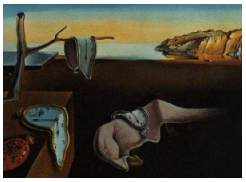
Rankcluster: An R package for clustering multivariate partial rankings

Julien Jacques, Quentin Grimonprez, Christophe Biernacki



Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *http://www.jstatsoft.org/*

Rankcluster: An R Package for clustering multivariate partial ranking

Julien Jacques
Univ. Lille 1, CNRS, Inria

Quentin Grimonprez
Inria

Christophe Biernacki
Univ. Lille 1, CNRS, Inria

Abstract

Rankcluster is the first R package dedicated to ranking data. This package proposes modelling and clustering tools for ranking data, potentially multivariate and partial. Ranking data are modelled by the Insertion Sorting Rank (ISR) model, which is a meaningful model parametrized by a central ranking and a dispersion parameter. A conditional independence assumption allows to take into account multivariate rankings, and clustering is performed by the mean of mixtures of multivariate ISR model. The clusters parameters (central rankings and dispersion parameters) help the practitioners in the interpretation of the clustering. Moreover, the **Rankcluster** package provides an estimation of the missing ranking positions when rankings are partial. After an overview of the mixture of multivariate ISR model, the **Rankcluster** package is described and its use is illustrated through two real datasets analysis.

Keywords: model-based clustering, multivariate ranking, partial ranking, R, **Rankcluster**.

1. Introduction

Ranking data occur when a number of subjects are asked to rank a list of objects according to their personal preference order. Such data are of great interest in human activities involving preferences, attitudes or choices like Psychology, Sociology, Politics, Marketing, *etc.* For instance, the voting system *single transferable vote* occurring in Ireland, Australia and New Zeeland, is based on preferential voting (Gormley and Murphy 2008). In a lot of applications, the study of ranking data discloses heterogeneity, due for instance to different political meanings, different human preferences, *etc.*

Recently, Jacques and Biernacki (2012) proposed a model-based clustering algorithm in order to analyse and explore such ranking data. This algorithm is able to take into account multivariate rankings with potential partial rankings (when a subject did not rank all the

objects). To the best of our knowledge, this is the only clustering algorithm for ranking data with a so wide application scope. This algorithm is based on an extension of the Insertion Sorting Rank (ISR) model (Biernacki and Jacques 2013) for ranking data, which is a meaningful and effective model obtained by modelling the ranking generating process assumed to be a sorting algorithm. The ISR model is parametrized by a position parameter (the modal ranking) and a dispersion parameter. The heterogeneity of the rank population is modelled by a mixture of ISR whereas conditional independence assumption allows the extension to multivariate rankings. Maximum likelihood estimation is performed through a SEM-Gibbs algorithm, in which partial rankings are considered as missing data, what allows to simulate them during the estimation process.

This algorithm has been implemented in C++ and is available through the **Rankcluster** package for R, available on R-forge (and soon on the CRAN website) and presented at long in the sequel of this paper.

The paper is organised as follows: Section 2 briefly presents the clustering algorithm proposed in Jacques and Biernacki (2012). Section 3 discusses the functionalities of the **Rankcluster** package for R, whereas Section 4 illustrates the use of **Rankcluster** through the cluster analysis of two datasets: 1. Fligner and Verducci's *words* dataset (univariate full rankings, Fligner and Verducci (1986)), and 2. the votes of some European countries to the last six editions of the Eurovision song contest (2007–2012) (multivariate partial rankings, Jacques and Biernacki (2012)).

2. Overview of the model-based clustering algorithm

This section gives an overview of the model-based clustering algorithm for multivariate partial rankings proposed in Jacques and Biernacki (2012). It relies on the univariate ISR model that we introduce first.

2.1. The univariate ISR model

Rank data arise when judges or subjects are asked to rank several objects $\mathcal{O}_1, \ldots, \mathcal{O}_m$ according to a given order of preference. The resulting ranking can be designed by its *ordering* representation $x = (x^1, \ldots, x^m) \in \mathcal{P}_m$ which signifies that Object $\mathcal{O}_{x^h}$ is the hth ($h = 1, \ldots, m$), where $\mathcal{P}_m$ is the set of the permutations of the first m integers. Based on the assumption that a rank datum is the result of a sorting algorithm based on paired comparisons, and that the judge who ranks the objects uses the insertion sort because of its optimality properties (minimum number of paired comparisons), Biernacki and Jacques (2013) state the following so-called ISR model:

$$\mathrm{p}(x;\mu,\pi) = \frac{1}{m!}\sum_{y\in\mathcal{P}_m}\mathrm{p}(x|y;\mu,\pi) = \frac{1}{m!}\sum_{y\in\mathcal{P}_m}\pi^{G(x,y,\mu)}(1-\pi)^{A(x,y)-G(x,y,\mu)}, \tag{1}$$

where

- $\mu \in \mathcal{P}_m$, the modal ranking, is a *location parameter*. Its *opposite* ranking $\bar{\mu}$ ($\bar{\mu} = \mu \circ \bar{e}$ with $\bar{e} = (m, \ldots, 1)$) is the rank of smallest probability,

- $\pi \in [\frac{1}{2}, 1]$, which is the probability of good paired comparison according to μ in the sort algorithm, is a *scale parameter*: the distribution is uniform when $\pi = \frac{1}{2}$ and the mode μ of the distribution is uniformly more pronounced when π grows, being a Dirac in μ when $\pi = 1$,

- the sum over $y \in \mathcal{P}_m$ corresponds to all the possible initial presentation orders of the objects to rank (with identical prior probabilities equal to $1/m!$),

- $G(x, y, \mu)$ is equal to the number of good paired comparisons during the sorting process leading to return x when the presentation order is y,

- $A(x, y)$ corresponds to the total number of paired comparisons (good or wrong).

The accurate definitions of $G(x, y, \mu)$ and $A(x, y)$ can be found in Biernacki and Jacques (2013).

2.2. Mixture of multivariate ISR

Let now redefine $x = (x^1, \ldots, x^p) \in \mathcal{P}_{m_1} \times \ldots \times \mathcal{P}_{m_p}$ as a *multivariate* rank, in which $x^j = (x^{j1}, \ldots, x^{jm_j})$ is a rank of m_j objects $(1 \leq j \leq p)$.

The population of multivariate ranks is assumed to be composed of K groups in proportions p_k ($p_k \in [0, 1]$ and $\sum_{k=1}^{K} p_k = 1$). Given a group k, the p components $x^1, \ldots, x^p$ of the multivariate rank datum x are assumed to be sampled from independent ISR distributions with corresponding modal rankings $\mu_k^1, \ldots, \mu_k^p$ (each $\mu_k^j \in \mathcal{P}_{m_j}$) and good paired comparison probabilities $\pi_k^1, \ldots, \pi_k^p \in [\frac{1}{2}, 1]$.

The unconditional probability of a rank x is then

$$\mathrm{p}(x; \boldsymbol{\theta}) = \sum_{k=1}^{K} p_k \prod_{j=1}^{p} \frac{1}{m_j!} \sum_{y \in \mathcal{P}_{m_j}} \mathrm{p}(x^j | y; \mu_k^j, \pi_k^j), \tag{2}$$

where $\boldsymbol{\theta} = (\pi_k^j, \mu_k^j, p_k)_{k=1,\ldots,K\,, j=1,\ldots,p}$ and $\mathrm{p}(x^j | y; \mu_k^j, \pi_k^j)$ is defined by (1).

Each component x^j of x can be full or partial. Frequently, the objects in the top positions will be ranked and the missing ones will be at the end of the ranking, but our model does not impose such situation and is able to work with partial ranking whatever are the positions of the missing data (see details in Jacques and Biernacki (2012)).

2.3. Estimation algorithm

Let $\mathbf{x} = \{x_1, \ldots, x_n\}$ be a sample of n multivariate rankings, and $\mathbf{z} = \{z_1, \ldots, z_n\}$ the corresponding latent cluster memberships. Let $\check{I}_i^j$ and $\hat{I}_i^j$ be respectively the sets of indices of observed and unobserved positions in the jth component x_i^j of the ith observation x_i. Similarly, let $\check{x}_i^j$ and $\hat{x}_i^j$ correspond to the previous notations for the jth component of the ith observation, $\check{x}_i = \{\check{x}_i^1, \ldots, \check{x}_i^p\}$ and $\hat{x}_i = \{\hat{x}_i^1, \ldots, \hat{x}_i^p\}$. Let also define $\check{\mathbf{x}} = \{\check{x}_i; i = 1, \ldots, n\}$ and $\hat{\mathbf{x}} = \{\hat{x}_i; i = 1, \ldots, n\}$. Finally $y_i = (y_i^1, \ldots, y_i^p) \in \mathcal{P}_{m_1} \times \ldots \times \mathcal{P}_{m_p}$ denotes the presentation orders of the objects for the ith observation and $\mathbf{y} = \{y_1, \ldots, y_n\}$.

Assuming that triplets (x_i, y_i, z_i) arise independently $(i = 1, \ldots, n)$, the observed-data log-likelihood of model (2) is:

$$l(\boldsymbol{\theta}; \check{\mathbf{x}}) = \sum_{i=1}^{n} \ln \left(\sum_{k=1}^{K} p_k \prod_{j=1}^{p} \frac{1}{m_j!} \sum_{y \in \mathcal{P}_{m_j}} \sum_{x \in \mathcal{X}_i^j} \mathrm{p}(x|y; \mu_k^j, \pi_k^j) \right),$$

where $\mathcal{X}_i^j = \{x \in \mathcal{P}_{m_j} : x^h = \check{x}_i^{jh}, \forall h \in \check{I}_i^j\}$ is the set of all the rankings compatible with the observed part $\check{x}_i^j$ of x_i^j.

Maximum likelihood estimation is not straightforward since several missing data occur: the cluster memberships z_i of the observations, the presentation orders y_i and the unobserved ranking positions $\hat{x}_i$ (for partial rankings). In such a situation, a convenient way to maximize the likelihood is to consider an EM algorithm (Dempster *et al.* 1977). This algorithm relies on the completed-data log-likelihood, and proceeds in iterating an E step, in which the conditional expectation of the completed-data log-likelihood is computed, and a M step, in which the model parameters are estimated by maximizing the conditional expectation computed in the E step. Unfortunately, the EM algorithm is tractable only for univariate full rankings with moderate m ($m \leq 7$), respectively for mathematical and numerical reasons. In particular, when partial rankings occur, the E step is intractable since the completed-data log-likelihood is not linear for all three types of missing data (refer to Jacques and Biernacki (2012) for its expression). A SEM-Gibbs approach is then proposed in Jacques and Biernacki (2012) to overcome these problems.

The fundamental idea of this algorithm is to reduce the computational complexity that is present in both E and M steps of EM by removing all explicit and extensive use of the conditional expectations of any product of missing data. First, it relies on the SEM algorithm (Geman and Geman 1984; Celeux and Diebolt 1985) which generates the latent variables at a so-called stochastic step (S step) from the conditional probabilities computed at the E step. Then these latent variables are directly used in the M step. Second, the advantage of the SEM-Gibbs algorithm in comparison with the basic SEM ones relies on the fact that the latent variables are generated without calculating conditional probabilities at the E step, thanks to a Gibbs algorithm. Refer to Jacques and Biernacki (2012) for more details.
Let noticed that label switching can occur with the SEM algorithm when clusters are not well separated. To avoid this situation, we recommend to use model selection criteria as BIC (Schwarz 1978) or ICL (Biernacki *et al.* 2000) to select the number K of clusters.

