HAL
open science

ordinalClust: an R package for analyzing ordinal data

Margot Selosse, Julien Jacques, Christophe Biernacki

To cite this version:

Margot Selosse, Julien Jacques, Christophe Biernacki. ordinalClust: an R package for analyzing ordinal data. 2019. hal-01678800v3

HAL Id: hal-01678800

https://inria.hal.science/hal-01678800v3

Preprint submitted on 8 Dec 2019 (v3), last revised 11 Sep 2020 (v4)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ordinalClust: An R Package to Analyse Ordinal Data

by Margot Selosse, Julien Jacques and Christophe Biernacki

Abstract Ordinal data are used in a lot of domains, especially when measurements are collected from people by observations, testings, or questionnaires. **ordinalClust** is an innovative R package dedicated to ordinal data that proposes tools for modeling, clustering, co-clustering and classifying them. Ordinal data are modeled by the BOS distribution, which is a model with two meaningful parameters referred to as "position" and "precision". The first one indicates the mode of the distributed and the other ones describes how scattered are data around the mode: the user is able to interpret easily the distribution of their data when they are given these two parameters. The package is based on the co-clustering framework (when rows and columns are simultaneously clustered). The co-clustering approach uses the Latent Block Model (LBM) and the 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. Indeed, the clustering can be seen as a simpler co-clustering whose column-clusters are not to be found. In the same way, classification can be seen as a simpler co-clustering whose row-clusters are not to be found on the training set. For the classification process, two approaches are proposed. In the first one, the BOS parameters are estimated on the training data set in a classic way. In the second approach parsimony is introduced by estimating the parameters and column-clusters on the training data set. We empirically show that this approach can yield better results. For the clustering and co-clustering process, the ICL-BIC criterion is used for model selection purposes. An overview of these methods is given, and the way of using them with the **ordinalClust** package is described through real data sets. The latest stable package version is available in source and binary form on the Comprehensive R Archive Network (CRAN).

Introduction

Ordinal data is a specific kind of categorical data occurring when the levels are ordered (Agresti, 2012). 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 the consumer's age. 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. The cumulative linked model (CLM) which assumes that:

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

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. CLMs are a powerful model class for ordinal data since observations are handled as categorical, their ordered nature is exploited and the regression framework allows interpretable analyses. In R, several packages implement these kind of models. The packages **MASS** (Venables and Ripley, 2002) implements the CLM with standard link functions, while **VGAM** (Yee, 2010), **rms** (Jr, 2019), **brms** (Bürkner, 2017) and **ordinal** (Christensen, 2015) bring additional functions and features. Other contributions implements algorithms for ordinal data classification. For instance, the **ordinalForest** package (Hornung, 2019) uses random forests and **monmlp** (Cannon, 2017) uses neural networks, both to predict ordinal response variables. Finally, the **ocapis** package (Heredia-Gómez et al., 2019) implements several methods (such as CMLs, Support Machine, Weighted k-Nearest-Neighbor) to classify and preprocess ordinal data.

However, these techniques' focus differs from ours in two ways. First, they work in a supervised framework (classification). Secondly, they work with data sets whose variables to predict are ordinal responses: the other variables are of different types. Our goal is to provide a tool for unsupervised and supervised tasks, and for data sets made of ordinal variables only (in the classification context, the response is categorical). From an unsupervised point a view, the Latent Gold Software Vermunt and Magidson is – to our knowledge – the only software that uses the CMLs to cluster the data. Nevertheless, the implementation of this method is known to be computationally expensive. In addition, it is not provided through a user-friendly R package.

Other contributions have defined clustering algorithms with variables of ordinal type. In McParland and Gormley (2013), the authors propose a model-based technique by considering the probability distribution of ordinal data as a discretisation of an underlying continuous variable. This approach is implemented in the **clustMD** package (McParland and Gormley, 2017), which is more generally for heterogeneous data. In Ranalli and Rocci (2016), the categorical variables are seen as a discretisation of an underlying finite mixture of Gaussians. In other works, authors use the Multinomial distribution to model the data. For instance in Giordan and Diana (2011), the Multinomial distribution and a cluster tree are used, whereas Jollois and Nadif (2009) applies a constrained Multinomial distribution. However, these contributions do not give a way to co-cluster and classify ordinal data. Furthermore, they are not always available as an R package (except for McParland and Gormley (2013)).

Finally, the CUB (Combination of a discrete Uniform and a shifted Binomial random variable) model (D'Elia and Piccolo, 2005) is widely used to analyse ordinal data sets. For instance, Corduas (2008) proposes a clustering algorithm based on a mixture of CUB models. In the CUB model, an answer is interpreted as the result of a cognitive process where the decision is intrinsically continuous but is expressed on a discrete scale of m levels. This approach interprets the choice of the respondent as a weighted combination of two components. The first reflects a personal feeling and is expressed by a shifted binomial random variable. The second component reflects an intrinsic uncertainty and is expressed by a uniform random variable. A lot of extensions for the CUB model were defined and the **CUB** package (Maria Iannario, 2018) implements the associated statistical methods.

More recently, Biernacki and Jacques (2016) proposed the so-called Binary Ordinal Search model, referred to as "BOS" 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, Jacques and Biernacki (2017) employed this distribution coupled to the Latent Block Model (Govaert and Nadif, 2003) in order to carry out a co-clustering on ordinal data. The co-clustering task consists in simultaneously clustering the rows and the columns of the data matrix. It is a useful way of clustering the data while introducing parsimony, and provide more interpretable partitions. The authors of Jacques and Biernacki (2017) 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. Selosse et al. (2019) used an extension of the Latent Block Model to overcome this issue. 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 **ordinalClust** package is to offer a complete tool for analyzing ordinal data by implementing these methods. Furthermore, it presents a novel approach for classifying ordinal data sets with categorical responses. The present document gives an overview of the underlying methods and illustrates the usage of **ordinalClust** through concrete examples. The paper is organised as follows. In Section 2.2, the notation and models are described. Section 2.3 presents the functions of **ordinalClust** and details a use-case on psychological survey data sets. Section 2.4 discusses the limits of **ordinalClust** and the future works on the package.

