Chapter 6

CoClust: An R Package for Copula-Based Cluster Analysis

Francesca Marta Lilja Di Lascio

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.74865

Abstract

The aim of this chapter is to present and describe the R package `CoClust`, which enables implementing a clustering algorithm based on the copula function. The copula-based clustering algorithm, called CoClust, was introduced by Di Lascio and Giannerini in 2012 (Journal of Classification, 29(1):50–75), improved in 2016 (Statistical Papers, p.1–17, DOI 10.1007/s00362-016-0822-3), and is able to find clusters according to the complex multivariate dependence structure of the data-generating process. Hence, among other advantages, the CoClust overcomes the limitations of classic approaches that only deal with linear bivariate relationships. The first part of the chapter briefly describes the clustering algorithm. The second part illustrates the clustering procedure through the R package `CoClust` and presents numerical examples showing how the main R commands can be used to perform a fully developed clustering of multivariate dependent data.

Keywords: clustering algorithm, CoClust, copula function, multivariate dependence structure, R package

1. Introduction

Cluster analysis is an unsupervised classification method that aims to detect a structure within data by assigning a set of objects (observations or variables) into groups, called clusters, whereby objects in the same cluster are in some sense more closely related to each other than objects assigned to different clusters.

Literature on clustering methods is very extensive and different criteria are taken into account to organize and present these methods. One such criterion is the mathematical object of the clustering methods: a distance or dissimilarity measure versus a probability

model. In this chapter, we consider the model-based clustering methods that assume the data matrix is generated according to a specific data generating process (henceforth DGP). The classic model-based clustering method in [1, 2] is based on a mixture of multivariate probability distributions, such as the multivariate normal. However, this approach only accounts for the linear dependence between objects so that it inherits all the limitations of the linear correlation coefficient. Hence, we here focus on clustering methods that assume the data matrix is generated by a K-dimensional copula [3] such that each of the K clusters is represented by a (continuous) univariate density function and the complex multivariate relationship among clusters is expressed by the copula and its dependence parameter. Specifically, this chapter aims to describe the copula-based clustering algorithm first introduced by [4] and improved by [5], presenting in detail its implementation in the `R` package called `CoClust`. The CoClust approach inspired the work of [6], while different copula-based clustering approaches can be found in [7–9], [10–13], [14], and in [15, 16]. To the best of our knowledge, none of these methods have been implemented in software available to the scientific community.

Most clustering algorithms take as input some parameters, such as number of clusters, the distance or density of clusters, or the number of points in a cluster, and a starting classification. Some important benefits of the `R` function `CoClust`, which implements the copula-based clustering algorithm in [5], are that (i) the user can simultaneously test a multiple number of clusters in a single function call, (ii) there is no need for a starting classification, and (iii) the algorithm can nonparametrically estimate the density of the clusters, which are distributional free. The package is available from the Comprehensive R Archive Network (CRAN) at https://cran.r-project.org/web/packages/CoClust/index.html.

The chapter is organized as follows. Section 2 presents the theoretical tools of copula theory essential to understanding the copula-based clustering algorithm introduced and described in Section 3. Section 4 describes in detail the R implementation of the CoClust algorithm and illustrates its use on simulated DGPs. In Section 5, an application to a real dataset is presented, while a brief conclusion follows in Section 6.

2. Copula theory

The copula function [17–20] was born in the probabilistic metric space with Sklar's theorem [3] stating that every joint distribution function $F(\cdot)$ can be expressed in terms of K marginal distribution function F_k and the copula distribution function C as follows:

$$F(x_1, \ldots, x_k, \ldots, x_K) = C(F_1(x_1), \ldots, F_k(x_k), \ldots, F_K(x_K)) \tag{1}$$

for all $(x_1, \ldots, x_k, \ldots, x_K) \in \overline{\mathrm{R}}^K$ (where $\overline{\mathrm{R}}$ denotes the extended real line). According to this theorem, any joint probability function $f(\cdot)$ can be split into the margins $f_k(\cdot)$ and a copula $c(\cdot)$, so that the latter represents the association among variables, that is, the multivariate dependence structure of a joint density function [17–20]:

$$f(x_1, \ldots, x_k, \ldots, x_K) = c(F_1(x_1), \ldots, F_k(x_k), \ldots, F_K(x_K)) \prod_{k=1}^{K} f_k(x_k). \tag{2}$$

Such separation determines the modeling flexibility of copulas, since it enables (i) freely choosing the distribution of the margins and, separately, that of the copula, (ii) decomposing the estimation problem into two steps: in the first step, the margins are estimated, in the second step, the copula model is estimated, and (iii) combining different estimation methods or approaches.

The log-likelihood function of $f(\cdot)$ is composed of two positive terms as follows:

$$l(\Theta) = \sum_{i=1}^{n} \log c\{F_1(X_{1i}; \beta_1), \ldots, F_k(X_{ki}; \beta_k), \ldots, F_K(X_{Ki}; \beta_K); \theta\} + \sum_{i=1}^{n} \sum_{k=1}^{K} \log f_k(X_{ki}; \beta_k) \tag{3}$$

where the first term involves the copula density $c(\cdot)$ and its parameter θ, and the second involves marginal densities $f_k(\cdot)$ and their parameters β_k, and the whole set of parameters to be estimated is $\Theta = (\beta_1, \ldots, \beta_k, \ldots, \beta_K, \theta)$. Thus, it is possible to estimate $f(\cdot)$ by exploiting the decomposition into two terms of Eq. (2) [20, Chapter 4] using a sequential two-step maximum likelihood method, called inference for margins (henceforth IFM) [21]. This method estimates the marginal parameters in the first step and uses them to estimate the parameter of the copula function in the second step, in either a full or semi-parametric approach.

A full parametric approach for the IFM method is based on the estimation of the marginal parameters $(\beta_1, \ldots, \beta_k, \ldots, \beta_K)$ in the first step by the maximum likelihood estimation for each margin:

$$\widehat{\beta}_k = \arg \max_{\beta_k} \sum_{i=1}^{n} \log f_k(X_{ki}; \beta_k) \tag{4}$$

where each marginal distribution f_k has its own parameters β_k. In the second step, the dependence parameter θ given $\widehat{\beta}_k$ for $k = 1, \ldots, K$ is estimated by:

$$\widehat{\theta} = \arg \max_{\theta} \sum_{i=1}^{n} \log c\left[F_1\left(X_{1i}; \widehat{\beta}_1\right), \ldots, F_k\left(X_{ki}; \widehat{\beta}_k\right), \ldots, F_K\left(X_{Ki}; \widehat{\beta}_K\right); \theta\right]. \tag{5}$$

using the maximum likelihood estimation method.

The IFM method can also be used in a semi-parametric approach [22] where the margins are modeled without assumptions on their parametric form, that is, through the following empirical cumulative distribution function $\widehat{F}_k(X_{ki})$:

$$\widehat{U}_{ki} = \frac{n\widehat{F}_k(X_{ki})}{(n+1)} \tag{6}$$

where $\widehat{F}_k(X_{ki})$ is computed from $(X_{k1}, \ldots, X_{ki}, \ldots, X_{kn})$ with $k = 1, 2, \ldots, K$ and n is the sample size, while the copula parameter θ is estimated by using the following maximum log-likelihood function:

$$\widehat{\theta} = \arg\max_{\theta} \sum_{i=1}^{n} \log c\left[\widehat{U}_{1i}, \ldots, \widehat{U}_{ki}, \ldots, \widehat{U}_{Ki}; \theta\right]. \tag{7}$$

Note that the scaling factor $n/(n+1)$ in Eq. (6) is typically introduced in the nonparametric computation of the margins to avoid numerical problems at the boundary of $[0,1]^K$.

2.1. Copula models

While many different copula models are available in literature (see [18, 19] for details), the Elliptical and Archimedean families are shown to be the most useful in empirical modeling. The Elliptical family includes the Gaussian copula and the *t*-copula: both are symmetric, exhibit the strongest dependence in the middle of the distribution, and can take into account both positive and negative dependence, since $-1 \le \theta \le 1$. The Archimedean family enables describing both left and right asymmetry as well as weak symmetry among the margins using the Clayton, Gumbel, and Frank models, respectively. Clayton's copula has the parameter $\theta \in (0, \infty)$ and as θ approaches zero, the margins become independent. The dependence parameter θ of a Gumbel model is restricted to the interval $[1, +\infty)$ where the value 1 means independence. Finally, the dependence parameter θ of a Frank copula may assume any real value and as θ approaches zero, the marginal distributions become independent. According to the type of copula model, the value of θ has a specific meaning and the magnitudes of the dependence parameter are not comparable across copulas. It is always true that the greater the value of the dependence parameter, the stronger the association among the margins, but since the relationship between θ and the concordance measures is well known, it is standard to convert θ to these, for example, to the Kendall's τ correlation coefficient. The families of copula models considered here are described in **Table 1** and shown in **Figure 1** in their bivariate version. Note that here only single parameter copula models are considered.

The copula model selection task is still an open research field. Although various statistical tests enable evaluating whether a specific model is plausible or not, no tool has thus far been recognized as the best. In the copula-based clustering context, this issue can be overcome,

Copula	$\boldsymbol{C(u_1, u_2; \theta)}$	**Parameter range**	**Kendall's τ**
Gaussian	$\Phi_G\left[\Phi^{-1}(u_1), \Phi^{-1}(u_2)\right]$	$\theta \in (-1, 1)$	$\frac{2}{\pi}\arcsin(\theta)$
Student-*t*	$t_{2,\nu}\left(t_\nu^{-1}(u_1), t_\nu^{-1}(u_2); \theta\right)$	$\theta \in [-1, 1], \nu \in (2, \infty)$	$\frac{2}{\pi}\arcsin(\theta)$
Clayton	$\left[u_1^{-\theta} + u_2^{-\theta} - 1\right]^{-\frac{1}{\theta}}$	$\theta \in (0, \infty)$	$\frac{\theta}{\theta+2}$
Frank	$-\frac{1}{\theta}\ln\left\{1 + \frac{\left(e^{-\theta u_1}-1\right)\left(e^{-\theta u_2}-1\right)}{\left(e^{-\theta}-1\right)}\right\}$	$\theta \in (-\infty, \infty)$	$1 - \frac{4}{\theta}\left[1 - D_1(\theta)\right]$
Gumbel	$e^{\left[-(-\log u_1 - \log u_2)^{1/\theta}\right]}$	$\theta \in [1, \infty)$	$1 - \frac{1}{\theta}$

Table 1. Some standard single parameter bivariate copulas with the range of the dependence parameter θ and its relation with Kendall's τ. u_k with $k = 1, 2$ are uniformly distributed variates so that $x_k = F^{-1}(u_k) \sim F_k$. Φ is the cumulative distribution function (cdf) of the standard normal distribution, $\Phi_G(u_1, u_2)$ is the standard bivariate normal distribution, $t_{2,\nu}(\cdot, \cdot; \theta)$ denotes the standard bivariate student-*t* distribution with ν degrees of freedom, and t_ν^{-1} the inverse univariate student-*t* distribution function. $D_1(x)$ denotes the "Debye" function $1/x \int_0^x t/\left(\exp^t - 1\right)dt$.

http://dx.doi.org/10.5772/intechopen.74865

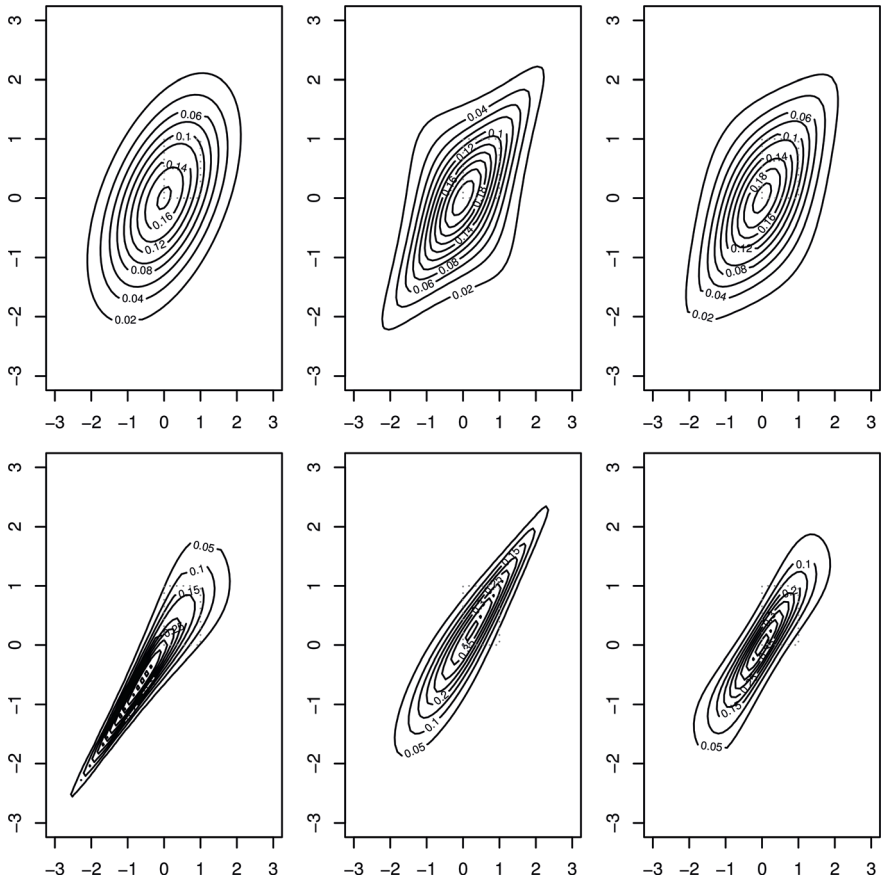

Figure 1. Contour plots of bivariate copula models with normal standard margins and dependence parameter θ such that the Kendall's correlation coefficient is $\tau = 0.7$; upper panel: Gaussian and t-Student copula models for 2 and 4 degrees of freedom; lower panel: Clayton, Gumbel, and Frank copula models.

since the choice of the type of model would seem less important in terms of the goodness of the final clustering (see [5], Section 4.3), and a classic information criterion can be used, such as the Bayesian or the Akaike information criterion. This topic is discussed in detail in Section 3.

3. CoClust algorithm

Di Lascio and Giannerini [4] proposed a clustering algorithm called CoClust that is able to cluster multivariate observations with a complex dependence structure. The basic underlying concept of CoClust is clustering multivariate dependent observations based on the likelihood copula fit estimated on the previously allocated observations. To do so, the CoClust assumes

that the data are generated by a multivariate copula function whose arguments represent the clusters, and each cluster is thus generated by a (marginal) univariate density function. The type and strength of multivariate dependence across clusters are modeled through a copula function and its dependence parameter, respectively. Being copula-based, CoClust inherits all the advantages of copula theory, and the multivariate complex dependence structure of the DGP can be taken into account to perform the cluster analysis. However, CoClust in its first version had some significant limitations. For example, it automatically allocated all the observations to the clusters without discarding potentially irrelevant observations, implying a high computational burden. Di Lascio and Giannerini [5] proposed a new version of the CoClust algorithm that satisfactorily overcomes these limitations. This section hereafter describes the latest version of the CoClust algorithm, which is implemented in the `R` package `CoClust`.

The starting point of the CoClust algorithm is the standard $(n \times p)$ data matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{ip} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{i'1} & \dots & x_{i'j} & \dots & x_{i'p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{nj} & \dots & x_{np} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_i \\ \vdots \\ \mathbf{x}_{i'} \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \tag{8}$$

in which np-dimensional objects have to be grouped in K groups. CoClust can be applied either to the row or to the column data matrix according to the purpose of the analysis. In both cases, CoClust works with vectors and treats each row (column) of the data matrix $\mathbf{X}$ as a single element to be allocated to a cluster. The values within a row (or column) vector are treated as independent realizations of the same density function, thus observations for each cluster from the same distribution. Here, CoClust is described as applied to the rows of the data matrix.

3.1. The basic procedure and selection of the number of clusters

The basic idea behind the CoClust consists in a forward procedure that allocates a K-plet of the row data matrix at a time, that is, a p-dimensional vector for each cluster at a time, and the decision on the allocation of each K-plet of rows is based on the value of the log-likelihood of the copula fit. This likelihood is computed by using the K-plets already allocated and one allocation candidate, say $\mathbf{x}_{i'} = \left(x_{i'1}, \dots, x_{i'j}, \dots, x_{i'p}\right)$, by varying the permutations of observations in $\mathbf{x}_{i'}$ in order to find, if it exists, the combination that maximizes the copula fit. If the log-likelihood of the copula fitted on the observations already allocated plus the permutation of the selected K-plet increases, then the candidate K-plet is allocated to the clusters; otherwise, it is discarded, since theoretically it could be either independent from the identified DGP or derive from another DGP.

http://dx.doi.org/10.5772/intechopen.74865

Before describing the clustering algorithm procedure, two aspects of the CoClust merit a discussion: the construction of the K-plet candidate to the allocation and the selection of number of clusters K. The K-plet of rows candidate to the allocation is constructed based on the following function $H(\cdot)$, which is a sort of multivariate measure of association based on the pairwise Spearman's ρ correlation coefficient:

$$H(\Lambda_2|\Lambda_1) = \max_{i' \in \Lambda_2} \left\{ \underset{i \in \Lambda_1}{\psi} \left(\rho(\mathbf{x}_i, \mathbf{x}_{i'})\right) \right\} \tag{9}$$

where Λ_1 is the subset of row index vectors already selected to compose a K-plet, Λ_2 is the subset of the remaining candidate row index vectors to complete it, and ψ is an aggregation function, for instance, the mean, median, or maximum.

As for the selection of K, one of the advantages of CoClust with respect to classic clustering techniques is its ability to automatically choose the number of clusters. Indeed, CoClust explores all the possibilities among those given by the user and selects the K on the basis of the log-likelihood of the copula estimated on the subsets of k-plets allocated up to the user's predefined step. The technical details are given below.

The main steps of the CoClust algorithm to cluster the n row data matrix are described in the following:

1. by varying the number of clusters k in the set of possibilities defined by the user and such that $2 \leq k \leq n$,
   a. select a subset of n_k k-plets of rows in the data matrix in Eq. (8) on the basis of the measure in Eq. (9);
   b. fit the copula model on the n_k k-plets of rows through the semiparametric estimation method described in Section 2;
2. select the subset of n_k k-plets of rows, say n_K K-plets, that maximizes the log-likelihood of the copula; hence, the number of clusters K, that is, the dimension of the copula, is automatically chosen and n_K K-plets are already allocated;
3. select a K-plet of rows among those remaining by using the measure in Eq. (9) and estimate $K!$ copulas by using the observations already clustered and a permutation of the candidate to the allocation;
4. allocate the permutation of the selected K-plet to the clustering by assigning each row to the corresponding cluster only if it increases the log-likelihood of the copula fit, otherwise drop the entire K-plet of rows;
5. repeat steps 3 and 4 until all the observations are evaluated, that is, either allocated or discarded.

At the end of the procedure, we obtain a clustering of K clusters each containing a maximum $(n/K)p$ independent observations such that the multivariate dependence relationship across clusters can be revealed. Hence, attention in recovering the multivariate relationships does not

rely on the within-cluster relationships, typical of classic clustering methods. A picture of the final CoClust clustering is given in **Figure 2**. Each cluster is a set of independent and identically distributed realizations from the same marginal distribution while observations across clusters share the same multivariate dependence structure.

Note that since in each step of the procedure non-nested models are compared, that is, copula models with a single dependence parameter, the described log-likelihood based criterion is equivalent to the well-known Bayes information criterion and Akaike information criterion. Finally, note that in the current CoClust version, the selection of the number of clusters K is based on a representative subset of n_k observations. Hence, the algorithm chooses the number of clusters K by estimating the $\sum_{k=K_{\min}}^{K_{max}} \binom{n_k}{k}$ fits required, where $[K_{\min}, K_{max}]$ is the range of the number of clusters predefined by the user with $K_{\min} \geq 2$ and n_k is chosen by the user with $n_k \ll p$. This allows keeping the computational complexity under control since it does not depend on sample size.

3.2. Selecting the copula model

The CoClust algorithm has not been implemented to automatically perform the selection copula model task and requires employing an information criterion *a posteriori*. The Bayesian information criterion (henceforth BIC) is expressed as follows for a K-dimensional copula model m:

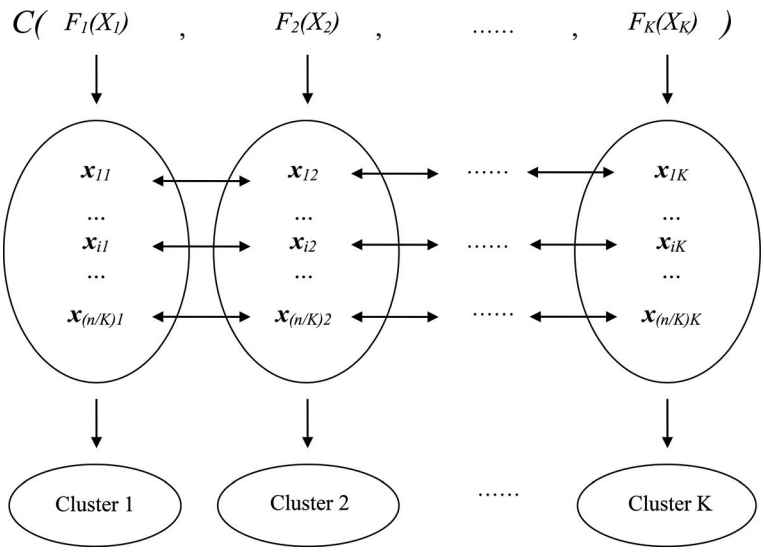

Figure 2. The basic concept underlying the CoClust algorithm. Each element in a cluster is a row data matrix of p elements.

$$BIC_{K,m} = -2\log\Pi_{i=1}^{n} c_m\left\{\widehat{F}_1(X_{1i}),\ldots,\widehat{F}_k(X_{ki}),\ldots,\widehat{F}_K(X_{Ki});\widehat{\theta}\right\} + s\log\left((n/K)p\right) \tag{10}$$

where $\widehat{\theta}$ is as in Eq. (5) or Eq. (7) with the summation over the number of allocated observations, which equals maximum $(n/K)p$ (i.e., n/K p-dimensional vectors) and s is the number of parameters. According to [23], we select the copula model that minimizes the BIC. Similarly, the Akaike information criterion (henceforth AIC) results in:

$$AIC_{K,m} = -2\log\Pi_{i=1}^{n} c_m\left\{\widehat{F}_1(X_{1i}),\ldots,\widehat{F}_k(X_{ki}),\ldots,\widehat{F}_K(X_{Ki});\widehat{\theta}\right\} + 2s \tag{11}$$

and can also be used to select the copula model.

3.3. Assessing the CoClust performance

The goodness of the CoClust algorithm in finding the true multivariate clustering structure underlying the data has been extensively investigated. Specifically, the first version of CoClust [4] was tested on simulated data for different scenarios and compared with model-based clustering [1, 2]. This shows that, both when the DGP is a copula and when it is misspecified, CoClust appears to be able to identify both the true number of clusters and their size in most situations. Moreover, in comparing model-based clustering, CoClust appears better suited to clustering dependent data. In [5], a more sophisticated Monte Carlo study was carried out, investigating the new features of the current version of the CoClust algorithm. Here, the current version of the algorithm clearly outperforms the previous version by [4] and CoClust's ability to find the correct number of clusters and to reconstruct the true k-plets by varying the dimension of the copula, the aggregation function ψ, and the copula model appears to be very satisfactory. Furthermore, [5] also obtained good results in assessing CoClust's ability to drop from the clustering observations that are independent of the true DGP as well as distinguishing two different DGPs in the same dataset.

As for real data applications, the CoClust algorithm has been successfully applied to several datasets. In relation to biomedical applications, [4] apply the CoClust to microarray data to formulate hypotheses on the possible co-regulation and functional relations between genes, [5] use the copula-based clustering method to identify biologically and clinically relevant groups of tumor samples, and [24] attempt to identify organ type from cancer cell lines from tumors. Applications in other fields include [25], where the purpose of the analysis is investigating changes in EU country diets in accordance with common European policies and guidelines on healthy diets, and [24], where CoClust is used to investigate the geographic distribution of (annual maxima) rainfall measurements.

4. The R implementation of the CoClust algorithm

The copula-based clustering algorithm procedure is implemented in the `R` package `CoClust` [26]. It must be installed in the usual way, that is:

```
R> install.packages("CoClust")
```

and then it must be loaded through the usual code:

```
R> library("CoClust")
```

The code of the `CoClust` package is entirely written in `R`, to enable using an easily accessible open source system and the input/output facilities.

4.1. List of functions and subroutines

The main R function is `CoClust()`, which performs the copula-based clustering, while the following auxiliary R functions.

`fit.margin()`, `fit.margin2()`, `fit.margin3()`, `fcond.mod()`, `CoClust_perm()`, `stima_cop()` are intended for internal use only and are not documented in the package.

4.2. The CoClust function

The main function of the package `CoClust` is the R function `CoClust()`, which performs copula-based clustering as described in Section 3. Some options are present, which mainly allow us to:

- fit a variety of copula models (by setting the argument `copula`) with different types of estimation procedures for margins and for copulas (arguments `method.ma` and `method.c`, respectively); specifically, all the copula models belonging to the Elliptical and the Archimedean family described in Section 2 can be estimated through the estimation methods implemented in the `R` package `copula` [27–30] that are maximum pseudo-likelihood estimators based on two different variance estimators, the inversion of Kendall's τ estimator and the inversion of Spearman's ρ estimator; as for the margins, two different estimation methods have been implemented, one parametric and one nonparametric: the maximum likelihood method as in Eq. (4) and the empirical cumulative distribution function in Eq. (6);
- set the range or set of dimensions for the copula model, that is, number of clusters, for which the function tries the clustering (argument `dimset`);
- set the dimension of the sample units used for selecting the number of clusters (argument `noc`);
- select the combination function of the pairwise Spearman's ρ used to select the k-plets among the mean, the median, or the maximum (argument `fun`) as defined in Eq. (9);
- specifies the likelihood criterion used for selecting the number of clusters among the AIC, the BIC (as defined in Eqs. (10) and (11)), and the log-likelihood without penalty terms (argument `penalty`).

The argument `copula` allows specifying a copula model among those described in Section 2.1. As for the selection of the "best" model, CoClust can be run by varying the type of models of

interest and selecting the one that fits best *a posteriori* using one of the criteria introduced in Section 3.2.

The typical use of the function `CoClust` is as follows:

```
CoClust(m, dimset = 2:5, noc = 4, copula = "frank", fun = median,
        method.ma = c("empirical", "pseudo"), method.c = c("ml", "mpl",
        "irho", "itau"),
        dfree = NULL, writeout = 5, penalty = c("BICk", "AICk", "LL"), …)
```

where `m` is the entry data matrix and the `writeout` argument allows monitoring the allocation process, since it informs on each new allocated observation. Further details on the input arguments are given in the package help files.

The main output of the function `CoClust` is an object of S4 class "CoClust" which is a list with the following elements:

1. Number of Clusters: the number K of selected and identified clusters;
2. Index Matrix: a $n.obs \times (K+1)$ matrix where *n.obs* is the number of observations put into each cluster; the matrix contains the row indexes of the observations of the data matrix `m` (Eq. (8)) and in the last column the log-likelihood of the copula fit;
3. Data Clusters: the data matrix of the final clustering; each column contains the observations allocated in a cluster;
4. Dependence: a list containing:
   a. `Model`: the copula model used for the clustering;
   b. `Param`: the estimated dependence parameter between/among clusters;
   c. `Std.Err`: the standard error of `Param`;
   d. `P.val`: the p-value associated to the null hypothesis $H_0 : \theta = 0$;
5. LogLik: the maximized log-likelihood copula fit;
6. Est.Method: the estimation method used for the copula fit;
7. Opt.Method: the optimization method used for the copula fit;
8. LLC: the value of the log-likelihood criterion for each k in `dimset`;
9. Index.dimset: a list that, for each k in `dimset`, contains the index matrix of the initial set of n_k observations used to select the number of clusters, together with the associated maximized log-likelihood copula fit.

4.3. Simulated examples

This section shows how to use the `CoClust` package on data simulated from different DGPs. In the first example, the data are drawn from a joint density function with different margins,

whereas in the second example, a misspecified DGP is used. In these examples, we focus only on the semi-parametric approach described in Section 2 due to its theoretical and computational advantages with respect to the full parametric approach. Moreover, the latter has only been implemented for Gaussian margins.

Example 1

In this example, we build a 3-variate joint density function through a 3-dimensional Frank copula with dependence parameter such that the Kendall's $\tau = 0.7$ and three different margins: a Gaussian with parameters $\mu = 7, \sigma = 2$, a Gamma with shape and rate, respectively, set to 3 and 4, and a Beta with parameters $\alpha = 2, \beta = 1$. To do so, we employ the function `mvdc` of the `copula` package [27–30]. Next, we generate a data matrix X with 15 rows and 21 columns and build the matrix of the true cluster indexes. Finally, we apply the function `CoClust` to the rows of X, recover the multivariate dependence structure of the data and compare the obtained clustering with the true one.

Code to generate the example dataset is given in the following. We first define the DGP:

```
R> n.marg <- 3
R> theta <- iTau(frankCopula(), 0.7)
R> copula <- frankCopula(theta, dim = n.marg)
R> mymvdc <- mvdc(copula, c("norm", "gamma", "beta"),list(list(mean=7, sd=2),
+              list(shape=3, rate=4), list(shape1=2, shape2=1)))
```

and then generate the data and save the true clustering in index.true:

```
R> set.seed(11)
R> n.col  <- 21
R> n.row  <- 15
R> n      <- n.row*n.col/n.marg
R> x.samp <- rMvdc(n, mymvdc)
R> X      <- matrix(x.samp, nrow = n.row, ncol = n.col, byrow=TRUE)
R> index.true <-  matrix(1:n.row, n.row/n.marg, n.marg)
R> colnames(index.true) <- c("Cluster 1", "Cluster 2", "Cluster 3")
R> index.true
```

Note that n is the number of observations for each margin.

We apply the `CoClust` to the 15×21 data matrix X using the maximum likelihood estimation method for the copula, the empirical cumulative distribution function for the three margins, and leaving by default the remaining arguments:

```
R> clust <- CoClust(X, dimset = 2:4, noc=2, copula="frank", method.
ma="empirical",
+          method.c="ml", writeout=1)
```

The output is as follows:

```
R> clust
```

```
An object of class "CoClust"
Slot "Number.of.Clusters":
[1] 3

Slot "Index.Matrix":
     Cluster 1 Cluster 2 Cluster 3 LogLik
[1,]       11         1         6 34.15693
[2,]       13         3         8 69.87149
[3,]       12         2         7 103.67653
[4,]       14         4         9 136.31506
[5,]       15         5        10 170.36557

Slot "Data.Clusters":
       Cluster 1 Cluster 2  Cluster 3
  [1,] 0.35776965  4.566417  0.1634203
  [2,] 0.36621352  5.532188  0.1470511
  [3,] 0.99290268 11.191092  1.4006169
  [4,] 0.60411081  6.613533  0.5457595
  [5,] 0.13946354  2.658381  0.3489739
  [6,] 0.80523424  9.526025  0.6908222
  [7,] 0.79477600  8.899494  0.7864765
  [..]    .......   .......    .......
[102,] 0.36904520 3.629722 0.2763105
[103,] 0.82647042 10.809628 1.3899675
[104,] 0.48283666 5.185873 0.2763133
[105,] 0.90394435 10.583053 0.9962130

Slot "Dependence":
$Copula
[1] "frank"

$Param
[1] 11.95576

$Std.Err
[1] 0.8261832

$P.value
[1] 0


Slot "LogLik":
[1] 170.3656
```

```
Slot "Est.Method":
[1] "maximum likelihood"

Slot "Opt.Method":
[1] "ml"

Slot "LLC":
         2           3         4
 -70.93179  -136.00532 -41.35904

Slot "Index.dimset":
$`2`
     1 2   LogLik
[1,] 11 1 19.75591
[2,]  8 3 37.33473

$`3`
     1 2 3   LogLik
[1,] 11 1 6 34.15693
[2,] 13 3 8 69.87149

$`4`
     1 2  3  4     LogLik
[1,] 11 1  6 12  3.653809
[2,]  7 3 13  8 22.548352
```

To look at specific objects of the resulting list, it is possible to select, among others, the following information:

```
R> clust@"Number.of.Clusters"   # Selected number of clusters
R> clust@"Dependence"$Param     # Estimated copula parameter
R> clust@"Data.Clusters"        # Clustered data
```

To compare the obtained clustering with the true clustering we can input:

```
R> index.clust <- clust@"Index.Matrix"
R> index.clust
R> index.true
```

to obtain as follows:

```
> index.clust
     Cluster 1 Cluster 2 Cluster 3   LogLik
[1,]        11         1         6  34.15693
[2,]        13         3         8  69.87149
[3,]        12         2         7 103.67653
```

```
[4,]        14         4          9  136.31506
[5,]        15         5         10  170.36557
> index.true
     Cluster 1 Cluster 2 Cluster 3
[1,]         1         6        11
[2,]         2         7        12
[3,]         3         8        13
[4,]         4         9        14
[5,]         5        10        15
```

The obtained clustering is perfect, CoClust is able to recognize the exact structure underlying the data. Note that the label of each cluster, that is, the order of the margins, is not relevant. The only important aspect is the composition of each cluster, that is, the row indexes in each column of `index.clust` and their order that has to be such that it reconstructs the exact 3-plets across the columns.

To apply the `CoClust` to the 15×21 data matrix X previously generated by changing the argument `fun` in `max`, or the range of number of clusters to be tried or the copula model, we can input respectively:

```
R> clust <- CoClust(X, dimset = 2:4, noc=2, copula="frank", fun=max,
+                   method.ma="empirical", method.c="ml", writeout=1)

R> clust <- CoClust(X, dimset = 3:5, noc=2, copula="frank",
+                   method.ma="empirical", method.c="ml", writeout=1)

R> clust <- CoClust(X, dimset = 2:4, noc=2, copula="clayton",
+                   method.ma="empirical", method.c="ml", writeout=1)
```

Example 2

In this example, we use a different DGP from the copula, thus showing the use of CoClust in the misspecification case. Specifically, a 30×21 data matrix is drawn from a three-dimensional skew-normal distribution through the R package `sn` [31], we then apply CoClust to cluster the row data matrix:

```
R> library(sn)
R> n.marg <- 3
R> rho    <- 0.7
R> mu     <- c(4,6,7)
R> v1     <- 1
R> v2     <- 1
R> v3     <- 1
R> omega <- matrix(c(v1, rho*sqrt(v1*v2), rho*sqrt(v1*v3), rho*sqrt(v1*v2), v2,
+        rho*sqrt(v3*v2), rho*sqrt(v1*v3), rho*sqrt(v3*v2), v3), n.marg)
R> alpha  <- c(-1,1,1)
R> n.col  <- 21
```

```
R> n.row  <- 60
R> n.k    <- n.row/n.marg
R> n      <- n.row*n.col/n.marg
R> set.seed(11)
R> x.samp <- rmsn(n, xi=mu, Omega=omega, alpha=alpha)
R> X      <- matrix(x.samp, nrow=n.row, ncol=n.col, byrow=TRUE)
R> clust  <- CoClust(X, dimset=2:5, noc=4, copula="clayton",method.
ma="empirical",
+                    method.c="ml", penalty = "BICk", writeout=1)
```

On the console, it is possible to monitor the number of observations already allocated (argument writeout). Indeed, while CoClust runs, the following information appears on the console:

```
Number of clusters selected: 3
Allocated observations: 5
Allocated observations: 10
Allocated observations: 15
```

To look at the obtained clustering and its details, one has to input:

```
R> clust
R> index.clust <- clust@"Index.Matrix"
R> index.true  <- matrix(1:n.row, n.row/n.marg, n.marg)
R> index.clust; index.true
```

Note that when the number of *K*-plets to be allocated is not small, the goodness of the obtained clustering is difficult to determine. Hence, for example, two functions can be exploited to assess the quality of the final clustering: `pca.coclust`, which counts how many *K*-plets of the true DGP have been correctly allocated in the final clustering, and `pcc.coclust`, which counts how many *K*-plets of the obtained clustering have been correctly allocated. In Appendix A., the R code of these two functions is shown. Here, we compute the true clustered index matrix as follows:

```
R> ind.t <- apply(matrix(1:n.row,n.k), FUN=paste, MARGIN=1, collapse="-")
```

and the two functions `pca.coclust` and `pcc.coclust` after loading the required package `gtools`:

```
R> library(gtools)

R> pca.coclust(clust, ind.t, n.marg)
[1] 65

R> pcc.coclust(clust, ind.t, n.marg)
[1] 86.66667
```

The obtained values inform us that 65% of 3-plets deriving from the true DGP are correctly allocated and 86.7% of 3-plets in the final clustering are correctly allocated.

5. Application to wine dataset

In this section, an application of the CoClust package to a real dataset is shown. [32] analyze a set of Italian wines by observing the chemical properties of 178 specimens of three types of wines (Barolo, Grignolino, and Barbera) produced in the Piedmont region in Italy. The data are available in the package sn under the name wines.

A subset of randomly selected wines has been analyzed through CoClust by varying the number of clusters from 2 to 7 and the copula model among the three models of the Archimedean family. Since Grignolino is a type of wine with characteristics between those of Barolo and Barbera, we work with a sample of only these two last types of wines. Code is as follows:

```
R> data(wines)
R> n <- 6
R> set.seed(11)
R> ind.sample <- c(sample(1:59,n,replace=FALSE),sample(131:178,n,replace=FALSE))
R> X <- wines[ind.sample,-1]

R> clustF <- CoClust(X, dimset = 2:7, noc=1, copula="frank", method.
ma="empirical",
+             method.c="ml",writeout=1)

R> clustC <- CoClust(X, dimset = 2:7, noc=1, copula="clayton", method.
ma="empirical",
+             method.c="ml",writeout=1)

R> clustG <- CoClust(X, dimset = 2:7, noc=1, copula="gumbel", method.
ma="empirical",

+             method.c = "ml",writeout = 1)
```

To evaluate the final clustering obtained with a specific copula model, say the Frank model, and to compare it with the true classification of the 12 selected wines, the code is as follows:

```
R> Type.wine <- wines[ind.sample,1]
R> Type.wine

R > K < — clustF@"Number.of.Clusters".

R > index.clust <— clustF@"Index.Matrix".

R> index.clust
R> index.fin <- matrix(Type.wine[index.clust[,1:K]],nrow=nrow(index.clust),
 +                  ncol=(ncol(index.clust)-1))
R> index.fin
     [,1]      [,2]        [,3]      [,4]      [,5]       [,6]
```

```
[1,] "Barolo"  "Barolo"  "Barolo"  "Barolo"  "Barolo"  "Barolo"
[2,] "Barbera" "Barbera" "Barbera" "Barbera" "Barbera" "Barbera"
```

CoClust selects 6 clusters and allocates to each cluster the two types of wines. Thus, across clusters, we can perfectly recognize the two types of Italian wines and in each cluster we have different wines with different (e.g., independent) chemical characteristics.

Similarly, the other two copula models can be used as in `clustC` and `clustG` above. The results appear to not be affected by the type of model used even though, based on the log-likelihood of the copula fitted on the final clustering, the more appropriate model appears to be the Gumbel model with a log-likelihood equal to 527.3022 (compared to 500.8835 for the Frank copula and 429.184 for the Clayton copula).

6. Conclusion

In this chapter, we describe a copula-based clustering algorithm and its implementation in the `R` package `CoClust`. One major advantage of this new package is that it provides an algorithm that is able to cluster multivariate observations by taking into account their underlying complex multivariate dependence structure. Being copula-based, the CoClust algorithm inherits the benefits of the copula. Thus, potentially any type of multivariate dependence structure can be handled and the most appropriate method can be employed to estimate both a probability model for each cluster/margin and the copula model.

The current version of the R package implements the clustering algorithm procedure in the main function `CoClust`. It enables the user to simultaneously choose the copula model, the estimation method for the margins and for the copula, the aggregation function for constructing the k-plet of observation allocation candidates. Moreover, the range (or set) of the number of clusters from among which the procedure automatically selects the best one and the sample size to be used to select it can be varied.

As with many other software packages, `CoClust` package is continually being augmented and improved. We are currently investigating possible graphical solutions for the final clustering and implementing some measures to validate the clustering solution. Another future direction includes expanding the functionality of the `CoClust` package to allow comparing the solution of other clustering algorithms, such as mixture-based clustering and hierarchical clustering methods.

Acknowledgements

The author acknowledges the support of the Free University of Bozen-Bolzano, Faculty of Economics and Management, via the project “Aggregation functions for Innovation and Data

Analysis (AIDA)" and Professor Simone Giannerini, University of Bologna, with whom the first version of the package was developed.

A. Appendix

The following two functions are useful to evaluate the goodness of the final clustering obtained through the CoClust algorithm when true clustering or benchmark clustering is available. The arguments of these two functions are `ccfit`, which is the object `CoClust` as given by the corresponding R function; `ind.t`, which is the true clustering expressed through the clustered index matrix with clusters by columns and the row index of matrix in Eq. (8) by rows; and `nmarg`, which is the dimension of the copula model, that is, the selected number of clusters. For an example of the use of these two functions see Section 4.3, "Example 2".

```
R> library("gtools")

pca.coclust <- function(ccfit, ind.t, nmarg){
    n.marg <- ccfit@"Number.of.Clusters"
    ind.perm <- permutations(n.marg,n.marg)
    n.comb <- nrow(ind.perm)
    if(n.marg==nmarg){
        ind.cc <- ccfit@"Index.Matrix"[,1:n.marg]
        n.kp <- nrow(ind.cc)
        res <- rep(NA,n.kp)
        for(i in 1:n.kp){
            dum <- ind.cc[i,]
            res0 <- rep(NA,n.comb)
            for(j in 1:n.comb){
                ind.ccs <- dum[ind.perm[j,]]
                ind.ccs <- paste(ind.ccs, collapse="-")
                 res0[j] <- as.integer(ind.ccs%in%ind. t)
            }
            res[i] <- any(res0)
        }
        pca.k <- sum(res)/length(ind.t)*100
    }
    return(pca.k=pca.k)
}
pcc.coclust <- function(ccfit, ind.t, nmarg){
    n.marg <- ccfit@"Number.of.Clusters"
    ind.perm <- permutations(n.marg,n.marg)
    n.comb <- nrow(ind.perm)
    if(n.marg==nmarg){
       ind.cc <- ccfit@"Index.Matrix"[,1:n.marg]
```

```
        n.kp <- nrow(ind.cc)
        res <- rep(NA,n.kp)
        for(i in 1:n.kp){
            dum <- ind.cc[i,]
            res0 <- rep(NA,n.comb)
            for(j in 1:n.comb) {
                ind.ccs <- dum[ind.perm[j,]]
                ind.ccs <- paste(ind.ccs, collapse="-")
                res0[j] <- sum(ind.ccs%in%ind.it)
            }
            res[i] <- any(res0)
        }
        pcc.k    <- sum(res)/nrow(ccfit@"Index.Matrix")*100
    }
    return(pcc.k=pcc.k)
}
```

Author details

Francesca Marta Lilja Di Lascio

Address all correspondence to: marta.dilascio@unibz.it

Faculty of Economics and Management, Free University of Bozen-Bolzano, Italy

References

[1] Fraley C, Raftery AE. How many clusters? Which clustering method? Answers via model–based cluster analysis. The Computer Journal. 1998;**41**(8):578-588

[2] Fraley C, Raftery AE. Model–Based Clustering, Discriminat Analysis and Density Estimation. Technical report. Department of Statistics, University of Washington; 2000

[3] Sklar A. Fonctions de répartition à n dimensions et leures marges. Publications de l'Institut de Statistique de L'Université de Paris. 1959;**8**:229-231

[4] Di Lascio FML, Giannerini S. A copula-based algorithm for discovering patterns of dependent observations. Journal of Classification. 2012;**29**(1):50-75

[5] Di Lascio FML, Giannerini S. Clustering dependent observations with copula functions. Statistical Papers. 2016:1-17

[6] Chessa A, Crimaldi I, Riccaboni M, Trapin L. Cluster analysis of weighted bipartite networks: A new copula-based approach. PLoS One. 2014;**9**(10):e109507

[7] Durante F, Pappadà R, Torelli N. Clustering of financial time series in risky scenarios. Advance in Data Analysis and Classification. 2014;**8**:359-376

[8] Durante F, Pappadà R. Cluster analysis of time series via Kendall distribution. In: Grzegorzewski P, Gagolewski M, Hryniewicz O, Gil MA, editors. Strengthening Links Between Data Analysis and Soft Computing, Volume 3l5 of Advances in Intelligent Systems and Computing. Springer International Publishing; 2015. pp. 209-216

[9] Durante F, Pappadà R, Torelli N. Clustering of time series via non–parametric tail dependence estimation. Statistical Papers. 2015;**56**(3):701-721

[10] De Luca G, Zuccolotto P. A tail dependence-based dissimilarity measure for financial time series clustering. Advances in Data Analysis and Classification. 2011;**5**(4):323-340

[11] De Luca G, Zuccolotto P. Time series clustering on lower tail dependence for portfolio selection. In: Corazza M, Pizzi C, editors. Mathematical and Statistical Methods for Actuarial Sciences and Finance. Berlin: Springer; 2014. pp. 131-140

[12] De Luca G, Zuccolotto P. Dynamic tail dependence clustering of financial time series. Statistical Papers, page in press. 2015

[13] De Luca G, Zuccolotto P. A double clustering algorithm for financial time series based on extreme events. Statistics *and Risk Modeling*. 2017;**34**(1–2):1-12

[14] D'Urso P, Disegna M, Durante F. Copula-based fuzzy clustering of time series. In: Mola F, Conversano C, editors. Book of Abstracts of the 10th Scientific Meeting of the Classification and Data Analysis Group, Page 4. Cagliari: CUEC; 2015

[15] Arakelian V, Karlis D. Clustering dependencies via mixtures of copulas. Communication in Statistics - Simulation and Compution. 2014;**43**(7):1644-1661

[16] Kosmidis I, Karlis D. Model-based clustering using copulas with applications. Statistics and Computing. 2016;**26**(5):1079-1099

[17] Cherubini U, Luciano E, Vecchiato W. Copula Methods in Finance. Chichester: Wiley Finance Series. John Wiley & Sons Ltd.; 2004

[18] Durante F, Sempi C. Principles of Copula Theory. Boca Raton: CRC Press; 2015

[19] Nelsen RB. Introduction to Copulas. New York: Springer; 2006

[20] Trivedi PK, Zimmer DM. Copula Modeling: An Introduction for Practitioners, volume 1. Foundations and Trends in Econometrics. 2005

[21] Joe H, Xu J. The Estimation Method of Inference Functions for Margins for Multivariate Models. Technical report. Department of Statistics, University of British Columbia; 1996

[22] Genest C, Ghoudi K, Rivest LP. A semiparametric estimation procedure of dependence parameters in multivariate families of distributions. Biometrika. 1995;**82**:543-552

[23] Raftery AE, Nema D. Variable selection for model-based clustering. Journal of the American Statistical Association. 2006;**101**(473):168-178

[24] Di Lascio FML, Durante F, Pappadà R. Copulas and Dependence Models with Applications - Contributions in Honor of Roger B. Nelsen, chapter copula–based clustering methods. Springer; 2017. pp. 49-67

[25] Di Lascio FML, Disegna M. A copula-based clustering algorithm to analyse eu country diets. Knowledge-Based Systems. 2017;**132**:72-84

[26] Di Lascio FML, Giannerini S. CoClust: Copula based cluster analysis. R package version 0.3-2; 2017

[27] Hofert M, Ivan Kojadinovic I, Maechler M, copula JY. Multivariate dependence with copulas. R package version 0.999-18; 2017

[28] Hofert M, Maechler M. Nested archimedean copulas meet r: The nacopula package. Journal of Statistical Software. 2011;**39**(9):1-20

[29] Kojadinovic I, Yan J. Modeling multivariate distributions with continuous margins using the copula r package. Journal of Statistical Software. 2010;**34**(9):1-20

[30] Yan J. Enjoy the joy of copulas: With a package copula. Journal of Statistical Software. 2007;**21**(4):1-21

[31] Azzalini A. The R Package `sn`: The Skew-Normal and Skew-T Distributions, version 1.5-0; 2017

[32] Forina M, Armanino C, Castino M, Ubigli M. Multivariate data analysis as a discriminating method of the origin of wines. Vitis. 1986;**25**:189-201