3. Overview of the Rankcluster functions

This section presents first the main function, `rankclust()`, which performs cluster analysis, and second several companion functions which can be helpful for additional ranking data analysis.

3.1. The main function: `rankclust()`

Cluster analysis can be performed with the `rankclust()` function. Illustration of its use is given in Section 4.

Input arguments

This function has only one mandatory argument, `data`, which is a matrix composed of the n observed ranks in their ordering representation. For **univariate rankings** the number of columns of `data` is m (default value of argument `m`). For **multivariate rankings**, `data` has $m_1 + \ldots + m_p$ columns: the first m_1 columns contain x^1 (first dimension), the columns $m_1 + 1$ to $m_1 + m_2$ contain x^2 (second dimension), and so on. In this case, the argument `m` must be filled with the vector of sizes $(m_1, \ldots, m_j)$. If the user works with a **ranking**[1] **representation** of the ranks, the `convertRank()` function can be used to transform ranking representation into ordering one.

The number of clusters (1 by default) can be set up with option `K`. Vector of numbers of clusters are possible (for instance `K=1:10`). In order to select the number of clusters, two criteria are available: BIC, by default, and ICL, selected by `criterion = "icl"`.

Additionally, several parameters allow to set up the different tuning parameters (iterations numbers) used in the SEM-Gibbs estimation. Refer to Jacques and Biernacki (2012) and to `rankclust()` help for more details. Section 4 gives also some examples of iterations numbers choices. The option `run` allows to set the number of initializations of the SEM-Gibbs algorithm (1 by default). In the case of multiple initializations, the best solution according to the approximated log-likelihood is retained.

Finally, the computing times (total, for the SE and M steps and for the likelihood approximation) can be printed by setting the option `detail` to `TRUE` (`FALSE` by default).

Output arguments

The `rankclust()` function returns an instance of the `ResultTab` class. Its attributes will contain 4 slots:

- `K`: a vector of the number of clusters,
- `results`: a list of `ResultList` class, containing the results for each number of clusters (one element of the list is associated to one number of clusters),
- `data`: the data used for clustering,
- `criterion`: the model selection criterion used,
- `convergence`: a boolean indicating if none problem of empty class has been encountered (for any number of clusters).

Each element of the list `results` contains all the results for a given number K of classes, which are summarized in the following 18 slots:

- `proportion`: a K-vector of proportions $p_1, \ldots, p_K$,
- `pi`: a $K \times p$-matrix composed of the scale parameters π_k^j $(1 \leq k \leq K$ and $1 \leq j \leq p)$,

[1]The *ranking* representation $x^{-1} = (x_1^{-1}, \ldots, x_m^{-1})$ contains the ranks assigned to the objects, and means that Object $\mathcal{O}_i$ is in the x_i^{-1}th position $(i = 1, \ldots, m)$. Notice that x is associated to the ordering representation.

- `mu`: a matrix with K lines and $m_1 + \ldots + m_p$ columns in which line k is composed of the location parameters $(\mu_k^1, \ldots, \mu_k^p)$ of cluster k,
- `ll, bic, icl`: values of the log-likelihood, BIC criterion and ICL criterion,
- `tik`: a $n \times K$-matrix containing the estimation of the conditional probabilities for the observed ranks to belong to each cluster,
- `partition`: a n-vector containing the partition estimation resulting from the clustering,
- `entropy`: a $n \times 2$-matrix containing for each observation its estimated cluster (column 2, similar to `partition`) and its entropy (column 1), defined as $-\sum_{k=1}^{K} t_{ik} \log(t_{ik})$ where t_{ik} is the conditional probabilities for the ith observation to belong to cluster k, given by `tik`. The `entropy` output illustrates the confidence in the clustering of each observation (a high entropy means a low confidence in the clustering),
- `probability`: a $n \times 2$-matrix similar to the `entropy` output, containing for each observation its estimated cluster (column 2, similar to `partition`) and its probability $p(x_i; \mu_k, \pi_k)$ given its cluster. This probability is estimated using the last simulation of the presentation orders used for the likelihood approximation. The `probability` output exhibits the best representative of each cluster.
- `convergence`: a boolean indicating if none problem of empty class has been encountered,
- `partial`: a boolean indicating the presence of partial rankings,
- `partialRank`: a matrix containing the full rankings, estimated using the within cluster ISR parameters when the ranking is partial. When ranking is full, `partialRank` simply contains the observed ranking. Available only in presence of at least one partial ranking.
- `distanceProp, distancePi, distanceMu`: distances between the final estimation and the current value at each iteration of the SEM-Gibbs algorithm (except the burning phase) for respectively: proportions p_k, scale parameters π_k^j, location parameters μ_k^j. For μ_k^j, the Kendall distance for ranking has been considered (Marden 1995). For p_k and π_k^j, the distance is the mean squared difference. `distanceProp`, `distancePi` and `distanceMu` are lists of `Qsem-Bsem` elements, each element being a $K \times p$-matrix. These elements are reachable by `res[k]@slotname[[iteration]]`.
- `distanceZ`: a vector of size `Qsem-Bsem` containing the rand index (Rand 1971) between the final estimated partition and the current value at each iteration of the SEM-Gibbs algorithm (except the burning phase). Let precise that the rand index is not affected by label switching.
- `distancePartialRank`: Kendall distance between the final estimation of the partial rankings (missing positions in such rankings are estimated) and the current value at each iteration of the SEM-Gibbs algorithm (except the burning phase). `distancePartialRank` is a list of `Qsem-Bsem` elements, each element being a matrix of size $n \times p$. Available only in presence of at least one partial ranking.
- `proportionInitial, piInitial, muInitial, partialRankInitial`: initializations of the parameters in the SEM-Gibbs algorithm (for expert use only).