Statistical methods

Data Notation

A data set 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 data set can contain missing data. While dealing with this aspect, the data set 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.

The BOS model

The BOS model (Biernacki and Jacques, 2016) is a probability distribution for ordinal data parameterized 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 Biernacki and Jacques (2016) 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 performed through the EM algorithm (Dempster et al., 1977).

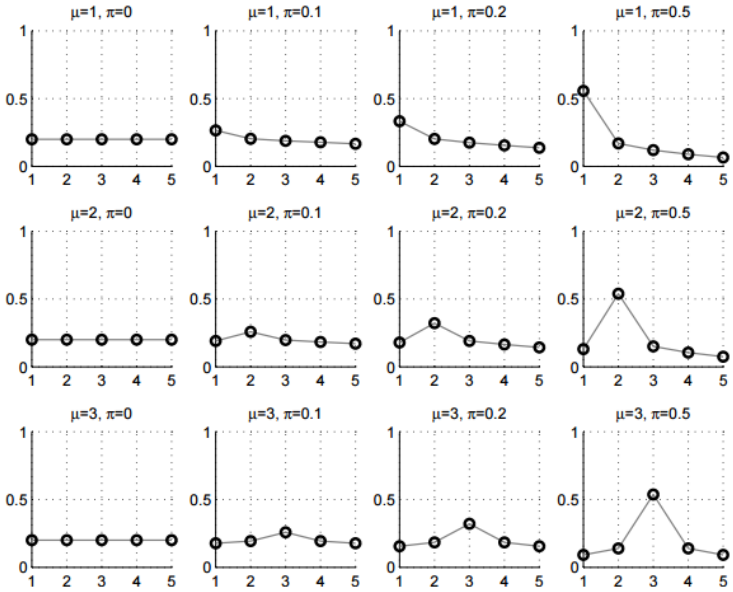

Figure 1: BOS distribution $p(x;\mu,\pi)$: shape for $m = 5$ and for different values of μ and π.

The co-clustering model

Notation Being in a co-clustering context, it is assumed that there are 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 gth (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 are G row-clusters and H column-clusters that respectively 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.

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 $\boldsymbol{\alpha} = (\alpha_{gh})_{g,h}$ is the distribution's parameters of block (g,h). Any univariate distribution can be used with respect 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:

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

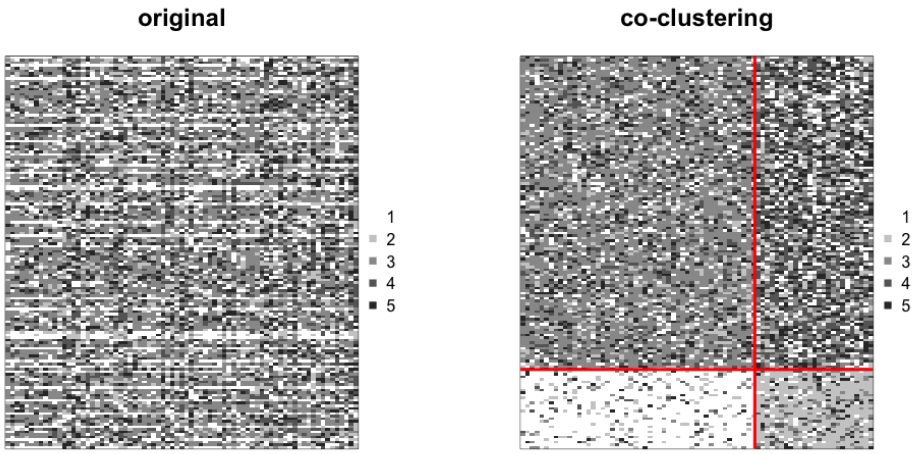

Figure 2: On the left: original data set made of ordinal data with $m = 5$. On the right: a co-clustering is performed with $G = H = 2$, the rows and columns are sorted by row-clusters and column clusters, which emphasizes a structure in the data set.

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 $\boldsymbol{\theta} = (\boldsymbol{\gamma}, \boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\pi})$, with $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_G)$ and $\boldsymbol{\rho} = (\rho_1, ..., \rho_H)$ the rows and columns mixing proportions; $\boldsymbol{\mu} = (\mu_{gh})_{g,h}$ and $\boldsymbol{\pi} = (\pi_{gh})_{g,h}$ are the blocks' distribution parameters. 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 $l(\boldsymbol{\theta}; \check{\boldsymbol{x}}) = \sum_{\hat{\boldsymbol{x}}} \log p(\boldsymbol{x}; \boldsymbol{\theta})$. The EM-algorithm (Dempster et al., 1977) 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 Section 2.3' example ($G = 3$, $H = 3$, $N = 117$ and $J = 28$) it would result in the computation of $3^{117} \times 3^{28} \approx 1 \times 10^{69}$ terms. There are different alternatives to the EM algorithm, such 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 (Keribin et al., 2010). Furthermore, it handles easily missing values $\hat{\boldsymbol{x}}$ in $\boldsymbol{x}$, which is an important advantage, particularly with real data sets. The SEM-algorithm is made of two

iteratively repeated steps that are detailed in Algorithm 1.

