ordinalClust: a package for analyzing ordinal data

Margot Selosse, Julien Jacques, Christophe Biernacki

ordinalClust: a package for analyzing ordinal data

Margot Selosse, Université de Lyon, Lyon 2, ERIC EA 3083.
Julien Jacques, Université de Lyon, Lyon 2, ERIC EA 3083.
Christophe Biernacki, Inria, Université de Lille, CNRS.

January 9, 2018

Abstract

Ordinal data are used in a lot of domains, especially when measurements are collected from persons by observations, testings, or questionnaires. ordinalClust is an R package dedicated to ordinal data that proposes tools for modeling, clustering, co-clustering and classification. Ordinal data are modeled by the BOS distribution, which is a meaningful model parametrized by a position and a precision parameter. On one hand, the co-clustering framework uses the Latent Block Model (LBM) and an SEM-Gibbs algorithm for the parameters inference. On the other hand, the clustering and the classification methods follow on from simplified versions of this algorithm. An overview of these methods is given, and the way of using them with the ordinalClust package is described through real datasets.

1 Introduction

Ordinal data is a specific kind of categorical data occurring when the levels are ordered [1]. Some common contexts for the collection of ordinal data include satisfaction survey, aptitude and personality testing or psychological questionnaires. In the present work, an ordinal variable is called x and it is considered to have m levels that are written $(1, ..., m)$.

So far, ordinal data have received more attention from a supervised point of view. For example: a marketing firm targets to investigate which factors influence the size of soda (small, medium, large or extra large) that people order at a fast-food chain. These factors may include which type of sandwich is ordered (burger or chicken), whether or not fries are also ordered, and age of the consumer. In this case, an observation consists in factors of different types and the variable to predict is of the ordinal kind. Several software propose to analyze ordinal data in a regression framework. ordinal [5] implements the cumulative linked model which assumes that:

$$\text{logit}(p(x \leq \mu)) = \log \frac{p(x_i \leq \mu)}{1-p(x_i \leq \mu)} = \beta_0(\mu) + \boldsymbol{\beta}^t \boldsymbol{t},$$

where x is the ordinal variable, μ one of its levels, $\boldsymbol{t}$ the covariates, and $\beta_0(\mu)$ increases with μ. In the absence of covariates, it is equivalent to a multinomial model. The Latent Gold Software [14] uses this kind of model in a clustering context. Other approaches consider the probability distribution of ordinal data as a discretization of an underlying continuous variable [11]. Nevertheless, the implementation of these methods is known to be computationally expensive. In addition, it is not provided through user-friendly R package. Furthermore, while most of these techniques focus on predicting an ordinal variable with factors of different types, the aim of the present package ordinalClust is to analyze datasets made of ordinal data, and potentially to predict a variable of the categorical kind.

Recently, [4] proposed the so-called BOS (Binary Ordinal Search) model. It is a probability distribution specific to ordinal data which is parametrized with meaningful parameters (μ, π), respectively linked to a position and precision role. The latter work also described how the BOS distribution can be used to perform clustering on multivariate ordinal data. Then, [9] employed this distribution coupled to the Latent Block Model [7] in order to carry out a co-clustering on ordinal data. Furthermore, the authors showed that the used algorithm can easily deal with missing values. However, this model could not take into account ordinal data with different number of levels. [13] used an extension of the Latent Block Model to overcome this issue. In the present work, a new classification technique that ensues directly from these methods is proposed.

The latter mentioned works have proved their proficiency and also provide efficient techniques to perform clustering and co-clustering of ordinal data. The purpose of the package ordinalClust is to offer a complete tool for analyzing ordinal data by implementing these methods, and by adding a classification framework. The present document gives an overview of these notions and illustrates the usage of ordinalClust through concrete examples.

2 Statistical methods

2.1 Data

A dataset of ordinal data will be written $\boldsymbol{x} = (x_{ij})_{i,j}$, with $1 \leq i \leq N$ and $1 \leq j \leq J$, N and J denoting respectively the number of individuals and the number of variables. Furthermore, a dataset can contain missing data. While dealing with this aspect, the dataset will be expressed by $\boldsymbol{x} = (\check{\boldsymbol{x}}, \hat{\boldsymbol{x}})$, $\check{\boldsymbol{x}}$ being the observed data, and $\hat{\boldsymbol{x}}$ being the missing data. Consequently an element of $\boldsymbol{x}$ will be annotated as follows: $\check{x}_{ij}$, whether x_{ij} is observed, $\hat{x}_{ij}$ otherwise.

2.2 The BOS model

The BOS model [4] is a probability distribution for ordinal data parametrized by a position parameter $\mu \in \{1, ..., m\}$ and a precision parameter $\pi \in [0, 1]$. It was built by assuming that an ordinal variable is the result of a stochastic binary search algorithm within the ordered table $(1, ..., m)$. This distribution rises from the uniform distribution when $\pi = 0$ to a more peaked distribution around the mode μ when π grows, and reaches a Dirac distribution at the mode μ when $\pi = 1$. Figure 1 illustrates the shape of the BOS distribution with different values of μ and π. It is shown in [4] that the BOS distribution is a polynomial function of π with degree $m - 1$ whose coefficients depend on the position parameter μ. For a univariate ordinal variable, the path in the stochastic binary search can be seen as a latent variable. Therefore, an efficient way to perform the maximum likelihood estimation is through an EM algorithm [6].

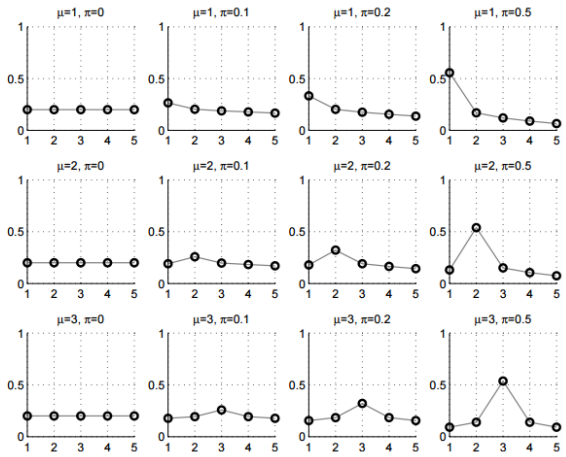

Figure 1: BOS distribution $p(x; \mu, \pi)$: shape for $m = 5$ and for different values of μ and π.

2.3 The co-clustering model

Notations Being in a co-clustering context, it is assumed that there exists G row-clusters and H column-clusters inherent to the $\boldsymbol{x}$ matrix. It is therefore useful to introduce g (resp. h) which represents the g^{th} (resp. h^{th}) row-cluster (resp. column-cluster), with $1 \leq g \leq G$ (resp. $1 \leq h \leq H$). In addition, the sums and the products relating to rows, columns, row-clusters and column-clusters will be subscripted respectively by the letters i, j, g, and h. So the sums and products will be written $\sum_i$, $\sum_j$, $\sum_g$ and $\sum_h$ and $\prod_i$, $\prod_j$, $\prod_g$ and $\prod_h$.

Latent Block Model Let consider the data matrix $\boldsymbol{x} = (x_{ij})_{i,j}$. It is assumed that there exists G row-clusters and H column-clusters that correspond to a partition $\boldsymbol{v} = (v_{ig})_{i,g}$ and a partition $\boldsymbol{w} = (w_{jh})_{j,h}$, with $1 \leq g \leq G$ and $1 \leq h \leq H$. We have noted $v_{ig} = 1$ if i belongs to cluster g, whereas $v_{ig} = 0$ otherwise, and $w_{jh} = 1$ when j belongs to cluster h, but $w_{jh} = 0$ otherwise. Each element x_{ij} is considered to be generated under a parameterized probability density function $p(x_{ij}; \alpha_{gh})$. Here, g denotes the cluster of row i, and h denotes the cluster of column j, while α_{gh} represents the parameters of probability density function of block (g, h), a block being the crossing of both a row-cluster and a column-cluster. Figure 2 is an example of co-clustering performed on an ordinal data matrix.

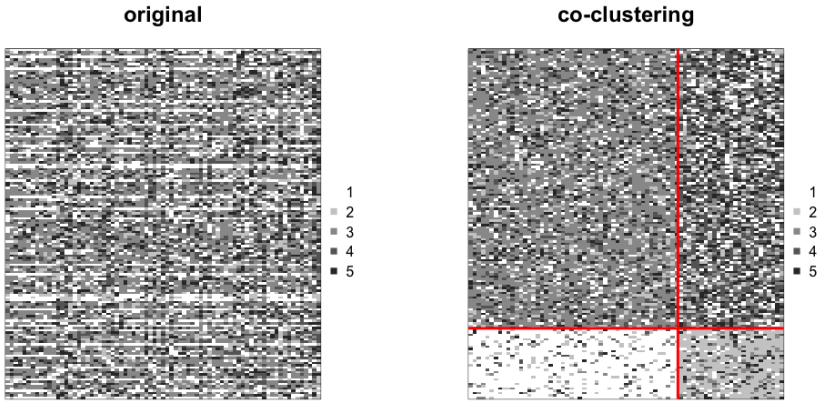

Figure 2: On the left: original dataset made of ordinal data with $m = 5$. On the right: a co-clustering is performed with $G = H = 2$, the rows and columns are sorted by row-clusters and column clusters, which emphasizes a structure in the dataset.

The univariate random variables x_{ij} are assumed to be conditionally independent given the row and column partitions $\boldsymbol{v}$ and $\boldsymbol{w}$. Therefore, the conditional probability density function of $\boldsymbol{x}$ given $\boldsymbol{v}$ and $\boldsymbol{w}$ can be written:

$$p(\boldsymbol{x}|\boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\alpha}) = \prod_{i,j,g,h} p(x_{ij}; \alpha_{gh})^{v_{ig} w_{jh}},$$

where $\boldsymbol{\alpha} = (\alpha_{gh})_{g,h}$ being the distribution's parameters of block (g, h). Any univariate distribution can be used in regard to the kind of data (e.g: Gaussian, Bernoulli, Poisson...). In the ordinalClust package, the BOS distribution is employed thus $\alpha_{gh} = (\mu_{gh}, \pi_{gh})$. For convenience, the label of row i is also denoted by $\boldsymbol{v}_i = (v_{i1}, ..., v_{iG}) \in \{0, 1\}^G$. Similarly, the label of column j is denoted by $\boldsymbol{w}_j = (w_{j1}, ..., w_{iH}) \in \{0, 1\}^H$. These latent variables $\boldsymbol{v}$ and $\boldsymbol{w}$ are assumed to be independent so $p(\boldsymbol{v}, \boldsymbol{w}; \boldsymbol{\gamma}, \boldsymbol{\rho}) = p(\boldsymbol{v}; \boldsymbol{\gamma}) p(\boldsymbol{w}; \boldsymbol{\rho})$ with:

$$p(\boldsymbol{v}; \boldsymbol{\gamma}) = \prod_{i,g} \gamma_g^{v_{ig}} \quad \text{and} \quad p(\boldsymbol{w}; \boldsymbol{\rho}) = \prod_{j,h} \rho_h^{w_{jh}},$$

knowing that $\gamma_g = p(v_{ig} = 1)$ with $g \in \{1, ..., G\}$ and $\rho_h = p(w_{jh} = 1)$ with $h \in \{1, ..., H\}$. This implies that, for all i, the distribution of $\boldsymbol{v}_i$ is the multinomial distribution $\mathcal{M}(\gamma_1, ..., \gamma_G)$ and does not depend on i. In a similar way, for all j, the distribution of $\boldsymbol{w}_j$ is the multinomial distribution $\mathcal{M}(\rho_1, ..., \rho_H)$ and does not depend on j. From these considerations, the parameter of the latent block model is defined as $\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{\alpha})$, with $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_G)$ and $\boldsymbol{\rho} = (\rho_1, ..., \rho_H)$ the rows and columns mixing proportions. Therefore, if V and W are the sets of all possible labels $\boldsymbol{v}$ and $\boldsymbol{w}$, the probability density function $p(\boldsymbol{x}; \boldsymbol{\theta})$ of $\boldsymbol{x}$ can be written:

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{(\boldsymbol{v}, \boldsymbol{w}) \in V \times W} \prod_{i,g} \gamma_g^{v_{ig}} \prod_{j,h} \rho_h^{w_{jh}} \prod_{i,j,g,h} p(x_{ij}; \alpha_{gh})^{v_{ig} w_{jh}}. \tag{1}$$

Model Inference In the co-clustering context, the inference aim is to maximize the observed log-likelihood $l(\boldsymbol{\theta}; \tilde{\boldsymbol{x}}) = \sum_{\tilde{\boldsymbol{x}}} p(\boldsymbol{x}; \boldsymbol{\theta})$. The EM-algorithm [6] is a very well known technique for maximizing parameters with latent variables. However, regarding the co-clustering case, it is not

computationally tractable. Indeed, this method needs to compute the expectation of the complete data log-likelihood. Though, this expression contains the probability $p(v_{ig} = 1, w_{jh} = 1|\boldsymbol{x}, \boldsymbol{\theta})$, which needs to consider all the possible values for $\boldsymbol{v}_{i'}$ and $\boldsymbol{w}_{j'}$ with $i' \neq i$ and $j' \neq j$. The E-step would require to calculate $G^N \times H^J$, With the values of the example below ($G = 4$, $H = 4$, $N = 121$ and $J = 28$) which would result in the computation of $4^{121} \times 4^{28} \approx 5 \times 10^{89}$ terms. There exists different alternatives to the EM algorithm as the variational EM algorithm, the SEM-Gibbs algorithm or other algorithm linked to a Bayesian inference. The SEM-Gibbs version is used because it is known to avoid spurious solutions [10]. Furthermore, it handles easily missing values $\hat{\boldsymbol{x}}$ in $\boldsymbol{x}$, which is an important advantage, particularly with real datasets. The SEM-algorithm is made of two iteratively repeated steps that are detailed in Algorithm 1.

Values initialization: $\boldsymbol{v}^{(0)}, \boldsymbol{w}^{(0)}, \boldsymbol{\theta}^{(0)}, \hat{\boldsymbol{x}}^{(0)}$
for (q in 1:(nbSEM)){
1. SE-step.
1.1 Generate the row partition with $v_{ig}^{(q)}|\check{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{(q-1)}, \boldsymbol{w}^{(q-1)}$ for all $1 \leq i \leq N$, $1 \leq g \leq G$:
$p(v_{ig} = 1|\boldsymbol{x}^{(q-1)}, \boldsymbol{w}^{(q-1)}; \boldsymbol{\theta}^{(q-1)}) \propto \gamma_g^{(q-1)} \prod_{jh} p(x_{ij}; \mu_{gh}^{(q-1)}, \pi_{gh}^{(q-1)})^{w_{jh}^{(q-1)}}$.
1.2 Generate column partition with $w_{jh}^{(q)}|\check{\boldsymbol{x}}, \hat{\boldsymbol{x}}^{(q-1)}, \boldsymbol{v}^{(q)}$ for all $1 \leq j \leq J$, $1 \leq h \leq H$:
$p(w_{jh} = 1|\boldsymbol{x}, \boldsymbol{v}^{(q)}; \boldsymbol{\theta}^{(q-1)}) \propto \rho_h^{(q-1)} \prod_{ig} p(x_{ij}; \mu_{gh}^{(q-1)}, \pi_{gh}^{(q-1)})^{v_{ig}^{(q)}}$.
1.3 Generate the missing data $\hat{x}_{ij}^{(q)}|\check{\boldsymbol{x}}, \boldsymbol{v}^{(q)}, \boldsymbol{w}^{(q)}$ as follows:
$p(\hat{x}_{ij}^{(q)}|\check{\boldsymbol{x}}, \boldsymbol{v}^{(q)}, \boldsymbol{w}^{(q)}; \boldsymbol{\theta}^{(q-1)}) = \prod_{g,h} p(\hat{x}_{ij}; {\mu_{gh}}^{(q-1)}, {\pi_{gh}}^{(q-1)})^{v_{ig}^{(q)} {w_{gh}}^{(q)}}$.
2. M-step.
Maximization of the completed log-likelihood by updating the co-clusters BOS parameters with an EM-algorithm (see [4] for further details).
}

Algorithm 1: SEM-Gibbs for co-clustering on ordinal data.

Initialization The ordinalClust package allows two modes for values initialization, through the argument **init** which can take values `"random"` or `"kmeans"`. The first one randomly initializes $\boldsymbol{v}^{(0)}$ and $\boldsymbol{w}^{(0)}$. The second one is the by default value and consists in performing a Kmeans algorithm [8] on the rows and on the columns. In both cases, $\boldsymbol{\theta}^{(0)}$ is estimated according to $\boldsymbol{v}^{(0)}$ and $\boldsymbol{w}^{(0)}$. Then, $\hat{\boldsymbol{x}}^{(0)}$ is sampled according to $\boldsymbol{v}^{(0)}$, $\boldsymbol{w}^{(0)}$ and $\boldsymbol{\theta}^{(0)}$.

Estimation of model parameters and partitions The first iterations of the SEM-Gibbs are called the burn-in period, which means the parameters are not stable yet. Consequently, only the iterations that occurred after this burn-in period are taken into account and are referred to as sampling distribution hereafter. While the final estimation of the position parameter $\hat{\mu}_{gh}$ is the mode of the sampling distribution, the final estimations of the continuous parameters $(\hat{\pi}_{gh}, \hat{\gamma}_g, \hat{\boldsymbol{\rho}}_h)$ are the median of the sample distribution. It leads to a final estimation of $\boldsymbol{\theta}$ that is called $\hat{\boldsymbol{\theta}}$. Then, a sample of $(\hat{\boldsymbol{x}}, \boldsymbol{v}, \boldsymbol{w})$ is generated by several SE-step (step **1.** from Algorithm 1) with $\boldsymbol{\theta}$ fixed to $\hat{\boldsymbol{\theta}}$. The final partitions $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{w}})$ and the missing observations $\hat{\boldsymbol{x}}$ are estimated by the mode of their sample distribution.

Model Selection To determine how many row-clusters and how many column-clusters are necessary, an adaptation of the ICL criterion [3] called ICL-BIC is proposed in [9]. In practice, the algorithm has to be executed with all the (G, H) to test, and the highest ICL-BIC is retained.

2.4 The clustering model

The clustering model described in this section is a particular case of the co-clustering model, in which each feature is in its own cluster ($H = J$). Consequently $\boldsymbol{w}$ is not a latent variable anymore since each variable represents a cluster of size 1. Let define a multivariate ordinal variable $\boldsymbol{x}_i = (x_{ij})_j$ with $1 \leq j \leq J$. Conditionally to cluster g, the distribution of $\boldsymbol{x}_i$ is assumed to be:

$$p(\boldsymbol{x}_i|v_{ig}=1;\boldsymbol{\mu}_g,\boldsymbol{\pi}_g)=\prod_j p(x_{ij};\mu_{gj},\pi_{gj}),$$

where $\boldsymbol{\mu}_g=(\mu_{gj})_j$ and $\boldsymbol{\pi}_g=(\pi_{gj})_j$ with $1\leq j\leq J$. This conditional independence hypothesis assumes that conditionally to the belonging to row-cluster g, the J ordinal responses of an individual are independently drawn from J univariate BOS models of parameters $(\mu_{gj},\pi_{gj})_{j\in\{1,...,J\}}$. Furthermore, as in the co-clustering case, the distribution of $\boldsymbol{v}_i$ is assumed to be the multinomial distribution $\mathcal{M}(\gamma_1,...,\gamma_G)$ and not to depend on i. In this configuration, the parameter of the clustering model is defined as $\boldsymbol{\theta}=(\boldsymbol{\gamma},\boldsymbol{\alpha})$, with $\alpha_{gj}=(\mu_{gj},\pi_{gj})$ being the position and precision BOS parameters of the row-cluster g and ordinal variable j. Consequently, with a matrix $\boldsymbol{x}=(x_{ij})_{i,j}$ of ordinal data, the probability density function $p(\boldsymbol{x};\boldsymbol{\theta})$ of $\boldsymbol{x}$ is written:

$$p(\boldsymbol{x};\boldsymbol{\theta}) \;=\; \sum_{v\in V}\prod_{i,g}\gamma_g^{v_{ig}}\prod_{i,j,g}p(x_{ij};\mu_{gj},\pi_{gj})^{v_{ig}}. \tag{2}$$

To infer the parameters of this model, the SEM-Gibbs Algorithm 1 is used with the **1.2** part removed from the SE-step. The **1.3** part about missing value imputation remains as well. It is here noticed that a clustering can be also obtained by using the co-clustering of Section 2.3, and by considering the resulting $\boldsymbol{v}$ partition as the outcome. As a matter of fact, in this case, the co-clustering is a parsimonious version of the clustering.

2.5 The classification model

By considering a classification task with a categorical variable to predict from ordinal data, the encountered configuration is the particular case where $\boldsymbol{v}$ is known for all $i\in\{1,...,N\}$ and for all $g\in\{1,...,G\}$. In ordinalClust, two classification models are proposed.

Multivariate BOS model This first model is similar to the clustering model: each variable represents a column-cluster of size 1, thus $\boldsymbol{w}$ is not a latent variable. This model assumes that, conditionally on the class of the observations, the J variables are independent. Since the row classes are observed, the algorithm only needs to estimate the parameter $\boldsymbol{\theta}$ that maximizes the log-likelihood $l(\boldsymbol{\theta};\check{\boldsymbol{x}})$. The probability density function $p(\boldsymbol{x},\boldsymbol{v};\boldsymbol{\theta})$ is therefore expressed as below:

$$p(\boldsymbol{x},\boldsymbol{v};\boldsymbol{\theta}) \;=\; \prod_{i,g}\gamma_g^{v_{ig}}\prod_{i,j,g}p(x_{ij};\alpha_{gj})^{v_{ig}}. \tag{3}$$

The inference of this model's parameters only requires the M-step of Algorithm 1. However, if there are missing data, the SE-step made of **1.3** part only is also required.

Parsimonious BOS model This model is a parsimonious version of the first model. Parsimony is introduced by grouping the features into H clusters (as in the co-clustering model). The main hypothesis is that given the row-cluster partitions and the column-cluster partitions, the realization x_{ij} is independent from the other ones. In practice the number H of column-clusters is chosen with a training dataset and a validation dataset as follows. Consequently, the probability density function $p(\boldsymbol{x},\boldsymbol{v};\boldsymbol{\theta})$ is annotated:

$$p(\boldsymbol{x},\boldsymbol{v};\boldsymbol{\theta}) \;=\; \sum_{\boldsymbol{w}\in W}\prod_{i,g}\gamma_g^{v_{ig}}\prod_{j,h}\rho_h^{w_{jh}}\prod_{i,j,g,h}p(x_{ij};\alpha_{gh})^{v_{ig}w_{jh}}. \tag{4}$$

To infer this model's parameters, Algorithm 1 is used with an SE-step containing part **1.2** only, and the entire M-step. Again, if there are missing data, the SE-step made of **1.3** part is also required.

2.6 Handling ordinal data with several numbers of levels

The Latent Block Model as it is described before is not able to take variables with different levels m into account. Indeed, the distributions of variables with different numbers of levels are not defined on the same support. This implies that it is impossible to gather two variables with different m within a same block.
In [13], a constrained Latent Block Model is proposed. Although it does not make possible to gather ordinal features with different m in a same column-cluster, it is able to take into account

the fact that there are several m and therefore to perform a co-clustering on more diverse datasets. The matrix $\boldsymbol{x}$ is considered to contain D different numbers of levels. Its representation is seen as D matrices put side by side, such that the d^{th} table is a $N \times J_d$ matrix written $\boldsymbol{x^d}$, composed of ordinal data with numbers of levels m_d.

$$\boldsymbol{x} = \left[\left[\begin{array}{c} \\ \boldsymbol{x}^1 \\ \\ \end{array}\right] \ ... \ \left[\begin{array}{c} \\ \boldsymbol{x}^D \\ \\ \end{array}\right]\right], \text{ with } \boldsymbol{x}^d = (x_{ij}^d)_{i=1,...,N;\ j=1,...,J_d} \text{ and for } d = 1, ..., D.$$

The model relies on the following hypothesis:

$$p(\boldsymbol{x}^1, ...\boldsymbol{x}^D|\boldsymbol{v}, \boldsymbol{w}^1, ..., \boldsymbol{w}^D) = p(\boldsymbol{x}^1|\boldsymbol{v}, \boldsymbol{w}^1) \times ... \times p(\boldsymbol{x}^D|\boldsymbol{v}, \boldsymbol{w}^D),$$

with $\boldsymbol{w}^d$ the column partition of $\boldsymbol{x}^d$. This means there is an independence between the D blocks, knowing their row and column partitions: the realization of the univariate random variable x_{ij}^d will not depend on the column partitions of the other blocks than d.

In this case, the SEM-Gibbs algorithm to infer the parameters is slightly changed: in the SE-step, a sampling step is appended to for every additional $\boldsymbol{x}^d$. For further details on this adapted SEM-Gibbs algorithm, see [13].

3 Application on the patients quality of life analysis in oncology

This section explains how to use the implementation of the methods described before through ordinalClust package.

3.1 Datasets

The included datasets are taken from QoLR package [2]. They contain responses to questionnaires that were given to patients affected by breast cancer. Furthermore, for all the questions, the most positive answer is given by the level "1". For example, for the question: *"During the past week, did you feel irritable?"* with possible responses: *"Not at all."* *"A little."* *"Quite a bit."* *"Very much."*, the following levels number are respectively assigned to the replies: 1 *"Not at all."*, 2 *"A little."*, 3 *"Quite a bit."*, 4 *"Very much."*, because it is perceived as more negative to have felt irritable. Two datasets are available:

- `dataqol` is a data.frame with 121 lines such that each line represents a patient and the columns contain information about the patient:
  - `Id`: patient Id,
  - `q1-q28`: responses to 28 questions with number of levels equals to 4,
  - `q29-q30`: responses to 2 questions with number of levels equals to 7.
- `dataqol.classif` is a data.frame with 40 lines such that a line represents a patient, and the columns contain information about the patient:
  - `Id`: patient Id,
  - `q1-q28`: responses to 28 questions with number of levels equals to 4,
  - `q29-q30`: responses to 2 questions with number of levels equals to 7,
  - `death`: if the patient deceased (2) or not (1).

The datasets contain missing values, that are set to 0. To load the package and its dataset, the following commands must be executed:

```
library(ordinalClust)
data("dataqol")
data("dataqol.classif")
```

Then, a seed is set so that the user finds results identical to this document:

```
set.seed(5)
```

The user must define how many SEM-Gibbs iterations (`nbSEM`) and how many burn-in iterations (`nbSEMburn`) are needed for Algorithm 1. Section 3.6 provides an empirical way of checking rightness of these values. Furthermore, the `nbindmini` argument has to be defined: it indicates how many cells at least must be present in a block.

```
nbSEM <- 100
nbSEMburn <- 50
nbindmini <- 1
```

3.2 Performing a classification

In this section, the `dataqol.classif` dataset is used. The aim is to predict the `death` variable from the ordinal data that corresponds to the patients answers. The following snippet shows how to setup the classification configuration. First, the $\boldsymbol{x}$ ordinal data matrix (the responses to the questionnaires) is defined, as well as the $\boldsymbol{v}$ vector, which is the variable `death` to predict.

```
x <- as.matrix(dataqol.classif[,2:29])
v <- as.vector(dataqol.classif$death)
```

ordinalClust offers two classification models. The first one, (chosen by the option `kc=0`), is a multivariate BOS model assuming that, conditionally on the class of the observations, the features are independent. The second model introduces parsimony by grouping the features into clusters and assuming that the features of a cluster have a common distribution. The number H of clusters of features is defined with the argument `kc=H`. H is chosen thanks to a training dataset and a validation dataset:

```
nb.sample <- ceiling(nrow(x)*2/3)
sample.train <- sample(1:nrow(x), nb.sample, replace=FALSE)
x.train <- x[sample.train,]
x.validation <- x[-sample.train,]
v.train <- v[sample.train]
v.validation <- v[-sample.train]
```

Then, the classification can be performed with function `bosclassif` and several `kc` parameters are tested. In the following example, the `predictions` matrix contains the predictions resulting from the classifications performed with different `kc`.

```
predictions <- matrix(0,nrow=4,ncol=nrow(x.validation))

for(kc in 0:3){
  res <- bosclassif(x=x.train,y=v.train,to.predict=x.validation,
                    kr=2,kc=kc,m=4,nbSEM=nbSEM,nbSEMburn=nbSEMburn,
                    nbindmini=nbindmini)
  predictions[kc,] <- res$zr.to.predict
}
```

Table 1 shows the precision, recall and specificity for each different `kc`. The code to get these values is available in Appendix. First of all, the results are globally satisfying since the precision and recalls are pretty high. Then, it is clearly observed that the parsimonious model (when `kc=1,2` or `3`) has better results than the multivariate model (`kc=0`). As a matter of fact, the most parsimonious model with `kc=1` obtains the best results. This illustrates the interest of introducing parsimonious models in a supervised context.

Return of bosclassif function:

- `probas.to.predict`: Matrix with `nrow=nrow(to.predict)` and `ncol=kr`. For each observation of argument `to.predict`, indicates the probability to belong to each class.

Table 1: Precision, recall and specificity for different kc.

	precision	recalls	specificity
kc=0	0.73	0.89	0.25
kc=1	**0.82**	**1**	**0.50**
kc=2	0.80	0.89	**0.50**
kc=3	0.80	0.89	**0.50**

- `zr.to.predict`: Vector of length `nrow(to.predict)` which indicates the class label the observations of `to.predict` has been associated to by maximum at posteriori. If row i was associated to class g, then `zr.to.predict[i]=g`.
- `V.to.predict`: Matrix with `nrow=nrow(to.predict)` and `ncol=kr`. Indicates at which class the observations of `to.predict` has been associated. `V.to.predict[i,g]=1` if row i was associated to class g and `V.to.predict[i,g]=0` otherwise.
- `xhat`: Dataset with missing values completed.
- `mus`: For each iteration, an array of dimension $G \times H$ that represents the μ_{gh} that were estimated.
- `ps`: For each iteration, an array of dimension $G \times H$ that represents the π_{gh} that were estimated.
- `gamma`: For each iteration, a vector of length G containing the row mixing proportions γ_g that were estimated.
- `rho`: For each iteration, a vector of length H containing the column mixing proportions ρ_h that were estimated.
- `W`: For each iteration, array of dimension $J \times H$ such that `W[j,h]=1` if j belongs to cluster h.
- `res_mus`: Array of dimension $G \times H$ that represents the resulting position parameter μ_{gh} at the end of SEM-algorithm.
- `res_ps`: Array of dimension $G \times H$ that represents the resulting precision parameter π_{gh} at the end of SEM-algorithm.
- `res_gamma`: Vector with the resulting row mixing proportions γ_g.
- **res_rho**: Vector with the resulting column mixing proportions ρ_h.
- **res_V**: Array of dimension $N \times G$ such that **V[i,g]=1** if i belongs to cluster g at the end of the SEM-algorithm.
- `res_W`: Array of dimension $J \times H$ such that `W[j,h]=1` if j belongs to cluster h at the end of the SEM-algorithm.
- `icl`: ICL-BIC result value.
- `zc`: Vector with resulting column partitions.
- `probaV`: Array representing the probability for each row to belong to each row-cluster.
- `probaW`: Array representing the probability for each column to belong to each column-cluster.

3.3 Performing a clustering

Choosing G and H In the examples below, the choice for G (and H in case of co-clustering) were made by performing several clustering with $G = (2, 3, 4, 5)$, and several co-clustering with $G = (3, 4, 5)$ and $H = (2, 3, 4)$. In both cases, the G (or (G, H)) with the highest ICL-BIC was retained. In case of several numbers of levels (as in Section 2.6), testing all the possible values for $(G, H_1, ..., H_D)$ can be very long. In [13], a strategy is described to find a fine set $(G, H_1, ..., H_D)$.

This section uses the `dataqol` dataset. The clustering purpose is to emphasize information regarding the rows of a data matrix. First, the $\boldsymbol{x}$ ordinal matrix is loaded, it corresponds to the patients responses:

```
x <- as.matrix(dataqol[,2:29])
```

The clustering is obtained thanks to the `bosclust` function:

```
object <- bosclust(x=x, kr=3, m=4, nbSEM=nbSEM, nbSEMburn=nbSEMburn)
```

The outcome can be plotted thanks to the `bosplot` function:

```
bosplot(object)
```

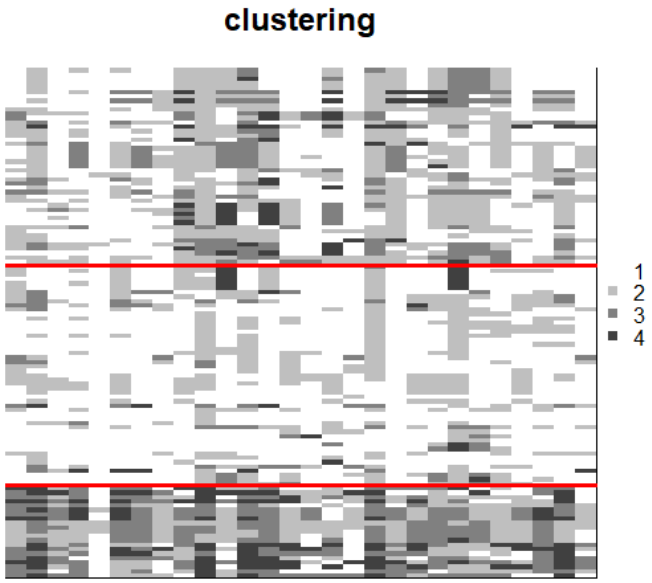

Figure 3: Obtained clustering when following the given example.

Figure 3 represents the clustering result. Among the 3 row-clusters, the second one stands out as the lightest. It means that the patients from this cluster globally chose levels close to 1, which is the most positive answer. Quite the opposite, the last row-cluster is darker which implies the patients from this group answered in a more negative way.

Return of bosclust function:

- `xhat`: Dataset with missing values completed.
- `mu`: BOS position parameters μ_{gj} that were estimated at each iteration of the SEM algorithm.
- `p`: BOS precision parameter π_{gj} that were estimated at each iteration of the SEM algorithm.
- `gamma`: Vector of length G indicating the γ_g row mixing proportions that were estimated at each iteration of the SEM algorithm.
- `V`: For each iteration of the SEM algorithm, array of dimension $N \times G$ such that `V[i,g]=1` if i belongs to cluster g and `V[i,g]=0` otherwise.
- `res_mu`: Array of dimension $G \times H$ with resulting position parameter μ_{gj} at the end of SEM-algorithm.
- `res_p`: Array of dimension $G \times H$ with resulting precision parameters π_{gj} at the end of SEM-algorithm.
- `res_gamma`: Vector of length G with the resulting row mixing proportions γ_g.
- `res_V`: Array of dimension $N \times G$ **V[i,g]=1** if i belongs to cluster g at the end of the SEM-algorithm and `V[i,g]=0` otherwise.

- `icl`: ICL-BIC resulting value.
- `zr`: Vector of length N with resulting row partitions labels.
- `probaV`: Array of dimension $N \times G$ representing the probability for each row to belong to each row-cluster.

3.4 Performing a co-clustering

Again, this section uses the `dataqol` dataset. The co-clustering is performed with the `boscoclust` function:

```
object <- boscoclust(x=x,kr=4,kc=4,m=4,nbSEM=nbSEM,nbSEMburn=nbSEMburn,
                     nbindmini=nbindmini)
```

As well as in the clustering context, the result can be plotted, with the command below, as in Figure 4.

```
bosplot(object)
```

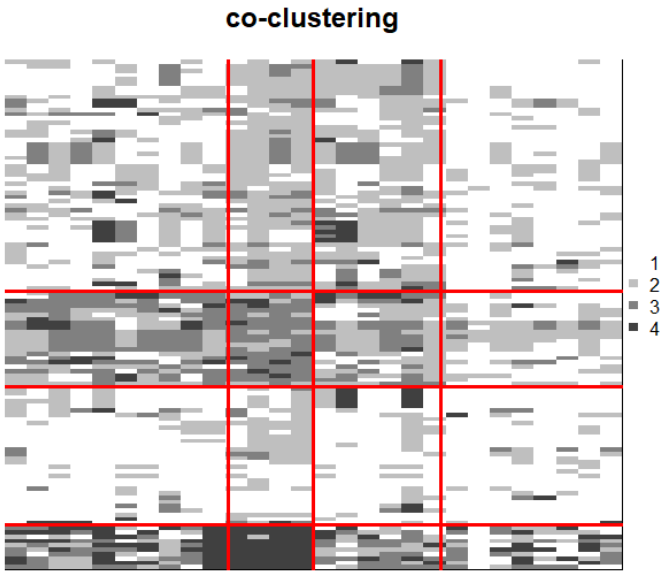

Figure 4: Obtained co-clustering when following the given example.

The resulting plot is given by Figure 4. In this case, the algorithm highlights a structure amid the rows. What's more, it also reveals a structure inherent to the columns: for example, the fourth column-cluster is lighter than the others, consequently, these questions are globally responded in a more positive way.

Return of boscoclust function:

- `xhat`: dataset with missing values completed.
- `mus`: For each iteration, an array of dimension $G \times H$ that represents the μ's that were estimated at each iteration.
- `ps`: For each iteration, an array of dimension $G \times H$ that represents the π's that were estimated at each iteration.
- `gamma`: Vector of length G indicating the row mixing proportions γ_g that were estimated at each iteration of the SEM algorithm.

- `rho`: Vector of length H indicating the column mixing proportions ρ_h that were estimated at each iteration of the SEM algorithm.
- `V`: For each iteration, array of dimension $N \times G$ such that `V[i,g]=1` if i belongs to cluster g.
- `W`: For each iteration, array of dimension $J \times H$ such that `W[j,h]=1` if j belongs to cluster h.
- `res_mus`: Array of dimension $G \times H$ that represents the resulting BOS position parameters μ's at the end of SEM-algorithm.
- `res_ps`: Array of dimension $G \times H$ that represents the resulting BOS precision parameters π's at the end of SEM-algorithm.
- `res_gamma`: Vector with the resulting row mixing proportions γ_g.
- `res_rho`: Vector with the resulting column mixing proportions ρ_h.
- `res_V`: Array of dimension $N \times G$ such that `V[i,g]=1` if i belongs to cluster g at the end of the SEM-algorithm.
- `res_W`: Array of dimension $J \times H$ such that `W[j,h]=1` if j belongs to cluster h at the end of the SEM-algorithm.
- `icl`: ICL-BIC resulting value.
- `zr`: Vector of length N with resulting row partitions labels.
- `zc`: Vector of length J with resulting column partitions labels.
- `probaV`: Array representing the probability for each row to belong to each row-cluster.
- `probaW`: Array representing the probability for each column to belong to each column-cluster.

3.5 Comparison of clustering and co-clustering.

ICL-BIC values Co-clustering can be seen as a parsimonious way of performing a clustering, that is why these two techniques are compared here. First, on a mathematical point of view, the ICL-BIC of the results is available with the following command:

```
object$icl
```

In clustering, the obtained ICL-BIC is equal to -3519.776, and in co-clustering -3309.310, which is better. Furthermore, the interpretation of row-clusters is more precise with the co-clustering. Indeed, on Figure 3, the row-clusters can be seen as a group of person which globally replied positively, a group of person which replied negatively, and a third one which replied in between. On the other hand, on Figure 4, an inherent structure of the data is better highlighted and brings more information: for each row-cluster, it is also easy to detect the questions that were replied negatively. Co-clustering can therefore be interpreted as a more efficient way of performing clustering.

ARI values on row partitions The Adjusted Rand Index [12] was computed on row partitions of co-clustering and clustering results. The obtained value is `0.41`, meaning that co-clustering creates a row partition close to clustering's one, without being identical.

3.6 Checking the SEMburn and nbSEMburn arguments.

The SEM-algorithm can be slow at reaching its stationary state, depending on the datasets. After having chosen arbitrary `nbSEM` and `nbSEMburn` arguments (in practice at least higher than 50), the stability of the algorithm has to be checked. For this reason, all the functions of the ordinalClust package return also the parameters estimation at each iteration of the SEM-algorithm. As a result, the evolution of the parameters can be analyzed and the user can be confident the returned parameters are well estimated. In the clustering case, for example, the evolution of the parameters can be visualized with a plot:

```
par(mfrow=c(2,1))
plot(object$ps[1,2,], type="l",ylim=c(0,1),
col="navy", main = "pi", ylab="pi values", xlab="SEM iterations")

plot(object$gamma[1,], type="l",ylim=c(0,1),
col="hotpink3", main = "gamma", ylab="pi values", xlab="SEM iterations")
```

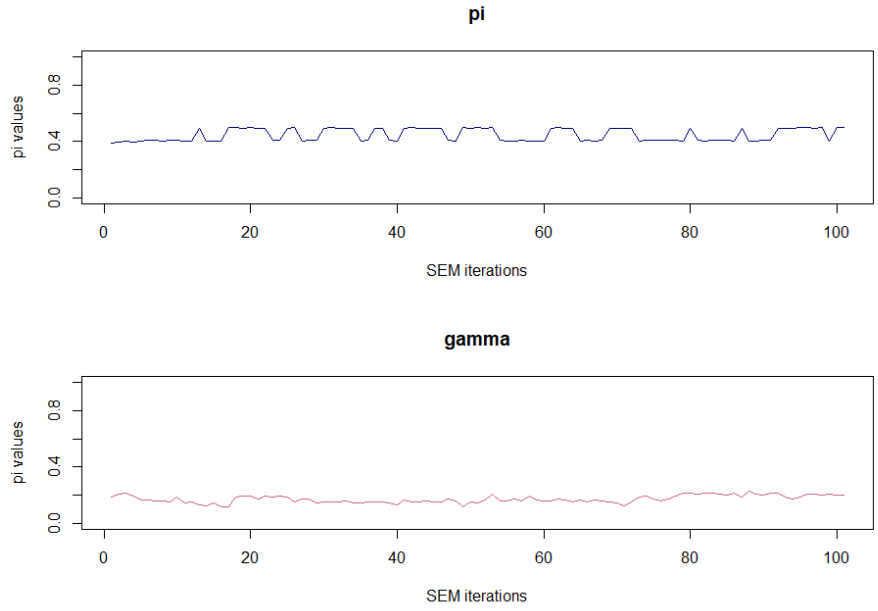

Figure 5: Evolution of some random parameters π and γ through SEM-Gibbs iterations, in the clustering example. It is observed that the parameters have reached a stationary state along time.

In Figure 5, it is easily observed that between the 50^{th} iteration and the 100^{th} iteration (corresponding to `nbSEM=100` and `nbSEMburn=50`), the parameters have reached on their stationary state. Therefore, `nbSEM=100` and `nbSEMburn=50` were well defined.

3.7 Handling data with different numbers of levels.

If the user wants to execute one of the function described before on variables with different m, then they should use the functions called `boscoclustMulti`, `bosclustMulti` and `bosclassifMulti`. Let assume the data is made of D different number of levels. The main changes regarding the arguments to pass are listed below:

- `m` must be vector of length D. The d^{th} element indicates the number of levels for the d^{th} group of variables.
- `kc` must be vector of length D. The d^{th} element indicates the number of column-clusters for the d^{th} group of variables.
- `d.list` is a new list argument of length D. The d^{th} item of the list indicates the column indexes that have the number of levels `m[d]`.

The vignettes provided with the package details an example on how to use `boscoclustMulti`.

4 Conclusion

The ordinalClust package presented in this paper implements several methods for analyzing ordinal data. In a first place, it proposes a co-clustering framework based on the Latent Block Model, coupled with a SEM-Gibbs algorithm and the BOS distribution. Moreover, it presents an adaptation of these techniques to perform clustering and classification on ordinal data. For the classification

method, two models are proposed, so that the user can introduce parsimony in their analysis. In a similar reasoning, it has been shown that the co-clustering method actually offers a parsimonious way of performing clustering. Besides, the framework is able to handle missing values which is notably relevant in the case of real datasets. Finally, these techniques are also implemented in the case of dataset with ordinal data having several numbers of levels.

References

[1] Alan Agresti. *Analysis of ordinal categorical data. Wiley Series in Probability and Statistics*, pages 397–405. John Wiley & Sons, Inc., 2012.

[2] Amelie Anota, Marion Savina, Caroline Bascoul-Mollevi, and Franck Bonnetain. Qolr: An r package for the longitudinal analysis of health-related quality of life in oncology. *Journal of Statistical Software, Articles*, 77(12):1–30, 2017.

[3] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(7):719–725, July 2000.

[4] Christophe Biernacki and Julien Jacques. Model-Based Clustering of Multivariate Ordinal Data Relying on a Stochastic Binary Search Algorithm. *Statistics and Computing*, 26(5):929–943, 2016.

[5] R. H. B. Christensen. ordinal—regression models for ordinal data, 2015. R package version 2015.6-28.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of he Royal Statistical Society, series B*, 39(1):1–38, 1977.

[7] Gérard Govaert and Mohamed Nadif. *Co-Clustering*. Computing Engineering series. ISTE-Wiley, 2014.

[8] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

[9] Julien Jacques and Christophe Biernacki. Model-Based Co-clustering for Ordinal Data. preprint <hal-01448299>, January 2017.

[10] Christine Keribin, Gérard Govaert, and Gilles Celeux. Estimation d'un modèle à blocs latents par l'algorithme SEM. In *42èmes Journées de Statistique*, Marseille, France, 2010.

[11] Damien McParland and Isobel Claire Gormley. *Algorithms from and for Nature and Life: Classification and Data Analysis*, chapter Clustering Ordinal Data via Latent Variable Models, pages 127–135. Springer International Publishing, Switzerland, 2013.

[12] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[13] Margot Selosse, Julien Jacques, and Christophe Biernacki. Analyzing health quality survey using constrained co-clustering model for ordinal data and some dynamic implication. preprint <hal-01643910>, 2018.

[14] J.K. Vermunt and J. Magidson. Technical guide for latent gold 4.0: Basic and advanced. statistical innovations inc., 2005.

Appendix

The following code allows to compute the precision, recall, specificity, and sensitivity that were obtained with the different **kc**, in Section 3.2

```
predictions = as.data.frame(predictions)
row.names <- c()
for(kc in kcol){
  name= paste0("kc=",kc)
  row.names <- c(row.names,name)
}
rownames(predictions)=row.names

library(caret)

actual <- v.validation -1
kcol <- c(0,1,2,3)

precisions <- rep(0,length(kcol))
recalls <- rep(0,length(kcol))
sensitivities <- rep(0,length(kcol))
specificities <- rep(0,length(kcol))

for(i in 1:length(kcol)){
  prediction <- unlist(as.vector(predictions[i,])) - 1
  conf_matrix<-table(prediction,actual)
  precisions[i] <- precision(conf_matrix)
  recalls[i] <- recall(conf_matrix)
  sensitivities[i] <- sensitivity(conf_matrix)
  specificities[i] <- specificity(conf_matrix)
}
```