If `res` is the result of `rankclust()`, each slot of `results` can be reached by `res[k]@slotname`, where `k` is the number of clusters and `slotname` is the name of the slot we want to reach (`proportion`, `pi` ...). For the slots `ll, bic, icl`, `res[''slotname'']` returns a vector of size K containing the values of the slot for each number of clusters.

3.2. Companion functions

In addition to the main function, `rankclust()`, several companion functions are available in **Rankcluster**:

- `convertRank()`: converts ranking representation x^{-1} of a rank to its ordering representation x, and *vice-versa* since $x \circ x^{-1} = x^{-1} \circ x$.
- `criteria()`: estimate the log-likelihood, BIC and ICL criteria from a dataset and a corresponding estimation of the ISR model parameters (see details with `help(criteria)`).
- `distCayley(), distHamming(), distKendall(), distSpearman()`: compute usual distances between rankings (refer to Marden (1995)) for either ranking or ordering representation (refer to the help of the functions for details),
- `frequence()`: transform a raw dataset composed of a list of ranks into a matrix rank/frequency, in which the last column is the frequency of observation of the rank. Conversely, `unfrequence()` transform a rank/frequency dataset in a raw dataset, as requested in input argument of `rankclust()`,
- `khi2()`: perform a chi-squared goodness-of-fit tests and return the p-value of the test (refer to Biernacki and Jacques (2013) for details).
- `kullback()`: estimate the Kullback-Leibler divergence between two ISR models,
- `simulISR()`: simulate a univariate and unimodal dataset of rankings according to the ISR model,

4. Rankcluster through examples

This section illustrates the use of the `rankclust()` function on two real datasets. The first one, *Words*, is a well-known dataset in ranking study, due to Fligner and Verducci (1986), which consists of words associations by students. The second one, *Eurovision*, presented in Jacques and Biernacki (2012), consists of the votes of European countries during the Eurovision Song Contest. Both datasets are available in the **Rankcluster** package, as well as the three other datasets analysed and described in Jacques and Biernacki (2012): `APA, quiz, sports`.

4.1. The Words dataset

These data was collected under the auspices of the Graduate Record Examination Board (Fligner and Verducci 1986). A sample of 98 college students were asked to rank five words according to strength of association (least to most associated) with the target word "Idea": 1 = Thought, 2 = Play, 3 = Theory, 4 = Dream and 5 = Attention.

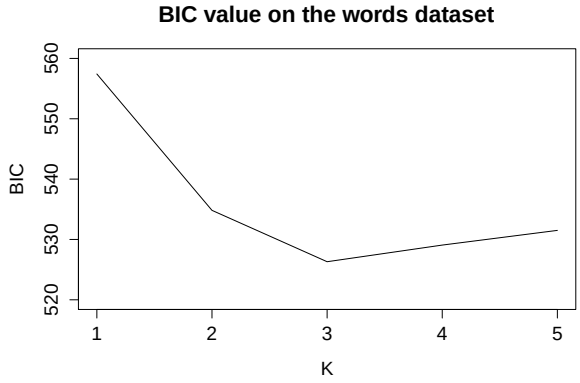


Figure 1: Value of the BIC criterion with mixture of ISR for the Words datatest.

First we start by installing and loading the **Rankcluster** package:

```
R> install.packages("Rankcluster", repos = "http://R-Forge.R-project.org")
R> library(Rankcluster)
```

and then loading the `words` dataset:

```
R> data(words)
```

Using the `rankclust()` function, a clustering with respectively 1 to 5 clusters is estimated:

```
R> res=rankclust(words$data,m=words$m,K=1:5,Qsem=1000,Bsem=100,Ql=1000,Bl=100,
maxTry=30,run=20)
```

The number of SEM-Gibbs iterations (`Qsem`) has been set to 1000, with a burning phase of 100 iterations (`Bsem`). The same values are considered for likelihood approximation (`Ql` and `Bl`). Option `maxTry=30` allows to restart the estimation algorithm in the limit of 30 times if one cluster becomes empty (frequent for $K = 5$). Finally, the SEM-Gibbs algorithm is initialized 20 times (`run=20`), and the best solution (according to the approximated likelihood) is retained. Computing time on a laptop with 2.80GHz CPU is about 7 seconds per run et per cluster (the number of runs can be reduced if the reader want to test this code more quickly).

The values of the BIC criterion, reached by `res[‘‘bic’’]` and plotted on Figure 1, tend to select three clusters.

The parameter estimation for $K = 3$ are given below for proportions p_k, probabilities π_k and modes μ_k:

```
> res[3]@proportion
[1] 0.3061224 0.4918367 0.2020408
> res[3]@pi
         dim 1
cl 1 0.9060649
cl 2 0.9416822
cl 3 0.8642753
> res[3]@mu
     dim 1
cl 1     2 5 3 4 1
```

```
cl 2     2 5 4 3 1
cl 3     5 2 4 3 1
```

The words Thought is the most associated with Idea for all clusters. Regarding the rankings of the four other words can suggest an interesting interpretation of the clusters. Indeed, the first cluster, composed of about 30% of the students, is characterized by the following modal ranking: Play, Attention, Theory, Dream, Thought. Students of this cluster are probably literary-minded students, rankings the word Dream just after Thought. Students of the second cluster (about half of total students) are probably more scientific since they rank the word Theory just after Thought, and so before the word Dream: Play, Attention, Dream, Theory, Thought. This cluster is also the most homogeneous, with a high scale parameter value (low dispersion): $\pi_2 \simeq 0.94$. Finally, the last cluster is characterized by the following mode: Attention, Play, Dream, Theory, Thought. The only difference in the modal ranking with the scientific students is the preference of Play rather than Attention. This cluster, which is the smallest (20% of the students), can be qualified as intermediary cluster, probably composed of a set of students not too scientific or too literary-minded, as evidenced by the smallest of the three scale parameter values ($\pi_3 \simeq 0.86$).

4.2. The Eurovision dataset

The Eurovision Song Contest is an annual competition held among active member countries of the European Broadcasting Union. Each member country submits a song to be performed on live television and then casts votes for the other countries' songs to determine the most popular song in the competition. The vote consists in ranking ten preferred song in order of preference. We consider in these experiments the votes of the $n = 34$ countries who participate to the competitions from 2007 to 2012. During these six years, only 8 countries have participated to the six finals of the competition: 1: France, 2: Germany, 3: Greece, 4: Romania, 5: Russia, 6: Spain, 7: Ukraine and 8: United Kingdom. The studied dataset is then composed of multivariate rankings ($p = 6$ corresponding to the six contests between 2007 and 2012), each rank being of size $m = 8$ (only the votes for the 8 countries which participated to the six finals are considered) and all rankings being partial. The absence of full ranking signifies that none country participating to the votes has ranked all of the 8 previously cited countries in its 10 preferences.

The `eurovision` dataset is loaded by:

```
R> data(eurovision)
```

This dataset is challenging since the number of observations ($n = 34$) is small compared to the size of the ranks ($m = 8$ and $p = 6$) and the presence of partial rankings (precisely, 57.7% of the rankings elements are missing). For this reason, large iterations numbers (`Qsem=Ql=1000`) have been chosen to estimate a clustering with respectively 1 to 9 clusters:

```
R> res=rankclust(eurovision$data,m=eurovision$m,K=1:9,Qsem=1000,Bsem=100,
Ql=1000,Bl=100,maxTry=30,run=20) #caution:  can be time consuming (see below)
```

With these large iterations numbers, `rankclust()` takes about 4 hours per number of clusters and per run (laptop 2.80GHz CPU). Most of this computing time is due to the likelihood approximation (detail of the computing time can be obtained by setting the input option `detail` to `TRUE`). The reason is the high proportion of missing elements, which leads to a

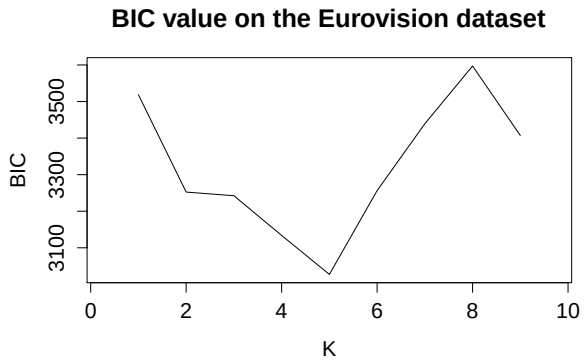


Figure 2: Value of the BIC criterion with mixture of ISR for the Eurovision datatest.

large number of different multivariate modal rankings $\mu_k^1, \ldots, \mu_k^p$ simulated during the SEM-Gibbs algorithm and then to a large number of likelihood approximations (let recall that the retained parameters at the end of the estimation algorithm are those leading to the highest approximated likelihood). Nevertheless, the total computing time can be easily reduced by parallelizing the `rankclust()` executions for each run and each number of clusters, thanks to the **doMC** package for R. For instance, the following code launch parallel executions of `rankclust()` for each number of clusters (`nbcore` being the number of available clusters):

```
R> library('doMC')
R> registerDoMC(nbcore)
R> ResFinal = foreach(k=1:9,.combine='c') %dopar% {
R> res=rankclust(eurovision$data,m=eurovision$m,K=k,Qsem=1000,Bsem=100,Ql=1000
,Bl=100,maxTry=30,run=30)
R> }
```

In the present work, large iterations numbers have intentionally be used in order to provide results with reduced variability, which can be interpreted with confidence. But the reader who want to test the package on this dataset can use less iteration numbers (dividing for instance by ten the iterations numbers leads to 1 minute of computing time per cluster and per run). In this case, the variability of parameter estimation will be larger, about two times larger for 100 SEM-Gibbs iterations rather than 1000 according to the variability indicator described at the end of this section (standard deviation of the distances between the final parameter estimation and the current value at each step of the SEM-Gibbs algorithm).

The values of the BIC criterion (obtained with `Qsem=Ql=1000`) are plotted on Figure 2. They tend to select five groups.

The printed outputs for $K = 5$ are given below: value of the log-likelihood (LL), values of BIC and ICL criteria, estimation of the proportions p_k's, the probabilities π_k^j's, the modes μ_k^j's, the estimated partition and finally the conditional probability of the observations to belong to each cluster (t_{ik}).

```
R> rankclust(eurovision$data,m=eurovision$m,K=5,Qsem=1000,Bsem=100,Ql=1000,
Bl=100,maxTry=30,run=30)
```

```
****************************************************************
Number of clusters: 5
****************************************************************
ll= -1400.816
bic = 3027.319
icl = 3027.321
proportion: 0.3235294 0.1470588 0.2352941 0.1176471 0.1764706
pi:
      dim 1      dim 2      dim 3      dim 4      dim 5      dim 6
cl 1  0.8373494  0.9195402  0.8596491  0.8588235  0.7891566  0.8400000
cl 2  0.9090909  0.9493671  0.8064516  0.8493151  0.7916667  0.8428571
cl 3  0.8771930  0.8728814  0.8800000  0.7920792  0.7983871  0.8914729
cl 4  0.9740260  0.8095238  0.9634146  0.8955224  0.8181818  0.8545455
cl 5  0.8543689  0.9047619  0.8333333  0.9263158  0.8200000  0.8854167
mu:
      dim 1            dim 2            dim 3            dim 4
cl 1  3 5 7 4 2 6 1 8  3 5 7 6 4 2 8 1  3 8 1 4 5 2 6 7  3 2 6 4 1 8 5 7
cl 2  5 7 3 2 4 1 6 8  5 7 3 1 4 2 6 8  8 4 5 1 7 3 6 2  4 2 5 8 7 6 1 3
cl 3  7 3 5 2 4 6 1 8  3 5 7 1 6 4 8 2  1 8 2 3 5 6 7 4  3 4 2 1 6 8 7 5
cl 4  7 5 4 6 3 1 2 8  5 7 3 1 4 8 6 2  5 1 8 6 7 2 4 3  2 5 7 6 4 1 3 8
cl 5  7 5 8 4 2 3 6 1  7 5 3 4 8 1 2 6  8 1 4 3 2 5 7 6  2 4 1 8 3 6 7 5
      dim 5            dim 6
cl 1  7 3 8 5 1 4 2 6  3 5 6 4 1 2 7 8
cl 2  7 5 8 6 4 3 1 2  4 5 7 2 1 3 8 6
cl 3  6 2 8 1 3 4 5 7  6 5 2 4 1 7 3 8
cl 4  5 8 3 1 2 7 4 6  5 2 3 7 1 8 6 4
cl 5  2 8 7 1 3 4 5 6  5 2 4 7 1 3 8 6
partition:
 1 2 1 3 1 5 1 5 2 2 3 3 1 2 3 5 4 4 4 5 2 5 3 1 1 1 3 5 1 1 3 1 4 3
tik:
                1          2          3  4          5
 1  1.000000e+00  4.926687e-15  3.395961e-11  2.175299e-20  2.172081e-14
 2  1.020925e-12  1.000000e+0 0 3.596271e-15  9.629061e-14  1.176418e-16
 3  9.999993e-01  1.631053e-1 3 7.254845e-07  8.621185e-17  6.955543e-13
 4  1.120337e-09  5.431846e-1 6 1.000000e+00  1.290113e-17  6.791088e-14
 5  1.000000e+00  5.245435e-1 8 9.260641e-13  2.667931e-21  8.113361e-16
 6  5.738516e-16  3.275921e-16  2.041697e-15  8.810364e-27  1.000000e+00
 7  1.000000e+00  2.177675e-19  7.848022e-13  6.013325e-29  5.084076e-18
 8  4.028680e-13  2.044741e-16  4.920668e-10  2.912902e-22  1.000000e+00
 9  1.547418e-09  1.000000e+00  2.268076e-11  1.711549e-10  4.627540e-12
10  2.535008e-12  1.000000e+00  1.524028e-17  5.826940e-17  1.425966e-14
11  3.249961e-09  7.008065e-12  1.000000e+00  2.493183e-17  2.333494e-10
12  3.438573e-12  2.480877e-20  1.000000e+00  6.300352e-18  8.826780e-15
13  1.000000e+00  1.241974e-15  6.370072e-09  5.750960e-25  1.884747e-17
14  6.535477e-15  1.000000e+00  5.803484e-13  1.043131e-12  9.389025e-15
15  2.237704e-09  2.923127e-13  1.000000e+00  1.137870e-12  4.878962e-13
16  1.303839e-16  8.068278e-12  3.667421e-14  1.131581e-18  1.000000e+00
```

```
17  1.142291e-15  7.192854e-11  2.523839e-13  1.000000e+00  1.012907e-19
18  9.853473e-23  9.429721e-17  8.455583e-18  1.000000e+00  1.746191e-21
19  1.320047e-15  2.978251e-16  1.266509e-14  1.000000e+00  7.754970e-11
20  8.550912e-12  2.466856e-13  4.301909e-10  5.708839e-23  1.000000e+00
21  5.288867e-13  1.000000e+00  1.220869e-14  1.212887e-15  1.316850e-14
22  7.037856e-14  1.914019e-18  2.234911e-14  4.996731e-23  1.000000e+00
23  3.835737e-03  1.161211e-03  9.949931e-01  5.409898e-11  9.921546e-06
24  1.000000e+00  3.673471e-22  3.072776e-11  1.504338e-24  1.937220e-16
25  1.000000e+00  2.579782e-13  2.734699e-10  2.205859e-15  2.174011e-09
26  1.000000e+00  4.005485e-17  2.967356e-13  3.898933e-20  7.700084e-15
27  1.807390e-02  3.457956e-07  5.457201e-01  9.152708e-10  4.362056e-01
28  4.262774e-15  3.084315e-13  2.330521e-13  1.932227e-20  1.000000e+00
29  9.999994e-01  1.060360e-09  5.685554e-07  2.121323e-16  5.110376e-08
30  8.980571e-01  2.670018e-11  1.019404e-01  1.149813e-15  2.475373e-06
31  1.112656e-13  2.914716e-19  1.000000e+00  2.719565e-17  6.941053e-17
32  1.000000e+00  1.530419e-11  1.757512e-12  3.221608e-13  3.433776e-11
33  5.265654e-15  1.111768e-14  1.023056e-17  1.000000e+00  2.158626e-20
34  2.900612e-14  1.798547e-20  1.000000e+00  1.473098e-26  6.098085e-16
**************************************************************
```

In addition to such information, the `summary()` function

```
R> summary(res)
```

gives an overview of the partition by printing the five ranks of highest probability (output `res[5]@probability`) and the five ranks of highest entropy (output `res[5]@entropy`) for each cluster. The ranks of highest probability are the best representatives of the cluster, whereas the ranks of highest entropy are those for which their belonging to the cluster are the less obvious. Notice that the full list of the cluster member with their probability and entropy are available through the slots `probability` and `entropy`. Table 1 gives an example of these outputs for cluster 1, which is composed of: Albania, Belgium, Bulgaria, Cyprus, Germany, Romania, Russia, Serbia, Sweden, Switzerland and Turkey. Cyprus is the best representative of the cluster, whereas the belonging of Switzerland to this cluster is relatively questionable (high entropy in comparison to others).

country	entropy	country	probability
Switzerland	6.587196e-01	Cyprus	8.290355e-14
Belgium	2.196254e-05	Germany	4.805708e-14
Sweden	1.935311e-05	Turkey	4.594954e-15
Germany	2.531678e-07	Belgium	2.308477e-15
Russia	1.036828e-07	Romania	6.515448e-16

Table 1: Countries with the highest probabilities and entropy in the first cluster.