Data: $\boldsymbol{x}$, G, H
Result: A sequence $(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{\theta}, \hat{\boldsymbol{x}})^{(q)}$ for $q \in \{1, ..., nbSEM\}$
Initialization of $\hat{\boldsymbol{x}}$, $\boldsymbol{v}$, $\boldsymbol{w}$ and $\boldsymbol{\theta}$ by $\hat{\boldsymbol{x}}^{(0)}$, $\boldsymbol{v}^{(0)}$, $\boldsymbol{w}^{(0)}$ and $\boldsymbol{\theta}^{(0)}$ respectively;
for *q in 1:nbSEM* **do**

1. SE-step.

1.1 Sample the row partitions for all $1 \leq i \leq N$, $1 \leq g \leq G$:

$$p(v_{ig} = 1 | \boldsymbol{x}^{(q-1)}, \boldsymbol{w}^{(q-1)}; \boldsymbol{\theta}^{(q-1)}) \propto \gamma_g^{(q-1)} \prod_{j,h} p(x_{ij}; \mu_{gh}^{(q-1)}, \pi_{gh}^{(q-1)})^{w_{jh}^{(q-1)}}.$$

1.2 Sample the column partitions for all $1 \leq j \leq J$, $1 \leq h \leq H$:

$$p(w_{jh} = 1 | \boldsymbol{x}, \boldsymbol{v}^{(q)}; \boldsymbol{\theta}^{(q-1)}) \propto \rho_h^{(q-1)} \prod_{i,g} p(x_{ij}; \mu_{gh}^{(q-1)}, \pi_{gh}^{(q-1)})^{v_{ig}^{(q)}}.$$

1.3 Generate the missing data:

$$p(\hat{x}_{ij}^{(q)} | \check{\boldsymbol{x}}, \boldsymbol{v}^{(q)}, \boldsymbol{w}^{(q)}; \boldsymbol{\theta}^{(q-1)}) = \prod_{g,h} p(\hat{x}_{ij}; \mu_{gh}{}^{(q-1)}, \pi_{gh}{}^{(q-1)})^{v_{ig}^{(q)} w_{gh}{}^{(q)}}.$$

2. M-step.

2.1 Update the mixing proportions:

$$\rho_h^{(q)} = \frac{1}{J} \sum_j w_{jh}^{(q)} \text{ and } \gamma_h^{(q)} = \frac{1}{N} \sum_i v_{ig}^{(q)}.$$

2.2 Update the parameters $\boldsymbol{\mu}^{(q)}$ and $\boldsymbol{\pi}^{(q)}$ (see Biernacki and Jacques (2016)).

end

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

Initializations The **ordinalClust** package allows three modes for values initialization. It is set through the argument `init`, which can take values `"random"`, `"kmeans"` or `"randomBurnin"`. The first one randomly initializes $\boldsymbol{v}^{(0)}$ and $\boldsymbol{w}^{(0)}$ with the Multinomial distribution $\mathcal{M}(1/G, \ldots, 1/G)$ and $\mathcal{M}(1/H, \ldots, 1/H)$ respectively. The second (by default) value consists in performing a Kmeans algorithm (Hartigan and Wong, 1979) on the rows and on the columns.

The third one, `"randomBurnin"` is a bit more complex and requires additional arguments for the algorithm. It aims at avoiding a degeneracy of the algorithm that leads to empty clusters, knowing that the degeneracy event arises more often at the early stage of the algorithm (thus during the burning period. It starts with a first random initialization. However, for the first `nbSEMburn` iterations (`nbSEMburn` $<$ `nbSEM`), whenever a row-cluster gets empty, a percentage `percentRandomB` of the row partitions is resampled from the Multinomial distribution $\mathcal{M}(1/G, \ldots, 1/G)$. Similarly when a column-cluster gets empty, a percentage of the column partitions is resampled from the Multinomial distribution $\mathcal{M}(1/H, \ldots, 1/H)$.

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 the "sampling distribution" hereafter. While the final estimation of the position parameters $\hat{\boldsymbol{\mu}}$ are the mode of the sampling distributions, the final estimations of the continuous parameters $(\hat{\pi}, \hat{\boldsymbol{\gamma}}, \hat{\boldsymbol{\rho}})$ are the mean 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-steps (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 (Biernacki et al., 2000) called ICL-BIC is proposed in Jacques and

Biernacki (2017). In practice, the algorithm has to be executed with all the (G, H) to test, and the highest ICL-BIC is retained.

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:

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

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 2.2.3, 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 procedure.

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 data set and a validation data set 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.

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

Figure 3: Data set matrix $\boldsymbol{x}$ when the ordinal data has D numbers of levels.

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 Selosse et al. (2019), 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 data sets. 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 dth table is a $N \times J_d$ matrix written $\boldsymbol{x}^d$, composed of ordinal data with numbers of levels m_d (see Figure 3).

The model relies on the following hypothesis:

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

with $\boldsymbol{w}^d$ the column partition of $\boldsymbol{x}^d$. It means there is 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 Selosse et al. (2019).

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.

Data sets

The data sets included were part of the **QoLR** package (Anota et al., 2017). They contain responses to the well known "EORTC QLQ-C30" (European Organization for Research and Treatment of Cancer (EORTC) Quality of Life Questionnaire (QLQ-C30)), that was given to patients affected by breast cancer. Furthermore, for all questions, the most positive answer is given by the level "1". For example, for question: *"During the past week, did you feel irritable?"* with possible responses: *"Not at all." "A little." "Quite a bit." "Very much."*, the following level numbers 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 data sets are available:

- `dataqol` is a data.frame with 117 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 passed away (2) or not (1).

The data sets contain missing values, that are coded as NA: in dataqol, there are 1.1% of missing values and 3.6% in dataqol.classif. To load the package and its data sets, 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(1)
```

The user must define how many SEM-Gibbs iterations (nbSEM) and how many burn-in iterations (nbSEMburn) are needed for Algorithm 1. Section 2.3.7 provides an empirical way of checking rightness of these values. Moreover, the nbindmini argument has to be defined: it indicates how many cells at least must be present in a block. At last, the init argument indicates how to initialize the algorithm. It can be set to "kmeans", "random" or "randomBurnin".

```
nbSEM <- 150
nbSEMburn <- 100
nbindmini <- 1
init <- "randomBurnin"
percentRandomB <- c(50, 50)
```

Here, percentRandom is a vector because it defines two percentages: the percentage of rows that will be resampled if a row-clusters gets empty, and the percentage of column that will be resampled if a column-cluster get empty.

Performing a classification

In this section, the dataqol.classif data set is used. The aim is to predict the death variable from the ordinal data that corresponds to the patients answers. The following commands show how to setup the classification configuration. First, the x ordinal data matrix (the responses to the questionnaires) is defined, as well as the 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 like in Equation 3. The second model introduces parsimony by grouping the features into clusters and assuming that the features of a cluster have a common distribution, like in Equation 4. This latter is a novel approach for classification. The number H of clusters of features is defined with the argument kc = H. H is chosen thanks to a training data set and a validation data set:

```
# sampling data sets for training and to predict
nb.sample <- ceiling(nrow(x)*7/10)
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]
```

We also indicate how many classes there are, and how many levels the ordinal data have:

```
# classes
kr <- 2
# levels
m <- 4
```

The training can be performed thanks to the function bosclassif. In the code below, several kc parameters are tested. When kc = 0, the multivariate model is used: all variables are considered to be independent. When kc >0, the parsimonious model is used: the variables are grouped into kc

groups. To classify new observation, the `predict` function is used: it takes as arguments the result from `bosclassif` and the observations to classify. In the following example, we stock in the `preds` matrix the predictions resulting from the classifications performed with different `kc`.

```
kcol <- c(0, 1, 2, 3, 4)
preds <- matrix(0, nrow = length(kcol), ncol = nrow(x.validation))

for( kc in 1:length(kcol) ){
  classif <- bosclassif(x = x.train, y = v.train, kr = kr, kc = kcol[kc],
                m = m, nbSEM = nbSEM, nbSEMburn = nbSEMburn,
                nbindmini = nbindmini, init = init,
                percentRandomB = percentRandomB)
  new.prediction <- predict(classif, x.validation)
  preds[kc,] <- new.prediction@zr_topredict
}
```

Then the `preds` matrix can be formatted to a data.frame:

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

preds
v.validation

> preds
     V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12
kc=0  2  1  2  2  2  2  1  1  1   2   1   2
kc=1  2  1  2  1  2  2  1  2  1   1   2   2
kc=2  2  1  2  2  2  2  1  2  1   2   2   2
kc=3  2  1  2  1  2  2  1  2  1   2   1   2
kc=4  1  1  2  1  1  1  1  2  1   2   1   2
> v.validation
 [1] 2 1 1 1 1 1 1 2 1 1 1 2
```

Table 1 shows the sensitivity and specificity for each different `kc`. The code to get these values is available in Appendix A1. First of all, the results are globally satisfying since the sensitivities and specificities are pretty high. Then, it is clearly observed that the parsimonious models (when `kc = 1,2,3,4`) have better results than the multivariate model (`kc = 0`). The two parsimonious models `kc = 1` and `kc = 3` obtains the best results. This illustrates the interest of introducing parsimonious models in a supervised context.

Table 1: Sensitivity and specificity for different `kc`.

	sensitivity	specificity
kc = 0	0.67	0.44
kc = 1	**1.00**	**0.57**
kc = 2	1.00	0.33
kc = 3	**1.00**	**0.57**
kc = 4	0.78	0.67

Performing a clustering

Clustering setting. This section uses the `dataqol` data set, plotted in Figure 4.

The clustering purpose is to emphasize information regarding the rows of a data matrix. First, the x ordinal matrix is loaded, it corresponds to the patients' responses:

```
set.seed(1)
x <- as.matrix(dataqol[,2:29])
```

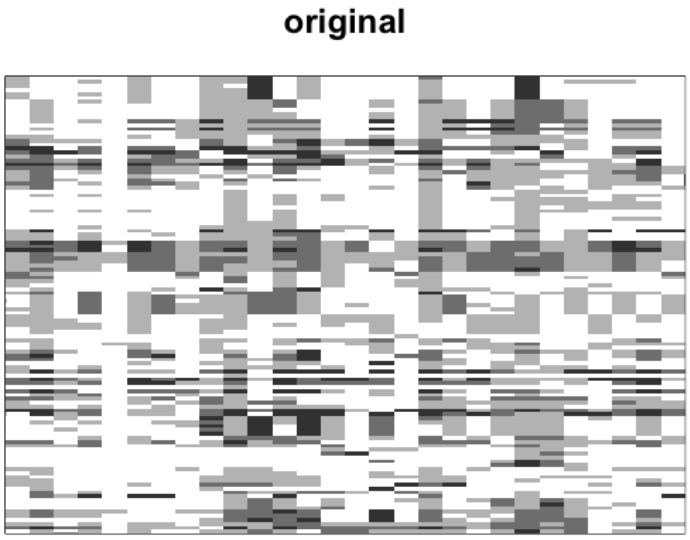

Figure 4: Plot of the dataqol dataset. The rows represent the patients, the columns represent the questions they answered to. A cell is the response of a patient to a question. The blacker the cell is, the more negative was the answer.

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

```
clust <- bosclust(x = x, kr = 3, m = 4,
            nbSEM = nbSEM, nbSEMburn = nbSEMburn,
            nbindmini = nbindmini, init = init)
```

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

```
plot(clust)
```

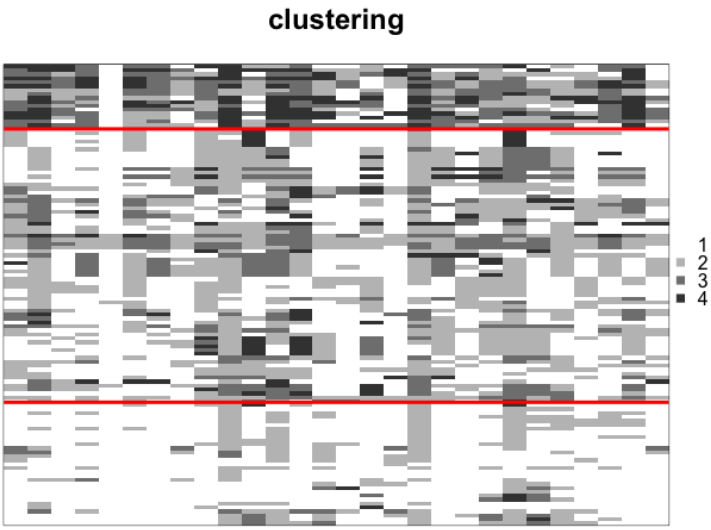

Figure 5: Clustering obtained when following the given example.

Figure 5 represents the clustering result. We count the clusters from the bottom to the top. Among the 3 row-clusters, the first one (at the bottom) 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 third row-cluster (at the top) is darker which implies the patients from this group answered in a more negative way.

Clusters interpretation. The parameters are obtained with the command `clust@params`:

```
> clust@params
[[1]]
[[1]]$mus
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14]
[1,]    1    1    1    1    1    1    1    1    1     2     1     2     1     1
[2,]    2    2    1    1    1    2    1    1    2     2     1     3     2     1
[3,]    3    4    3    3    1    4    3    2    3     4     2     4     3     4
     [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25] [,26]
[1,]     1     1     1     2     1     1     1     2     1     1     1     1
[2,]     1     1     1     2     2     1     2     2     1     2     1     1
[3,]     1     1     1     4     3     3     3     3     2     2     2     3
     [,27] [,28]
[1,]     1     1
[2,]     1     1
[3,]     4     1

[[1]]$pis
          [,1]      [,2]     [,3]      [,4] [,5]      [,6]      [,7]      [,8]
[1,] 0.8079608 0.6673682 0.961979 0.7770536    1 0.9619790 1.0000000 0.8852379
[2,] 0.3946294 0.3736864 0.722322 0.4690402    1 0.3567357 0.5546162 0.6402318
[3,] 0.4319502 0.5928978 0.347433 0.4930463    1 0.2718517 0.5888644 0.3310052
          [,9]     [,10]     [,11]     [,12]     [,13]     [,14]     [,15]
[1,] 0.9246885 0.5903583 0.6951631 0.5438752 0.9226941 0.4932884 0.8825371
[2,] 0.4767814 0.6937982 0.1481492 0.1859040 0.1176366 0.6624020 0.7916167
[3,] 0.3220447 0.7079570 0.4084469 0.5779180 0.5745136 0.1691940 0.3161048
         [,16]     [,17]     [,18]     [,19]     [,20]     [,21]     [,22]
[1,] 0.8036703 0.7364791 0.6643935 1.0000000 0.9619790 0.6951631 0.5681893
[2,] 0.3054584 0.8394348 0.5440131 0.3395749 0.4757433 0.4142450 0.3805989
[3,] 0.1255990 0.4281432 0.5470879 0.4280508 0.2300193 0.5776385 0.2632960
         [,23]     [,24]     [,25]     [,26]     [,27]     [,28]
[1,] 0.4905033 0.5510665 0.8167944 0.7477762 0.8521366 0.9226941
[2,] 0.3870155 0.4064222 0.6484691 0.4666815 0.3530825 0.6599010
[3,] 0.4183768 0.4709545 0.1959082 0.5465595 0.6419857 0.4174326
```

`cluser@params` is a list: when the data have D numbers of levels like in Figure 3, the list is $D-$long. Here the data has only one number of levels, so `cluser@params` has one element. Each element of the list has two attributes `pis` and `mus`. They indicate the π and μ values for each row-cluster and each column. Here, we see that, as observed with Figure 5, the first row-cluster has globally lower parameters μ, which means that people from this cluster globally answered in a more positive way to the questions. We also notice the π parameters for the fifth variable, (fifth question): they are all equal to 1. It means that the dispersion around the position μ is null. When observing the μ parameters for the fifth variables, they are also all equal to 1. It means that everybody answered in a positive way to this question. The fifth question of the EORTC QLQ-C30 questionnaire is "Do you need help with eating, dressing, washing yourself or using the toilet?". Therefore we know that none of the participants had problems to get ready and eat the week before they answered the questionnaire.

Choosing G. In the example above, the choice for G was made by performing several clustering with $G = (2, 3, 4)$. Thanks to the command `object@icl`, we know which result has the highest ICL value. The G with the highest ICL-BIC was retained, that is to say $G = 3$. The code to perform these clusterings is available in Appendix A2.

Performing a co-clustering

Co-clustering setting. Again, this section uses the `dataqol` data set. The co-clustering is performed with the `boscoclust` function:

```
set.seed(1)
coclust <- boscoclust(x = x, kr = 3, kc = 3, m = 4,
```

```
                    nbSEM = nbSEM, nbSEMburn = nbSEMburn,
                    nbindmini = nbindmini, init = init)
```

As well as in the clustering context, the result can be plotted, with the command below, as in Figure 6.

```
plot(coclust)
```

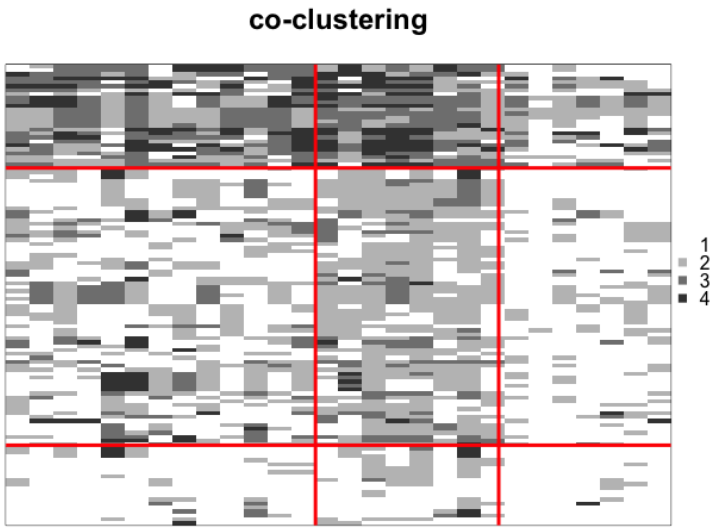

Figure 6: Co-clustering obtained when following the given example.

In this case, the algorithm highlights a structure amid the rows, as for the clustering Figure 5. What's more, it also reveals a structure inherent to the columns: for example, the third column-cluster is lighter than the others, consequently, these questions are globally responded in a more positive way.

Co-clusters interpretation. Once again, the parameters of the co-clustering are available through the command `coclust@params`:

```
> coclust@params
[[1]]
[[1]]$mus
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    2    1
[3,]    3    3    1

[[1]]$pis
          [,1]      [,2]      [,3]
[1,] 0.8496224 0.6266097 0.9426305
[2,] 0.4876194 0.5340329 0.7722278
[3,] 0.2638594 0.3044552 0.3623779
```

To know which questions belong to the third column-cluster (the one whose corresponding blocks are lighter), we need the command `coclust@zc`, which indicates the column-cluster of each column. `coclust@zc` is also a list of length D (when we have different numbers of levels). Here, $D = 1$ so we need `coclust@zc[[1]]`:

```
which(coclust@zc[[1]] == 3)
\end{CodeInput}
\begin{CodeOutput}
[1]  3  5  8 15 17 25 28
```

We know that questions 3, 5, 8, 15, 17, 25 and 28 are globally the ones that were answered the more positively. Here is the list of these questions in the EORTC QLQ C30:

- 3. Do you have any trouble taking a short walk outside of the house?

- 5. Do you need help with eating, dressing, washing yourself or using the toilet?
- 8. During the past week, were you short of breath?
- 15. During the past week, have you vomited??
- 17. During the past week, have you had diarrhea?
- 25. During the past week, have you had difficulty remembering things?
- 28. During the past week, has your physical condition or medical treatment caused you financial difficulties?

Choosing G and H. In the examples above, the choice for G and H were made by performing several co-clustering with $G = (2, 3, 4)$ and $H = (2, 3, 4)$. In both cases, the couple (G, H) with the highest ICL-BIC value was retained, that is to say, for $(G, H) = (3, 3)$. The code to search the highest ICL value is given in Appendix A3[1]

Missing values.

In this section we use the `dataqol` dataset. It has 1.1% of missing values (40 missing cells). The SEM-algorithm is able to handle these values since at each Expectation step (see 1, it computes the expectation of the missing values. The following code get the missing values index and print their values imputed by the clustering (resp. co-clustering) algorithm in Section 2.3.4 (resp. Section 2.3.3.

```
missing <- which(is.na(x))
missing

values.imputed.clust <- clust@xhat[[1]][missing]
values.imputed.clust

values.imputed.coclust <- coclust@xhat[[1]][missing]
values.imputed.coclust
```

```
> missing
 [1]  148  177  278  352  380  440  450  559  996 1058 1496 1513 1611 1883 1981
[16] 2046 2047 2050 2085 2285 2402 2450 2514 2517 2518 2663 2754 2785 2900 2902
[31] 2982 2986 3060 3152 3366 3367 3368 3520 3572 3602
> values.imputed.clust
 [1] 4 4 4 1 1 1 4 4 1 4 4 4 4 1 4 1 1 1 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 4 1 1 1
> values.imputed.coclust
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

We see that the co-clustering and the clustering algorithm had different values imputed for the missing data.

Comparison of clustering and co-clustering.

Co-clustering as a parsimonious clustering. Co-clustering can be seen as a parsimonious way of performing a clustering, that is why these two techniques are compared here. For example, the interpretation of row-clusters is more precise with the co-clustering. Indeed, on Figure 5, the row-clusters can be seen as a group of people who globally replied positively, a group of people who replied negatively, and a third one who replied in between. On the other hand, on Figure 6, 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. Furthermore the interpretation of the parameters was easier with the co-clustering result because it had only 18 parameters: $kr \times kc$ for $\boldsymbol{\pi}$ and $kr \times kc$ for $\boldsymbol{\mu}$. The clustering result had 168 parameters ($kr \times J$ for $\boldsymbol{\pi}$ and $kr \times J$ for $\boldsymbol{\mu}$), which is a lot to process for the user.

ARI values on row partitions The Adjusted Rand Index (Rand, 1971) was computed on row partitions of co-clustering and clustering results, thanks to the package **mclust** Chris Fraley (2019).

```
mclust::adjustedRandIndex(coclust@zr, clust@zr)
```

[1] In case of several numbers of levels, testing all the possible values for $(G, H_1, ..., H_D)$ can be tedious. In that case, the user is invited to implement a specific heuristical strategy as in Selosse et al. (2019).

The value obtained is `0.41`, meaning that co-clustering creates a row partition related to clustering's one, without being identical.

Setting the SEMburn and nbSEMburn arguments.

The SEM-algorithm can be slow at reaching its stationary state, depending on the data sets. 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. Indeed, the `pichain`, `rhochain` and `paramschain` slots respectively represent the γ, $\boldsymbol{\rho}$ and $\boldsymbol{\alpha}$ values for each iteration. 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 co-clustering case, for example, the evolution of the parameters can be visualized through a plot:

```
par(mfrow=c(3,3))
for(kr in 1:3){
    for(kc in 1:3){
        toplot <- rep(0, nbSEM)
        for(i in 1:nbSEM){
            toadd <- coclust@paramschain[[1]]$pis[kr,kc,i]
            toplot <- c(toplot, toadd)
        }
        plot.default(toplot, type = "l",ylim = c(0,1),
                 col = "hotpink3", main = "pi",
                 ylab = paste0("pi_", kr, kc, "values"),
                 xlab = "SEM-Gibbs iterations")
    }
}
```

In Figure 7, it is easily observed that between the 100th iteration and the 150th iteration (corresponding to `nbSEM=100` and `nbSEMburn=150`), the parameters have reached on their stationary state. Therefore, `nbSEM=150` and `nbSEMburn=100` were well defined.

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 same function following some changes in the arguments definition. Let assume the data is made of D different number of levels. First of all, the matrix `x`'s columns have to be grouped by same number of level `m[d]`. The additional changes regarding the arguments to pass are listed below:

- `m` must be vector of length D. The dth element indicates the number of levels for the dth group of variables.
- `kc` must be vector of length D. The dth element indicates the number of column-clusters for the dth group of variables.
- `idx_list` is a new vector argument of length D. The dth item of the vector indicates the index of the first column that have the number of levels `m[d]`.

An example on the `dataqol` dataset is available in Appendix A4.

Conclusion

The **ordinalClust** package presented in this paper implements several methods for analyzing ordinal data. First, it proposes a clustering and co-clustering framework based on the Latent Block Model, coupled with a SEM-Gibbs algorithm and the BOS distribution. Moreover, it defines a novel approach to classify 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 data sets. Finally, these techniques are also implemented in the case of data set with ordinal data having several numbers of levels. The package **ordinalClust** is available in source and binary form on the Comprehensive R Archive Network (CRAN). It is still under active development. A future work will implement the method defined in Gelman and Rubin (1992), to automatically define the number of iterations of the SEM-Gibbs algorithm.

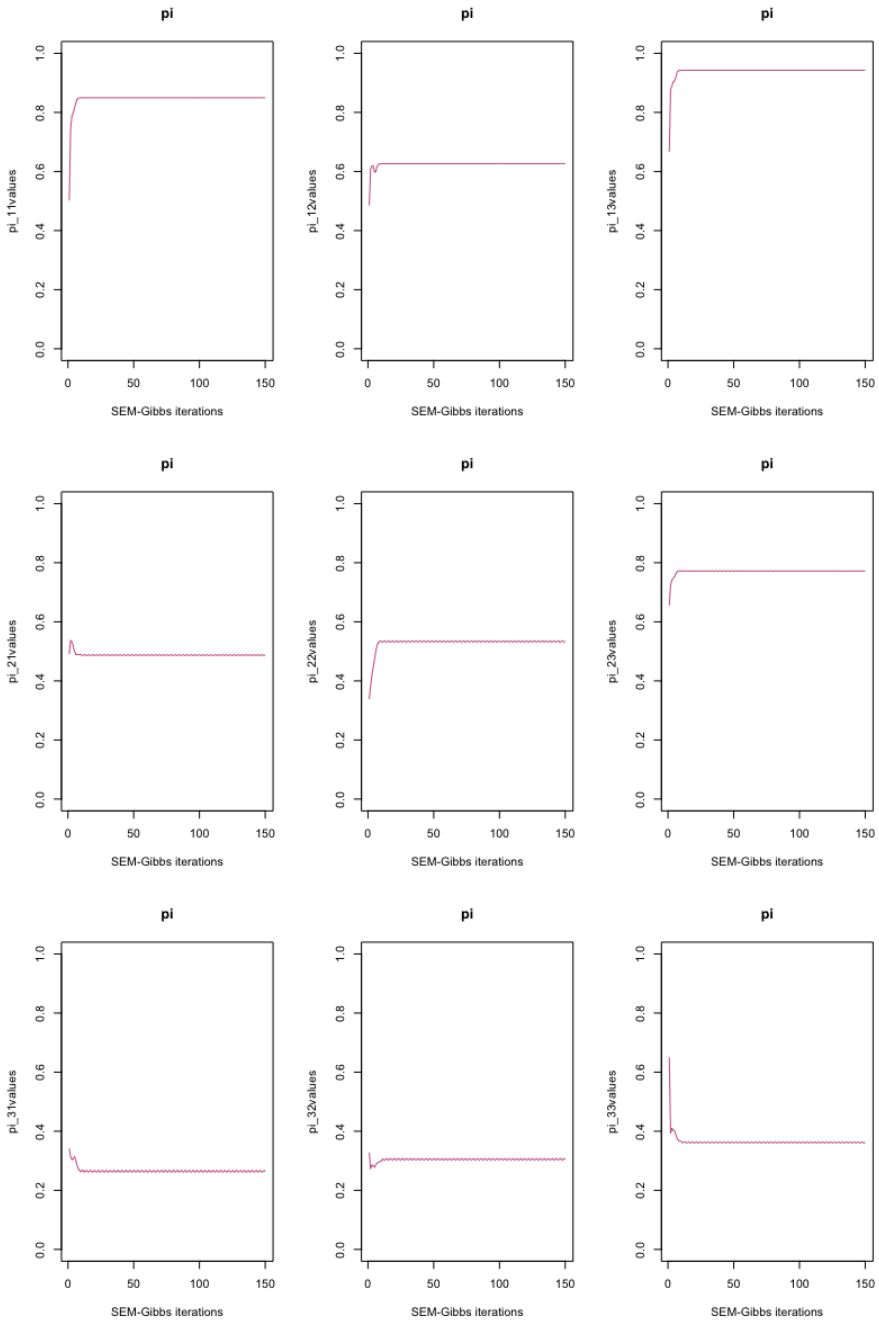

Figure 7: Evolution of π parameters through SEM-Gibbs iterations, in the clustering example. It is observed that the parameters have reached a stationary state along time.

Bibliography

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

A. Anota, M. Savina, C. Bascoul-Mollevi, and F. 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. [p]

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

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

P.-C. Bürkner. brms: An r package for bayesian multilevel models using stan. *Journal of Statistical Software, Articles*, 80(1):1–28, 2017. ISSN 1548-7660. [p]

A. J. Cannon. *monmlp: Multi-Layer Perceptron Neural Network with Optional Monotonicity Constraints*, 2017. R package version 1.1.5. [p]

F. Chris, R. Adrian E., S. Luca, M. Thomas Brendan, and F. Michael. mclust: Gaussian mixture modelling for model-based clustering, classification, and density estimation. 2019. R package version 5.4.3. [p]

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

M. Corduas. A statistical procedure for clustering ordinal data. *Quaderni di statistica*, 10:177–189, 2008. [p]

A. D'Elia and D. Piccolo. A mixture model for preferences data analysis. *Computational Statistics & Data Analysis*, 49(3):917–934, June 2005. [p]

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. [p]

A. Gelman and D. Rubin. Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457–472, 1992. ISSN 08834237. [p]

M. Giordan and G. Diana. A clustering method for categorical ordinal data. *Communications in Statistics - Theory and Methods*, 40(7):1315–1334, 2011. [p]

G. Govaert and M. Nadif. Clustering with block mixture models. *Pattern Recognition*, 36:463–473, 2003. [p]

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

M. C. Heredia-Gómez, S. García, P. A. Gutiérrez, and F. Herrera. Ocapis: R package for ordinal classification and preprocessing in scala. *Progress in Artificial Intelligence*, Mar 2019. ISSN 2192-6360. [p]

R. Hornung. *ordinalForest: Ordinal Forests: Prediction and Variable Ranking with Ordinal Target Variables*, 2019. R package version 2.3-1. [p]

V. J. and M. J. Technical guide for latent gold 4.0: Basic and advanced. statistical innovations inc., 2005. [p]

J. Jacques and C. Biernacki. Model-Based Co-clustering for Ordinal Data. preprint <hal-01448299>, Jan. 2017. [p]

F.-X. Jollois and M. Nadif. Classification de données ordinales : modèles et algorithmes. In *41èmes Journées de Statistique, SFdS, Bordeaux*, Bordeaux, France, 2009. [p]

F. E. H. Jr. *rms: Regression Modeling Strategies*, 2019. R package version 5.1-3.1. [p]

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

R. S. Maria Iannario, Domenico Piccolo. *CUB: A Class of Mixture Models for Ordinal Data*, 2018. R package version 1.1.3. [p]

D. McParland and I. C. 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. [p]

D. McParland and I. C. Gormley. *clustMD: Model Based Clustering for Mixed Data*, 2017. R package version 1.2.1. [p]

M. Ranalli and R. Rocci. Mixture models for ordinal data: A pairwise likelihood approach. *Statistics and Computing*, 26(1-2):529–547, Jan. 2016. ISSN 0960-3174. [p]

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

M. Selosse, J. Jacques, C. Biernacki, and F. Cousson-Gélie. Analysing a quality-of-life survey by using a coclustering model for ordinal data and some dynamic implications. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 2019. [p]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0. [p]

T. W. Yee. The vgam package for categorical data analysis. *J Stat Softw*, 2010. [p]

Appendix

Specificity and sensitivity

The following code allows to compute the specificity, and sensitivity that were obtained with the different kc, in Section 2.3.2:

```
library(caret)

actual <- v.validation - 1

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

for(i in 1:length(kcol)){
  prediction <- unlist(as.vector(preds[i,])) - 1
  u <- union(prediction, actual)
  conf_matrix <- table(factor(prediction, u),factor(actual, u))
  sensitivities[i] <- recall(conf_matrix)
  specificities[i] <- specificity(conf_matrix)
}

sensitivities
specificities
```

```
> sensitivities
[1] 0.6666667 1.0000000 1.0000000 1.0000000 0.7777778
> specificities
[1] 0.4444444 0.5555556 0.3333333 0.5555556 0.6666667
```

ICL search for clustering

```
set.seed(1)

library(ordinalClust)
data("dataqol")
M <- as.matrix(dataqol[,2:29])

nbSEM <- 150
nbSEMburn <- 100
nbindmini <- 2
init <- "randomBurnin"
percentRandomB <- c(50)
icl <- rep(0,3)

for(kr in 2:4){
    object <- bosclust(x = M, kr = kr, m = 4, nbSEM = nbSEM,
                nbSEMburn = nbSEMburn, nbindmini = nbindmini,
                percentRandomB = percentRandomB, init = init)

    if(length(object@icl)) icl[kr-1] <- object@icl
}
icl
```

```
> icl
[1] -3713.311 -3192.351 0
```

We see that the clustering algorithm could not find a solution without empty cluster for `kr = 4`. The highest icl is for `kr = 3`.

ICL search for co-clustering

```
set.seed(1)
library(ordinalClust)
```

```
data("dataqol")
M <- as.matrix(dataqol[,2:29])

nbSEM <- 150
nbSEMburn <- 100
nbindmini <- 2
init <-  "randomBurnin"
percentRandomB <- c(50, 50)
icl <- matrix(0, nrow = 3, ncol = 3)

for(kr in 2:4){
    for(kc in 2:4){
        object <- boscoclust(x = M,kr = kr, kc = kc, m = 4, nbSEM = nbSEM,
                        nbSEMburn = nbSEMburn, nbindmini = nbindmini,
                        percentRandomB = percentRandomB, init = init)
        if(length(object@zr)){
             icl[kr-1, kc-1] <- object@icl
            }
    }

}

icl

> icl
          [,1]      [,2]      [,3]
[1,] -3529.423     0.000 -3503.235
[2,]     0.000 -3373.573     0.000
[3,]     0.000 -3361.628 -3299.497
```

We note that the co-clustering algorithm could not find a solution without empty cluster for `(kr,kc) = (2,3),(3,2),(3,4),(4,2)`. The higest ICL-BIC is obtained when `(kr,kc) = (3,3)`.

Handling different numbers of levels

The following code shows how to handle different numbers of levels, in a co-clustering context. It may take several minutes, due to the high number of levels of the two last columns.

```
set.seed(1)

library(ordinalClust)

# loading the real dataset
data("dataqol")

# loading the ordinal data
x <- as.matrix(dataqol[,2:31])


# defining different number of categories:
m <- c(4,7)


# defining number of row and column clusters
krow <- 3
kcol <- c(3,1)

# configuration for the inference
nbSEM <- 20
nbSEMburn <- 15
nbindmini <- 2
init <- 'random'
```

```
d.list <- c(1,29)

# Co-clustering execution
object <- boscoclust(x = x,kr = krow, kc = kcol, m = m,
                idx_list = d.list, nbSEM = nbSEM,
                nbSEMburn = nbSEMburn, nbindmini = nbindmini,
                init = init)
```

Margot Selosse
Université de Lyon, Lyon 2, ERIC EA 3083.
5 Avenue Pierre Mendés France, 69500 Bron
France
margot.selosse@gmail.com

Julien Jacques
Université de Lyon, Lyon 2, ERIC EA 3083.
5 Avenue Pierre Mendés France, 69500 Bron
France
julien.jacques@univ-lyon2.fr

Christophe Biernacki
Inria, Université de Lille, CNRS Université Lille - UFR de Mathématiques - Cité Scientifique - 59655 Villeneuve d'Ascq Cedex
France
christophe.biernacki@inria.fr