Journal of Statistical Software

February 2017, Volume 76, Issue 9. *doi: 10.18637/jss.v076.i09*

blockcluster: An R Package for Model-Based Co-Clustering

Parmeet Singh Bhatia
Siemens Healthineers

Serge Iovleff
Univ. Lille 1, CNRS, Inria

Gérard Govaert
UTC, CNRS

Abstract

Simultaneous clustering of rows and columns, usually designated by bi-clustering, co-clustering or block clustering, is an important technique in two way data analysis. A new standard and efficient approach has been recently proposed based on the latent block model (Govaert and Nadif 2003) which takes into account the block clustering problem on both the individual and variable sets. This article presents our R package **blockcluster** for co-clustering of binary, contingency and continuous data based on these very models. In this document, we will give a brief review of the model-based block clustering methods, and we will show how the R package **blockcluster** can be used for co-clustering.

Keywords: model-based clustering, block mixture model, EM and CEM algorithms, simultaneous clustering, co-clustering, R, **blockcluster**.

1. Introduction

Cluster analysis is an important tool in a variety of scientific areas such as pattern recognition, information retrieval, micro-array, data mining, and so forth. Although many clustering procedures such as hierarchical clustering, k-means or self-organizing maps, aim to construct an optimal partition of objects or, sometimes, of variables, there are other methods, called block clustering methods, which consider simultaneously the two sets and organize the data into homogeneous blocks. Let $\mathbf{x}$ denotes a $n \times d$ data matrix defined by $\mathbf{x} = \{(x_{ij}); i \in I \text{ and } j \in J\}$, where I is a set of n objects (rows, observations, cases, etc.) and J is a set of d variables (columns, attributes, etc.). The basic idea of these methods consists in making permutations of objects and variables in order to draw a correspondence structure on $I \times J$. For illustration, consider Figure 1 where a binary data set defined on set of $n = 10$ individuals $I = \{A, B, C, D, E, F, G, H, I, J\}$ and set of $d = 7$ binary variables $J = \{1, 2, 3, 4, 5, 6, 7\}$ is re-organized into a set of 3×3 clusters by permuting the rows and columns.

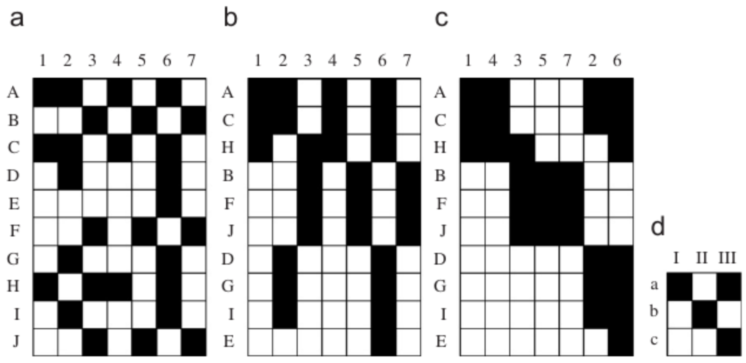


Figure 1: Binary data set (a), data reorganized by a partition on I (b), by partitions on I and J simultaneously (c) and summary matrix (d).

In recent years, co-clustering has found numerous applications in the fields ranging from data mining, information retrieval, biology, computer vision and so forth. Dhillon (2001) published an article on text data mining by simultaneously clustering the documents and content (words) using bipartite spectral graph partitioning. This is quite a useful technique for instance to manage a huge corpus of unlabeled documents. Xu, Zong, Dolog, and Zhang (2010) presents another co-clustering application (again using a bipartite spectral graph) to understand subset aggregates of web users by simultaneously clustering the users (sessions) and the page view information. Giannakidou, Koutsonikola, Vakali, and Kompatsiaris (2008) employed a similarity metric based co-clustering technique for a social tagging system. In the field of bioinformatics, co-clustering is mainly used to find structures in gene expression data. This is useful for instance to find a set of genes corresponding to a particular kind of disease. Some of the pioneer material in this context can be found in Kluger, Basri, Chang, and Gerstein (2003). Recently many co-clustering based algorithms have also been developed to target computer vision applications. For instance, Qiu (2004) demonstrated the utility of co-clustering in image grouping by simultaneously clustering images with their low-level visual features. Guan, Qiu, and Xue (2005) extended this work and presented opportunity to develop a novel content based image retrieval system. Similarly, Rasiwasia and Vasconcelos (2009) used co-clustering to model scenes. We have presented here only a very brief survey demonstrating the diversity of co-clustering applications. Unfortunately, many of the algorithms developed above for co-clustering are suitable in that particular context only and cannot be considered as a generic framework. Here, we take the opportunity to emphasize on the fact that the algorithms used in our software can be more or less adopted in most or all the above contexts.

In practice, several software packages exist for block clustering such as **Bicat** (Barkow, Bleuler, Prelic, Zimmermann, and Zitzler 2006), **biclust** (Kaiser and Leisch 2008), and **BiGGEsTS** (Goncalves, Madeira, and Oliveira 2009), or refer to Madeira and Oliveira (2004) for a review. In this paper, we introduce our R (R Core Team 2016) package **blockcluster** (Iovleff and Singh Bhatia 2017), which provides a bridge between a newly developed C++ library and the R statistical computing environment. The R package **blockcluster** allows to estimate the parameters of the co-clustering models (Govaert and Nadif 2003) for binary, contingency and continuous data. This package is unique from the point of view of generative models it implements (latent blocks), the used algorithms (BEM, BCEM) and, apart from that, special attention has been given to design the library for handling very huge data sets in reasonable time. The R package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=blockcluster`.

The rest of the article is organized as follows. In Section 2, we give the mathematical foundation (without going into detailed proofs) of latent block models for binary, contingency and continuous datasets. In Section 3, we elaborate various details of the **blockcluster** package and illustrate its usage with the help of several examples. This part can also act as a first-hand comprehensive tutorial for the audience who are interested in running the **blockcluster** package directly. Finally we illustrate our package on real datasets in Section 4 and terminate with some concluding remarks in Section 5.

2. Block mixture models

2.1. Mixture models: Classic formulation

In model-based clustering, it is assumed that the data arises from a mixture of underlying probability distributions, where each component k of the mixture represents a cluster. Hence the matrix data $\mathbf{x}=(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$ is assumed to be i.i.d. and generated from a probability distribution with the density

$$f(\boldsymbol{x}_i;\boldsymbol{\theta}) = \sum_k \pi_k f_k(\boldsymbol{x}_i;\boldsymbol{\alpha}),$$

where f_k is the density function for the kth component. Generally, these densities belong to the same parametric family (say for example Gaussian). The parameters $(\pi_k)_{k=1}^g$ give the probabilities that an observation belongs to the kth component and the number of components in the mixture is g which is assumed to be known. The parameter $\boldsymbol{\theta}$ of this model is the vector $(\pi_1,\ldots,\pi_g,\boldsymbol{\alpha})$ containing the parameter $\boldsymbol{\alpha}$ and the mixing proportions $(\pi_1,\ldots,\pi_g)$. Using this, the density of the observed data $\mathbf{x}$ can be expressed as

$$f(\mathbf{x};\boldsymbol{\theta}) = \prod_i \sum_k \pi_k f_k(\boldsymbol{x}_i;\boldsymbol{\alpha}). \tag{1}$$

An important property of the pdf $f(\mathbf{x};\boldsymbol{\theta})$ shown by Govaert and Nadif (2003) is that the density (1) can be rewritten as

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{\mathbf{z}\in\mathcal{Z}} p(\mathbf{z}) f(\mathbf{x}|\mathbf{z};\boldsymbol{\alpha}), \tag{2}$$

where $\mathcal{Z}$ denotes the set of all possible partitions of I in g clusters, $f(\mathbf{x}|\mathbf{z};\boldsymbol{\alpha}) = \prod_i f_{z_i}(\boldsymbol{x}_i;\boldsymbol{\alpha})$ and $p(\mathbf{z}) = \prod_i \pi_{z_i}$. With this formulation, the data matrix is assumed to be a sample of size 1 from a random (n,d) matrix.

