability should be negatively correlated with its distance from $\pi_0$. According to this framework, Diaconis (1988)

proposed a family of distance-based model,

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

where $\pi_0$ denotes the modal ranking, $D(\pi,\pi_0)$ is a distance function between $\pi$ and $\pi_0$, $\lambda \geq 0$ is the dispersion parameter and $C(\lambda)$ is the normalization constant. The usual 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 $\pi,\sigma$ and $\nu$. 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 is said to be a metric. The commonly used distances include

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.

2. Spearman distance:

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

3. Hamming distance:

$$D_H(\pi,\sigma) = t - \sum_{i=1}^{t}\sum_{j=1}^{t} I\{\pi(i) = j\}\, I\{\sigma(i) = j\}$$

4. Footrule distance:

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

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

Note that the Spearman footrule and Kendall distances are metrics. These distance measures are developed in different settings. For example the Hamming distance is closely related to coding theory and the Cayley distance has an intimate relationship with the Cayley graph in group theory. Each of these distances can be formulated into a ranking model as in (1). However, these models do not generally have a tractable normalization constant $C(\lambda)$, which limits their practical use.

## 2.2. Mallows' $\phi$ model

Among the above-mentioned distance functions, the Kendall disance is the most popular one as its distance-based model, also called the Mallows' $\phi$ model (Mallows 1957), can have a closed-form normalization constant:

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

In addition, Kendall distance satisfies all five properties given in the last section. It can also be interpreted as the minimum number of adjacent transpositions required to transform $\pi$ to $\sigma$.

Although the Mallows' $\phi$-model has a closed-form normalization constant, it may be too restrictive since it has only one model parameter. Efforts have been taken to introduce more flexibility into the Mallows' $\phi$-model, and we will describe two such examples below.

### 2.3. The $\phi$-component model

The $\phi$-component model proposed by Fligner and Verducci (1986) is a generalization of Mallows' $\phi$-model. Its basic idea is motivated by the fact that the Kendall distance can be decomposed into a sum of $t-1$ scores $V_i(\pi, \sigma), i = 1, , \cdots, t-1$ 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 needed to place the object $\sigma^{-1}(1)$ (the object ranked first in $\sigma$) in the first position in ranking $\pi$. In the $i$th stage $(2 \leq i \leq t-1)$, $\pi^{-1}(j) = \sigma^{-1}(j)$ for $j = 1, \cdots, i-1$, and $V_i(\pi, \sigma)$ is the number of adjacent transpositions needed to place the object $\sigma^{-1}(i)$ in the $i$th position in ranking $\pi$. 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 $(i-1)$ stage and it takes value $0, 1, \cdots, (t-i)$.

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

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

where $\boldsymbol{\lambda} = (\lambda_1, \cdots, \lambda_{t-1})'$ and $C(\boldsymbol{\lambda})$ is the normalization constant, 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 (3) can be simplified as

$$\mathsf{P}(\pi|\boldsymbol{\lambda}, \pi_0) = \prod_{i=1}^{t-1} \frac{e^{-\lambda_i V_i(\pi, \pi_0)}}{C(\lambda_i)} \tag{4}$$

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

The term $\sum_{i=1}^{t-1} \lambda_i V_i$ in the $\phi$-component model extends the notion of Kendall distance between two rankings and it degenerates to Kendall distance if all $\lambda_i$'s are the same. However, Lee and Yu (2012) showed that this generalization violates the symmetric property of distance. For this reason the $\phi$-component model does not belong to the class of distance-based ranking models.

### 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. In this way, the properties (1)-(4) of a distance can be preserved. For instance, the weighted version of Kendall distance between two rankings $\pi$ and $\sigma$ with the modal ranking $\pi_0$ and weights $\boldsymbol{w} = (w_1, \cdots, w_t)$ under this model is defined as

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

Lee and Yu (2012) called this distance as *weighted Kendall tau* distance or simply *weighted tau* distance. The probability of observing a ranking $\pi$ under the weighted tau distance-based ranking model is

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

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 $\pi$ and $\pi_0$. The shortcoming of this model is that the normalization constant $C_{wt}(\boldsymbol{w})$ becomes hard to compute for a large number of objects to be ranked.

## 3. Weighted Kendall distance model

### 3.1. Weighted Kendall distance

The new model presented in this paper is motivated by another generalization of Kendall distance which was axiomized by Farnoud and Milenkovic (2014), and has been shown to be a distance metric, named *weighted Kendall distance*. 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 $\tau_a$.

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

$$A(\pi, \sigma) = \{< \tau_{(1)}, \tau_{(2)}, ... > | \sigma = \pi\tau_{(1)}\tau_{(2)}...\}$$

to be the set of all ordered sequences of adjacent transpositions that transform $\pi$ to $\sigma$. In the above definition $\tau_{(i)}$ is the adjacent transposition taking place in the $i$th step. The weighted Kendall distance between two rankings $\pi$ and $\sigma$ is defined as

$$D_{wK}(\pi, \sigma) = \min_{<\tau_{(1)}, \tau_{(2)}, ...> \in A(\pi, \sigma)} \sum_i w_{(i)}, \tag{5}$$

where $w_{(i)}$ is the weight associated with the $i$th adjacent transposition $\tau_{(i)}$ applied to $\sigma$ in the sequence. If we define $Q_j(\pi, \sigma)$ to be the total number of adjacent transposition $\tau_j$ applied to $\sigma$ occurring in the minimizer, this new distance can be expressed in the following form:

$$D_{wK}(\pi, \sigma) = \sum_{j=1}^{t-1} w_j Q_j(\pi, \sigma), \tag{6}$$

where the weights $w_j$'s can be interpreted as the "difficulty" of transposing adjacent objects at certain rank positions in the ranking. For example, if we have three orderings, $\sigma^{-1} = A|B|C|D$, $\pi_1^{-1} = A|B|D|C$ and, $\pi_2^{-1} = B|A|C|D$ then in terms of the original Kendall distance: $D(\pi_1, \sigma) = D(\pi_2, \sigma) = 1$. However, in some situations, raters may care more about the top positions than the bottom positions. Therefore, transposing the top objects would be more difficult than transposing the bottom objects. This feature can be captured by assigning decreasing weights in the distance, i.e., $w_i \geq w_j$ for all $i < j$.

Similar to the original Kendall distance, $D_{wK}(\pi, \sigma)$ is also a graphic distance. The distance between $\pi$ and $\sigma$ is the length of the shortest path between vertices $\pi$ and $\sigma$ on the Cayley graph of the permutation group $S_t$. The vertices of such a graph are all possible rankings $\pi$'s (or orderings $\pi^{-1}$'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 $\tau_i$ this edge has a weight $w_i$. By definition, the shortest path minimizes the right hand side of (5). It follows that when all the $w_i$'s equal 1, the weighted Kendall distance degenerates to the original Kendall distance.

Applying shortest path algorithms on the weighted Cayley graph to find the weighted 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 has complexity $O(t^2)$. Surprisingly, the sequence of adjacent transpositions used to iteratively find $V_i$ in the $\phi$-component model in (2) is the minimizer of (5) (Farnoud and Milenkovic 2014).

Figure 1 provides an illustration of paths used in weighted Kendall distance. The label on each vertex is an ordering $\pi^{-1}$. The color of an edge represents the weight associated with that edge. A blue edge illustrates 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 $< \tau_{(1)}, \tau_{(2)}, ... >$ is not unique in this case and is marked with arrows. The $Q_j$ in (6) 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 have 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.

### 3.2. The normalization constant

The weighted Kendall distance introduced in the last section 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 $\pi_0$. We refer to this new model as the weighted Kendall distance-based

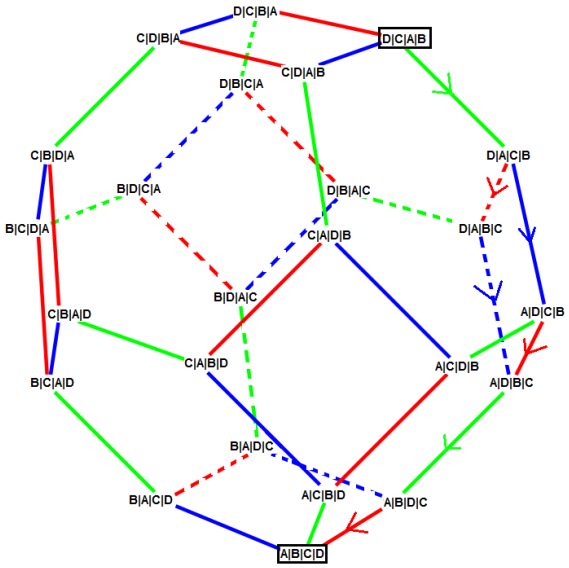


Figure 1: Graphical representation of weighted Kendall distance between four objects.

model. The probability of observing a ranking $\pi$ is

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

where $C_{wK}(\boldsymbol{w})$ is the normalization constant. Although the distance term in the model can be computed efficiently, 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.

$$C_{wK}(\boldsymbol{w}) = \prod_{i=1}^{t-1}\left[1 + \sum_{j=1}^{i} e^{-\sum_{k=j}^{i} w_{t-k}}\right]. \tag{7}$$

The derivation can be found in Appendix A. By the composition rule of convex functions one can easily 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.

### 3.3. Parameter estimation

*Estimating central ranking $\pi_0$*

Given a set of weights $w$ and $n$ observed rankings $\sigma_1, \ldots, \sigma_n$, finding the maximum likelihood estimate of central ranking $\pi_0$ is equivalent to minimize the total distance between $\pi_0$ and the observed rankings (8).

$$\pi_0 = \arg\min \left\{ \sum_{i=1}^{n} D_{wK}(\pi_0, \sigma_i) \right\} \tag{8}$$

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, in most cases the optimization algorithm will only be able to find a local optimal. A ranking $\pi$ is a local optimal of problem 8 if there is no ranking $\pi'$ that can be obtained from $\pi$ by performing a single adjacent transposition and $\sum_{i=1}^{n} D_{wK}(\pi, \sigma_i) > \sum_{i=1}^{n} D_{wK}(\pi', \sigma_i)$.

Dwork *et al.* (2001) showed that in the equal-weighting case all the local optimums of problem 8 have a desirable property, the extended Condorcet property. The extended Condorcet property requires that if there exists a partition of items $(C, C')$ such that for any $x \in C$ and $y \in C'$ the majority of $\sigma_1, \ldots, \sigma_n$ prefers $x$ over $y$, then $x$ must be ranked above $y$. Roughly speaking, the extended Condorcet property ensures that the "consensus" in the observed rankings are preserved in the aggregated ranking. Not all rank aggregation algorithm has extended Condorcet property and the Borda count algorithm is such an exception (Young 1974).

When the weights are different, the local optimums of problem 8 do not have extended Condorcet property either, but this should be perceived as a feature of the weighted Kendall distance. Essentially, the weighted Kendall distance captures the scenario when the disagreement at the top of the ranked list of objects matters more then the one at the bottom. Hence, both the number of disagreements (consenses) and the location of the disagreements will affect the weighted Kendall aggregation while the extended Condorcet property does not take the location into account.

We propose the following huristic algorithm to find local optimums. The algorithm is provided with an initial value of $\pi_0$, denoted as $\pi_{00}$. This initial value could be a frequently observed ranking or the result of another rank aggregation algorithm such as the Borda count algorithm. Then all its neighboring rankings are considered and the corresponding log-likelihoods are calculated. If $\pi_{00}$ achieves the largest likelihood among all its neighbouts then we have found the local optimum and the algorithm stops. Otherwise, we select the one with the largest likelihood as $\pi_{01}$ and consider all of its neighbors again. This process repeats until $\pi_{0k}$ has larger likelihood than all its neighbors. Since in each step the objective function 8 always decreases and there are only a limited number of $\pi_0$'s to choose from, the algorithm will stop eventually. In section 7.1.2 we evaluate this huristic 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. Hence, the estimation of weights is an important part of the model.

For the weighted Kendall model the log-likelihood function is given by:

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

where $Q_j(\pi_i, \pi_0)$ are defined as in (6), and they are constants if $\pi_0$ is given. In section 3.2 we've shown that the logrithm 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 reparametrize $w_j$ 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 $\phi$. The log-likelihood function is still concave and it can be rewritten in terms of $\phi$ as:

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

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** is used for the optimization in **rankdist**. The simulation studies in section 7.1.3 confirms that the optimization procedure is reliable.

*Estimating central ranking $\pi_0$ and weights $w$ jointly*

In most cases, neither $\pi_0$ nor $w$ is known to us and both need to be estimated from the data. It is hard to estimate $\pi_0$ and $w$ simuntanously because $\pi_0$ is a ranking while $w$ is a vector of real numbers. Instead, we apply a stage-wise method that iteratively searches for the two paramters.

The algorithm is provided with an initial value of $\pi_0$, denoted as $\pi_{00}$, which can be the result of any rank aggregation algorithm. The algorithm finds the optimal weights for $\pi_{00}$, denoted as $w_0$ and records the resulting log-likelihood value. Then all the neighboring rankings of $\pi_{00}$ are considered. For each of them, the algorithm finds the optimal weights $w$ and calculate the resulting log-likelihood value. If $\pi_{00}$ and $w_0$ achieve the highest likelihood among all neighbours then the algorithm stops and returns $\pi_{00}$ and $w_0$. Otherwise, the algorithm selects the neighbour with the largest likelihood as $\pi_{01}$ and consider all of its neighbors again. This process repeats until $\pi_{0k}$ and $w_k$ has larger likelihood than all neighbors of $\pi_{0k}$. The algorithm will stop eventually because there are only a limited number of candidate central rankings.

We evaluate the joint optimization algorithm in section 3.3.3 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 reletively 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 unimodal. Fortunately these models can be easily extended to a finite mixture of several 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:

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

where $p_g$ is proportion of raters in the $g$th 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 $z = (z_1, z_2, \ldots, z_G)$ 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 the following steps:

**Algorithm 1** EM-algorithm for fitting mixture models

1. E-Step: compute the value of $z_g$ for each observation $i$:

$$\hat{z}_g^{(i)} = \frac{p_g \mathsf{P}(\pi^{(i)} | \boldsymbol{w}_g, \pi_{0,g})}{\sum_{g'=1}^{G} p_{g'} \mathsf{P}(\pi^{(i)} | \boldsymbol{w}_{g'}, \pi_{0,g'})}$$

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.

### 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 adding $\frac{1}{(t-q)!}$ observation 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. The shortcoming of this approach is that if the number of compatible rankings $(t-q)!$ is large the computational cost of transforming the data is huge.

The proposed weighted Kendall distance 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 as $w_k = 0$ for all $k > q$, $D_{wK}(\pi_i, \sigma) = D_{wK}(\pi_j, \sigma)$ for any two complete

**Algorithm 2** 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
```

rankings $\pi_i$ and $\pi_j$ compatible with $\pi(q)$. Therefore, we do not need to list out all complete rankings compatible with $\pi(q)$ and simply define $D_{wK}(\pi(q), \sigma) = D_{wK}(\pi, \sigma)$ with $w_k = 0$ for all $k > q$ and any $\pi$ compatible with $\pi(q)$.

The algorithm is presented in the box "Algorithm 2". 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:

$$\begin{aligned} \ell(\vec{\pi}(q_1), \vec{\pi}(q_2), \ldots, \vec{\pi}(q_m)) &= \log \prod_{i=1}^{m} \mathsf{P}(\vec{\pi}(q_i)) \\ &= \sum_{i=1}^{m} \left[ -\sum_{k=1}^{n_{q_i}} D_{wK}(\pi_k(q_i), \pi_0) - n_{q_i} \log(C_{q_i}) \right] \end{aligned} \tag{9}$$

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 $\vec{\pi}(q_i)$.

## 5. Package architecture and implementation

The **rankdist** package is available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=rankdist. The latest development version can be

found at https://github.com/ZhaozhiQIAN/rankdist. The software is written in R and C++.

## 5.1. Existing software for modelling ranking data

At present, there are several alternative packages for modeling ranking data in R including **PerMallows** (Irurozki 2015), **Rankcluster** (Grimonprez and Jacques 2014), **pmr** (Lee and Yu 2015) and **mlogit** (Croissant 2013).

Package **PerMallows** implements six common distance-based models. The software is primarily written in C++ but it includes an interface to R. 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 seven. 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 behaviour of MCMC sampling. Package **Rankcluster** is also able to model partial rankings. The rankings of unrated items are treated as missing variables, which 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.

The flowchart in Figure 2 illustrates the model fitting 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 3.3.1. 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.

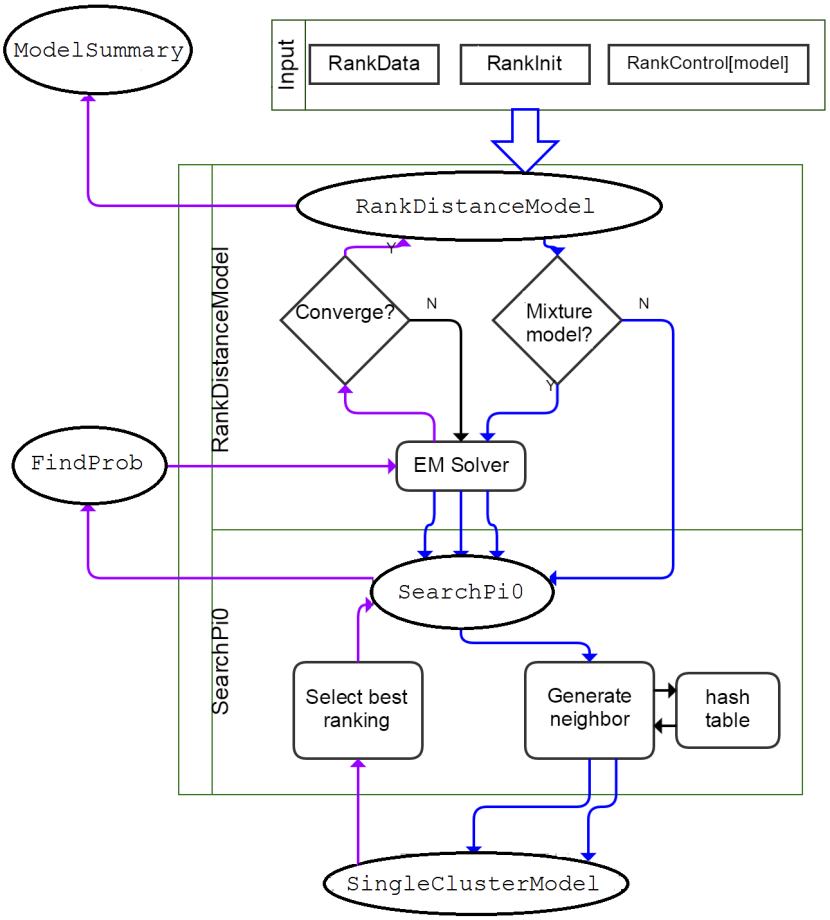


Figure 2: Flow chart of the model fitting procedure.

The purple arrows illustrate how the functions return values. `SingleClusterModel` returns fitted parameters and log-likelihood. `SearchPi0` selects the modal ranking according to the log-likelihood. Another generic function `FindProb` is called to give the fitted probabilities of

observed rankings. Similar to `SingleClusterModel`, this function is also model dependent. The EM solver updates each component data set according to these fitted probabilities. The `RankDistanceModel` function returns the final model if EM algorithm converges or a single component model is required. The function `ModelSummary` can display the parameters in the fitted model in a clear way.

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.

### 5.3. Estimation of weights

For models that utilize unweighted distance metric as defined in (1) there is only one free parameter, namely the dispersion parameter $\lambda$. The log-likelihood function of such model will become:

$$\ell(\pi_1, \pi_2, ..., \pi_n) = -\lambda \sum_{i=1}^{n} D(\pi_i, \pi_0) - n \log(C).$$

Given $\pi_0$, one dimensional optimization techniques can then be used to determine the maximum likelihood estimate of $\lambda$. In the package we use `optimize` function in R to implement this (Team 2014).

For the $\phi$-component model defined in (3) or equivalently (4), the log-likelihood function will be:

$$\ell(\pi_1, \pi_2, ..., \pi_n) = -\sum_{j=1}^{t-1} \left[ \sum_{i=1}^{n} \lambda_j \cdot V_j(\pi_i, \pi_0) - n \log(C_j(\lambda_j)) \right].$$

Therefore the maximization of log-likelihood function over $t-1$ parameters is reduced to maximizing $t-1$ independent one-dimensional functions. The package also uses `optimize` function in R to implement this.

For the weighted tau model, the optimization cannot be simplified into a series of one-dimensional optimization, and the gradient function of the log-likelihood function is also intractable. Hence, the Nelder-Mead method available in the R package **Optimx** is used (Nash and Varadhan 2011).

## 6. Using the rankdist package

### 6.1. Preprocessing

The ranking data can 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 $i$th. 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 need to initialize the object once using either representation and the software would handle the internal data storage problem implicitly. 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 the `GenerateExample` function 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 `count`. The `RankData` objects `dat1` and `dat2` are then initilized from the two raw data sets by supplying the constructor with apporiate 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 the `GenerateExampleTopQ` function to generate a simple data set containing top-three rankings of five objects. Two additional arguments need to be provided to the constructor. `nobj` represents the total number of objects, which is typically the number of columns of the ranking matrix. `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 model class and controls the behavior of the parameter estimation. In the following example, we create two `RankControl` objects, `ctrl1` and `ctrlq`. They specify the desired model class to be the Mallows' $\phi$ model and the weighted Kendall model respectively. We provide an additional argument, `SearchPi0_show_message=FALSE`, to supress 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 initilized with any rank aggregation procedure. The initilization of weight parameters is less important as we have shown in section 3.3.2 that weights estimation is a convex optimization problem which has global optimal solution. Function `MomentsEst` is a function that 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 $\pi_0$ 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 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 initilization is done by Borda count method. Note that the algorithm will treat all rankings greater than $q$ properly. The weights are initlized by an arbitary 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 value and the sum of squares of Pearson residuals (SSR).

$$BIC = -2 \cdot \ell(\pi_1, \pi_2, ..., \pi_n) + dof \cdot \log(n)$$

$$SSR = \sum_{i=1}^{t!} \frac{(O_i - E_i)^2}{E_i},$$

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 $i$th 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 of large $t$, the first problem is to generate samples from the pre-defined 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 Metropolice-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 weighted Kendall model*

A sample ranking $\sigma$ can be recursively generated from weighted Kendall model with $\pi_0$ 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 $\pi_0$ via adjacent transpositions.

After $\sigma^{-1}(1)$ is sampled, the item will be removed from $\pi_0$. The weights should also be updated by removing $w_1$ and keeping the remaining ones. The same procedure is the used again with updated $\pi_0$ and $\boldsymbol{w}$. The algorithm chooses one item from the remaining $t-1$ items to be $\sigma^{-1}(2)$.

After $t$ steps, $\sigma$ 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 $\pi_0$ with given $w$*

This simulation study tests the heuristic algorithm presented in section 3.3.1. A data set of $n$ sample rankings of $t$ objects is generated from weighted Kendall model with a certain $\pi_0$ and $\boldsymbol{w}$. We choose $t$ to be 40 to capture the large-$t$ senario and test the scalability of the algorithm. The choice of $\pi_0$ 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 $\pi_{00}$ estimated using the Borda count method. We record the Kendall distance between the central ranking found by the algorithm and the true $\pi_0$. This procedure is then repeated for

500 times. On a laptop with a Intel Core i7-6700HQ 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 $\pi_0$ is also shown for reference. We observe that in general the rankings produced by the heuristic algorithm is closer to the true $\pi_0$ than the initial ranking.

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

Table 1: Distribution of Kendall distance between the true $\pi_0$ and the estimated ones. The results for two simulations with sample sizes 200 and 500. Type 'initial' represents the distance between the initial ranking (Borda count method) and true $\pi_0$, and type 'final' represents the distance between the ranking estimated by the algorithm and true $\pi_0$. 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.

*Estimating $w$ with given $\pi_0$*

This simulation study tests the algorithm for estimating weights $\boldsymbol{w}$ presented in section 3.3.2. The data generation procedure is the same as section 7.1.2. The choice of $\pi_0$ 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 $\pi_0$. 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 3 illustrates the true weights $w_i$ (red dots) and the distribution of estimated weights in the 500 runs (boxplot). 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 $\pi_0$ and $w$*

This simulation study tests the algorithm that jointly estimates $\pi_0$ and $w$ presented in section 3.3.3. The data generation procedure is the same as section 7.1.2. The choice of $\pi_0$ 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 $\pi_{00}$ estimated using the Borda count method. We record the Kendall distance between the central ranking found by the algorithm and the true $\pi_0$ 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. The distance between the initial ranking (Borda count method) and $\pi_0$ is also shown for reference. We observe that in general the rankings estimated by the algorithm is closer to the true $\pi_0$ than the initial ranking.

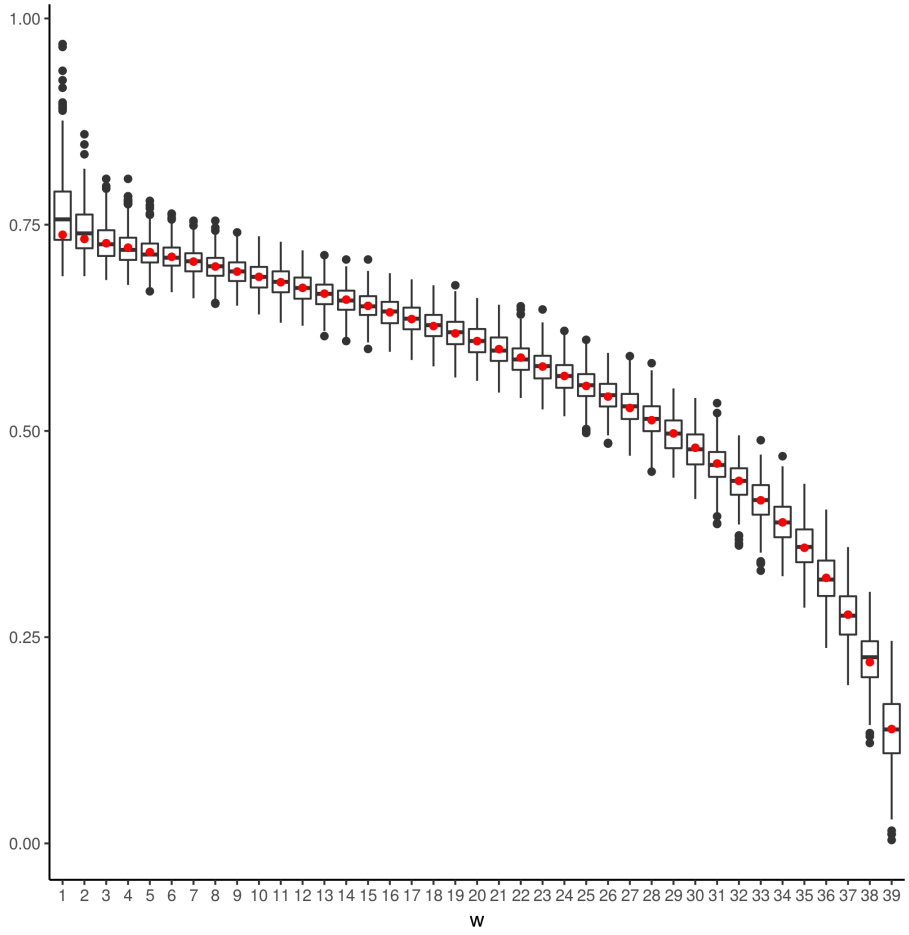


Figure 3: Visualization of true weights and estimated weights. The simulation involves 39 weight parameters, which are plotted across the horizontal axis. The true weight is represented as a red dot. The distribution of the estimated weight in the 500 runs is illustrated via a boxplot. The boxplot encodes the median, lower and upper quartiles, and outliers.

Compared with the results in 7.1.2, the estimation of $\pi_0$ does not seem to be significantly worse when weights are also unknown.

Figure 4 illustrates the true weights $w_i$ (red dots) and the distribution of estimated weights in the 500 runs (boxplot). Similar to the results in section 7.1.3 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 $\pi_0$ and $w$ can be estimated reliably by the algorithm.

| | Distance | 0 | 1 | 2 |
|---|---|---|---|---|
| n=500 | initial | 419 | 88 | 1 |
| | final | 453 | 47 | 0 |

Table 2: Distribution of Kendall distance between the true $\pi_0$ and the estimated ones.

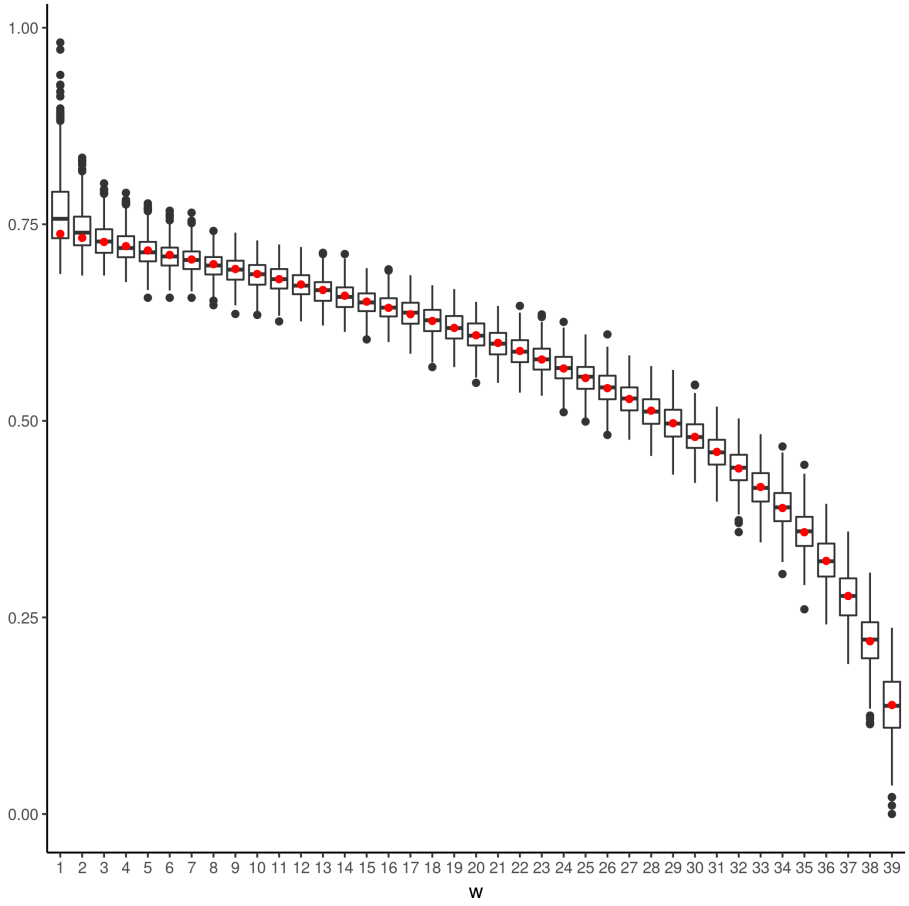


Figure 4: Visualization of true weights and estimated weights.

## 7.2. Modeling ranking probability

*Data set and methodology*

The purpose of this experitment 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 senario which can be captured by weighted Kendall model.

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

- Mixture of Mallows' $\phi$ 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 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 in order to avoid convergence to local maximal value.

*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.

| Model | Components | Free Parameters | BIC |
|---|---|---|---|
| Weighted Kendall | 3 | 9 | 53685.4 |
| Mallows' $\phi$ | 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

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 B is a community psychologist. Candidates naturally form three competing subgroups, and voters show different preferences towards different subgroups. We can see from Table 4 that the model based on weighed Kendall distance forms three clusters. Based on the estimates of the modal rankings of the three clusters, candidates A and C 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 obvious. The weighted tau model does not capture this phenomenon although a three-component mixture model is found. The Mallows' $\phi$ 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 new 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. 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 | Cluster $g$ | Modal ranking $\pi_{0g}$ | | | | | $w_{1g}$ | $w_{2g}$ | $w_{3g}$ | $w_{4g}$ | $w_{5g}$ | Proportion $p_g$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | D | E | | | | | | |
| Weighted Kendall | 1 | 2 | 3 | 1 | 5 | 4 | 1.02 | 1.02 | 0.51 | 0.33 | - | 37% |
| | 2 | 3 | 4 | 5 | 1 | 2 | 0.47 | 0.47 | 0.35 | 0.35 | - | 33% |
| | 3 | 4 | 2 | 5 | 3 | 1 | 0.23 | 0.21 | 0.21 | 0.21 | - | 30% |
| Mallows' $\phi$ | 1 | 3 | 2 | 5 | 1 | 4 | 0.38 | - | - | - | - | 27% |
| | 2 | 2 | 3 | 1 | 5 | 4 | 0.83 | - | - | - | - | 25% |
| | 3 | 3 | 4 | 5 | 1 | 2 | 0.61 | - | - | - | - | 21% |
| | 4 | 3 | 4 | 2 | 5 | 1 | 0.24 | - | - | - | - | 20% |
| | 5 | 2 | 5 | 1 | 3 | 4 | 0.72 | - | - | - | - | 8% |
| Weighted tau | 1 | 2 | 3 | 5 | 1 | 4 | 1.72 | 0 | -0.99 | -0.28 | 0.39 | 40% |
| | 2 | 5 | 2 | 4 | 1 | 3 | 3.65 | 0.91 | -0.08 | 0.1 | -0.04 | 31% |
| | 3 | 1 | 2 | 4 | 3 | 5 | 1.78 | 0.58 | -0.36 | -1.14 | -0.58 | 29% |

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

We apply a mulitdimensional scaling (MDS) to the $120 \times 120$ distance matrix with each cell being the Kendall distance between two observed rankings. Figure 5 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.

It can be observed that larger points often have darker color, 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. As mentioned before, 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.

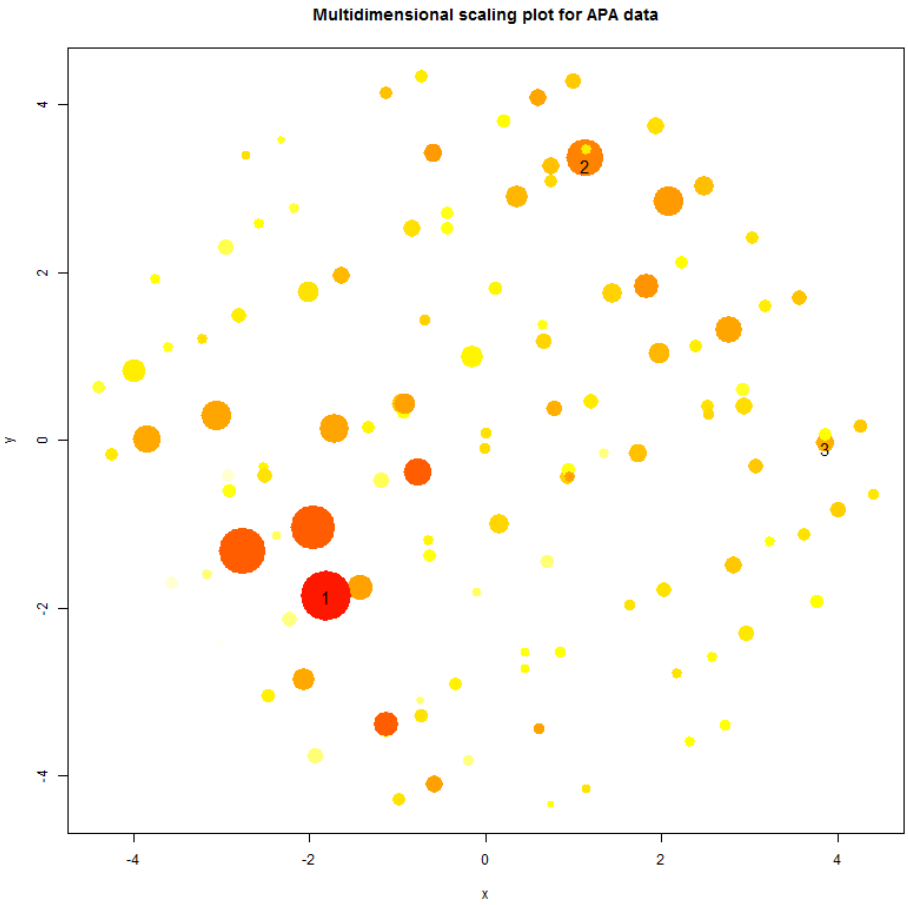


Figure 5: Visualization of APA data with fitted model.

| 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.

[2]Figure 1 of Jacques and Biernacki (2014) reports wrose BIC value (around 151800) than we have obtained with package **Rankcluster** (version 0.94) using default arguments.

| Assumption | Cluster $g$ | Modal ranking $\pi_{0g}$ | | | | | $w_{1g}$ | $w_{2g}$ | $w_{3g}$ | $w_{4g}$ | Proportion $p_g$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | A | B | C | D | E | | | | | |
| Equal probability | 1 | 3 | 4 | 5 | 1 | 2 | 0.40 | 0.40 | 0.24 | 0.17 | 44% |
| | 2 | 2 | 3 | 1 | 5 | 4 | 1.08 | 1.08 | 0.37 | 0.22 | 29% |
| | 3 | 3 | 1 | 5 | 4 | 2 | 0.19 | 0.19 | 0.15 | 0.15 | 27% |
| Tied rank | 1 | 3 | 4 | 5 | 2 | 1 | 0.52 | 0.52 | 0.41 | 0.03 | 43% |
| | 2 | 2 | 1 | 5 | 3 | 4 | 0.27 | 0.27 | 0.27 | 0.03 | 32% |
| | 3 | 2 | 3 | 1 | 5 | 4 | 1.75 | 1.75 | 0.26 | 0.26 | 25% |

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

BIC suggests a 3-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.

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. Rank aggregation

The problem of rank aggregation is to combine the rankings of objects given by a panel of judges. Typical applications can be found in many fields including social sciences, bioinformatics, meta search studies and sport competitions. Klementiev, Roth, and Small (2008) suggested using Mallows' $\phi$ model to aggregate complete/incomplete rankings. The problem can then be often formulated as the following optimization task

$$\pi^* = \arg\min \left\{ \sum_i D_K(\pi_i, \pi^*) \right\}.$$

However, finding the global optimal aggregated ranking can be very computationally intensive. In fact, it has been shown that the above problem is NP-hard (Bartholdi, Tovey, and Trick 1989). Instead of using Kendall distance, we can fit the weighted Kendall model to the data and use the estimated modal ranking to be the aggregated ranking.

In this section, we test whether the weighted Kendall distance model fitted by the greedy search scales up well and provides reasonable results. We compare the result with the ground truth and the aggregated ranking given by Borda Count method (Marden 1995), which is based on the arithmetic mean of the observed rankings.

We applied the model to 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 either general knowledge of existing ground truths or predictions of future outcomes. As the true rankings

| Data Set Name | Nobj | Nobs | BetterObs | DistToTruth | BordaToTruth |
|---|---|---|---|---|---|
| Nine of Ten Amendments | 9 | 63 | 0% | 1 | 1 |
| Book Releases | 10 | 78 | 12% | 6 | 7 |
| Classic Oscar Releases | 10 | 78 | 10% | 4 | 3 |
| Country Landmasses | 10 | 142 | 0% | 1 | 1 |
| Country Populations | 10 | 78 | 18% | 11 | 11 |
| European City Populations | 10 | 78 | 24% | 12 | 11 |
| Hardness of Materials | 10 | 78 | 28% | 13 | 11 |
| Movie Releases | 10 | 78 | 1% | 1 | 2 |
| Recent Oscar Releases | 10 | 78 | 0% | 1 | 4 |
| River Lengths | 10 | 78 | 15% | 12 | 11 |
| Super bowl | 10 | 78 | 3% | 8 | 10 |
| Ten Amendments | 10 | 78 | 5% | 4 | 6 |
| Ten Commandments | 10 | 78 | 12% | 9 | 12 |
| US City Populations | 10 | 142 | 1% | 6 | 2 |
| US Holidays | 10 | 146 | 0% | 0 | 2 |
| US Presidents | 10 | 142 | 0% | 0 | 0 |
| US States West to East | 10 | 78 | 5% | 2 | 3 |
| World City Populations | 10 | 144 | 1% | 7 | 9 |
| NBA East 2010 Season | 15 | 148 | 20% | 35 | 36 |
| NBA West 2010 Season | 15 | 135 | 11% | 20 | 24 |

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

of the objects are finally known in all 20 tasks, we can assess the goodness of the aggregated ranking given by the estimated modal ranking of the fitted weighted Kendall model.

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 Borda Count method.

The rank aggregation results reveal that in most cases, only a small fraction of observed rankings are better than the aggregated ranking. Furthermore, in 15 out of 20 cases the aggregated ranking obtained from the new model is closer or has the same distance to the truth than the one obtained by Borda Count method. This indicates that the weighted Kendall distance model generally performs better than Borda Count methods in aggregating rankings.

## 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 parametrization allows an elegant graphical interpretation. We have shown that the

model has a nice analytical form and can be easily extended into some more complicated forms.

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 neighbour-checking step in the search of modal ranking. In this step the program checks all neighbours 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.

Bartholdi J I, Tovey C, Trick M (1989). "Voting Schemes for Which It Can Be Difficult to Tell Who Won the Election." *Social Choice and Welfare*, **6**(2), 157–165. ISSN 0176-1714. doi:10.1007/BF00303169. URL http://dx.doi.org/10.1007/BF00303169.

Biernacki C, Jacques J (2013). "A generative model for rank data based on insertion sort algorithm." *Computational Statistics & Data Analysis*, **58**, 162–176.

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

Croissant Y (2013). ***mlogit**: Multinomial Logit Model*. R package version 0.2-4, URL https://cran.r-project.org/web/packages/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. ACM.

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.

Fligner MA, Verducci JS (1986). "Distance-based Ranking Models." *Journal of the Royal Statistical Society Series B*, **48**(3), 359–369.

Fligner MA, Verducci JS (1988). "Multi-stage Ranking Models." *Journal of the American Statistical Association*, **83**, 892–901.

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/web/packages/Rankcluster.

Irurozki E (2015). ***PerMallows**: Permutations and Mallows Distributions*. R package version 1.8, URL https://cran.r-project.org/web/packages/PerMallows.

Jacques J, Biernacki C (2014). "Model-based clustering for multivariate partial ranking data." *Journal of Statistical Planning and Inference*, **149**, 201–217.

Joe H (2001). "Multivariate Extreme Value Distributions and Coverage of Ranking Probabilities." *Journal of Mathematical Psychology*, **45**, 180–188.

Klementiev A, Roth D, Small K (2008). "Unsupervised Rank Aggregation with Distance-based Models." In *Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland*.

Lee MD, Steyvers M, Miller B (2014). "A Cognitive Model for Aggregating People's Rankings." *PloS one*, **9**(5), e96431.

Lee PH, Yu PLH (2012). "Mixtures of Weighted Distance-based Models for Ranking Data with Applications in Political Studies." *Computational Statistics and Data Analysis*, **56**, 2486–2500.

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

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

Mallows CL (1957). "Non-null Ranking Models." *Biometrika*, **44**, 114–130.

Marden JI (1995). *Analyzing and Modeling Rank Data*. Chapman Hall, New York.

Murphy TB, Martin D (2003). "Mixtures of distance-based models for ranking data." *Computational Statistics and Data Analysis*, **41**, 645–655.

Nash JC, Varadhan R (2011). "Unifying Optimization Algorithms to Aid Software System Users: Optimx for R." *Journal of Statistical Software*, **43**(9), 1–14.

Smith BB (1950). "Discussion of Professor Ross's Paper." *Journal of the Royal Statistical Society Series B*, **12**, 53–56.

Team RC (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Thurstone LL (1927). "A Law of Comparative Judgement." *Psychological Reviews*, **34**, 273–286.

Young HP (1974). "An axiomatization of Borda's rule." *Journal of economic theory*, **9**(1), 43–52.

Yu PLH (2000). "Bayesian Analysis of Order-statistics Models for Ranking Data." *Psychometrika*, **65**(3), 281–299.

# Appendix

## A. Derivation of the Normalization Constant

Denote $S_t$ be the set of all possible permutations of $t$ objects labelled by integers $\{1, \cdots, t\}$, and $\pi_0$ be the modal ranking of interest. For any ranking $\pi$ in $S_t$, we can use (2) to construct a vector $\vec{V} = < V_1, V_2, \cdots, V_{t-1} >$, 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 $0, 1, \cdots, t-i$. Let $\Omega = \{< V_1, V_2, \cdots, V_{t-1} > | V_i \leq t-i, V_i \in \mathbb{N}_0\}$ be the set of all possible values of the $V_i$'s, where $\mathbb{N}_0 = \{0, 1, 2, \cdots\}$. 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 $\Gamma_1, \Gamma_2, \cdots, \Gamma_m$ partition the set $\Omega$, then the sets $f^{-1}(\Gamma_1), f^{-1}(\Gamma_2), \cdots, f^{-1}(\Gamma_m)$ partition the set $S_t$, and vice versa.

In the first step, we partition $\Omega$ into the following two sets:

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

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

$$g(< V_1, V_2, \cdots, V_{t-2}, 0 >) =< V_1, V_2, \cdots, V_{t-2}, 1 > .$$

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 $\pi_0$ is longer than the path between $f^{-1}(\vec{V})$ and $\pi_0$ by one adjacent transposition $\tau_{t-1}$. 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:

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

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, $\Gamma_{t-1,0} = \{< V_1, 0, \cdots, 0, 0, 0 > | V_1 \leq t-1, V_1 \in \mathbb{N}_0\}$, 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 (7).

## B. Derivation of the 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 $P(\pi(q)) = \sum_\pi P(\pi)$, where $\pi$ is compatible complete rankings of the top-$q$ ranking $\pi(q)$. Furthermore, for all pairs of complete rankings $\pi_i$ and $\pi_j$ compatible with the same top-$q$ ranking $P(\pi_i) = P(\pi_j)$ if and only if $w_k = 0$ if $k > q$. It means that under the assumption $P(\pi(q)) = (t-q)!\cdot P(\pi)$. That is,

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

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

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd