Model-based Clustering with Application of Copulas for Symbolic Data

by

Wenhao Pan

(Under the Direction of Lynne Billard)

Abstract

Contemporary data sets can be too large or complex for traditional statistical methods to handle. One approach is to use symbolic data first introduced by Diday (1987). Our interest is the study of model-based clustering for symbolic data, especially for distributions (i.e., observations are not single numerical point values). We will describe symbolic data and consider differences between symbolic data and classical data. For multivariate data, with $p > 1$, we only have the marginal distributions; so we do not know the dependence relationship between random variables. One approach to measure these dependences is that of Vrac et al. (2012) in which a copula function is used to describe the cumulative joint distribution function of random variables in a mixture model. We further develop the algorithm from various perspectives. The model-based clustering algorithm is also implemented in R and applied to simulated data.

Index words: Distributional Data, Model-based Clustering, Copulas

Model-based Clustering with Application of Copulas for Symbolic Data

by

Wenhao Pan

B.S., Shandong University of Economics and Finance, 2011

M.S., Rutgers University, 2013

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2018

Model-based Clustering with

Application of Copulas for Symbolic Data

by

Wenhao Pan

Major Professor:	Lynne Billard
Committee:	Cheolwoo Park
	Jaxk H Reeves
	Nicole Lazar
	Ray Bai

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
August 2018

Acknowledgments

I would like to express my deepest appreciation to my major advisor Dr. Lynne Billard. I can still remember how I was motivated by her attitude towards research after the first meeting of discussion about four years ago. Without the support of Dr. Billard, I would not be such lucky to devote myself to the research topic that I am truly interested in. She leads me to the real world of academic research and helps me to think independently and study with great eagerness. She is always there for me not only with her professional guidance but also when I need more confidence. She is my role model in pursuing my career path and following my heart to be the person that I want to be. I will never forget her encouraging words: you will always be better than you expect.

I would also like to thank my committee members, Dr. Cheolwoo Park, Dr. Jaxk H Reeves, Dr. Nicole Lazar and Dr. Ray Bai for their help in reviewing my work and giving valuable suggestions. A thank you to Dr. Paul Schliekelman and Dr. Liang Liu for their support with my comprehensive exams and proposal defense. I appreciate Dr. Daniel Hall for his tremendous help of being our team advisor for the 2016 SAS Analytics Shootout Annual Student Competition. I am also truly grateful for Dr. Jaxk Reeves and Dr. Xianyan Chen' extensive personal and professional guidance when I work in the Statistical Consulting Center. Moreover, I would like to thank the faculty members, staff and my classmates from the department of statistics. You have made these five years at the University of Georgia an

unforgettable and awesome memory in my life.

This work would not have been possible without the support of my family. I would like to thank my parents and grandparents, for their unconditional love and endless support in pursuing my Ph.D. study. I wish to especially thank my loving and supportive husband, Chenjiang, who always trust me and let me follow my dream; my parents-in-law, for their understanding and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The focus of our research is to study the method of model-based clustering for symbolic data, especially for distributions. Figure 1.1 illustrates the difference between clustering algorithms for classical data and distributional data, which is one of the symbolic types of data. When we have a data set with repeated measurements in each statistical units, the existing clustering methods can only allow us to take the average of all the measurements in each unit and then apply the clustering algorithm for all the average values. However, taking average will lose lots of useful information of the original data, say the within variations. Vrac et al. (2012) proposed to use the cumulative distribution function as the distributional-data to represent all the measurements in each unit to keep the original information as much as possible. When we only have the marginal distributions, it is difficult to define the dependence relationship between random variables. Vrac et al. (2012) applied the use of a copula function to describe the cumulative joint distribution function of random variables in a mixture model. However, Vrac et al. (2012) only considered one simple case example when the joint distribution is non-differentiable. In addition, for the choices of copula functions, only the Frank copula is applied in the algorithm. We first extended the algorithms in any possible cases when the joint distribution is non-differentiable and differentiable. We also developed

the algorithms with six candidate copula functions in one iteration and let the data itself to pick the best one. In addition, the algorithms were developed for one variable case and multi-variable case. We finally made our algorithms applicable with the implementation in R language.

Symbolic data were first introduced by Diday (1987). In Chapter 2, section 2.1 introduces what are symbolic data and what is the difference between symbolic data and classical data. Section 2.2 describes the commonly used hierarchical and partitioning methods. In section 2.3 and section 2.4, after reviewing model-based clustering techniques for classical data, we introduce how to apply copula functions in mixture models to do model-based clustering for symbolic data. The final mixture models (see equation 2.7) are given for clustering of distributions. In Chapter 3, we first introduce the dynamic cluster algorithm (Diday and Simon, 1976) and two commonly used clustering criterions: Log-likelihood criterion and Log-likelihood classification criterion to estimate the final mixture models. Then, equation 3.2 shows the final likelihood equation to be maximized. By using equation 3.2, section 3.2 introduces empirical density estimation to determine the form of the marginal density function. Then, section 3.3 shows commonly used copula famlies and their density functions to be estimated. In Chapter 4, we first summarize the density estimation methods in section 4.1. The methods of copula estimation and corresponding R packages are illustrated in section 4.2. In Chapter 5, we first introduce the theories that we use to develop the algorithms of model-based clustering for distributional-data in section 5.1. Section 5.2 shows the detailed partition algoritms with different estimation methods for differentiable and non-differentiable joint distributions in one dimensional and multi-dimensional cases. Chapter 6 gives six different simulation examples by applying the algorithms introduced in Chapter 5.

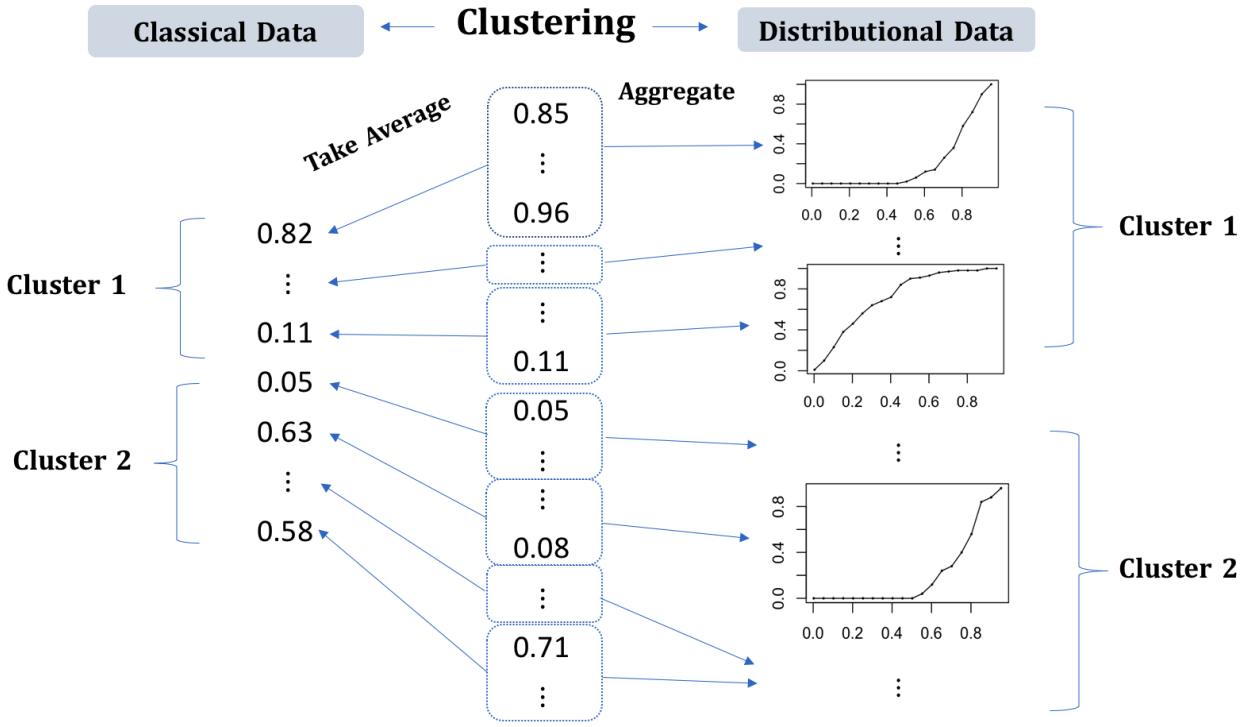

Figure 1.1: Comparison of Clustering Algorithms for Classical Data and Symbolic Data.

Chapter 2

Literature Review

We present a brief introduction to symbolic data in section 2.1 and an overview of the major clustering methods in Section 2.2. Then we summarize the model-based clustering methods for classical data in Section 2.3. Section 2.4 is an extension of model-based clustering from classical data to symbolic data.

2.1 Introduction of Symbolic Data

In contrast to classical observations which are points, symbolic values can be lists, intervals, histograms and so on. They arise in different ways. Some symbolic data exist in their own right to keep more information. For example, if the random variable of interest is color and the population is species of birds, one realization for a given species can take several possible colors, e.g., a magpie with colors {white, black} is a realization of a symbolic random variable. An all-black bird is not a magpie. Some other symbolic data may come from aggregating original data sets to produce collective data based on research interests. For example, a supermarket keeps track of the sales of its products. It is not really interested in the sale of any one item, like Mary's purchase of bread yesterday, but it is more interested in the variation of

its sales of bread in a given period. Therefore, some observed values of the sales of bread for a month will be aggregated as symbolic data. In other circumstances, the original data sets are too large to be analyzed. Reduction of the data sets into a more manageably sized data set will result in symbolic data. For example, a computer requires more memory to invert a matrix than is needed to store that matrix; so aggregation of very large data sets is necessary.

Based on the construction, symbolic data on p random variables are p-dimensional hypercubes, or a Cartesian product of p distributions, while classical data on p random variables are expressed by single points in p-dimensional space $\mathbb{R}^p$. Therefore, a classical data value as a single point is a special case of a symbolic data value. Before we go into more details, we first need to establish some basic notations. Much of these definitions and examples are drawn from Billard and Diday (2006).

Notation and Definitions

Notation 2.1.1: Write $E = \{\omega_u, u = 1, ..., m\}$ as a set of m symbolic **concepts/categories**.

Notation 2.1.2: For the random variable Y_j, j = *1,...,p*, a classical value or realization on the individual i = *1,...,n*, is denoted by x_{ij} and a symbolic value or realization is denoted as ξ_{ij}. Therefore, for a classical variable, $Y_j(i) = x_{ij}$, while for a symbolic variable, $Y_j(i) = \xi_{ij}$.

If Y_j is measured on a category $\omega_u \in E$, then it could be written as $Y_j(\omega_u) = \xi_{uj}$, and the complete set as $\boldsymbol{\xi}$=(ξ_{uj}). If the domain of Y_j is $\mathcal{Y}_j$, j = *1,...,p*, the matrix $\boldsymbol{\xi}$ takes values in $\mathcal{Y} = \mathcal{Y}_1 \times ... \times \mathcal{Y}_p$. A symbolic observation represents the set of individuals that satisfy the description of the associated symbolic concept, or category, ω_u, u = *1,...,m*.

Then, we go further to have some formal definitions.

Definition 2.1.1: Let the random variables Y_j, j = 1,...,p, have domain $\mathcal{X} = \mathcal{Y}_1 \times ... \times \mathcal{Y}_p$. Then, the point $\boldsymbol{x} = (x_1, ..., x_p)$ in $\mathcal{X}$ is called a **description vector**.

Definition 2.1.2: Let $D_j \subseteq \mathcal{Y}_j$ be a subset of $\mathcal{Y}_j$. The p-dimensional subspace $D = (D_1, ..., D_p) \subseteq \mathcal{X}$ is called a **description set**. If $D = D_1 \times ... \times D_p$ is the Cartesian product of the sets D_j, the D is called a **Cartesian description set**.
The combination of description vectors and description sets is called a description d.

Definition 2.1.3: To link specific descriptions to the corresponding random variable, we define a **relation** $\boldsymbol{R}$. It is written as YRd for description vector d, or YRD for the description set D.
Symbolic objects are generally constructed by observing whether a given relationship is true or not.

Definition 2.1.4: A **symbolic object** is $s = (a,\ R,\ d)$ where a is a mapping from $\Omega = \{1, ..., n\} \rightarrow L$ depending on the relation R and the description d. If L=$\{0, 1\}$, a is a binary mapping and s is a Boolean symbolic object. If L=[0,1], then s is a modal symbolic object. A symbolic object is a mathematical model of a concept.

Types of Symbolic Data

Symbolic values can be multi-valued, interval-valued, or modal-valued, among others. Multi-valued and interval-valued data are the most commonly considered in the literature to date.

Definition 2.1.5: A **multi-valued** symbolic random variable Y takes one or more values from the list of values in its domain $\mathcal{Y}$.

Table 2.1: Bird colors.

ω_u	Bird	Major Colors
ω_1	Magpie	{black, white}
ω_2	Kookaburra	{brown, black, white, blue}
ω_3	Galah	{pink, gray}
ω_4	Cardinal	{red, black}
ω_5	Goldfinch	{black, yellow}
ω_6	Quetzal	{red, green, white}

Example 2.1: Let us consider the birds example we mentioned at the beginning of this section. Table 2.1 consists of 8 types of birds and their colors. If the random variable of interest is "color" (Y) and the population is "Species of bird", we could have

$$Y(\omega_1 = color\ of\ magpie) = \{black, white\}$$

or

$$Y(\omega_5 = color\ of\ goldfinch) = \{black, yellow\}.$$

In each case, the actual realization is a subset of possible colors from the domain $\mathcal{Y} = \{list\ of\ colors\}$.

Definition 2.1.6: An **interval-valued** symbolic random variable Y takes values in an interval, which can be closed or open at either end. We usually assume that the interval observation is uniformly spread within the intervals.

Example 2.2: Suppose we have some demographic information for some patients in a hospital, like age, weight, race, gender and so on. Suppose we would like to learn about the

ages of the white females in the hospital. Then, the variable Y=age could be:

$$Y_{(age)}(white\ females) = [11, 87]$$

where 11 is the minimum age of white females in the hospital and 87 is the maximum.

Definition 2.1.7: A particular outcome of a random variable Y is **modal-valued** if it takes the form $Y(\omega_u) = \xi_u = \{\eta_k, \pi_k; k = 1, ..., s_u\}$ for an observation u; where π_k is a non-negative measure associated with η_k and where s_u is the number of values actually taken from the domain $\mathcal{Y}$. The measures $\{\pi_k\}$ are typically weights, probabilities, relative frequencies, and the like, corresponding to the respective outcome cluster η_k. They are the support of η_k in $\mathcal{Y}$. The outcome can be a distribution, a histogram, a model, or related stochastic entity.

Example 2.3: Consider again the example of the demographic information of the patients in Example 2.2. This time we would like to learn about the marriage status of white males. Then, the modal-valued variable will be:

$$Y(white\ males) = \{married, \frac{2}{3}; single, \frac{1}{3}\}.$$

Modal-values observations could also be specified distributions. They could be models.

Example 2.4: Table 2.2 shows the distribution of different types of energy consumption for 50 states of USA. (These data are based on rescaled values from US Census Bureau (2004) at www.census.gov) These distributions could either be empirical distributions (values of parameters estimated from data) or known distributions with known values of parameters. Therefore, according to the table, taking petroleum consumption for example, petroleum could follow the normal distribution with mean $\mu = 76.7$ and standard deviation $\sigma = 92.5$.

Table 2.2: Distribution of energy consumption.

ω_μ	Type	μ	σ
ω_1	Petroleum	76.7	92.5
ω_2	Natural gas	45.9	69.5
ω_3	Coal	47.0	43.3
ω_4	Hydroelectric power	6.4	14.3
ω_5	Nuclear power	25.4	20.4

Definition 2.1.8: An outcome of the random variable Y which can take values on a finite number of non-overlapping intervals $\{[a_k,\mathrm{b}_k), \mathrm{k} = 1, 2,...\}$ with $a_k \leq \mathrm{b}_k$ is an **histogram interval-valued** random variable. It takes the form:

$$Y(\omega_u) = \xi_u = \{[\mathrm{a}_{uk},\mathrm{b}_{uk}),p_{uk}; k = 1, ..., s_u\}$$

where s_u is a finite number of intervals forming the support for the outcome $Y(\omega_u)$ for observation ω_u, and p_{uk} is the weight for the particular subinterval $[\mathrm{a}_{uk},\mathrm{b}_{uk})$ with $\sum_{k=1}^{s_u} p_{uk} = 1$.

Example 2.5: Suppose in the patients information of Example 2.2, we are interested in level of cholesterol. Specifically, those whose cholesterol level is 240 or greater are at risk of heart disease, those with a level between 200 to 239 are borderline at risk, while those whose level is less than 200 are not at risk. Therefore, our interest is to study the histogram distribution of cholesterol values over these three intervals. The cholesterol levels for white males could be:

$$Y_{(cholesterol)}(white\ males) = \{[< 200), \frac{4}{9}; [200, 239), \frac{4}{9}; [\geq 240), \frac{1}{9}\}$$

which means that 4/9 of the white males in the hospital are at the borderline at risk and 1/9 are at risk of heart disease while 4/9 are not at risk.

In symbolic data analysis, histogram interval-valued observations could be specified distributions. The distributions may come from common distributions such as exponential, normal, and so on, or from empirical distributions with the parameter values estimated from the data.

Definition 2.1.9: The values of a **modal multi-valued** variable are a subset of a multi-valued random variable Y with domain $\mathcal{Y}$. Each of the values of the subset is attached with a non-negative measure. Therefore, a particular observation, for the category ω_u, takes the form:

$$Y(\omega_u) = \xi_u = \{\eta_{u1}, p_{u1}; ...; \eta_{us_u}, p_{us_u}\}, u = 1, ..., m,$$

where $\{\eta_{u1}, ..., \eta_{us_u}\} \subseteq \mathcal{Y}$ and $\sum_{k=1}^{s_u} p_{uk} = 1$.

Example 2.6: A company wishes to investigate 1000 potential clients' opinions on a product. The clients were asked to give an answer from the choices of Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree. Then, we may have the aggregated results:

$$Y\,(Product) = \{Strongly\ Agree, 0.3; Agree, 0.4; Neutral, 0.15;$$

$$Disagree, 0.12; Strongly\ Disagree, 0.03\}.$$

Thus, we have 70% of the 1000 clients Agree or Strongly Agree that they like the product.

Comparison With Classical Data

Since the value of a classical data point is a single point, it is a special case of a sym-

bolic value. In addition, a symbolic-valued observation is always attached to a category or a concept ω in the sample (or population) E of m concepts. In this way, the classical-valued observation could be treated as the realization of an individual i in the sample (or population) Ω of n individuals. Therefore, symbolic data are inherently different from classical data. We can see that symbolic data have internal variation within each observation and also variation between observations. In addition, to maintain the integrity of the original data, sometimes we need to add a rule when aggregating a larger dataset. It is important to pay attention to the differences between classical and symbolic data.

Example 2.7: Let us consider the credit card dataset of Table 2.3, which is a partial list of credit card expenditures for different purposes (Food, Social, Travel, Gas and Clothes) of individuals over a year. However, rather than the information of each expenditure by each person, a researcher may be more interested in the use of the credit card of individuals in a specific period, say a month. Therefore, the person's spending record in a month is the concept /category according to their interest. For example, the credit card use by Tom in January for food is [23.28, 30.00], for social is [8.67,18.31], and for travel is [193.53,206.53]. These are interval-valued symbolic data. Instead of using symbolic data analysis technology, people usually use the mean or median values of the original observations. Then the information of credit card use by Tom in January for food will be 26.64, for social will be 13.49, and for travel will be 200.03. This will lose much of the information in the data, for example, the inner variation of these original observations.

In summary, we typically have three steps in analyzing symbolic data. The first is to determine the categories and/or concepts of interest in order to build symbolic datasets according to the research interest at hand. Second, if we have additional related classical or symbolic valued variables of interest, we need to augment the established dataset. Finally,

Table 2.3: Credit card dataset.

i	Name	Month	Food	Social	Travel	Gas	Clothes
1	Jon	February	23.65	14.56	218.02	16.79	45.61
2	Leigh	May	28.47	8.99	141.60	21.74	86.04
3	Jon	April	23.40	11.61	179.38	23.73	48.89
4	Jon	January	25.94	12.38	197.90	20.06	47.09
5	Tom	July	24.14	15.97	190.40	35.71	20.02
6	Tom	August	23.01	13.20	220.52	29.44	18.09
7	Tom	July	24.25	15.71	149.01	30.68	21.75
8	Leigh	July	30.86	9.55	193.14	24.26	95.68
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

we should learn knowledge from the symbolic dataset.

2.2 Overview of Major Clustering Methods

There is a saying that - "the cluster solution is not generalizable because it is totally dependent upon the variables used as a basis for the similarity measure". We know that no single clustering method is valid for all types of data.

Cluster analysis is an explorative analysis technique that tries to identify structures within the data set. The purpose of a cluster analysis is to place objects into groups, or clusters, suggested by the data, such that objects in a given cluster are similar to each other, and objects in different clusters are dissimilar. Fraley and Raftery (1998) suggest dividing the clustering methods into two main methods: hierarchical and partitioning methods. Hierarchical clustering methods produce a sequence of cluster partitions (e.g., see Figure 2.1 for an example from Galili, 2016). In contrast, partitioning clustering procedures produce sets of distinct non-overlapping cluster; (e.g., see Figure 2.2 for a simulated data set).

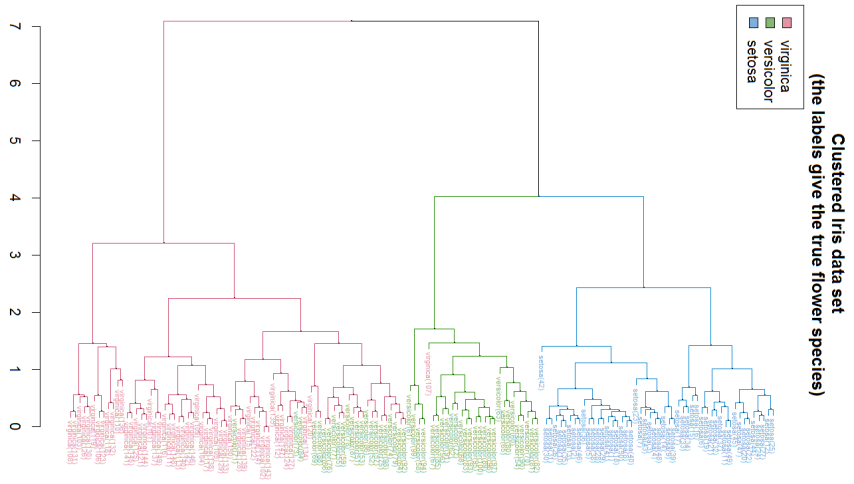

Figure 2.1: Example of Hierarchical Clustering.

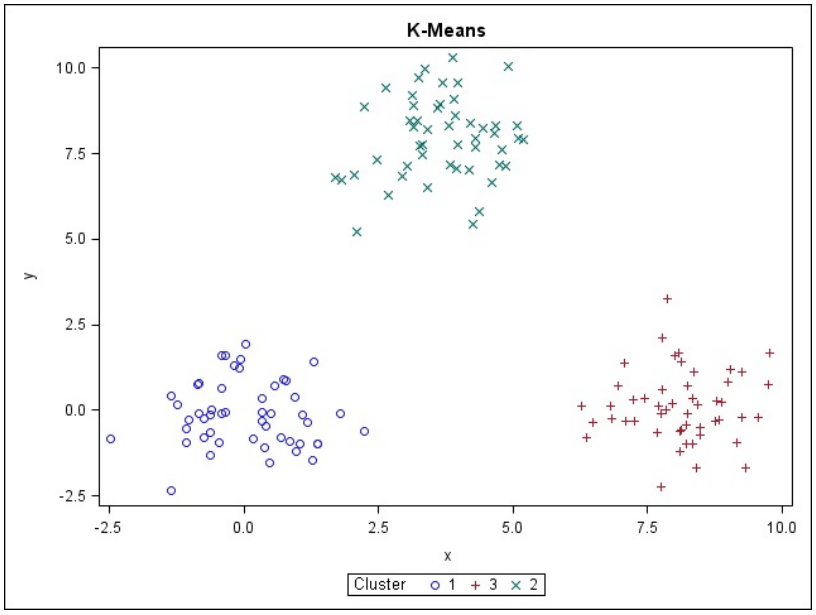

Figure 2.2: Example of Partitioning Clustering.

2.2.1 Hierarchical Clustering

Hierarchical methods construct the clusters by placing the observations in either a bottom-up or a top-down direction. We call them agglomerative hierarchical clustering and divisive hierarchical clustering, respectively. For agglomerative hierarchical clustering, each object initially represents a cluster of its own. Then clusters are merged until the desired cluster structure is obtained. For divisive hierarchical clustering, all objects initially belong to one cluster. Then the cluster is divided into sub-clusters. This process continues until the desired cluster structure is obtained.

A result of a hierarchical method is a dendrogram, representing the nested grouping of objects. A final result of clusters is obtained by cutting the dendrogram at the desired similarity level used in different methods.

The merging or division of clusters is performed according to some criteria which usually involves similarity measures, chosen so as to optimize some criterion (such as a sum of squares). Hierarchical clustering methods could be further divided according to the manner that the similarity measure is calculated.

Agglomerative clustering can be further classified as follows.

(1) **Single-linkage clustering** defines the distance between two clusters as the minimum distance between their members. This algorithm only wants separation, and does not care about compactness or balance.

(2) **Complete-linkage clustering** defines the distance between clusters as the maximum distance between their members. This method usually produces more compact clusters.

(3) **Average-link clustering** occurs when the distance between two clusters A and B is defined as the mean of all pairwise distances between items contained in A and B. It is a compromise between the sensitivity of complete-link clustering to outliers and the tendency of single-link clustering to form long chains. However, it may cause elongated clusters to split and portions of neighboring elongated clusters to merge.

(4) **Centroid clustering** defines the distance between two clusters as the Euclidean distance between their centroids. The cluster centroid is the middle of a cluster. A centroid is a vector containing the mean of a variable for the observations in that cluster.

(5) **Ward's minimum variance cluster analysis** defines the Ward's (1963) distance between two clusters as the difference between the total within cluster sum of squares for the two clusters separately, and the within cluster sum of squares resulting from merging the two clusters into cluster AB.

Hastie et al. (2009) has more details.

2.2.2 Partitioning Methods

Partitioning methods relocate observations by moving them from one cluster to another, starting from an initial partitioning, until some pre-specified criteria are optimally satisfied. Such methods typically require that the number of clusters will be pre-set. To achieve global optimality in partitioned-based clustering, an exhaustive enumeration process of all possible partitions is required. Because this is usually not feasible, certain heuristics are used in the form of iterative optimization. Namely, a relocation method iteratively relocates points between the K clusters. There are two broad approaches used.

(1) **Error minimization algorithms**

Error minimization algorithm methods tend to be computationally less intensive than other partitioning techniques. They aim at minimizing an error function which varies for different methods.

The most popular method of error minimization algorithm is the k-means algorithm (MacQueen, 1967). The main idea is to define K centers, one for each cluster. The better choice is to place them as much as possible as far away from each other. The next step is to take each point belonging to a given data set and associate it with the nearest center. When all the observations have been allocated, the first step is completed and an early group

allocation is done. At this point, we need to re-calculate the K new centroids as centers of the clusters resulting from the previous step. After we have these K new centroids, a new relocation has to be done between the same data set points and the nearest new centers. A loop has been generated. As a result of this loop, we may notice that the K centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function known as squared error function given by:

$$R=\sum_{k=1}^{K}\sum_{u=1}^{c_k}\|x_u-\gamma_k\|^2$$

where $\|x_u-\gamma_k\|$ is the Euclidean distance between x_u and the centers γ_k , c_k is the number of data points in the k^{th} cluster, *k=1,...,K*, with $\sum_k c_k = m$; K is the number of clusters.

(2) **Density-based methods**

Density-based clustering works by identifying "dense" clusters of points, which allows the identification of clusters of arbitrary shape and identify outliers in the data. It is based on nonparametric probability density estimates. In these methods, a dissimilarity measure is computed between all pairs of objects that are considered adjacent by some criteria. It has played a vital role in finding non-linear shapes structures based on the density.

We can see more in details from Han et al. (2012).

2.3 Model-based Clustering for Classical Data

The common heuristic clustering methods, such as k-means, decision trees, among others, have clear steps to finding clusters. However, they are largely exploratory and are not based on formal models. Model-based clustering is an alternative method.

2.3.1 Basic ideas of model-based clustering

The basic ideas of model-based clustering are as follows:

(1) Observe data for N p-dimensional objects, $\{X_{ij}, i = 1, ..., N, j = 1, ..., p\}$. Observations arise from a distribution that is a mixture of K components/clusters.

(2) Each component/cluster is described by a density function distributed around its center observation. Different components have different probabilities of appearing in the respective cluster. Therefore, each of them has an associated weight in the mixture.

(3) In principle, any distribution model could be used for the components. However, in order to manage the inference of unknown parameters, we typically assume that components follow a known distribution such as a p-dimensional multivariate normal (Gaussian) density.

(4) Thus, a mixture of multivariate normal distributions is the probability model for clustering. This is the model connection between the observations and the cluster memberships and parameters.

(5) Based on the mixture model, the methods find the values of the parameters that make the observations most likely to occur.

2.3.2 Model Set-up

We assume that the joint distribution is a mixture of K clusters, each of which is an independent multivariate normal density with cluster centers $\{\boldsymbol{u}_k, k = 1, ..., K\}$, covariance matrix $\{\boldsymbol{\Sigma}_k, k = 1, ..., K\}$, and associated probability of an object belonging to cluster k, $\{\pi_k, k = 1, ..., K\}$ and density function

$$f_k(x|\boldsymbol{u}_k, \boldsymbol{\Sigma}_k) = |\boldsymbol{\Sigma}_k|^{\frac{1}{2}} \exp\{-\frac{1}{2}(x - \boldsymbol{u}_k)^T \boldsymbol{\Sigma}_k^{-1}(x - \boldsymbol{u}_k)\},\ k = 1, ..., K.$$

The likelihood of the mixture model with K clusters is :

$$f(x|\pi, \boldsymbol{u}, \boldsymbol{\Sigma}) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k |\boldsymbol{\Sigma}_k|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_i - \boldsymbol{u}_k)^T \boldsymbol{\Sigma}_k^{-1}(x_i - \boldsymbol{u}_k)\}$$

where $0 < \pi_k < 1$, $\sum_{k=1}^{K} \pi_k = 1$.

Data generated by this mixture of multivariate normal densities are characterized by groups or clusters centered at the means μ_k, with increased density for points closer to the center. The corresponding surfaces of constant density are ellipsoidal. The $\boldsymbol{\Sigma}_k$ determines the geometric feature of the k^{th} cluster, $k = 1, ..., K$.

Model-based agglomerative clustering proceeds by successively merging pairs of clusters corresponding to the greatest increase in the classification likelihood among all possible pairs. When we do not have any information about the clusters, the procedure starts by treating each observation as a single cluster. McNicholas (2016) has more details for model-based clustering for classical data.

2.4 Introduction of Model-Based Clustering for Symbolic Data

2.4.1 Introduction

In classical data analysis, a clustering technique is one of the main vehicles of exploratory data mining and used in many fields. In some sense, it is well defined. However, the number of observations and the number of variables can be too large for contemporary computers to handle. In order to reduce the dataset into a more manageable size and also to retain enough information of the original dataset, the data are aggregated; thus, we obtain symbolic data. Therefore, we should establish new computational statistical approaches to group symbolic

types of data into classes. Here, we will focus on one model-based clustering method for distributions. This means that the observations in our study are distributions instead of single numerical values of points (classical data). To illustrate why we need to group distributions into classes and how these techniques can be used in practice, here is an example.

Consider the weather example of Vrac et al. (2012). The objective is to partition the weather world into well-defined temperature and humidity regions by latitude and longitude values including estimation of the underlying probability distribution function for each identified region. The temperature-humidity patterns are developed for every other latitude-longitude grid point. Hence, $m = 16,200$. At each of these m grid-points, the temperature distributions and humidity distributions are calculated from 37 and 24 altitude level values, respectively. The main idea is that the distributions characterize the variability of the temperature and humidity all along the vertical of the grid-point. This representation of the data is more informative than many of the classical representations, such as the average. Distributions can retain the within variation of each observation whereas the average does not.

We will describe a computational statistical approach of a new methodology to group distributions into classes. Decomposition of mixture models of probability distributions is performed to seek partitions, through a function of distributions and multi-dimensional copulas. This means we model the data as a mixture of distributions via an application of copulas.

Our objective is to partition this sample of m distributions into K classes. Similarly with model-based clustering analysis for classical data, we note the following:

(1) Sample observations arise from a distribution that is a mixture of K clusters. Each component/cluster is described by a density function. Since our observations are distribution

values, it is necessary to define the distribution function of distribution values.

(2) Different clusters have a different mixture probability. Therefore, each of them has an associated weight in the mixture.

(3) In model-based clustering analysis for classical data, we usually apply a p-dimensional multivariate normal (Gaussian) density to describe the joint distribution of p-dimensional variables. For model-valued symbolic data, we only have the marginal distributions; so, we apply the concept of copulas to model the dependency between the variables.

(4) Thus, a mixture of copulas is the probability model for clustering. This is the model connection of the observations and the cluster memberships and parameters.

(5) Based on the mixture model, our goal is to obtain the estimates of the underlying distributions associated with specific clusters, and the proportions of the observations in each cluster.

In the following paragraphs of this section, we will illustrate: (1) what is a mixture decomposition for probability distributions, by introducing two definitions of distribution functions of distribution values and joint distribution functions of distribution values; (2) how to model dependent distributions with multi-dimensional copulas; (3) the final model and the how to estimate it; (4) the plan for the next step.

2.4.2 Mixture Decomposition for Probability Distributions

Let $\boldsymbol{Y} = (Y_1, ..., Y_p)$ be a p-dimensional random vector taking values in $\mathbb{R}^p$ and let F^j be the distribution function associated with $Y_j, j = 1, ..., p$. Here, a distribution function is taken to be a cumulative distribution function. Then, we have a sample $\psi = (F_1, ..., F_m)$ of m p-dimensional distributions, where $F_u = (F_u^1, ..., F_u^p), u = 1, ..., m$, are realizations of a random vector with F_u^j being the realization of the distribution F^j for observation $u, u = 1, ..., m$. Each F_u belongs to $\Omega_F = \Omega_F^1 \times \cdots \times \Omega_F^p$. The Ω_F^j is the set of possible distributions to de-

scribe the individuals from Ω_F for the j^{th} variable and "$\times$" is the product of spaces operator.

Definition 2.4.2.1: **Distribution Function of Distribution Values**

The distribution function for distribution values is defined by:

$$G_z(x) = P(F(Z) \leq x), x \in \mathbb{R}, \tag{2.1}$$

where $F(Z)$ is a distibution function with the domain of Z corresponding to the domain of F and where $G_z : [0,1] \to [0,1]$.

Definition 2.4.2.2: **Joint Distribution Function of Distribution Values**

The joint distribution function for distribution values is defined by:

$$H_z(x_1, ...x_n) = P(F_u \in \psi; F_u(Z_1) \leq x_1, ..., F_u(Z_n) \leq x_n, u = 1, ..., m).$$

The function $G_z(x)$ is a distribution function of the random variable $F(Z)$ with the domain of [0,1]. The $H_z(x_1, ...x_n)$ is an n-dimensional joint distribution function of the random variable $F(Z_1, ..., Z_n)$ with marginal distributions $G_{Z_i}(x_i)$. It also has domain of [0,1]. Therefore, the properties that are defined for classical univariate and multivariate distributions are also appropriate for $G(.)$ and $H(.)$.

2.4.3 Modeling Dependent Distributions with Copulas

We first need to introduce the definition of copula functions. We first introduce subcopulas as a certain class of grounded 2-increasing functions with margins; then we will have copulas as subcopulas with domain $\mathbf{I}^2$. Further details can be found in Nelsen (2006).

Definition 2.4.3.1: A two-dimensional subcopula is a function C^* with the following prop-

erties:

1. Dom$C^* = S_1 \times S_2$, where S_1 and S_2 are subsets of $\mathbf{I}$ containing 0 and 1;

2. C^* is grounded and 2-increasing (see Definition 2.4.3.2 below);

3. For every u in S_1 and every v in S_2,

$$C^*(u,1) = u \text{ and } C^*(1,v) = v.$$

For every (u,v) in DomC^*, $0 \leq C^*(u,v) \leq 1$, so that RanC^* is also a subset of $\mathbf{I}$.

Definition 2.4.3.2: A two-dimensional copula is a 2-subcopula C whose domain is $\mathbf{I}^2$. Equivalently, a copula is a function C from $\mathbf{I}^2$ to $\mathbf{I}$ with the following properties:

1.(Grounded) For every u,v in $\mathbf{I}$,

$$C(u,0) = 0 = C(0,v) \tag{2.2}$$

and

$$C(u,1) = u \; and \; C(1,v) = v. \tag{2.3}$$

2. (2-Increasing) For every u_1, u_2, v_1, v_2 in $\mathbf{I}$ such that $u_1 \leq u_2$ and $v_1 \leq v_2$,

$$C(u_2,v_2) - C(u_2,v_1) - C(u_1,v_2) + C(u_1,v_1) \geq 0.$$

Sklar's Theorem

The most important theorem for copulas is Sklar's Theorem, Sklar (1959). Let H be an n-dimensional distribution function with unidimensional marginal distribution functions $F_1, ..., F_n$. Then, there exists a copula C such that, for all $x_1, ..., x_n$ in $\mathbb{R}^n$,

$$H(x_1, ..., x_n) = C(F_1(x_1), ...F_n(x_n)). \tag{2.4}$$

If $F_1, ..., F_n$ are continuous, the C is unique. Conversely, if $F_1, ..., F_n$ are distribution functions and C is a copula, the function H defined by the equation 2.4 is an n-dimensional distribution function with marginal distribution functions $F_1, ..., F_n$.

Archimedean Copulas

An Archimedean copula is an important class of 2-copula with a multivariate margin with attractive properties. We will introduce the definition here according to Nelson (2006).

Definition 2.4.3.2: Let φ be a continuous, strictly decreasing function from $\mathbf{I}$ to $[0, \infty]$ such that $\varphi(1) = 0$. The pseudo-inverse of φ is the function $\varphi^{[-1]}$ with $\mathrm{Dom}\varphi^{[-1]} = [0, \infty]$ and $\mathrm{Ran}\varphi^{[-1]} = \mathbf{I}$ given by:

$$\varphi^{[-1]}(t) = \begin{cases} \varphi^{-1}(t), 0 \leq t \leq \varphi(0), \\ 0, \qquad \varphi(0) \leq t \leq \infty. \end{cases} \tag{2.5}$$

Note that $\varphi^{[-1]}$ is continuous and nonincreasing on $[0, \infty]$, and strictly decreasing on $[0, \varphi(0)]$. Furthermore, $\varphi^{[-1]}(\varphi(u)) = u$ on $\mathbf{I}$, and

$$\varphi(\varphi^{[-1]}(t)) = \begin{cases} t, \qquad 0 \leq t \leq \varphi(0), \\ \varphi(0), \ \varphi(0) \leq t \leq \infty, \end{cases} = \min(t, \varphi(0)).$$

Finally, if $\varphi(0) = \infty$, then $\varphi^{[-1]} = \varphi^{-1}$.

Lemma 2.4.3.1: Let φ be a continuous, strictly decreasing function from $\mathbf{I}$ to $[0, \infty]$ such that $\varphi(1) = 0$, and let $\varphi^{[-1]}$ be the pseudo-inverse of φ defined by equation 2.5. Let C be the function from $\mathbf{I}^2$ to $\mathbf{I}$ given by

$$C(\mu, \nu) = \varphi^{[-1]}(\varphi\mu) + \varphi(\nu)). \tag{2.6}$$

Then C satisfies the boundary condition in equations 2.2 and 2.3 for a copula.

Copulas of the form of equation 2.6 are called Archimedean copulas. The function φ is called a *generator* of the copula.

Archimedean copulas are often investigated as families, parameterized by one-dimensional or two-dimensional parameters. The most frequently encountered parameterized families of Archimedean copula are the Clayton (1978), Frank(1979) and Gumbel (1958) families.

For example, the 2-dimensional Frank family is defined as: For $(\mu, \nu) \in [0,1]^n$,

$$C(\mu, \nu; \beta) = (\ln \beta)^{-1} \ln\{1 + [(\beta^{\mu} - 1)(\beta^{\nu} - 1)]/(\beta - 1)\})$$

where $\beta > 0$ and $\beta \neq 1$.

In order to model the dependencies between marginal distribution functions, we can use Sklar's theorem. Let $G_{Z_1}, ..., G_{Z_n}$ denote the distribution functions at the points $Z_1, ..., Z_n$, and let $H_{z_1,...,z_n}$ be the joint distribution function of these distributions. Then, there exists an n-copula C such that, for all $x_1, ..., x_n$, belongs to $\mathbb{R}^n$. Then, we can have the following form:

$$H_{z_1,...,z_n}(x_1, ..., x_n) = C(G_{Z_1}(x_1), ..., G_{Z_n}(x_n))$$

where C is uniquely determined on $RanG_{Z_1} \times \cdots \times RanG_{Z_n}$ for continuous G_{Z_i}, *i=1,...,n.*

2.4.4 Final Model and Estimation

After establishing the mixture models for distribution values and the copula to model dependencies between marginal distributions, we have the final K clusters mixture models for distributions:

$$H(x_1, ..., x_n; \boldsymbol{\gamma}) = \sum_{k=1}^{K} p_k C_k(G^k_{z_1}(x_1; \boldsymbol{b}^k_1), ..., G^k_{z_n}(x_n; \boldsymbol{b}^k_n); \boldsymbol{\beta}_k) \tag{2.7}$$

where $\boldsymbol{\gamma}$ is the parameter associated with $H(.)$, $\boldsymbol{\beta}_k$ is the parameter of the copula $C_k(.)$, and $\boldsymbol{b}_1^k, ..., \boldsymbol{b}_n^k$ are the parameters associated with the marginal distributions $G_{z_1}^k(.), ..., G_{z_n}^k(.)$, respectively. Note that in any case, any one of $H(.)$, $C_k(.)$ and $G^k(.)$ can be non-parametric.

Our goal is to seek the best grouping of the m distributions (observations) into K classes $P = (P_1, ..., P_K)$. There are many possible clustering criteria, such as the log-likelihood criterion. In this case, the number of classes K is usually pre-specified.

After we choose the clustering criteria, we need to:

(1) Estimate the parameters $\boldsymbol{b}_i^k$ of the marginal distributions $G_{z_i}^k(.)$, *i=1,...,n*; *k=1,...,K*.

(2) Fix the copula models $C_k(.)$, *k=1,...,K*.

(3) Estimate the associated parameters $\boldsymbol{\beta}_k$ when parametric copulas are chosen.

We address this process in detail in Chapter 3.

Chapter 3

Model-Based Clustering for Symbolic Data

3.1 General Methodology

In Chapter 2, we have introduced copula functions to describe the cumulative joint distribution function of random variables and established the final mixture model equation 2.7 for clustering of distributions. Our goal is to find the best partitions of the m observations (distributions) into K classes. Therefore, the main task is to estimate the mixture model of equation 2.7 that best represents the groups of input data. Chapter 3 will introduce how to estimate the mixture model equation 2.7 and the methods that will be used.

We will apply the partitional clustering method of the dynamic cluster algorithm (Diday and Simon, 1976). This method optimizes an adequacy criterion which expresses the best fitting between the clusters and their representation. It aims to obtain both a single partition of the observations and the identification of a prototype (e.g., center) for each cluster at each iteration. In our situation, in each iteration, the main job is to estimate the densities in the current step based on the given criterion.

There are many possible clustering criteria, $W(P, \boldsymbol{\gamma})$, where $\boldsymbol{\gamma}$ is the corresponding set of parameters. We will focus on two commonly used criteria.

Log-likelihood criterion (e.g., Symons, 1981) is:

$$W_1(P, \boldsymbol{\gamma}) = \sum_{u=1}^{N} \ln \left[\sum_{k=1}^{K} p_k h_k(F_u(Z_1), ..., F_u(Z_n); \boldsymbol{\gamma}_k) \right]$$

where $h_k, k = 1, ..., K$, is the cumulative density function of $F_u(Z_i), i = 1, ..., n$, and $p_k, k = 1, ..., K$, is the mixing probability.

Log-likelihood classification criterion (e.g., Celeux et al., 1989) is:

$$W_2(P, \boldsymbol{\gamma}^*) = \sum_{k=1}^{K} \sum\nolimits_{u \in p_k} \ln[h_k(F_u(Z_1), ..., F_u(Z_n); \boldsymbol{\gamma}_k)] \tag{3.1}$$

where $\boldsymbol{\gamma}^* = (\gamma_k, k = 1, ..., K)$. Equation 3.1 does not include the mixing probabilities $p_k, k = 1, ..., K$. It is more useful when we focus on seeking classes than on the whole density estimation, since it uses the distribution functions $h_k, k = 1, ..., K$, directly.

Vrac et al. (2012) has proved that the dynamical clustering algorithm of the log-likelihood classification criterion converges to a locally optimal solution in a finite number of iterations.

To illustrate, suppose we choose to use the log-likelihood classification criterion of equation 3.1, and suppose we apply a parametric copula. Then, we need to find the best partitions to maximize the function 3.1. From the model of equation 2.7 and the optimization criterion of equation 3.1, the final likelihood equation to be maximized should be:

$$\begin{aligned} L = \sum_{k=1}^{K} \sum_{u \in p_k} \Bigg[\left\{ \prod_{i=1}^{n} \frac{dG_{Z_I}}{dx_i}(x_i; \boldsymbol{b}_i^k) \right\} \times \frac{\partial^n}{\partial x_1 ... \partial x_n} \\ \times C_k(G_{Z_1}^k(x_1; \boldsymbol{b}_1^k), ..., G_{Zn}^k(x_n; \boldsymbol{b}_n^k); \boldsymbol{\beta}_k) \Big] . \end{aligned} \tag{3.2}$$

From equation 3.2, in order to estimate the parameters $\boldsymbol{b}_i^k$ and $\boldsymbol{\beta}_k$ with the proposed

maximum likelihood based method, it is required to first determine:

(1) the marginal distributions $G^k_{Z_i}(\cdot),\ i = 1, ..., n,\ k = 1, ..., K$. We will apply three different methods: Empirical Density Estimation, Kernel Density Function and Parametric Approach to estimate the marginal distribution functions.

(2) the copula models $C_k(\cdot),\ k = 1, ..., K$. Vrac et al. (2012) has proven that the Frank (1979) family of copulas is a good choice. We will apply three other frequently encountered parameterized families of Archimedean copula: Clayton (1978), Gumbel(1958) and Joe(1997) in Section 3.3.

3.2 Estimation of Marginal Distribution $G_z(x)$

Density estimation builds an estimate, based on an observed data sample, of some unobservable underlying probability density functions. The unobservable density function is considered as the density of a large population and the observed data are usually thought of as a random sample of the population. Density estimation can either be parametric where the data are from a known family, or nonparametric which is trying to estimate an unknown distribution. There are many approaches to determining the distribution functions and the corresponding density function. Here we will only introduce the empirical density estimation method and the details of some other frequently used methods are illustrated in Chapter 4.

3.2.1 Empirical Density Estimation

By applying the empirical density estimation method, the marginal distribution $G_z(x)$ can be estimated by extending the classical histogram approach to obtain the empirical frequency. More details can be found in Silverman (1986).

In Chapter 2, we introduced the distribution function of distribution values (Definition 2.4.2.1). In equation 2.1, if $G_Z(x)$ is empirically modeled from ψ, the distribution function

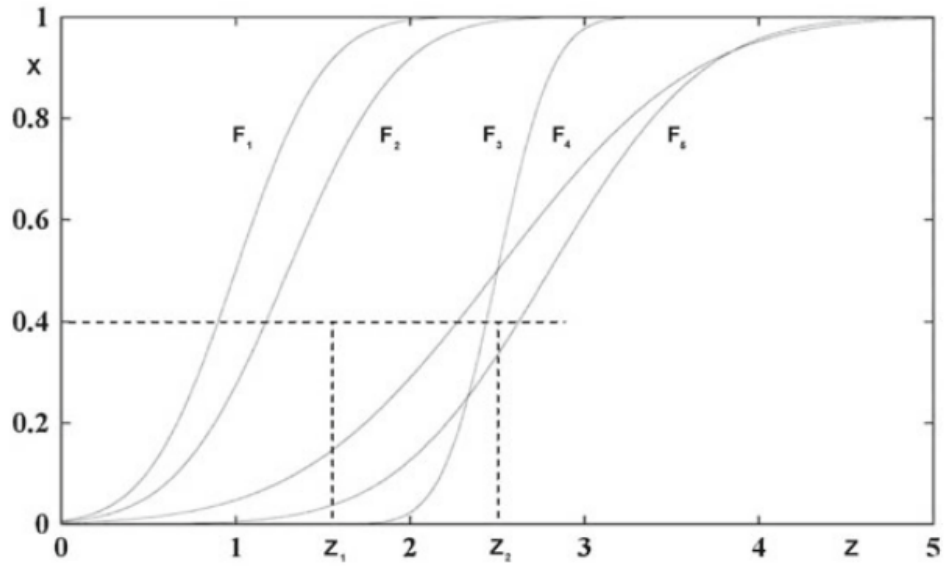

Figure 3.1: Observed Frequency Distributions

is obtained by

$$\begin{aligned} G_Z^e(x) &= P(F_u \in \psi; F_u(Z) \leq x, u = 1, ..., m) \\ &= \frac{card(F_u \in \psi; F_u(Z) \leq x, u = 1, ..., m)}{card(\psi)}. \end{aligned} \tag{3.3}$$

Figure 3.1 shows an example of $m = 5$ distributions $\{F_u, u = 1, ..., 5\}$ from Vrac et al. (2012). Therefore, in this example, $card(\psi) = 5$. Suppose we want to find the empirical distribution when x=0.4 (the horizontal dotted line in Figure 3.1). That is, we want to calculate the percentage of distributions whose values are smaller than or equal to 0.4 at point Z_i. For example, at point Z_1, the values of $F_3(Z_1)$, $F_4(Z_1)$ and $F_5(Z_1)$ are all smaller than 0.4. Therefore, we have $G_{Z_1}(0.4) = 3/5$. Similarly, we can have $G_{Z_2}(0.4) = 1/5$.

3.3 Estimation of the Copulas

3.3.1 Choice of Copula Function

We need to determine the form of copula functions $C_k(\cdot), k = 1, ..., K$, in the likelihood equation 3.2. The simplest one parameter copulas are given first. Then, we focus on four commonly used parameterized families of Archimedean copulas and two Elliptical copulas. We first give the 2-dimensional function of copulas and introduce the process of driving the multi-dimensional case. We also give the form of the derivatives to obtain the density functions in equation 3.2 for 2-dimensional Archimedean copulas.

The Simplest Copulas

The simplest copulas are defined with three parts: M, Π, W for the 2-dimensional case as:

$$M(\mu, \nu) = min(\mu, \nu), \quad \Pi(\mu, \nu) = \mu\nu, \quad W(\mu, \nu) = max(\mu + \nu - 1, 0). \tag{3.4}$$

The form of equation 3.4 is a special case of some commonly used parametric families of copulas. For example, the Clayton copulas is the special case with:

$$C_{-1} = W, \quad C_0 = \Pi, \quad C_\infty = M.$$

(The function of Clayton copulas is given by equation 3.6)

Archimedean Copulas

The Frank (1979) Family of Copulas

The 2-dimensional Frank family of copula is defined as: for $(\mu, v) \in [0, 1]^n$,

$$C(\mu, \nu; \beta) = (\ln \beta)^{-1} \ln\{1 + [(\beta^\mu - 1)(\beta^\nu - 1)]/(\beta - 1)\}) \tag{3.5}$$

where $\beta > 0$ and $\beta \neq 1$. The derivative is given by

$$\frac{\partial^2 C}{\partial \mu \partial \nu} = \frac{(\beta - 1)\beta^{\mu+\nu} \ln \beta}{[(\beta - 1) + (\beta^{\mu} - 1)(\beta^{\nu} - 1)^2]}.$$

The Clayton (1978) Family of Copulas

The 2-dimension Clayton family of copulas has the form of:

$$C(\mu, v; \beta) = [\max(\mu^{-\beta} + \nu^{-\beta} - 1, 0)]^{-1/\beta} \tag{3.6}$$

where $\beta \in [-1, 0) \cup (0, \infty)$. The derivative is given by

$$\frac{\partial^2 C}{\partial \mu \partial \nu} = (\beta + 1)(\mu v)^{-\beta-1}(\mu^{-\beta} + \nu^{-\beta} - 1)^{-2-1/\beta}.$$

The Joe (1997) Family of Copulas

The 2-dimension Joe family of copulas has the form of

$$C(\mu, \nu; \beta) = 1 - [(1 - \mu)^{\beta} + (1 - \nu)^{\beta} - (1 - \mu)^{\beta}(1 - \nu)^{\beta})]^{1/\beta} \tag{3.7}$$

where $\beta \in [1, \infty)$. The derivative is given by

$$\frac{\partial^2 C}{\partial \mu \partial \nu} = -\frac{(1 - \mu)^{\beta} \left(\left((1 - \mu)^{\beta} - 1 \right) (1 - y)^{\beta} - (1 - \mu)^{\beta} - \beta + 1 \right)}{(\mu - 1) \left(\left((1 - \mu)^{\beta} - 1 \right) (1 - \nu)^{\beta} - (1 - \mu)^{\beta} \right)^2 (\nu - 1)} \times$$
$$\left(- (1 - \mu)^{\beta} (1 - y)^{\beta} + (1 - \nu)^{\beta} + (1 - \mu)^{\beta} \right)^{\frac{1}{\beta}} (1 - \nu)^{\beta}.$$

The Gumbel (1958) Family of Copulas

The 2-dimension Gumbel family of copulas has the form of

$$C(\mu, \nu; \beta) = \exp[-((-\log(\mu))^{\beta} + (-\log(\nu))^{\beta})^{1/\beta}] \tag{3.8}$$

where $\beta \in [1, \infty)$. The derivative is given by:

$$\frac{\partial^2 C}{\partial \mu \partial \nu} = \frac{(-\ln(\mu))^{\beta} \, \mathrm{e}^{-\left((-\ln(\nu))^{\beta}+(-\ln(\mu))^{\beta}\right)^{\frac{1}{\beta}}} \left(\left((-\ln(\nu))^{\beta}+(-\ln(\mu))^{\beta}\right)^{\frac{1}{\beta}}+\beta-1\right)}{\mu \ln(\mu) \, \nu \ln(\nu)}$$

$$\times \left((-\ln(\nu))^{\beta}+(-\ln(\mu))^{\beta}\right)^{\frac{1}{\beta}-2} (-\ln(\nu))^{\beta}.$$

Elliptical Copulas

More details of elliptical copulas can be seen from Fang et al. (1987) and Cambanis et al. (1981).

The Gaussian Copulas

The 2-dimension Gaussian copulas have the form of

$$C(\mu, \nu) = \int_{-\infty}^{\Phi^{-1}(\mu)} \int_{-\infty}^{\Phi^{-1}(\nu)} \frac{1}{2\pi(1-\beta^2)^{1/2}} exp\{-\frac{s^2 - 2\beta st + t^2}{2(1-\beta^2)}\} ds \; dt \tag{3.9}$$

where β is simply the usual linear correlation coefficeint of the corresponding bivariate normal distribution.

The t-Copulas

The 2-dimension t-copulas have the form of

$$C(\mu, \nu) = \int_{-\infty}^{t^{-1}(\mu)} \int_{-\infty}^{t^{-1}(\nu)} \frac{1}{2\pi(1-\beta^2)^{1/2}} \{1 + -\frac{s^2 - 2\beta st + t^2}{\nu(1-\beta^2)}\}^{-(\nu+2)/2} ds \; dt \tag{3.10}$$

where β is similarly the linear correlation coefficient of the bivariate t-distribution.

Usually, it is difficult to define high dimension (especially for more than two variables) copula functions. However, an n-dimensional Archimedean copula, $C(v_1, ..., v_n)$, is a function from $[0, 1]^n$ to [0,1] satisfying

$$C(v_1, ..., v_n) = \phi^{[-1]}(\phi(v_1) + \cdots + \phi(v_n)). \tag{3.11}$$

According to Diday and Vrac (2005), from equation 3.11, $C_n(v_1, ..., v_n)$ satisfies

$$C_n(v_1, ..., v_n) = {\phi_n}^{[-1]}(\phi_n(C_{n-1}(v_1, ..., v_{n-1})) + \phi_n(v_n)) \tag{3.12}$$

where

$$C_{n-1}(v_1, ..., v_{n-1}) = {\phi_{n-1}}^{[-1]}(\phi_{n-1}(C_{n-2}(v_1, ..., v_{n-2})) + \phi_{n-1}(v_{n-1}))$$

and so on. Also, $0 \leq v_1, ..., v_n \leq 1$ and $\phi_i, i = 1, ..., n$, is a continuous strictly decreasing convex function. Therefore, copulas for $n > 2$ can be easily generated. In each iteration, $\phi_n(\cdot)$ and $\phi_{n-1}(\cdot)$ can have different values for the parameters. In terms of estimation, suppose $C_1(v_1, v_2; \beta_1)$ is the link function for variables V_1 and V_2 and $C_2(\cdot)$ is the link function for $C_1(\cdot; \beta_1)$ and V_3, written as

$$C_2(C_1(v_1, v_2; \beta_1), v_3; \beta_2).$$

We first find the realization of $C_1(\cdot; \beta_1)$ by estimating β_1 as

$$\{C_1(v_{11}, v_{21}; \hat{\beta}_1), ..., C_1(v_{1N}, v_{2N}; \hat{\beta}_1)\}.$$

Then, we estimate $C_2(v_1, v_2, v_3; \hat{\beta}_2)$ by estimating β_2. By doing this continuously, we can obtain the estimation of $C_n(v_1, ..., v_n; \hat{\boldsymbol{\beta}})$ with $\hat{\boldsymbol{\beta}} \equiv (\hat{\beta}_1, ..., \hat{\beta}_{n-1})$.

3.3.2 Estimation of the Copulas

After specifying the copula functions $C(.;\boldsymbol{\beta}_k)$ in Section 3.3.1, in order to estimate those parameters $\boldsymbol{\beta}_k$, it is necessary to maximize the term $l_k(\boldsymbol{\beta}_k)$ which is given by

$$l_k(\boldsymbol{\beta}_k) = \frac{\partial^n}{\partial\omega_1^k ... \partial\omega_n^k} C_k(G_{Z_1}^k(x_1; b_1^k), ..., G_{Zn}^k(x_n; b_n^k); \boldsymbol{\beta}_k)$$

where $\omega_i^k \equiv G_{Z_i}^k(x_i; b_i^k), i = 1, ..., n$.

Some numerical methods will be used, if explicit expressions for $\hat{\boldsymbol{\beta}}_k$ cannot be obtained. We will focus on the Newton-Raphson method. For example, at each iteration $s, s = 1, 2, ...$, for the 2-dimensional Archimedean copula with $\boldsymbol{\beta}_k = (\boldsymbol{\beta}_k^1, \boldsymbol{\beta}_k^2),\ k = 1, ..., K$, the iterative relationship will be

$$\boldsymbol{\beta}_k^{s+1} = \boldsymbol{\beta}_k^s + [I(\hat{\boldsymbol{\beta}}_k)]^{-1} grad(\boldsymbol{\beta}_k^s),\ k = 1, ..., K,$$

where the information matrix is

$$I(\boldsymbol{\beta}) = (\frac{-\partial^2 l_k(\boldsymbol{\beta}_k^1, \boldsymbol{\beta}_k^2)}{\partial\boldsymbol{\beta}_k^i \partial\boldsymbol{\beta}_k^j}),\ i, j = 1, 2,\ k = 1, ..., K,$$

and the gradient vector is

$$grad(\beta_k) = \frac{\partial l_k(\beta_k^1, \beta_k^2)}{\partial\beta_k^i},\ i = 1, 2,\ k = 1, ..., K.$$

Then, for n-dimensional copula functions, we can also apply the relationship of equation 3.12. Take $\omega = (\omega_1, ..., \omega_3)$ with 3 variables $Y = (Y_1, ..., Y_3)$ for example. Let us define the copula $C_1(\omega_1, \omega_2; \beta_1)$ as the link function between variables Y_1 and Y_2 and $C_2(.; \beta_2)$ as the

link function between $C_1(.;\beta_1)$ and the variable Y_3. Then, we have

$$C_2(C_1(\omega_1, \omega_2; \beta_1), \omega_3; \beta_2).$$

With respect to estimation, we can first estimate β_1 in $C_1(.)$ and compute realizations of $C_1(.)$, then move forward to estimate β_2. With these continuing steps, we can have any multi-dimension copula functions.

Chapter 4

Density Estimation and Copula Estimation

4.1 Density Estimation

Density estimation builds an estimate, based on an observed data sample, of some unobservable underlying probability density function. The unobservable density function is considered as the density of a large population and the observed data are usually thought of as a random sample of the population. Density estimation can either be parametric, where the data are from a known family of distributions; or nonparametric, which is trying to estimate an unknown distribution. We will start with a parametric method and then follow with three commonly used nonparametric estimation methods: histogram method, kernel density estimation and penalized likelihood approaches.

4.1.1 Parametric Density Estimation

If we have prior knowledge of the underlying distribution of a random variable, then based on the known form of the distribution function, the density estimation problem turns into a

parameter estimation problem. The standard method of parameter estimation is maximum likelihood estimation.

In maximum likelihood estimation, the main idea is to select the most likely parameter values that achieve the largest likelihood of the observed data. This is also saying that the method is maximizing the probability of obtaining the sample that has actually been observed. Suppose the unknown parameters are $\boldsymbol{\theta}$. The parameters are unknown but fixed. They completely defined the density function with n independent observations as

$$L(\boldsymbol{X} = (x_1, ..., x_n)|\boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i|\boldsymbol{\theta}).$$

Then, the maximum log-likelihood estimator is:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{argmax}(ln \prod_{i=1}^{n} f(x_i|\boldsymbol{\theta})) = \underset{\boldsymbol{\theta}}{argmax}(\sum_{i=1}^{n} lnf(x_i|\boldsymbol{\theta})).$$

The family of Gaussian densities parametrized by $\boldsymbol{\theta} = (\mu, \sigma)$ has the form of:

$$f(x; \boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \ -\infty < x < +\infty,$$

and is the most popular parametric function to be used.

Another choice is the Dirichlet's law. This is a multivariate generalization of the beta distribution. Therefore, it has the form of, for the multivariate case,

$$f(x; \boldsymbol{b}) = \frac{x^{\alpha_1 - 1}(1-x)^{\alpha_2 - 1}}{\int_0^1 y^{\alpha_1 - 1}(1-y)^{\alpha_2 - 1} dy}, \ 0 < x < 1,$$

where $\boldsymbol{b} = (\alpha_1, \alpha_2)$ are parameters with $\alpha_i > 0, \ i = 1, 2$. Therefore, the marginal distribution

can be determined by

$$G(x; \boldsymbol{b}) = \int_0^x f(t; \boldsymbol{b}) dt.$$

In parametric estimation, $f(x)$ is assumed to follow some parametric distribution and then a frequent approach is to use the maximum likelihood method to estimate the unknown parameters. The problem is if we do not have adequate knowledge of the appropriate assumptions for the underlying distribution, when a bad assumption may lead to a totally different estimation of the true density function. Therefore, various nonparametric methods are good choices in this case.

4.1.2 Histogram

The histogram (Silverman, 1986) is the oldest and most widely used method of density estimation. The main advantages are its extreme simplicity and speed of computation. A histogram is an accurate representation of the distribution of numerical data. It uses a kind of bar plot to represent the probability distribution of a continuous variable. The total area under a histogram is normalized to 1. The most important step of constructing a histogram is to divide the range of the continuous variable into a series of intervals, which are called bins. Therefore, it is not smooth and can be very sensitive to the choice of bins. For a random sample with n observations, the histogram estimate has the form as

$$\hat{f}(x) = \frac{number\ of\ bins}{nh}$$

where h is the width of the bins.

Scott (1985) proposed a simple improvement to the classic histogram using an average shifted histogram (ASH). This method takes m shifted histograms with explicit bin intervals with width h and origins of $0, \frac{h}{m}, \frac{2h}{m}, ..., \frac{(m-1)h}{m}$. It estimates the density as the arithmetic

mean of these m shifted histograms estimates as

$$\hat{f}_{ASH}(x) = \frac{1}{m}\sum_{i=1}^{m}\hat{f}_i(x).$$

Another form of weighted average shifted histogram (ASH) is to weight bins closer to the data more highly and use all bins to estimate the density at each point as

$$\hat{f}_{weightedASH}(x) = \frac{1}{m}\sum_{i=1}^{m\times n} W(l_i - x)\hat{c}_i(x)$$

where W is a weighting function, l_i is the center of bin i, $i = 1, ..., m \times n$, and $\hat{c}_i(x)$ is the number of points in the corresponding bin. This provides a bridge between the histogram and the advanced kernel methods (see Scott, 2009).

The ASH is smoother than the classic histogram and not very sensitive to the choice of origin. It also has a better visual interpretation, better approximation and still is computationally efficient.

4.1.3 Kernel Density Estimation

The histogram method is piecewise constant, which is discrete and not smooth. By centering a smooth kernel function at each data point, the kernel density estimation method is improved. It sums all the smooth kernel functions to estimate the underlying density of a random variable of size n as

$$\hat{f}_{kernel}(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{4.1}$$

where $K(.)$ is the *kernel* and h is the *bandwidth*, which is also called the smoothing parameter (Scott, 1997). According to equation 4.1, the kernel estimator is a mixture of n densities where each density centers on a data point x_i. The bandwidth h is the most important

feature in kernel estimation. It produces smooth estimates when it is large and can be overestimated when it is small. The effect of varying the bandwidth is illustrated in Figure 4.1. Three different bandwidths, $h = 0.1,\ h = 0.3,$ and $h = 0.6$, are used for kernel estimates of 200 simulated data from a bimodal density (see Silverman, 1986).

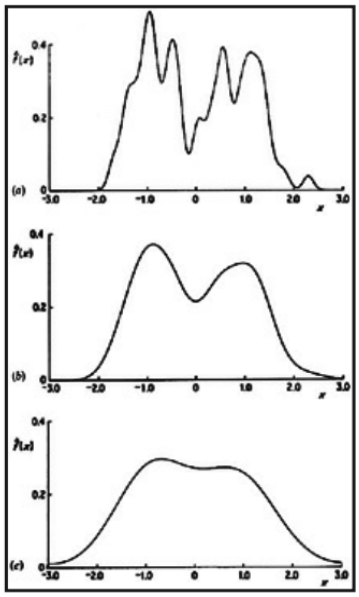

Figure 4.1: Kernel estimate with bandwidth (a) 0.1 (top); (b) 0.3 (middle); (c) 0.6 (bottom).

From the geometric perspective, the kernel function $K(.)$ determines the shape of each density curve while the bandwidth h determines their width. Figure 4.2 shows the individual density curve at each sample point and the overall estimate density by adding them up.

One way to select the value of the bandwidth h is by minimizing the average of the integrated squared-error (IMSE) loss function as

$$ISE(h) = \int [\hat{f}_h(x) - f(x)]^2 dx = \int \hat{f}_h(x)^2 - 2\int \hat{f}_h(x) f(x) dx + \int f(x)^2 dx.$$

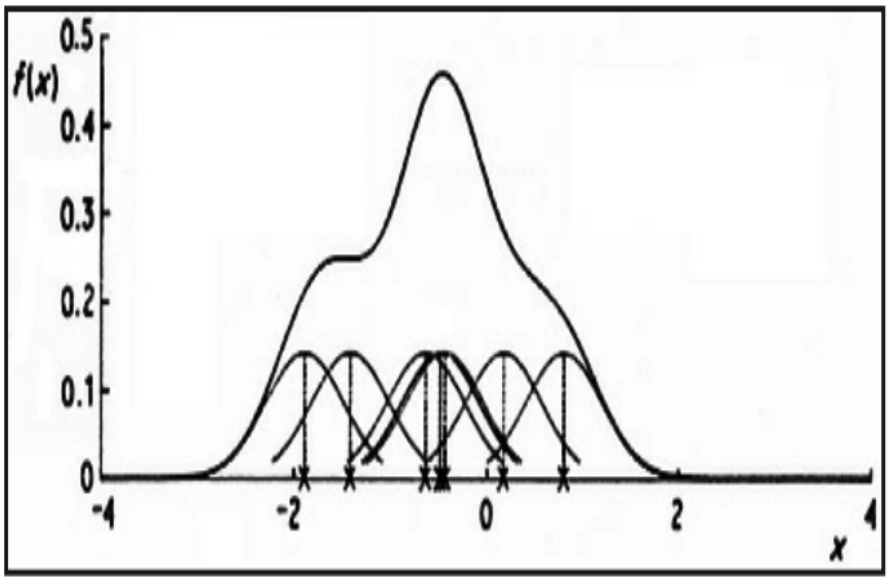

Figure 4.2: Kernel Estimate (from Silverman, 1986).

However, since $f(x)$ is to be estimated, it is not easy to implement this optimization process. A general way of solving this problem is to use a resampling method, e.g., cross validation over a pool of candidates. Therefore, this method of selecting the bandwidth is called least-squares cross-validation. A more advanced way of varying the bandwidths for different regions of the data according to the local sparseness can be found in Terrell and Scott (1992). The kernel is a symmetric function which is integrated to one as

$$\int K = 1 \text{ and } \int xK = 0.$$

Therefore, any probability density that is square integrable can be select for the kernel. The Uniform, Gaussian, triangle, and cosine fucntions are some commonly used kernel functions.

For a bivariate probability density function, the kernel estimator has a simple extension

using $K(x, y)$ as the kernel for size of n observations,

$$\hat{f}_{kernel}(x, y) = \frac{h_x h_y}{1} \sum_{i=1}^{n} K(\frac{x - x_i}{h_x}, \frac{y - y_i}{h_y})$$

where h_x and h_y are different smoothing parameters (bandwidth) for each coordinate direction. More details can be seen in Silverman (1986).

4.1.4 Penalized Likelihood Approaches

The penalized likelihood method was first proposed by Good and Gaskins (1971) and further developed by Silverman (1982), O'Sullivan (1988), Gu and Qiu (1993), and Gu (1993).

The maximum likelihood method is a standard statistical technique that is widely used in various problems. The *likelihood* for a set of independent identically distributed observations of size n with density function f is defined as

$$L(f|X_1, ..., X_n) = \prod_{i=1}^{n} f(X_i).$$

However, since the likelihood has no finite maximum over the class of all densities, it is not possible to use the maximum likelihood method directly for density estimation (Silverman, 1986). To illustrate this, suppose $\hat{f}_h$ is the naive density estimate with window width equal to half of the bandwidth h as $\frac{1}{2}h$; then for each i, we have

$$\hat{f}_h(X_i) \geq \frac{1}{nh}$$

and

$$\prod \hat{f}_h(X_i) \geq \frac{1}{n^n h^n} \rightarrow \infty \; as \; h \rightarrow 0. \tag{4.2}$$

Equation 4.2 means the likelihood can be made arbitrarily large by taking densities ap-

proaching a mixture of m density functions as

$$\hat{f}_{penalized}(x) = \frac{1}{m}\sum_{i=1}^{m} K(\frac{x-u_i}{h})$$

where the densities are not limited to being centered on the data points (compared to the kernel denisty estimation). Therefore, a penalized term should be added to restrict the densities over which the likelihood is to be maximized. One approach is to define a penalized term as

$$R(f) = \int_{-\infty}^{+\infty} (f^{''})^2.$$

Here, $R(f)$ is used to describe the roughness of the function f, where $f^{''}$ is the second derivative of f. Then, by adding the penalty term, the penalized log-likelihood is defined as

$$l_\alpha(f) = \sum_{i=1}^{n} \log f(X_i) - \alpha R(f) = \sum_{i=1}^{n} \log f(X_i) - \int_{-\infty}^{+\infty} (f^{''})^2$$

where α is a positive smoothing parameter. The smoothing parameter α controls the amount of smoothness in the density estimation. There are many available methods of selecting the smoothing parameter, such as the Akaike Information Criterion (AIC) and cross-validation.

4.2 Copula Estimation

4.2.1 General Methods for the Estimation of Copula Functions

The empirical estimation of copulas is a hard and tricky task. Referring to the introduction of copula functions in Chapter 2, we have learned that copulas include two main components: a multivariate cumulative joint distribution function and the marginal cumulative distribution function. One of the problems is that every marginal distribution of the underlying random variables should be evaluated and considered together with an estimated

multivariate joint distribution. Fermanian and Scaillet (2005) provided a discussion about the unexpected effects with respect to the estimating procedure for copulas. Therefore, it is important first to specify how to estimate separately the marginal distributions and the joint distribution. Same as for univariate density estimation, with different assumptions, the estimation of copula functions can be parametrically, semi-parametrically or non-parametrically determined. It is obvious that a known marginal distribution is so much more helpful on improving the results. However, if the marginal distributions are misspecified, the final estimated probabilities will be underestimated, especially in the tails (Charpentier and Segers, 2006).

It turns out that the main job of estimating copula functions is to set up the framework for the marginal cumulative distribution function (CDF) and the joint cumulative distribution function (JCDF). Let us consider the estimation of a p-dimensional copula C, which is

$$C(u) = F(F_1^{-1}(u_1), ..., F_p^{-1}(u_p))$$

where F denotes the JCDF and $F_j,\ j = 1, ..., p$, denotes all the marginal CDF's. Then, for each marginal distribution F, we have

$$F^{-1}(x) = \inf\{z|F(z) \geq x\}.$$

Suppose the observed sample for the p-dimensional random vector $\boldsymbol{X} = (X_1, ..., X_p)$ is $(\boldsymbol{X}_u), u = 1, ..., m$. Then, for each marginal CDF, we can apply any one of the univariate density estimation methods, such as the empirical density estimation, kernel density, parametric model estimation and so on. In the same way, the JCDF can be also modeled using either of the methods for the multivariate case. For example, by using the kernel method,

we have

$$F(x) = \frac{1}{m}\sum_{u=1}^{m} K(\frac{x - X_u}{h})$$

where the p-dimensional kernel K(.) is such that

$$K(x) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_p} K(.)$$

for every $x = (x_1, ..., x_p) \in \mathbb{R}^p$. For the parametric method, it is assumed that the JCDF follows a well-defined multivariate distribution indexed by a set of parameters $\boldsymbol{\beta}$. Therefore, without any prior knowledge of the parametric assumptions, there is no obvious discriminant between all these methods. However, from a different perspective, each estimation method has its own strengths and weakness. When any of the marginal CDFs or the JCDF are estimated empirically, it is difficult to have a nice visualization of the copula due to the discontinuous feature common to an empirical CDF. What is more, the inverse function of the marginal distribution does not have a unique solution. As long as there is one CDF estimated empirically, the associated copula density cannot be derived because of the non-differentiable copula function.

The kernel estimator largely depends on a smoothing parameter (e.g., the bandwidth h). This estimator also has worse performance when the dimension of the variables becomes larger in terms of convergence rates. When we have small dimensionality and a large sample size, it has a better performance. What is more, it is more intuitive to describe the underlying parametric distribution of the sample. Fermanian and Scaillet (2003) has detailed information about the theory when both the marginal distributions and the joint distribution are estimated by the kernel method.

For parametric model estimation, it is usual to have a parametric assumption for the marginal distribution. People are free to choose a parametric family for the joint cumulative distribution function (JCDF) and other marginal parametric families for each of the random

vector. It is not necessary for all the parametric families to be related to each other and they can be free of constraints. This is a way of the estimation procedure which is called semi-parametric: the copula is a function of some parameter $\boldsymbol{\gamma} = (\boldsymbol{\beta}, \boldsymbol{b}_1, ..., \boldsymbol{b}_p)$. It is known that we can obtain the copula density $c_{\boldsymbol{\gamma}}$ by taking the derivative of the copula function $C_{\boldsymbol{\gamma}}$ with respect to each of its arguments as

$$c_{\boldsymbol{\gamma}}(u) = \frac{\partial^p}{\partial_1 \cdots \partial_p} C_{\boldsymbol{\gamma}}(u)$$

where c_γ is the copula density with a set of parameters $\boldsymbol{\gamma}$. The estimator of $\boldsymbol{\gamma}$ can be given by maximizing the log-likelihood

$$\sum_{u=1}^{m} \log\, c_{\boldsymbol{\gamma}}(\hat{F}_1(x_{u1}), ..., \hat{F}_p(x_{up})).$$

4.2.2 Estimation of the Copula Density

After we have the estimated the distribution function $C_{\boldsymbol{\gamma}}$ of the copula, an estimate of the copula density function can be obtained by

$$c_{\boldsymbol{\gamma}}(u) = \frac{\partial^p}{\partial_1 \cdots \partial_p} C_{\boldsymbol{\gamma}}(u)$$

for every $u \in [0,1]^p$. However, this is only possible when the distribution function C is differentiable. This is true if the marginal distrbutions and the JCDFs are estimated parametrically. Unfortunately, if either of them is estimated empirically or by some other non-parametric ways of estimation, the final copula distribution is not differentiable. In this case, the nonparametric estimation procedures for the density of a copula function should be applied. Behnen et al. (1985) and Gijbels and Mielniczuk (1990) have proposed related theories. These theories depend on symmetric kernels and are not consistent with the boundaries

of $[0,1]^p$. Due to different sizes of the bandwidth h, the boundary bias can be significant. Charpentier and Segers (2006) proposed some solutions to deal with such issues.

Let us take a kernel-based estimator of 2-dimensional copulas $c(\mu,\nu)$ as an example. The density is based on pseudo-observations $(F_{X_1}(X_{1i}), F_{X_2}(X_{2i}))$ where F_{X_1},F_{X_2} are the empirical distribution functions

$$F_{X_1}(x) = \frac{1}{n+1}\sum_{i=1}^{n}\mathbb{I}(X_{1i} \leq x) \text{ and } F_{X_2}(x) = \frac{1}{n+1}\sum_{i=1}^{n}\mathbb{I}(X_{2i} \leq x)$$

where $F_{X_1}(X_{1i})$ and $F_{X_2}(X_{2i})$ are the ranks of the $X_{1i}^{'}s$ and the $X_{2i}^{'}s$ divided by $n+1$, which are

$$\left\{\frac{1}{n+1}, \frac{2}{n+1}, \cdots, \frac{n}{n+1}\right\}.$$

Instead of using n, the use of $n+1$ allows the avoidance of boundary problems (Charpentier and Segers, 2006). With diagonal bandwidths, the standard kernel-based estimators of the copula density $\hat{c}(\mu,\nu)$ based on pseudo-observations have the form

$$\hat{c}(\mu,\nu) = \frac{1}{nh^2}\sum_{i=1}^{n}K(\frac{\mu - F_{X_1}(X_{1i})}{h}, \frac{\nu - F_{X_2}(X_{2i})}{h}) \tag{4.3}$$

for a bivariate kernel $K(.)$: $\mathbb{R}^2 \to \mathbb{R}, \int K(.) = 1$. The variance of the estimator 4.3 is asymptotically normal at every point $(u,v) \in (0,1)$, i.e.,

$$\frac{\hat{c}(\mu,\nu) - E(\hat{c}(\mu,\nu))}{\sqrt{Var(\hat{c}(\mu,\nu))}} \to \mathbb{N}(0,1).$$

It is well known that the kernel estimation technique has a boundary bias which is underestimated around the boundaries. Charpentier and Segers (2006) shows the "ill" underestimation with a benchmark using the theoretical density of a Frank copula and the standard Gaussian kernel estimator based on pseudo-observations. Figure 4.3 is the plot of a Frank

copula with a Kendal tau equal to 0.5 and Figure 4.4 is the plot of the estimation density based on 1000 observations drawn from a Frank copula using a Gaussian kernel estimator. It is obvious to see the underestimation from the kernel density estimator when compared with the real Frank copula density. A more detailed theoretical explanation of this phenomenon is provided by Charpentier and Segers(2006).

For estimation in the univariate case, several techniques have been proposed to have a better performance on the borders estimation issue: mirror image modification, see Schuster (1985), Deheuvels and Hominal (1979); transformed kernels, (see Devroye and Gyorfi (1985), Wand et al. (1991); and boundary kernels, see Gasser and Muller (1979), Rice (1984), Muller (1991).

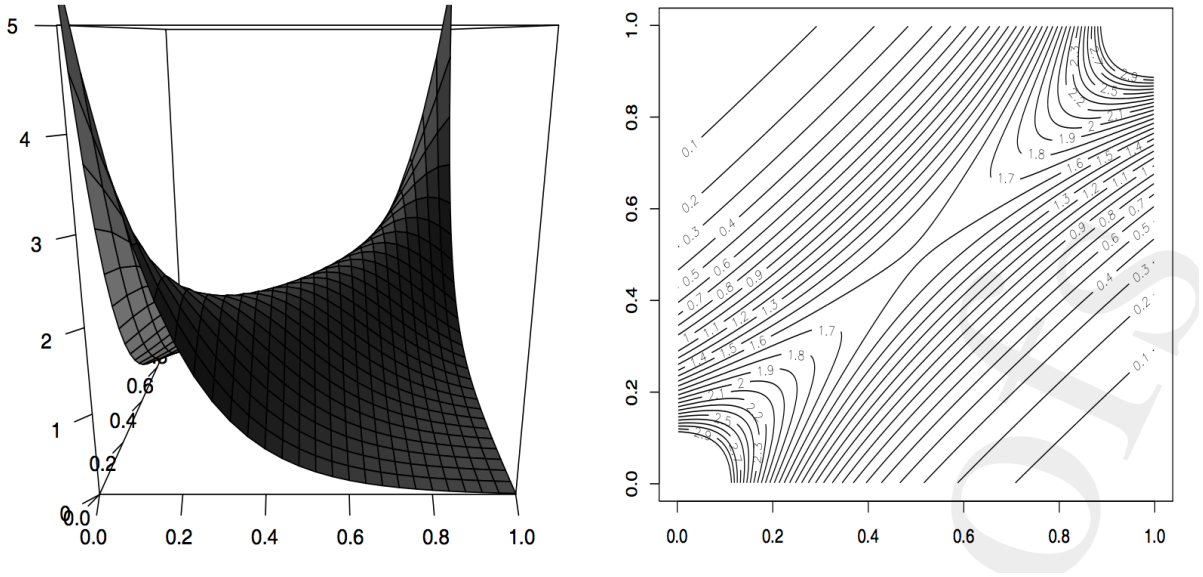

Figure 4.3: Density of Frank coplua (from Charpentier and Segers, 2006).

4.2.3 An Introduction of the R package *copula*

We have introduced general methods of density estimation for copula functions. It is important to apply an appropriate copula function to describe the dependence structure between

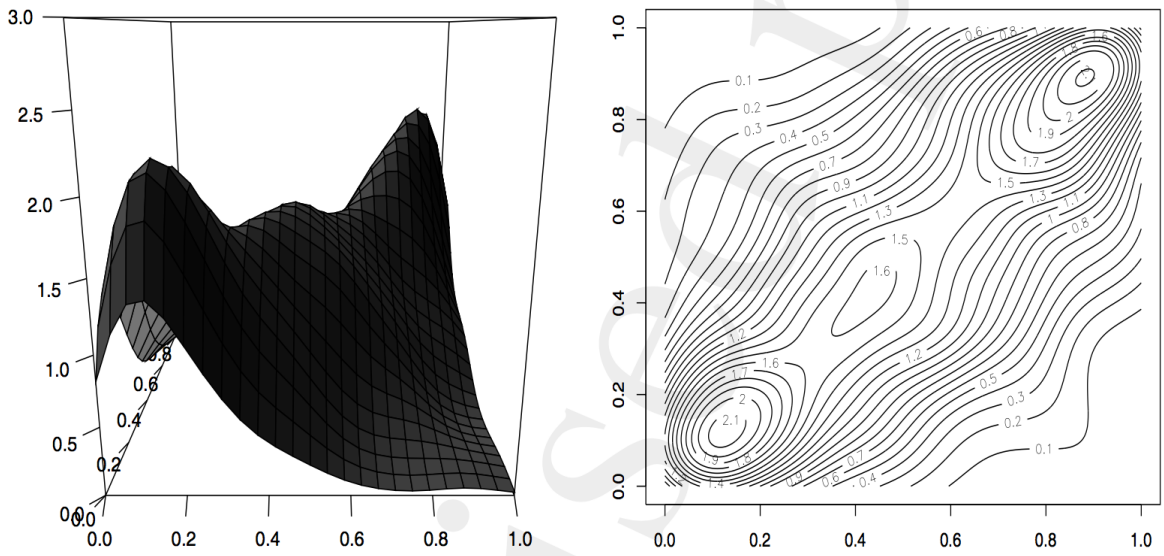

Figure 4.4: Estimation of copula density (from Charpentier and Segers, 2006).

the marginal distributions and to represent the joint distribution correctly. In this section, we will introduce a good open-source implementation of copulas, which is the R package *copula*. The R package *copula* is designed using the object-oriented features of the S language (Chambers, 1998). It is publicly available at the Comprehensive R Archive Network (CRSN, http:// CRAN.R-project.org/). The package provides an awesome designed and easily applicable platform for multivariate modeling with copulas in R. (Yan, 2007)

Classes

There are two main classes defined in the package: **copula** and **mvdc**. The **copula** class is used to define copulas and the **mvdc** class is used to define multivariate distributions via copula.

The *copula* Class

The two most frequently used copula families are Elliptical copulas and Archimedean copulas. Both Elliptical copulas and Archimedean copulas are defined in the packages.

An elliptical copula is a copula corresponding to an elliptical distribution by the Sklar's theorem. Fang et al. (1990) have a detailed discussion about elliptical distributions. In the copula package, the normal copula (normalCopula) and the t-copula (tCopula) are implemented for elliptical copula classes. They are specified by multivariate normal and multivariate t distributions. The dispersion structures are also well defined with different options: autoregressive of order 1 (ar1), exchangeable (ex), Toeplitz (toep) and unstructured (un). For example, in the case of three variables with p=3 , the corresponding dispersion matrices are:

$$\begin{pmatrix} 1 & \rho_1 & \rho_1^2 \\ \rho_1 & 1 & \rho_1 \\ \rho_1^2 & \rho_1 & 1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & \rho_1 & \rho_1 \\ \rho_1 & 1 & \rho_1 \\ \rho_1 & \rho_1 & 1 \end{pmatrix},$$

$$\begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_1 \\ \rho_2 & \rho_1 & 1 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} 1 & \rho_1 & \rho_2 \\ \rho_1 & 1 & \rho_3 \\ \rho_2 & \rho_3 & 1 \end{pmatrix}$$

where $\rho_j^{'}s$ are dispersion parameteres.

As we introduced in Chapter 2, an Archimedean copula is constructed with a generator φ as

$$C(u_1, ..., u_p) = \varphi^{-1}\left\{\varphi(u_1) + \cdots + \varphi(u_1)\right\}$$

where φ^{-1} is the inverse of the generator. Nelsen (2006) provides the details of generators for various Archimedean copulas, summarized in Chapter 2. In the *copula* package, one-parameter families, the Clayton copula, the Frank copula and the Gumbel copula are implemented. The functions implemented in R for distribution and density of a **copula**

object are **pcopula** and **dcopula**.

The *mvdc* Class

With given marginal distributions, the **mvdc** class is defined to build multivariate distributions from those marginal distributions. It has three major components: ***copula***, ***margins***, and ***paramMargins***, which specifies the copula function, the marginal distributions, and the list of parameter values, respectively. The common distributions designed with basic R functions can be used to specify the marginal distributions. Similarly, as for the object **copula**, the R functions for distribution and density for **mvdc** are **pmvdc** and **dmvdc**.

Fit a Copula Model

In this package, the functions **fitCopula** and **fitMvdc** provide the estimation of copula functions using three different methods: maximum likelihood estimate, inference functions for marginal distributions and a canonical maximum likelihood method. All of these methods are based on maximum likelihood estimates, which are used to select the candidate copulas that have the highest likelihood (Frees and Wang, 2005).

Maximum Likelihood Estimate

With the help of functions **loglikCopula** and **loglikMvdc** implemented in the package, the log-likelihood of the data under the specified copula functions can be evaluated. For a p-dimensional copula function C with density function c and marginal distributions G_j, density function $g_j, j = 1, ..., P$, the log-likelihood function is

$$l(\boldsymbol{\gamma}) = \sum_{i=1}^{n} \log\, c\left\{G_1(X_{i1}; \boldsymbol{b}_1), ..., G_p(X_{iP}; \boldsymbol{b}_P); \boldsymbol{\beta}\right\} + \sum_{i=1}^{n}\sum_{j=1}^{P} \log\, g_j(X_{ij}; \boldsymbol{b})$$

where $\boldsymbol{\gamma} = (\boldsymbol{\beta}, \boldsymbol{b})$ is the parameter vector to be estimated. The maximum likelihood estimator of $\boldsymbol{\gamma}$ is

$$\hat{\boldsymbol{\gamma}}_{ML} = \underset{\boldsymbol{\gamma} \in \boldsymbol{\Upsilon}}{argmax}\ l(\boldsymbol{\gamma})$$

where $\boldsymbol{\Upsilon}$ is the parameter space. To obtain $\hat{\boldsymbol{\gamma}}_{ML}$, the estimation of the marginal parameters separately for each margin will be used as an initial search for the starting values. Then, the evaluated log-likelihood will be optimized using the optimization routine *optim* in R base.

Inference Functions of Margins (Joe and Xu, 1996))

The inference functions for margins (IFM) method is a two-stage estimation method. It is called the IFM method because the optimality criteria are imposed on the functions instead of the estimators in the estimating equations. It is known that the number of parameters increases when the dimension increases. Therefore, the IFM method first estimates the marginal parameters by maximum likelihood estimation for each marginal distribution as

$$\hat{b}_{jIFM} = \underset{b_j}{argmax} \sum_{i=1}^{n} \log\ g_j(X_{ij}; \boldsymbol{b}_j),\ j = 1, ..., p,$$

and then, in the second step, it estimates the parameters for the copula functions with the estimated parameters in the first step as

$$\hat{\beta}_{IFM} = \underset{\boldsymbol{\beta}}{argmax} \sum_{i=1}^{n} \log\ c(G_1(X_{i1}; \hat{\boldsymbol{b}}_{IFM}), ...G_p(X_{ip}; \hat{\boldsymbol{b}}_{IFM}); \boldsymbol{\beta}).$$

By using a two-stage approach, the number of parameters to be estimated is much smaller in each maximization step. Each estimating equation is a score function from the marginal distributions, which is the partial derivative of a log-likelihood function. This method is also implemented with the **fitCopula** function in the package.

Canonical Maximum Likelihood

The canonical maximum likelihood (CML) method is used when the marginal distributions cannot be specified and a consistent estimation of the copula parameter $\boldsymbol{\beta}$ is important. Instead of using parametric estimation, the CML method transforms the sample observations into uniform variates using a nonparametric estimation of the CDF of the marginal distributions. This transformation is closely related to the empirical CDF. The transformed data are used as if they had uniform marginal distributions. Therefore, we call them pseudo-observations. Then, the parameter $\boldsymbol{\beta}$ is estimated by maximum likelihood as

$$\hat{\boldsymbol{\beta}}_{CML} = \underset{\boldsymbol{\beta}}{argmax} \sum_{i=1}^{n} \log\, c(U_{i1}, ..., U_{ip}; \boldsymbol{\beta})$$

where $U_{ij}, j = 1, ..., p$, are the uniform marginal distributions. This CML method is also defined with the **fitCopula** in the package.

Other Functions Defined in the Package

Random Number Generator

According to Sklar's Theorem, random numbers of a copula function can be generated by transforming each marginal distribution of a multivariate distribution with its probability integral transformation. A general way of generating variables from a copula function is to use a iterative conditioning method (Bouye et al., 2000). When the dimension becomes larger with $p > 2$, Marshall and Olkin (1988) and Frees and Valdez (1998) proposed fast algorithms for some commonly used Archimedean copulas. It is used when the inverse generator function φ^{-1} is known to be the Laplace transform of some positive random variable. The *Copula* package applies the R base function for a gamma variable generator and the Fortran implementation of Nolan (2006) to generate positive stable variables. When $p = 2$, negative association is allowed for Archimedean copulas, while only positive association is

defined when $p > 2$. The function defined for the random number generator in the package is **rcopula** for a **copula** and **rmvdc** for an **mvdc** object.

Graphics

Visualization is very important for presenting the results of copula functions. The package **scatterplot3d** can be used to create the 3D scatter plot. The functions **persp** and **contour** are defined in the **copula** package to describe density functions and distribution functions.

Chapter 5

Algorithms of Model Based Clustering for Distributional Data

5.1 Algorithms and Theories

We set out the basic theory and principles of the algorithms for model-based clustering for distributional data. Then, the algorithms are used in an extensive simulation study in Chapter 6. In Chapter 2 and Chapter 3, we established the model function for mixture decomposition with the application of copulas. The description of the set of units $E = \{\omega_1, ..., \omega_N\} \subset \Omega$ is the distributional data base $\{F_1, ..., F_N\}$. The $\{F_1, ..., F_N\}$ is a sample of N observations from a random variable Y such that $Y(\omega)\epsilon F$ for $\omega \in \Omega$. The p-dimensional joint distribution function is defined as $H(x, \boldsymbol{\gamma})$. The mixture decomposition problem of distribution functions of distrbutions becomes: given the number of clusters K, find a partition $P = (P_1, ..., P_K)$ of the distributional-data $\{F_1, ..., F_N\}$ and the value of the coresponding parameters $(\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_k)$ and the mixture ratios $(p_1, ..., p_K)$, such that

$$H = \sum_{k=1}^{K} p_k H_k(X, \boldsymbol{\gamma}_k)$$

where $\sum_{k=1}^{K} p_k = 1$. The parameters $\boldsymbol{\gamma_k} = (\boldsymbol{b_k}, \boldsymbol{\beta_k})$ where $\boldsymbol{b_k}$ is the set of paramters of the chosen distribution family for the marginal distribution $G_k(.)$ and $\boldsymbol{\beta_k}$ is the set of paramters of the chosen copula family model $C_k(.)$.

In order to estimate the paramteres, such that each class follows a joint distribution function $H_k(., \boldsymbol{\gamma}_k)$, we need to obtain the density function $h_k(., \boldsymbol{\gamma_k})$, with $h = \sum_{k=1}^{K} p_k h_k(X, \boldsymbol{\gamma_k})$ where $\sum_{k=1}^{K} p_k = 1$. However, there are many possibilities for the joint distribution function H. We should study carefully each possible combination of marginal distributions and copula functions. If the joint distribution function H is differentiable, $h(X) = H^{'}(X)$ and likewise, for $k = 1, ..., K$,

$$h_k(X, \boldsymbol{\gamma_k}) = H_k^{'}(X, \boldsymbol{\gamma_k}).$$

If H is not differentiable, we should find the approximation of the density function. We will illustrate the two cases separately in detail.

5.1.1 Joint Distribution Function When H is Not Differentiable

We know that the joint dstribution function of distributions H consists of two parts: the copula function $C(.)$ and the marginal distributions $G(.)$, see equation 2.7. Therefore, if one of them is non-differentiable, then the joint distritbuion function can not be differentiable. Here, we introduce the method of finding an approximation of the joint density function. It will be used when either the copula function or the marginal distribution is not differentiable or both of them are not differentiable.

Approximation of the joint density function

Following Vrac (2005), we define a mapping a which measures the fit between a distribution $Y(w) = F_w$ and a given 2-dimensional joint distribution function of distribution $H_{t_1,t_2} =$

$C(G_{t_1}, G_{t_2}) \in \mathbb{R}^+$, by setting a variable $a : \Omega \times [0,1]^2 \to \mathbb{R}^+$ with

$$a(\omega, \varepsilon) = [Y(\omega)R(\varepsilon)C(G_{t_1}, G_{t_2})] \in \mathbb{R}^+$$

where $\varepsilon = (\varepsilon_1, \varepsilon_2)$ is a threshold. Here, $x_i = F(t_i)$ and $R(\varepsilon)$ are defined by

$$F_\omega R(\varepsilon) H_{t_1,t_2} = V_{H_{t_1,t_2}}(a, b),$$

with $a = (x_1 - \varepsilon_1, x_1 + \varepsilon_1)$ and $b = (x_2 - \varepsilon_2, x_2 + \varepsilon_2)$. Then, we have

$$\begin{aligned} F_\omega R(\varepsilon) C(G_{t_1}, G_{t_2}) =& C(G_{t_1}(x_1 + \varepsilon_1), G_{t_2}(x_2 + \varepsilon_2)) - C(G_{t_1}(x_1 + \varepsilon_1), G_{t_2}(x_2 - \varepsilon_2)) \\ & - C(G_{t_1}(x_1 - \varepsilon_1), G_{t_2}(x_2 + \varepsilon_2)) + C(G_{t_1}(x_1 - \varepsilon_1), G_{t_2}(x_2 - \varepsilon_2)). \end{aligned} \tag{5.1}$$

From Definition 2.4.3.2, G_{t_1} and G_{t_2} are increasing; so we have

$$G_{t_i}(x_i + \varepsilon_i) \geq G_{t_i}(x_i - \varepsilon_i).$$

Proposition (Vrac, 2005)

Let h be the density function of the random variable $Y = (Y_{t_1}, Y_{t_2})$. Let $a(w, \varepsilon) \in [0, 1]$. If $\partial\varepsilon = 4\varepsilon_1\varepsilon_2$, then $h_\varepsilon(x) = a(w, \varepsilon)/\partial\varepsilon$ is an approximation of $h(x_1, x_2)$ and $h_\varepsilon(x) \to h(x)$ a.s. when $\partial\varepsilon \to 0$. (See Vrac, 2005 for a detailed proof.)

Then, we have

$$h_k(X; \boldsymbol{\gamma}_k, \varepsilon_k)\partial\varepsilon_k = a_k(\omega; \boldsymbol{\gamma}_k, \varepsilon_k) = [Y(\omega)R(\varepsilon_k)C_k(G^k_{t_1}(X_1; \boldsymbol{b}_{k_1}), G^k_{t_2}(X_2; \boldsymbol{b}_{k_2}), \boldsymbol{\beta}_k)]$$

where $X = (X_1, X_2) = (F_\omega(t_1), F_\omega(t_2))$, $F_\omega = Y(\omega), \boldsymbol{\gamma}_k = (\boldsymbol{b}_k, \boldsymbol{\beta}_k)$ with $\boldsymbol{b}_k = (b_{k_1}, b_{k_2})$, $\varepsilon_k =$

$(\varepsilon_{1k}, \varepsilon_{2k})$ and $\partial\varepsilon_k = 4\varepsilon_{1k}\varepsilon_{2k}$. Therefore, in the non-differentiable case, the mixture decomposition model becomes

$$h(X) = \sum_{k=1}^{K} p_k h_k(X, \boldsymbol{\gamma}_k, \varepsilon_k).$$

5.1.2 Joint Distribution Function When H is Differentiable

When the joint distribution function H is differentiable, the marginal distribution $G(.)$ and the copula function $C(.)$ are both differentiable. Then, we have

$$h(X) = \frac{\partial H(X)}{\partial X_1 ... \partial X_p} \quad \text{and} \quad h_k(X, \gamma_k) = \frac{\partial H_k(X; \boldsymbol{\gamma}_k)}{\partial X_1 ... \partial X_p}$$

where $\boldsymbol{\gamma}_k = (\boldsymbol{b}_k, \boldsymbol{\beta}_k)$. The mixture decompsition density equation becomes

$$h(X) = \sum_{k=1}^{K} h_k(x, \boldsymbol{\gamma}_k).$$

In the case of one variable, we have

$$H_{T_1,...,T_n}(X, \boldsymbol{\gamma}) = \sum_{k=1}^{K} p_k C^k(G^k_{T_1}(X, \boldsymbol{b}^k_1), ..., G^k_{T_n}(X, \boldsymbol{b}^k_n); \boldsymbol{\beta}_k),$$

and the associated density function becomes

$$\begin{aligned} h_{T_1,...,T_n}(X, \boldsymbol{\gamma}) = \sum_{k=1}^{K} p_k \frac{\partial^n H_k}{\partial \mu_1 ... \partial \mu_n} &= \sum_{k=1}^{K} \{ \frac{\partial G^k_{T_i}(X; \boldsymbol{b}^k_i)}{\partial X} \} \\ &\times \frac{\partial^n}{\partial \mu_1 ... \partial \mu_n} C_k(G^k_{T_1}(X; \boldsymbol{b}^k_1), ..., G^k_{T_n}(X; \boldsymbol{b}^k_n); \boldsymbol{\beta}_k) \end{aligned}$$

where n is the pre-defined dimension of T, $\mu_i = G_{T_i}(X), i = 1, ..., n$. In the case of $n = 2$, $G_T = (G_{T_1}, G_{T_2})$ and $H_{T_1,T_2} = C(G_T)$. Then, the derivative of the joint disribution function

of distributions H is the derivative of $C(G_T)$. This gives us

$$h(X) = H'_{T_1,T_2}(X) = \frac{\partial^2 H_{T_1,T_2}(X)}{\partial X \partial X} = \frac{\partial^2 C(G_{T_1}(X), G_{T_2}(X))}{\partial X \partial X}$$

$$= \frac{\partial G_{T_1}(X)}{\partial X} \times \frac{\partial G_{T_2}(X)}{\partial X} \times \frac{\partial^2 C(G_{T_1}(X), G_{T_2}(X))}{\partial \mu_1 \partial \mu_2}$$

where $\mu_1 = G_{T_1}(X)$ and $\mu_2 = G_{T_2}(X)$.

In the case of more than one variable, suppose we have P variables $\{Y^1, ..., Y^P\}$ and n-dimensional values $T_1^p, ..., T_n^p,\ p = 1, ..., P$. For each $\omega \in \Omega$, and for each $F_\omega = (F_\omega^1, ..., F_\omega^P)$, we have

$$H^p_{(T_1^p,...,T_n^p)}(X^p) = \sum_{k=1}^{K} p_k^p C_k^p (G^p_{T_1}(X^p), ..., G^p_{T_n}(X^p)),\ p = 1, ..., P,$$

where, for each $p = 1, ..., P$,

- $H^p_{(T_1^p,...,T_n^p)}$ is an n-dimensional joint distribution function of distributions at the point $(T_1^p, ..., T_n^p)$ for the variable Y^p;
- p_k^p is the k^{th} mixture ratio for the variable Y^p, $k = 1, ..., K$;
- C_k^p is the copula model of the cluster k for the variable Y^p,$k = 1, ..., K$.

From each unit $\omega_i,\ i = 1, ..., N$, we obtain a total of N pairs of P-dimensional probability values of the distributions $(H^1_{(T_1^1,...,T_n^1)}(X^1), ..., H^P_{(T_1^P,...,T_n^P)}(X^P))$. Then, a mixture decomposition of distributions by copulas should be applied on this new database again to obtain the final results in the case of more than one variable.

5.2 Algorithms for Solving the Symbolic Mixture Decomposition Problem

We will first illustrate the whole struture of the symbolic mixture decomposition process, which is the general partition steps of dynamic clusters (more details are given by Diday, et. al., 1974). Then, we will move forward to introduce the detailed partition process for each situation for one variable and then for more than one variable, and then when H is not differentiable and differentiable.

The general partition process is defined in two steps with the input of a set of units described by distributions, a given partition $(P_1, ..., P_K)$, a parametric or non-parametric copula family and optionally a parametric distribution family. The output is a partition and a copula model $C = (C_1, ..., C_K), k = 1, ..., K$, with estimated parameters for each class and optionally a distribution model G_k with estimated parameters for each class at each $T_j, j = 1, ..., n$. The criteria to be maximized is the log-likelihood classification criterion (Celeux et al., 1989) given in equation 3.1. This criterion does not include the mixing probability term and uses the distribution functions $h_k,\ k = 1, ..., K$, directly.

The partition process is formed as follows:

Step 1:

Initialize the parition as $P^0 = (P_1^0, ..., P_K^0)$.

Step 2:

Define the partition after the i^{th} iteration as $P^i = (P_1^i, ..., P_K^i)$. Then, the allocation step consists of two steps:

Step 2 (a):

Estimate the parameters $(\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_K)$ by maximizing the chosen criterion based on $P^{(i)}$, and obtain $P_k^{(i+1)}$ and $\boldsymbol{\gamma}_k^{(i+1)}$ for each cluster; and

Step 2 (b):

Obtain the new partition $\{P_k^{(i+1)}, k = 1, ..., K\}$, where $P_k^{(i+1)}$ is

$$P_k^{(i+1)} = \{F_u; p_k^{(i+1)} h_k(F_u; \boldsymbol{\gamma}_k^{(i+1)}) \geq p_v^{(i+1)} h_v(F_u; \boldsymbol{\gamma}_v^{(i+1)}) \ for \ all \ v \neq k, v = 1, ..., K\}.$$

Step 3 :

When $|W(P^{(i+1)}, \boldsymbol{\gamma}^{i+1}) - W(P^{(i)}, \boldsymbol{\gamma}^{i})| < \epsilon$ for some predefined small value of ϵ, the process stops.

It can be shown that the algorithm converges when the criterion is bounded. Then, in the following section of Chapter 5, we will illustrate in details for different situations by applying different methods.

5.2.1 Partition Algorithm with Non-differentiable $C(.)$ in the Case of One Variable

When the copula function is non-differentiable, the joint density function of distributions will be approximated. The marginal distribution $G(.)$ can be estimated with an empirical distribution or a parametric density esimation. The detailed partition process is described as follows:

Step 1:

Initialize the parition as $P^0 = (P_1^0, ..., P_K^0)$ where the number of clusters K is pre-defined.

Step 2:

Define the partition after the i^{th} iteration as $P^i = (P_1^i, ..., P_K^i)$. Then, the allocation step consists of:

Step 2 (a):

Obtain the cumulative distribution function of the marginal distributions by an empirical distribution or a parametric density estimation. In the non-parametric case, this is estimated with

$$G^k_{T_j}(x) = P(\{F \in P_k \,|F(T_j) \leq x\}), \quad k = 1, ..., K, \quad j = 1, ..., n.$$

Step 2 (b):

Estimate the parameters $(\boldsymbol{\beta}_1, ...\boldsymbol{\beta}_K)$ of the copula functions $C_k(.; \boldsymbol{\beta}_k), k = 1, ..., K$, by maximizing

$$L(X, \boldsymbol{\gamma}) = \sum_{k=1}^{K} p_k \sum_{\omega \in P_k} h_k(X, \boldsymbol{\gamma}_k).$$

When $n = 2$, where $T = (T_1, T_2)$, the approximation of the density function is defined as

$$\begin{aligned} h_k(X, \boldsymbol{\gamma}) =& C_k((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)); \boldsymbol{\beta}_k) - C_k((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)); \boldsymbol{\beta}_k) \\ & - C_k((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)); \boldsymbol{\beta}_k) + C_k((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)); \boldsymbol{\beta}_k) \end{aligned}$$

where $\varepsilon_j = (Max\ G^k_{T_j}(F(T_j)) - Min\ G^k_{T_j}(F(T_j)))/N$. Therefore, in order to maximize $H(X, \boldsymbol{\gamma})$, it suffices to find $\boldsymbol{\beta}_k$ which maximizes

$$\begin{aligned} H(X, \boldsymbol{\gamma}) =& \sum_{k=1}^{K} p_k \sum_{\omega \in P_k} h_k(X, \boldsymbol{\gamma}_k) = \sum_{k=1}^{K} p_k \sum_{\omega \in P_k} C_k((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)); \boldsymbol{\beta}_k) \\ & - C_k((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)), \boldsymbol{\beta}_k) - C_k((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)); \boldsymbol{\beta}_k) \\ & + C_k((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)), \boldsymbol{\beta}_k). \end{aligned} \tag{5.2}$$

Step 2 (c):

Obtain the density function values $h^k_{\omega_i}(X, \boldsymbol{\gamma})$ for each unit $\omega_i, i = 1, ..., N$, under a certain cluster $k = 1, ..., K$.

Step 2 (d):

Allocate each unit to the cluster with the largest density value and obtain the new partition $\{P_k^{(i+1)}, k = 1, ..., K\}$.

Step 3 :

When $|W(P^{(i+1)}, \boldsymbol{\gamma}^{i+1}) - W(P^{(i)}, \boldsymbol{\gamma}^{i})| < \epsilon$ for some predefined small value of ϵ, the process stops.

5.2.2 Partition Algorithm with Differentiable H in the Case of One Variable

In the case of one variable, when both the marginal distributions $G_k(.)$ and the copula function $C_k(.)$ are differentiable for each cluster, the H_k is differentiable. Here, we introduce the detailed steps of the partition algorithm with a differentiable parametric copula function.

Step 1:

Initialize the parition as $P^0 = (P_1^0, ..., P_K^0)$ where the number of clusters K is pre-defined.

Step 2:

Define the partition after the i^{th} iteration as $P^i = (P_1^i, ..., P_K^i)$. Then, the allocation step consists of:

Step 2 (a):

Define T values and obtain $y_{ij} = F_i(T_j), i = 1, ..., m, j = 1, ..., n$, where each $F_i, i = 1, ..., m$, is a distributional-data unit represented by an empirical cumulative distribution. Figure 5.2.2 illustrates an example with 10 distributional-data units and 2-dimensional T values. Then, the probability values $y_{ij}, i = 1, ..., m, j = 1, ..., n$, form a $m \times n$ matrix, which will be used as the marginal distribution probability values for the next step.

Step 2 (b):

Select and estimate the parameters $\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_K$ by maximizing the log-likelihood function of

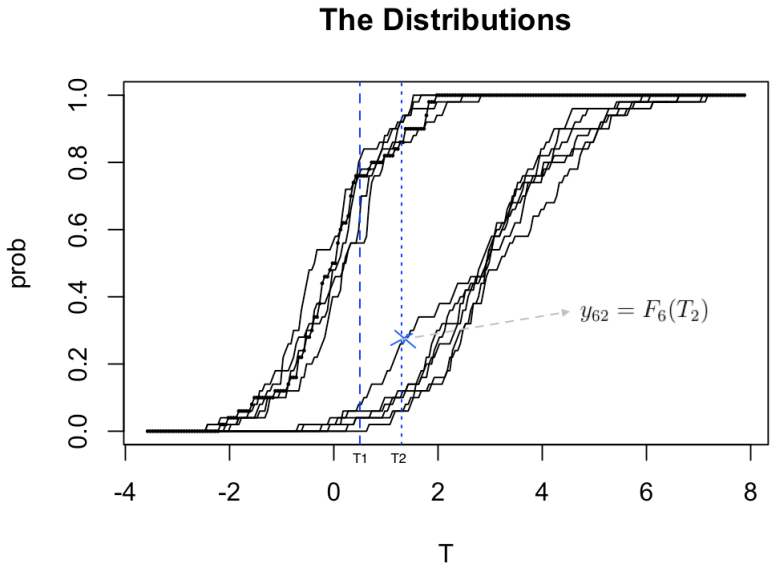

Figure 5.1: Example of distributional-data with 10 units and 2 dimensional T as $F_i(T_i), i = 1, ..., 10, j = 1, 2$.

the n dimensional candidate copula functions based on observations $y_{ij}, i = 1, ..., m, j = 1, ..., n$, for each cluster.

Step 2 (c):

Fit the selected copula model from Step 2(b) and obtain density values $h_{\omega_i^k}(X, \boldsymbol{\gamma}), i = 1, ..., N, k = 1, ..., K$, for each unit under each copula function of a certain cluster k. Allocate each unit to the cluster with the largest density value and obtain the $(i+1)^{th}$ partition result $\{P_k^{(i+1)}, k = 1, ..., K\}$.

Step 3 :

When $|W(P^{(i+1)}, \boldsymbol{\gamma}^{i+1}) - W(P^{(i)}, \boldsymbol{\gamma}^i)| < \epsilon$ for some predefined small value of ϵ, the process stops.

5.2.3 Partition Algorithm with Differentiable H in the Case of More Than One Variable

Since the joint distribution function H is differentiable, the estimation process of the parametric copula function is similar to the one variable case. However, in the case of more than one variable, the copula function should be applied twice to represent the joint distribution function of distributions. Suppose we have P-dimension variables $X^1, ..., X^P$. The detailed partition process is as follows:

Step 1:

Initialize the parition as $P^0 = (P_1^0, ..., P_K^0)$ where the number of clusters K is pre-defined.

Step 2:

Define the partition after the i^{th} iteration as $P^i = (P_1^i, ..., P_K^i)$. Then, the allocation step consists of:

Step 2 (a):

Define n-dimensional T values for each variable X^p and obtain $y_{ij}^p = F_i^p(T_j), i = 1, ..., m, j = 1, ..., n, p = 1, ..., P$, where each $F_i^p, i = 1, ..., m$, is a distributional-data unit represented by an empirical cumulative distribution. Then, we have in total P $m \times n$ matrices of the probability values $y_{ij}^p, i = 1, ..., m, j = 1, ..., n$, which will be used as the marginal distribution probability values for the next step.

Step 2 (b):

Select and estimate the parameters $\{\boldsymbol{\gamma}_1^p, ..., \boldsymbol{\gamma}_K^p\}$ by maximizing the log-likelihood function of n dimensional candidate copula functions based on $y_{ij}^p, i = 1, ..., m, j = 1, ..., n$, of each cluster for each variable.

Step 2 (c):

Fit the selected copula models for each variable from Step 2(b) and obtain the P-dimensional

probability values as:

$$(G^1(y^1_{i,T^1_1}, ..., y^1_{i,T^1_n}), ..., G^P(y^P_{i,T^P_1}, ..., y^P_{i,T^P_n}),\ i = 1, ..., N) \tag{5.3}$$

for each unit under the copula function of a certain cluster.

Step 2 (d):

Select and estimate $\{\boldsymbol{\gamma}_1, ..., \boldsymbol{\gamma}_K\}$ by maximizing the log-likelihood function of the P-dimensional candidate copula functions based on probability values obtained from Step 2(c) (equation 5.3) for each cluster. Fit the selected copula models for each cluster and obtain density values $h_{\omega^k_i}(X, \boldsymbol{\gamma}), i = 1, ..., N, k = 1, ..., K$, for each unit under the copula function of a certain cluster.

Step 2 (e):

Allocate each unit to the cluster with the largest density value and obtain the $(i+1)^{th}$ partition result $\{P^{(i+1)}_k, k = 1, ..., K\}$.

Step 3 :

When $|W(P^{(i+1)}, \boldsymbol{\gamma}^{i+1}) - W(P^{(i)}, \boldsymbol{\gamma}^i)| < \epsilon$ for some predefined small value of ϵ, the process stops.

The R code is available for each algorithm in the appendix. In the next Chapter 6, we will apply these algorithms for a variety of data sets to further illustrate the details.

Chapter 6

Simulation Study and Applications

In this chapter, we first present some simulation results for one-dimensional data X_1 with mixed clusters in section 6.1 and two-dimensional data X_1, X_2 in section 6.2. Then, in section 6.3, we consider the general multi-dimensional case $X_1, ..., X_P$. In each case, we will apply the respective algorithms described in Chapter 5.

6.1 Simulation Example: Mixture Decomposition for One Dimensional Data X_1

For one-dimensional data, we first introduce a simulation example when the copula function $C(.)$ is non-differentiable and the marginal distribution $G(.)$ is estimated with an empirical distribution (with mixed clusters in section 6.1.1 and with one outlier cluster in section 6.1.2) and by the parametric method (in section 6.1.3). Then, we consider the parametric copula estimation algorithm for one-dimensional data in section 6.1.4.

6.1.1 Example 1: $G(.)$ and $C(.)$ are both non-differentiable for mixed clusters

In this example, we will show how to do the mixture decomposition for one-dimensional data with mixed clusters when the marginal distribution $G(.)$ and the copula function $C(.)$ are both non-differentiable.

Data Simulation

We first simulate the random sample from two normal distributions as follows:

a) Generate 300 data points randomly from a normal distribution with mean = 2, and variance = 16.

b) Generate 200 data points randomly from a normal distribution with mean = 10 and variance = 9.

c) Combine the two data sets and obtain the final data set.

Figure 6.1 shows how the data look like; the x-axis represents the observation number and the y-axis gives the observation value. The overlap in cluster values is evident. Our goal is to partition this simulated data set into two clusters with the symbolic mixture decomposition with the algorithm described in section 5.2.1.

The Symbolic Mixture Decomposition

The set of 500 classical observations was subdivided into five sets of 100 observations each, taking observations 1,...,100 for set 1 (ω_1),..., and observations 401,...,500 for set 5 (ω_5). Suppose the symbolic data set is defined by five units w_i, each one described by a distribution F_i, which is the cumulative distribution of every 100 single value of points. The five distributions are displayed in Figure 6.2. From equation 2.7, we have the final $K = 2$ clusters mixture models for distributions. We want to estimate the mixture models and seek the best

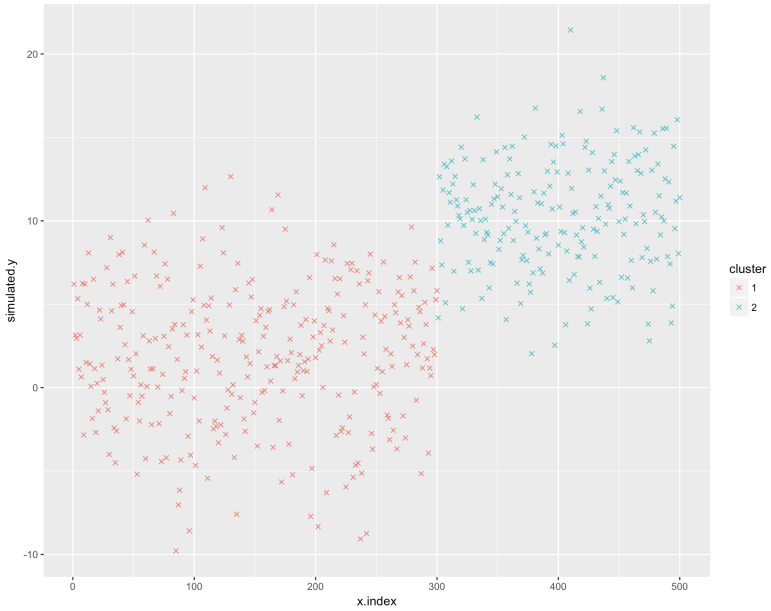

Figure 6.1: Scatter Plot of Simulated Data.

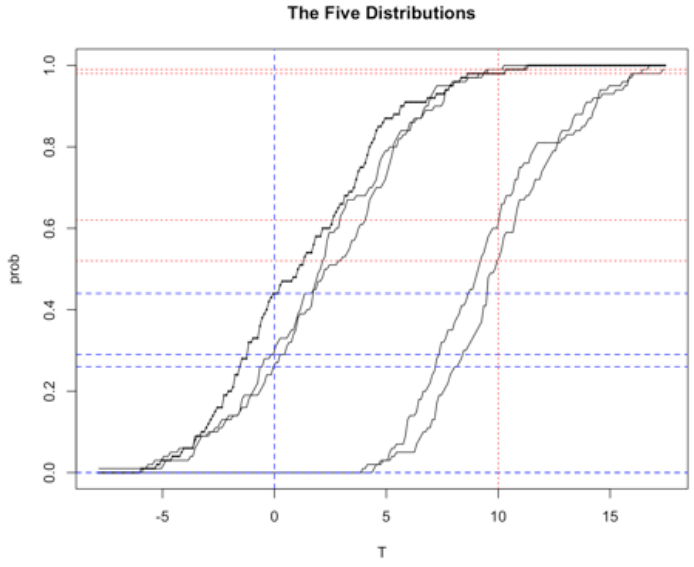

Figure 6.2: The Five Distributions.

grouping of the five units into $K = 2$ classes.

We use the family of the simplest copulas' parametric model (given in equation 3.4) with one parameter β as (more details are given by Nelsen, 2006)

$$C_k(\mu, \nu) = \begin{cases} \min(\mu, \nu), \text{ if } \beta = 1, \\ \max(\mu + \nu + 1, 0), \text{ if } \beta = 0. \end{cases}$$

From Figure 6.2, we can see the cumulative distribution curves of the five distributional data. In order to partition these five units into two clusters, we need an extra axis T to describe the behavior of each unit and also their relationship to find the best grouping of them. After defining the dimension n of T to be $n = 2$, in this example as $T = (T_1, T_2)$, we need to choose the values of each T. Intuitively, a good choice of T is to represent distinct cumulative values of $F_i(T), i = 1, ..., 5$, in order to have a more clear separation based on these values. What is more, it is better to let different T's describe different areas of the units to cover more properties of each unit. For example, the two vertical dotted lines in Figure 6.2 show the locations of selected T values for this example, where $T_1 = 0$ and $T_2 = 10$. Both of them show clear distinct classes of representing the cumulative values of the five units and they are separate from each other with some distance to describe the lower value and also higher values, respectively.

The marginal distributions of $F_i(T_j), i = 1, ..., 5, j = 1, 2$, are derived by their empirical distribution using equation 3.3. Here, we assume there will be two clusters for the simulated data with $P = (P_1, P_2)$, P_k is the k^{th} cluster in the partition and p_k is the associated probability and $\boldsymbol{\gamma}_k$ is the parameter of the density $h_k(.)$ with $k = 1, 2$ (as shown in equation 6.1),

$$h(x_1, x_2; \boldsymbol{\gamma}) = \sum_{k=1}^{2} h_k(x_1, x_2; \boldsymbol{\gamma_k}) = \sum_{k=1}^{2} p_k c_k(G^k_{z_1}(x_1; \boldsymbol{b}^k_1), G^k_{z_2}(x_2; \boldsymbol{b}^k_2); \boldsymbol{\beta}_k) \tag{6.1}$$

where $\boldsymbol{\gamma}_k = (\boldsymbol{b}^k, \boldsymbol{\beta}_k)$. In this case, H is not differentiable and so we estimate the copula parameters by maximizing equation 5.2, where $\varepsilon_1 = (1 - 0.5)/5 = 0.1$ and $\varepsilon_2 = (1 - 1/3)/5 \approx 0.1$.

Suppose the initial partition is given as $P_1^0 = (\omega_1, \omega_2)$ and $P_2^0 = (\omega_3, \omega_4, \omega_5)$.

Step 1: By maximizing the criterion of equation 5.2, we first need to calculate the values of $h_k(X, \boldsymbol{\gamma}),\ k = 1, 2$, for each cluster under different values of $\beta = 0$ and $\beta = 1$ and then select the parameter for each cluster with the highest value of $h_k(X, \boldsymbol{\gamma})$. Since we only have the parameter for the copula function and there is no parameter for the marginal distributions by using empirical distritbuions, $h_k(X, \boldsymbol{\gamma})$ equals $h_k(X, \beta)$. This means β is the only parameter that we need to estimate in this case. For example, for the first cluster $P_1^0 = (\omega_1, \omega_2)$, when $\beta = 0$, we have:

$$
\begin{aligned}
h_1(X, \gamma) &= \sum_{\omega_i \in P_1^0} h_i(X, \beta = 0) \\
&= \sum_{\omega_i, i=1,2} \max(G^1_{T_1}(x_{i1} + \varepsilon_1) + G^1_{T_2}(x_{i2} + \varepsilon_2) + 1, 0) \\
&\quad - \max(G^1_{T_1}(x_{i1} + \varepsilon_1) + G^1_{T_2}(x_{i2} - \varepsilon_2) + 1, 0) \\
&\quad - \max(G^1_{T_1}(x_{i1} - \varepsilon_1) + G^1_{T_2}(x_{i2} + \varepsilon_2) + 1, 0) \\
&\quad + \max(G^1_{T_1}(x_{i1} - \varepsilon_1) + G^1_{T_2}(x_{i2} - \varepsilon_2) + 1, 0) = 1
\end{aligned}
$$

where $X_i = (x_{i1}, x_{i2}), x_{ij} = F_i(T_j), i = 1, 2$, for the first cluster and $G^k_{T_j}(x) = Pr(\frac{\omega \in P_k}{F(T_j) \leq x}),\ k =$

1, for the first cluster. Similarly, when $\beta = 1$, for the first cluster $P_1^0 = (\omega_1, \omega_2)$, we have:

$$\begin{aligned} h_1(X, \gamma) &= \sum_{\omega_i \in P_1^0} h_i(X, \beta = 1) \\ &= \sum_{\omega_i, i=1,2} \min(G^1_{T_1}(x_{i1} + \varepsilon_1), G^1_{T_2}(x_{i2} + \varepsilon_2)) - \min(G^1_{T_1}(x_{i1} + \varepsilon_1), G^1_{T_2}(x_{i2} - \varepsilon_2)) \\ &\quad - \min(G^1_{T_1}(x_{i1} - \varepsilon_1), G^1_{T_2}(x_{i2} + \varepsilon_2)) + \min(G^1_{T_1}(x_{i1} - \varepsilon_1), G^1_{T_2}(x_{i2} - \varepsilon_2)) = 1. \end{aligned}$$

For the second cluster $P_2^0 = (\omega_3, \omega_4, \omega_5)$ and $\beta = 0$, we have:

$$\begin{aligned} h_2(X, \gamma) &= \sum_{\omega_i \in P_2^0} h_i(X, \beta = 0) \\ &= \sum_{\omega_i, i=3,4,5} \max(G^2_{T_2}(x_{i1} + \varepsilon_1) + G^2_{T_2}(x_{i2} + \varepsilon_2) + 1, 0) \\ &\quad - \max(G^2_{T_1}(x_{i1} + \varepsilon_1) + G^2_{T_2}(x_{i2} - \varepsilon_2) + 1, 0) \\ &\quad - \max(G^2_{T_1}(x_{i1} - \varepsilon_1) + G^2_{T_2}(x_{i2} + \varepsilon_2) + 1, 0) \\ &\quad + \max(G^2_{T_1}(x_{i1} - \varepsilon_1) + G^2_{T_2}(x_{i2} - \varepsilon_2) + 1, 0) = 2/3. \end{aligned}$$

Finally, when $\beta = 1$, we have:

$$\begin{aligned} h_2(X, \gamma) &= \sum_{\omega_i \in P_2^0} h_i(X, \beta = 1) \\ &= \sum_{\omega_i, i=3,4,5} \min(G^2_{T_1}(x_{i1} + \varepsilon_1), G^2_{T_2}(x_{i2} + \varepsilon_2)) - \min(G^2_{T_1}(x_{i1} + \varepsilon_1), G^2_{T_2}(x_{i2} - \varepsilon_2)) \\ &\quad - \min(G^2_{T_1}(x_{i1} - \varepsilon_1), G^2_{T_2}(x_{i2} + \varepsilon_2)) + \min(G^2_{T_1}(x_{i1} - \varepsilon_1), G^2_{T_2}(x_{i2} - \varepsilon_2)) = 1. \end{aligned}$$

Table 6.1 summarizes these results.

In order to maximize the likelihood of the mixture model with the associated probability

Table 6.1: Induction of the copulas from the initial partition.

Class	Copula	$h_k(X, \gamma)$
$P_1^0 = (\omega_1, \omega_2)$	$\beta = 0$	1
	$\beta = 1$	1
$P_2^0 = (\omega_3, \omega_4, \omega_5)$	$\beta = 0$	2/3
	$\beta = 1$	1

for each cluster, we need to further calculate the likelihood

$$L(X, \boldsymbol{\gamma}) = \sum_{k=1,2} p_k h_k(X, \boldsymbol{\gamma}_k) = \sum_{k=1,2} p_k h_k(X, \beta)$$

for different combinations of β. For example, when $\beta = 0$ for the first cluster and $\beta = 0$ for the second cluster, we have:

$$L(X, \boldsymbol{\gamma}) = p_1 \times h_1(.; \beta = 0) + p_2 \times h_2(.; \beta = 0) = (2/5) \times 1 + (3/5) \times (2/3) = 4/5$$

where $p_1 = 2/5$ and $p_2 = 3/5$. These values for p_1 and p_2 are because we assume the initial partition is $P_1^0 = (\omega_1, \omega_2)$ and $P_2^0 = (\omega_3, \omega_4, \omega_5)$. Likewise, we can calculate this likelihood $L(X, \boldsymbol{\gamma})$ for other combination of β values. The results are shown in Table 6.2. From these results, we see that $h_1(X, \beta = 0)$ and $h_2(X, \beta = 1)$ or $h_1(X, \beta = 1)$ and $h_2(X, \beta = 1)$ achieve the best performance with the largest likelihood $L(X, \boldsymbol{\gamma})$, which means for the initial partition, they are the best parameters.

Step 2: After estimating the parameters, each unit $\omega_i, i = 1, ..., 5$, will be evaluated under each estimated cluster model. Then, a unit will be allocated into the cluster for which the unit achieves the best performance (largest value of the density function).This

Table 6.2: Likelihood $L(X, \boldsymbol{\gamma})$ for different combinations of β_k.

	First Cluster : $\beta = 0$	First Cluster : $\beta = 1$
Second Cluster : $\beta = 0$	1*(2/5)+(2/3)*(3/5)=4/5	1*(2/5)+(2/3)*(3/5)=4/5
Second Cluster : $\beta = 1$	1*(2/5)+1*(3/5)=1	1*(2/5)+1*(3/5)=1

involves looking at combinations of values for β in each cluster. For this step, we only illustrate the calculation for the case that $h_1(X, \beta = 1)$ and $h_2(X, \beta = 1)$; the cases that $h_1(X, \beta = 0)$ and $h_2(X, \beta = 1)$ are done similarly.

When $h_1(X, \beta = 1)$ and $h_2(X, \beta = 1)$, $C_1(.) = C_2(.) = min(\mu, \nu)$, $h_{\omega_i}(X, \boldsymbol{\gamma})$ is evaluated under a certain cluster $k, k = 1, 2$, for each $w_i, i = 1, ..., 5$, as

$$\begin{aligned} h^k_{\omega_i}(X, \boldsymbol{\gamma}) =& C((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2))) - C((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2))) \\ & - C((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2))) + C((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2))) \\ & = \min(G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)) - \min(G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)) \\ & - \min(G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)) + \min(G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)). \end{aligned} \tag{6.2}$$

Table 6.3 shows all these results of $h^k_{\omega_i}(X, \boldsymbol{\gamma})$ for the five units $\omega_i, i = 1, ..., 5$, under the two clusters, $k = 1, 2$. The last column of the table shows the membership of class with the highest $h^k_{\omega_i}(X, \boldsymbol{\gamma})$. For example, the third unit ω_3 has a larger density value under the first cluster (with $h_1(X, \boldsymbol{\gamma}) = 1/2$) than under the second cluster (where $h_1(X, \boldsymbol{\gamma}) = 1/3$). Therefore, the unit ω_3 is allocated to the first cluster. Remember the unit ω_3 is assumed to be in the second cluster with a wrong label in our initial partition and it is now placed into its true cluster by our algorithm.

Then, the new partition is updated to $P^1_1 = (F_1, F_2, F_3)$ and $P^1_2 = (F_4, F_5)$.

Table 6.3: Allocation of each unit to the best fit class.

Unit	$h^1_{\omega_i}(X, \boldsymbol{\gamma})$	$h^2_{\omega_i}(X, \boldsymbol{\gamma})$	Class membership
ω_1	$1/2$	0	P_1
ω_2	$1/2$	$1/3$	P_1
ω_3	$1/2$	$1/3$	P_1
ω_4	0	$1/3$	P_2
ω_5	0	$2/3$	P_2

Step 3: Repeat Step 1 and Step 2 under the updated partition and calculate the $h_{\omega_i}(X, \boldsymbol{\gamma})$ for each unit $i = 1, ..., 5$. The results show that there are no more reallocations and therefore the final partition should be $P_1^1 = (\omega_1, \omega_2, \omega_3)$ and $P_2^1 = (\omega_4, \omega_5)$. This is exactly the true simulated partition, which demonstrates the success of our algorithm introduced in section 5.2.1.

6.1.2 Example 2: $G(.)$ and $C(.)$ are both non-differentiable for one outlier cluster

In this example, we will show how to do the mixture decomposition for one-dimensional data when one cluster is an outlier with non-differentiable marginal distribution $G(.)$ and copula function $C(.)$.

Data Simulation

We first simulate the random sample from two normal distributions as follows:

a) Generate 700 data points randomly from a normal distribution with mean = 2, and variance = 16.

b) Generate 300 data points randomly from a normal distribution with mean = 30 and

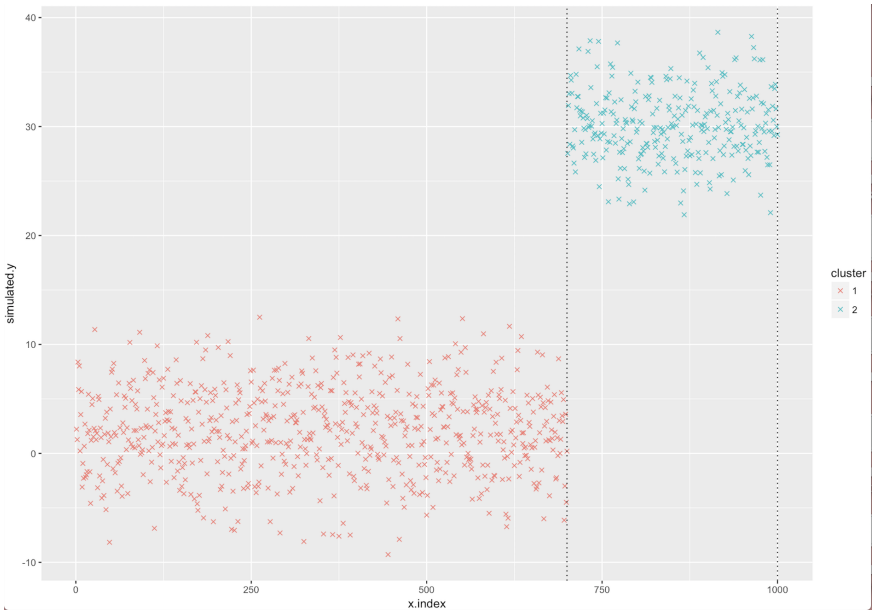

Figure 6.3: Scatter Plot of Simulated Data.

variance = 9.

c) Combine the two data sets and obtain the final data set.

Figure 6.3 shows how the data look like; the x-axis represents the observation number and the y-axis gives the observation value. It is obvious that the two clusters are separate from each other with some distance. Our goal is to see if we first assign some of the observations from the first cluster (the blue one in Figure 6.3) to the second cluster (the pink one in Figure 6.3), will the algorithm described in section 5.2.1 can re-allocate them back to the right cluster.

The Symbolic Mixture Decomposition

The set of 1000 classical observations was subdivided into twenty sets of 50 observations each, taking observations 1,...,50 for set 1 (ω_1),..., and observations 951,...,1000 for set 20

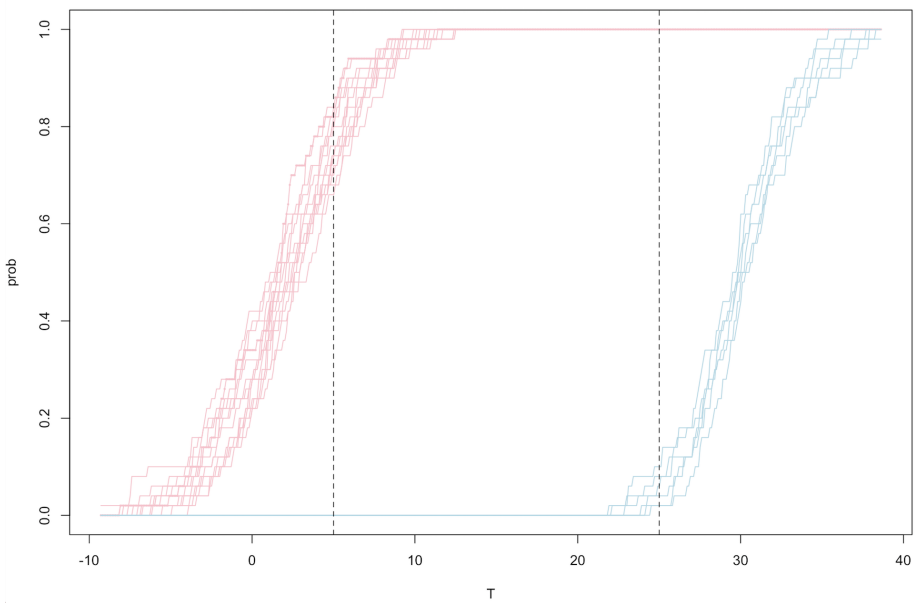

Figure 6.4: The Cumulative Distributions.

(ω_{20}) as summarized in Table 6.4. Suppose the symbolic data set is defined by twenty units w_i, each one described by a distribution F_i, which is the cumulative distribution of every 50 single value of points. The twenty distributions are displayed in Figure 6.4. From equation 2.7, we have the final $K = 2$ clusters mixture models for distributional-data. We want to estimate the mixture models and seek the best grouping of the twenty units into $K = 2$ classes.

We also use the family of the simplest copulas' parametric model (given in equation 3.4) with one parameter β as (more details are given by Nelsen, 2006)

$$C_k(\mu, \nu) = \begin{cases} \min(\mu, \nu), \text{ if } \beta = 1, \\ \max(\mu + \nu + 1, 0), \text{ if } \beta = 0. \end{cases}$$

From Figure 6.4, we can see the cumulative distribution curves of the twenty units (dis-

Table 6.4: Simulated Data for Non-parametric Estimation of One Dimension Case X_1.

Class	# sample (single value of points)	Distribution	#original observations	# units
1	700	$Normal(2, 4)$	50	14
2	300	$Normal(30, 3)$	50	6

tributional data). In order to represent the properties of the units, we need an extra axis T to describe the behavior of each unit and also their relationship to find the best grouping of them. After defining the dimension n of T to be $n = 2$, in this example as $T = (T_1, T_2)$, we need to choose the values of each T. Intuitively, a good choice of T is to represent distinct cumulative values of $F_i(T), i = 1, ..., 20$, in order to have a more clear separation based on these values. What is more, it is better to let different T's describe different areas of the units to cover more properties of each unit. For example, the two vertical dotted lines in Figure 6.4 shows the locations of selected T values for this example, where $T_1 = 5$ and $T_2 = 25$. Both of them show clear distinct classes of representing the cumulative values of the twenty units and they are separate from each other with some distance to describe the lower value and also higher values, respectively.

The marginal distributions of $F_i(T_j), i = 1, ..., 20, j = 1, 2$, are derived by their empirical distribution using equation 3.3. Here, we assume there will be two clusters for the simulated data with $P = (P_1, P_2)$, P_k is the k^{th} cluster in the partition and p_k is the associated probability and $\boldsymbol{\gamma}_k$ is the parameter of the density $h_k(.)$ with $k = 1, 2$ (as shown in equation 6.3),

$$h(x_1, x_2; \boldsymbol{\gamma}) = \sum_{k=1}^{2} h_k(x_1, x_2; \boldsymbol{\gamma_k}) = \sum_{k=1}^{2} p_k c_k(G^k_{z_1}(x_1; \boldsymbol{b}^k_1), G^k_{z_2}(x_2; \boldsymbol{b}^k_2); \boldsymbol{\beta}_k) \tag{6.3}$$

where $\boldsymbol{\gamma}_k = (\boldsymbol{b}^k, \boldsymbol{\beta}_k)$. In this case, H is not differentiable and so we estimate the copula pa-

Table 6.5: Assignment of Starting Partition.

Class	True Partition	Starting Partition
1	$P_1 = (\omega_1, ..., \omega_{14})$	$P_1^0 = (\omega_1, ..., \omega_{10})$
2	$P_2 = (\omega_{15}, ..., \omega_{20})$	$P_2^0 = (\omega_{11}, ..., \omega_{20})$

rameters by maximizing equation 5.2, where $\varepsilon_1 = (1-0.1)/10 = 0.09$ and $\varepsilon_2 = (1-0.2)/10 = 0.08$.

Suppose the initial partition is given as $P_1^0 = (\omega_1, ..., \omega_{10})$ and $P_2^0 = (\omega_{11}, ..., \omega_{20})$. We assign $(\omega_{11}, ..., \omega_{14})$ to the wrong cluster and hope to see they will be re-allocated to the right cluster as summarized in Table 6.5.

Step 1: By maximizing the criterion of equation 5.2, we first need to calculate the values of $h_k(X, \boldsymbol{\gamma})$, $k = 1, 2$, for each cluster under different values of $\beta = 0$ and $\beta = 1$ and then select the parameter for each cluster with the highest value of $h_k(X, \boldsymbol{\gamma})$. Since we only have the parameter for the copula function and there is no parameter for the marginal distributions by using empirical distritbuions, $h_k(X, \boldsymbol{\gamma})$ equals $h_k(X, \beta)$. This means β is the only parameter that we need to estimate in this case. For example, for the first cluster $P_1^0 = (\omega_1, ..., \omega_{10})$, when $\beta = 0$, we have:

$$
\begin{aligned}
h_1(X,\gamma) &= \sum_{\omega_i \in P_1^0} h_i(X, \beta = 0) \\
&= \sum_{\omega_i, i=1,...10} \max(G^1_{T_1}(x_{i1}+\varepsilon_1) + G^1_{T_2}(x_{i2}+\varepsilon_2) + 1, 0) \\
&\quad - \max(G^1_{T_1}(x_{i1}+\varepsilon_1) + G^1_{T_2}(x_{i2}-\varepsilon_2) + 1, 0) \\
&\quad - \max(G^1_{T_1}(x_{i1}-\varepsilon_1) + G^1_{T_2}(x_{i2}+\varepsilon_2) + 1, 0) \\
&\quad + \max(G^1_{T_1}(x_{i1}-\varepsilon_1) + G^1_{T_2}(x_{i2}-\varepsilon_2) + 1, 0) = 6.8
\end{aligned}
$$

where $X_i = (x_{i1}, x_{i2}), x_{ij} = F_i(T_j), i = 1, ..., 10$, for the first cluster and $G^k_{T_j}(x) = Pr(\frac{\omega \in P_k}{F(T_j) \leq x}),\ k = 1$, for the first cluster. Similarly, when $\beta = 1$, for the first cluster $P_1^0 = (\omega_1, ..., \omega_{10})$, we have:

$$
\begin{aligned}
h_1(X,\gamma) &= \sum_{\omega_i \in P_1^0} h_i(X, \beta = 1) \\
&= \sum_{\omega_i, i=1,...10} \min(G^1_{T_1}(x_{i1}+\varepsilon_1), G^1_{T_2}(x_{i2}+\varepsilon_2)) - \min(G^1_{T_1}(x_{i1}+\varepsilon_1), G^1_{T_2}(x_{i2}-\varepsilon_2)) \\
&\quad - \min(G^1_{T_1}(x_{i1}-\varepsilon_1), G^1_{T_2}(x_{i2}+\varepsilon_2)) + \min(G^1_{T_1}(x_{i1}-\varepsilon_1), G^1_{T_2}(x_{i2}-\varepsilon_2)) = 6.8.
\end{aligned}
$$

For the second cluster $P_2^0 = (\omega_{11}, ..., \omega_{20})$ and $\beta = 0$, we have:

$$
\begin{aligned}
h_2(X,\gamma) &= \sum_{\omega_i \in P_2^0} h_i(X, \beta = 0) \\
&= \sum_{\omega_i, i=11,...20} \max(G^2_{T_2}(x_{i1}+\varepsilon_1) + G^2_{T_2}(x_{i2}+\varepsilon_2) + 1, 0) \\
&\quad - \max(G^2_{T_1}(x_{i1}+\varepsilon_1) + G^2_{T_2}(x_{i2}-\varepsilon_2) + 1, 0) \\
&\quad - \max(G^2_{T_1}(x_{i1}-\varepsilon_1) + G^2_{T_2}(x_{i2}+\varepsilon_2) + 1, 0) \\
&\quad + \max(G^2_{T_1}(x_{i1}-\varepsilon_1) + G^2_{T_2}(x_{i2}-\varepsilon_2) + 1, 0) = 1.2.
\end{aligned}
$$

Table 6.6: Induction of the copulas from the initial partition.

Class	Copula	$h_k(X, \gamma)$
$P_1^0 = (\omega_1, ..., \omega_{10})$	$\beta = 0$	6.8
	$\beta = 1$	6.8
$P_2^0 = (\omega_{11}, ..., \omega_{20})$	$\beta = 0$	1.2
	$\beta = 1$	4.4

Finally, when $\beta = 1$, we have:

$$\begin{aligned} h_2(X, \gamma) &= \sum_{\omega_i \in P_2^0} h_i(X, \beta = 1) \\ &= \sum_{\omega_i, i=11,...,20} \min(G^2_{T_1}(x_{i1} + \varepsilon_1), G^2_{T_2}(x_{i2} + \varepsilon_2)) - \min(G^2_{T_1}(x_{i1} + \varepsilon_1), G^2_{T_2}(x_{i2} - \varepsilon_2)) \\ &\quad - \min(G^2_{T_1}(x_{i1} - \varepsilon_1), G^2_{T_2}(x_{i2} + \varepsilon_2)) + \min(G^2_{T_1}(x_{i1} - \varepsilon_1), G^2_{T_2}(x_{i2} - \varepsilon_2)) = 4.4. \end{aligned}$$

Table 6.6 summarizes these results.

In order to maximize the likelihood of the mixture model with the associated probability for each cluster, we need to further calculate the likelihood

$$L(X, \boldsymbol{\gamma}) = \sum_{k=1,2} p_k h_k(X, \boldsymbol{\gamma}_k) = \sum_{k=1,2} p_k h_k(X, \beta)$$

for different combinations of β. For example, when $\beta = 0$ for the first cluster and $\beta = 0$ for the second cluster, we have:

$$L(X, \boldsymbol{\gamma}) = p_1 \times h_1(.; \beta = 0) + p_2 \times h_2(.; \beta = 0) = (1/2) \times 6.8 + (1/2) \times 1.2 = 4$$

where $p_1 = p_2 = 1/2$. These values for p_1 and p_2 are because we assume the initial partition is $P_1^0 = (\omega_1, ..., \omega_{10})$ and $P_2^0 = (\omega_1 1, ..., \omega_{20})$. Likewise, we can calculate this likelihood

Table 6.7: Likelihood $L(X, \boldsymbol{\gamma})$ for different combinations of β_k.

	First Cluster : $\beta = 0$	First Cluster : $\beta = 1$
Second Cluster : $\beta = 0$	6.8*(1/2)+1.2*(1/2)=4	6.8*(1/2)+1.2*(1/2)=4
Second Cluster : $\beta = 1$	6.8*(1/2)+4.4*(1/2)=5.6	6.8*(1/2)+4.4*(1/2)=5.6

$L(X, \boldsymbol{\gamma})$ for other combination of β values. The results are shown in Table 6.7. From these results, we see that $h_1(X, \beta = 0)$ and $h_2(X, \beta = 1)$ or $h_1(X, \beta = 1)$ and $h_2(X, \beta = 1)$ achieve the best performance with the largest likelihood $L(X, \boldsymbol{\gamma})$, which means for the initial partition, they are the best parameters.

Step 2: After estimating the parameters, each unit $\omega_i, i = 1, ..., 20$, will be evaluated under each estimated cluster model. Then, a unit will be allocated into the cluster for which the unit achieves the best performance (largest value of the density function).This involves looking at combinations of values for β in each cluster. For this step, we only illustrate the calculation for the case that $h_1(X, \beta = 1)$ and $h_2(X, \beta = 1)$; the cases that $h_1(X, \beta = 0)$ and $h_2(X, \beta = 1)$ are done similarly.

When $h_1(X, \beta = 1)$ and $h_2(X, \beta = 1)$, $C_1(.) = C_2(.) = min(\mu, \nu)$, $h_{\omega_i}(X, \boldsymbol{\gamma})$ is evaluated under a certain cluster $k, k = 1, 2$, for each $w_i, i = 1, ..., 20$, as

$$\begin{aligned} h^k_{\omega_i}(X, \boldsymbol{\gamma}) =& C((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2))) - C((G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2))) \\ & - C((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2))) + C((G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2))) \\ & = \min(G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)) - \min(G^k_{T_1}(x_1 + \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)) \\ & - \min(G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 + \varepsilon_2)) + \min(G^k_{T_1}(x_1 - \varepsilon_1), G^k_{T_2}(x_2 - \varepsilon_2)). \end{aligned} \tag{6.4}$$

Table 6.8: Allocation of each unit to the best fit class.

Unit	$h^1_{\omega_i}(X, \boldsymbol{\gamma})$	$h^2_{\omega_i}(X, \boldsymbol{\gamma})$	Class membership
ω_{11}	0.6	0.1	P_1
ω_{12}	0.4	0.3	P_1
ω_{13}	0.7	0.3	P_1
ω_{14}	0.4	0.3	P_1

Table 6.8 shows the results of $h^k_{\omega_i}(X, \boldsymbol{\gamma})$ for the four units $\omega_i, i = 11, ..., 14$, that are incorrectly assigned under the two clusters, $k = 1, 2$. The last column of the table shows the membership of class with the highest $h^k_{\omega_i}(X, \boldsymbol{\gamma})$. We can see that all the four units have a larger density value under the first cluster than under the second cluster. Therefore, the unit $\omega_1 1, ...\omega_1 4$ are allocated to the first cluster. Remember these four units are assumed to be in the second cluster with a wrong label in our initial partition and it is now placed into its true cluster by our algorithm.

Then, the new partition is updated to $P^1_1 = (\omega_1, ..., \omega_{14})$ and $P^1_2 = (\omega_{15}, ..., \omega_{20})$.

Step 3: Repeat Step 1 and Step 2 under the updated partition and calculate the $h_{\omega_i}(X, \boldsymbol{\gamma})$ for each unit $i = 1, ..., 20$. The results show that there are no more reallocations and therefore the final partition should be $P^1_1 = (\omega_1, ..., \omega_{14})$ and $P^1_2 = (\omega_1 5, ..., \omega_{20})$. This is exactly the true simulated partition, which demonstrates the success of our algorithm introduced in section 5.2.1.

6.1.3 Example 3: $G(.)$ is differentiable and $C(.)$ is not differentiable

When the marginal distribution $G(.)$ is differentiable and $C(.)$ is not differentiable, the joint distribution function of distributions H is also non-differentiable. We will still use the approximation of the joint density function to model $h(X, \boldsymbol{\gamma})$ as in equation 5.1. The marginal distribution $G(.)$ will be estimated with a parametric approach. The algorithm that we use in this example is also the one described in section 5.2.1 (same as in example 1). The main difference is the parametric estimation of the marginal distribution, while we use the empirical distribution as in example 1.

Data Simulation

In this example, we simulate the random sample from three normal distributions as below:

a) Generate 700 data points randomly from a normal distribution with mean = 2, and variance = 16.

b) Generate 300 data points randomly from a normal distribution with mean = 10 and variance = 9.

c) Generate 500 data points randomly from a normal distribution with mean = 15 and variance = 25.

d) Combine the two data sets and obtain the final data set.

Figure 6.5 shows how the data look. The x-axis is the index from 1 to 1500 and the y-axis represents the values of the simulated data. It is obvious to see that group 2 and group 3 have a large area of overlapped data points in the direction of the y-axis. Our goal is to partition this simulated data set into three clusters with the symbolic mixture decomposition algortihm.

Figure 6.5: Scatter Plot of Simulated Data.

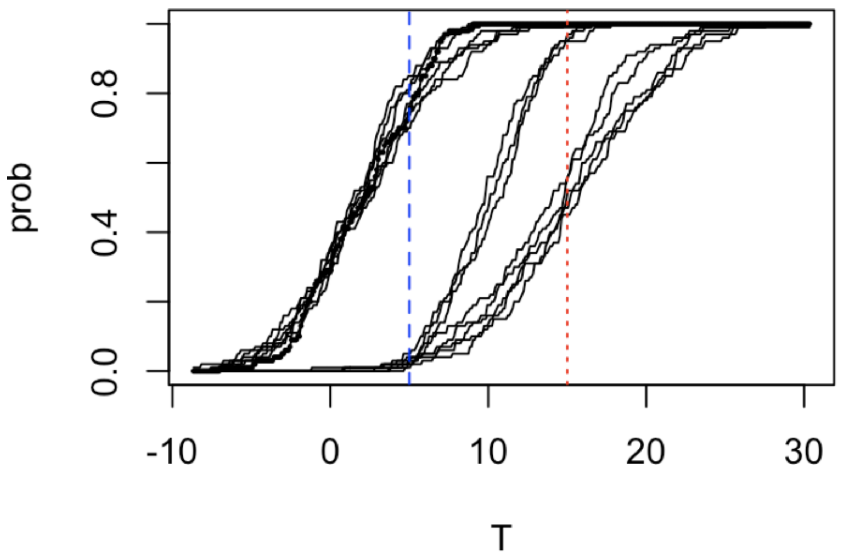

Figure 6.6: The Distributions of the Data of Figure 6.5.

The Symbolic Mixture Decomposition

The set of 1500 classical observations were subdivided into fifteen sets of 100 observations each. Suppose the symbolic data set is defined by 15 units $w_i, i = 1, ..., 15$, each one described by a distribution $F_i, i = 1, ..., 15$, which is the cumulative distribution of every succesive 100 single values points. The distributions are given in Figure 6.5. In order to partition these fifteen units into three clusters, we also need an extra axis T as for example 1. After defining the dimension n of T, where n = 2 in this example as $T = (T_1, T_2)$, we need to choose the values of each T. Here, we define $T_1 = 5$ and $T_2 = 15$ to represent distinct cumulative values of $F_i(T), i = 1, ..., 15$, in order to have a more clear separation. The two vertical dotted lines in Figure 6.6 show the locations of selected T values for this example, specifically, $T_1 = 5$ and $T_2 = 15$. Both of them show clear distinct classes of representing the cumulative values of the fifteen units and they are separate from each other by some distance to describe the lower value and also the higher values, respectively.

We first assume that the marginal distribution ${G^k}_T(., \boldsymbol{b}_k)$ follows a normal distrbution with mean μ_k and variance σ_k^2, with $\boldsymbol{b}_k = (\mu_k, \sigma_k^2)$ for each cluster P_k of the partition at a given T. The paramters are estimated by maximum likelihood estimation based on the value of the points obtained by $F_i(T_j),\ i = 1, ..., 15,\ j = 1, 2$, for each given T_j.

We also use the family of copula parametric models with one parameter β (from equation 3.4) as:

$$C(\mu, \nu) = \begin{cases} \min(\mu, \nu), \text{ if } \beta = 1, \\ \max(\mu + \nu + 1, 0), \text{ if } \beta = 0. \end{cases}$$

Since the copula function is non-differentiable, we still use the approximation of the density function by using equation 5.1. We start the partition as $P_1^0 = \{\omega_1, ..., \omega_6\}, P_2^0 = \{\omega_7, ..., \omega_{11}\}$ and $P_3^0 = \{\omega_{12}, ..., \omega_{15}\}$. Note the true partition is $P_1^{True} = \{\omega_1, ..., \omega_7\}, P_2^{True} = \{\omega_8, ..., \omega_{10}\}$ and $P_3^{True} = \{\omega_{11}, ..., \omega_{15}\}$, which means our main purpose is to see whether we

can move units ω_7 and ω_{11} back to the right place.

Step 1: The marginal distribution is assumed to follow a nomal distribution with $\boldsymbol{b}_k = (\mu_k, \sigma_k^2)$ for each cluster P_k. We first estimate the marginal parameters by using the maximum likelihood estimation method. Then, by maximizing the criterion of equation (5.2), where $X_i = (x_{i1}, x_{i2}), x_{ij} = F_i(T_j), i = 1, ..., 15$, for each w_i and β_{kj}, we compute $h_k(X, \boldsymbol{\gamma}), k = 1, 2, 3$, for each cluster under different values of $\beta = 0$ and $\beta = 1$. Then, we will select the parameter with the highest value of $h_k(X, \boldsymbol{\gamma})$. For example, when $\beta = 0$, for the first cluster, we have:

$$\begin{aligned} h_1(X, \boldsymbol{\gamma}) &= \sum_{\omega_i \in P_1^0} h_i(X, (\boldsymbol{b}_1, \beta = 0)) \\ &= \sum_{\omega_i, i=1,...,6} \max(G^1_{T_1}(x_{i1} + \varepsilon_1, \boldsymbol{b}_1) + G^1_{T_2}(x_{i2} + \varepsilon_2, \boldsymbol{b}_1) + 1, 0) \\ &\quad - \max(G^1_{T_1}(x_{i1} + \varepsilon_1, \boldsymbol{b}_1) + G^1_{T_2}(x_{i2} - \varepsilon_2, \boldsymbol{b}_1) + 1, 0) \\ &\quad - \max(G^1_{T_1}(x_{i1} - \varepsilon_1, \boldsymbol{b}_1) + G^1_{T_2}(x_{i2} + \varepsilon_2, \boldsymbol{b}_1) + 1, 0) \\ &\quad + \max(G^1_{T_1}(x_{i1} - \varepsilon_1, \boldsymbol{b}_1) + G^1_{T_2}(x_{i2} - \varepsilon_2, \boldsymbol{b}_1) + 1, 0) = 5.134 \end{aligned}$$

where $G^1_{T_j}(x; \boldsymbol{b}_1)$, are the normally distributed marginal distributions estimated based on $F_i(T_j), i = 1, ..., 6$. Similarly, when $\beta = 1$, for the first cluster $P_1^0 = (\omega_1, ..., \omega_6)$, we have:

$$\begin{aligned} h_1(X, \boldsymbol{\gamma}) &= \sum_{\omega_i \in P_1^0} h_i(X, (\boldsymbol{b}_1, \beta = 1)) \\ &= \sum_{\omega_i, i=1,...,6} \min(G^1_{T_1}(x_{i1} + \varepsilon_1, \boldsymbol{b}_1), G^1_{T_2}(x_{i2} + \varepsilon_2, \boldsymbol{b}_1)) \\ &\quad - \min(G^1_{T_1}(x_{i1} + \varepsilon_1, \boldsymbol{b}_1), G^1_{T_2}(x_{i2} - \varepsilon_2, \boldsymbol{b}_1)) \\ &\quad - \min(G^1_{T_1}(x_{i1} - \varepsilon_1, \boldsymbol{b}_1), G^1_{T_2}(x_{i2} + \varepsilon_2, \boldsymbol{b}_1)) \\ &\quad + \min(G^1_{T_1}(x_{i1} - \varepsilon_1, \boldsymbol{b}_1), G^1_{T_2}(x_{i2} - \varepsilon_2, \boldsymbol{b}_1)) = 5.134. \end{aligned}$$

Table 6.9: Induction of the copulas from the initial partition.

Class	Copula	$h_k(X, \boldsymbol{\gamma})$
$P_1^0 = (\omega_1, ..., \omega_6)$	$\beta = 0$	5.134
	$\beta = 1$	5.134
$P_2^0 = (\omega_7, ..., \omega_{11})$	$\beta = 0$	0.674
	$\beta = 1$	0.011
$P_3^0 = (\omega_{12}, ..., \omega_{15})$	$\beta = 0$	3.434
	$\beta = 1$	3.434

For the second cluster $P_2^0 = (\omega_7, ..., \omega_{11})$ and $\beta = 0$, we have:

$$
\begin{aligned}
h_2(X, \boldsymbol{\gamma}) &= \sum_{\omega_i \in P_2^0} h_i(X, (\boldsymbol{b}_2, \beta = 0)) \\
&= \sum_{\omega_i, i=7,...,11} \max(G^2_{T_1}(x_{i1} + \varepsilon_1, \boldsymbol{b}_2) + G^2_{T_2}(x_{i2} + \varepsilon_2, \boldsymbol{b}_2) + 1, 0) \\
&\quad - \max(G^2_{T_1}(x_{i1} + \varepsilon_1, \boldsymbol{b}_2) + G^2_{T_2}(x_{i2} - \varepsilon_2, \boldsymbol{b}_1) + 1, 0) \\
&\quad - \max(G^2_{T_1}(x_{i1} - \varepsilon_1, \boldsymbol{b}_2) + G^2_{T_2}(x_{i2} + \varepsilon_2, \boldsymbol{b}_2) + 1, 0) \\
&\quad + \max(G^2_{T_1}(x_{i1} - \varepsilon_1, \boldsymbol{b}_2) + G^2_{T_2}(x_{i2} - \varepsilon_2, \boldsymbol{b}_2) + 1, 0) = 5.134
\end{aligned}
$$

where $G^2_{T_j}(., \boldsymbol{b}_2)$, are the normally distributed marginal distributions estimated based on $F_i(T_j), i = 7, ..., 11$. The other results of $h_k(X, \boldsymbol{\gamma})$ are calculated by similar methods and are summarized in Table 6.9.

Based on the assumed initial partition, we have the initial probability to be $p_1 = 6/15 = 2/5$, $p_2 = 5/15 = 1/3$, $p_3 = 4/15$. Based on these probabilities, we need to further calculate

the likelihood of the mixture model as:

$$L(X, \boldsymbol{\gamma}) = \sum_{k=1}^{3} p_k h_k(X, \boldsymbol{\gamma}_k) = \sum_{k=1,2} p_k h_k(X, (\boldsymbol{b}_k, \beta_k)$$

for different choices of β. For example, when $\beta = 0$ for all the three clusters, we have:

$$\begin{aligned} L(X, \boldsymbol{\gamma}) &= p_1 \times h_1(.; \beta = 0) + p_2 \times h_2(.; \beta = 0) + p_3 \times h_3(.; \beta = 0) \\ &= (2/5) \times 5.134 + (1/3) \times (0.674) + (4/15) \times 3.434 = 3.194. \end{aligned}$$

From Table 6.9, we can see that $h_k(X, \boldsymbol{\gamma})$ has the same values when $\beta = 0$ and $\beta = 1$ for the first cluster and third cluster. This indicates that to achieve the best likelihood we only need to decide the β for the second cluster. It is obvious that $h_k(X, \beta = 0) > h_k(X, \beta = 1)$. Therefore, as long as $\beta = 0$ for the second cluster, we will have the best likelihood no matter which value we choose for the β in the first cluster and third cluster.

Step 2: For this step, we will show the allocation process when $\beta = 0$ for all the three clusters. When $\beta = 0$, $h_{\omega_i}(X, \boldsymbol{\gamma})$ is evaluated under a certain cluster for $k = 1, 2, 3,\ \omega_i, i = 1, ..., 15$, as

$$\begin{aligned} h^k_{\omega_i}(X, \boldsymbol{\gamma}) &= C((G^k_{T_1}(x_1 + \varepsilon_1, \boldsymbol{b}_k), G^k_{T_2}(x_2 + \varepsilon_2, \boldsymbol{b}_k))) - C((G^k_{T_1}(x_1 + \varepsilon_1, \boldsymbol{b}_k), G^k_{T_2}(x_2 - \varepsilon_2, \boldsymbol{b}_k))) \\ &\quad - C((G^k_{T_1}(x_1 - \varepsilon_1, \boldsymbol{b}_k), G^k_{T_2}(x_2 + \varepsilon_2, \boldsymbol{b}_k))) + C((G^k_{T_1}(x_1 - \varepsilon_1, \boldsymbol{b}_k), G^k_{T_2}(x_2 - \varepsilon_2, \boldsymbol{b}_k))) \\ &= \max(G^k_{T_1}(x_1 + \varepsilon_1, \boldsymbol{b}_k) + G^k_{T_2}(x_2 + \varepsilon_2, \boldsymbol{b}_k) + 1, 0) \\ &\quad - \max(G^k_{T_1}(x_1 + \varepsilon_1, \boldsymbol{b}_k) + G^k_{T_2}(x_2 - \varepsilon_2, \boldsymbol{b}_k) + 1, 0) \\ &\quad - \max(G^k_{T_1}(x_1 - \varepsilon_1, \boldsymbol{b}_k) + G^k_{T_2}(x_2 + \varepsilon_2, \boldsymbol{b}_k) + 1, 0) \\ &\quad + \max(G^k_{T_1}(x_1 - \varepsilon_1, \boldsymbol{b}_k) + G^k_{T_2}(x_2 - \varepsilon_2, \boldsymbol{b}_k) + 1, 0) \end{aligned}$$

where $G^k_{T_j}(., \boldsymbol{b}_k)$, are the normally distributed marginal distributions estimated by maximum

Table 6.10: Allocation of each unit to the best fit class.

Unit	$h^1_{\omega_i}(X, \boldsymbol{\gamma})$	$h^2_{\omega_i}(X, \boldsymbol{\gamma})$	$h^3_{\omega_i}(X, \boldsymbol{\gamma})$	Class membership
ω_7	0.899	0	0	P_1
ω_8	0	0.229	0	P_2
ω_9	0	0.226	0	P_2
ω_{10}	0	0.219	0	P_2
ω_{11}	0	0	0.957	P_3

likelihood estimation. Since we are only interested in how to allocate units ω_7 and ω_{11}, Table 6.10 shows the results of $h_k(X, \gamma),\ k = 1, 2, 3$, for $\omega_7, ..., \omega_{11}$.

Then, from Table 6.10, it is obvous that we should move ω_7 from the second class P_2^0 to the first class P_1^0 and move ω_{11} from the second calss P_2^0 to the thrid class P_3^0, which is exactly the true partition.

Step 3: Repeat Step 1 and Step 2 under the updated partitions and calculate the $h_{\omega_i}(X, \boldsymbol{\gamma})$ for each unit. The results show that there are no more reallocations to be made and therefore the final partition is $P_1^{True} = \{\omega_{1,}..., \omega_7\}, P_2^{True} = \{\omega_8, ..., \omega_{10}\}$ and $P_3^{True} = \{\omega_{11}, ..., \omega_{15}\}$.

6.1.4 Example 4: Parametric Estimation

In this simulation study, we will apply parametric estimation of the parameters when the joint distribution function of distributions H is differentiable. Further, unlike the copula of equation 3.4 used in example 1 and example 2, in this example, we use four Archimedean copulas (the Frank copula given by equation 3.5, the Clayton copula given by 3.6, the Joe copula given by equation 3.7 and the Gumbel copula given by 3.8) and one Elliptical copula (the Gaussian copula given by equation 3.9) introduced in section 3.3. Here, all the copula functions have one paramter β in the two-dimensional copula $C(\mu, \nu; \beta)$.

Data Simulation

The sample data are simulated from three clusters with each coming from a Beta distribution $Beta(\alpha, \beta)$ as described in Table 6.11. Figure 6.7 shows the three clusters from left to the right. The x-axis is the index from 1 to 15000 and the y-axis represents the values of the simulated data. There is a large overlapped area between the first cluster and the second cluster and the second cluster is slightly overlapped with the third one in the y-axis direction. Our goal is to partition this simulated dataset into three clusters with parametric estimation for the symbolic mixture decomposition algorithm introduced in section 5.2.2.

The Symbolic Mixture Decomposition

Based on the simulated classical data points (original data), we segment them into 150 units as our symbolic dataset each consisting of 100 successive points. Each unit is described by a cumulative distribution function $F_i, i = 1, ..., 150$, over 100 single values of points. The cumulative distributions are shown in Figure 6.8. We use the similar way as in example 1 and example 2 of defining T values with dimension n equals 2. The two vertical dotted lines in Figure 6.8 indicate the values of $T_1 = 0.45$ and $T_2 = 0.65$. They represent distinct cumulative values of $F_i, i = 1, ..., 150$. They are also separate from each other with some distance to describe the lower value and the higher values, respectively. From Figure 6.8, we can see an obvious separation of these three clusters described by distributional data. Table 6.12 describes the details of each of the three symbolic clusters. According to Table 6.12, the first 70 distributional data $(\omega_1, ..., \omega_{70})$ come from the first cluster (i.e., from a $Beta(2, 2)$ distritbuion), the next 30 distributional data $(\omega_{71}, ..., \omega_{100})$ come from the second cluster (i.e., from a $Beta(1, 3)$ distritbuion) and the rest of data come from the third cluster (i.e., from a $Beta(5, 1)$ distritbuion). To examine the effectiveness of our algorithm, we first set up the starting partition as $P_1^0 = (\omega_1, ..., \omega_{80}), P_2^0 = (\omega_{81}, ..., \omega_{110})$ and $P_3^0 = (\omega_{111}, ..., \omega_{150})$. This means that we assign some of the distributional data into a wrong cluster (as shown

in Table 6.13). We hope to see that the algorithm relocates the observations to their right cluster.

Step 1: Obtain the $F_i(T_j), i = 1, ..., 150,\ j = 1, 2$, values. After having calculated the 150 cumulative distribution functions, with the pre-defined T_1 and T_2, it is easy to obtain the $F_i(T_j)$ values to be estimated for the marginal distributions. We define $y_{ij} = F_i(T_j), i = 1, ..., 150, j = 1, 2$.

Step 2: For the first iteration P^0, calculate the log-likelihood and AIC values for each candidate copula function with y_{ij} and select the copula function with the largest log-likelihood value and smallest AIC for each cluster. The first cluster has the observations $y_{ij} = F_i(T_j), i = 1, ..., 80, j = 1, 2$; the second cluster has the obserations $y_{ij} = F_i(T_j), i = 81, ..., 110, j = 1, 2$; and the third cluster has the observations $y_{ij} = F_i(T_j), i = 111, ..., 150, j = 1, 2$. For each cluster, we use five candidate copulas, specifically, the Gaussian copula, the Gumbel copula, the Joe copula, the Frank copula and the Clayton copula. For each copula, we calculate the log-likelihood value and AIC value based on y_{ij}, for $\omega_i \in P_k^0, j = 1, 2, k = 1, 2, 3$, with equation 6.5 and 6.6, specifically, as

$$lnL_{P_k^0}(X, \boldsymbol{\gamma}_k) = \sum_{\omega_i \in P_k^0} ln\{c(y_{i1}, y_{i2}|\boldsymbol{\gamma}_k)\} \tag{6.5}$$

$$AIC_{P_k^0} = -2 \sum_{\omega_i \in P_k^0} ln\{c(y_{i1}, y_{i2}|\boldsymbol{\gamma}_k)\} + 2l \tag{6.6}$$

where $c(.)$ is the density function of the candidate copula function given in section 3.3 and l is the number of parameters. In this example, $l = 1$. The associated parameters are esimated by using the R function **fitCopula** which was introduced in section 4.2.3. Under the assumption of the starting partitions, Tables 6.14 - 6.16 summarize the estimated parameters

and corresponding log-likelihood and AIC values for each candidate copula function of each cluster, $k = 1, 2, 3$, respectively.

Table 6.14 shows results for the estimation of the candidate copula functions along with the log-likelihood and AIC values for the first cluster $k = 1$. From Table 6.14, the Joe copula function (with estimated parameter $\beta = 2.88$) has the largest log-likelihood value at $lnL(.) = 33.74$ and the smallest AIC at AIC = -65.48. Therefore, this Joe copula is selected as the final copula model for the first cluster. Figure 6.9 is the visualization of the density function (left plot) and the probability function (right plot) for the Joe copula with this estimated paramter $\beta = 2.88$.

For the second cluster $k = 2$, Table 6.15 shows the estimated parameters and corresponding log-likelihood and AIC values for the five candidate copula functions. We see that the Clayton copula (with estimated parameter $\beta = 2.668$) achieves the best performance with the $lnL(.) = 15.83$ and AIC = -29.66. The density plot (left plot) and the probability plot (right plot) for the Clayton copula with the estimated paramter $\beta = 2.667$ are shown in Figure 6.10.

For the third cluster, $k = 3$, Table 6.16 summarizes the estimation for the five candidate copula functions. The Clayton copula also achieves the best performance for the third cluster with estimated parameter $\beta = 0.479$, $lnL(.) = 1.673$ and AIC = -1.34. The visualization of the density function plot (left plot) and probability plot (right plot) for the Clayton copula with esimated parameter $\beta = 0.48$ are shown in Figure 6.11.

Step 3: Fit the final selected copula models from Step 2 for each cluster and obtain the density values $h^k_{\omega_i}(X, \boldsymbol{\gamma}), i = 1, ..., 150$, for all the units $\omega_i, i = 1, ..., 150$ (distributional-data), under the certain cluster P^0_k by the R function **dCopula** introduced in section 4.2.3.

Table 6.11: Simulated Data for Parametric Estimation of One Dimension Case X_1.

Class	# sample (single value of points)	Distribution
1	7000	$Beta(2, 2)$
2	3000	$Beta(1, 3)$
3	5000	$Beta(5, 1)$

Figure 6.7: Scatter Plot of Simulated Data.

Table 6.12: Summary Table of Simulated Symbolic clusters.

Class	#sample (distributional data)	#original data (created from)
1	70	100
2	30	100
3	50	100

Table 6.13: Assignment of Starting Partition.

Class	True Partition	Starting Partition
1	$P_1 = (\omega_1, ..., \omega_{70})$	$P_1^0 = (\omega_1, ..., \omega_{80})$
2	$P_2 = (\omega_{71}, ..., \omega_{100})$	$P_2^0 = (\omega_{81}, ..., \omega_{110})$
3	$P_3 = (\omega_{101}, ..., \omega_{150})$	$P_3^0 = (\omega_{111}, ..., \omega_{150})$

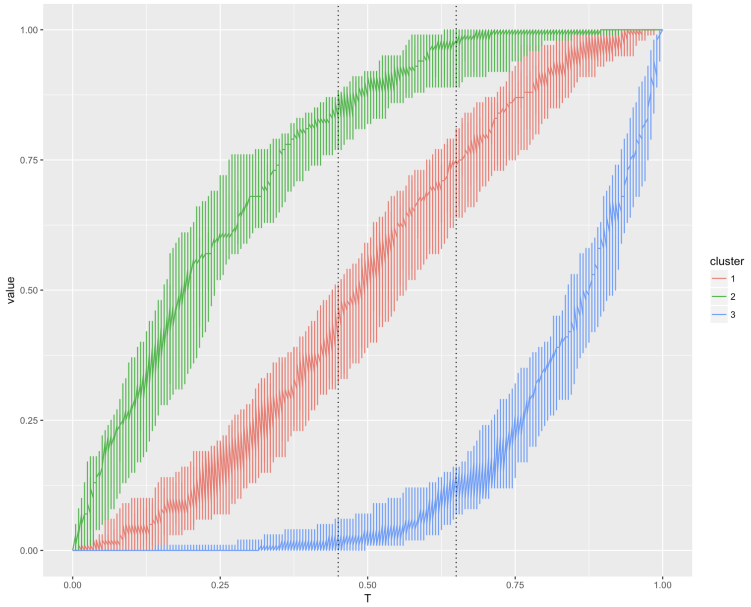

Figure 6.8: The Cumulative Distrbution Functions $F_i, i = 1, 2, ..., 150$.

Table 6.14: Estimation of Candidate Copula Functions: First Iteration and First Cluster. (Fit based on 80 observations $(F_i(T_1), F_i(T_2)), i = 1, ..., 80$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.696	23.75	-45.51
Gumbel	2.133	31.38	-60.76
Joe	2.880	33.74	-65.48
Frank	5.593	23.00	-44.00
Clayton	1.032	12.57	-23.14

Table 6.15: Estimation of Candidate Copula Functions: First Iteration and Second Cluster.(Fit based on 30 observations $(F_i(T_1), F_i(T_2)), i = 81, ..., 110$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.826	14.68	-27.36
Gumbel	2.158	10.57	-19.14
Joe	2.270	6.70	-11.41
Frank	8.300	14.76	-27.52
Clayton	2.668	15.83	-29.66

Table 6.16: Estimation of Candidate Copula Functions: First Iteration and Third cluster.(Fit based on 40 observations $(F_i(T_1), F_i(T_2)), i = 111, ..., 150)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.285	1.06	-0.11
Gumbel	1.158	0.59	0.82
Joe	1.154	0.24	1.53
Frank	1.306	0.78	0.44
Clayton	0.479	1.67	-1.34

Table 6.17 shows part of the results. The second to the fourth columns show the values of $h^k_{\omega_i}(X, \boldsymbol{\gamma}), k = 1, 2, 3$, for each unit i evaluated under a certain cluster. The last column shows the allocation results of the assigned cluster with the highest $h^k_{\omega_i}(X, \boldsymbol{\gamma})$ value. We can see that most of the units are allocated into the right cluster. For those units that are assigned with a wrong label under the starting partition assumption, the majority of them are successfully relocated to the right places, except for one unit. For example, units $\omega_{71}, ..., \omega_{80}$ are simulated from the second cluster and assigned to the first cluster in the starting partitions. Fortunately, they are all correctly relocated to the second cluster after the first iteration. Units $\omega_{101}, ..., \omega_{110}$ are simulated from the third cluster which are assigned to the second cluster for the starting partitions. The majority of them are correctly relocated to the true class except for one unit (ω_{107}). This unit has the highest $h^k_{\omega_i}(X, \boldsymbol{\gamma}) = 2.425$ when $k = 1$; it is re-allocated to the first cluster.

Step 4: Repeat Step 1 to Step 3 under the updated partition labels. There are no more re-allocations. Therefore, we stop and the partition result is as summarized in Figure 6.12. From Figure 6.12, we can see that for the first cluster, 61 units out of 70 units are correctly allocated with an accuracy equal to 87%. The second cluster has the same accuracy of 87%

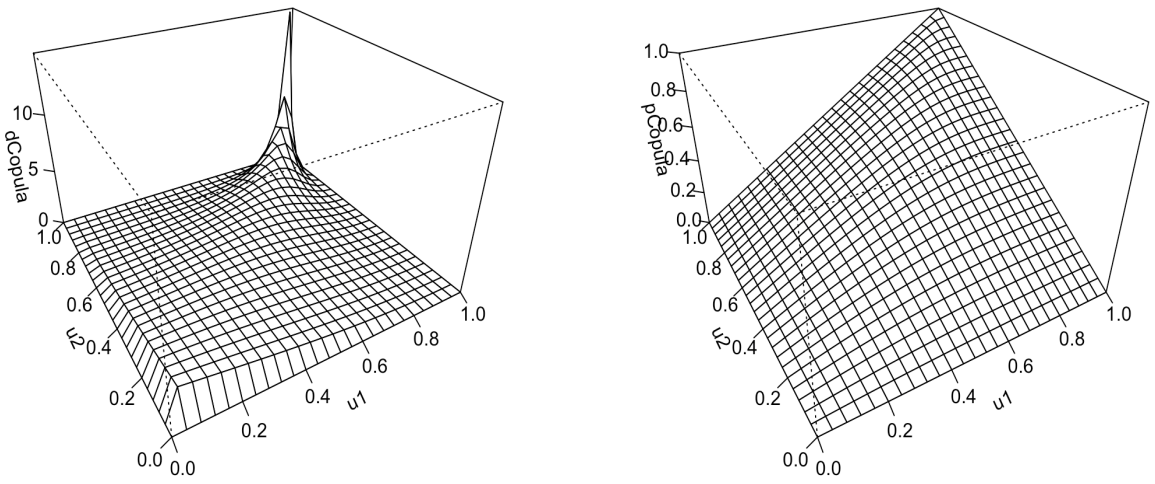

Figure 6.9: Visualization of final selected copula model for the first cluster: Density plot (left) and Copula plot (right) of the Joe copula with parameter 2.88.

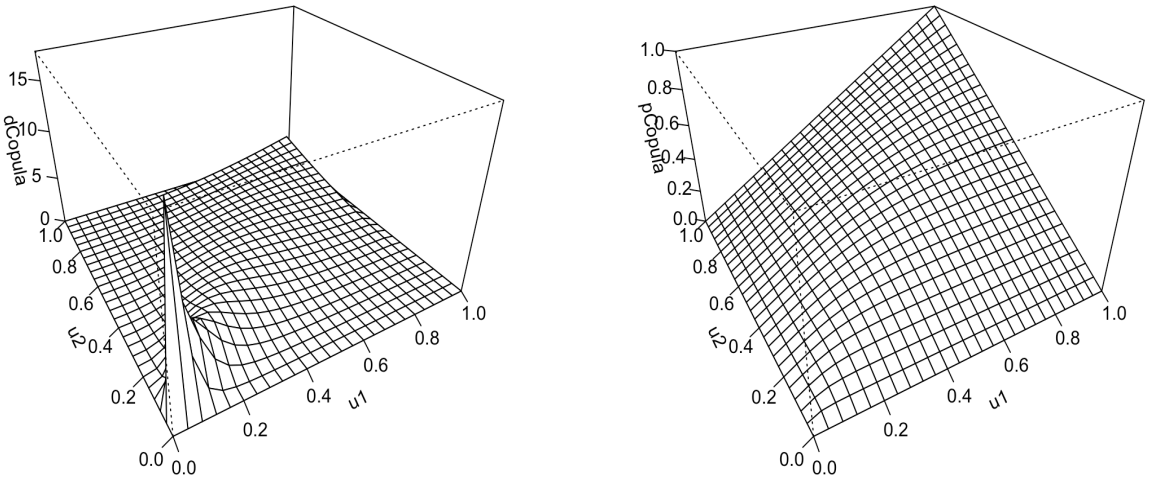

Figure 6.10: Visualization of final selected copula model for the second cluster: Density plot (left) and Copula plot (right) of the Clayton Copula with parameter 2.667.

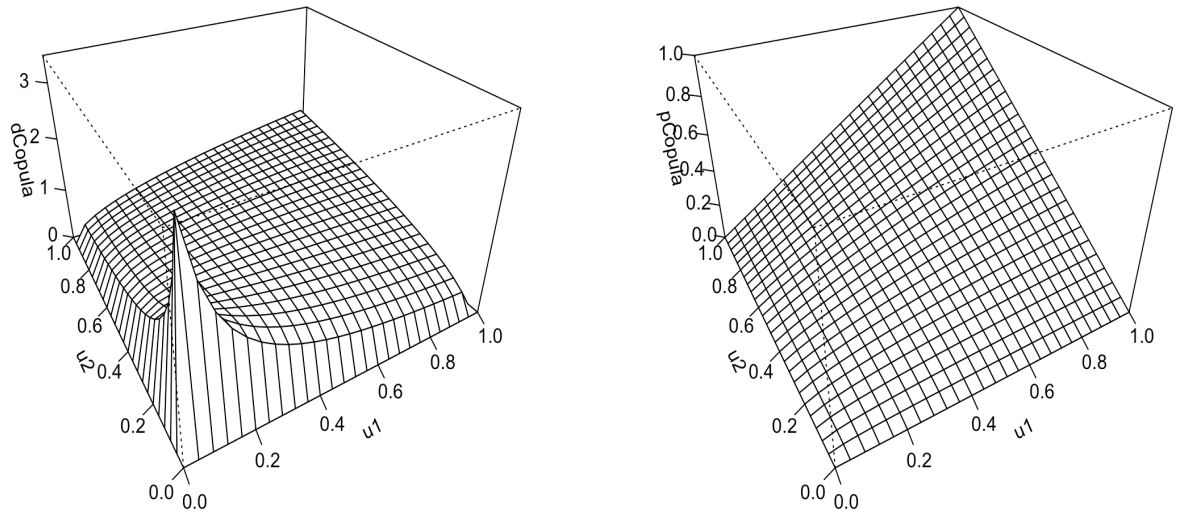

Figure 6.11: Visualization of final selected copula model for the third cluster: Density plot (left) and Copula plot (right) of the Clayton copula with parameter 0.48.

		True Class Assignment		
		C1	C2	C3
Predicted Class	C1	61	0	7
	C2	9	26	1
	C3	0	4	42
	Accuracy	0.87	0.87	0.84
Overall Accuracy: 0.86				

Figure 6.12: Summary Results of Clustering with Parametric Estimation with One Variable.

Table 6.17: Allocation of each unit to the best fit class.

Unit	$h^1_{\omega_i}(X, \boldsymbol{\gamma})$	$h^2_{\omega_i}(X, \boldsymbol{\gamma})$	$h^3_{\omega_i}(X, \boldsymbol{\gamma})$	Class membership
ω_1	1.006	0.728	0.618	P_1
ω_2	0.992	0.715	0.705	P_1
ω_3	1.025	0.859	0.751	P_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{10}	1.014	0.706	0.499	P_1
ω_{11}	1.038	0.939	0.847	P_1
ω_{12}	1.023	0.907	0.847	P_1
ω_{13}	0.998	00.840	0.914	P_1
ω_{14}	1.029	1.002	0.966	P_1
ω_{15}	1.049	1.069	0.972	P_2
ω_{16}	1.025	0.859	0.751	P_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{69}	1.054	0.919	0.666	P_1
ω_{70}	1.057	1.028	0.859	P_1
ω_{71}	1.345	2.221	0.198	P_2
ω_{72}	1.291	1.975	1.330	P_2
ω_{73}	1.323	2.179	1.851	P_2
ω_{74}	1.336	2.207	0.713	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{79}	1.305	2.328	1.009	P_2
ω_{80}	1.350	2.328	1.009	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{88}	1.299	2.023	1.508	P_2
ω_{89}	1.366	2.488	2.663	P_2
ω_{90}	1.332	2.199	1.059	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{100}	1.377	2.539	1.123	P_2
ω_{101}	1.684	0.322	2.089	P_3
ω_{102}	2.277	0.484	2.319	P_3
ω_{103}	1.536	0.088	2.058	P_3
ω_{104}	1.677	0.029	2.202	P_3
ω_{105}	1.452	0.017	2.111	P_3
ω_{106}	2.116	0.966	2.255	P_3
ω_{107}	2.425	1.919	2.338	P_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{148}	2.110	0.343	2.275	P_3
ω_{149}	2.116	0.966	2.255	P_3
ω_{150}	1.535	0.088	2.058	P_3

with 26 units out of 30 units correctly assigned. The accuracy rate is a little bit lower for the third cluster, which is 84% with 42 units out of 50 units correctly assigned. Therefore, the overall accuracy rate is 86%, which means 86% of the observations are predicted with the correct cluster labels.

6.2 Simulation Example: Mixture Decomposition for Two Dimensional Data (X_1, X_2)

In section 6.1, we conducted simulations for symbolic clustering with both non-parametric and parametric estimation methods in the one-dimensional case, which means we only have one variable X_1. In this section, we will extend the simulation examples into two-dimensional data (X_1, X_2) and focus on the parametric estimation method by using the algorithm described in section 5.2.3. In this example, we also use four Archimedean copulas (the Frank copula given by equation 3.5, the Clayton copula given by 3.6, the Joe copula given by equation 3.7 and the Gumbel copula given by 3.8) and two Elliptical copula (the Gaussian copula given by equation 3.9 and the t-copula given by equation 3.10) introduced in section 3.3. Thus, all the copula functions have one parameter β in the two-dimensiona copula $C(\mu, \nu; \beta)$.

6.2.1 Data Simulation

In this example, we consider the simulation of two-dimensional data (X_1, X_2) with two clusters $k = 2$. Each cluster contains observations from a different bivariate normal distribution, which is described in Table 6.18, i.e., for the first cluster of the first 7000 observations are

Table 6.18: Simulated Data for Parametric Estimation of Two-Dimension Case.

Class	# sample (single value of points)	Distribution
1	7000	$\mathcal{N}(\begin{pmatrix}3\\0\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix})$
2	3000	$\mathcal{N}(\begin{pmatrix}4\\4\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix})$

from the bivariate normal disribution with

$$(X_1, X_2) \sim \mathcal{N}(\begin{pmatrix}3\\0\end{pmatrix}, \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}) \tag{6.7}$$

and the second cluster contains the last 3000 observations from the bivariate normal distribution with

$$(X_1, X_2) \sim \mathcal{N}(\begin{pmatrix}4\\4\end{pmatrix}, \begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}). \tag{6.8}$$

Figure 6.13 is the scatter plot of the simulated data, where now the axes correspond to the X_1 and X_2 values, respectively. This plot shows that the two groups of data have some overlapped observations in the middle. Our goal is to partition this simulated dataset into two clusters using the parametric estimation method for symbolic mixture decompositions.

6.2.2 The Symbolic Mixture Decomposition

Similarly, as in all the previous examples, we first need to segment the original single values of points into symbolic units, which are the distributional data. In this example, we also use every successive 100 observations as a segment to define the distributional data. However, in the two-dimensional case (X_1, X_2), each classical observation is a vector of two single

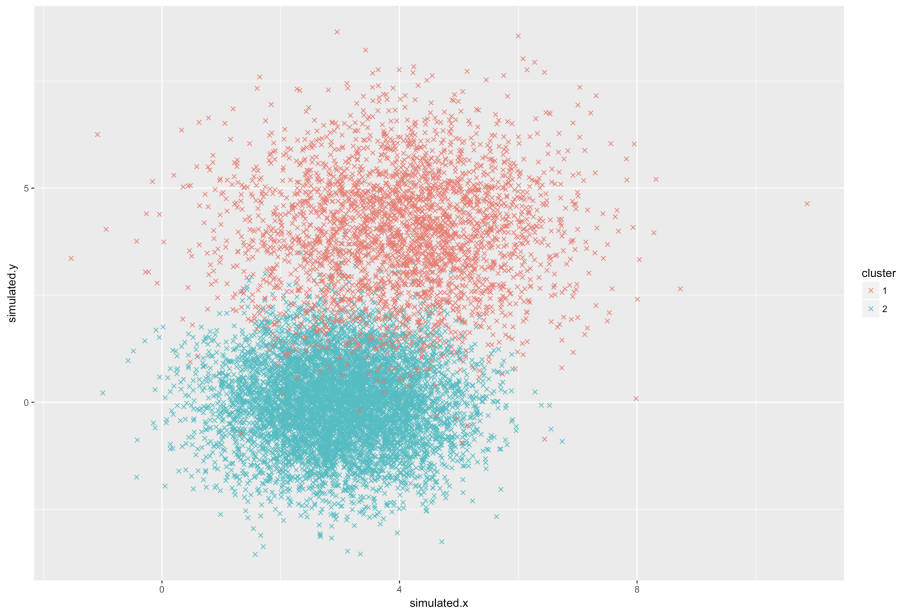

Figure 6.13: Scatter Plot of Simulated Data (Two Dimension).

values of points. To create the distributional units, the segmentation should be applied to each dimension for the same segment of classical observations. This means that each 2-dimensional distributional unit is created by a cumulative distribution function of 100 original single values of points for every dimension and in total we have 100 units for each dimension as: $\omega_i = (F_i^1, F_i^2), i = 1, ..., 100$. Table 6.19 describes the relationship between the 100 units and the symbolic clusters. The first 70 units $(\omega_1, ..., \omega_{70}) = ((F_1^1, F_1^2), ..., (F_{70}^1, F_{70}^2))$ come from the first cluster, and the last 30 units $(\omega_{71}, ..., \omega_{100}) = ((F_{71}^1, F_{71}^2), ..., (F_{100}^1, F_{100}^2))$ come from the second cluster. To examine the effectiveness of our algorithm, we set up the starting partition as $P_1^0 = (\omega_1, ..., \omega_{60}) = ((F_1^1, F_1^2), ..., (F_{60}^1, F_{60}^2))$ and $P_2^0 = (\omega_{61}, ..., \omega_{100}) = ((F_{61}^1, F_{61}^2), ..., (F_{100}^1, F_{100}^2))$. This means that we assign some units into a wrong cluster (see Table 6.20). We hope to see that the algorithm could relocate the observations to the right cluster.

Step 1: Consider first the first dimension X_1. Define T values and obtain $F_i^1(T_j^1), i = 1, ..., 100, j = 1, 2$, for the first dimension. Figure 6.14 is the visualization of 100 distributional-data for the first dimension X_1. To set up the values of T, as before we use the dimension of T to be $n = 2$ as $T^1 = (T_1^1, T_2^1)$ with $T_1^1 = 3$, $T_2^1 = 4$ (as indicated by the vertical dotted lines in Figure 6.14). From Figure 6.14, we can see that the reason for defining these two values of T is to represent distinct cumulative values of $F_i^1(T_j^1), i = 1, ..., 100, j = 1, 2$. After defining the T values, we evaluate the 100 cumulative distrbution functions for the first dimension X_1 as $F_i^1, i = 1, ..., 100$; we can write $y_{ij}^1 = F_i^1(T_j^1), i = 1, ..., 100, j = 1, 2$.

Then, we calculate the log-likelihood and AIC values from equation 6.5 and equation 6.6 for each of the five candidate copulas function with $y_{ij}^1 = F_i^1(T_j^1)$ for $\omega_i \in P_k^0, j = 1, 2, k = 1, 2$, and select the copula function with the largest log-likelihood value and smallest AIC value for each cluster from equation 6.5 and 6.6, respectively. The associated parameters are esimated by using the R function **fitCopula** which is introduced in section 4.2.3. Under the assumption of the starting partition, Table 6.21 and 6.22 summarize the estimated parameters and corresponding log-likelihood and AIC values for each candidate copula function of the first cluster and the second cluster, respectively. For example, for the first cluster of dimension X_1, Table 6.21 shows that the Joe copula with estimated parameter $\beta = 1.435$ has the largest $lnL(.) = 4.15$ and the smallest AIC = -6.31. For the second cluster of dimension X_1, Table 6.22 shows that the Joe copula with estimated parameter $\beta = 3.802$ has the largest $lnL(.) = 23.94$ and the smallest AIC = -45.88.

For the second dimension X_2, we repeat the same procedure that is applied to the first dimension X_1. Figure 6.15 describes the 100 distributional-data we obtained in this case. The T values are defined as $T_1^2 = 0.8$ and $T_2^2 = 2$ (indicated by the vertical dotted lines in

Table 6.19: Summary Table of Simulated Symbolic clusters.

Class	#units	#original observations
1	70	100
2	30	100

Figure 6.15). Then, $y_{ij}^2 = F_i^2(T_j^2), i = 1, ..., 100, j = 1, 2$, are calculated. Table 6.23 and 6.24 show the log-likelihood and AIC values for the candidate copula models for each cluster. For example, for the first cluster of dimension X_2, Table 6.23 shows that the Gaussian copula with estimated parameter $\beta = 0.277$ has the largest $lnL(.) = 1.70$ and the smallest AIC = -1.40. For the second cluster of dimension X_2, Table 6.24 shows that the Gumbel copula with estimated parameter $\beta = 2.385$ has the largest $lnL(.) = 17.62$ and the smallest AIC = -33.24.

The final selected models and the corresponding estimated parameters are summarized in Table 6.25 for the two clusters of each dimension X_1 and X_2. Figure 6.16 - Figure 6.19 are visualizations of the density function and probability function of all the selected copula models. Figure 6.16 shows the density function plot and probability plot for the Joe copula for X_1 of the first cluster $k = 1$. Figure 6.17 shows the density function plot and probability plot for the Joe copula for X_1 of the second cluster $k = 2$. Figure 6.18 shows the density function plot and probability plot for the Gaussian copula for X_2 of the first cluster $k = 1$. Figure 6.19 shows the density function plot and probability plot for the Joe copula for X_2 of the second cluster $k = 2$.

Step 2: Evaluate each unit under a certain cluster by the copula functions selected from Step 1 to obtain the marginal probability values $(G^1(y_{i,T_1^1}^1, y_{i,T_2^1}^1), G^2(y_{i,T_1^2}^2, y_{i,T_2^2}^2)), i = 1, ..., 100$. For example, $G^1(y_{i,T_1^1}^1, y_{i,T_2^1}^1), i = 1, ..., 60$, is obtained by using the R function

Table 6.20: Assignment of Starting Partition.

Class	True Partition	Starting Partition
1	$((F_1^1, F_1^2), ..., (F_{70}^1, F_{70}^2))$	$P_1^0 = ((F_1^1, F_1^2), ..., (F_{60}^1, F_{60}^2))$
2	$((F_{71}^1, F_{71}^2), ..., (F_{100}^1, F_{100}^2))$	$P_2^0 = ((F_{61}^1, F_{61}^2), ..., (F_{100}^1, F_{100}^2))$

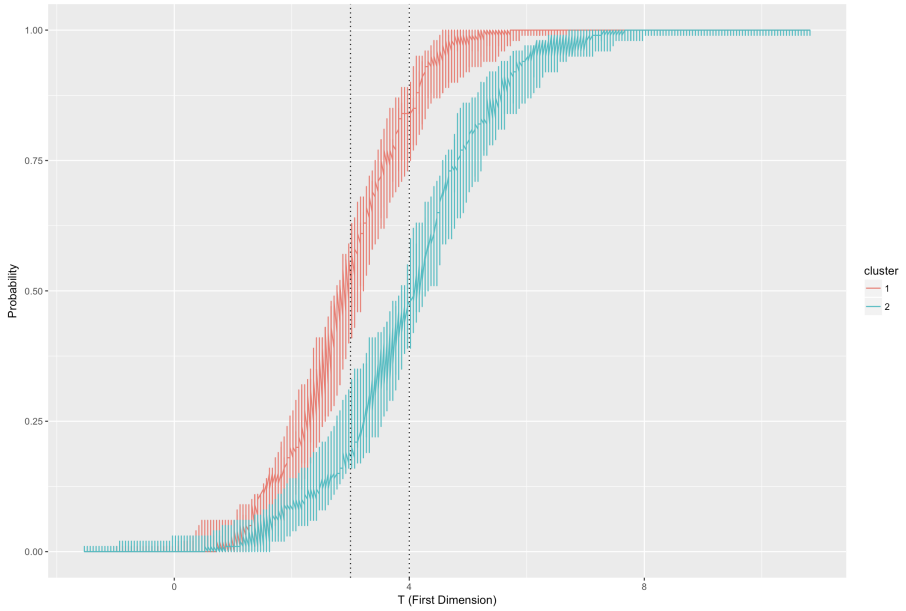

Figure 6.14: The Cumulative Distribution Functions $F_i^1, i = 1, ..., 100$, for the First Dimension X_1.

Table 6.21: Estimation of Candidate Copula Functions: First Iteration and First Dimension X_1 and First cluster $k = 1$. (Fit based on 60 observations $(y_{i1}^1, y_{i2}^1) = (F_i^1(T_1^1), F_i^1(T_2^1)), i = 1, ..., 60$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.283	1.94	-1.88
Gumbel	1.249	3.28	-4.65
Joe	1.435	4.15	-6.31
Frank	1.259	1.13	-0.25
Clayton	0.278	0.319	1.363

Table 6.22: Estimation of Candidate Copula Functions: First Iteration and First Dimension X_1 and Second Cluster $k = 2$ (Fit based on 40 observations $(y^1_{i1}, y^1_{i2}) = (F^1_i(T^1_1), F^1_i(T^1_2)), i = 61, ..., 100$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.841	21.83	-41.66
Gumbel	2.722	23.57	-45.14
Joe	3.802	23.94	-45.88
Frank	8.632	21.13	-40.26
Clayton	4.306	-7.40	16.81

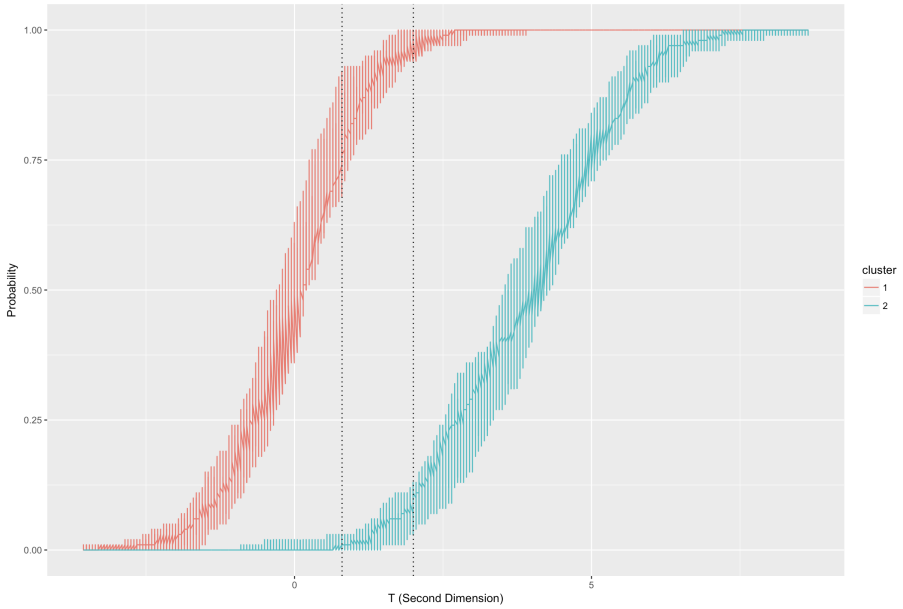

Figure 6.15: The Cumulative Distribution Functions $F^1_i, i = 1, ..., 100$ for the Second Dimension X_2.

Table 6.23: Estimation of Candidate Copula Functions: First Iteration and Second Dimension X_2 and First Cluster $k = 1$. (Fit based on 60 observations $(y_{i1}^2, y_{i2}^2) = (F_i^2(T_1^2), F_i^2(T_2^2)), i = 1, ..., 60)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.277	1.70	-1.40
Gumbel	1.157	1.01	-0.01
Joe	1.195	0.61	0.77
Frank	1.266	1.24	-0.49
Clayton	0.394	1.37	-0.76

Table 6.24: Estimation of Candidate Copula Functions: First Iteration and Second Dimension X_2 and Second Cluster $k = 2$. (Fit based on 40 observations $(y_{i1}^2, y_{i2}^2) = (F_i^2(T_1^2), F_i^2(T_2^2)), i = 61, ..., 100)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.784	15.92	–29.84
Gumbel	2.385	17.62	-33.24
Joe	3.222	18	-34
Frank	7.071	15.88	-29.76
Clayton	1.39	9.00	-16.004

Table 6.25: Final Selection of Copula Function and Estimated Parameter

Dimension cluster	Copula Function	Estimated Parameter β
First Dimension X_1 First Cluster $k = 1$	Joe	1.435
First Dimension X_1 Second Cluster $k = 2$	Joe	3.802
Second Dimension X_2 First Cluster $k = 1$	Gaussian	0.277
Second Dimension X_2 Second Cluster $k = 2$	Joe	3.222

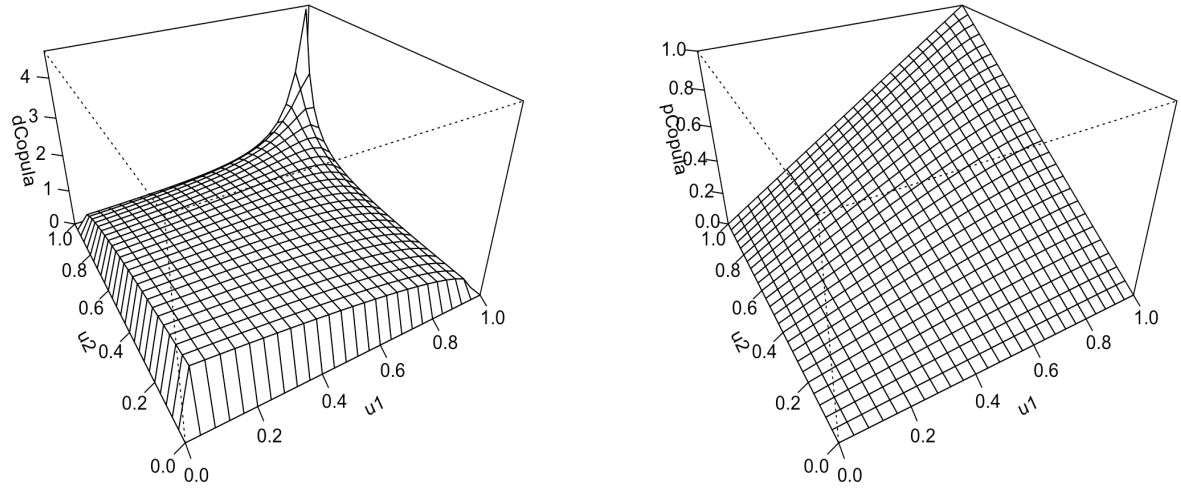

Figure 6.16: Visualization of final selected copula model (First Dimension X_1 and First Cluster $k = 1$): Density plot (left) and Copula plot (right) of the Joe Copula with parameter 1.435.

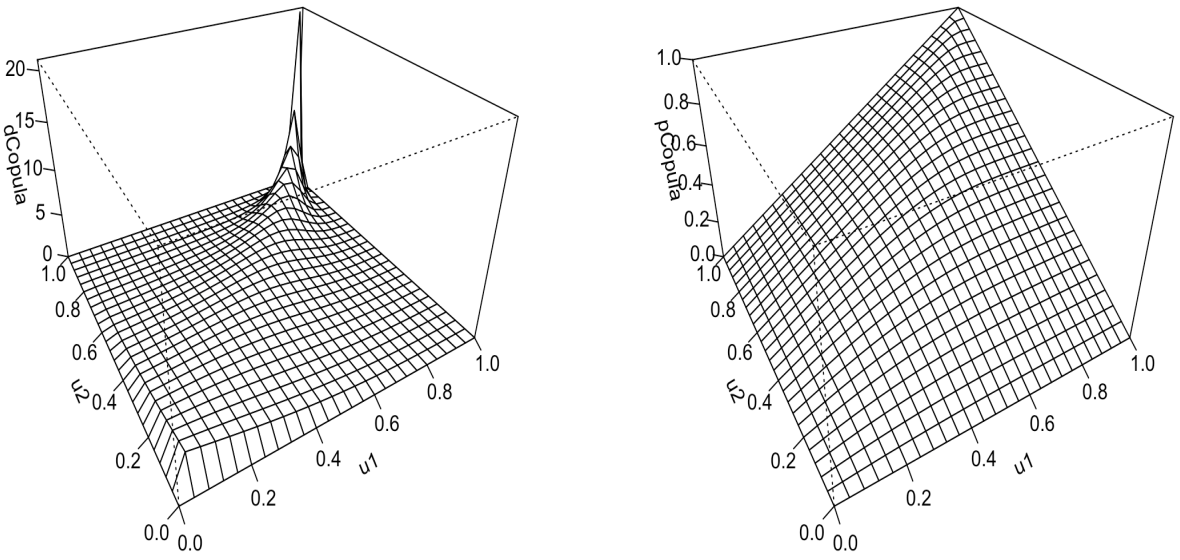

Figure 6.17: Visualization of final selected copula model (First Dimension X_1 and Second Cluster $k = 2$): Density plot (left) and Copula plot (right) of the Joe Copula with parameter 3.802.

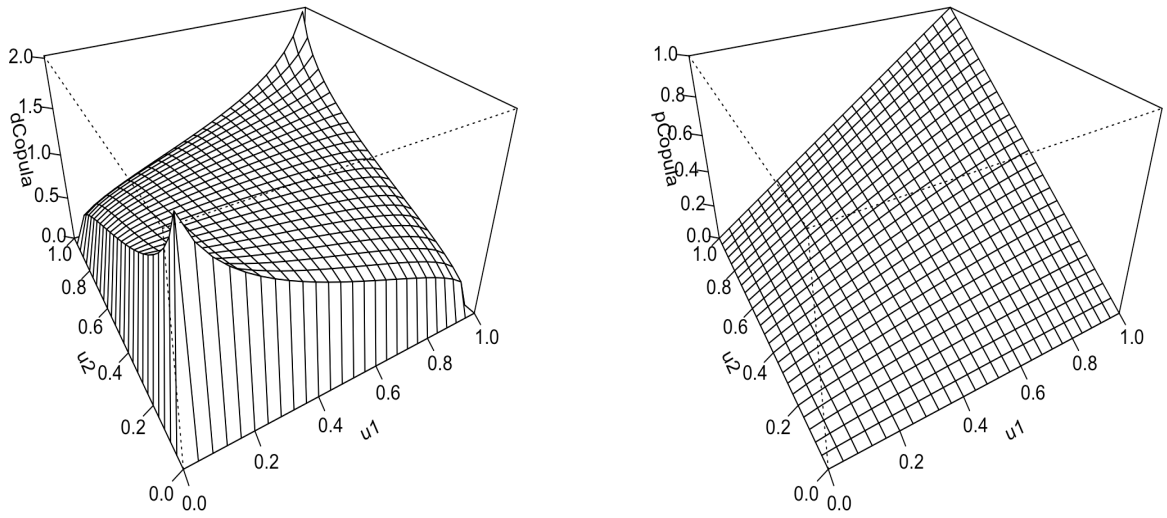

Figure 6.18: Visualization of final selected copula model (Second Dimension X_2 and First Cluster $k = 1$): Density plot (left) and Copula plot (right) of the Gaussian Copula with parameter 0.277.

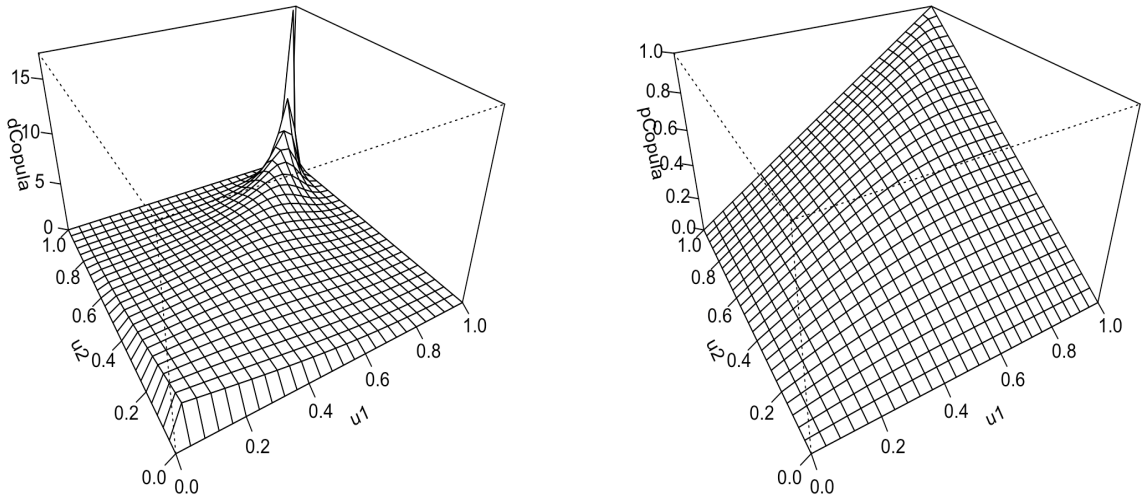

Figure 6.19: Visualization of final selected copula model (Second Dimension X_2 and Second Cluster $k = 2$): Density plot (left) and Copula plot (right) of the Joe Copula with parameter 3.222.

pCopula with the Joe copula with an estimated parameter $\beta = 1.435$ based on $y^1_{ij}, i = 1, ..., 60, j = 1, 2$, for the observations in P^0_1. Likewise, for the observations in P^0_2, $i = 61, ..., 100$, $G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2})$ is obtained by using the R function **pCopula** with the Joe copula with estimated parameter $\beta = 3.802$ based on $y^1_{ij}, i = 61, ..., 100, j = 1, 2$. Then, based on $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), G^2(y^2_{i,T^2_1}, y^2_{i,T^2_2})), i = 1, ..., 100$, we need to model the joint relationship between for the two dimensional units.

By using the same methods of calculating the log-likelihood and AIC values for the candidate copula functions, we select and estimate the copula function for each cluster. Table 6.26 and Table 6.27 summarize the log-likelihood values and AIC values for all the candidate copula models. We should note that the t-copula function is also considered for the first cluster. This is because all the five candidate copula models that we considered previously have low log-likelihood value. Although the t-copula achieves a large improvement according to the $lnL(.) = 2.829$, while the others are all less than 1, the final clustering accuracy is just improved a little bit. We select the t-copula here for the first cluster to make it a possible choice for future analysis. Table 6.26 shows that the Joe copula achieves the best performance with the largest log likelihood value at $lnL(.) = 14.84$ and smallest AIC = -27.68 for the second cluster (with an estimated parameter $\beta = 2.731$). Figure 6.20 shows the density function and the probability distribution plots for the t-copua for the cluster $k = 1$. Figure 6.21 shows the density function and the probability distribution plots for the Joe copula for the cluster $k = 2$.

Step 3: Fit the final selected copula models from Step 2 under a certain cluster with the marginal probability values $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), G^2(y^2_{i,T^2_1}, y^2_{i,T^2_2})), i = 1, ..., 100$, and obtain the density values $h^k_{\omega_i}(X, \gamma), i = 1, ..., 100, k = 1, 2$, for all the units $\omega_i, i = 1, ..., 100$, under each cluster. Table 6.28 records part of these results. The majority of the units are allocated to

Table 6.26: Estimation of Candidate Copula Functions for Joint Distribution of Two Dimensions: First Iteration and First cluster. (Fit based on 60 observations $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), G^2(y^2_{i,T^2_1}, y^2_{i,T^2_2})), i = 1, ..., 60)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	-0.082	0.16	1.69
Gumbel	1.000	$-4.13 * 10^{-8}$	2.00
Joe	1.00	$-4.45 * 10^{-8}$	2.00
Frank	0.146	0.01	1.97
Clayton	0.053	-0.015	2.03
t-Copula	0.091	2.829	-1.658

Table 6.27: Estimation of Candidate Copula Functions for Joint Distribution of Two Dimensions: First Iteration and Second cluster. (Fit based on 40 observations $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), G^2(y^2_{i,T^2_1}, y^2_{i,T^2_2})), i = 61, ..., 100)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.716	11.74	-21.48
Gumbel	2.076	13.81	-25.62
Joe	2.731	14.84	-27.68
Frank	5.701	11.57	-21.14
Clayton	1.073	5.78	-9.55
t-Copula	0.7172	11.79	-21.58

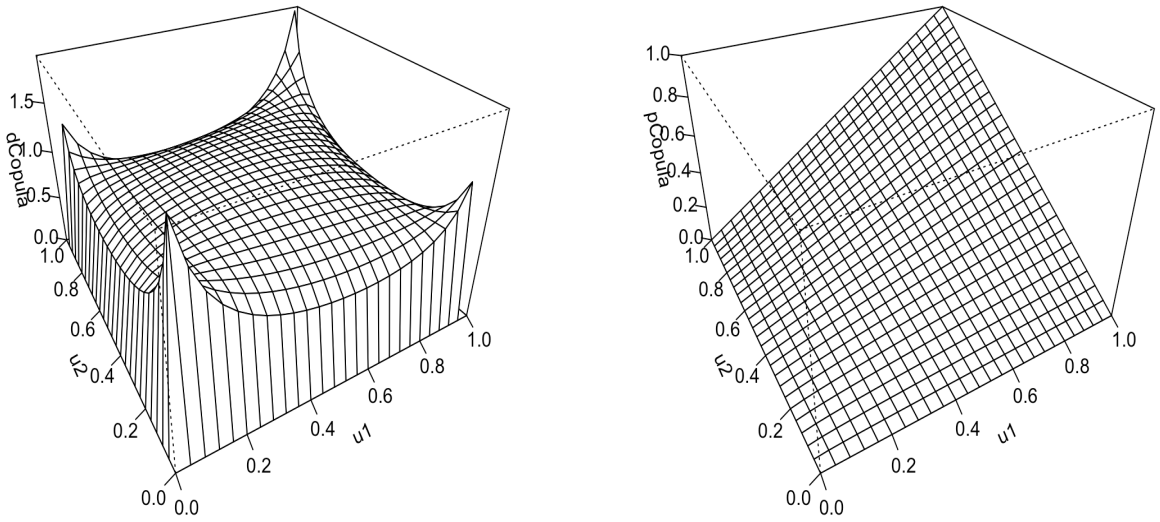

Figure 6.20: Visualization of final selected copula model (Joint Distribution for First Cluster): Density plot (left) and Copula plot (right) of the t-Copula with parameter 2.829.

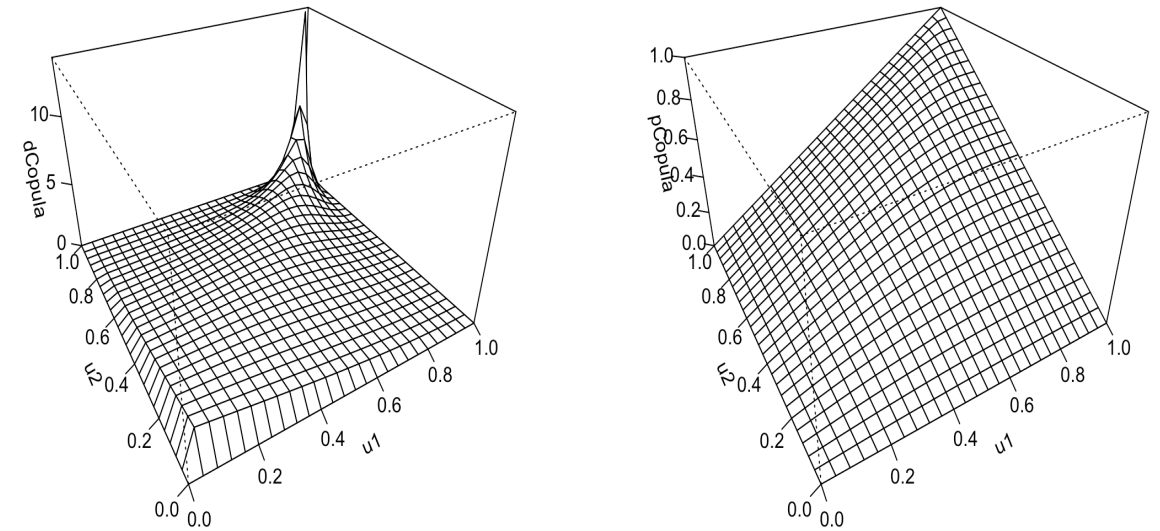

Figure 6.21: Visualization of final selected copula model (Joint Distribution for Second Cluster): Density plot (left) and Copula plot (right) of the Joe Copula with parameter 2.731.

the right cluster. For those units that are assigned with a wrong label under the starting partition assumption, $\omega_{60}, ..., \omega_{70}$, 80% of them are successfully reallocated to the right cluster. There are several units whose density values are both equal to 0 when evaluated under each cluster. This produces some uncertain units that our model can not tell from which cluster they come. The reason for having density values of 0 is because of the T values defined in the first step. Remember, we have $(y_{ij}^1, y_{ij}^2) = (F_i^1(T_j), F_i^2(T_j)), i = 1, ..., 100, j = 1, 2$. Therefore, it is quite possible to have probability values $y_{ij}^1 = 0 or y_{ij}^2 = 0, i = 1, ..., 100, j = 1, 2$. We will discuss the choices of T further in section 6.4.

Step 4: Repeat Step 1 to Step 3 under the updated partition labels. There is no obvious change in estimated density values and no more re-allocation. Therefore, we stop. Figure 6.22 summarizes the partition result. The overall accuracy is 84% with 91% for the first cluster and 67% for the second cluster. The reason that the accuracy of the first cluster is much higher than that for the second cluster is because of the unbalanced data as the first cluster has 70 units and the second cluster has only 30 units.

6.3 Simulation Example: Mixture Decomposition for Multidimension Data

In section 6.2, we conducted a simulation for a mixture decomposition model with the parametric estimation method when the observations are two-dimensional as (X_1, X_2). In this section, we will apply the similar algorithm for multidimensional data $p = 3$ as (X_1, X_2, X_3) and also focus on the parametric estimation method. The allocation process is essentially the same as the 2-dimensional case. However, this example is a more general illustration for any multi-dimensional case. In this example, we also use four Archimedean copulas (the Frank copula given by equation 3.5, the Clayton copula given by 3.6, the Joe copula given by

Table 6.28: Allocation of each unit to the best fit class.

Unit	$h^1_{\omega_i}(X,\gamma)$	$h^2_{\omega_i}(X,\gamma)$	Class membership
ω_1	1.072	0.944	P_1
ω_2	1.060	0.982	P_1
ω_3	1.052	0.791	P_1
ω_4	1.041	0.686	P_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{12}	0.994	0.531	P_1
ω_{13}	1.069	1.095	P_2
ω_{14}	1.027	0.724	P_1
ω_{15}	0.996	0.541	P_1
ω_{16}	1.018	0.639	P_1
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{69}	1.049	0.811	P_1
ω_{70}	1.072	1.178	P_2
ω_{71}	0.716	2.021	P_2
ω_{72}	0.846	2.019	P_2
ω_{73}	0.592	1.852	P_2
ω_{74}	0.607	1.629	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{79}	0.000	0.000	P_2
ω_{80}	0.924	1.981	P_2
ω_{81}	0.000	0.000	P_2
ω_{82}	0.600	1.788	P_2
ω_{83}	0.000	0.000	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{99}	0.532	1.869	P_2
ω_{100}	0.715	1.945	P_2

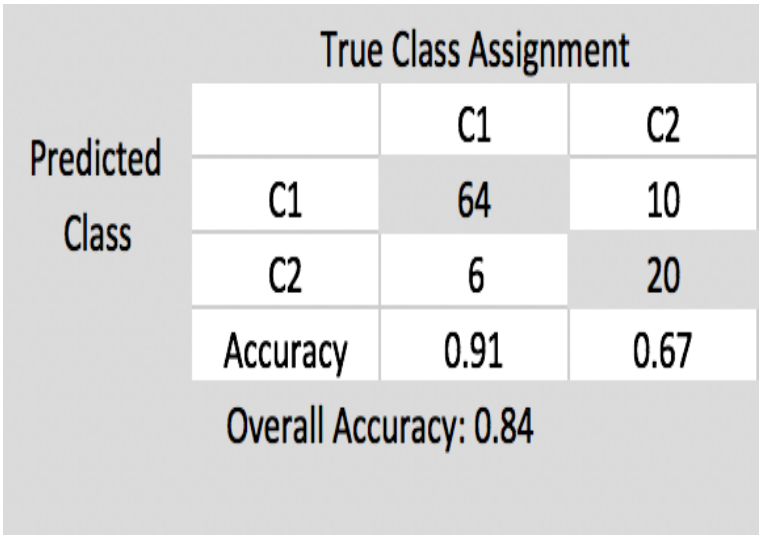

Figure 6.22: Clustering Results and Accuracy Table

equation 3.7 and the Gumbel copula given by 3.8) and one Elliptical copula (the Gaussian copula given by equation 3.9) introduced in section 3.3.

6.3.1 Data Simulation

In this example, we consider the simulation of p-dimensional data with two clusters for observations (X_1, X_2, X_3). Each of the clusters is distributed from a multivariate normal distribution with $p = 3$. The first cluster contains the first 7000 observations simulated from a multivariate normal distribution as

$$(X_1, X_2, X_3) \sim \mathcal{N}(\begin{pmatrix} 1.5 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix})$$

Table 6.29: Simulated Data for Parametric Estimation in Three Dimension Case.

Class	# sample (single value of points)	Distribution
1	7000	$\mathcal{N}(\begin{pmatrix}1.5\\1\\0\end{pmatrix}, \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix})$
2	3000	$\mathcal{N}(\begin{pmatrix}4\\3\\3\end{pmatrix}, \begin{pmatrix}2&0&0\\0&2&0\\0&0&2\end{pmatrix})$

and the last 3000 observations form the second cluster simulated from a multivariate normal distribution as

$$(X_1, X_2, X_3) \sim \mathcal{N}(\begin{pmatrix}4\\3\\3\end{pmatrix}, \begin{pmatrix}2&0&0\\0&2&0\\0&0&2\end{pmatrix}).$$

Table 6.29 summarizes the details. Figure 6.23 shows the 3-D scatter plot in the direction of X_1 (the top left one), in the direction of X_2 (the top right one) and in the direction of X_3 (the bottom plot), respectively. We can see there is an obvious overlapped area between the two clusters. Our goal is to partition this simulated sample dataset into two clusters with parametric estimation for the symbolic mixture model decomposition.

6.3.2 The Symbolic Mixture Decomposition

To prepare for the symbolic data, which is the distributional data in our case, the original simulated single-valued dataset should be segmented into symbolic units. In this example, we use the cumulative distribution to represent over 50 original observations for each dimension. In the multivariate case, each observation is a vector of $p-$single-valued data points. The segmentation process will be applied to each dimension for the same segment of obser-

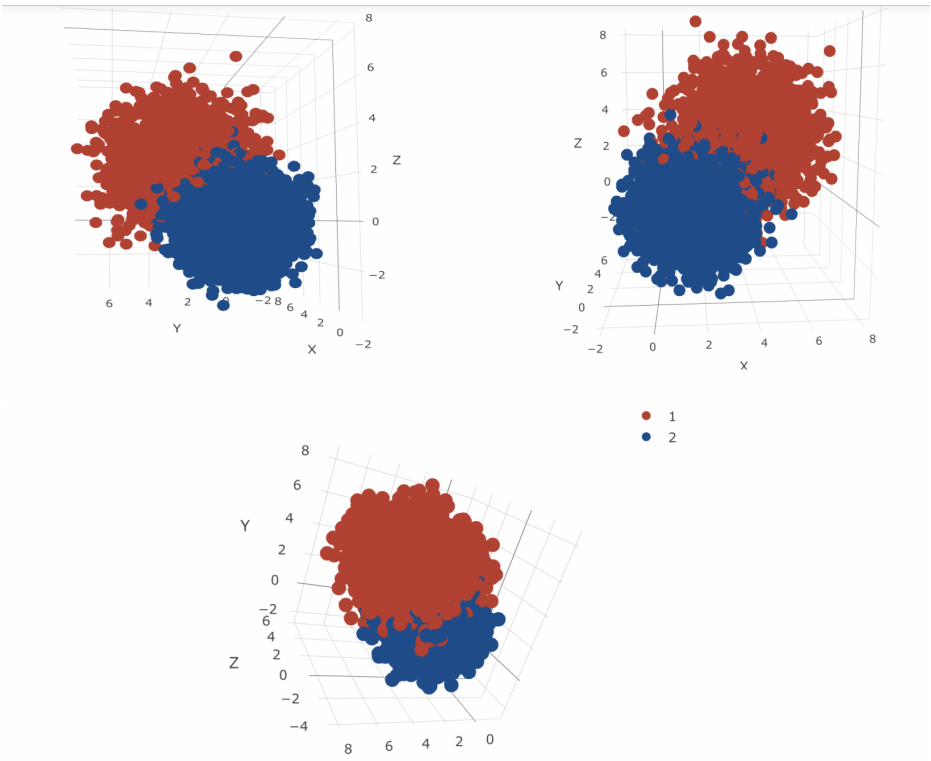

Figure 6.23: 3D Scatter Plot of Simulated Data from the direction of X_1 (the top left one) and X_2 (the top right one) and from the direction of X_3 (the bottom one) (Three Dimensional Case $p = 3$).

Table 6.30: Summary Table of Simulated Symbolic clusters.

Class	#units	#original observations (created from)
1	140	50
2	60	50

Table 6.31: Assignment of Starting Partition ($p = 3$).

Class	True Partition	Starting Partition
1	$((F_1^1, ..., F_1^p), ..., (F_{140}^1, ..., F_{140}^p))$	$P_1^0 = ((F_1^1, ..., F_1^p), ..., (F_{120}^1, ..., F_{120}^p))$
2	$((F_{141}^1, ..., F_{141}^p), ..., (F_{200}^1, ..., F_{200}^p))$	$P_2^0 = ((F_{121}^1, ..., F_{121}^p), ..., (F_{200}^1, ..., F_{200}^p))$

vations. In this example, in total we have 200 units as: $(F_i^1, ..., F_i^p), i = 1, ..., 100, p = 3$. Table 6.30 describes the details of creating the distributional units. The first 140 units $((F_1^1, ..., F_1^p), ..., (F_{140}^1, ..., F_{140}^p)), p = 3$, come from the first cluster, and the last 60 units $((F_{141}^1, ..., F_{141}^p), ..., (F_{200}^1, ..., F_{200}^p)), p = 3$, come from the second cluster. Similarly as before, we set up the starting partition with some wrong labels as:

$$P_1^0 = ((F_1^1, ..., F_1^p), ..., (F_{120}^1, ..., F_{120}^p)) \text{ and } P_2^0 = ((F_{121}^1, ..., F_{121}^p), ..., (F_{200}^1, ..., F_{200}^p)), p = 3.$$

This means that we assign about 20 units into a wrong cluster (as shown in Table 6.31). We hope to see that the algorithm could keep those correctly allocated in their right cluster and re-allocate the wrong ones into their right cluster.

Step 1: Define T values and obtain $(F_i^1(T_j^1), ..., F_i^p(T_j^p)), i = 1, ..., 200, j = 1, 2, p = 3$. Figure 6.24 - Figure 6.26 are the visualizations of the 200 distributional-data for the first dimension X_1, the second dimension X_2 and the third dimension X_3, respectively. The T values are defined as summarized in Table 6.32. Thus, we have that for the X_1 variable, $T_1^1 = 1.8, T_2^1 = 2.6$; for the X_2 variable, $T_1^2 = 1.0, T_2^2 = 2.0$; and for the X_3 vari-

able, $T_1^3 = 0.9, T_2^3 = 2.3$. After defining the T values, we can determine $(y_{ij}^1, ..., y_{ij}^p) = (F_i^1(T_j^1), ..., F_i^p(T_j^1)), i = 1, ..., 200, j = 1, 2, p = 3$. Then, the log-likelihood and AIC values are evaluated under a certain cluster for each dimension.

Consider the first dimension X_1. We define T values as $T^1 = (T_1^1, T_2^1) = (1.8, 2.6)$ (as indicated by the vertical dotted lines in Figure 6.24). After evaluating the 100 cumulative distribution functions at the given T values, we obtain $(y_{i1}^1, y_{i2}^1) = (F_i^1(T_1^1), F_i^1(T_2^1)), i = 1, ..., 200$, for the first dimension X_1. Based on (y_{i1}^1, y_{i2}^1) where $\omega_i \in P_k^0$, we calculate the log-likelihood and AIC values for each cluster with all candidate copula functions that we listed at the beginning of this section. Table 6.33 and Table 6.34 summarize the results for the first cluster $k = 1$ and the second cluster $k = 2$, respectively. From Table 6.33, we can see that the Clayton copula has the largest log-ikelihood value at $lnL = 14.77$ and the smallest value of AIC = -27.52 with the estimated parameter $\beta = 0.774$ for the first cluster of X_1. For the second cluster of X_1, we can see from Table 6.34 that the Joe copula has the largest log-likelihood value at $lnL = 47.48$ and the smallest value of AIC = -92.96 with the estimated parameter $\beta = 3.766$. Figure 6.27 shows the visualization of the density plot and the probability plot of the Clayton copula for the first dimension X_1 and the first cluster $k = 1$. Figure 6.28 shows the visualizations of the density plot and the probability plot of the Joe copula for the first dimension X_1 and the second cluster $k = 2$.

For the second dimension X_2, we define the T values as $T^2 = (T_1^2, T_2^2) = (1.0, 2.0)$ (as indicated by the vertical dotted lines in Figure 6.25). After evaluating the 100 cumulative distribution functions at the given T values, we obtain $(y_{i1}^2, y_{i2}^2) = (F_i^2(T_1^2), F_i^2(T_2^2)), i = 1, ..., 200$, for the second dimension X_2. Based on (y_{i1}^2, y_{i2}^2) where $\omega_i \in P_k^0$, we calculate the log-likelihood and AIC values for each cluster with all candidate copula functions. Table 6.35 and Table 6.36 summarize the results for the first cluster $k = 1$ and the second cluster

$k = 2$, respectively. From Table 6.35, we can see that the Clayton copula has the largest log-ikelihood value at $lnL = 16.31$ and the smallest value of AIC = -30.62 with the estimated parameter $\beta = 0.863$ for the first cluster of X_2. From Table 6.36, we can see that for the second cluster of X_2, the Gumbel copula has the largest log-likelihood value at $lnL = 41.02$ and the smallest value of AIC = -80.04 with the estimated parameter $\beta = 2.496$. Figure 6.29 shows the visualization of the density plot (right) and the probability plot (left) of the Clayton copula for the second dimension X_2 and the first cluster $k = 1$. Figure 6.30 shows the visualizations of the density plot (right) and the probability plot (left) of the Gumbel copula for the second dimension X_2 and the second cluster $k = 2$.

For the third dimension X_3, we define the T values as $T^3 = (T_1^3, T_2^3) = (0.9, 2.3)$ (as indicated by the vertical dotted lines in Figure 6.26). After evaluating the 100 cumulative distribution functions at the given T values, we obtain $(y_{i1}^3, y_{i2}^3) = (F_i^3(T_1^3), F_i^3(T_2^3)), i = 1, ..., 200$, for the third dimension X_3. Based on (y_{i1}^3, y_{i2}^3) where $\omega_i \in P_k^0$, we calculate the log-likelihood and AIC values for each cluster for all candidate copula functions that were listed at the beginning of this example. Table 6.37 and Table 6.38 summarize the results for the first cluster $k = 1$ and the second cluster $k = 2$, respectively. From Table 6.37, we can see that the Gaussian copula has the largest log-likelihood value at $lnL = 30.72$ and the smallest value of AIC = -59.44 with the estimated parameter $\beta = 0.661$ for the first cluster of X_3. From Table 6.38, for the second cluster of X_3, the Frank copula has the largest log-likelihood value at $lnL = 67.25$ and the smallest value of AIC = -132.50 with the estimated parameter $\beta = 13.72$. Figure 6.31 shows the visualization of the density plot and the probability plot of the Gaussian copula for the third dimension X_3 and the first cluster $k = 1$. Figure 6.32 shows the visualizations of the density plot and the probability plot of the Frank copula for the third dimension X_3 and the second cluster $k = 2$.

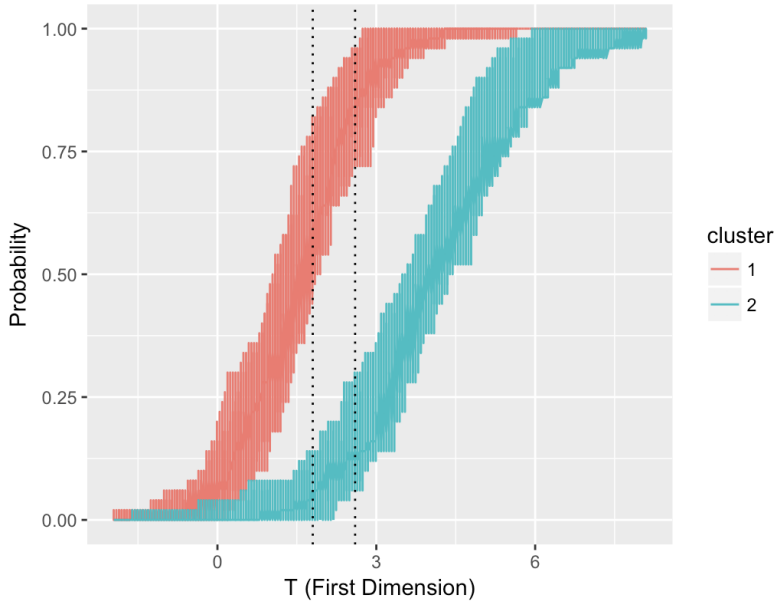

Figure 6.24: The Cumulative Distribution Functions $F_i^1, i = 1, ..., 200$, for the First Dimension X_1. (The vertical dotted lines represent $T_1^1 = 1.8$ and $T_2^1 = 2.6$)

Table 6.39 summarizes the final selected copula functions for all the three dimensions (X_1, X_2, X_3).

Step 2: Evaluate each unit under a certain cluster by the copula function selected from Step 1 and obtain the marginal probability values $(G^1(y^1_{i,T_1^1}, y^1_{i,T_2^1}), ..., G^p(y^p_{i,T_1^p}, y^p_{i,T_2^p})), i = 1, ..., 200, p = 3$, for all three variables (X_1, X_2, X_3). Then, select and estimate the copula function to model the multivariate joint relationship between the marginal distributions for each cluster. Take X_1 for example, $G^1(y^1_{i,T_1^1}, y^1_{i,T_2^1}), i = 1, ..., 120$, is obtained by using the R function **pCopula** with the Clayton copula with estimated paramter $\beta = 0.774$ based on $(y^1_{i1}, y^2_{i2}), i = 1, ..., 120$. For $i = 121, ..., 200$, $G^1(y^1_{i,T_1^1}, y^1_{i,T_2^1})$ is obtained by using the R function **pCopula** with the Joe copula with estimated paramter $\beta = 3.766$ based on $(y^1_{i1}, y^2_{i2}), i = 121, ..., 200$. Similarly, we can obtain $G^2(y^2_{i,T_1^2}, y^2_{i,T_2^2})$ and $G^3(y^3_{i,T_1^3}, y^3_{i,T_2^3})$.

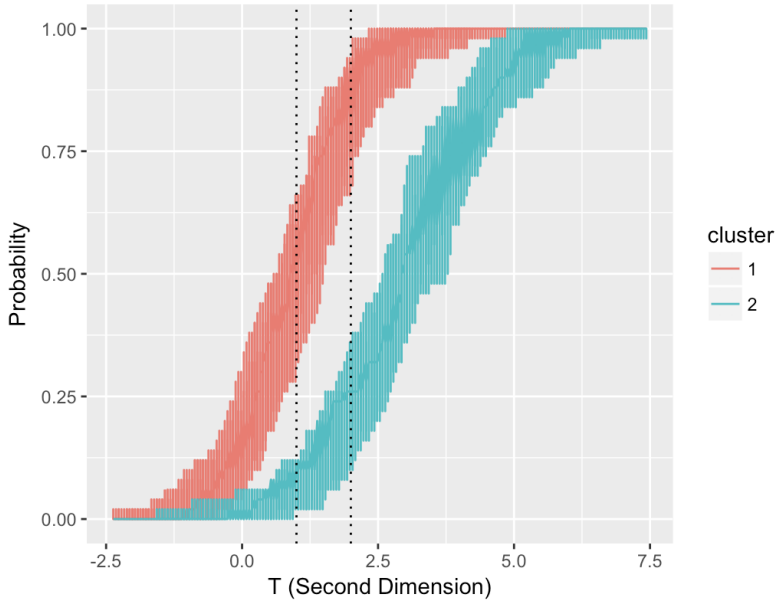

Figure 6.25: The Cumulative Distribution Functions $F_i^2, i = 1, ..., 200$, for the Second Dimension X_2. (The vertical dotted lines represent $T_1^2 = 1.0$ and $T_2^2 = 2.0$)

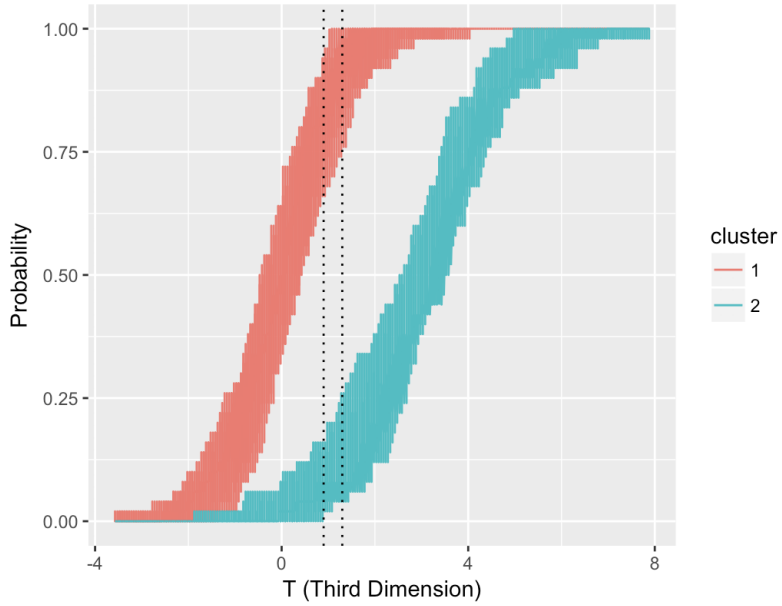

Figure 6.26: The Cumulative Distribution Functions $F_i^3, i = 1, ..., 200$, for the Third Dimension X_3. (The vertical dotted lines represent $T_1^3 = 0.9$ and $T_2^3 = 2.3$)

Table 6.32: Choices of T Values for Each Dimension.

Dimension	T_1	T_2
X_1	1.8	2.6
X_2	1.0	2.0
X_3	0.9	2.3

Table 6.33: Estimation of Candidate Copula Functions: First Iteration and First Dimension X_1 and First Cluster $k = 1$. (Fit based on 120 observations $(y^1_{i1}, y^1_{i2}), i = 1, ..., 120$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.451	11.83	-21.66
Gumbel	1.286	6.15	-10.31
Joe	1.264	2.24	-2.47
Frank	2.525	9.647	-17.29
Clayton	0.774	14.77	-27.52

Table 6.34: Estimation of Candidate Copula Functions: First Iteration and First DimensionX_1 and Second Cluster $k = 2$. (Fit based on 80 observations $(y^1_{i1}, y^1_{i2}), i = 121, ..., 200$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.783	34.52	-67.04
Gumbel	1.597	43.55	-85.10
Joe	3.766	47.48	-92.96
Frank	7.812	36.16	-70.32
Clayton	1/203	16.26	-30.52

Table 6.35: Estimation of Candidate Copula Functions: First Iteration and Second Dimension X_2 and First Cluster $k = 1$ (Fit based on 120 observations $(y^2_{i1}, y^2_{i2}), i = 1, ..., 120$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.506	15.54	-29.08
Gumbel	1.387	11.39	-20.78
Joe	31.443	7.28	-12.55
Frank	2.925	12.15	-22.30
Clayton	0.863	16.31	-30.62

Table 6.36: Estimation of Candidate Copula Functions: First Iteration and Second Dimension X_2 and Second Cluster $k = 2$. (Fit based on 80 observations $(y_{i1}^2, y_{i2}^2), i = 121, ..., 200$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.797	36.53	-71.06
Gumbel	2.496	41.02	-80.04
Joe	3.298	40.17	-78.34
Frank	8.312	38.77	-75.54
Clayton	1.361	19.70	-37.40

Table 6.37: Estimation of Candidate Copula Functions: First Iteration and Third Dimension X_3 and First Cluster $k = 1$. (Fit based on 120 observations $(y_{i1}^3, y_{i2}^3), i = 1, ..., 120$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.661	30.72	-59.44
Gumbel	1.802	29.61	-57.22
Joe	2.024	22.61	-43.22
Frank	4.987	28.33	-54.66
Clayton	1.374	30.02	-58.04

Table 6.38: Estimation of Candidate Copula Functions: First Iteration and Third Dimension X_3 and Second Cluster $k = 2$. (Fit based on 80 observations $(y_{i1}^3, y_{i2}^3), i = 121, ..., 200$)

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.899	60.97	-119.94
Gumbel	3.417	62.61	-123.22
Joe	4.684	58.66	-115.32
Frank	13.72	67.25	-132.50
Clayton	8.658	55.71	-109.42

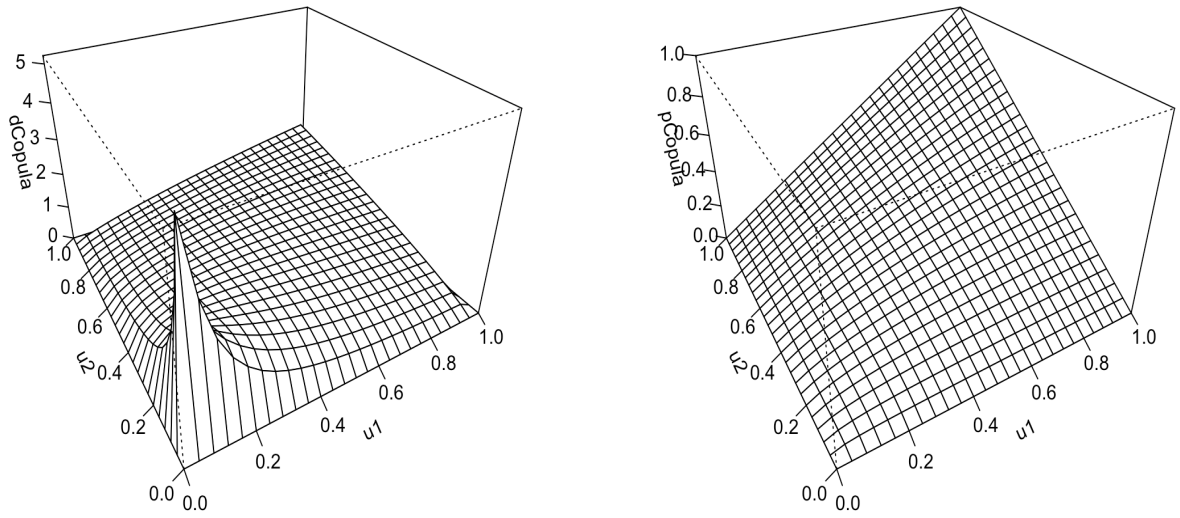

Figure 6.27: Visualization of final selected copula model (First Dimension X_1 and First Cluster $k = 1$): Density plot (left) and Copula plot (right) of the Clayton Copula with parameter 0.774.

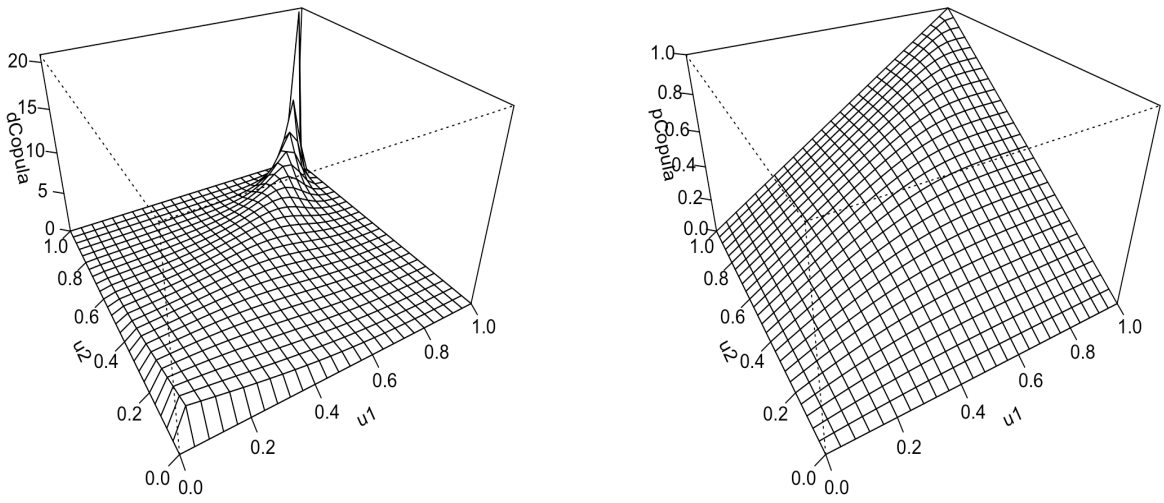

Figure 6.28: Visualization of final selected copula model (First Dimension X_1 and Second Cluster $k = 2$): Density plot (left) and Copula plot (right) of the Joe Copula with parameter 3.766.

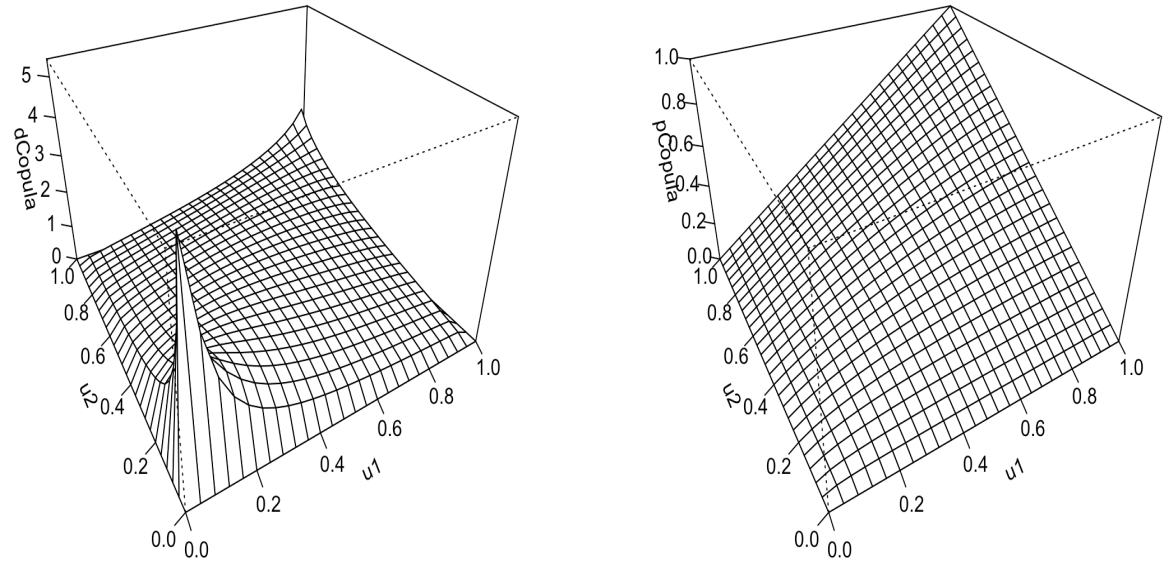

Figure 6.29: Visualization of final selected copula model (Second Dimension X_2 and First Cluster $k = 1$): Density plot (left) and Copula plot (right) of the Clayton Copula with parameter 0.863.

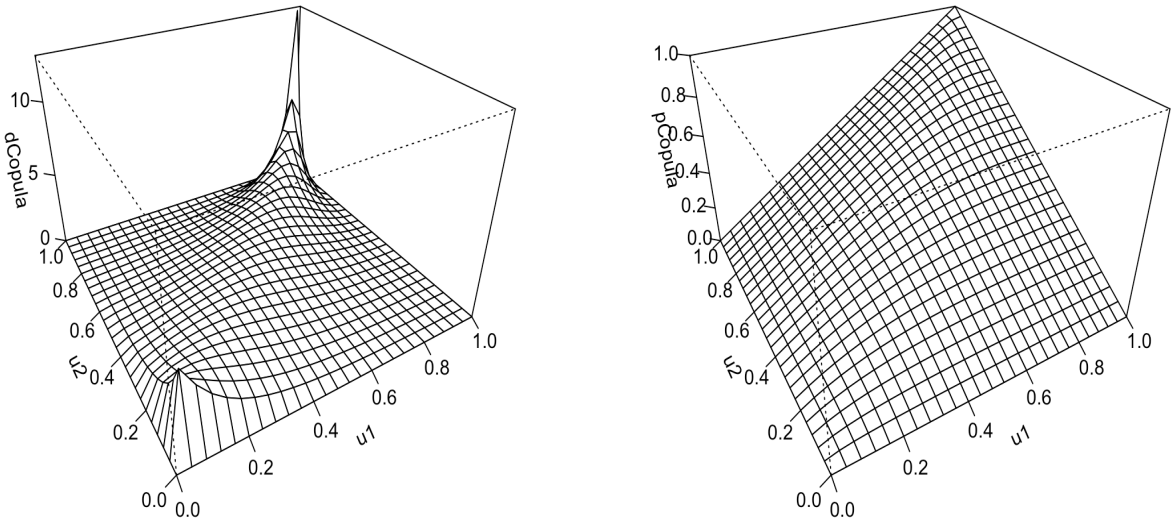

Figure 6.30: Visualization of final selected copula model (Second Dimension X_2 and Second Cluster $k = 2$): Density plot (left) and Copula plot (right) of the Gumbel Copula with parameter 2.496.

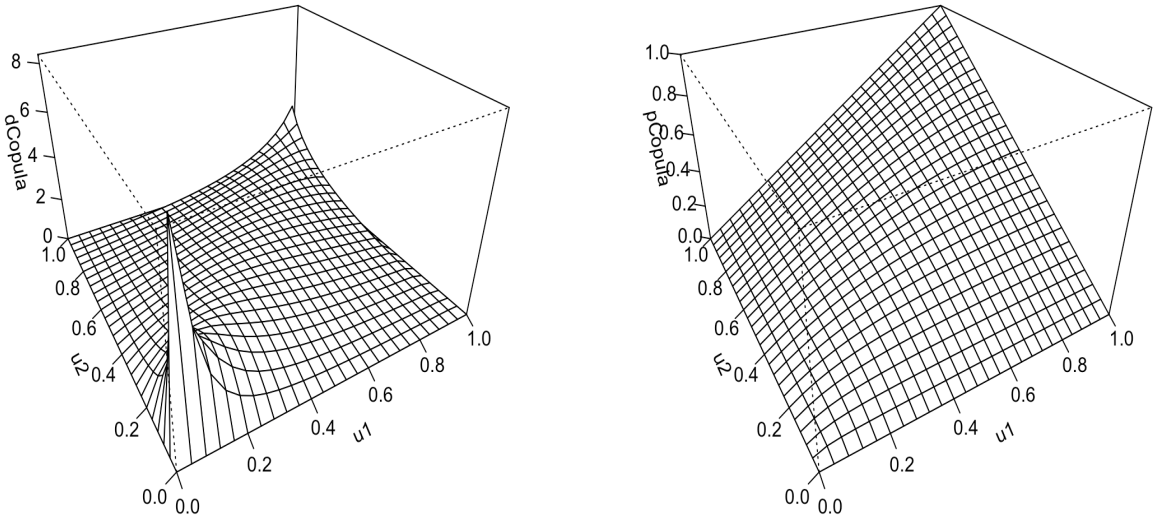

Figure 6.31: Visualization of final selected copula model (Third Dimension X_3 and First Cluster $k = 1$): Density plot (left) and Copula plot (right) of the Gaussian Copula with parameter 0.661.

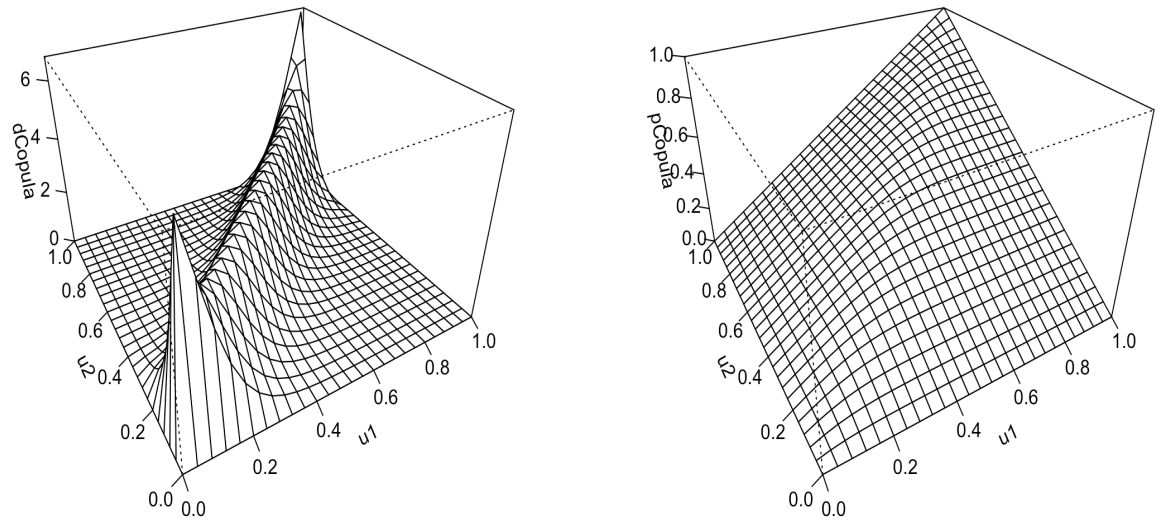

Figure 6.32: Visualization of final selected copula model (Third Dimension X_3 and Second Cluster $k = 2$): Density plot (left) and Copula plot (right) of the Frank Copula with parameter 13.720.

Table 6.39: Final Selected Copula Models and Estimated Parameters

Dimension and Cluster	Copula Function	Estimated Parameter β
X_1 and C_1	Clayton	0.774
X_1 and C_2	Joe	3.766
X_2 and C_1	Clayton	0.863
X_2 and C_2	Gumbel	2.496
X_3 and C_1	Gaussian	0.661
X_3 and C_2	Frank	13.720

Then, based on the three-dimensional data

$$(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), ..., G^p(y^p_{i,T^p_1}, y^p_{i,T^p_2})), i = 1, ..., 200, p = 3,$$

we calculate the log-likelihood values and AIC value for each cluster of all the candidate copula functions. Table 6.40 summarizes the log-likelihood and AIC values for candidate copula models in the 3-dimensional case with (X_1, X_2, X_3) for the first cluster $k = 1$. Table 6.41 summarizes the log-likelihood and AIC values for the candidate copula models in the 3-dimensional case with (X_1, X_2, X_3) for the second cluster $k = 2$.

In section 4.2.3, we introduced the four different dispersion matrices in the case of $p = 3$. Here, in this example, we choose to use the candidate copula models with one parameter β, which is the dispersion parameter of exchangeable dispersion structure as

$$\begin{pmatrix} 1 & \beta_1 & \beta_1 \\ \beta_1 & 1 & \beta_1 \\ \beta_1 & \beta_1 & 1 \end{pmatrix}.$$

Table 6.40: Estimation of Candidate Copula Function for Joint Cumulative Distribution: First Iteration and First cluster.
(Fit based on 120 observations $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), ..., G^p(y^p_{i,T^p_1}, y^p_{i,T^p_2})), i = 1, ..., 120, p = 3)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.090	1.198	-0.396
Gumbel	1.043	0.901	0.198
Joe	1.047	0.639	0.722
Frank	0.393	0.746	0.507
Clayton Copula	0.107	1.194	-0.388

Table 6.41: Estimation of Candidate Copula Function for Joint Cumulative Distribution: First Iteration and Second cluster.
(Fit based on 80 observations $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), ..., G^p(y^p_{i,T^p_1}, y^p_{i,T^p_2})), i = 121, ..., 200, p = 3)$

Copula Function	Estimated Parameter β	Log-likelihood	AIC
Gaussian	0.539	28.32	-54.64
Gumbel	1.539	34.67	-67.34
Joe	1.926	40.19	-78.38
Frank	3.267	28.38	-54.76
Clayton	0.486	9.503	-17.00

Table 6.40 shows that the Gaussian copula achieves the best performance with the largest log-likelihood at $lnL = 1.198$ and smallest AIC = -0.396 for the first cluster $k = 1$. For the second cluster $k = 2$, the best one is the Joe copula model with the largest log-likelihood at $lnL = 40.19$ and smallest AIC = -78.38 with estimated parameter $\beta = 1.926$.

Step 3: Fit the final selected copula models from Step 2 under a certain cluster with the marginal probability values $(G^1(y^1_{i,T^1_1}, y^1_{i,T^1_2}), ..., G^p(y^p_{i,T^p_1}, y^p_{i,T^p_2})), i = 1, ..., 200, p = 3$, and obtain the density values $h^k_{\omega_i}(X, \boldsymbol{\gamma}), k = 1, 2$, for all the units $\omega_i, i = 1, ..., 200$. The value of $h^k_{\omega_i}(X, \boldsymbol{\gamma}), i = 1, ..., 200, k = 1, 2$, is calculated by using the R function **dCopula** based on

values of $(G^1(y^1_{i,T_1^1}, y^1_{i,T_2^1}), ..., G^p(y^p_{i,T_1^p}, y^p_{i,T_2^p})), i = 1, ..., 200, p = 3$, with the Gaussian copula with $\beta = 0.09$ for the first cluster $k = 1$ and the Joe copula with $\beta = 40.19$ for the second cluster $k = 2$. Table 6.42 shows part of the density values evaluated under each cluster $k = 1, 2$ and the corresponding allocation result for the first iteration. All the units that come from the second cluster $(\omega_{141}, ..., \omega_{200})$ stay in the right place, except three units whose density values equal to 0. However, only 43 out of 140, which is about one third of the units are correctly predicted into the first cluster. Therefore, the total accuracy is 51.5%. This is low compared with the first iteration result in the previous examples. Let us see the performance of the second iteration.

Step 4: Conduct the second iteration by repeating Step 1 to Step 3 with the updated allocation labels from the first iteration. Table 6.43 shows the final prediction results with the comparison of the first iteration. We can see that most of the units that were incorrectly predicted as the second class have now been relocated to the first cluster. The total accuracy is improved to 77%. There are no more re-allocations for the next iteration. Therefore, we stop the process and have the final predicted label as shown in Table 6.43.

6.4 Conclusion and Discussion

We have illustrated different algorithms for a non-differentiable joint distribution function H and for a differentiable joint distribution function H in the one-dimensional case X_1, the two-dimensional case (X_1, X_2), and the multi-dimensional case $(X_1, ..., X_p)$ with different simulation data.

In the one-dimensional case X_1, when the number of symbolic units is small (e.g., N=5 and N=15), the application of the non-differentiable joint distribution function H estimated

Table 6.42: Allocation of each unit to the best fit class (Simulation Example 6.3 First Iteration).

Unit	$h_1(X,\gamma)$	$h_2(X,\gamma)$	Class membership
ω_1	0.998	1.208	P_2
ω_2	0.996	0.932	P_1
ω_3	0.989	0.910	P_1
ω_4	0.998	1.168	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{12}	0.999	1.185	P_2
ω_{13}	1.004	1.202	P_2
ω_{14}	1.006	0.778	P_1
ω_{15}	1.003	1.128	P_2
ω_{16}	1.025	1.266	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{120}	1.009	0.434	P_1
ω_{121}	1.037	1.446	P_2
ω_{122}	1.047	1.454	P_2
ω_{123}	1.057	1.804	P_2
ω_{124}	1.033	1.056	P_2
ω_{125}	1.018	1.348	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{138}	1.009	0.951	P_1
ω_{139}	1.037	1.339	P_2
ω_{140}	1.052	1.295	P_2
ω_{141}	2.213	3.269	P_2
ω_{142}	1.742	3.099	P_2
ω_{143}	2.036	3.249	P_2
ω_{144}	2.309	3.384	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{198}	2.159	3.279	P_2
ω_{199}	1.846	3.148	P_2
ω_{200}	2.106	3.304	P_2

Table 6.43: Allocation of each unit to the best fit class (Simulation Example 6.3 Second Iteration).

Unit	$h_1(X,\gamma)$	$h_2(X,\gamma)$	Class membership in Second Iteration	Class in First Iteration
ω_1	1.568	1.275	P_1	P_2
ω_2	1.664	1.314	P_1	P_1
ω_3	1.414	1.239	P_1	P_1
ω_4	1.804	1.503	P_1	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{12}	1.081	1.246	P_2	P_2
ω_{13}	1.269	1.374	P_2	P_2
ω_{14}	2.561	1.587	P_1	P_1
ω_{15}	2.265	1.587	P_1	P_2
ω_{16}	1.826	1.373	P_1	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{120}	11.091	5.746	P_1	P_1
ω_{121}	10.248	4.028	P_2	P_2
ω_{122}	14.166	8.854	P_2	P_2
ω_{123}	15.575	10.381	P_2	P_2
ω_{124}	13.471	8.081	P_2	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{138}	16.979	12.868	P_1	P_1
ω_{139}	16.493	12.821	P_2	P_2
ω_{140}	13.566	8.464	P_2	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
ω_{196}	0.664	1.117	P_2	P_2
ω_{197}	0.887	1.039	P_2	P_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

by the empirical distribution has excellent results of 100% accuracy. However, when the joint distribution function H is non-differentiable, the calculation of the approximation of the joint density function is complicated and time-consuming. We recommend using a parametric copula function with a parametric marginal distribution when the number of symbolic units is large (e.g., N=150). For the differentiable joint distribution function H, we have considered a total of six parametric copulas (the Frank copula given by equation 3.5, the Clayton copula given by 3.6, the Joe copula given by equation 3.7, the Gumbel copula given by 3.8, the Gaussian copula given by equation 3.9 and the t-copula given by equation 3.10). Based on our simulation examples, these six candidate copula models are good enough to represent the simulated data with at least 77% clustering accuracy. Therefore, in practice, for any kind of data set, we recommend first fitting the model with these six copula models. If none of them can have a good result of clustering accuracy, other parametric copula families could be used with our algorithms (e.g., Survival Clayton, Survival Gumbel, Survival Joe and so on).

According to all the simulation examples, we learn that the choices of the dimension n of T and the exact T values are very important. We chose to use $n = 2$ for all the simulation examples, which is efficiently adequate to represent distinct values of the distributional-data (cumulative values of $F(T)$). Therefore, it can be used for general practical cases to have a clear separation of the clusters. We also let the two different T values describe different areas of the units to cover as much as possible properties of each unit (as shown in Figure 6.2). When T is very small or very large, we may have $F(T) = 0$ or $F(T) = 1$. This is the reason that we have the final estimated density values of 0 in the simulation example of Section 6.2. However, some small values of T represent more clear separations of the distributional-data. Therefore, this is a trade-off when selecting the values of T. In practice, like many other machine learning algorithms, we can first select several T values or define a

range of possible T values and see which T values have the best performance of clustering accuracy. This is a way of letting the data themselves to tell the users which choice is the best.

For all the simulation examples, we first need to segment the original classical data into distributional-data and then apply the model-based clustering algorithms. We have aggregated every 100 original classical data into one unit of distributional-data in most of the simulation examples, except the case in Section 6.3 where we use each unit aggregated by 50 classical observations. The reason that we do not give an exact criterion of how to aggregate the original data is that the most important advantage of our algorithm is to solve the clustering problems for grouped data that the classical clustering algorithm cannot make. This means that for so many practical problems, the original data set has its own segmentation according to the underlying research interest. Therefore, in such cases, we just need to use its own segmentation. In some other cases, our algorithms can also be applied to an extremely large dataset to reduce the sample size to a more manageable size. In this situation, we can aggregate the original data according to the requirement of a manageable sample size and for a more meaningful scientific goal.

We all know that in classical regression problems, the mixed effect model is particularly useful for repeated measurements in the same statistical unit. Our algorithm can be applied by representing the repeated measurements using a cumulative distribution for each statistical unit. Then, the model-based clustering algorithm can be used to group the distributional-data into clusters.

Chapter 7

Future Work

As we discussed in Section 6.4, the choices of the dimension n of T and the exact T values are very important. Although $n = 2$ is efficient to obtain good clustering results based on the simulation examples, it is worth to further study the relationship between the dimension n and the final clustering performance (accuracy). Vrac (2002) and Diday and Vrac (2005) have proposed a triangle method which is helpful to select the values of T. However, a general criterion of the best choices of T is also a question for further development.

On the other hand, more diverse data structures can be considered with our algorithms. For example, since our algorithm has an advantage of representing the inner variation of each observation, it is interesting to see if it can separate two groups of data with the same centroid into different clusters. What is more, if one of the clusters is far away from the others and the initial assumption mixes them together, can our algorithm finally re-allocate them into the right cluster?

In a different direction, as we state in Section 6.4., our algorithm can be applied to some of the repeated measurements clustering problems. It is because we can represent the re-

peated measurements with a cumulative distribution for each statistical unit. However, in the case of a longitudinal study (panel study), how to appropriately describe each unit over time also needs to be undertaken.

Finally, making all the algorithms into a well-defined R package is also a remaining work.

Chapter 8

Appendix

R code:

Algorithms for model-based clustering of distributional-data

```
usePackage <- function(p)
{
        if (!is.element(p, installed.packages()[,1]))
        install.packages(p, dep = TRUE)
        require(p, character.only = TRUE)
}
usePackage("ggplot2")
usePackage("reshape2")
usePackage("MASS")
usePackage("copula")
usePackage("rlist")
usePackage("plotly")
usePackage("VineCopula")
usePackage("mixtools")
```

```
# @mydata: is a vector for one variable;
#    is a matrix for multi-variables
# @c: number of original observation in each unit;
#        it can be a number or a vector
# @nk: the vector of number of observations in each cluster
# @K: number of clusters
# @mydata: a vector
# @P0: a vector of starting partition, every two numbers
#are the starting number of the unit and the ending
#  number of the unit in cluster
# @Tvalues: a 2*P matrix


####scatter plot function
scatter_plot1D=function(mydata,K,nk){
        #one variable
classTrue=as.factor(rep(c(1:K),nk[1:K]))
N=length(mydata)
mydata_class=data.frame(simulated=mydata,cluster=classTrue)
ggplot(mydata_class, aes(x=1:n,y=simulated,
           color=cluster))
           +geom_point()+labs(x="Index")
}

scatter_plot2D=function(mydata,K,nk){
```

```
#two variable
classTrue=as.factor(rep(c(1:K),nk[1:K]))
N=dim(mydata)[1]
mydata_class=data.frame(simulated.x=mydata[,1]
,simulated.y=mydata[,2], cluster=classTrue)
ggplot(mydata_class, aes(x=simulated.x,
y=simulated.y, color=cluster))
+geom_point(size=1.5, shape=4)
}

scatter_plot3D=function(mydata,K,nk){
        #three variable
        classTrue=as.factor(rep(c(1:K),nk[1:K]))
        N=dim(mydata)[1]
mydata_class=data.frame(simulated.x=mydata[,1],
simulated.y=mydata[,2],
simulated.z=mydata[,3],cluster=classTrue)
        ###3D scatter plor
plot_ly(mydata_class, x = ~simulated.x, y = ~simulated.y,
z = ~simulated.z, color = ~cluster,
symbols=c(4,4),sizes = 0.3,
colors = c('#BF382A', '#0C4B8E')) %>%
        add_markers() %>%
        layout(scene = list(xaxis = list(title = 'X'),
        yaxis = list(title = 'Y'),
        zaxis = list(title = 'Z')))
```

```
}

##function of getting cumulative distributions
#(distributional data)
cdf_list=function(mydata,c){##mydata is a vector
        cdf=list()
        N=length(mydata)
        for(i in 1:(N/c)){
                currentData=mydata[((i-1)*c+1):(i*c)]
                cdf.cur=ecdf(currentData)
                cdf=c(cdf,cdf.cur)
        }
        return (cdf)
}

#####function of creating plot of  distributional data

cdf_plot1D=function(mydata,c,K,nk){
        cdf_list=function(mydata,c){##mydata is a vector
                cdf=list()
                N=length(mydata)
                for(i in 1:(N/c)){
                        currentData=mydata[((i-1)*c+1):(i*c)]
                        cdf.cur=ecdf(currentData)
                        cdf=c(cdf,cdf.cur)
                }
```

```
                return (cdf)
        }
        N=lenght(mydata)
        m=N/c
        myFs=cdf_list(mydata,c)

        R=range(mydata)
        T=seq(min(R),max(R),0.1)
        classTrue=as.factor(rep(c(1:K),nk[1:K]))
mydata_class=data.frame(simulated=mydata, cluster=classTrue)

        cdf_values=NULL
        for(i in 1:m){
                cdf_values=cbind(cdf_values,myFs[[i]](T))
        }
        cdf_data=cbind(T, cdf_values)
        cdf_data=as.data.frame(cdf_data)
        data_long <- melt(cdf_data, id="T")
data_long$cluster=
as.factor(rep(c(1:K),(nk/c)*dim(cdf_values)[1]))

        ggplot(data=data_long,
        aes(x=T, y=value, colour=cluster))
}

cdf_plot.T1D=function(mydata,c,K,nk,Tvalues){
```

```
#add vertical dotted lines of T values
# @Tvalues: is a vector of T values
cdf_list=function(mydata,c){##mydata is a vector
        cdf=list()
        N=length(mydata)
        for(i in 1:(N/c)){
                currentData=mydata[((i-1)*c+1):(i*c)]
                cdf.cur=ecdf(currentData)
                cdf=c(cdf,cdf.cur)
        }
        return (cdf)
}

N=lenght(mydata)
m=N/c
myFs=cdf_list(mydata,c)

R=range(mydata)
T=seq(min(R),max(R),0.1)
classTrue=as.factor(rep(c(1:K),nk[1:K]))
mydata_class=data.frame(simulated=mydata,cluster=classTrue)

cdf_values=NULL
for(i in 1:m){
        cdf_values=cbind(cdf_values,myFs[[i]](T))
}
```

```
        cdf_data=cbind(T,cdf_values)
        cdf_data=as.data.frame(cdf_data)
        data_long <- melt(cdf_data, id="T")
data_long$cluster=
       as.factor(rep(c(1:K),(nk/c)*dim(cdf_values)[1]))

        ggplot(data=data_long,
        aes(x=T, y=value, colour=cluster)) +
    geom_line()+ geom_vline(xintercept=Tvalues[1],color=1,lty=3)
       +geom_vline(xintercept=Tvalues[2],color=1,lty=3)

}

cdf_plot.multiD=function(mydata,c,K,nk){
        cdf_list=function(mydata,c){##mydata is a vector
                cdf=list()
                N=length(mydata)
                for(i in 1:(N/c)){
                        currentData=mydata[((i-1)*c+1):(i*c)]
                        cdf.cur=ecdf(currentData)
                        cdf=c(cdf,cdf.cur)
                }
                return (cdf)
        }
        #creat cdf plots for each variable
        N=dim(mydata)[1]
```

```
        P=dim(mydata)[2]
        m=N/c

for (p in 1:P){
        myFs=cdf_list(mydata[,p],c)
        R=range(mydata[,p])
        T=seq(min(R),max(R),0.1)
        classTrue=as.factor(rep(c(1:K),nk[1:K]))
mydata_class=data.frame(simulated=mydata[,p],
    cluster=classTrue)

        cdf_values=NULL
for(i in 1:m){
        cdf_values=cbind(cdf_values,myFs[[i]](T))}
cdf_data=cbind(T,cdf_values)
cdf_data=as.data.frame(cdf_data)
data_long <- melt(cdf_data, id="T")
data_long$cluster=as.factor(rep(c(1:K),(nk/c)*dim(cdf_values)[1]))

ggplot(data=data_long,
aes(x=T, y=value, colour=cluster))
+geom_line()+labs(x=paste("T(varaible",p,")"), y = "Probability")
        }
}

cdf_plot.TmultiD=function(mydata,c,K,nk,Tvalues){
```

```
        cdf_list=function(mydata,c){##mydata is a vector
                cdf=list()
                N=length(mydata)
                for(i in 1:(N/c)){
                        currentData=mydata[((i-1)*c+1):(i*c)]
                        cdf.cur=ecdf(currentData)
                        cdf=c(cdf,cdf.cur)
                }
                return (cdf)
        }
#Tvalues is a 2*P matrix
#(pth colunm is the T values for the pth variable)
#creat cdf plots for each variable
        N=dim(mydata)[1]
        P=dim(mydata)[2]
        m=N/c

for (p in 1:P){
myFs=cdf_list(mydata[,p],c)
R=range(mydata[,p])
T=seq(min(R),max(R),0.1)
classTrue=as.factor(rep(c(1:K),nk[1:K]))
mydata_class=data.frame(simulated=mydata[,p],cluster=classTrue)

cdf_values=NULL
for(i in 1:m){
```

```
cdf_values=cbind(cdf_values,myFs[[i]](T))}
cdf_data=cbind(T,cdf_values)
cdf_data=as.data.frame(cdf_data)
data_long <- melt(cdf_data, id="T")
data_long$cluster=as.factor(rep(c(1:K),
                (nk/c)*dim(cdf_values)[1]))

ggplot(data=data_long,
                  aes(x=T, y=value, colour=cluster))
                  +geom_line()+labs(x=paste("T(varaible",p,")"),
                  y = "Probability")
                  +geom_line()
+ geom_vline(xintercept=Tvalues[1,p],color=1,lty=3)
+geom_vline(xintercept=Tvalues[2,p],color=1,lty=3)
        }
}


##################################################
#############Algorithms when H is differentiable
######function of select the best copula model
multiCopulaSelect=function(my_dim,pobs.matrix){
test_target = c(quote(normalCopula(dim = my_dim)),
quote(claytonCopula(dim = my_dim)),
quote(gumbelCopula(dim = my_dim)),
quote(frankCopula(dim = my_dim)),
```

```
quote(joeCopula(dim = my_dim)))
# browser()
result = lapply(test_target, function(test_target,
                        data = pobs.matrix ){
result = summary(fitCopula(eval(test_target),
                        data = pobs.matrix))
return(c(loglik = result$loglik, result$coefficients[1]))
})
# browser()
result = as.data.frame(do.call(rbind, result))
#cat(paste0(test_target[idx], ' with Log likelihood ',
result$loglik[idx],' and parameter ',result$V2[idx]))
copula_models=list(normalCopula(par=result[1,2],dim=my_dim),
claytonCopula(par = result[2,2],dim=my_dim),
gumbelCopula(par = result[3,2],dim=my_dim),
frankCopula(par = result[4,2],dim=my_dim),
joeCopula(par = result[5,2],dim=my_dim)
)
idx = which.max(result$loglik)
# browser()
finalmodel=copula_models[[idx]]
return(list(estimate=result,modelselect=finalmodel))
}


##model-based clustering for ditributional data with one variable
```

```
mbc.sym1D.para=function(mydata,c,P0,K,Tvalues){

# @c: number of original observation
#       in each unit(distributional data);
#          it can be a number or a vector
# @nk: the vector of number of observations in each cluster
# @K: number of clusters
# @mydata: a vector
# @P0: a vector of starting partition, every two numbers are the
# starting number of the original observation and the ending
# number of the orginal obervation

cdf_list=function(mydata,c){##mydata is a vector
cdf=list()
N=length(mydata)
for(i in 1:(N/c)){
currentData=mydata[((i-1)*c+1):(i*c)]
cdf.cur=ecdf(currentData)
cdf=c(cdf,cdf.cur)
}
return (cdf)
}
multiCopulaSelect=function(my_dim,pobs.matrix){
test_target = c(quote(normalCopula(dim = my_dim)),
quote(claytonCopula(dim = my_dim)),
quote(gumbelCopula(dim = my_dim)),
```

```
quote(frankCopula(dim = my_dim)),
quote(joeCopula(dim = my_dim)))
# browser()
result = lapply(test_target, function(test_target,
                        data = pobs.matrix ){
result = summary(fitCopula(eval(test_target),
              data = pobs.matrix))
return(c(loglik = result$loglik, result$coefficients[1]))
})
# browser()
result = as.data.frame(do.call(rbind, result))
#cat(paste0(test_target[idx], ' with Log likelihood ',
#    result$loglik[idx],'
#and parameter ',result$V2[idx]))
copula_models=list(normalCopula(par=result[1,2],dim=my_dim),
claytonCopula(par = result[2,2],dim=my_dim),
gumbelCopula(par = result[3,2],dim=my_dim),
frankCopula(par = result[4,2],dim=my_dim),
joeCopula(par = result[5,2],dim=my_dim)
)
idx = which.max(result$loglik)
# browser()
finalmodel=copula_models[[idx]]
return(list(estimate=result,modelselect=finalmodel))
}
```

```
N=lenght(mydata)
m=N/c
myFs=cdf_list(mydata,c)

y_t1=NULL
for(i in 1:m){
yi=myFs[[i]](Tvalues[1])
y_t1=c(y_t1,yi)
}

y_t2=NULL
for(i in 1:m){
yi=myFs[[i]](Tvalues[2])
y_t2=c(y_t2,yi)
}
y=cbind(y_t1,y_t2)

ite=1
yt1_list=list()
yt2_list=list()
pob_list=list()
result_list=list()
finalmodel_list=list()
h_matrix=NULL
h_list=list()
accuracy=NULL
```

```
for (k in 1:K){
j=2*k-1
partition=P0[j]:P0[j+1]
for (i in partition){
yt1=y_t1[partition]
yt2=y_t2[partition]
pob=pobs(as.matrix(cbind(yt1,yt2)))
}
model.estimate=multiCopulaSelect(2,pob)
finalmodel_list=list.append(finalmodel_list,model.estimate[[2]])
result_list=list.append(result_list,model.estimate[[1]])



yt1_list= list.append(yt1_list,yt1)
yt2_list= list.append(yt2_list,yt2)
pob_list=list.append(pob_list,pob)
h=dCopula(copula=model.estimate[[2]],
     as.matrix(cbind(y_t1,y_t2)))
h_matrix=cbind(h_matrix,h)
h_matrix=cbind(h_matrix,apply(h_matrix,1,which.max),
                    rep(c(1:K),nk[1:K]/c))
h_list=list.append(h_list,h_matrix)
cluster=rep(c(1:K),nk[1:K]/c)
accuracy=c(accuracy,sum(diag(table(h_matrix[,K+1],cluster)))/m)
```

```
###
#iteration>1
while (h_list[[ite]]!=h_list[[ite-1]]){
ite=ite+1
yt1_list=list()
yt2_list=list()
pob_list=list()
result_list=list()
finalmodel_list=list()
for (k in 1:K){
j=2*k-1
partition=which(h_list[[ite-1]][,K+1]==k)
for (i in partition){
yt1=y_t1[partition]
yt2=y_t2[partition]
pob=pobs(as.matrix(cbind(yt1,yt2)))
}
model.estimate=multiCopulaSelect(2,pob)
finalmodel_list=list.append(finalmodel_list,model.estimate[[2]])
result_list=list.append(result_list,model.estimate[[1]])



yt1_list= list.append(yt1_list,yt1)
yt2_list= list.append(yt2_list,yt2)
pob_list=list.append(pob_list,pob)
h=dCopula(copula=model.estimate[[2]], as.matrix(cbind(y_t1,y_t2)))
```

```
h_matrix=cbind(h_matrix,h)
}
h_matrix=cbind(h_matrix,apply(h_matrix,1,which.max),
                                rep(c(1:K),nk[1:K]/c))
h_list=list.append(h_list,h_matrix)
cluster=rep(c(1:K),nk[1:K]/c)
accuracy=c(accuracy,sum(diag(table(h_matrix[,K+1],cluster)))/m)
}
}
}


mbc.sym.multiD.para=function(mydata,c,P0,K,Tvalues){
cdf_list=function(mydata,c){##mydata is a vector
cdf=list()
N=length(mydata)
for(i in 1:(N/c)){
currentData=mydata[((i-1)*c+1):(i*c)]
cdf.cur=ecdf(currentData)
cdf=c(cdf,cdf.cur)
}
return (cdf)
}

multiCopulaSelect=function(my_dim,pobs.matrix){
test_target = c(quote(normalCopula(dim = my_dim)),
```

```
quote(claytonCopula(dim = my_dim)),
quote(gumbelCopula(dim = my_dim)),
quote(frankCopula(dim = my_dim)),
quote(joeCopula(dim = my_dim)))
# browser()
result = lapply(test_target, function(test_target,
                        data = pobs.matrix ){
result = summary(fitCopula(eval(test_target),
                                data = pobs.matrix))
return(c(loglik = result$loglik, result$coefficients[1]))
})
# browser()
result = as.data.frame(do.call(rbind, result))
#cat(paste0(test_target[idx], ' with Log likelihood ',
#result$loglik[idx],'
#and parameter ',result$V2[idx]))
copula_models=list(normalCopula(par=result[1,2],dim=my_dim),
claytonCopula(par = result[2,2],dim=my_dim),
gumbelCopula(par = result[3,2],dim=my_dim),
frankCopula(par = result[4,2],dim=my_dim),
joeCopula(par = result[5,2],dim=my_dim)
)
idx = which.max(result$loglik)
# browser()
finalmodel=copula_models[[idx]]
return(list(estimate=result,modelselect=finalmodel))
```

```
}
# @c: number of original observation in
#each unit(distributional data);
#it can be a number or a vector
# @nk: the vector of number of observations in each cluster
# @K: number of clusters
# @mydata: a vector
# @P0: a vector of starting partition, every two numbers
# are the starting number of the original observation
# and the ending number of the orginal obervation
# @Tvalues: a 2*P matrix
N=dim(mydata)[1]
P=dim(mydata)[2]
m=N/c
#first estimate the copula model for each variable
p_matrix=NULL #the result of pCopula for each variable
#  p_list=list()
#fisrt obtain the p_matrix with each colunm
 # is the probability of selected
#copula models for each variable
ite=1
for(p in 1:P){
myFs=cdf_list(mydata[,p],c)

y_t1=NULL
for(i in 1:m){
```

```
yi=myFs[[i]](Tvalues[1,p])
y_t1=c(y_t1,yi)
}

y_t2=NULL
for(i in 1:m){
yi=myFs[[i]](Tvalues[2,p])
y_t2=c(y_t2,yi)
}
y=cbind(y_t1,y_t2)

yt1_list=list()
yt2_list=list()
pob_list=list()
result_list=list()
finalmodel_list=list()
p=NULL
for (k in 1:K){
j=2*k-1
partition=P0[j]:P0[j+1]
for (i in partition){
yt1=y_t1[partition]
yt2=y_t2[partition]
pob=pobs(as.matrix(cbind(yt1,yt2)))
}
model.estimate=multiCopulaSelect(2,pob)
```

```
finalmodel_list=list.append(finalmodel_list,model.estimate[[2]])
result_list=list.append(result_list,model.estimate[[1]])



yt1_list= list.append(yt1_list,yt1)
yt2_list= list.append(yt2_list,yt2)
pob_list=list.append(pob_list,pob)
p_k=pCopula(copula=model.estimate[[2]],
        as.matrix(cbind(yt1,yt2)))
p=c(p,p_k)
}
p_matrix=cbind(p_matrix,p)
}
#Then estimate copula model based on p_matrix
#  yt1_list=list()
# yt2_list=list()
pob_list=list()
result_list=list()
finalmodel_list=list()
h_matrix=NULL
h_list=list()
accuracy=NULL


for (k in 1:K){
j=2*k-1
partition=P0[j]:P0[j+1]
```

```
for (i in partition){
pob=pobs(p_matrix[partition,])
}
model.estimate=multiCopulaSelect(P,pob)
finalmodel_list=list.append(finalmodel_list,
                            model.estimate[[2]])
result_list=list.append(result_list,model.estimate[[1]])



#     yt1_list= list.append(yt1_list,yt1)
#     yt2_list= list.append(yt2_list,yt2)
pob_list=list.append(pob_list,pob)
h=dCopula(copula=model.estimate[[2]],p_matrix)
h_matrix=cbind(h_matrix,h)
h_matrix=cbind(h_matrix,apply(h_matrix,1,which.max),
                  rep(c(1:K),nk[1:K]/c))
h_list=list.append(h_list,h_matrix)
cluster=rep(c(1:K),nk[1:K]/c)
accuracy=c(accuracy,sum(diag(table(h_matrix[,K+1],cluster)))/m)}

while (h_list[[ite]]!=h_list[[ite-1]]){
ite=ite+1
for(p in 1:P){
myFs=cdf_list(mydata[,p],c)

y_t1=NULL
```

```
for(i in 1:m){
yi=myFs[[i]](Tvalues[1,p])
y_t1=c(y_t1,yi)
}

y_t2=NULL
for(i in 1:m){
yi=myFs[[i]](Tvalues[2,p])
y_t2=c(y_t2,yi)
}
y=cbind(y_t1,y_t2)

yt1_list=list()
yt2_list=list()
pob_list=list()
result_list=list()
finalmodel_list=list()
p=NULL
for (k in 1:K){
j=2*k-1
partition=which(h_list[[ite-1]][,K+1]==k)
for (i in partition){
yt1=y_t1[partition]
yt2=y_t2[partition]
pob=pobs(as.matrix(cbind(yt1,yt2)))
}
```

```
model.estimate=multiCopulaSelect(2,pob)
finalmodel_list=list.append(finalmodel_list,model.estimate[[2]])
result_list=list.append(result_list,model.estimate[[1]])



yt1_list= list.append(yt1_list,yt1)
yt2_list= list.append(yt2_list,yt2)
pob_list=list.append(pob_list,pob)
p_k=pCopula(copula=model.estimate[[2]],
        as.matrix(cbind(yt1,yt2)))
p=c(p,p_k)
}
p_matrix=cbind(p_matrix,p)
}
#Then estimate copula model based on p_matrix
#  yt1_list=list()
# yt2_list=list()
pob_list=list()
result_list=list()
finalmodel_list=list()
h_matrix=NULL
h_list=list()
accuracy=NULL

for (k in 1:K){
j=2*k-1
```

```
partition=which(h_list[[ite-1]][,K+1]==k)
for (i in partition){
pob=pobs(p_matrix[partition,])
}
model.estimate=multiCopulaSelect(P,pob)
finalmodel_list=list.append(finalmodel_list,model.estimate[[2]])
result_list=list.append(result_list,model.estimate[[1]])


#     yt1_list= list.append(yt1_list,yt1)
#     yt2_list= list.append(yt2_list,yt2)
pob_list=list.append(pob_list,pob)
h=dCopula(copula=model.estimate[[2]],p_matrix)
h_matrix=cbind(h_matrix,h)
h_matrix=cbind(h_matrix,apply(h_matrix,1,which.max),
                  rep(c(1:K),nk[1:K]/c))
h_list=list.append(h_list,h_matrix)
cluster=rep(c(1:K),nk[1:K]/c)
accuracy=c(accuracy,sum(diag(table(h_matrix[,K+1],cluster)))/m)
}}}


##################################################
############Algorithms when H is non-differentiable

mbc.sym.emp =function(Tvalues,K,m_vector,mydata,c){
```

```
cdf_list=function(mydata,c){##mydata is a vector
cdf=list()
N=length(mydata)
for(i in 1:(N/c)){
currentData=mydata[((i-1)*c+1):(i*c)]
cdf.cur=ecdf(currentData)
cdf=c(cdf,cdf.cur)
}
return (cdf)
}
myFs=cdf_list(mydata,c)
#@Tvalues is a vector of T values
#@m_vector is a vector of every two numbers are the
#starting number of the unit and the ending number
#of the unit each cluster
n=length(Tvalues)

##obtain y matrix, each columm is F(T)
y_matrix=NULL
for (j in 1:n){
y_t=NULL
for(i in 1:m){
yi=myFs[[i]](Tvalues[j])
y_t=c(y_t,yi)
}
y_matrix=cbind(y_matrix,y_t)
```

```
}

#Step 1: Assume the initial partition is P1=(F1, F_m1),
#             P2=(F_(m1+1),..,F_m)
# Find G(F(T)) and calculate eps1, eps2
#G(y)=#ys less than y
K=2 ###number of clusters
m_vector=c(1,m1,m1+1,m) #the length is 2*K (1,2,3,5)



G_list=list() ##K lists
for(k in 1:K){
j=2*k-1
G_t=NULL
partition=m_vector[j]:m_vector[j+1]
for (i in partition){
g=sum(y_matrix[partition,k]<=y_matrix[i,k])/length(partition)
G_t=c(G_t,g)
}
G_list=list.append(G_list,G_t)
}

epson=NULL
for (i in 1:K){
eps=(max(G_list[[i]])-min(G_list[[i]]))/m
epson=c(epson,eps)
```

```
}

##Calculate equation 5.8 for each unit
#first define the function when beta=0 and beta=1
#when beta=0
e1234_W=function(x1p,x2p,x1m,x2m){
out=max(x1p+x2p-1,0)-max(x1p+x2m-1,0)-
        max(x1m+x2p-1,0)+max(x1m+x2m-1,0)
return(out)
}

#when beta=1
e1234_M=function(x1p,x2p,x1m,x2m){
out=min(x1p,x2p)-min(x1p,x2m)-min(x1m,x2p)+min(x1m,x2m)
return(out)
}

#obtain the omega_matrix for each unit

w_list=list() #this is a 2*2 matrix with values: G_t1(x1+epson1),
#     G_t2(x2+epson2), G_t1(x1-epson1), G_t2(x2-epson2)
for(k in 1:K){
j=2*k-1

partition=m_vector[j]:m_vector[j+1]
for (i in partition){
```

```
w_matrix=matrix(0,2,2)

w_matrix[1,1]=sum(y_matrix[partition,1]<=(y_matrix[i,1]
    +epson[1]))/length(partition)
w_matrix[2,1]=sum(y_matrix[partition,2]<=(y_matrix[i,2]
    +epson[1]))/length(partition)
w_matrix[1,2]=sum(y_matrix[partition,1]<=(y_matrix[i,1]
    -epson[2]))/length(partition)
w_matrix[2,2]=sum(y_matrix[partition,2]<=(y_matrix[i,2]
    -epson[2]))/length(partition)

w_list=list.append(w_list,w_matrix)
}
}

#Then apply e1234_W and e1234_M to each unit
#define beta_w as a 2*m matrix, each colunm is the h_w
#for beta=0 and beta=1
beta_w=matrix(0,2,m)
for (i in 1:m){
beta_w[1,i]=e1234_W(w_list[[i]][1,1],w_list[[i]][2,1],
            w_list[[i]][1,2],w_list[[i]][2,2])
beta_w[2,i]=e1234_M(w_list[[i]][1,1],w_list[[i]][2,1],
            w_list[[i]][1,2],w_list[[i]][2,2])
}
#get the h_k for each k=1,...,K
```

```
beta0=NULL #the vector of h_k when beta=0
beta1=NULL #the vector of h_k when beta=1
for (k in 1:K){
j=2*k-1
partition=m_vector[j]:m_vector[j+1]
b0=sum(beta_w[1,partition])
b1=sum(beta_w[2,partition])
beta0=c(beta0,b0)
beta1=c(beta1,b1)
}

##calculate p_k for each cluster
#  (the associate probability for each cluster k)
p=NULL
for (k in 1:K){
j=2*k-1
partition=m_vector[j]:m_vector[j+1]
p=c(p,length(partition)/m)
}

#calculate the Likelihood
L_matrix=matrix(c(sum(p*beta0),
sum(p*c(beta0[1],beta1[2])),sum(p*c(beta0[2],beta1[1])),
        sum(p*beta1)),2,2)
beta.option=list(c(0,0),c(0,1),c(1,0),c(1,1))
beta.estimate=beta.option[[which.max(L_matrix)]]
```

```
#evaluate each unit under the other partition
w_list2=list()
k=1
j=2*k-1
partition=m_vector[j]:m_vector[j+1]
k=k+1
j=2*k-1
partition.evaluate=m_vector[j]:m_vector[j+1]
for (i in partition){
w_matrix=matrix(0,2,2)
w_matrix[1,1]=sum(y_matrix[partition.evaluate,1]<=(y_matrix[i,1]
    +epson[1]))/length(partition.evaluate)
w_matrix[2,1]=sum(y_matrix[partition.evaluate,2]<=(y_matrix[i,2]
    +epson[1]))/length(partition.evaluate)
w_matrix[1,2]=sum(y_matrix[partition.evaluate,1]<=(y_matrix[i,1]
    -epson[2]))/length(partition.evaluate)
w_matrix[2,2]=sum(y_matrix[partition.evaluate,2]<=(y_matrix[i,2]
   -epson[2]))/length(partition.evaluate)

w_list2=list.append(w_list2,w_matrix)
}
k=2
j=2*k-1
partition=m_vector[j]:m_vector[j+1]
```

```
k=k-1
j=2*k-1
partition.evaluate=m_vector[j]:m_vector[j+1]
for (i in partition){
w_matrix=matrix(0,2,2)
w_matrix[1,1]=sum(y_matrix[partition.evaluate,1]<=(y_matrix[i,1]
   +epson[1]))/length(partition.evaluate)
w_matrix[2,1]=sum(y_matrix[partition.evaluate,2]<=(y_matrix[i,2]
   +epson[1]))/length(partition.evaluate)
w_matrix[1,2]=sum(y_matrix[partition.evaluate,1]<=(y_matrix[i,1]
   -epson[2]))/length(partition.evaluate)
w_matrix[2,2]=sum(y_matrix[partition.evaluate,2]<=(y_matrix[i,2]
   -epson[2]))/length(partition.evaluate)

w_list2=list.append(w_list2,w_matrix)
}

if (all(beta.estimate==c(0,0))){
h1=beta_w[1,m_vector[1]:m_vector[2]]
for (i in m_vector[3]:m_vector[4]){
h_w=e1234_W(w_list2[[i]][1,1],w_list2[[i]][2,1],
       w_list2[[i]][1,2],w_list2[[i]][2,2])
h1=c(h,h_w)
}
h2=beta_w[1,m_vector[3]:m_vector[4]]
for (i in m_vector[1]:m_vector[2]){
```

```
h_w=e1234_W(w_list2[[i]][1,1],w_list2[[i]][2,1],
        w_list2[[i]][1,2],w_list2[[i]][2,2])
h2=c(h2,h_w)
}} else if (beta.estimate==c(1,1)){
h1=beta_w[2,m_vector[1]:m_vector[2]]
for (i in m_vector[3]:m_vector[4]){
h_w=e1234_M(w_list2[[i]][1,1],w_list2[[i]][2,1],
        w_list2[[i]][1,2],w_list2[[i]][2,2])
h1=c(h,h_w)
}
h2=beta_w[2,m_vector[3]:m_vector[4]]
for (i in m_vector[1]:m_vector[2]){
h_w=e1234_M(w_list2[[i]][1,1],w_list2[[i]][2,1],
        w_list2[[i]][1,2],w_list2[[i]][2,2])
h2=c(h2,h_w)
}} else if (beta.estimate==c(0,1)){
h1=beta_w[1,m_vector[1]:m_vector[2]]
for (i in m_vector[3]:m_vector[4]){
h_w=e1234_W(w_list2[[i]][1,1],w_list2[[i]][2,1],
        w_list2[[i]][1,2],w_list2[[i]][2,2])
h1=c(h,h_w)
}
h2=beta_w[2,m_vector[3]:m_vector[4]]
for (i in m_vector[1]:m_vector[2]){
h_w=e1234_M(w_list2[[i]][1,1],w_list2[[i]][2,1],
        w_list2[[i]][1,2],w_list2[[i]][2,2])
```

```
h2=c(h2,h_w)
}} else {
h1=beta_w[2,m_vector[1]:m_vector[2]]
for (i in m_vector[3]:m_vector[4]){
h_w=e1234_M(w_list2[[i]][1,1],w_list2[[i]][2,1],
      w_list2[[i]][1,2],w_list2[[i]][2,2])
h1=c(h,h_w)
}
h2=beta_w[1,m_vector[3]:m_vector[4]]
for (i in m_vector[1]:m_vector[2]){
h_w=e1234_W(w_list2[[i]][1,1],w_list2[[i]][2,1],
       w_list2[[i]][1,2],w_list2[[i]][2,2])
h2=c(h2,h_w)
}}
h_matrix=cbind(h1,h2)
h_matrix=cbind(h_matrix,apply(h_matrix,1,which.max),
                  rep(c(1:K),nk[1:K]/c))
cluster=rep(c(1:K),nk[1:K]/c)
accuracy=sum(diag(table(h_matrix[,K+1],cluster)))/m
result=list(h_matrix,accuracy)
return(result)
}

########Simulation Example When H is non-differentiable
#                  with one variable
############Section 6.1: Example 1
```

```
set.seed(5)
p1=rnorm(300, 2, 4)
p2=rnorm(200,10,3)
mydata=c(p1,p2)
K=2
nk=c(300,200)
c=100
m_vector=c(1,2,3,5)
scatter_plot1D(mydata,K,nk)
cdf_plot1D(mydata,c,K,nk)
Tvalues=c(0,10)
cdf_plot.T1D(mydata,c,K,nk,Tvalues)

mbc.sym.emp(Tvalues,K, m_vector,mydata,c)

############Section 6.1: Example 2
set.seed(31)
n1=700
n2=300
n=c(n1,n2)
c=50
p1=rnorm(n1, 2, 4)
p2=rnorm(n2,30,3)
mydata=c(p1,p2)
K=2
m_vector=c(1,10,11,20)
```

```
scatter_plot1D(mydata,K,nk)
cdf_plot1D(mydata,c,K,nk)
Tvalues=c(5,25)
cdf_plot.T1D(mydata,c,K,nk,Tvalues)

mbc.sym.emp(Tvalues,K, m_vector,mydata,c)



############Section 6.1: Example 3
set.seed(3)
n1=700
n2=300
n3=500
nk=c(n1,n2,n3)
c=100
p1=rnorm(n1, 2, 4)
p2=rnorm(n2,10,3)
p3=rnorm(n3,15,5)
mydata=c(p1,p2,p3)
K=3
nk=c(n1,n2,n3)
m_vector=c(1,6,7,11,12,15)
scatter_plot1D(mydata,K,nk)
cdf_plot1D(mydata,c,K,nk)
Tvalues=c(5,15)
```

```
cdf_plot.T1D(mydata,c,K,nk,Tvalues)

mbc.sym.para_margin(Tvalues,K, m_vector,mydata,c)

#####################Simulation Examples When H is differentiable
###Simulation Example for 1 variables with parametric estimation
###3 classes
set.seed(33)
n1=7000
n2=3000
n3=5000
c=100
p1=rbeta(n1,2,2,ncp=0)
p2=rbeta(n2, 1, 3, ncp = 0)
p3=rbeta(n3, 5, 1, ncp = 0)
mydata=c(p1,p2,p3)
K=3
nk=c(n1,n2,n3)
P0=c(1,80,81,110,111,150)
scatter_plot1D(mydata,K,nk)
cdf_plot1D(mydata,c,K,nk)
Tvalues=c(0.45,0.65)
cdf_plot.T1D(mydata,c,K,nk,Tvalues)
mbc.sym1D.para(mydata,c,P0,K,Tvalues)

###Simulation Example for 1 variables with parametric estimation
```

```
###3 classes
n1=7000
n2=3000
set.seed(2014)
sigma1=diag(2)
x1=mvrnorm(n1,c(3,0),sigma1)##results A
set.seed(2015)
sigma2=2*diag(2)
x2=mvrnorm(n2,c(4,4),sigma2)
n=n1+n2
c=100
mydata=rbind(x1,x2)
nk=c(n1,n2)
K=2
P0=c(1,60,61,100)

scatter_plot2D(mydata,K,nk)
cdf_plot.multiD(mydata,c,K,nk)
Tvalues=matrix(c(3,4,0.8,2),2,2)
cdf_plot.TmultiD(mydata,c,K,nk,Tvalues)
mbc.sym.multiD.para(mydata,c,P0,K,Tvalues)

###Simulation Example for 3 variables with parametric estimation
###2 classes
n1=7000
n2=3000
```

```
set.seed(333)
sigma1=diag(3)
x1=mvrnorm(n1,c(1.5,1,0),sigma1)##results A
set.seed(33)
sigma2=2*diag(3)
x2=mvrnorm(n2,c(4,3,3),sigma2)
mydata=rbind(x1,x2)
c=50
nk=c(n1+n2)
K=2
P0=c(1,120,121,200)
scatter_plot3D(mydata,K,nk)
cdf_plot.multiD(mydata,c,K,nk)
Tvalues=matrix(c(1.8,2.6,1,2,0.5,2,3),2,3)
cdf_plot.TmultiD(mydata,c,K,nk,Tvalues)
mbc.sym.multiD.para(mydata,c,P0,K,Tvalues)
```

Reference

Behnen, K., Hušková, M. and Neuhaus, G. (1985). Rank estimators of scores for testing independence. *Statistics and Risk Modeling* 4, 157-174.

Billard, L. and Diday, E. (2006). *Symbolic Data Analysis: Conceptual Statistics and Data Mining.* Wiley, Chichester.

Bouyé, E., Durrleman, V., Nikeghbali, A., Riboulet, G. and Roncalli, T. (2000). Copulas for finance - a reading guide and some applications. *Technical Report.*

Cambanis, S., Huang, S. and Simons, G. (1981). On the theory of elliptically contoured distributions. *Journal of Multivariate Analysis* 11(3), 368-385.

Celeux, G., Diday, E., Govaert, G., Lechevallier, Y. and Ralambondrainy, H. (1989). *Classification Automatique des Données.* Dunod, Paris.

Chambers, J. M. (1998). *Programming with Data: A Guide to the S Language.* Springer.

Charpentier, A. and Segers, J. (2006). Convergence of archimedean copulas. CentER Discussion Paper Series No. 2006-28.

Clayton, D. G. (1978). A model for association in bivariate life tables and its application in epidemiological studies of familial tendency in chronic disease incidence. *Biometrika* 65, 141-151.

Deheuvels, P. and Hominal, P. (1979). Estimation non paramétrique de la densité compte tenu d'informations sur le support. *Revue de Statistique Appliquée* 27, 47-68.

Devroye, L. and L. Gyorfi (1985). *Nonparametric Density Estimation: The L1 View*. John Wiley, New York.

Diday, E. (1987). Introduction *à* l'approche symbolique en analyse des données. *Premier Jounelles Symbolique-Numerique*, CEREMADE, Universit*é* de Paris, Dauphine, 21-56.

Diday, E. and Simon, J. C. (1976). Clustering analysis. In: *Digital Pattern Recognition* (ed. K. S. Fu). Springer, Berlin, 47-94.

Diday, E., Schroeder, A. and Ok, Y. (1974). The dynamic clusters method in pattern recognition. *Proceedings of International Federation for Information Processing Congress*. Elsevier, New York, p691-697.

Diday, E. and Vrac, M. (2005). Mixture decomposition of distributions by copulas in the symbolic data analysis framework. *Discrete Applied Mathematics* 147, 27-41.

Fang, K. T., Fan, J. Q. and Xu, J. L. (1987). The distributions of quadratic forms of random idempotent matrices with their applications. *Chinese Journal of Applied Probability and Statistics* 3, 289 - 297.

Fang, K.T., Kotz, S. and Ng, K.W. (1990). *Symmetric Multivariate and Related Distributions*. Chapman and Hall, London.

Fermanian, J. D. and Scaillet, O. (2005). Some statistical pitfalls in copula modeling for financial applications. In: *Nova Science Capital Formation, Governance and Banking* 57 - 72.

Fraley, C. and Raftery, A. E. (1998). How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal* 41, 578-588.

Frank, M. J. (1979). On the simultaneous associativity of $F(x, y)$ and $x+y-F(x, y)$. *Aequationes Mathematicae* 19, 194-226.

Frees, E. W. and Valdez, E. A. (1998). Understanding relationships using copulas. *North American Actuarial Journal* 2, 1-25.

Frees, E. W. and Wang, P. (2005). Credibility using copulas. *North American Actuarial Journal* 9(2), 31-48.

Galili, T. (2016). Hierarchical cluster analysis on famous data sets - enhanced with the dendextend package. *Technical Report.*

Gasser, T. and Mueller, H. G. (1979). Kernel estimation of regression functions. In Smoothing Techniques for Curve Estimation. In *Lecture Notes in Mathematics* 757, 23-68. New York: Springer Verlag.

Gijbels, I. and Mielniczuk, J. (1990). Estimating the density of a copula function. *Communications in Statistics-Theory and Methods* 19(2), 445-464.

Good, I. J. and Gaskins, R. A. (1971). Nonparametric roughness penalties for probability densities. *Biometrika* 58, 255-277.

Gu, C. (1993). Smoothing spline density estimation: a dimensionless automatic algorithm. *Journal of the American Statistical Association* 88, 495-504.

Gu, C. and Qiu, C. (1993). Smoothing spline density estimation. *The Annals of Statistics* 21(1), 217-234.

Gumbel, E. J. (1958). *Statistics of Extremes.* Columbia University Press.

Gumbel, E.J. (1960) Bivariate exponential distributions. *Journal of the American Statistical Association*55, 698-707.

Han, J., Kamber, M. and Pei, J. (2012). *Data Mining: Concepts and Techniques*. Elsevier.

Hastie, T., Tibshirani,R. and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.

Joe, H. (1997). *Multivariate Models and Dependence Concepts.* Chapman and Hall, London.

Joe, H. and Xu, J. (1996). The estimation method of inference functions for margins for multivariate models. Technical Report No. 166, Department of Statistics, University of British Columbia.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In: *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability* (eds. L. M. LeCam and J. Neyman). University of California Press, Berkeley, 1, 281-299.

Marshall, A.W. and Olkin, I. (1988). Families of multivariate distributions. *Journal of the American Statistical Association* 83, 834-841.

McNicholas, P.D. (2016). Model-based clustering. *Journal of Classification* 33, 331-373.

Müller, H. (1991). Smooth optimum kernel estimators near endpoints. *Biometrika* 78, 521-520.

Nelsen, R. B. (2006). *An Introduction to Copulas.* Second Edition, Springer.

Nolan, J. P. (2006). *Stable Distributions - Models for Heavy Tailed Data*. Birkhauser.

O'Sullivan, F. (1988). Fast computation of fully automated log-density and log-hazard estimators. *Journal on Scientific and Statistical Computing* 9, 363-379.

Rice, J. (1984). Bandwidth choice for nonparametric regression. *The Annals of Statistics* 12(4), 1215-1230.

Schuster, E. F. (1985). Incorporating support constraints into nonparametric estimators of densities. *Communications in Statistics-Theory and Methods* 14(5), 1123-1136.

Scott, D. W. (2009). Averaged shifted histogram. *Computational Statistics* 2(2), 160-164.

Scott, D.W. (1992). *Multivariate Density Estimation - Theory, Practice and Visualization*. Wiley, New York.

Scott, D. W. (1985). Averaged shifted histograms effective nonparametric density estimators in several dimensions. *Annals of Statistics* 13, 1024-1040.

Scott, D.W., Tapia, R.A. and Thompson, J.R. (1977). Kernel density estimation revisited. *Nonlinear Analysis* 1, 339-372.

Silverman, B.W. (1982). Algorithm as 176: kernel density estimation using the fast Fourier transform. *Applied Statististics* 31, 93-97.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London.

Sklar, A. (1959.) Fonction de répartition *à* n dimensions et leurs marges. *Institute Statistics Université de Paris* 8, 229-231.

Symons, M. J. (1981). Clustering criteria and multivariate normal mixtures. *Biometrics* 37, 35-43.

Vrac, M., Billard, L., Diday, E. and Chédin, A. (2012). Copula analysis of mixture models. *Coputational Statistics* 27, 427-457.

Vrac, M., Che'din, A. and Diday, E. (2005). Clustering a global field of atmospheric profiles by mixture decomposition of copulas. *Journal of Atmospheric and Ocean Technology* 22, 1445-1459.

Vrac, M. (2002). *Analyse et Modelisation de Données Probabilistes Par Décomposition de Mélange de Copules et Application à une Base de Données Climatologiques*. Thèse de Doctorat, Université de Paris, Dauphine.

Wand, M. P., Marron, J. S. and Ruppert, D. (1991). Transformations in density estimation. *Journal of the American Statistical Association* 86(414), 343-353.

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association* 58, 236-244.

Yan, J. (2007). Enjoy the joy of copulas: with a package copula. *Journal of Statistical Software* 21(4), 1-21.