2.2. Latent block model: General formulation

Let the dataset $\mathbf{x}$ be defined on a set I with n elements (individuals) and a set J with d elements (variables). We represent a partition of I into g clusters by $\mathbf{z} = (z_{11},\ldots,z_{ng})$ with $z_{ik} = 1$ if i belongs to cluster k and $z_{ik} = 0$ otherwise, $z_i = k$ if $z_{ik} = 1$ and we denote by $z_{.k} = \sum_i z_{ik}$ the cardinality of row cluster k. Similarly, we represent a partition of J into m clusters by $\mathbf{w} = (w_{11}, \dots, w_{dm})$ with $w_{j\ell} = 1$ if j belongs to cluster ℓ and $w_{j\ell} = 0$ otherwise, $w_j = \ell$ if $w_{j\ell} = 1$ and we denote $w_{.\ell} = \sum_j w_{j\ell}$ the cardinality of column cluster ℓ. Furthermore, note that for any variable V_{ij}, we denote $V_{.k}$ to represent $\sum_i V_{ik}$ and similarly $V_{i.}$ shall represent $\sum_k V_{ik}$.

To study the block clustering problem, the previous formulation (2) is extended to propose the block mixture model defined by the following probability density function

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{\mathbf{u}\in\mathcal{U}} p(\mathbf{u};\boldsymbol{\theta}) f(\mathbf{x}|\mathbf{u};\boldsymbol{\theta}),$$

where $\mathcal{U}$ denotes the set of all possible labelings of $I \times J$ and $\boldsymbol{\theta}$ contains all the unknown parameters of this model. By restricting this model to a set of labelings of $I \times J$ defined by a product of labelings of I and J, and further assuming that the labelings of I and J are independent of each other, we obtain the decomposition

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{(\mathbf{z},\mathbf{w})\in\mathcal{Z}\times\mathcal{W}} p(\mathbf{z};\boldsymbol{\theta}) p(\mathbf{w};\boldsymbol{\theta}) f(\mathbf{x}|\mathbf{z},\mathbf{w};\boldsymbol{\theta}), \tag{3}$$

where $\mathcal{Z}$ and $\mathcal{W}$ denote the sets of all possible labelings $\mathbf{z}$ of I and $\mathbf{w}$ of J.

By extending the latent class principle of local independence to our block model, each data point x_{ij} will be independent once z_i and w_j are fixed. Hence we have

$$f(\mathbf{x}|\mathbf{z},\mathbf{w};\boldsymbol{\theta}) = \prod_{i,j} f_{z_i w_j}(x_{ij};\boldsymbol{\alpha}),$$

where $f_{z_i w_j}(x,\boldsymbol{\alpha})$ is either a probability density function (pdf) defined on the real set $\mathbb{R}$, or a discrete probability function. Denoting $\boldsymbol{\theta} = (\boldsymbol{\pi},\boldsymbol{\rho},\boldsymbol{\alpha})$, where $\boldsymbol{\pi} = (\pi_1,\dots,\pi_g)$ and $\boldsymbol{\rho} = (\rho_1,\dots,\rho_m)$ are the vectors of probabilities π_k and ρ_ℓ that a row and a column belong to the kth row component and to the ℓth column component respectively, we obtain the latent block mixture model with pdf

$$f(\mathbf{x};\boldsymbol{\theta}) = \sum_{(\mathbf{z},\mathbf{w})\in\mathcal{Z}\times\mathcal{W}} \prod_{i,j} \pi_{z_i}\rho_{w_j} f_{z_i w_j}(x_{ij};\boldsymbol{\alpha}). \tag{4}$$

Using the above formulation, the randomized data generation process can be described by the three steps row labelings (R), column labelings (C), and data generation (X) as follows:

(R) Generate the labelings $\mathbf{z} = (z_1,\dots,z_n)$ according to the distribution $\boldsymbol{\pi} = (\pi_1,\dots,\pi_g)$.

(C) Generate the labelings $\mathbf{w} = (w_1,\dots,w_d)$ according to the distribution $\boldsymbol{\rho} = (\rho_1,\dots,\rho_m)$.

(X) Generate for $i = 1,\dots,n$ and $j = 1,\dots,d$ a value x_{ij} according to the (density) distribution $f_{z_i w_j}(.;\boldsymbol{\alpha})$.

2.3. Model parameter estimation

EM based algorithms are the standard approach to estimate model parameters by maximizing the observed data log-likelihood. In our case, the complete data is represented as a vector

$(\mathbf{x}, \mathbf{z}, \mathbf{w})$ where the unobservable vectors $\mathbf{z}$ and $\mathbf{w}$ are the labels. The *complete data log-likelihood* can then be written

$$L_C(\mathbf{z}, \mathbf{w}, \boldsymbol{\theta}) = \sum_k z_{.k} \log \pi_k + \sum_\ell w_{.\ell} \log \rho_\ell + \sum_{i,j,k,\ell} z_{ik} w_{j\ell} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}). \tag{5}$$

The EM algorithm maximizes the log-likelihood $L(\boldsymbol{\theta})$ iteratively by maximizing the conditional expectation $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(c)})$ of the complete data log-likelihood given a previous current estimate $\boldsymbol{\theta}^{(c)}$ and $\mathbf{x}$:

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(c)}) = \sum_{i,k} t_{ik}^{(c)} \log \pi_k + \sum_{j,\ell} r_{j\ell}^{(c)} \log \rho_\ell + \sum_{i,j,k,\ell} e_{ikj\ell}^{(c)} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}),$$

where

$$t_{ik}^{(c)} = P(z_{ik} = 1 | \mathbf{x}, \boldsymbol{\theta}^{(c)}), \qquad r_{j\ell}^{(c)} = P(w_{j\ell} = 1 | \mathbf{x}, \boldsymbol{\theta}^{(c)}), \qquad e_{ikj\ell}^{(c)} = P(z_{ik} w_{j\ell} = 1 | \mathbf{x}, \boldsymbol{\theta}^{(c)}).$$

Unfortunately, difficulties arise owing to the dependence structure in the model, and more precisely, to determinate $e_{ikj\ell}^{(c)}$. To solve this problem a new approximate solution is proposed (Govaert and Nadif 2003) using the Hathaway (1986) and Neal and Hinton (1998) interpretation of the EM algorithm. By constraining the probability p to verify the relation $p(z_i, w_j) = p(z_i)p(w_j)\, \forall i, j$, it can be shown that the fuzzy clustering criterion proposed by Neal and Hinton (1998) takes the form

$$\tilde{F}_C(\mathbf{t}, \mathbf{r}; \boldsymbol{\theta}) = L_C(\mathbf{t}, \mathbf{r}, \boldsymbol{\theta}) + H(\mathbf{t}) + H(\mathbf{r}) \tag{6}$$

in case of the latent block model, where

$$H(\mathbf{t}) = -\sum_{ik} t_{ik} \log t_{ik} \quad \text{and} \quad H(\mathbf{r}) = -\sum_{j\ell} r_{j\ell} \log r_{j\ell}$$

and L_c is the fuzzy complete-data log-likelihood associated to the block latent model given by

$$L_C(\mathbf{t}, \mathbf{r}; \boldsymbol{\theta}) = \sum_k t_{.k} \log \pi_k + \sum_{.\ell} r_\ell \log \rho_\ell + \sum_{i,j,k,\ell} t_{ik} r_{j\ell} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}). \tag{7}$$

This is a variational approach and its objective is to maximize the function $\tilde{F}_C$. Doing that, the maximization of the likelihood is replaced by the maximization of an approximation of this likelihood defined by

$$\tilde{L}(\boldsymbol{\theta}) = \operatorname*{argmax}_{\mathbf{t},\mathbf{r}} \tilde{F}_C(\mathbf{t}, \mathbf{r}, \boldsymbol{\theta}).$$

To simplify some notations, we will also use the *restricted complete data log-likelihood*

$$L_{\mathrm{CR}}(\mathbf{z}, \mathbf{w}, \boldsymbol{\theta}) = \sum_{i,j,k,\ell} z_{ik} w_{j\ell} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}) \tag{8}$$

and the *restricted fuzzy complete-data log-likelihood*

$$L_{\mathrm{CR}}(\mathbf{t}, \mathbf{r}; \boldsymbol{\theta}) = \sum_{i,j,k,\ell} t_{ik} r_{j\ell} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}). \tag{9}$$

2.4. Algorithms

Here we give a brief outline of two algorithms used in our package.

Block expectation maximization (BEM) algorithm

The fuzzy clustering criterion given in (6) can be maximized using the EM algorithm. We here only outline the various expressions evaluated during the E and M steps.

E-Step: Here we compute conditional row and column class probabilities which are respectively given by:

$$t_{ik} = \log \pi_k + \sum_{j\ell} r_{j\ell} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}) \tag{10}$$

and

$$r_{jl} = \log \rho_\ell + \sum_{i,k} t_{ik} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}). \tag{11}$$

M-Step: Here we calculate row proportions $\boldsymbol{\pi}$ and column proportions $\boldsymbol{\rho}$: The maximization of $\tilde{F}_C$ w.r.t. $\boldsymbol{\pi}$, and w.r.t. $\boldsymbol{\rho}$, is obtained by maximizing $\sum_k t_{.k} \log \pi_k$, and $\sum_\ell r_{.\ell} \log \rho_\ell$ respectively, which leads to $\pi_k = \frac{t_{.k}}{n}$ and $\rho_\ell = \frac{r_{.\ell}}{d}$. Also the estimate of model parameters $\boldsymbol{\alpha}$ will be obtained by maximizing

$$\sum_{i,j,k,\ell} t_{ik} r_{j\ell} \log f_{k\ell}(x_{ij}; \boldsymbol{\alpha}), \tag{12}$$

which will depend on the pdf's $f_{k\ell}$ and shall be treated later separately in subsections discussing block mixture models for the binary, contingency and continuous case.

Using the *E* and *M* steps defined above, the *BEM* algorithm can be enumerated as follows:

1. Initialize $\mathbf{t}^{(0)}, \mathbf{r}^{(0)}$ and $\boldsymbol{\theta}^{(0)} = (\boldsymbol{\pi}^{(0)}, \boldsymbol{\rho}^{(0)}, \boldsymbol{\alpha}^{(0)})$.
2. Compute $\mathbf{t}^{(c+1)}, \boldsymbol{\pi}^{(c+1)}, \boldsymbol{\alpha}^{(c+(1/2))}$ by using the EM algorithm for the data matrix $u_{i\ell} = \sum_j r_{j\ell} x_{ij}$ and starting from $\mathbf{t}^{(c)}, \boldsymbol{\pi}^{(c)}, \boldsymbol{\alpha}^{(c)}$.
3. Compute $\mathbf{r}^{(c+1)}, \boldsymbol{\rho}^{(c+1)}, \boldsymbol{\alpha}^{(c+1)}$ by using the EM algorithm for the data matrix $v_{jk} = \sum_i t_{ik} x_{ij}$ and starting from $\mathbf{r}^{(c)}, \boldsymbol{\rho}^{(c)}, \boldsymbol{\alpha}^{(c+(1/2))}$.
4. Iterate steps (2) and (3) until convergence.

Block classification expectation maximization (BCEM) algorithm

To apply the complete maximum likelihood (CML) approach for parameter estimation of the latent block model (4), the partitions $\mathbf{z}$ and $\mathbf{w}$ are added to the parameters to be estimated, and the objective is to maximize the complete data log-likelihood associated with the latent block model (4). Unlike the BEM situation, this maximization does not require an approximation and can be done, for instance, by maximizing with fixed $\mathbf{z}$ and $\boldsymbol{\rho}$ and then with fixed $\mathbf{w}$ and $\boldsymbol{\pi}$. This algorithm can be seen as a variant of the *BEM* algorithm described above and to perform this algorithm, it is sufficient to add a C-step in each of the EM algorithms, which converts the t_{ik}'s and r_{jl}'s to a discrete classification before performing the M-step by assigning each individual and variable to the cluster having the highest posterior probability.

2.5. Block mixture models for binary, contingency and continuous datasets

Binary data

The parameters $\boldsymbol{\alpha}$ of the underlying distribution of a binary data set is given by the matrix $\mathbf{p} = (p_{k\ell})$ where $p_{k\ell} \in]0,1[\ \forall\ k = 1,\ldots,g$ and $\ell = 1,\ldots,m$ and the probability distribution $f_{k\ell}(x_{ij};\mathbf{p}) = f(x_{ij};p_{k\ell})$ is the Bernoulli distribution

$$f(x_{ij};p_{k\ell}) = (p_{k\ell})^{x_{ij}}(1-p_{k\ell})^{1-x_{ij}}.$$

We re-parameterize the model density as follows:

$$f_{k\ell}(x_{ij};\boldsymbol{\alpha}) = (\varepsilon_{kj})^{|x_{ij}-a_{k\ell}|}(1-\varepsilon_{kj})^{1-|x_{ij}-a_{k\ell}|},$$

where

$$\begin{cases} a_{k\ell}=0,\ \varepsilon_{k\ell}=p_{k\ell} & \text{if } p_{k\ell}\leqslant 0.5 \\ a_{k\ell}=1,\ \varepsilon_{k\ell}=1-p_{k\ell} & \text{if } p_{k\ell}>0.5. \end{cases}$$

Hence the parameters $p_{k\ell}$ of the Bernoulli mixture model are replaced by the following parameters:

- The binary value $a_{k\ell}$, which acts as the center of the block k,ℓ and which gives, for each block, the most frequent binary value,
- The value $\varepsilon_{k\ell}$ belonging to the set $]0,1/2]$ that characterizes the dispersion of the block k,ℓ and which, for each block, represents the probability of having a different value than the center.

For this model, using Equations 10 and 11 the row and column class conditional probabilities are respectively given by:

$$t_{ik} = \log\pi_k + \sum_\ell |u_{i\ell} - r_{.\ell}a_{k\ell}| \log\frac{\varepsilon_{k\ell}}{1-\varepsilon_{k\ell}} + \sum_\ell r_{.\ell}\log(1-\varepsilon_{k\ell})$$

and

$$r_{j\ell} = \log\rho_\ell + \sum_k |v_{jk} - t_{.k}a_{k\ell}| \log\frac{\varepsilon_{k\ell}}{1-\varepsilon_{k\ell}} + \sum_k t_{.k}\log(1-\varepsilon_{k\ell}),$$

where $u_{i\ell} = \sum_j r_{j\ell}x_{ij}$ and $v_{jk} = \sum_i t_{ik}x_{ij}$.

Furthermore, we can write the restrained data log-likelihood for this model in terms of previously defined explicit parameters as

$$L_{\text{CR}}(\mathbf{t},\mathbf{r},\boldsymbol{\alpha}) = \sum_{k\ell}\left(|y_{k\ell} - t_{.k}r_{.\ell}a_{k\ell}|\log\frac{\varepsilon_{k\ell}}{1-\varepsilon_{k\ell}} + t_{.k}r_{.\ell}\log(1-\varepsilon_{k\ell})\right), \tag{13}$$

where $y_{k\ell} = \sum_{i,j} t_{ik}r_{j\ell}x_{ij}$. It can be easily shown that the values of model parameters maximizing Equation 13 is given by

$$\begin{cases} a_{k\ell}=0,\ \varepsilon_{k\ell}=\frac{y_{k\ell}}{t_{.k}r_{.\ell}} & \text{if } \frac{y_{k\ell}}{t_{.k}r_{.\ell}} < 0.5 \\ a_{k\ell}=1,\ \varepsilon_{k\ell}=1-\frac{y_{k\ell}}{t_{.k}r_{.\ell}} & \text{otherwise.} \end{cases}$$

For a detailed discussion on Bernoulli latent block models we would kindly refer our readers to Govaert and Nadif (2008).

Contingency data

To apply the block latent mixture model on contingency data, it is assumed that for each block k, ℓ, the values x_{ij} are distributed according to a Poisson distribution $\mathcal{P}(\mu_i \nu_j \gamma_{k\ell})$ where the Poisson parameter is split into μ_i and ν_j, the effects of the row i and the column j respectively, and $\gamma_{k\ell}$, the effect of the block $k\ell$ (Govaert and Nadif 2010). Then, we have

$$f_{k\ell}(x_{ij};\boldsymbol{\alpha}) = \frac{e^{-\mu_i\nu_j\gamma_{k\ell}}(\mu_i\nu_j\gamma_{k\ell})^{x_{ij}}}{x_{ij}!},$$

where $\boldsymbol{\alpha} = (\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\gamma})$ with $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$, $\boldsymbol{\nu} = (\boldsymbol{\nu}_1, \ldots, \nu_d)$ and $\boldsymbol{\gamma} = (\gamma_{11}, \ldots, \gamma_{gm})$. Unfortunately, this parameterization is not identifiable. It is therefore not possible to estimate simultaneously μ_i, ν_j and $\gamma_{k\ell}$. The following two solutions are considered to tackle this problem:

- μ_i and ν_j are known: In this case, it only remains to estimate the matrix $\gamma_{k\ell}$.
- Constraints are imposed on the parameter $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\rho}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\gamma})$: In this situation, the following constraints

$$\sum_k \pi_k \gamma_{kl} = \sum_\ell \rho_\ell \gamma_{kl} = 1, \quad \sum_i \mu_i = 1, \sum_j \nu_j = 1 \tag{14}$$

appear to make the model identifiable, but this remains yet to be proved.

The transformation of the parameter $\boldsymbol{\theta}$ on a parameter $\boldsymbol{\theta}'$ defined as follows

$$\pi'_k = \pi_k \quad \text{and} \quad \rho'_\ell = \rho_\ell$$
$$\mu'_i = \mu_i N \quad \text{and} \quad \nu'_j = \nu_j N$$
$$\gamma'_{k\ell} = \frac{\gamma_{k\ell}}{N},$$

shows that it is possible to parameterize the model with the constraints

$$\sum_k \pi'_k \gamma'_{kl} = \sum_\ell \rho'_\ell \gamma'_{kl} = 1/N, \quad \sum_i \mu'_i = \sum_j \nu'_j = N, \tag{15}$$

where N is any positive real.

Case-1: *The row and column effects are known.*

In this situation, $\boldsymbol{\alpha} = \boldsymbol{\gamma} = (\gamma_{11}, \ldots, \gamma_{gm})$ and $f_{k\ell}(x_{ij};\boldsymbol{\gamma})$ can be written as

$$f_{k\ell}(x_{ij};\boldsymbol{\gamma}) = K_{ij}.g_{k\ell}(x_{ij};\boldsymbol{\gamma}),$$

where

$$g_{k\ell}(x_{ij};\boldsymbol{\gamma}) = e^{-\mu_i\nu_j\gamma_{k\ell}}(\gamma_{k\ell})^{x_{ij}} \quad \text{and} \quad K_{ij} = \frac{(\mu_i\nu_j)^{x_{ij}}}{x_{ij}!}.$$

Then, we can replace the function $f_{k\ell}$ by the function $g_{k\ell}$ in the log-likelihoods and in the expressions t_{ik} and $r_{j\ell}$. Hence, using Equations 10 and 11 the row and column class conditional probabilities are respectively given by:

$$t_{ik} = \log \pi_k + \sum_{j\ell} r_{j\ell} x_{ij} \log \gamma_{k\ell} - \mu_i \sum_{\ell} \gamma_{k\ell}\nu_\ell,$$
$$r_{j\ell} = \log \rho_\ell + \sum_{i,k} t_{ik} x_{ij} \log \gamma_{k\ell} - \nu_j \sum_{k} \gamma_{k\ell}\mu_k.$$

Also the restrained complete data log-likelihood is given by:

$$L_{\mathrm{CR}}(\mathbf{t}, \mathbf{r}, \boldsymbol{\alpha}) = \sum_{k,\ell} y_{k\ell} \log \gamma_{k\ell} - \mu_k \nu_\ell \gamma_{k\ell}, \tag{16}$$

where $y_{k\ell} = \sum_{ij} t_{ik} r_{j\ell} x_{ij}$. The maximization of Equation 16 gives:

$$\gamma_{k\ell} = \frac{y_{k\ell}}{\mu_k \nu_\ell} \quad \forall k, \ell.$$

Case-2: *The row and column effects are unknown.*

Using the constraints given by Equation 15, we obtain $\mathsf{E}(x_{i.}) = \mu_i$ and $\mathsf{E}(x_{.j}) = \nu_j$. We can propose as an estimator of μ_i and ν_j margins $x_{i.}$ and $x_{.j}$ and take the results of *Case-1* to calculate row and column class conditional probabilities by replacing μ_i and ν_j by $x_{i.}$ and $x_{.j}$ respectively. Hence the row and column class conditional probabilities are respectively given by:

$$t_{ik} = \log \pi_k + \sum_{j\ell} r_{j\ell} x_{ij} \log \gamma_{k\ell} - \boldsymbol{x}_{i.} \sum_{\ell} \gamma_{k\ell} y_{.\ell},$$
$$r_{j\ell} = \log \rho_\ell + \sum_{i,k} t_{ik} x_{ij} \log \gamma_{k\ell} - \boldsymbol{x}_{.j} \sum_{k} \gamma_{k\ell} y_{k.}.$$

Furthermore the derivation of $\gamma_{k\ell}$ is exactly the same as in the previous case. Using this particular form of $\gamma_{k\ell}$, the calculation of t_{ik} and $r_{j\ell}$ is simplified and becomes

$$t_{ik} = \log \pi_k + \sum_{j\ell} r_{j\ell} x_{ij} \log \gamma_{k\ell} - \boldsymbol{x}_{i.},$$
$$r_{j\ell} = \log \rho_\ell + \sum_{i,k} t_{ik} x_{ij} \log \gamma_{k\ell} - \boldsymbol{x}_{.j}.$$

Continuous data

In this case, the continuous data is modeled using the unidimensional normal distribution. Hence the density for each block is given by:

$$f_{k\ell}(x_{ij}; \boldsymbol{\alpha}) = \frac{1}{\sqrt{2\pi\sigma_{k\ell}^2}} \exp - \left\{ \frac{1}{2\sigma_{k\ell}^2}(x_{ij} - \mu_{k\ell})^2 \right\}.$$

The parameters of the model are $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_{11}, \ldots, \boldsymbol{\alpha}_{gm})$, where $\boldsymbol{\alpha}_{k\ell} = (\mu_{k\ell}, \sigma^2_{k\ell})$. For this model, using Equations 10 and 11 the row and column class conditional probabilities are respectively given by:

$$t_{ik} = \log \pi_k - \frac{1}{2} \sum_{\ell} (r_{.\ell} \log \sigma^2_{k\ell} + \frac{1}{\sigma^2_{k\ell}} a_{ik\ell}),$$

where

$$a_{ik\ell} = \sum_j r_{j\ell}(x_{ij} - u_{i\ell}/r_{.\ell})^2 + r_{.\ell} \sum_{\ell} (\frac{u_{i\ell}}{r_{.\ell}} - \mu_{k\ell})^2$$

and

$$r_{j\ell} = \log \rho_\ell - \frac{1}{2} \sum_k (t_{.k} \log \sigma^2_{k\ell} + \frac{1}{\sigma^2_{k\ell}} b_{jk\ell}),$$

where

$$b_{jk\ell} = \sum_i t_{ik}(x_{ij} - v_{jk}/t_{.k})^2 + t_{.k} \sum_k (v_{jk} - \mu_{k\ell})^2.$$

The computation of parameters $\boldsymbol{\alpha}$ is given by maximization of restrained log-likelihood which in this case is given by:

$$L_{\mathrm{CR}}(\mathbf{t}, \mathbf{r}, \boldsymbol{\alpha}) = -\frac{1}{2} \sum_{k,\ell} (t_{.k} r_{.\ell} \log \sigma^2_{k\ell} + \frac{1}{\sigma^2_{k\ell}} \sum_{i,j} t_{ik} r_{j\ell} (x_{ij} - \mu_{k\ell})^2). \tag{17}$$

It can be deduced easily that the value of $\mu_{k\ell}$ and $\sigma^2_{k\ell}$ is given by minimization of $\sum_{ij} t_{ik} r_{j\ell} (x_{ij} - \mu_{k\ell})^2$ and $t_{.k} r_{.\ell} \log \sigma^2_{k\ell} + \frac{1}{\sigma^2_{k\ell}} \sum_{ij} t_{ik} r_{j\ell} (x_{ij} - \mu_{k\ell})^2 \quad \forall k, \ell$ respectively.

Hence we obtain:

$$\mu_{k\ell} = \frac{\sum_j r_{j\ell} v_{jk}}{t_{.k} r_{.\ell}}, \qquad \sigma^2_{k\ell} = \frac{\sum_{ij} t_{ik} r_{j\ell} x^2_{ij}}{t_{.k} r_{.\ell}} - \mu^2_{k\ell}.$$

3. Block clustering with package blockcluster

blockcluster is a newly developed R package for co-clustering of binary, contingency and continuous data. The core library is written in C++ and the **blockcluster** API acts as a bridge between the C++ core library and the R statistical computing environment. A brief introduction of the C++ library is given in Appendix A. In the subsequent sections, we shall provide detailed explanation of the various functionalities and utilities of the package with help of examples wherever possible.

3.1. Strategy for parameters estimation

Correct parameters estimation in EM based algorithms has always been a challenging problem due to various local optima. In case of co-clustering algorithms, this problem could be even more apparent and hence the final results shall be directly affected by the choice of a strategy. As being explained in previous sections, we have two block clustering algorithms namely *BEM* and *BCEM* and the way we run these algorithms in our library is named respectively as `"XEMStrategy"` and `"XCEMStrategy"`. Both the strategies are equivalent except that they run different algorithms. The various steps involved in `"XEMStrategy"` (equivalently in `"XCEMStrategy"`) are:

1. Set the pseudo log-likelihood $\mathbf{L}_{\text{XEM}}$ to minimum possible value.
2. Run the *BEM* algorithm for some pre-defined number of times (we denote this number by `nbxem` in our package) as follows:
   (a) Set the pseudo log-likelihood $\mathbf{L}_{\text{xem}}$ to minimum possible value.
   (b) Initialize the model parameters.
   (c) Run the algorithm with a high tolerance (epsilon) and few iterations using the initialized parameters, and calculate the current pseudo log-likelihood $\mathbf{L}_{\text{current}}$.
   (d) If $\mathbf{L}_{\text{current}}$ is greater than $\mathbf{L}_{\text{xem}}$, copy the currently estimated parameters to $\boldsymbol{\alpha}_{\text{start}}$ and set $\mathbf{L}_{\text{xem}}$ to $\mathbf{L}_{\text{current}}$.
3. Starting with the parameters $\boldsymbol{\alpha}_{\text{start}}$, run the algorithm with a low value of epsilon (low tolerance) and a high number of iterations and calculate the current pseudo log-likelihood $\mathbf{L}_{\text{current}}$.
4. If $\mathbf{L}_{\text{current}}$ is greater than $\mathbf{L}_{\text{XEM}}$, copy the currently estimated parameters to $\boldsymbol{\alpha}_{\text{final}}$ and set $\mathbf{L}_{\text{XEM}}$ to $\mathbf{L}_{\text{current}}$.
5. Repeat Steps 2 through 3 for some pre-set times (we denote this number by `nbtry` in our package) in order to obtain a good estimation of the parameters.

Although the above strategy does not explore all the possible solutions and hence cannot ensure to obtain the global optimum, it is still a very promising approach in practice. The tuning of the values of `nbxem` and `nbtry` needs to be done intuitively, and could have a substantial effect on the final results. A good way to set these values is to run co-clustering a few number of times and check if the final log-likelihood is stable. If not, one may need to increase these values. In practice, it is better to increment `nbxem` as it could lead to better (stable) results without compromising too much the running time.

In the package **blockcluster**, we have a function called `coclusterstrategy`[1] which allows to set the values of various input parameters including `nbxem` and `nbtry`. In the following example, we get an object `defaultstrategy` of S4 class '`strategy`' by calling the function `coclusterstrategy` without any arguments and then we called the overloaded function `summary` to see various default values.

```
R> defaultstrategy <- coclusterStrategy()
R> summary(defaultstrategy)

****************************************************************
Algorithm:  XEMStrategy
Initialization method(There is no default value):
Stopping Criteria:  Parameter

Various Iterations
```

[1]This function is also used to set all other input parameters like the number of iterations and tolerance values. They are not described here explicitly for the sake of brevity. We refer our readers to the package documentation for details.

```
******************
Number of global iterations while running initialization:  10
Number of iterations for internal E-step:  5
Number of EM iterations used during xem:  50
Number of EM iterations used during XEM:  500
Number of xem iterations:  5
Number of tries:  2

Various epsilons
****************
Tolerance value used while initialization:  0.01
Tolerance value for internal E-step:  0.01
Tolerance value used during xem:  1e-04
Tolerance value used during XEM:  1e-10
******************************************************************
```

One thing which is worth noting in the summary output (above) is that there is no default value for the initialization method. It will be set automatically depending on the type of input model as explained in Section 3.2. Any of these input parameters can be set by passing appropriate arguments to function `coclusterStrategy` as shown in the example below where we set the `nbtry`, `nbxem` and `algo` parameters.

```
R> newstrategy <- coclusterStrategy(nbtry = 5, nbxem = 10,
+    algo = "XCEMStrategy")
```

3.2. Model initialization

Initializing model parameters is arguably the most important step in getting a good final estimate of model parameters. The three most important initialization strategies for EM algorithms are:

- *Random:* In this method, the model parameters are initialized randomly using the input data. One may go with several random initializations and choose the one which gives the best log-likelihood.

- *CEM:* The model parameters are first initialized using random values by making use of the input data, and then the CEM algorithm (Celeux and Govaert 1992) is run for a small number of iterations with high tolerance. The *CEM* algorithm optimizes directly the completed log-likelihood by replacing the M step by a classification step based on the MAP. In general, the *CEM* algorithm converges very quickly (not necessarily to a good optimum) which is an advantage as we spend few time in initialization compared to the overall run time.

- *FuzzyCEM (Stochastic EM):* This method is equivalent to *CEM* except for the fact that instead of classifying samples into various clusters based on MAP, we stochastically select the class for each sample using the corresponding class probability distribution.

Model	Datatype	Proportions	Dispersion/Variance	Initialization
`"pik_rhol_epsilonkl"`	binary	unequal	unequal	CEM
`"pik_rhol_epsilon"`	binary	unequal	equal	CEM
`"pi_rho_epsilonkl"`	binary	equal	unequal	CEM
`"pi_rho_epsilon"`	binary	equal	equal	CEM
`"pik_rhol_sigma2kl"`	continuous	unequal	unequal	CEM
`"pik_rhol_sigma"`	continuous	unequal	equal	CEM
`"pi_rho_sigma2kl"`	continuous	equal	unequal	CEM
`"pi_rho_sigma2"`	continuous	equal	equal	CEM
`"pik_rhol_unknown"`	contingency	unequal	N.A.	CEM
`"pi_rho_unknown"`	contingency	equal	N.A.	CEM
`"pik_rhol_known"`	contingency	unequal	N.A.	Random
`"pi_rho_known"`	contingency	equal	N.A.	Random

Table 1: Various models available in package **blockcluster**.

$\mathbf{a}$, $\boldsymbol{\epsilon}$	0, 0.1	0, 0.3	1, 0.1
	1, 0.3	1, 0.2	0, 0.1

$\boldsymbol{\pi}$	0.6	0.4	
$\boldsymbol{\rho}$	0.3	0.3	0.4

Table 2: Parameters for simulation of binary data.

Unfortunately, in **blockcluster**, we do not have the freedom to initialize any model with all of the methods above. Instead, for every model we only have one possible initialization method at present (mainly *CEM*) as shown in Table 1. Hence the initialization strategy will be selected automatically by the package based on the input model. This is the reason why we have no default initialization strategy. This is indeed on the priority list of our future works to provide as many initialization strategies as possible for various kinds of models.

3.3. Examples with simulated datasets

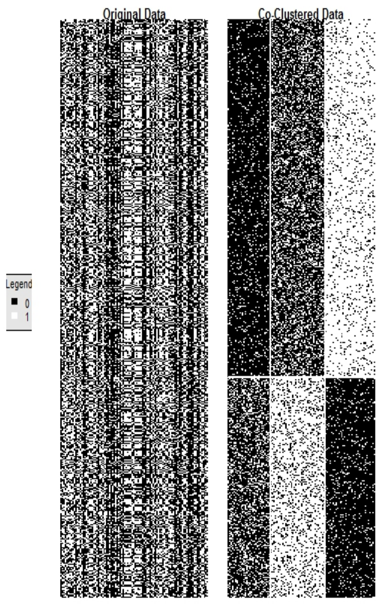
Binary data

We have simulated binary data using the data generation process explained in Section 2 and with parameters given in Table 2. The data consist of 1000 rows (samples) and 100 columns (variables) with two clusters on rows and three clusters on columns.

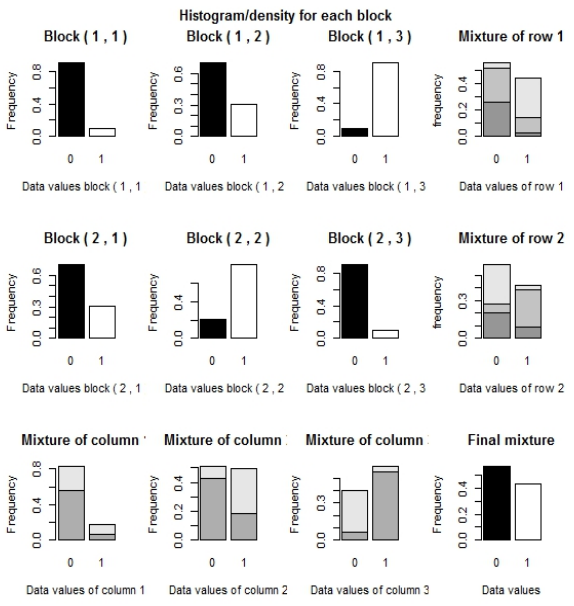
The following R commands shows how to load the package, process the data and visualize/summarize results using **blockcluster**.

```
R> library("blockcluster")
R> data("binarydata", package = "blockcluster")
R> out <- cocluster(binarydata, datatype = "binary", nbcocluster = c(2, 3))
R> summary(out)
```

```
******************************************************************
Model Family : Bernoulli Latent block model
Model Name : pik_rhol_epsilonkl
Co-Clustering Type : Unsupervised
```



(a)



(b)

Figure 2: Original and co-clustered binary data (a), and distributions for each block along with various mixture densities (b).

```
ICL value:  -45557.07

Model Parameters..

Class Mean:
      [,1]  [,2]  [,3]
[1,] FALSE FALSE  TRUE
[2,] FALSE  TRUE FALSE

Class Dispersion:
          [,1]       [,2]      [,3]
[1,] 0.1011803 0.09798014 0.3022391
[2,] 0.1006314 0.30176927 0.2003679

Row proportions:  0.618 0.382
Column proportions:  0.34 0.29 0.37
Pseudo-likelihood:  -0.4552043
hyperparam:  1 1
*****************************************************************
```

The following R command is used to plot the original and coclustered data (Figure 2(a)) with aspect ratio set to false (it is true by default).

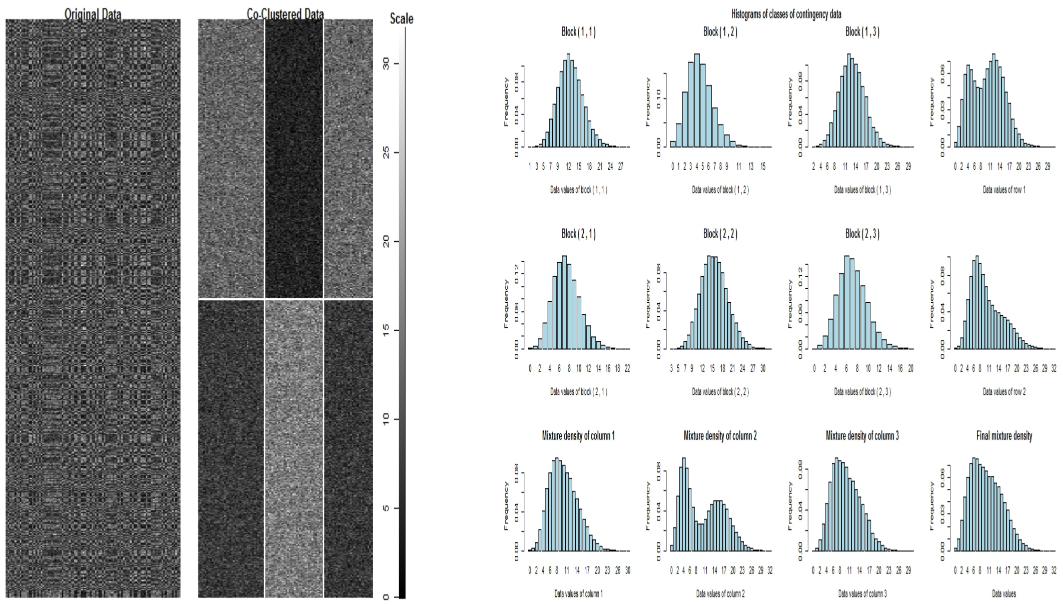
```
R> plot(out, asp = 0)
```

To plot the various block distributions (Figure 2(b)), the following R command is used with

γ(*1e-6)	1.290	0.450	1.260
	0.740	1.550	0.710

π	0.6	0.4	
ρ	0.4	0.3	0.3

Table 3: Parameters for simulation of contingency data.



(a) (b)

Figure 3: Original and co-clustered contingency data (a), and distributions for each block along with various mixture densities (b).

`type` set to `"distribution"` (`type` is `"cocluster"` by default).

```
R> plot(out, type = "distribution")
```

Contingency data

In this case, we simulated contingency data as explained earlier in Section 2 and using the parameters given in Table 3. Again the data dimensions are 1000 × 100 with two clusters on rows and three clusters on columns.

The following R commands load the simulated contingency data into the R environment, co-clusters the contingency data, summarizes the results and shows the graphics respectively.

```
R> data("contingencydataunknown", package = "blockcluster")
R> out <- cocluster(contingencydataunknown, datatype = "contingency",
+    nbcocluster = c(2, 3))
R> summary(out)
R> plot(out)
R> plot(out, type = "distribution")
```

```
******************************************************************
Model Family : Poisson Latent block model
```

$\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$	-10, 20	0, 10	10, 20
	10, 10	0, 20	-10, 10

$\boldsymbol{\pi}$	0.6	0.4	
$\boldsymbol{\rho}$	0.3	0.3	0.4

Table 4: Parameters for simulation of continuous data.

```
Model Name : pik_rhol_unknown
Co-Clustering Type : Unsupervised
ICL value:  1385292

Model Parameters..

Class Gamma:
          [,1]      [,2]     [,3]
[1,] 12.912520  4.490885 12.57979
[2,]  7.129589 15.535055  7.43389

Row proportions:  0.484 0.516
Column proportions:  0.3645123 0.34 0.2954877
Pseudo-likelihood:  1385335
******************************************************************
```

Continuous data

In this final example, we generated continuous data using parameters given in Table 4. The dataset again contains 1000 rows and 100 columns with two clusters on rows and three clusters on columns.

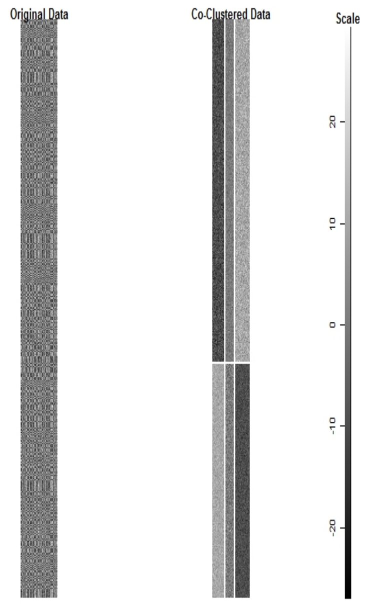
The following R commands performs co-clustering on the simulated continuous dataset and shows the results.

```
R> data("gaussiandata", package = "blockcluster")
R> out <- cocluster(gaussiandata, datatype = "continuous",
+    nbcocluster = c(2, 3))
R> summary(out)
R> plot(out, asp = 0)
R> plot(out, type = "distribution")
```

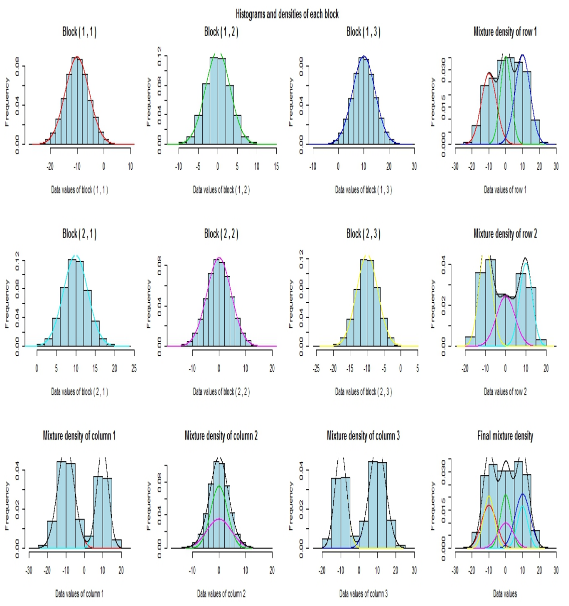
```
******************************************************************
Model Family : Gaussian Latent block model
Model Name : pik_rhol_sigma2kl
Co-Clustering Type : Unsupervised
ICL value:  -78.97326

Model Parameters..

Class Mean:
              [,1]      [,2]      [,3]
```



(a)



(b)

Figure 4: Original and co-clustered continuous data (a), and distributions for each block along with various mixture densities (b).

```
[1,] -0.0113171859  9.933649 -9.975354
[2,]  0.0008994489 -9.966977 10.031539

Class Variance:
         [,1]      [,2]     [,3]
[1,] 20.79533  9.854595 10.02491
[2,] 10.00552 19.958890 19.79939

Row proportions:  0.5 0.5
Column proportions:  0.28 0.32 0.4
Pseudo-likelihood:  -1.836655
*****************************************************************
```

4. Examples with real datasets

In what follows, we shall give the outcome of running package **blockcluster** on real datasets.

4.1. Image segmentation

Automatic image segmentation is an important technique and have numerous applications, especially in the field of medical imaging. Here we present an interesting application of co-clustering (as pre-processing step) for segmenting object(s) in an image. We assume that the object pixels follow a Gaussian distribution. Hence we ran the **blockcluster** package with the Gaussian family model `"pik_rhol_sigma2kl"`on the image shown in Figure 5. It can be

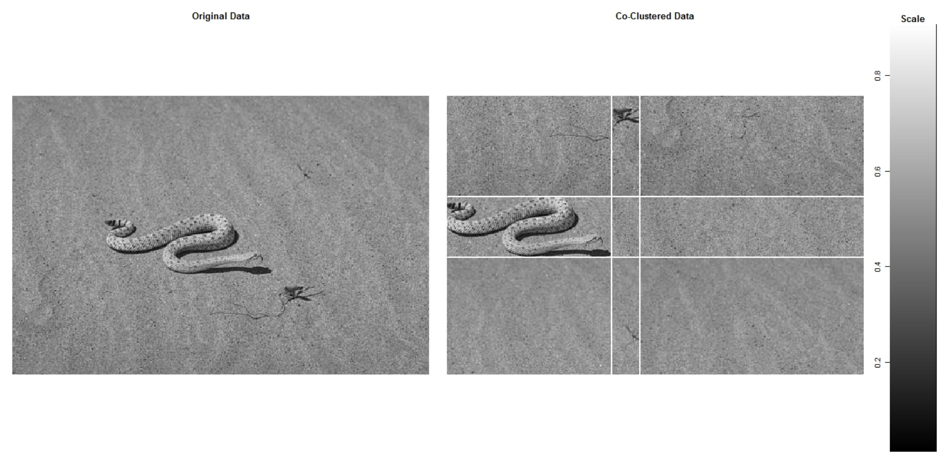


Figure 5: Original and co-clustered (segmented) image.

clearly seen that we are nicely able to segment the snake and insect in two different blocks. Co-clustering has not been exploited in this context so far to the best of our knowledge. Hence it seems this offers a new opportunity for co-clustering.

4.2. Document clustering

Document clustering is yet another data mining technique where co-clustering seems to be very useful as demonstrated in Dhillon (2001). Here we run our package on one of the datasets being used in Dhillon (2001) which is publicly available at `ftp://ftp.cs.cornell.edu/pub/smart`. We mix the datasets Medline (1033 medical abstracts) and Cranfield (1398 aeronautical abstracts) leading to a total of 2431 documents. Furthermore, we use all the words (excluding stop words) as features making a total of 9275 unique words. The data matrix consists of words on the rows and documents on the columns with each entry giving the term frequency, that is the number of occurrences of the corresponding word in the corresponding document. To run our package, we assume that the term frequency follows a Poisson distribution. Hence we can apply the model `"pik_rhol_unknown"` available in our package for contingency (Poisson family) datasets with unknown row and column effects. Table 5 shows the confusion matrix and compares our results with the classical bipartite spectral graph partitioning algorithm of Dhillon (2001) where we have obtained 100 percent correct classification.

Figure 6 depicts the 2×2 checkerboard pattern in the data matrix, hence confirming the more frequent occurrence of particular set of words in one document and vice-versa. Please note that the data matrix images are extremely sparse (data points almost invisible) and have been processed using simple image processing tools for the visualization purpose only.

5. Conclusion and future directions

This article presents our newly developed R package for co-clustering algorithms based on latent block models. In the first part of the article, we have presented a brief mathematical foundation of co-clustering algorithms based on latent block models. In the second part,

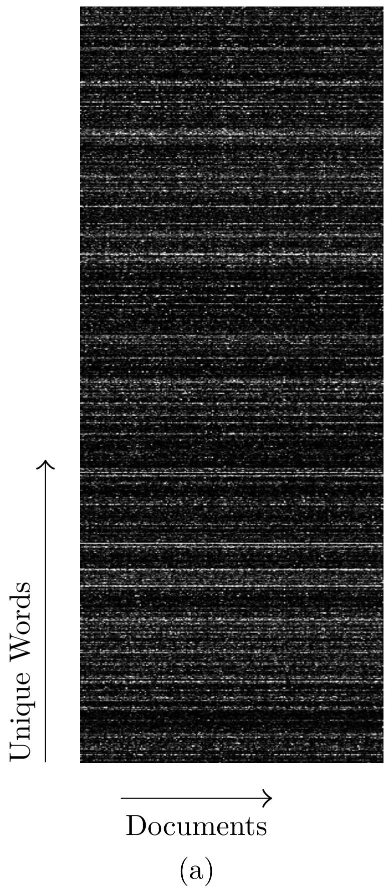
	Medline	Cranfield
Medline	1026	0
Cranfield	7	1400

(a)

	Medline	Cranfield
Medline	1033	0
Cranfield	0	1398

(b)

Table 5: Confusion matrix: Results reported in Dhillon (2001) (a), and results using **blockcluster** (b). The difference in number of Cranfield documents is because we made use of the ready-made data extracted from the documents and there are two documents less in this data.



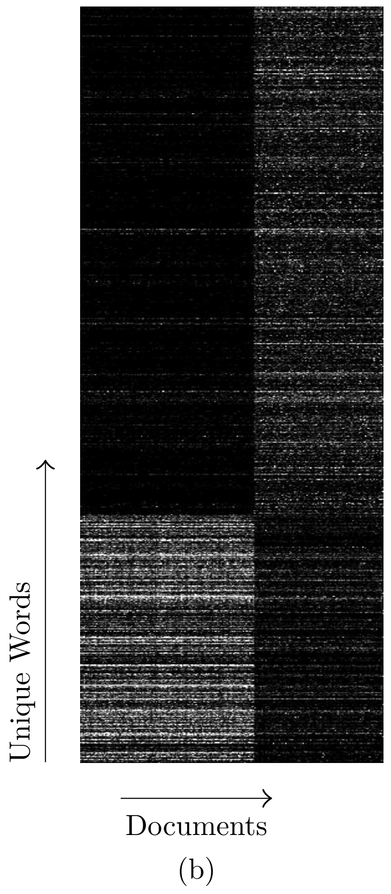


Figure 6: Original data matrix with words on rows and documents on columns (a), and checkerboard pattern in words by documents matrix obtained after performing co-clustering (b).

we have given details on the strategy adopted to run these algorithms. Furthermore, we have completed the presentation of the **blockcluster** package by providing the examples on simulated datasets for each data type. Finally we have shown applicability of our software on real datasets. As the package facilitates co-clustering for all types of data, it provides a generic framework and a one stop solution for a wide variety of co-clustering applications.

We are currently working on new initialization strategies as well as model selection criteria that should be part of upcoming releases of the package. The next release of the package will provide support for categorical models and a Gibbs EM algorithm (Keribin, Brault, Celeux, and Govaert 2015) for parameter estimation as well as implement semi-supervised co-clustering in which some row and/or column classes are provided by the user.

Acknowledgments

The authors are grateful to Christophe Biernacki for his leading support and constant motivation throughout the project. We are also thankful to Gilles Celeux for his contributions and providing his support in proposing the project plan to INRIA-Lille. Furthermore we are thankful to Alhousseiny Pam who worked partly on the graphical part of the R package. Finally we are grateful to INRIA-Lille for funding this project.

References

Barkow S, Bleuler S, Prelic A, Zimmermann P, Zitzler E (2006). "**BicAT**: A Biclustering Analysis Toolbox." *Bioinformatics*, **22**(10), 1282–1283. doi:10.1093/bioinformatics/btl099.

Celeux G, Govaert G (1992). "A Classification EM Algorithm for Clustering and Two Stochastic Versions." *Computational Statistics & Data Analysis*, **14**(3), 315–332. doi:10.1016/0167-9473(92)90042-e.

Dagum L, Enon R (1998). "**OpenMP**: An Industry Standard API for Shared-Memory Programming." *IEEE Computational Science & Engineering*, **5**(1), 46–55. doi:10.1109/99.660313.

Dhillon IS (2001). "Co-Clustering Documents and Words Using Bipartite Spectral Graph Partitioning." In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pp. 269–274. ACM, New York. doi:10.1145/502512.502550.

Freeman E, Robson E, Bates B, Sierra K (2004). *Head First Design Patterns*. O'Reilly.

Giannakidou E, Koutsonikola V, Vakali A, Kompatsiaris Y (2008). "Co-Clustering Tags and Social Data Sources." In *WAIM'08 – The Ninth International Conference on Web-Age Information Management*, pp. 317–324. IEEE.

Goncalves J, Madeira S, Oliveira A (2009). "**BiGGEsTS**: Integrated Environment for Biclustering Analysis of Time Series Gene Expression Data." *BMC Research Notes*, **2**, 124. doi:10.1186/1756-0500-2-124.

Govaert G, Nadif M (2003). "Clustering with Block Mixture Models." *Pattern Recognition*, **36**(2), 463–473. doi:10.1016/s0031-3203(02)00074-2.

Govaert G, Nadif M (2008). "Block Clustering with Bernoulli Mixture Models: Comparison of Different Approaches." *Computational Statistics & Data Analysis*, **52**(6), 3233–3245. doi:10.1016/j.csda.2007.09.007.

Govaert G, Nadif M (2010). "Latent Block Model for Contingency Table." *Communications in Statistics – Theory and Methods*, **39**(3), 416–425. doi:10.1080/03610920903140197.

Guan J, Qiu G, Xue XY (2005). "Spectral Images and Features Co-Clustering with Application to Content-Based Image Retrieval." In *2005 IEEE 7th Workshop on Multimedia Signal Processing*, pp. 1–4. IEEE.

Hathaway RJ (1986). "Another Interpretation of the EM Algorithm for Mixture Distributions." *Statistics & Probability Letters*, **4**(2), 53–56. doi:10.1016/0167-7152(86)90016-7.

Iovleff S, Singh Bhatia P (2017). ***blockcluster**: Coclustering Package for Binary, Categorical, Contingency and Continuous Data-Sets.* R package version 4.2.3, URL https://CRAN.R-project.org/package=blockcluster.

Kaiser S, Leisch F (2008). "A Toolbox for Bicluster Analysis in R." In P Brito (ed.), *COMPSTAT 2008 – Proceedings in Computational Statistics*, volume II, pp. 201–208. Physica Verlag, Heidelberg.

Keribin C, Brault V, Celeux G, Govaert G (2015). "Estimation and Selection for the Latent Block Model on Categorical Data." *Statistics and Computing*, **25**(6), 1201–1216. doi:10.1007/s11222-014-9472-2.

Kluger Y, Basri R, Chang JT, Gerstein M (2003). "Spectral Biclustering of Microarray Data: Coclustering Genes and Conditions." *Genome Research*, **13**(4), 703–716. doi:10.1101/gr.648603.

Madeira SC, Oliveira AL (2004). "Biclustering Algorithms for Biological Data Analysis: A Survey." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **1**(1), 24–45. doi:10.1109/tcbb.2004.2.

Neal RM, Hinton GE (1998). "A View of the EM Algorithm That Justifies Incremental, Sparse, and Other Variants." In *Learning in Graphical Models*, volume 89 of *NATO ASI Series D: Behavioural and Social Sciences*, pp. 355–368. Kluwer Academic Publishers. doi:10.1007/978-94-011-5014-9_12.

Qiu G (2004). "Image and Feature Co-Clustering." In *ICPR 2004 – Proceedings of the 17th International Conference on Pattern Recognition*, volume 4, pp. 991–994. IEEE.

Rasiwasia N, Vasconcelos N (2009). "Holistic Context Modeling Using Semantic Co-Occurrences." In *CVPR 2009 – IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1889–1895. IEEE.

R Core Team (2016). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Xu G, Zong Y, Dolog P, Zhang Y (2010). "Co-Clustering Analysis of Weblogs Using Bipartite Spectral Projection Approach." In R Setchi, I Jordanov, RJ Howlett, LC Jain (eds.), *Knowledge-Based and Intelligent Information and Engineering Systems*, volume 6278, pp. 398–407. Springer-Verlag. doi:10.1007/978-3-642-15393-8_45.

A. The core library: Structure and usage

In this paper we have given a detailed analysis on the usage of the R package **blockcluster** to estimate parameters of latent block models, but behind the scenes the kernel (computational) part lies in the C++ core library[2]. Our purpose to provide this section is only to briefly introduce the kernel part and partly for developers who might be interested in integrating the library with their own codes, but we highly recommend to use only the R package **blockcluster** for research purposes as lot of utilities are absent in the C++ codes, for instance the graphics part.

A.1. Library structure

The C++ library is based on the famous strategy design pattern (Freeman, Robson, Bates, and Sierra 2004) and the (not so complete) architecture is revealed in Figure 7. The choice of the architecture is motivated from the fact that it allows easy extension to the library (accommodating any future research) with as little changes as possible in the existing codes. Hence it follows the famous design principle "open for extension but closed for modification" (Freeman *et al.* 2004). Furthermore, we have used **OpenMP** (Dagum and Enon 1998) to provide parallel implementations of the software by exploiting various data and task parallelism in latent block model estimation. By default, the library shall make use of multiple cores on the machines that support **OpenMP**.

A.2. Usage

In our library, we have provided a simple main function (to illustrate its usage) that can be run from the console as follows.

```
$ coclustmain -f path/to/optionfile.txt
```

`coclustmain` is a binary executable file which takes the path to a text file as command line argument. A sample input text file is shown below.

```
[InputOptions]
DataType = Binary
nbrowclust = 2
nbcolclust = 3
DataFileName = Data/BinaryData/testbinarypikrholepskl.dat
nbinititerations = 10
initepsilon = .01
nbtry = 1
nbxem = 1
epsilon_xemstart = .00001
epsilon_xem = .00000001
nbiterations_xemstart = 50
nbiterations_xem = 50
epsilon_int = 0.01
```

[2]The source of the C++ library can be checkout anonymously from the repository `svn://scm.gforge.inria.fr/svnroot/cocluster`.

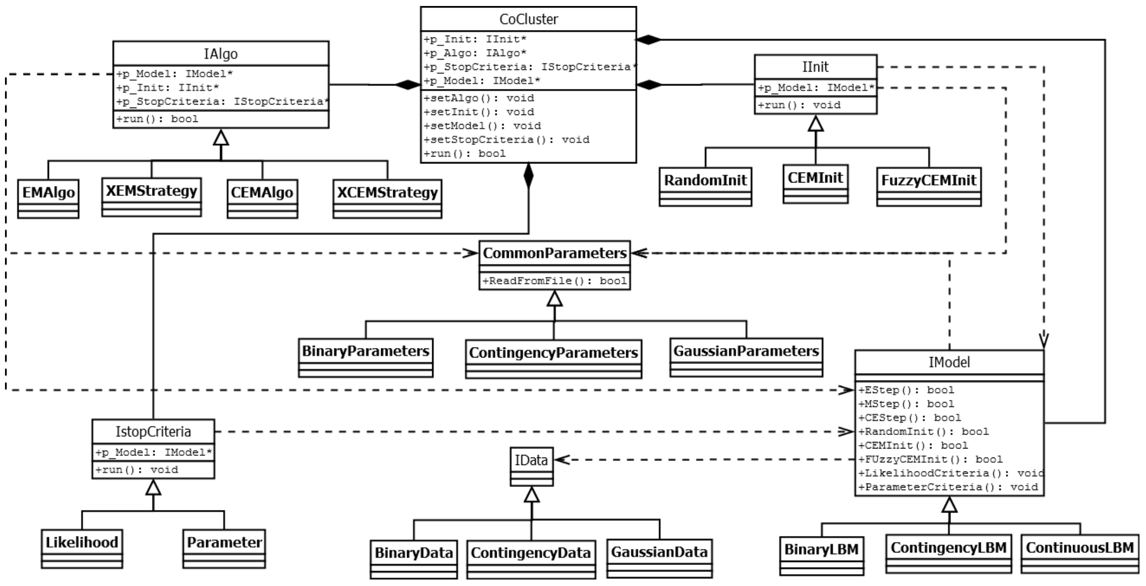


Figure 7: C++ core library software architecture based on strategy design pattern.

```
nbiterations_int = 5
Algorithm = XEMStrategy
StopCriteria = Parameter
Initialization = CEMInit
ModelName = pik_rhol_epsilonkl
```

This file contains all the input options needed to run the library. The only mandatory inputs are the data type, number of row/column clusters and the path to the data file. As the reader may have already guessed, this file is more or less similar to the **blockcluster** package function `coclustStrategy()` where the optional parameters in the sample text file are equivalent to arguments of the function. By looking at the `coclustmain` source file, one can easily understand how to integrate the core library with their own C++ codes.

Affiliation:

Parmeet Singh Bhatia
Siemens Healthineers
65 Valley Stream Parkway
Malvern-PA, 19355, United States of America
E-mail: parmeet.bhatia@siemens.com

Serge Iovleff
Laboratoire Paul Painlevé
Université Lille 1 & CNRS
Modal Team – Inria Lille – Nord Europe
59655 Villeneuve d'Ascq Cedex, France
E-mail: serge.iovleff@inria.fr

Gérard Govaert
Laboratoire Heudiasyc
Université de Technologie de Compiègne & CNRS
60205 Compiègne Cedex, France
E-mail: gerard.govaert@utc.fr

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
February 2017, Volume 76, Issue 9 *Submitted:* 2015-02-06
doi:10.18637/jss.v076.i09 *Accepted:* 2015-11-22