The `summary()` function prints also the estimated full ranking for each partial ranking. For instance, Table 2 gives the real votes of France to the Eurovision contest between 2007 and 2012 (top line of the table) and the estimated ones (bottom line). In the top line (real votes), a "0" in the ranking means that the country ranked by France at this position does not belong to the eight studied countries (1: France, 2: Germany, 3: Greece, 4: Romania, 5: Russia, 6:

Spain, 7: Ukraine and 8: United Kingdom). This example suggests an interesting comment. Of course, countries can not vote for themselves. However, if France could vote for itself, we show in the estimated full rankings that France will often rank itself in a good position. Indeed, France appears almost always in the second place of the missing positions.

	2007	2008	2009
observed ranking	4 6 7 3 5 0 0 0	6 4 3 5 0 0 0 0	8 2 0 0 0 0 0 0
estimated ranking	4 6 7 3 5 2 1 8	6 4 3 5 7 1 8 2	8 2 3 1 5 6 7 4
	2010	2011	2012
observed ranking	3 2 7 0 0 0 0 0	6 2 8 0 0 0 0 0	2 6 5 4 0 0 0 0
estimated ranking	3 2 7 4 1 6 8 5	6 2 8 4 5 1 3 7	2 6 5 4 7 1 3 8

Table 2: Votes of France to the Eurovision contest between 2007 and 2012: the top line is the real ranking with missing position (0) and the bottom line is the estimated full ranking. (1: France, 2: Germany, 3: Greece, 4: Romania, 5: Russia, 6: Spain, 7: Ukraine and 8: United Kingdom)

A geographical representation of the estimated clustering is given by Figure 3. As noticed in Jacques and Biernacki (2012), this clustering can suggest some geographical interpretation of the clustering: indeed, the countries of a same groups seem to be geographically close. For instance, the cluster 1 (black) is essentially composed of East Europe countries whereas cluster 3 (green) mainly concerns West European countries. The geographical proximity of clusters members can be due either to cultural proximity of these countries or to geographical alliances, often suspected for this contest.

Finally, the variability of estimation of the model parameters can be achieved by the mean of the distances between the final estimation and the current value at each step of the SEM-Gibbs algorithm. These distances are available in the slots `distanceProp, distancePi, distanceMu` of the output `res[5]`. The standard deviation of these distances can be used for instance as an indicator of estimation variability. Let note also that a plot of these distances (Figure 4) can be used to empirically judge if the burning phase size of the SEM-Gibbs algorithm is sufficiently large. Indeed, we remark on Figure 4 that the variability of estimation is larger for the first 50 iterations than for the following ones (we recall that the first 100 iterations corresponding to the burning phase are not plotted on Figure 4). A burning phase of 150 iterations rather than 100 iterations would probably be a wise choice.

Similarly, the slot `distancePartition` illustrates the convergence of the SEM-Gibbs algorithm by given the rand index between the final partition and the current partition at each SEM-Gibbs iteration (Figure 5). Notice that the partition is relatively stable at the end of the algorithm iterations, although the cluster memberships of the observations are simulated at each iteration of the SEM-Gibbs algorithm. This phenomenon is due to the data space which is high-dimensional because of the complexity of the dataset (high proportion of missing ranking elements, low number of observations ...). Thus, even if the location and scale parameters move during the SEM-Gibbs iterations (Figure 4), the clusters stay well separated and the partition is stable.

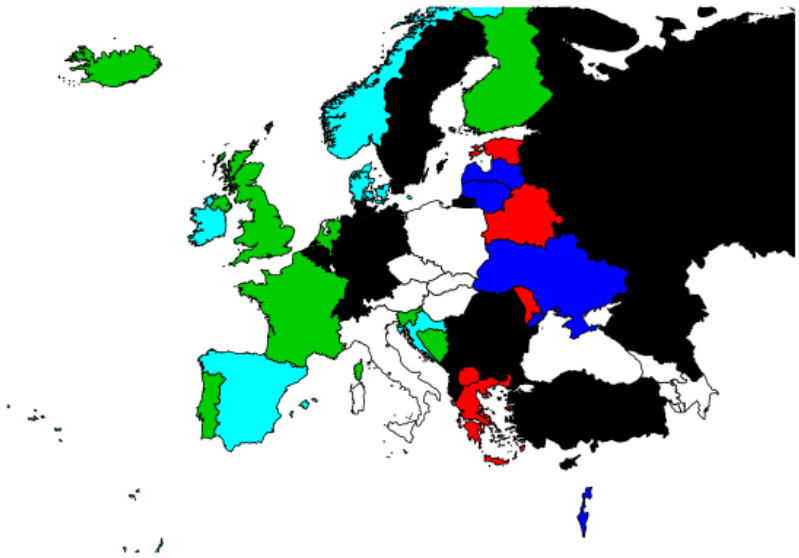


Figure 3: Clustering of the European countries according to their votes at the Eurovision contest between 2007 and 2012.

5. Conclusion

Rankcluster is the first R package dedicated to ranking data, allowing modelling and cluster analysis for multivariate partial ranking data. Available on R-forge, this package is simple of use with its main function, `rankclust()`, having only one mandatory argument, the ranking dataset. By default a modelling is performed, and mentioning the number of desired clusters leads to perform a cluster analysis, with selection of the number of clusters if several numbers are given.

