Technical Report



UNIVERSITY OF THE BASQUE COUNTRY Department of Computer Science and Artificial Intelligence

An R package for permutations, Mallows and Generalized Mallows models

Ekhine Irurozki, Borja Calvo, José A. Lozano

January, 2014

San Sebastian, Spain www.ehu.es/ccia-kzaa hdl.handle.net/10810/4562

An R package for permutations, Mallows and Generalized Mallows models

Ekhine Irurozki, Borja Calvo, Jose A. Lozano January 17, 2014

Index terms— Permutations Mallows Model Sampling Learning R C

Abstract

Probability models on permutations associate a probability value to each of the permutations on n items. This paper considers two popular probability models, the Mallows model and the Generalized Mallows model. We describe methods for making inference, sampling and learning such distributions, some of which are novel in the literature. This paper also describes operations for permutations, with special attention in those related with the Kendall and Cayley distances and the random generation of permutations. These operations are of key importance for the efficient computation of the operations on distributions. These algorithms are implemented in the associated R package. Moreover, the internal code is written in C++.

1 Introduction

Permutations are ordered sets of items that arise naturally in many domains, such as genomics [Bader(2011)], cryptography, scheduling, computer vision [Ziegler et al.(2012)Ziegler, Christ etc. The hottest topic is preference learning [Pre(2013)], since its commercial applications have increased exponentially in the last years. However, most of the theoretical foundation basis in use have been discussed in the literature for years.

In order to deal with uncertainty probability distributions are often used. Probability distributions over permutations assign a probability value to each possible permutation of n items. Since the space of permutations grows factorially with n (recall that the number of permutations of n items is n!) there is a need of using probability models for medium-large values of n. However, the particularities of permutation data does not allow us to adapt many concepts of distributions over binary or real data to permutations. Consider, for example, the notion of independence. For a distribution over binary variables, independence between two variables implies that the first variable will have value 0 or 1 without regard to the value of the second variable. However, any two given positions of the permutation can not have the same value by any chance, so the classic notion of independence does not naturally translate to the permutation domain. Therefore, permutation spaces need specific probability models and concepts. However, the requirements for these models are very much the same,

that is, correct fitness to real data and the efficient computation of the most common operations, such as making inference, learning and sampling. We think that efficient algorithms for distributions over permutations can not be given without considering the particular nature of permutations. Therefore, efficient algorithms for dealing with permutations and its common operations are the key to giving efficient algorithms for probability models over permutations.

We have consider two of the most popular probability models on permutations, the Mallows models (MM) and the Generalized Mallows model (GMM). The MM is a distance-based ranking model in which the probability value of any given permutation depends on its closeness to the mode permutation. It has a simple definition relaying on just two parameters, the mode (central) permutation, σ_0 , which is the mode of the distribution, and the dispersion parameter, θ which controls the sharpness of the distribution. The closeness between permutations is given by one of the several metrics, being the most popular Kendall, Cayley, Ulam, Hamming, Spearman's- ρ and Spearman's footrule. When the distance for permutations is Kendall the MM also belongs to the family of multistage models, is also known as the Mallows ϕ -model and can also be motivated as a paired comparison model, [Critchlow $et\ al.(1991)$ Critchlow, Fligner, and Verducci].

The GMM is the most referenced among the extensions of the MM. The GMM under the Kendall distance is a multistage algorithm also known as Mallows ϕ -component model. As the MM, the GMM is also an unimodal distribution centered around σ_0 . However, instead of one single spread parameter as in the MM, the GMM makes use of n-1 dispersion parameters. The idea is to model situations in which the distribution is close to uniformity for some positions of the permutation while being close to the mode for some others.

The two metrics for permutations considered in this paper -and package- are the Kendall and the Cayley. The Kendall distance is related to the number of inversions of a permutation. Cayley distance, on the other hand, is related to the number of swaps and also to the cyclic structure of a permutation.

This package aims to be a compilation of functions on permutations, MM and GMM. We include the most common operations for probability models such as inference, sampling and learning. In order to have efficient algorithms, the nature of the space of permutations on which the distribution acts must be taken into consideration. Therefore, the basis of the package consists on several functions for dealing with permutations with special attention on those related with the Kendall and Cayley metrics and the random generation of permutations. We distribute all these functions as an R package. In this way, any researcher, even those not familiar with permutations or probability models, can have an intuitive interface to operations some of which are very complex. Also, most methods are coded in C++ internally, which runs much faster than R code. All the code can be publicly accessed so any researcher can adapt it for its specific purposes. Therefore, we think that this package can be attractive for any kind of user, from the beginners in the fields of permutations and/or probability models that want to get the intuition on how these functions behave to those who want to extend the models or apply them to deal with real and large sets of permutation data.

This is not the first package in the literature to deal with distributions over permutations. The prefmode package models preference data in the form of paired comparisons. In the particular case of the distance-based models there exist two packages. The RMallows package uses a EM algorithm to fit the MM

under the Kendall distance to full or partial rankings, with and without ties. The pmr on the other hand, implements the MM under the Kendall distance, among other probability models. This package is aimed to help in the analysis of preference data in tasks such as visualizing data and computation of descriptive statistics. It also includes an application of the problem to label ranking (classification).

This paper is organized as follows. Section 2 introduces the notation and basic concepts on permutations. Several operations for dealing with this particular data type are introduced and explained since they will be the key to the development of efficient operations for distributions over permutations. The probability models, MM and GMM, are detailed in Section 3 while the next two sections, 4 and 5 introduce several algorithms for sampling and learning respectively. The usage of the package is shown as the problems are stated and the algorithms are described.

2 Dealing with permutations

Permutations or rankings are ordered sets of the first n natural numbers. We will denote permutations with Greek letters, mostly π and σ . The permutation that places every item i in position i is called identity permutation and it is denoted as $e = [123 \dots n]$. For every permutation σ its inverse σ^{-1} is defined as $\sigma(i) = j \Leftrightarrow \sigma^{-1}(j) = i$. Two permutations can be composed resulting in a new permutation. The composition operation is defined as $\sigma\pi(i) = \sigma(\pi(i))$. It is worth noticing that $\sigma\sigma^{-1} = e$.

There are many distance metrics for permutations. Two are considered in this package, Kendall and Cayley. For the next version of the package we plan to include Ulam and Hamming metrics as well. All these metrics are right invariant, what means that $d(\sigma,\pi)=d(\sigma\gamma,\pi\gamma)$ for every permutation γ . Particularly taking $\gamma=\pi^{-1}$ and since $\pi\pi^{-1}=e$ one can w.l.o.g. write $d(\sigma,\pi)=d(\sigma\pi^{-1},e)$. For the sake of clarity the distance is denoted as a one parameter function when the reference permutation is the identity, $d(\sigma\pi^{-1},e)=d(\sigma\pi^{-1})$ what simplifies the notation. The right invariant property implies that we can always and w.l.o.g. take the identity permutation as the reference one. An example of application of the invariance property appears when counting the permutations, since the number of permutations at distance d from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations at distance e from e equals the number of permutations are distance.

The Kendall distance $d_k(\sigma, \pi)$ counts the number of pairwise disagreements between σ and π . It is used mainly in voting theory. When comparing a permutation σ to the identity, $d_k(\sigma)$ counts the number of inversions of σ . It is sometimes called bubble sort distance because $d_k(\sigma)$ equals the number of adjacent swaps that the bubble sort algorithm performs to order the items in σ increasingly.

For permutation σ , the bubble sort algorithm starts by checking item $\sigma(n-1)$ and will swap it with item $\sigma(n)$ iff $\sigma(n-1) > \sigma(n)$. Then, it will check item $\sigma(n-2)$ and order it respect the tail of the permutation, so $\sigma(n-2) < \sigma(n-1) < \sigma(n)$. For this step, item $\sigma(n-2)$ can be swapped 0, 1 or 2 times. In general, we can define a vector $V(\sigma) = (V_1(\sigma), \ldots, V_{n-1}(\sigma))$ such that $V_j(\sigma)$ equals the number of times that the bubble sort algorithm swaps item $\sigma(j)$, where

 $0 \le V_j(\sigma) \le n - j$ and $1 \le j < n$. Note that $V_j(\sigma)$ is the number of items smaller than $\sigma(j)$ in the tail of the permutation, and that it can be expressed as follows:

$$V_j(\sigma) = \sum_{i=j+1}^n I(\sigma(i) < \sigma(j))$$
 (1)

where $I(\cdot)$ denotes the identity function. Clearly, $d_k(\sigma) = \sum_{j=1}^{n-1} V_j(\sigma)$. Moreover, there is a bijection between each $\sigma \in S_n$ and each possible $V(\sigma)$ vector. Therefore, when dealing with the Kendall distance we can use the $V(\sigma)$ vector as an alternative representation of σ . The conversion from $V(\sigma)$ to σ and vice versa is supported in the current package and done in time $O(n^2)$.

The Cayley distance $d_c(\sigma, \pi)$ counts the number of swaps (not necessary adjacent) that have to be made to transform σ into π . When the reference permutation is the identity, $d_c(\sigma)$ equals n minus the number of cycles of σ . This distance is not natural for the voting domain, but has applications in disciplines that range from cryptography to genetics. Therefore, we will not use the name of ranking when dealing with this metric.

The same as the Kendall distance $d_k(\sigma)$ can be decomposed into the $V(\sigma)$ vector, Cayley distance $d_c(\sigma)$ can be decomposed into the vector $X(\sigma)$ of n-1 binary terms, $X(\sigma) = (X_1(\sigma), \ldots, X_{n-1}(\sigma))$ for $1 \leq j < n$. Formally, it is defined as follows:

$$X_j(\sigma) = \begin{cases} 0 & \text{if } j \text{ is the largest item in its cycle in } \sigma \\ 1 & \text{otherwise} \end{cases}$$

Clearly, $d_c(\sigma) = \sum_{j=1}^{n-1} X_j(\sigma)$. However, there is not a bijection between every possible $X(\sigma)$ vector and $\sigma \in S_n$: although each $\sigma \in S_n$ has one unique $X(\sigma)$, the opposite is not necessarily true. The first algorithm for the random generation of σ given $X(\sigma)$ and assuming that every permutation consistent with $X(\sigma)$ is equally probable, has been introduced in [?]. The conversion from $X(\sigma)$ to one of the possible many σ and from σ to $X(\sigma)$ is supported in the current package and done in time $O(n^2)$.

In the present R package permutations are represented as vectors. Let us enumerate related function which are included in the package. The function is.permu(permu) indicates if the vector permu is a valid permutation. The function identity.permutation(permu.length) returns the identity permutation of permu.length items and permutations.of(permu.length) generates every possible permutation of permu.length items. Permutation inversion and compositions are supported by functions inverse.permu(permu) and compose(permu) respectively. Basic operations for permutations are swapping two items, adjacent or not, and inserting an item. These are done by the operations inversion.at(permu, i), swap(permu, i, j) and insert(permu, i, j). It is possible to load a sample of permutations included in a file with the function read.permu.file(path). The permutation matrix of a permutation or a sample of them is obtained with permu.matrix(permu).

The present package implements functions to compute the Kendall and Cayley distance between permutations, as distance(permu1,permu2=identity.permu(length(permu1)), method='kendall'). The cycle decomposition to which the Cayley distance is closely related is also supported. It is possible to obtain the cycle notation from a

given permutation and vice versa with the functions permutation2cycles(permu) and cycles2permutation(cycles). It also implements a function to obtain the distance decompositions, permutation2decomposition(permu, method='kendall'), that is, it can obtain the $V(\sigma)$ and $X(\sigma)$ for any σ . Moreover, this package implements the function for recovering a permutation σ given $V(\sigma)$ or $X(\sigma)$, decomposition2permutation(vec, method='kendall'). As we have already stated, there exists one unique σ for a given $V(\sigma)$. However, given $X(\sigma)$, this package will generate any of the possibly many permutations consistent with such decomposition with equal probability using the procedure introduced in [?].

2.1 Counting and generating permutations

The random generation of permutations is a problem of interest in many disciplines It can be efficiently carried out with the well known Fisher-Yates shuffle (also known as Knuth shuffle). A more restrictive version of the problem is that of generating a permutation at distance d. This problem is closely related to that of counting the number of permutations at distance d from the identity. In this section we deal with these questions. In particular, we are interested in the following problems:

- Given a metric, the number of items n and a distance d, how many permutations are there at distance d from the id?
- Given a metric, the number of items n and a distance d, generate u.a.r. a permutation at the given distance from the id.

We will approach both questions, counting and u.a.r. generation of permutations at distance d, with recursive procedures.

The question of counting the number of permutations at distance d has been largely studied for several metrics. In particular, there is an entry in the Online encyclopedia of Integer Sequences (OEIS) for each of the metrics treated in this package since there is no closed expression, neither for Kendall nor for Cayley distances. The sequence for Kendall distance is given by Triangle of Mahonian numbers, with code A008302 in the OEIS. One can find there eleven alternative interpretations for the sequence, references, links and much more. Also, there is a recurrence for its computation

$$S_k(n,d) = \begin{cases} 1 & n = 1 \land d = 1 \\ 0 & n = 1 \land d \neq 1 \\ S_k(n,d-1) + S_k(n-1,d) - S_k(n-1,d-n) & \text{otherwise} \end{cases}$$
(2)

The computational cost is $O(n^3)$.

The next question is the random generation of a permutation at distance d from the identity. Note that this is equivalent to generating a permutation with a given number of inversions, which, until now, was an open problem, [Arndt(2010)]. We will now show our proposed algorithm to solve this question.

Recall that there is a bijection between every permutation σ of n items and every possible $V(\sigma)$. Every position $1 \leq j < n$ of vector $V(\sigma)$, $V_j(\sigma)$, is restricted to have values $0 \leq (n-j)$. Moreover, $d_k(\sigma) = \sum_{j=1}^{n-1} V_j(\sigma)$. The problem of randomly generating a permutation at distance d from the identity is thus equivalent to that of randomly generating a vector $V(\sigma)$ such that $0 \leq 1$

 $V_j(\sigma) \leq (n-j)$ for $1 \leq j < n$. This induces the next alternative representation for the problem:

Definition 1 Given n buckets of capacity $n-1, n-2, n-3, \ldots, 1, 0$, distribute d indistinguishable balls in such a way that any possible configuration is equally probable.

Since, as shown in Equation 2, the number of permutations of n items at Kendall distance d is $S_k(n,d)$, then, the number of $V(\sigma)$ vectors such that $\sum_{j=1}^{n-1} V_j(\sigma) = d$ is $S_k(n,d)$. Equivalently, the number of ways to distribute d indistinguishable balls into n buckets of capacity $n-1, n-2, n-3, \ldots, 1, 0$ is $S_k(n,d)$.

We approach the problem in Definition 1 with a recursive procedure. Its base case is d=0 where nothing is done. In the general case, we introduce k balls in the first bucket and then recursively solve the problem with the rest of the n-1buckets and d-k balls. The main question is how to choose k if we want every configuration to be equally probable. Note that if we introduce 1 ball in the first bucket and try to recursively solve the problem with the remaining buckets and balls, then we will have to choose one of the $S_k(n-1,d-1)$ configurations of inserting the rest of the d-1 balls into the n-1 buckets. If we insert two balls in the first bucket, we will have $S_k(n-1,d-2)$ possible configurations for the rest of the buckets. In general, the insertion of k balls in the first bucket implies that there will be $S_k(n-1,d-k)$ possible ways to introduce the d-k remaining balls into the n-1 remaining buckets. In other words, of the $S_k(n,d)$ total possible configurations of distributing d balls in the n buckets, exactly $S_k(n-1,d-k)$ of them will have k balls in the first bucket. Therefore, for a u.a.r. distribution of the balls into the buckets, the probability of introducing k balls in the first bucket is

probability of introducing
$$k$$
 balls in the first bucket $\frac{S_k(n-1,d-k)}{S_k(n,d)}$

The analogy with the generation of the $V(\sigma)$ vector is quite trivial. Instead of the n buckets of capacity $n-1, n-2, n-3, \ldots, 1, 0$ we have a vector $V(\sigma)$ of n-1 positions, each restricted to have values smaller than or equal to $n-1, n-2, n-3, \ldots, 1$. The base case is the situation when d=0. In this case the output and input vectors are the same. Otherwise, we randomly choose k where the probability of setting $P(V_1(\sigma) = k)$ equals $S_k(n-1, d-k)/S_k(n, d)$. Then, we recursively solve the problem of inserting d-k balls into the remaining positions of the vector, $V_2(\sigma), \ldots, V_{n-1}(\sigma)$.

Regarding the computational complexity of the current algorithm, if the values for $S_k(n,d)$ are given, the cost of generating a permutation at Kendall distance d (or equivalently with d inversions) is $O(n^2)$.

Let us now show how to count and randomly generate permutations at a given Cayley distance. Recall that the Cayley distance $d_c(\sigma)$ can be expressed as the number of cycles of σ , in fact equals n minus the number of cycles of σ . Therefore, the number of permutations at Cayley distance d equals the number of permutations with k = n - d cycles. Stirling numbers of the first kind $S_c(n, k)$ count the number of permutations of n items with k cycles. They also appear in the OEIS with code A008275 where the next recurrence for its computation

```
Algorithm 1: generate\_V\_vector(V, d)
```

Generates a $V(\sigma)$ vector where every possible vector is equally probable.

can be found:

$$S_c(n,k) = \begin{cases} 1 & n = 0 \land k = 0 \\ 0 & n = 0 \lor k = 0 \\ S_c(n-1,k-1) - (n-1) * S_c(n-1,k) & \text{otherwise} \end{cases}$$

The cost of computing the Stirling numbers of the first kind is $O(n^2)$.

We also include in this package a recursive process for the u.a.r. generation of a permutation at Cayley distance d. The generation of a permutation of n items and k cycles implies first, the recursive generation of a permutation of the first n-1 items and then, the insertion of item n. This is due to the fact that the whole set of permutations of n items and k cycles can be split in two groups: The group of permutations in which item n is in a cycle of length one $(\sigma(n) = n)$ and the group of permutations in which item n is in a cycle of length greater than one $(\sigma(n) \neq n)$. If the cardinality of those sets can be given, the generation of the permutation can be efficiently performed.

Let us now show how the algorithm, whose pseudocode can be found in 2, performs. In its base case, k = 1. In this case the algorithm generates a single cycle with the items in the permutation. In the general case, k > 1, the algorithm will randomly choose one of the next options:

- Recursively generate a permutation of n-1 items and k-1 cycles and set $\sigma(n) = n$, so item n is alone in its own cycle.
- Recursively generate a permutation of n-1 items and k cycles. Then, u.a.r. select $1 \le i \le n-1$ and set $\sigma(i) = n$ and $\sigma(n) = i$. In this way, items n is merged in a cycle of length greater than one.

The probability of selecting one or another will be proportional to the number of permutations that can be built in each way. Note that there are $S_c(n,k)$ permutations of n items and k cycles. This group can be split in two: the group of permutations in which $\sigma(n) = n$, which is built by taking the first option, and the group of permutations in which $\sigma(n) \neq n$, which is built by taking the second option. By choosing the first option the algorithm generates a permutation of n-1 items and k-1 cycles. There are $S_c(n-1,k-1)$ of this kind. On the other hand, the second path implies generating one of the $S_c(n-1,k)$ permutations

of n-1 items and k cycles. Moreover, one of those n-1 items is chosen to be inserted in position n, so the number of permutations that can be generated in the second path is $S_c(n-1,k)*(n-1)$. Therefore, the probability of selecting the first path equals $S_c(n-1,k)/S_c(n,k)$ and the probability of selecting the second path is $1 - S_c(n-1,k)/S_c(n,k) = S_c(n-1,k)*(n-1)/S_c(n,k)$.

The computational cost of generating a random permutation at Cayley distance d from the identity given the Stirling numbers is O(n).

Algorithm 2: $generate_permu(n, k)$

This algorithm generates a permutation of n items with k cycles. Note that every permutation of n items with k cycles is equally probable.

```
Input: n, num, of items: k, num, of cycles
Output: \pi, permutation of n elements with k cycles
if k=1 then \pi=generate a cycle with the n elements; /* base case */
   prob = S(n-1, k-1)/S(n, k);
   with probability prob /* n stands in a cycle alone
                                                                     */
       \pi(1\dots n-1) = generate\_permu(n-1,k-1);
      \pi(n) = n;
   end
   otherwise /* n is in a cycle with other items
      \pi(1...n-1) = qenerate\_permu(n-1,k);
      ran = random number in the range [1, n-1];
      \pi(ran) = n;
      \pi(n) = ran;
   \quad \mathbf{end} \quad
end
return \pi;
```

This package includes several functions for counting and generating permutations. The random permutation generation is done with the function runif.permu(permu.length). The generic count.permus.dist.d(permu.length, d, method='kendall') counts the number of permutations of permu.length items at distance d when the particular metric is given by method. For the particular case of the Cayley distance the function count.permus.k.cycles(permu.length, k) is also helpful.

The generation of permutations at a given distance is supported via the function r.dist.d(permu.length, n, d, method='kendall'). Also, a for the particular case of the Cayley distance there is a function r.permu.k.cycles(permu.length, k).

3 Dealing with distributions over permutations

This section introduces the Mallows model and its most popular extension, the Generalized Mallows model. Both of them are usually refer to as distance-based model because both of them define a mode, σ_0 and the probability of any other permutation depends on its distance to σ_0 .

3.1 Mallows model

The Mallows model is one of first probability models for rankings or permutations. However, it is still one of the most used models in both theoretical and applicated papers. Just two parameters are required in its definition: the central permutation σ_0 and the spread (or dispersion) parameter θ . The central permutation is the mode of the distribution, i.e. the permutation at which the probability distribution takes it maximum value. The probability of any other permutation decays exponentially as its distance to the central permutation increases. The spread parameter controls how fast this fall happens. It can be expressed as follows:

$$p(\sigma) = \frac{exp(-\theta d(\sigma, \sigma_0))}{\psi(\theta)}$$

where $\psi(\theta) = \sum_{\sigma} exp(-\theta d(\sigma, \sigma_0))$. Note that when the dispersion parameter θ is greater than 0, then σ_0 is the mode. On the other hand, with $\theta = 0$ we obtain the uniform distribution and when $\theta < 0$ then σ_0 is the anti mode.

The computation of the normalization constant $\psi(\theta)$ is infeasible for medium size values of n. The good news is that this constant can be factored for both Kendall and Cayley distances, as explained in the next section.

A particular Mallows model is defined via the function dmm(mode, theta) and the probability of a particular permutation under a given Mallows model is obtained with the function proba(permu, distri, method='kendall').

3.2 Generalized Mallows model

This extension of the MM tries to break the restriction that imposes every permutation at the same distance to have the same probability value. Instead of one single spread parameter, it requires the definition of n-1 spread parameters θ_j for $1 \leq j < n$, each affecting a particular position of the permutation. This allows modeling a distribution with more emphasis on the consensus of certain positions of the permutation while having more uncertainty in some others. This model is more restrictive regarding the distance metric for permutations since it requires the metric to be decomposed in n-1 terms as follows:

$$d(\sigma, \sigma_0) = \sum_{j=1}^{n-1} S_j(\sigma \sigma_0^{-1})$$
(3)

For any distance that decomposes as the above equation, the GMM is defined as follows:

$$p(\sigma) \propto \prod_{j=1}^{n-1} exp(-\theta_j S_j(\sigma \sigma_0^{-1}))$$

As shown in previous sections both Kendall and Cayley distances can be decomposed as in Equation (3), see Equations (1) and (2). If $\{S_1(\pi), \ldots, S_{n-1}(\pi)\}$ are independent random variables when π is uniformly at random drawn from the set of permutations of n items, the normalization constant can be factorized as $\psi(\theta) = \prod_{j=1}^{n-1} \psi_j(\theta_j)$.

When π is uniformly at random drawn from the set of permutations of n items, $\{V_1(\pi), \ldots, V_{n-1}(\pi)\}$ are independent random variables, [Fligner and Verducci(1986)].

Therefore, the MM for the Kendall distance can be expressed as follows:

$$p(\sigma) = \prod_{j=1}^{n-1} \frac{exp(-\theta_j V_j(\sigma \sigma_0^{-1}))}{\psi_j(\theta_j)} \qquad \text{where } \psi_j(\theta_j) = \frac{1 - exp(-\theta_j(n-j+1))}{1 - exp(-\theta_j)}$$
(4)

Also, $\{X_1(\pi), \ldots, X_{n-1}(\pi)\}$ are independent Bernoulli random variables with parameters $P(X_j(\pi) = 1) = (n-j)/(n-j+1)$ for $j = 1, \ldots, n-1$ when π is u.a.r. drawn from the set of permutations of n items, [Feller(1968)], [Fligner and Verducci(1986)]. Therefore, when the metric for permutations is the Cayley distance, the GMM is expressed as follows:

$$p(\sigma) = \prod_{j=1}^{n-1} \frac{exp(-\theta_j X_j(\sigma \sigma_0^{-1}))}{\psi_j(\theta_j)} \quad \text{where } \psi_j(\theta_j) = (n-j)exp(-\theta_j) + 1 \quad (5)$$

By taking the moment generating function as in [Fligner and Verducci(1986)], we can give the probability of each random variable for the GMM, what will be very helpful to sample the distribution, as will be explained in the next section. Under Kendall the probability of each $V_i(\sigma\sigma_0^{-1}) = r$ is given by:

$$p(V_j(\sigma\sigma_0^{-1}) = r) = \frac{exp(-\theta_j r)}{\psi_j(\theta_j)}$$
(6)

The probability of each random variable under the Cayley distance, on the other hand is as follows:

$$p(X_j(\sigma\sigma_0^{-1}) = 1) = \frac{(n-j)exp(-\theta_j)}{\psi_j(\theta_j)}$$
(7)

Note that Equation 7 is a correction of that given in [Fligner and Verducci(1986)]. Since the MM is the particular case of the GMM where every θ_j has equal value, all these expressions can be also used for the MM.

A particular Generalized Mallows model is defined via the function dmm(mode, theta) and the probability of a particular permutation under a given Generalized Mallows model is obtained with the function proba(permu, distri, method='kendall').

4 Sampling

The present package implements three different algorithms for generating permutations from a given distribution. The first two, Distances and Multistage, sample directly from the distribution, while the last one, the Gibbs sampler, samples a Markov chain whose stationary distribution is the distribution of interest.

4.0.1 Distances sampling algorithm

The Distances sampling algorithm has been used for the generation of a MM under Cayley metric [?]. Here we also use it for the generation of permutations from a MM under Kendall. It can not be used to sample a GMM model since it s based on the fact that every permutation at the same distance from the central permutations has the same probability.

This sampling method proceeds as follows. Let the metric $d(\sigma, \sigma_0)$ range between 0 and d-max for permutations of n items. Also, let S(n, d) be the number of permutations at distance d from the identity permutation¹. Then, the probability of a permutation at distance d is as follows:

$$p(\sigma|d(\sigma,\sigma_0) = d) = \frac{S(n,d)exp(-\theta d)}{\psi(\theta)}$$
(8)

Note that the normalization constant $\psi(\theta) = \sum_{\sigma} exp(-\theta d(\sigma \sigma_0^{-1}))$ can be expressed as the sum of d-max terms in the following way:

$$\psi(\theta) = \sum_{d=0}^{d-max} S(n, d) exp(-\theta d)$$

Taking into consideration the previous expressions of the probability function, the process of simulating from the distribution can be done in three stages:

- Randomly select the distance at which the permutation will lay using Equation (8). Note that the proposed equation is a function on the number of permutations at each possible distance. We remind the reader that the functions to count permutations at each Kendall or Cayley distance are also implemented by the present package and explained in Section 2.1.
- Pick uniformly at random a permutation π at distance d from the identity permutation e, i.e. $d(\pi) = d$. This step relays on the u.a.r. generation of a permutation at a given distance. The random generation of permutations at a given Kendall or Cayley distance is implemented in this package as detailed in Section 2.1 of the current manuscript.
- In case $\sigma_0 = e$ then π is output. Otherwise, Kendall's and Cayley's invariance property lets us obtain $\sigma = \pi \sigma_0$ since $d = d(\pi) = d(\pi \sigma_0, \sigma_0) = d(\sigma, \sigma_0)$.

The computational complexity of the first step is $O(d_max)$ given the count of the number of permutations at each distance. The second step, which implies the random generation of a permutation at a given distance, is also $O(d_max)$ given the count of the number of permutations at each distance. Finally, the last step is done in time O(n). Therefore, we can conclude that this is a quick as well as precise algorithm for the simulation of the MM. However, it does not work with the GMM. Also, as n increases so does the count of the number of permutations at each distance in which the first two steps relay, being impossible to store them with standard programming libraries for n > 150.

This sampling algorithm is supported by the function rmm(n, model, method='kendall', sampling.method='distances') for the MM and rgmm(n, model, method='kendall', sampling.method='distances') for the GMM.

4.0.2 Multistage sampling algorithm

This section exploits the fact that each term in which the distance $d(\sigma)$ decomposes is a random variable and that the probability of each random variable for

 $^{^{1}}$ recall that every permutation has the same number of permutations at distance d.

the GMM is given by Equations (6) and (7) for Kendall and Cayley distances respectively. Since the MM is the particular case of the GMM in which every θ_j is equal, this sampling algorithm can be used to generate permutations under both MM and GMM.

Using this algorithm, the sampling process can be divided in three stages, namely:

- Randomly generate a $V(\pi)$ (respectively $X(\pi)$) vector by using Equation (6) for the Kendall distance (resp. Equation (7) for the Cayley distance).
- Using the techniques in Section 2 generate a permutation π consistent with the given distance decomposition vector.
- In case $\sigma_0 = e \pi$ is output. Otherwise, we get the final permutation by composing π with σ_0 , obtaining $\sigma = \pi \sigma_0$ since $\pi = \sigma_0 \sigma_0^{-1}$.

The computational complexity of the first step is O(n). The second step, on the other hand, is harder to compute. The generation of a permutation consistent with a given vector decomposition is detailed at the end of Section 2. In that section it is shown that the cost is $O(n^2)$ for both Kendall and Cayley distances.

As a summary we can state that this method is not as fast as the Distances sampler of the previous section. However, it can generate from both MM and GMM. Moreover, it can efficiently handle distributions on permutations of large n.

This sampling algorithm is supported by the function rmm(n, model, method='kendall', sampling.method='multistage') for the MM and rgmm(n, model, method='kendall', sampling.method='multistage') for the GMM.

4.0.3 Gibbs sampling algorithm

The Gibbs sampler is a Markov Chain Montecarlo algorithm based on sampling a Markov chain whose stationary distribution is the distribution of interest. Therefore, it is an approximated algorithm for the simulation of the distribution. We have adapted this algorithm to generate samples for both MM and GMM under both Kendall and Cayley metrics.

The Gibbs sampler under each distance considers a particular neighborhood for permutations. Under the Kendall distance the neighborhood of a permutation is the set of permutations that result of randomly selecting a position $1 \le i < n$ and swapping $\sigma(i)$ and $\sigma(i+1)$. On the other hand, for the Gibbs algorithm under the Cayley distance the neighborhood of a permutation is the set of permutations that result of randomly selecting two positions $1 \le i < j \le n$ and swapping $\sigma(i)$ and $\sigma(j)$.

The Gibbs algorithm proceeds as follows:

- 1. Generate uniformly at random a permutation $\sigma.$
- 2. Select uniformly at random a permutation σ' in the neighborhood of σ .
- 3. Let $\gamma = min\{1, p(\sigma')/p(\sigma)\}$. With probability γ the algorithm accepts the candidate permutation moving the chain to the candidate permutation, $\sigma = \sigma'$, and goes back to 2. Otherwise, it discards σ' and goes back to step 2.

The initial samples are discarded (burn-in period) until the Markov chain approaches its stationary distribution and so samples from the chain are samples from the distribution of interest. Then, the above process in repeated until the algorithm generates a given number of permutations. Recall that so far we assumed that the central permutation is the identity, e = [123...n]. If not, we can center the sample around σ_0 by composing each of the permutations π in the sample with σ_0 , obtaining $\pi\sigma_0$.

Let us now focus on the computational complexity of the Gibbs sampler. Under the Kendall distance, the new permutation σ' is obtained by swapping two adjacent items from σ , say i and i+1. Therefore, if $\sigma(i) > \sigma(i+1)$ then $\sigma'(i) < \sigma'(i+1)$ and the new solution is accepted. On the other hand, if $\sigma(i) < \sigma(i+1)$, then $\sigma'(i) > \sigma'(i+1)$ and thus $V_i(\sigma') = V_i(\sigma) + 1$. In this case, under the MM the probability of accepting the new solution σ' is $exp(-\theta)$ while under the GMM is as follows:

$$exp(-\theta_i * v_{i+1}(\sigma) - \theta_{i+1} * v_i(\sigma) + \theta_i * v_i(\sigma) + \theta_{i+1} * v_{i+1}(\sigma))$$

The complexity of each iteration is thus O(n).

When dealing with the Cayley distance, on the other hand, a new permutation σ' is built by swapping 2 items $1 \leq i < j \leq n$ of σ . If both items were part of the same cycle in σ then after the swap the cycle has been split into two new cycles and each swapped item is in a different cycle in σ' . In this case the distance decreases in one unit and the chain moves to the new permutation σ' . On the other hand, in case both items i and j were in different cycles in σ then after the swap, both cycles are merged into a single one in σ' . In this case, the distance has been increased in one unit and the chain moves to σ' with probability $p(\sigma')/p(\sigma)$. Under the MM this ratio equals $exp(-\theta)$. Under the GMM, where the probability of a permutation is $p(\sigma) \propto -\sum_{j=1}^{n-1} \theta_j X_j(\sigma \sigma_0^{-1})$ the ratio equals $exp(-\theta_k)$ where k is the item such that $X_k(\sigma \sigma_0^{-1}) = 0$ and $X_k(\sigma' \sigma_0^{-1}) = 1$. Therefore, under the GMM it is not necessary to compute the entire $\mathbf{X}(\sigma' \sigma_0^{-1})$ vector but just the $X_k(\sigma' \sigma_0^{-1})$ of the items k in the cycles of the swapped items. The computational complexity of generating each permutation is thus O(n).

Summarizing, the Gibbs sampler can generate samples for both MM and GMM under both Kendall and Cayley distances. Moreover, it is very fast, performs an iteration in time O(n). However, we should emphasize the fact that this an approximated sampling algorithm.

This sampling algorithm is supported by the function rmm(n, model, method='kendall', sampling.method='gibbs') for the MM and rgmm(n, model, method='kendall', sampling.method='gibbs') for the GMM.

5 Learning

In this section we deal with the maximum likelihood estimation of the parameters of the distribution given a sample of m i.i.d. permutations $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$. The log likelihood of the GMM is given by

$$Ln \mathcal{L}(\{\sigma_1, \sigma_2, \dots, \sigma_m\} | \sigma_0, \boldsymbol{\theta}) = \sum_{s=1}^m Ln \ p(\sigma_s | \sigma_0, \boldsymbol{\theta})$$

Although the MM is a particular case of the GMM in which every θ_j has the same value, the calculation of the maximum likelihood parameters are different for each model. Moreover, its expression differs regarding the distance on the permutations considered. In this way, we will describe the maximum likelihood estimation for each model and distance separately. Finally, we introduce algorithms to fit the parameters of a given sample.

5.1 Mallows model

In the case of the MM the likelihood expression is given by the next equation.

$$Ln \mathcal{L}(\{\sigma_1, \sigma_2, \dots, \sigma_m\} | \sigma_0, \theta) = Ln \prod_{s=1}^m \frac{exp(-\theta d(\sigma_s \sigma_0^{-1}))}{\psi(\theta)}$$
$$= -\theta \sum_{s=1}^m (d(\sigma_s \sigma_0^{-1})) - mLn \psi(\theta) \qquad (9)$$

By looking at Equation (9), we can see that calculating the value of σ_0 that maximizes the equation is independent of θ . Therefore the maximum likelihood estimation problem for the MM can be posed as a two step process in which first the central permutation is obtained and then the dispersion parameter for the given $\hat{\sigma}_0$ is calculated.

The maximum likelihood estimator (MLE) for the consensus permutation is given by the next equation.

$$\hat{\sigma}_0 = \arg\max_{\sigma_0} \sum_{s=1}^m -d(\sigma_s \sigma_0^{-1}) = \arg\min_{\sigma_0} \sum_{s=1}^m d(\sigma_s \sigma_0^{-1})$$

Problems consisting on finding the permutation that minimizes the sum of the distances to the permutations in the sample are often called median problems. Obviously, the solution will depend on whether the metric is Kendall or Cayley. Moreover, the MLE for the dispersion parameter θ will also differ for each distance. For these reasons, we will deal with each distance separately.

5.1.1 Kendall distance

The problem of finding the permutation that minimizes the sum of the Kendall distance to the permutations in a given sample appeared first in the voting domain. It is also referred to as consensus ranking or Kemeny rank aggregation problem. It has been shown to be an NP-hard problem in [Bartholdi et al.(1989)Bartholdi, Tovey, and Trick].

Once the consensus permutation $\hat{\sigma}_0$ is known, the second and last stage of the learning process of a MM concerns the estimation of the spread parameter. The MLE for the dispersion parameter, $\hat{\theta}$, is the θ that satisfies the following expression:

$$\frac{n-1}{\exp(\theta) - 1} - \bar{d} - \sum_{k=2}^{n} \frac{k \exp(-\theta k)}{1 - \exp(-\theta k)} = 0$$
 (10)

where $\bar{d} = \sum_{i=0}^{m} d_k(\sigma_i, \hat{\sigma}_0)/m$. This expression is obtained by deriving the likelihood in Equation (9) and making it equal zero. Recall that the normalization constant for the Kendall distance is given in Equation (4). Although there

is no closed expression for the MLE for θ , the solution to this equation can be easily calculated with numerical methods such as Newton-Raphson. This package implements a learning algorithm for the MM under the Kendall distance. This algorithm and many others are discussed in Section 5.3.1.

5.1.2Cayley distance

As stated, the learning problem is done in two separate stages. First, we look for the mode of the distribution, σ_0 and then calculate the dispersion parameter.

The problem of finding the median permutation under the Cayley distance is also called swap median problem. Its computational complexity is an open problem although it is supposed to be NP-complete. Note that the problem can be posed as a function of the $X(\sigma\sigma_0^{-1})$ vector as follows:

$$\hat{\sigma}_0 = \arg\min_{\sigma_0} \sum_{s=1}^m d(\sigma_s \sigma_0^{-1}) = \arg\min_{\sigma_0} \sum_{i=1}^{n-1} \bar{X}_i$$

where $\bar{X}_j = \sum_{s=1}^m X_j(\sigma_s \sigma_0^{-1})/m$. Suppose that the consensus permutation $\hat{\sigma}_0$ is known, the second and last stage of the learning process of a MM concerns the estimation of the spread parameter. The MLE for the dispersion parameter is the θ that satisfies the following expression:

$$\sum_{i=1}^{n-1} \frac{j}{j + exp(\theta)} = \frac{\sum_{s=1}^{m} d(\sigma_s \hat{\sigma}_0^{-1})}{m}$$

This expression is obtained by deriving the likelihood in Equation (9) and making equal to zero and taking into account that the normalization constant $\psi(\theta)$ is given in Equation (5). Although there is no closed expression for θ , the solution to this equation can be easily calculated with numerical methods such as Newton-Raphson.

There is an heuristic and an exact algorithm for the MLE of the parameters of the MM under the Cayley distance in Section 5.3.2.

5.2 Generalized Mallows model

In this section we deal with the expression of the likelihood under the GMM. Recall that the GMM can consider any distance that can be decomposed in n-1 terms as expressed in Equation (3). In particular, for both Kendall and Cayley metrics, the likelihood of the GMM model is given as follows:

$$Ln \mathcal{L}(\{\sigma_{1}, \sigma_{2}, \dots, \sigma_{m}\} | \sigma_{0}, \boldsymbol{\theta}) = Ln \prod_{s=1}^{m} p(\sigma_{s})$$

$$= \sum_{j=1}^{n-1} \sum_{i=1}^{m} -\theta_{j} S_{j} + \sum_{i=1}^{m} \sum_{j=1}^{n-1} Ln \ \psi_{j}(\theta_{j})$$

$$= -m \left(\sum_{j=1}^{n-1} \theta_{j} \bar{S}_{j} + \sum_{j=1}^{n-1} Ln \ \psi_{j}(\theta_{j}) \right) = \sum_{j=1}^{n-1} (\mathbf{L}_{\frac{1}{2}})$$

where $\bar{S}_j = \sum_{s=1}^m S_j(\sigma_s \hat{\sigma}_0^{-1})/m$. Note that the learning process of the GMM can not be divided in two separate stages, so an exact algorithm must look simultaneously for every parameter. The distance decomposition $\{S_1(\pi),\ldots,S_{n-1}(\pi)\}$ is different for the Kendall and Cayley distances and thus, so will be the likelihood expression. Therefore, we will discuss the expression for each distance separately.

5.2.1 Kendall distance

Recall that the Kendall distance decomposition is $d_k(\sigma) = \sum_{j=1}^{n-1} V_j(\sigma)$ as defined in Equation (1). Also, the expression of the normalization constant $\psi_i(\theta_i)$ for the GMM under the Kendall distance is given in Equation (4). By combining Equations (11) and (4) the following expression for the likelihood of the GMM under the Kendall distance is obtained.

$$\sum_{j=1}^{n-1} \mathcal{L}_{j} = \sum_{j=1}^{n-1} -m \left(\theta_{j} \bar{V}_{j} + Ln(1 - exp(-\theta_{j}k)) - Ln(1 - exp(-\theta_{j})) \right)$$

where $\bar{V}_j = \sum_{s=1}^m V_j(\sigma_s \hat{\sigma}_0^{-1})/m$. For any given σ_0 , the MLE for the spread parameters, θ , is given by the equalling to zero the derivative of the previous expression, and is as follows:

$$\sum_{i=1}^{n-1} \left(\bar{V}_j + \frac{k}{exp(\theta_j k) - 1} - \frac{1}{exp(\theta_j) - 1} \right) = 0$$
 (12)

One can find in Section 5.3.1 how to use our package to approximately solve this problem.

Cayley distance

The Cayley distance decomposition is $d_c(\sigma) = \sum_{j=1}^{n-1} X_j(\sigma)$ as defined in Equation (2). Also, Equation (5) expresses the normalization constant $\psi_j(\theta_j)$ for the GMM under the Cayley metric. By combining Equations (11) and (5) we obtain the expression for the likelihood of the GMM under the Cayley distance, which is as follows.

$$\sum_{j=1}^{n-1} \mathcal{L}_j = \sum_{j=1}^{n-1} -\bar{X}_j Ln(n-j) + (n-j)^2 + \frac{(n-j)^2}{\bar{X}_j} + \bar{X}_j Ln \frac{\bar{X}_j}{1-\bar{X}_j} - m \quad (13)$$

where $\bar{X}_j = \sum_{s=1}^m X_j(\sigma_s \hat{\sigma}_0^{-1})/m$ Assuming that the MLE for σ_0 is known, the MLE for the dispersion parameters are computed by the equalling to zero the derivative of Equation (13). In this case, there is a closed expression, which is as follows:

$$\hat{\theta}_{i} = Ln(n-j) - Ln(\bar{X}_{i}/(1-\bar{X}_{i}))$$
 (14)

By combining Equations (13) and (14) we raise the following expression for the GMM under the Cayley distance.

$$\hat{\sigma}_{0} = \arg\max_{\sigma_{0}} Ln \ \mathcal{L}(\{\sigma_{1}, \sigma_{2}, \dots, \sigma_{m}\} | \sigma_{0}, \boldsymbol{\theta}) = \arg\max_{\sigma_{0}} \sum_{j=1}^{n-1} \mathcal{L}_{j} = \arg\max_{\sigma_{0}} \sum_{j=1}^{n-1} [-\bar{X}_{j} Ln(n-j) + (n-j)^{2} + \frac{(n-j)^{2}}{\bar{X}_{j}} + \bar{X}_{j} Ln \frac{\bar{X}_{j}}{1 - \bar{X}_{j}} - m]$$
(15)

An exact learning algorithm would search the space of permutations looking for the σ_0 that maximizes the above expression. Section 5.3.2 describes how to use our proposed package to obtain an exact solution in an efficient way where there is some consensus in the sample. We also provide an heuristic algorithm for the situations where there is no consensus.

5.3 Learning algorithms

5.3.1 Kendall distance

As stated in Section 5.1.1, the MLE of the parameters $\hat{\sigma}_0$ and $\hat{\theta}$ for MM under the Kendall distance is done by, first, obtaining the consensus permutation $\hat{\sigma}_0$ and then, with Equation (10) obtaining $\hat{\theta}$ for the given $\hat{\sigma}_0$. The main difficulty is founded on the estimation of the consensus permutation.

The search for the consensus permutation $\hat{\sigma}_0$ can be done in an exhaustive or an heuristic way. A comprehensive comparison of more than a hundred methods for the consensus ranking problem can be found in [Ali and Meila(2011)]. They conclude that the well known Borda algorithm, [Borda(1781)], an heuristic algorithm, offers a very good trade-off between accuracy and time. In this package we include the Borda algorithm. Borda builds a central ranking $\hat{\sigma}_0$ by, first, calculating the average at each position i of the permutations σ_s $1 \le s \le m$ in the sample, $\bar{\sigma}(i) = \sum_{s=0}^m \sigma_s(i)$, and second, sorting the items in $\hat{\sigma}_0$ according to $\bar{\sigma}(i)$. It is a fast algorithm, it obtains a solution in time O(n). For the MM, this algorithm is asymptotically optimal as shown in [Fligner and Verducci(1986)].

This package does not implement any exact algorithm for the consensus ranking problem. The interested reader can find in [Ali and Meila(2011)] the description and performance analysis of several exact and approximate algorithms.

The problem of the exact MLE for the parameters of a GMM must be done simultaneously for every parameter, as shown in Section 5.2.1. The first attempt to give an exact solution for this problem was that in [Meila et al.(2007)Meila, Phadnis, Patterson, and Bilmes However, a similar but much more efficient algorithm is proposed in [Mandhani and Meila(2009)]. The latter proposes an A* search method over the space of permutations with a non-trivial admissible heuristic function.

Approximated solutions for the estimation of the MLE for the parameters of a GMM, on the other hand, can be found in the packages prm and RMallows.

Summarizing, we do not consider the exact learning of the parameters neither for MM nor form GMM in this package. We offer however an approximate learning of the parameters of the distribution by splitting the problem into two smaller problems:

- Approximately estimating $\hat{\sigma}_0$ with the Borda algorithm and then
- Estimating the dispersion parameters for the given $\hat{\sigma}_0$ with Equation (10) for the MM or Equation (12) for the GMM.

The MLE for the parameters of a MM and GMM under the Kendall distance can be done with functions lmm(sample, method='kendall') and lgmm(sample, method='kendall') respectively.

5.3.2 Cayley distance

This package includes a heuristic and an exhaustive algorithm to find the MLE for the central permutation and the spread parameters for both MM and GMM. In this section we briefly summarize the algorithms. They are introduced in [?] and the improvements over those algorithms are detailed here.

We briefly summarize the exact algorithm here and refer the interested reader to [?] where the complete details are given since the algorithms are the same in both cases. The same process can be applied to the estimation of a MM and a GMM. The exact algorithm is based on the fact that a certain kind of partial solutions can be evaluated under both MM and GMM. The algorithm explores the set of partial permutations of the first k out of n positions in a branch and bound strategy. We consider as candidate solutions σ_0^{-1} . In this way, the evaluation of these partial permutations σ_0^{-1} of k items is a lower bound on the evaluation of any complete permutation consistent with σ_0^{-1} of k' > k items. Moreover, we can give a lower bound on the likelihood of the rest of the permutation -which is unknown. In this way, the branch and bound strategy results on an efficient searching procedure specially when there is some consensus in the input sample.

The heuristic algorithm proceeds in two stages. The first stage consists on the generation of an initial solution in a greedy way. Then, this initial solution is improved with a metaheuristic search. The initial solution of the heuristic algorithm starts from an empty vector and adds at each step an item to a position of the solution. The process differs regarding we are solving a MM or GMM.

Under the MM at each step the item i that appears most frequently at any position j in the samples is chosen. Therefore, we set $\sigma_0(i) = j$ and proceed in a stage-wise manner until the complete solution is built. Under the GMM on the other hand, we start by start by choosing the most frequent item i at the first position of the inverse of the samples $\sigma_s^{-1}(1)$ and set $\sigma_0^{-1}(1) = i$. Then, the algorithms selects the most frequent item i at the second position of the inverse of the samples $\sigma_s^{-1}(2)$ and set $\sigma_0^{-1}(2) = i$. This approach is justified by the fact that given a partial solution σ_0^{-1} of the first k positions we can exactly evaluate \bar{X}_j for all j < k. Moreover, the \bar{X}_j for low values of j have more weight in the likelihood as one can see by looking to Equation (15).

Once a permutation is obtained, a Variable Neighborhood Search algorithm is run trying to improve the solution until it gets stuck in a local optima for two different given neighborhood systems, the Insert and the Swap.

The Insert neighborhood of a permutation σ is the set of permutations $\{\sigma'_r\}$ that result of removing an item from its position and inserting it in other, shifting any other item if necessary. For example, $\sigma' = \lceil 12456378 \rceil$ is in the

neighborhood of $\sigma = [12345678]$. The Swap neighborhood of a given permutation, on the other hand is the set of permutations that result of swapping two items. For example, $\sigma' = [16345278]$ is in the neighborhood of $\sigma = [12345678]$.

The improvement from the algorithms in [?] comes from the evaluation of the swap neighborhood. Note that there are n^2 neighbors of a given permutation and therefore, the explicit evaluation of each of them is $O(n^3)$ can be inefficient. Therefore, we propose the next algorithm for the selection of the best neighbor.

Let $\tau_{i,j}$ be involution of i and j, i.e., the permutation that swaps items i and j. Then $\sigma' = \sigma \tau_{ij}$ is in the swap neighborhood of σ . Moreover, if i and j are in the same cycle in σ , then they will be in separate cycles in σ' and vice versa. In general, if i and j are in the same cycle of $\sigma_s \sigma_0^{-1}$, then they will be in different cycles in $\sigma_s \sigma_0^{-1} \tau_{ij}$ and vice versa. Therefore, in order to obtain the best neighbor of a particular solution σ_0^{-1} our proposed algorithm just computes the number of permutations in the sample in which each pair of items are in the same cycle in $\sigma_s \sigma_0^{-1}$, denoted as c_{ij} . Then, select the pair of items i and j for which c_{ij} is maximized. The number of cycles in $\sigma_s \sigma_0^{-1} \tau_{ij}$ decrease in $m-2*c_{ij}$ with respect to the number of cycles in $\sigma_s \sigma_0^{-1}$. Therefore, if $c_{ij} > m/2$ then, $\sigma_0 \tau_{ij}$ is closer to the sample than σ_0 . Summarizing, we have transformed the explicit evaluation of the neighbors from $O(n^3)$ to $O(n^2)$.

The MLE for the parameters of a MM and GMM under the Cayley distance can be done with functions lmm(sample, method='cayley') and lgmm(sample, method='cayley') respectively.

6 Usage

ijecho=false;;= library(PermutationsMallows) @ Esto esta en el manual-sweave.rnw

7 Conclusions

In this paper we present an R package for dealing with permutations. It includes several functions for the generation and manipulation of permutations. We pay special attention to those functions related to the Kendall and Cayley distances and the process of counting and generating permutations at a given distance. Moreover, two probability models are considered, the Mallows model and the Generalized Mallows model. Both models can be associated with both Kendall and Cayley distances. This package implements functions to generate from the distributions as well as for the maximum likelihood estimation of the parameters of the distribution.

A future version of this package will include the Hamming and Ulam metrics. We expect this package to be helpful to every kind of user, from the noviced in the field of permutations and/or probability models for permutation spaces to the advanced users. Moreover, the internal code has been written in C++ for the maximum efficiency. Also, the code is public for any researcher which wants to extend or improve it.

References

- [Pre(2013)] (2013). "Special Issue on Preference Learning and Ranking." Machine Learning, 93(2-3).
- [Ali and Meila(2011)] Ali A, Meila M (2011). "Experiments with Kemeny ranking: What works when?" Mathematical Social Sciences Accepted for publication 10.1016/j.mathsocsci.2011.08.008. ISSN 01654896. doi:10.1016/j.mathsocsci.2011.08.008. URL http://linkinghub.elsevier.com/retrieve/pii/S0165489611000989.
- [Arndt(2010)] Arndt J (2010). Generating Random Permutations. Ph.D. thesis, Australian National University.
- [Bader(2011)] Bader M (2011). "The transposition median problem is NP-complete." Theoretical Computer Science, 412(12-14), 1099-1110. doi:http://dx.doi.org/10.1016/j.tcs.2010.12.009. URL http://dblp.uni-trier.de/db/journals/tcs/tcs412.html#Bader11.
- [Bartholdi et al.(1989)Bartholdi, Tovey, and Trick] Bartholdi J, Tovey CA, Trick MA (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://www.springerlink.com/index/10.1007/BF00303169.
- [Borda(1781)] Borda J (1781). Memoire sur les Elections au Scrutin. Histoire de l{}Academie Royal des Sciences.
- [Critchlow et al.(1991)Critchlow, Fligner, and Verducci] Critchlow DE, Fligner MA, Verducci JS (1991). "Probability Models on Rankings." Journal of Mathematical Psychology, 35, 294–318.
- [Feller(1968)] Feller W (1968). An Introduction to Probability Theory and Its Applications, volume 1. Wiley. ISBN 0471257087.
- [Fligner and Verducci(1986)] Fligner MA, Verducci JS (1986). "Distance based ranking models." *Journal of the Royal Statistical Society*, **48**(3), 359–369.
- [Mandhani and Meila(2009)] Mandhani B, Meila M (2009). "Tractable Search for Learning Exponential Models of Rankings." *Journal of Machine Learning Research*, **5**, 392–399.
- [Meila et al.(2007)Meila, Phadnis, Patterson, and Bilmes] Meila M, Phadnis K, Patterson A, Bilmes J (2007). "Consensus ranking under the exponential model." In *Uncertainty in Artificial Intelligence (UAI)*, pp. 285–294. Corvallis, Oregon.
- [Ziegler et al.(2012)Ziegler, Christiansen, Kriegman, and Belongie] Ziegler A, Christiansen E, Kriegman D, Belongie S (2012). "Locally Uniform Comparison Image Descriptor." In P Bartlett, FCN Pereira, CJC Burges, L Bottou, KQ Weinberger (eds.), Advances in Neural Information Processing Systems 25, pp. 1-9. URL http://books.nips.cc/papers/files/nips25/NIPS2012_0012.pdf.