References

Biernacki C, Celeux G, Govaert G (2000). "Assessing a mixture model for clustering with the integrated completed likelihood." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(4), 719–725.

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

Celeux G, Diebolt J (1985). "The SEM algorithm: A probabilistic teacher algorithm derived

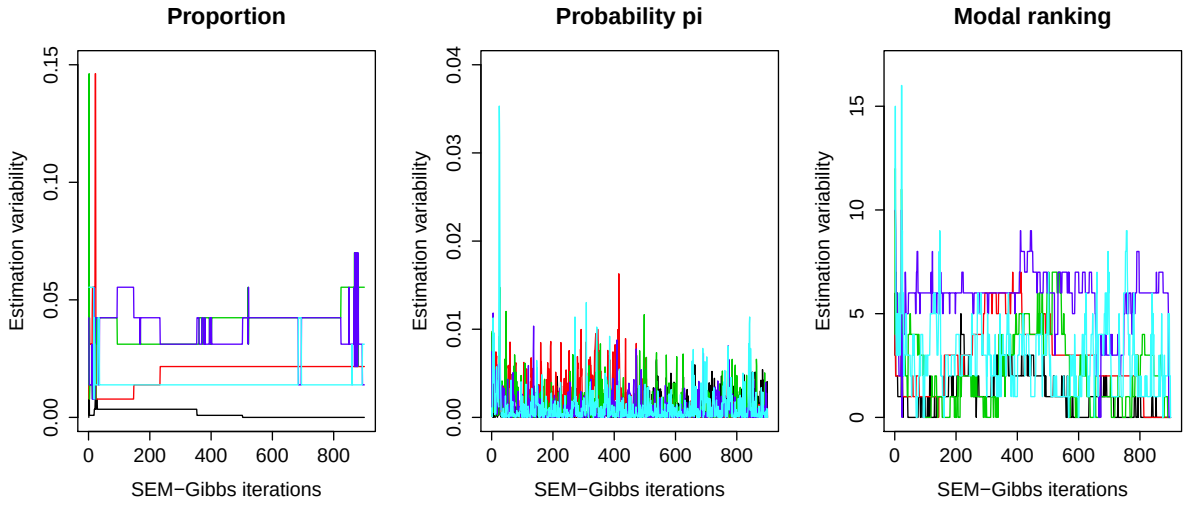


Figure 4: Variability of the parameters estimation (proportion and scale and location parameters for the first dimension) along with the SEM-Gibbs iterations (without the burning phase): cluster 1 in black, 2 in red, 3 in green, 4 in blue, 5 in cyan.

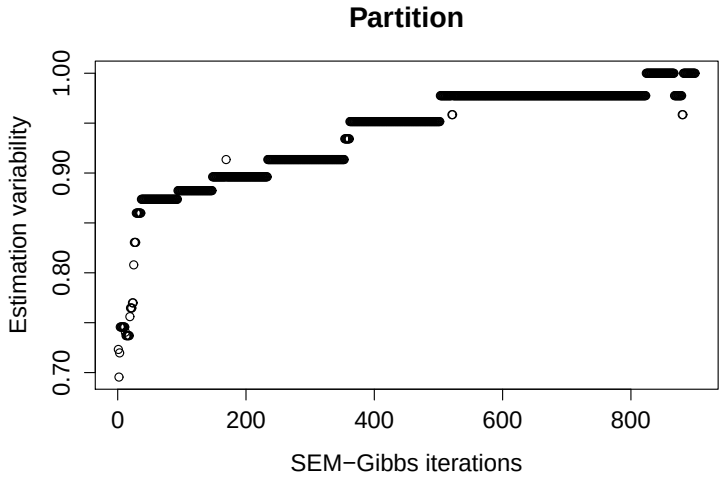


Figure 5: Evolution of the partition along with the SEM-Gibbs iterations: values of the rand index between current and final partitions.

from the EM algorithm for the mixture problem." *Computational Statistics Quarterly*, **2**(1), 73–82.

Dempster A, Laird N, Rubin D (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society. Series B. Methodological*, **39**(1), 1–38. With discussion.

Fligner M, Verducci J (1986). "Distance based ranking models." *Journal of the Royal Statistical Society. Series B. Methodological*, **48**(3), 359–369.

Geman A, Geman D (1984). "Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images." *IEEE Transactions on Pattern Analysis and Matching Intelligence*, **6**, 721–741.

Gormley I, Murphy T (2008). "A mixture of experts model for rank data with applications in election studies." *Annals of Applied Statistics*, **2**(4), 1452–1477.

Jacques J, Biernacki C (2012). "Model-based clustering for multivariate partial ranking data." *Technical Report 8113*, Inria Research Report.

Marden J (1995). *Analyzing and modeling rank data*, volume 64 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London.

Rand W (1971). "Objective criteria for the evaluation of clustering methods." *Journal of the American Statistical Association*, **55**, 846–850.

Schwarz G (1978). "Estimating the dimension of a model." *The Annals of Statistics*, **6**(2), 461–464.

Affiliation:

Julien Jacques
Laboratoire Paul Painlevé – Université Lille 1 & CNRS
Modal Team – Inria Lille - Nord Europe
59655 Villeneuve d'Ascq Cedex – France
E-mail: julien.jacques@polytech-lille.fr

Quentin Grimonprez
Modal Team – Inria Lille - Nord Europe
40 avenue Halley, 59650 Villeneuve d'Ascq – France
E-mail: quentin.grimonprez@inria.fr

Christophe Biernacki
Laboratoire Paul Painlevé – Université Lille 1 & CNRS
Modal Team – Inria Lille - Nord Europe
59655 Villeneuve d'Ascq Cedex – France
E-mail: christophe.biernacki@inria.fr

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd