# COPULA-BASED MIXTURES OF REGRESSION MODELS FOR MULTIVARIATE RESPONSE DATA

by

Claire Cui

Submitted in partial fulfillment of the requirements
for the degree of Master of Science

at

Dalhousie University
Halifax, Nova Scotia
December 2023

# Table of Contents

# List of Tables

# List of Figures

## Abstract

Clustering is an unsupervised learning method that serves as a powerful tool for uncovering hidden patterns and structures within complex datasets. In recent years, the use of mixtures of multiple linear regression models in clustering has gained popularity due to its ability to account for underlying heterogeneity in the data and provide a more representative interpretation of covariate effects. However, there is a paucity of these models for data with multivariate responses, particularly when these response variables are dependent. One approach that has been applied in the case of multivariate response data is copula regression models. Copulas are joint distribution functions with uniform margins and can be seen as representing the dependence structure of a random vector. In copula regression, a copula function is employed to induce dependence between different response variables through the random error term in the regression model. The concept behind using copula regression models is that they allow us to capture complex dependencies among variables while still maintaining flexibility and interpretability.

In this work, we propose a finite mixture of copula regression (CMixR) models for clustering and interpreting covariate effects in heterogeneous multivariate continuous response data. We present an Expectation Conditional Maximization algorithm for estimation. The model performance is tested and compared to existing methods using a simulation study and a data analysis on the morphological properties of purple rock crabs. These results show an overall better clustering performance in comparison to existing methods.

## Acknowledgements

I would like to express my gratefulness to all individuals who have played a pivotal role in the successful completion of this thesis. First, I am deeply thankful to my supervisor, Dr. Orla Murphy. Her unwavering support, guidance, and expertise have been invaluable throughout this research journey. Weekly meetings with Dr.Murphy to discuss areas for improvement and how to address them have generated tremendous support in the process. Especially in the process of revising the paper, Dr.Murphy took great pains to give professional guidance not only on the specific content of the paper but also on the grammar and format of the paper. Second, I would like to thank my co-supervisor, Dr. Paul McNicholas, for his important guidance during the research. Third, I extend my sincerest appreciation to my family, who have been a constant source of encouragement and support. Finally, I would express a special thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC) for providing me with funding to conduct this research project.

# Chapter 1

# Introduction

Clustering has been a popular topic in past decades due to its numerous applications and benefits. In fields as diverse as biology, finance, marketing, and social sciences, clustering plays a pivotal role. Biologists utilize it to categorize species based on genetic similarities, while financial analysts employ it to identify distinct market segments. In marketing, clustering helps create targeted advertising campaigns, and sociologists use it to understand community dynamics. Take the e-commerce industry for example, businesses often collect data about their customers, including their purchase history, browsing behavior, demographics, and more, and then clustering can be used to divide customers into distinct groups based on their preferences. For example, cluster 1 might represent budget-conscious shoppers who frequently purchase discounted items, cluster 2 might represent high-value customers who make large purchases and prefer premium products, and cluster 3 could include occasional shoppers who only make purchases during seasonal sales. Therefore, by understanding these customer segments, businesses can adjust their marketing strategies and customize promotions to better meet the needs of each group. Ultimately, this will increase customer satisfaction and revenue. A comprehensive review of clustering applications can be found in the paper written by Saxena et al. (2017).

There are two main approaches used for clustering, one is distance-based and the other is model-based. Distance-based clustering methods, like K-means and hierarchical clustering, rely on measuring the similarity or dissimilarity between data points using distance metrics. Distance-based clustering is widely used due to its relatively straightforward implementation and computational efficiency. However, it may struggle with non-spherical or irregularly shaped clusters and might be sensitive to initializations. K-Means, in particular, is sensitive to the initial assignment of cluster centres. Different initializations can lead to different clustering results, potentially resulting in sub-optimal solutions. Even though multiple initializations can help mitigate this issue, they do not necessarily eliminate it. Also, distance-based clustering does not provide probabilistic assignments of data points to clusters, making it less suitable for situations where overlapping clusters are present.

Model-based clustering operates within a probabilistic framework. It assumes that the data is generated from a mixture of probability distributions, and each probability distribution represents a cluster. So, the clustering process involves estimating the parameters of these distributions. By doing parameter estimation, model-based clustering often provides more interpretable results because it is easier to describe and understand the characteristics of each group. In addition, model-based clustering can capture clusters with different shapes, which makes them more suitable for complex clusters. In contrast, distance-based clustering using Euclidean distance as metrics assumes spherical clusters, so it is primarily expected to perform well when the data within clusters follows a normal distribution. Furthermore, instead of assigning each point to a single cluster, model-based clustering calculates the likelihood of a data point belonging to each cluster. This allows for a probabilistic interpretation of cluster membership, which can be valuable when data points have overlapping characteristics. In short, model-based clustering offers flexibility when dealing with complex cluster shapes and provides probabilistic cluster assignments. However, it requires assumptions about data distributions and is thus sensitive to model misspecification. Also, like the distance-based method, the model-based method suffers from initialization. Furthermore, the computational cost for model-based clustering can be relatively high, as it involves estimating model parameters usually by optimizing likelihood functions and potentially using multiple initializations, making it computationally time-consuming.

In the context of regression, the clustering problem involves grouping similar data points into distinct clusters based on their covariate values and corresponding regression responses. The goal of traditional regression is usually to predict a response value, but clustering within regression aims to identify clusters of data with similar regression patterns. When data points are from different underlying regression models, we can uncover these latent groups using a clustering analysis. So, if each cluster is characterized by a unique regression model, this is called a finite mixture of regression models. In the clustering context, this model is important because it takes covariate effects into account. Leisch (2004) proposed the famous framework, `flexmix`, to fit finite mixture models and latent class regression. However, this existing method focuses on the univariate response variable. This model could be extended to use in multivariate response cases but assumes the response variables are mutually independent, which is often unsuitable. Therefore, finite mixtures of regression models with dependent responses are needed.

Copula-based regression models consider the dependence structure between observations by inducing a copula through the random error term in the regression model. As proposed by Pitt et al. (2006) and summarized by Kolev and Paiva (2009), copula regression could be a helpful tool when dealing with variables that have multiple observations. For example, copula regression can be used in longitudinal data analysis if repeated measurements on the same subject are correlated. Copula regression is later developed by Masarotto and Varin (2012) to include correlated observations collected sequentially in time and spatially correlated data. However, the traditional copula regression model does not consider the potential heterogeneity in data induced by latent groups. Also, the Gaussian copula marginal regression (`gcmr`) developed by Masarotto and Varin (2012) is only applicable to a single response variable with multiple observations, e.g., longitudinal data. It is worth noting that Bermúdez and Karlis (2022) proposed a copula-based finite mixture of bivariate discrete distributions to analyze the counts of automobile insurance claims. However, this model has not been flexibly extended to high-dimensional discrete cases and continuous cases.

To tackle the problem of adding flexibility to multivariate response regression models, we extract elements from both the mixture of regression models and the copula regression and combine them to create a new model called the copula-based mixtures of regression (CMixR) model. The primary contribution of the CMixR model is addressing the limitations of each approach and capitalizing on their respective strengths. Consequently, the CMixR model offers significantly greater flexibility to capture the dependence structure between correlated responses and enhanced performance when compared to other existing methods.

The remainder of the thesis is organized as follows. In Chapter 2, basic ideas on copulas, finite mixture models, copula-based regression, and the Expectation-Maximization (EM) algorithm are introduced. In Chapter 3, the CMixR model is proposed with computational details on the Maximum Likelihood (ML) estimation via the Expectation Conditional Maximization (ECM) algorithm. A simulation study and a data analysis on purple rock crab has been conducted in Chapter 4. In Chapter 5, we discuss the results, conclude the work, and list several ideas for future work.

# Chapter 2

# Background

In Section 2.1, background knowledge of copulas is presented, including examples and their usage in regression. Then, finite mixture models are introduced in Section 2.2 with an extension to mixtures of regression models. Section 2.3 provides computational details on the EM algorithm and presents an example using a Gaussian mixture model. Finally, Section 2.4 includes a discussion on clustering performance assessment.

## 2.1 Copulas

A $p$-variate copula $C : [0,1]^p \to [0,1]$ is a multivariate cumulative distribution function with uniform marginals. Sklar's theorem (Sklar, 1959) states that any multivariate distribution can be written in terms of its univariate marginal distributions and a copula which describes the dependence structure between the variables. Consider the continuous bivariate random vector $(X, Y)$ with joint cumulative distribution function (cdf) $H(x, y)$ and corresponding marginal cdfs $F(x)$ and $G(y)$, then there exists a copula $C : [0,1]^2 \to [0,1]$, such that,

$$H(x,y) = C\{F(x), G(y); \boldsymbol{\theta}\}\,, \quad x, y \in R\,, \tag{2.1}$$

where $\boldsymbol{\theta}$ is the parameter(s) in copula $C$. In this case, $C$ is unique as $X$ and $Y$ are continuous. Consequently, the probability density function (pdf) $h$ of $X$ and $Y$ can be derived in terms of the copula using chain rule *viz.*

$$\begin{aligned} h(x,y) &= \frac{\partial^2 H(x,y)}{\partial x \partial y} \\ &= \frac{\partial^2 C\{F(x), G(y)\}}{\partial x \partial y} \\ &= \frac{\partial^2 C\{F(x), G(y)\}}{\partial F(x) \partial G(y)} \frac{\partial F(x)}{\partial x} \frac{\partial G(y)}{\partial y} \\ &= c\{F(x), G(y)\} f(x) g(y)\,, \end{aligned} \tag{2.2}$$

where $c$ is the copula density function of the copula $C$ and $f(x)$ and $g(y)$ are the marginal pdfs of $X$ and $Y$, respectively. Note that the density in (2.2) can be easily extended to the multivariate case. Consider a $p$-dimensional random vector $\boldsymbol{X} = (X_1, X_2, \ldots, X_p)$ with continuous margins, its joint density function can be written as:

$$h(x_1, \ldots, x_p) = c\{F_1(x_1), \ldots, F_p(x_p); \boldsymbol{\theta}\} f_1(x_1) \cdots f_p(x_p)\,, \tag{2.3}$$

where $f_i(x_i)$ for $i = 1, \ldots, p$ are the marginal densities for $X_1, \ldots, X_p$ and $c$ is the copula density, which describes the multivariate dependence structure.

Kendall's $\tau$, proposed by Kendall (1938), is a common measure of rank correlations for continuous variables. Let $F$ be a continuous bivariate cdf and let $(X_1, X_2)$, $(X_1', X_2')$ be independent random pairs with distribution $F$. A pair of data observations are considered concordant if both $(X_1 - X_1')$ and $(X_2 - X_2')$ have the same sign. Conversely, if these differences have opposite signs, the pair is labelled discordant. In cases where $X_1 = X_1'$, $X_2 = X_2'$, or both, we refer to the comparison as a tie. Note that ties are not categorized as either concordant or discordant. Kendall's $\tau$ is defined as the probability of concordance minus the probability of discordance, i.e.,

$$\begin{aligned} \tau &= \Pr((X_1 - X_1')(X_2 - X_2') > 0) - \Pr((X_1 - X_1')(X_2 - X_2') < 0) \\ &= 2\Pr((X_1 - X_1')(X_2 - X_2') > 0) - 1 \\ &= 4 \int F dF - 1\,. \end{aligned}$$

Kendall's $\tau$ ranges from $-1$ to 1, measures the strength of the monotonic relationship between variables and can be viewed as a quantitative summary of pairwise dependence. A positive Kendall's $\tau$ value indicates a positive association between the two variables, while a negative value suggests a negative association. If Kendall's $\tau$ is 0, it implies that there is no association between the two variables. In a bivariate copula model for continuous variables, the dependence structure between the two variables is determined by the copula parameters. Interestingly, for continuous variables, Kendall's $\tau$ can be written as a function of the copula only. For many common copula models, $\tau$ can be written explicitly in terms of the copula dependence parameter. Some relationships between Kendall's $\tau$ and copula parameter $\theta$ are listed in Table 2.1, all of which are available using the `iTau` function in the `copula` R package. This property can be useful for simulations as the copula parameters can be chosen to correspond to the desired level of Kendall's $\tau$. This relationship has also been used for moment-based estimation.

Table 2.1: Relationships between Kendall's $\tau$ and the copula parameter $\theta$ for common copula models (Joe, 1997). Note for the Frank copula, $D(\theta) = \int_0^\theta \frac{x/\theta}{\exp(x)-1} dx$ is the Debye function.

| Name | Range of $\theta$ | Kendall's $\tau \in [-1, 1]$ |
|---|---|---|
| Gaussian | $\theta \in (-1, 1)$ | $\frac{2}{\pi} \arcsin \theta$ |
| Clayton | $\theta \in (0, \infty)$ | $\frac{\theta}{\theta+2}$ |
| Gumbel | $\theta \in (1, \infty)$ | $1 - \frac{1}{\theta}$ |
| Frank | $\theta \in (-\infty, \infty)$ | $1 - \frac{4}{\theta} + \frac{4}{\theta} D(\theta)$ |

For the discrete margin case, the copula $C$ for a random vector is no longer unique and Sklar's Equation (2.1) is satisfied by infinitely many copulas. Also, the computational cost for discrete margins is quite high in even moderate dimensions. For these reasons, we restrict ourselves to the continuous case in this thesis.

### 2.1.1 Examples of copulas

There are several families of copula models. Two common families which will be used in this thesis are the elliptical and Archimedean copula families.

Elliptical copulas are based on elliptical distributions, such as the multivariate Gaussian and multivariate Student t distributions, the explicit formula for Gaussian copula is listed in Table 2.2. The Gaussian copula is a well-known member of this family. For example, the two-dimensional Gaussian copula assumes the dependence structure is captured by a correlation coefficient. Unlike the Gaussian copula, the Student t copula is more robust to outliers as it allows for upper and lower tail dependence. Plots of a random sample of size 500 from the Gaussian and the Student t copula with Kendall's $\tau$ value of 0.6 are shown in Figure 2.1. The difference in tail dependence can be seen in the bottom left and top right corners of the plot.

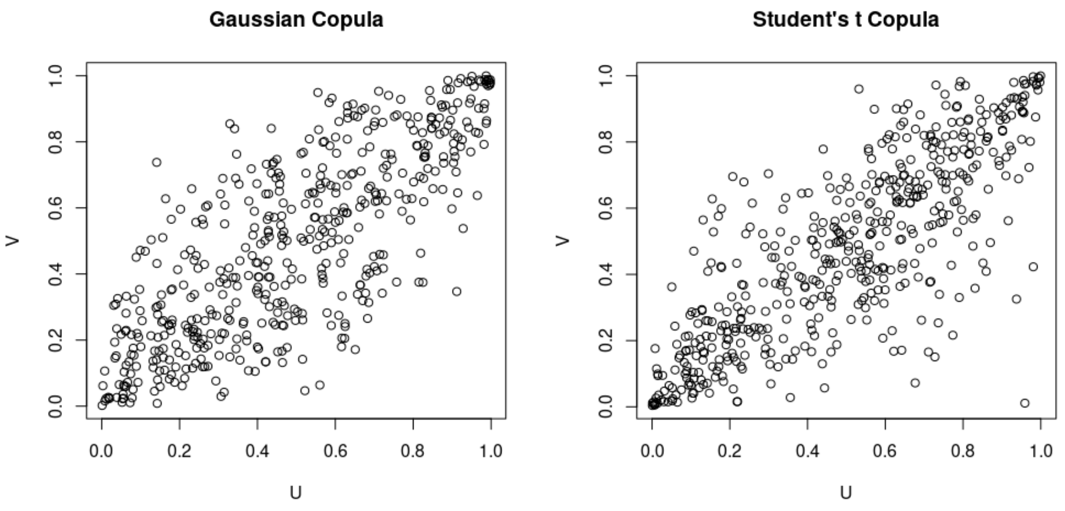

Figure 2.1: Samples of size 500 of from Clayton and Gumbel copula with $\tau = 0.6$.

Archimedean copulas are completely determined by a generator function, $\phi(t)$, for $t \in (0, 1]$. Examples of Archimedean copulas include the Clayton, Gumbel, and Frank copulas. As an example, the generator function for the Gumbel copula is $\phi(t) = |ln(t)|^{\theta+1}$. Different Archimedean copulas offer different types of tail dependence. For example, Clayton copulas exhibit lower-tail dependence, while Gumbel copulas have upper-tail dependence. On the other hand, Frank copulas exhibit no tail dependence. We can adjust the strength and shape of the dependence by choosing an appropriate value for the parameter. For example, the Clayton copula with parameter $\theta = 2$ exhibits stronger lower-tail dependence than $\theta = 0.5$. Plots of random samples of size 500 from the Clayton, Gumbel, and Frank copulas with Kendall's $\tau$ value of 0.6 are shown in Figure 2.2 and 2.3.

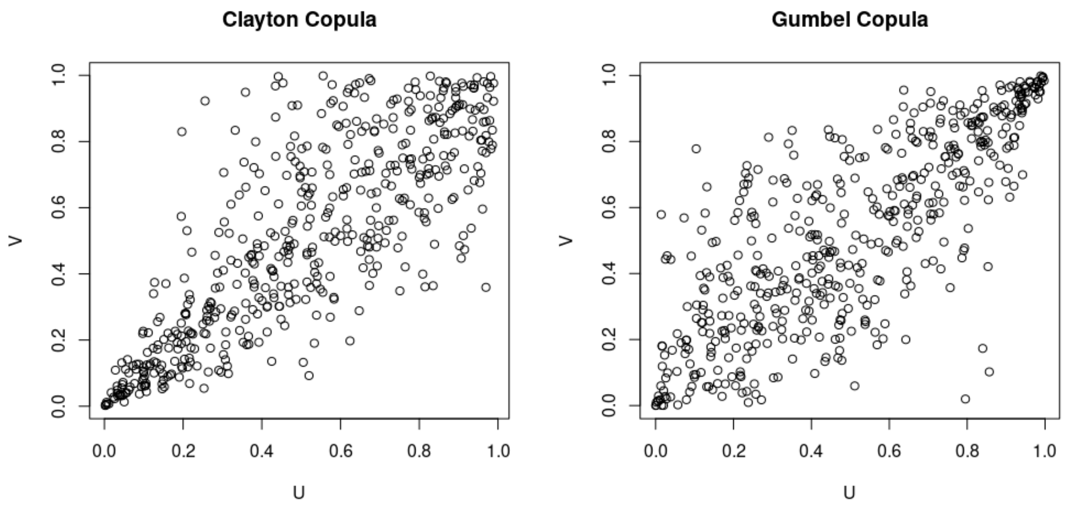

Figure 2.2: Samples of size 500 of from Clayton and Gumbel copula with $\tau = 0.6$.

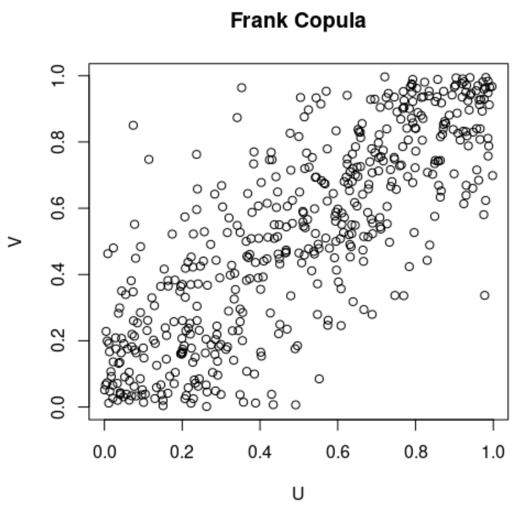

Figure 2.3: Samples of size 500 of from Frank copula with $\tau = 0.6$.

A summary of the bivariate Gaussian, Clayton, Gumbel, and Frank copulas, which will be used in this thesis, can be found in Table 2.2. Only bivariate cases are shown in the table, but these copulas can be generalized to the multivariate case as well.

Table 2.2: Summary of bivariate copulas used in this thesis. Note for the Gaussian copula, $\Phi^{-1}$ is the inverse CDF of a standard normal and $\Phi_\theta$ is the joint CDF of a bivariate normal distribution with mean vector zero and covariance matrix $\begin{bmatrix} 1 & \theta \\ \theta & 1 \end{bmatrix}$.

| Name | $C(u,v)$ | Parameter |
|---|---|---|
| Gaussian | $\Phi_\theta\left\{\Phi^{-1}(u), \Phi^{-1}(v)\right\}$ | $\theta \in (-1,1)$ |
| Clayton | $\{\max(u^{-\theta}) + v^{-\theta} - 1, 0\}^{\frac{-1}{\theta}}$ | $\theta \in (0,\infty)$ |
| Gumbel | $e^{-(\lvert\log u\rvert^{\theta+1}+\lvert\log v\rvert^{\theta+1})^{1/(\theta+1)}}$ | $\theta \in (1,\infty)$ |
| Frank | $-\theta^{-1}\ln\left(\frac{1-e^{-\theta}-(1-e^{-\theta u})(1-e^{-\theta v})}{1-e^{-\theta}}\right)$ | $\theta \in (-\infty,\infty)$ |

### 2.1.2 Copula regression

As seen in the previous section, copulas can be a useful tool for dependence analysis. In recent years, copulas have been applied to regression problems, yielding copula-based regression models. One of the most popularly used models is the Gaussian copula regression model, which was proposed by Pitt et al. (2006) and further developed by Masarotto and Varin (2012). Suppose there is a single dependent variable $Y$ with $n$ correlated observations denoted as $\boldsymbol{y} = (y_1, \ldots, y_n)'$, a case which, for example, appears in longitudinal data. We denote the marginal cumulative distribution of $Y_i$ by $F(Y_i|\boldsymbol{X})$ where $\boldsymbol{X} = (X_1, \ldots, X_P)'$ is a $P$-dimensional vector of covariates. Furthermore, assume $F(Y_i|\boldsymbol{X})$ is parameterized in terms of a location parameter $\mu_i$, which depends on covariates $\boldsymbol{X}$ through

$$g(\mu_i) = \boldsymbol{X}'\boldsymbol{\beta}\,, \tag{2.4}$$

where $g(\cdot)$ is a suitable link function. For example, if $F(Y_i|\boldsymbol{X})$ is a normal distribution, then

$$g(\mu_i) = \mu_i = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p\,. \tag{2.5}$$

Therefore, the joint cdf is written as

$$F(y_1, \ldots, y_n) = \Phi_n(\epsilon_1, \ldots, \epsilon_n; \boldsymbol{R})\,, \tag{2.6}$$

where $\Phi_n(\cdot\,; \boldsymbol{R})$ is the $n$-dimensional multivariate standard normal cdf with correlation matrix $\boldsymbol{R}$ and $\epsilon_i = \Phi^{-1}\{F(Y_i|\boldsymbol{X})\}$ with $\Phi(\cdot)$ denoting the univariate standard normal cdf. Therefore, the Gaussian copula regression model assumes that

$$Y_i = h(\boldsymbol{x}, \epsilon_i) = F^{-1}\{\Phi(\epsilon_i)|\boldsymbol{x}\}\,, \quad \text{for} \quad i = 1, \ldots, n\,, \tag{2.7}$$

where $h(\cdot)$ is a function of the regressors $\boldsymbol{x}$ and unobserved error term $\epsilon_i$ and $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)'$ follows multivariate standard normal distribution with correlation matrix $\boldsymbol{R}$. Let $\boldsymbol{\Theta}$ be the vector of all model parameters, including the Gaussian copula correlation matrix $\boldsymbol{R}$ and marginal parameters. Therefore, the likelihood function for $\boldsymbol{\Theta}$ is

$$L(\boldsymbol{\Theta}; y_1, \ldots, y_n) = \phi_n(\epsilon_1, \ldots, \epsilon_n; \boldsymbol{R}) \prod_{i=1}^{n} \frac{f(y_i|\boldsymbol{x})}{\phi(\epsilon_i)}\,. \tag{2.8}$$

From here, parameters can be estimated by the EM algorithm as described in Section 2.3. A detailed explanation can be found in the work by Masarotto and Varin (2012).

In summary, the Gaussian copula regression model not only preserves the marginal univariate distributions but also has correlated errors through a multivariate normal distribution. However, the Gaussian copula regression model only deals with correlated observations from a single response variable and has yet to be extended to multivariate responses.

## 2.2 The finite mixture model

The application of finite mixture models is gaining popularity because it enables the utilization of conventional statistical methods to assess and evaluate clustering across a wide range of scenarios (Kosmidis and Karlis, 2016). Consider a $d$ dimension random vector $\boldsymbol{Y} = (Y_1, \ldots, Y_D)'$ from a parametric finite mixture model, then the probability density function is defined as

$$f(\boldsymbol{y}; \boldsymbol{\Theta}, \boldsymbol{\pi}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{y}; \boldsymbol{\Theta}_g)\,, \tag{2.9}$$

where $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)'$ are the mixing proportions such that $\pi_g \in (0, 1)$ for $g = 1, \ldots, G$ with $\sum_{g=1}^{G} \pi_g = 1$, $f_g(\boldsymbol{y}; \boldsymbol{\Theta}_g)$ is the $g$th component density, $\boldsymbol{\Theta} = (\boldsymbol{\Theta}_1', \ldots, \boldsymbol{\Theta}_G')'$ and each $\boldsymbol{\Theta}_g$ is the vector of parameters in the $g$th component. The model shown above is called a $G$-component finite mixture density (McLachlan and Chang, 2004).

In this finite mixture modelling framework, we view each component as a cluster such that each cluster $g$ has its own distribution and corresponding density $f_g$.

An example of a commonly used finite mixture model for continuous data is the Gaussian mixture model (GMM), which replaces component densities $f_g(\boldsymbol{y}; \boldsymbol{\Theta}_g)$ with the multivariate normal density. The $G$-component GMM has a density in the form

$$f(\boldsymbol{y}; \boldsymbol{\Theta}, \boldsymbol{\pi}) = \sum_{g=1}^{G} \pi_g \phi(\boldsymbol{y}; \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)\,, \tag{2.10}$$

where $\phi(\boldsymbol{y}; \boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)$ is the multivariate normal density of component $g$, with mean $\boldsymbol{\mu}_g$ and covariance matrix $\boldsymbol{\Sigma}_g$. This Gaussian mixture model has a total of $(G-1) + GD + GD(D+1)/2$ parameters, in which $G-1$ are the component mixing proportion $\pi_1, \ldots, \pi_G$, $GD$ are the mean vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_G$, and $GD(D+1)/2$ are used in covariance matrices $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_G$. The choice of the multivariate normal distribution is primarily based on convenience in estimation, as it has closed-form solutions shown in Section 2.3.3. In addition, constraints have been considered on the covariance matrix to reduce the number of free parameters. For example, parsimonious Gaussian mixture models (PGMM) were proposed by McNicholas and Murphy (2008), which is particularly effective for high dimensional situations. Compared with GMMs, the number of covariance parameters in the PGMM family grows linearly with data dimension, which can make it more suitable for high-dimensional data modelling.

However, the resulting clusters by using the Gaussian mixture models are constrained to possess an elliptical shape. To add more flexibility, other variations from the GMM have been proposed. For example, multivariate t distributions (Andrews and McNicholas, 2011), multivariate skew-normal distribution (Frühwirth-Schnatter and Pyne, 2010), multivariate normal inverse Gaussian distributions (Karlis and Santourian, 2009), etc. have been developed in order to deal with non-elliptically contoured distributions. From these studies, it is shown that using heavy-tailed and/or skewed distributions creates more appropriate models for skewed-type data compared to multivariate normal distributions. Even so, all of the above variations force the data to follow the same margins and fail to capture extreme tail dependence as illustrated by Kosmidis and Karlis (2016).

### 2.2.1 Mixtures of regression

Mixtures of regression, also known as finite mixtures of regression models, is a modelling technique that combines both regression analysis and finite mixture modelling. This method is used to capture heterogeneity in a dataset by assuming that the data is generated from a mixture of several underlying sub-populations, and each sub-population has its own regression relationship. Therefore, mixtures of regression models are a powerful tool, which can be used to identify latent groups or clusters. Bermúdez and Karlis (2022) proposed finite mixtures of regression models with bivariate discrete responses to analyze the counts of automobile insurance claims. However, their models for discrete response data have not been extended to higher dimensions yet due to the high computational cost. An existing well-known framework for mixtures of regression is the `flexmix` R package created by Leisch (2004). The `flexmix` package includes finite mixture models and latent class regression, which are modelled as follows:

$$\begin{aligned} f(y|\boldsymbol{x},\boldsymbol{\phi}) &= \sum_{g=1}^{G} \pi_g f_g(y|\boldsymbol{x},\boldsymbol{\theta}_g)\,, \\ \pi_g \in (0,1)\,, \quad & \sum_{g=1}^{G} \pi_g = 1\,. \end{aligned} \tag{2.11}$$

where $y$ is the dependent variable with conditional density $f$, $\boldsymbol{x}$ is a vector of independent variables, $\pi_g$ is the mixing proportion for component $g$, $\boldsymbol{\theta}_g$ is a vector of parameters used for the $g$th component density function $f$, and $\boldsymbol{\phi} = (\pi_1, ..., \pi_G, \boldsymbol{\theta}'_1, .., \boldsymbol{\theta}'_G)'$ is the vector of all parameters. For example, if each component density $f_g$ follows the normal distribution with mean $\boldsymbol{\beta}'_g\boldsymbol{x}$ and variance $\sigma_g^2$, then the model is a mixture of standard linear regressions. This model can be written as:

$$f(y|\boldsymbol{x},\boldsymbol{\phi}) = \sum_{g=1}^{G} \pi_g \phi(y|\mu_g(\boldsymbol{x}), \sigma_g^2)\,, \tag{2.12}$$

where $\mu_g(\boldsymbol{x}) = \boldsymbol{\beta}'_g\boldsymbol{x}$. Also, if the component density $f_g$ is a member of the exponential family, then the model is called the mixture of generalized linear models.

The model in 2.11 can be extended to multivariate responses *viz.*

$$\begin{aligned} f(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\phi}) &= \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_g) \\ &= \sum_{g=1}^{G} \pi_g \prod_{d=1}^{D} f_{dg}(y_d|\boldsymbol{x},\boldsymbol{\theta}_{dg}) \,, \end{aligned} \tag{2.13}$$

where $\boldsymbol{Y} = (Y_1, \ldots, Y_D)'$ is $D$-dimensional response variable and $f_{dg}$ is the density for $d$th response in $g$th component. However, even though the above model is extended to multivariate responses, it assumes that response variables are mutually independent. Thus, this model is restricted to correctly specifying the case of independent multivariate responses only, which is not true in most situations.

## 2.3 Estimation by EM algorithm

The Expectation-Maximization (EM) algorithm is widely used for finding maximum likelihood (ML) estimates of parameters in statistical models within the context of incomplete or missing data in cases where there are no analytical solutions. The EM algorithm is an iterative algorithm that alternates between two main steps: the Expectation (E) step and the Maximization (M) step until convergence, at which point the algorithm has found a local maximum of the likelihood function. The key idea behind the EM algorithm is to iteratively refine parameter estimates by alternating between estimating the missing or latent variables given the current model parameters (the E-step) and optimizing the model parameters given the estimated latent variables (the M-step).

Suppose the complete data set consists of $\boldsymbol{x} = (\boldsymbol{y}, \boldsymbol{z})$, where $\boldsymbol{y}$ is the observed data and $\boldsymbol{z}$ is the missing data. Denote the log-likelihood of the complete-data as $\ell_c(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{z})$ where $\boldsymbol{\theta}$ is the vector of unknown parameters for which we aim to find the ML estimates. In short, the steps of the EM algorithm can be summarized as

- E-step: evaluate $Q(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}^{(t)}) = E[\ell_c(\boldsymbol{\theta}; \boldsymbol{y}, \boldsymbol{z})|\boldsymbol{y}; \hat{\boldsymbol{\theta}}^{(t)}]\,.$
- M-step: update $\hat{\boldsymbol{\theta}}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}^{(t)})\,.$

Note that the function $Q(\boldsymbol{\theta}|\hat{\boldsymbol{\theta}}^{(t)})$ created in the E-step is a minorizing function of the observed log-likelihood at $\hat{\boldsymbol{\theta}}^{(t)}$, which is later maximized in the M-step (Dempster et al., 1977). One powerful property of the EM algorithm is the ascent property, written as

$$\ell(\hat{\boldsymbol{\theta}}^{(t+1)}|\boldsymbol{y}) = \log p(\boldsymbol{y};\hat{\boldsymbol{\theta}}^{(t+1)}) \geq \log p(\boldsymbol{y};\hat{\boldsymbol{\theta}}^{(t)}) = \ell(\hat{\boldsymbol{\theta}}^{(t)}|\boldsymbol{y})\,, \tag{2.14}$$

i.e., the observed likelihood value is non-decreasing for each iteration of the EM. This ascent property of the EM algorithm helps ensure that the algorithm converges to a point with zero gradient with respect to the parameters. However, this does not guarantee convergence to the global maximum of the likelihood function, as the EM algorithm may get stuck at a critical point, either local maxima or saddle points. Therefore, the results of the EM algorithm are sensitive to the choice of initial values. To overcome this issue, multiple initializations should be applied, which will be discussed in the next section.

### 2.3.1 Initial values

In this thesis, the EM algorithm will be initialized at several randomly chosen starting points to mitigate the sensitivity to initializations. The following is an example initialization of the algorithm.

The starting values for component variables, $z_{ig}^{(0)}$, can be obtained by using a hard-partitioning distance-based algorithm like K-means. Then, the $g$th mixing proportion $\pi_g$ can be initialized as

$$\pi_g^{(0)} = \frac{\sum_{i=1}^{n} \hat{z}_{ig}^{(0)}}{n}\,, \tag{2.15}$$

where $n$ is the number of observations. Consequently, analytical solutions or a numerical optimization method (`optim` function in R) can be used to obtain initializations for each group of parameters in $g$th component, so that

$$\boldsymbol{\theta}_g^{(0)} = \text{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} z_{ig}^{(0)} \ell_c(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y})\,. \tag{2.16}$$

After initializing all parameters in the model, we enter the 1st iteration of the EM algorithm.

### 2.3.2 Convergence Criterion

A common stopping criterion for the EM algorithm is a lack of progress of the log-likelihood. I.e., the algorithm is stopped when

$$\ell^{(t+1)} - \ell^{(t)} < \epsilon \tag{2.17}$$

for some small $\epsilon$, where $\ell^{(t)}$ is the observed log-likelihood value at iteration $t$. Note here, that the absolute value of the difference is not needed due to the ascent property. So, the EM algorithm travels around the parameter space, evaluates the log-likelihood value at the current point and then compares it with the previous point. However, the progression of likelihood values is not a guaranteed outcome. For example, if the EM path temporarily moves to a location where the log-likelihood function is approximately flat but increases again later, the EM algorithm may stop early and underestimate the value of maximum log-likelihood using this criterion (McNicholas et al., 2010).

In order to overcome this problem, a convergence criterion based on Aitken's acceleration (Aitken, 1926) is used. Aitken's acceleration is defined at iteration $t$ by

$$a^{(t)} = \frac{\ell^{(t+1)} - \ell^{(t)}}{\ell^{(t)} - \ell^{(t-1)}}\,, \tag{2.18}$$

where $\ell^{(t+1)}$, $\ell^{(t)}$, $\ell^{(t-1)}$ are the log-likelihood values from iterations $t+1$, $t$, and $t-1$ respectively. So at iteration $t+1$, we calculate the log-likelihood $\ell^{(t+1)}$ and then calculate the asymptotic estimate of the log-likelihood

$$\ell_{\infty}^{(t+1)} = \ell^{(t)} + \frac{1}{1-a^{(t)}}(\ell^{(t+1)} - \ell^{(t)})\,. \tag{2.19}$$

If the stopping criterion $0 < \ell_{\infty}^{(t)} - \ell^{(t)} < \epsilon$ is satisfied, the algorithm will stop (McNicholas et al., 2010). Convergence criteria based on Aitken's acceleration will take more iterations to stop but offer a more rigorous result.

### 2.3.3 Example: The Gaussian mixture model

In this section, we will describe the use of the EM algorithm to estimate the popular Gaussian mixture model in 2.10. First, we define latent variable $z_i = (z_{i1}, \ldots, z_{iG})$, such that $z_{ig}$ denotes the component membership, where $z_{ig} = 1$ if observation $i$

belongs to component $g$, and $z_{ig} = 0$ otherwise. Therefore, the complete-data is $(\boldsymbol{y}, \boldsymbol{z})$ with corresponding complete likelihood

$$L_c(\boldsymbol{\phi}|\boldsymbol{y}, \boldsymbol{z}) = \prod_{i=1}^{n} \prod_{g=1}^{G} [\pi_g \phi_g(\boldsymbol{y}|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)]^{z_{ig}}\,, \tag{2.20}$$

where $\boldsymbol{\phi} = (\pi_1, \ldots, \pi_G, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_G, \boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_G)$ is all the model parameters. Consequently, the complete-data log-likelihood is given by

$$\ell_c(\boldsymbol{\phi}|\boldsymbol{y}) = \sum_{i=1}^{n} \sum_{g=1}^{G} z_{ig}[\log \pi_g + \log \phi_g(\boldsymbol{y}|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)]\,. \tag{2.21}$$

- In the E-step, the algorithm calculates the expected values of the latent variables, given the current estimates of the model parameters. At iteration $t+1$, $\hat{z}_{ig}^{(t+1)}$ is calculated as

$$\begin{aligned}
\hat{z}_{ig}^{(t+1)} &= E_{z|y}[z_{ig}|\boldsymbol{y_i}, \hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)}] \\
&= P(z_{ig} = 1|\boldsymbol{y_i}, \hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)}) \\
&= \frac{P(z_{ig} = 1, \boldsymbol{y_i}|\hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})}{P(\boldsymbol{y_i}|\hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})} \\
&= \frac{P(\boldsymbol{y_i}|z_{ig} = 1, \hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})P(z_{ig} = 1|\hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})}{P(\boldsymbol{y_i}|\hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})} \\
\hat{z}_{ig}^{(t+1)} &= \frac{\phi_g(\boldsymbol{y_i}|\hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})\hat{\pi}_g^{(t)}}{\sum_{g=1}^{G} \hat{\pi}_g^{(t)} \phi_g(\boldsymbol{y_i}|\hat{\boldsymbol{\mu}}_g^{(t)}, \hat{\boldsymbol{\Sigma}}_g^{(t)})}\,. \qquad (2.22)
\end{aligned}$$

  It follows that the conditional expectation of the complete-data log-likelihood is

$$\begin{aligned}
Q(\boldsymbol{\phi}|\hat{\boldsymbol{\phi}}^{(t)}) &= \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z}_{ig}^{(t+1)}[\log \pi_g + \log \phi_g(\boldsymbol{y}|\boldsymbol{\mu}_g, \boldsymbol{\Sigma}_g)] \\
&= \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z}_{ig}^{(t+1)}[\log \pi_g - \frac{p}{2}\log 2\pi - \frac{1}{2}\log|\boldsymbol{\Sigma}_g| - \frac{1}{2}\mathrm{tr}\{(\boldsymbol{y}_i - \boldsymbol{\mu}_g)(\boldsymbol{y}_i - \boldsymbol{\mu}_g)'\boldsymbol{\Sigma}_g^{-1}\}] \\
&= \sum_{g=1}^{G} n_g \log \pi_g - \frac{np}{2}\log 2\pi - \frac{n_g}{2}\log|\boldsymbol{\Sigma}_g| - \frac{n_g}{2}\mathrm{tr}\{\boldsymbol{S}_g\boldsymbol{\Sigma}_g^{-1}\}\,, \qquad (2.23)
\end{aligned}$$

  where $p$ is the dimension of the data vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, $n_g = \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)}$, and

$$\boldsymbol{S}_g = \frac{1}{n_g}\sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)}(\boldsymbol{y}_i - \boldsymbol{\mu}_g)(\boldsymbol{y}_i - \boldsymbol{\mu}_g)'\,.$$

- In the M-step, the algorithm updates the estimates of the model parameters by maximizing the expected complete likelihood from the E-step. Maximizing $Q(\boldsymbol{\phi}|\hat{\boldsymbol{\phi}}^{(t)})$ with respect to $\pi_g, \boldsymbol{\mu}_g$, and $\boldsymbol{\Sigma}_g$ gives us

$$\begin{aligned}
\hat{\pi}_g^{(t+1)} &= \frac{n_g}{n}\,, \\
\hat{\boldsymbol{\mu}}_g^{(t+1)} &= \frac{1}{n_g}\sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)}\boldsymbol{y}_i\,, \\
\hat{\boldsymbol{\Sigma}}_g^{(t+1)} &= \frac{1}{n_g}\sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)}(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_g^{(t+1)})(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_g^{(t+1)})'\,.
\end{aligned}$$

As described in detail above, the EM algorithm for the $G$-component Gaussian mixture model alternates between the E and M-steps until convergence. A summary of this EM algorithm is included in Algorithm 1.

**Algorithm 1** EM Algorithm for Gaussian mixture model.

```
Initialize $\hat{z}_{ig}^{(0)}, \hat{\pi}_g^{(0)}, \hat{\boldsymbol{\mu}}_g^{(0)}$, and $\hat{\boldsymbol{\Sigma}}_g^{(0)}$
while convergence criterion is not met do
    E-step: update $\hat{z}_{ig}^{(t+1)} = \frac{\hat{\pi}_g^{(t)}\phi_g(\boldsymbol{y};\hat{\boldsymbol{\mu}}_g^{(t)},\hat{\boldsymbol{\Sigma}}_g^{(t)})}{\sum_{g=1}^{G}\hat{\pi}_g^{(t)}\phi_g(\boldsymbol{y};\hat{\boldsymbol{\mu}}_g^{(t)},\hat{\boldsymbol{\Sigma}}_g^{(t)})}$
    M-step: update $\hat{\pi}_g^{(t+1)} = \frac{n_g}{n}$
            update $\hat{\boldsymbol{\mu}}_g^{(t+1)} = \frac{1}{n_g}\sum_{i=1}^{n}\hat{z}_{ig}^{(t+1)}\boldsymbol{y}_i$
            update $\hat{\boldsymbol{\Sigma}}_g^{(t+1)} = \frac{1}{n_g}\sum_{i=1}^{n}\hat{z}_{ig}^{(t+1)}(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_g^{(t+1)})(\boldsymbol{y}_i - \hat{\boldsymbol{\mu}}_g^{(t+1)})'$
    check convergence criterion
end while
```

## 2.4 Performance Assessment

After clustering, the quality of the results should be assessed. Traditionally, if the component memberships are known, the adjusted Rand index (ARI) and maximum *a posteriori* (MAP) table are used. The ARI assesses the similarity between two data partitions by comparing the agreement in the assignments of data points into partitions. It is a correction of the Rand Index (RI) to account for chance agreement.

In our setting, we are using ARI to measure the performance of clustering results with knowledge of the true classes. Consider two partitions $U$ and $V$, where we suppose $U$ is the partition formed by the true classes and $V$ is the partition formed by the clustering result. There are four categories of data point pairs:

A: Pairs of objects that are in the same class in $U$ and in the same cluster in $V$.

B: Pairs of objects that are in the same class in $U$ but different clusters in $V$.

C: Pairs of objects that are in the same cluster in $V$ but different classes in $U$.

D: Pairs of objects that are in different classes in $U$ and different clusters in $V$.

Therefore, the quantities $A$ and $D$ can be interpreted as agreements, and the quantities $B$ and $C$ can be interpreted as disagreements. Then, the RI is calculated by

$$\text{RI} = \frac{A+D}{A+B+C+D} = \frac{A+D}{N}\,, \tag{2.24}$$

where $N = A+B+C+D$ is the total number of pairs.

To correct for chance agreements, the ARI was proposed by Hubert and Arabie (1985) as

$$\text{ARI} = \frac{\text{RI} - \text{Expected RI}}{\text{Maximum RI} - \text{Expected RI}}\,, \tag{2.25}$$

where the Maximum RI is 1 and the Expected RI is calculated as

$$\text{Expected RI} = \frac{(A+B)(A+C)+(C+D)(B+D)}{N^2}\,. \tag{2.26}$$

Therefore, the ARI is calculated as

$$\begin{aligned}\text{ARI} &= \frac{\text{RI} - \text{Expected RI}}{\text{Maximum RI} - \text{Expected RI}} \\ &= \frac{\frac{A+D}{N} - \frac{(A+B)(A+C)+(C+D)(B+D)}{N^2}}{1 - \frac{(A+B)(A+C)+(C+D)(B+D)}{N^2}} \\ &= \frac{N(A+D) - [(A+B)(A+C)+(C+D)(B+D)]}{N^2 - [(A+B)(A+C)+(C+D)(B+D)]}\,. \end{aligned} \tag{2.27}$$

The ARI has a maximum value of 1, with higher values indicating better agreement between partitions $U$ and $V$. An ARI value of 1 indicates a perfect clustering result, i.e., a perfect agreement between $U$ and $V$, and the expected value of the ARI under random cluster assignment is 0. If a negative ARI occurs, it indicates that the clustering is even worse than random assignments. Note that the ARI has no well-defined lower bound (McNicholas, 2016; Hubert and Arabie, 1985).

Table 2.3: A sample MAP table, notations here follows the work by Hubert and Arabie (1985).

| $U \setminus V$ | $v_1$ | $v_2$ | $\cdots$ | $v_C$ | *Sums* |
|---|---|---|---|---|---|
| $u_1$ | $n_{11}$ | $n_{12}$ | $\cdots$ | $n_{1C}$ | $n_{1\cdot}$ |
| $u_2$ | $n_{21}$ | $n_{22}$ | $\cdots$ | $n_{2C}$ | $n_{2\cdot}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | $\vdots$ |
| $u_R$ | $n_{R1}$ | $n_{R2}$ | $\cdots$ | $n_{RC}$ | $n_{R\cdot}$ |
| *Sums* | $n_{\cdot 1}$ | $n_{\cdot 2}$ | $\cdots$ | $n_{\cdot C}$ | $n_{\cdot\cdot} = n$ |

As for the MAP table, it directly gives the information for the clustering result. Once the data with a total of $n$ observations has been clustered, we can generate a MAP table. A sample MAP table is shown in Table 2.3. Therefore, the values of $A, B, C$, and $D$ can be calculated as

$$\begin{aligned} A &= \sum_{i,j} \binom{n_{ij}}{2}, \\ B &= \sum_{i} \binom{n_{i\cdot}}{2} - \sum_{i,j} \binom{n_{ij}}{2}, \\ C &= \sum_{j} \binom{n_{\cdot j}}{2} - \sum_{i,j} \binom{n_{ij}}{2}, \\ D &= N - A - B - C = \binom{n}{2} - A - B - C \,. \end{aligned}$$

Therefore, the MAP table and ARI value can be easily calculated after clustering and used as a direct indicator for the clustering performance. There are alternative measures available for evaluating clustering performance, such as the Silhouette coefficient (Rousseeuw, 1987), Calinski-Harabasz Index (Caliński and Harabasz, 1974) and Davies-Bouldin Index (Davies and Bouldin, 1979). These measures can be used when the true classes are unknown, but will not be required for this thesis work.

## 2.5 Summary

In this chapter, we presented the foundational knowledge of copulas, along with bivariate examples and their usage in regression analysis. Also, finite mixture models with an extension to finite mixtures of regression models were introduced. The conventional computational technique for handling finite mixture models, namely the EM algorithm, was provided with an example of a Gaussian mixture model. In the end of this chapter, we discussed the methods for assessing clustering performance. In the next chapter, the proposed methodology is introduced.

# Chapter 3

# Methodology

In this chapter, we propose a copula-based mixture of regression models to effectively handle the multivariate response data by combining the strengths of both mixtures of regression models and copula regression. To address the computational challenges in the maximization step, we replace the originally considered EM algorithm with the ECM algorithm. Furthermore in Section 3.2.1, we provide comprehensive computational details for ML estimation and explore model variations in Section 3.3.

## 3.1 Proposed model

The proposed model in this thesis extends the finite mixture model from Section 2.2 to regression context, such that each marginal distribution can be defined by conditioning on a vector of covariates. Let $\boldsymbol{Y} = (Y_1, \dots, Y_D)'$, a random vector of dimension $D \geq 1$, be the response variables of interest and $\boldsymbol{X} = (X_1, \dots, X_P)'$, where $P \geq 1$, be the vector of covariates. Suppose $\boldsymbol{Y}|\boldsymbol{X}$ follows a finite mixture model of $G$ components, thus, the joint pdf of $\boldsymbol{Y}|\boldsymbol{X}$ is defined as

$$f(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Phi}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Theta}_g) , \tag{3.1}$$

where $\boldsymbol{\Phi} = (\pi_1, \dots, \pi_G; \boldsymbol{\Theta}_1, \dots, \boldsymbol{\Theta}_G)'$ are all parameters in the model, and $\pi_g \in (0,1)$ for $g = 1, \dots, G$ are the mixing components defined such that $\sum_{g=1}^{G} \pi_g = 1$. Each component density $f_g(\boldsymbol{y}|\boldsymbol{x})$ can be expressed in a copula framework using (2.3) *viz.*

$$f_g(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\Theta}_g) = c_g\{F_{1g}(y_1|\boldsymbol{x}), \dots, F_{Dg}(y_D|\boldsymbol{x}); \boldsymbol{\theta}_g\} f_{1g}(y_1|\boldsymbol{x}; \boldsymbol{\beta}_{1g}) \cdots f_{Dg}(y_D|\boldsymbol{x}; \boldsymbol{\beta}_{Dg}) , \tag{3.2}$$

where $\boldsymbol{\Theta}_g$ is the vector of all parameters involved in the $g$th component, i.e., $\boldsymbol{\Theta}_g = (\boldsymbol{\theta}'_g, \boldsymbol{\beta}'_{1g}, \dots, \boldsymbol{\beta}'_{Dg})'$. Note in regression context, we assume each response variable $Y_d$ given the covariates $\boldsymbol{x}$ in $g$th component has marginal density $F_d(y_d|\boldsymbol{x})$ parameterized in terms of a parameter $\mu_{dg}$, where $\mu_{dg}$ is a regression using covariates $\boldsymbol{X}$ through a suitable link function $g(\cdot)$, written as $g(\mu_{dg}) = \boldsymbol{X}'\boldsymbol{\beta}_{dg}$. For example, the marginal models are specified as

$$\begin{aligned}
&Y_{dg}|\boldsymbol{X} \sim N(\mu_{dg}, \sigma_{dg}) \text{ with } \mu_{dg} = \boldsymbol{X}'\boldsymbol{\beta}_{dg}\,, \\
&Y_{dg}|\boldsymbol{X} \sim \text{Gamma}(\text{Scale} = \mu_{dg}, \text{Shape} = \sigma_{dg}) \text{ with } \mu_{dg} = e^{-\boldsymbol{X}'\boldsymbol{\beta}_{dg}}\,.
\end{aligned}$$

Therefore, the complete joint distribution of $\boldsymbol{Y}|\boldsymbol{X}$ in (3.1) is

$$f(\boldsymbol{y}|\boldsymbol{x}) = \sum_{g=1}^{G} \pi_g f_g(\boldsymbol{y}|\boldsymbol{x}) = \sum_{g=1}^{G} \pi_g c_g\{F_{1g}(y_1|\boldsymbol{x}), \ldots, F_{Dg}(y_D|\boldsymbol{x})\} f_{1g}(y_1|\boldsymbol{x}) \cdots f_{Dg}(y_D|\boldsymbol{x})\,. \tag{3.3}$$

The goal of this model is to cluster the observations by solving parameters in this joint distribution. To achieve this goal, the algorithm for ML estimation described in the following section will be used.

## 3.2 ML estimation via ECM algorithm

The parameters in this proposed model include the mixing proportion parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_G)'$, the copula parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1', \ldots, \boldsymbol{\theta}_G')'$, and the marginal parameters $\boldsymbol{\beta} = (\boldsymbol{\beta}_1', \ldots, \boldsymbol{\beta}_G')'$. Note that each $\boldsymbol{\beta}_g = (\boldsymbol{\beta}_{1g}', \ldots, \boldsymbol{\beta}_{Dg}')'$ for $g = 1, \ldots, G$ and each $\boldsymbol{\beta}_{dg} = (\beta_{dg0}, \beta_{dg1}, \ldots, \beta_{dgP})$ for $d = 1, \ldots, D$. There are $P + 1$ parameters in each regression coefficient vector $\boldsymbol{\beta}_{dg}$, as we have $P$ covariates $\boldsymbol{X} = (X_1, \ldots, X_P)'$. To clarify the notation, a summary of all parameters is listed in the Tree Diagram 3.1. The initial goal is to find the maximum likelihood (ML) estimates for all parameters using the EM algorithm as described in Section 2.3. However, due to the high computational cost in the maximization step, a variant of the EM algorithm called the ECM algorithm is used instead (Meng and Rubin, 1993).

### 3.2.1 Computational details

Suppose we have $n$ observations of $\boldsymbol{Y} = (Y_1, \ldots, Y_D)'$, where the $i$th observation is labelled as $\boldsymbol{Y_i} = (Y_{1i}, \ldots, Y_{Di})'$. Using the density as written in Equation (3.3), the

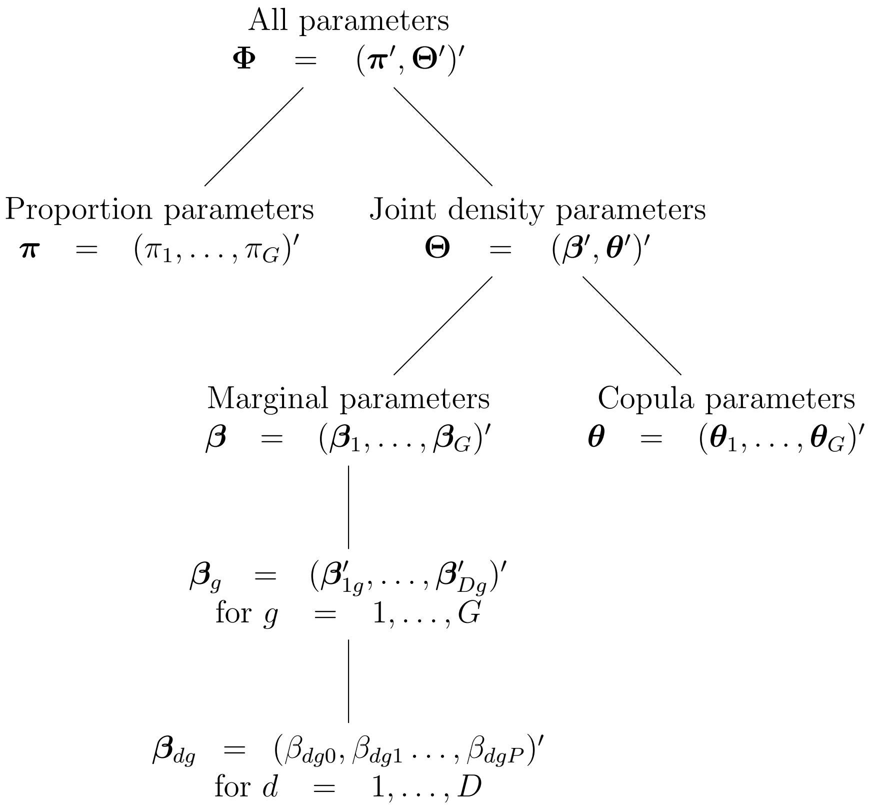
All parameters
$\mathbf{\Phi} = (\boldsymbol{\pi}', \mathbf{\Theta}')'$
Proportion parameters
$\boldsymbol{\pi} = (\pi_1, \dots, \pi_G)'$
Joint density parameters
$\mathbf{\Theta} = (\boldsymbol{\beta}', \boldsymbol{\theta}')'$
Marginal parameters
$\boldsymbol{\beta} = (\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_G)'$
Copula parameters
$\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_G)'$
$\boldsymbol{\beta}_g = (\boldsymbol{\beta}'_{1g}, \dots, \boldsymbol{\beta}'_{Dg})'$
for $g = 1, \dots, G$
$\boldsymbol{\beta}_{dg} = (\beta_{dg0}, \beta_{dg1} \dots, \beta_{dgP})'$
for $d = 1, \dots, D$

Figure 3.1: Notation for all parameters involved in the proposed model.

likelihood function of the observed data is

$$\begin{aligned} L(\mathbf{\Phi}|\boldsymbol{x}, \boldsymbol{y}) &= \prod_{i=1}^{n} f(\boldsymbol{y_i}|\boldsymbol{x_i}; \mathbf{\Phi}) \\ &= \prod_{i=1}^{n} \sum_{g=1}^{G} \pi_g c_g\{F_{1g}(\cdot; \boldsymbol{\beta}_{1g}), \dots, F_{Dg}(\cdot; \boldsymbol{\beta}_{Dg}); \boldsymbol{\theta}_g\}\ f_{1g}(\cdot; \boldsymbol{\beta}_{1g}) \cdots f_{Dg}(\cdot; \boldsymbol{\beta}_{Dg}) \,. \end{aligned} \tag{3.4}$$

We introduce missing data by defining latent variables $\boldsymbol{z_i} = (z_{i1}, \dots, z_{iG})$, such that $z_{ig}$ denotes the $g$th component membership of observation $i$. So, $z_{ig} = 1$ if the $i$th observation belongs to the $g$th component and $z_{ig} = 0$ otherwise. Thus, the likelihood function of the complete data (i.e. the observed and missing data) can be written as:

$$L_c(\mathbf{\Phi}|\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \prod_{i=1}^{n} \prod_{g=1}^{G} \left[\pi_g c_g\{F_{1g}(\cdot; \boldsymbol{\beta}_{1g}), \dots, F_{Dg}(\cdot; \boldsymbol{\beta}_{Dg}); \boldsymbol{\theta}_g\}\ f_{1g}(\cdot; \boldsymbol{\beta}_{1g}) \cdots f_{Dg}(\cdot; \boldsymbol{\beta}_{Dg})\right]^{z_{ig}} \,. \tag{3.5}$$

Therefore, the complete log-likelihood function is

$$\begin{aligned}
&\ell_c(\boldsymbol{\Phi}|\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\\
&\quad=\sum_{i=1}^{n}\sum_{g=1}^{G} z_{ig}\log\left\{\pi_g c_g\{F_{1g}(\cdot;\boldsymbol{\beta}_{1g}),\ldots,F_{Dg}(\cdot;\boldsymbol{\beta}_{Dg});\boldsymbol{\theta}_g\}\ f_{1g}(\cdot;\boldsymbol{\beta}_{1g})\cdots f_{Dg}(\cdot;\boldsymbol{\beta}_{Dg})\right\}\\
&\quad=\sum_{i=1}^{n}\sum_{g=1}^{G} z_{ig}\left\{\log\pi_g+\log c_g\{F_{1g},\ldots,F_{Dg};\boldsymbol{\theta}_g\}+\log f_{1g}(\cdot;\boldsymbol{\beta}_{1g})+\cdots+\log f_{Dg}(\cdot;\boldsymbol{\beta}_{Dg})\right\}\\
&\quad=\sum_{i=1}^{n}\sum_{g=1}^{G} z_{ig}\left\{\log\pi_g+\log c_g\{F_{1g},\ldots,F_{Dg};\boldsymbol{\theta}_g\}+\sum_{d=1}^{D}\log f_{dg}(\cdot;\boldsymbol{\beta}_{dg})\right\}. \qquad (3.6)
\end{aligned}$$

To maximize the complete log-likelihood function in (3.6), an ECM algorithm will be performed. At the $t$th iteration of this algorithm:

- **E-step**: The expected value of the complete log-likelihood function is updated, and the $z_{ig}$s are replaced by their expected values conditional on the observed data and $\hat{\boldsymbol{\Phi}}^{(t)}$ from the previous iteration. Therefore $\hat{z}_{ig}^{(t+1)}$ is calculated as

$$\begin{aligned}
\hat{z}_{ig}^{(t+1)} &= E_{z|x,y}[z_{ig}|\boldsymbol{x_i},\boldsymbol{y_i},\hat{\boldsymbol{\Phi}}^{(t)}]\\
&= P(z_{ig}=1|\boldsymbol{x_i},\boldsymbol{y_i},\hat{\boldsymbol{\Phi}}^{(t)})\\
&= \frac{P(z_{ig}=1,\boldsymbol{y_i}|\boldsymbol{x_i},\hat{\boldsymbol{\Phi}}^{(t)})}{P(\boldsymbol{y_i}|\boldsymbol{x_i},\hat{\boldsymbol{\Phi}}^{(t)})}\\
&= \frac{P(\boldsymbol{y_i}|z_{ig}=1,\boldsymbol{x_i},\hat{\boldsymbol{\Phi}}^{(t)})P(z_{ig}=1|\hat{\boldsymbol{\Phi}}^{(t)})}{P(\boldsymbol{y_i}|\boldsymbol{x_i},\hat{\boldsymbol{\Phi}}^{(t)})}\\
&= \frac{f_g(\boldsymbol{y_i}|\boldsymbol{x_i},\hat{\boldsymbol{\Phi}}_g^{(t)})\hat{\pi}_g^{(t)}}{\sum_{g=1}^{G}\hat{\pi}_g^{(t)}f_g(\boldsymbol{y_i}|\boldsymbol{x_i},\hat{\boldsymbol{\Phi}}_g^{(t)})}\\
&= \frac{\hat{\pi}_g^{(t)}c_g\{F_{1g}(\cdot;\hat{\boldsymbol{\beta}}_{1g}^{(t)}),\ldots,F_{Dg}(\cdot;\hat{\boldsymbol{\beta}}_{Dg}^{(t)});\hat{\boldsymbol{\theta}}_g^{(t)}\}\ f_{1g}(\cdot;\hat{\boldsymbol{\beta}}_{1g}^{(t)})\cdots f_{Dg}(\cdot;\hat{\boldsymbol{\beta}}_{Dg}^{(t)})}{\sum_{g=1}^{G}\hat{\pi}_g^{(t)}c_g\{F_{1g}(\cdot;\hat{\boldsymbol{\beta}}_{1g}^{(t)}),\ldots,F_{Dg}(\cdot;\hat{\boldsymbol{\beta}}_{Dg}^{(t)});\hat{\boldsymbol{\theta}}_g^{(t)}\}\ f_{1g}(\cdot;\hat{\boldsymbol{\beta}}_{1g}^{(t)})\cdots f_{Dg}(\cdot;\hat{\boldsymbol{\beta}}_{Dg}^{(t)})}.
\end{aligned}$$
$$(3.7)$$

  It follows that the conditional expectation of the complete log-likelihood is

$$Q(\boldsymbol{\Phi}|\hat{\boldsymbol{\Phi}}^{(t)})=\sum_{i=1}^{n}\sum_{g=1}^{G}\hat{z}_{ig}^{(t+1)}\left\{\log\pi_g+\log c_g\{F_{1g},\ldots,F_{Dg};\boldsymbol{\theta}_g\}+\sum_{d=1}^{D}\log f_{dg}(\cdot;\boldsymbol{\beta}_{dg})\right\}. \qquad (3.8)$$

- **M-step**: Maximize $Q(\boldsymbol{\Phi}|\hat{\boldsymbol{\Phi}}^{(t)})$ with respect to $\pi_g$, $\boldsymbol{\theta}_g$, and $\boldsymbol{\beta}_g$. In the M-step, we note that the $\hat{z}_{ig}$s are fixed and the $Q(\boldsymbol{\Phi}|\hat{\boldsymbol{\Phi}}^{(t)})$ function can be decomposed into two parts:

$$\sum_{i=1}^{n}\sum_{g=1}^{G}\hat{z}_{ig}^{(t+1)}\log\pi_g \tag{3.9}$$

and

$$\sum_{i=1}^{n}\sum_{g=1}^{G}\hat{z}_{ig}^{(t+1)}\left\{\log c_g\{F_{1g},\ldots,F_{Dg};\boldsymbol{\theta}_g\}+\sum_{d=1}^{D}\log f_{dg}(y_{di}|\boldsymbol{x_i};\boldsymbol{\beta}_{dg})\right\}. \tag{3.10}$$

The first part (3.9) includes proportion parameters $\boldsymbol{\pi}$ and the second part (3.10) includes all other parameters involved in the model, thus we can maximize the two parts separately as follows:

1. In M-step 1, the Equation (3.9) is maximized. In other words, we solve

$$\hat{\pi}_g^{(t+1)} = \text{argmax}_{\pi_g}\sum_{i=1}^{n}\sum_{g=1}^{G}\hat{z}_{ig}^{(t+1)}\log\pi_g$$

subject to the constraint $\sum_{g=1}^{G}\pi_g = 1$. To perform constrained optimization, Lagrange multipliers are used:

$$\ell(\boldsymbol{\Phi},\lambda)=\sum_{i=1}^{n}\sum_{g=1}^{G}\hat{z}_{ig}^{(t+1)}\log\pi_g-\lambda\left(\sum_{g=1}^{G}\pi_g-1\right)+h(\boldsymbol{\Phi}). \tag{3.11}$$

Taking derivative of $\ell(\boldsymbol{\Phi},\lambda)$ with respect with $\pi_g$ and setting it equal to zero gives

$$\frac{\partial\ell}{\partial\pi_g}=\left(\sum_{i=1}^{n}\hat{z}_{ig}^{(t+1)}/\pi_g\right)-\lambda=0 \tag{3.12}$$

and

$$\frac{\partial\ell}{\partial\lambda}=\sum_{g=1}^{G}\pi_g-1=0. \tag{3.13}$$

Therefore, from Equation (3.12), we have

$$\left.\frac{\partial\ell}{\partial\pi_g}\right|_{\hat{\pi}_g^{(t+1)}}=\left(\sum_{i=1}^{n}\hat{z}_{ig}^{(t+1)}/\hat{\pi}_g^{(t+1)}\right)-\lambda=0 \implies \hat{\pi}_g^{(t+1)}=\frac{1}{\lambda}\sum_{i=1}^{n}\hat{z}_{ig}^{(t+1)}, \tag{3.14}$$

and from Equation (3.13), we have

$$\left.\frac{\partial\ell}{\partial\lambda}\right|_{\hat{\pi}_g^{(t+1)}}=\sum_{g=1}^{G}\hat{\pi}_g^{(t+1)}-1=0 \implies \sum_{g=1}^{G}\hat{\pi}_g^{(t+1)}=1. \tag{3.15}$$

Therefore, substituting (3.14) into (3.15) gives

$$\sum_{g=1}^{G} \frac{1}{\lambda} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} = 1 \implies \lambda = \sum_{g=1}^{G} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} = n \, . \tag{3.16}$$

Thus

$$\hat{\pi}_g^{(t+1)} = \frac{1}{\lambda} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)}}{n} \, . \tag{3.17}$$

2. In M-step 2, the Equation (3.10) is maximized. In other words, we wish to solve

$$\hat{\boldsymbol{\Theta}}_g^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\Theta}_g} \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z}_{ig}^{(t+1)} \left[ \log c_g\{F_{1g}, \ldots, F_{Dg}; \boldsymbol{\theta}_g\} + \sum_{d=1}^{D} \log f_{dg}(\cdot; \boldsymbol{\beta}_{dg}) \right] , \tag{3.18}$$

where $\boldsymbol{\Theta}_g = (\boldsymbol{\theta}_g, \boldsymbol{\beta}'_{1g}, \ldots, \boldsymbol{\beta}'_{Dg})'$. However, due to its high computational complexity, solving this M-step 2 in a joint optimization is not ideal. Instead, the ECM algorithm proposed by Meng and Rubin (1993) is employed as an alternative. In this approach, the M-step 2 is substituted with two CM steps for enhanced efficiency.

- In CM-step 1, the marginal parameters $\boldsymbol{\beta} = (\boldsymbol{\beta}'_1, \ldots \boldsymbol{\beta}'_G)'$ are first estimated given the current value of the copula parameter $\hat{\boldsymbol{\theta}}^{(t)}$, i.e.,

$$\hat{\boldsymbol{\beta}}^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\beta}} \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z}_{ig}^{(t+1)} \left[ \log c_g\{F_{1g}, \ldots, F_{Dg}; \hat{\boldsymbol{\theta}}_g^{(t)}\} + \sum_{d=1}^{D} \log f_{dg}(\cdot; \boldsymbol{\beta}_{dg}) \right] . \tag{3.19}$$

We can break down this maximization task into $G$ independent maximizations across components. Therefore for $g$th component:

$$\hat{\boldsymbol{\beta}}_g^{(t+1)} = \operatorname{argmax}_{\boldsymbol{\beta}_g} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} \left[ \log c_g\{F_{1g}, \ldots, F_{Dg}; \hat{\boldsymbol{\theta}}_g^{(t)}\} + \sum_{d=1}^{D} \log f_{dg}(\cdot; \boldsymbol{\beta}_{dg}) \right] , \tag{3.20}$$

where $\hat{\boldsymbol{\beta}}_g^{(t+1)} = (\hat{\boldsymbol{\beta}}_{1g}^{(t+1)}, \ldots, \hat{\boldsymbol{\beta}}_{Dg}^{(t+1)})$. Therefore, we estimate $D(P+1)$ parameters in each of $G$ independent maximizations, as we have $P$ covariates. Consequently, there are $GD(P+1)$ parameters get estimated in CM-step 1. Note that if there are additional parameters involved in the marginal distribution but not related to the regression, such as the standard deviation $\sigma_{dg}$ in the normal marginal density, we will simultaneously estimate them along with Equation (3.20). In such cases, the total number of parameters estimated in CM-step 1 will be $GD(P+2)$.

- In CM-step 2, we estimate the copula parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}'_1, \dots \boldsymbol{\theta}'_G)'$ given the updated values of marginal parameters $\hat{\boldsymbol{\beta}}^{(t+1)}$, calculated in CM-step 1. Since the marginal part $\sum_{d=1}^{D} \log f_d(\cdot; \boldsymbol{\beta}_{dg})$ in the Equation (3.20) does not incorporate copula parameter $\boldsymbol{\theta}$, the conditional maximization step can be simplified to:

$$\hat{\boldsymbol{\theta}}^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \sum_{g=1}^{G} \hat{z}_{ig}^{(t+1)} \log c_g\{F_{1g}(\cdot; \hat{\boldsymbol{\beta}}_{1g}^{(t+1)}), \dots, F_{Dg}(\cdot; \hat{\boldsymbol{\beta}}_{Dg}^{(t+1)}); \boldsymbol{\theta}_g\}\,. \tag{3.21}$$

  Similarly, as in CM-step 1, this step can be broken down into $G$ independent maximizations across components:

$$\hat{\boldsymbol{\theta}}_g^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}_g} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} \log c_g\{F_{1g}(\cdot; \hat{\boldsymbol{\beta}}_{1g}^{(t+1)}), \dots, F_{Dg}(\cdot; \hat{\boldsymbol{\beta}}_{Dg}^{(t+1)}); \boldsymbol{\theta}_g\}\,, \tag{3.22}$$

  for $g = 1, \dots, G$.

Unfortunately, the two CM-steps don't have closed-form solutions, thus numerical optimization methods are used. In this thesis, R function `optim` with BFGS method is used to find the numerical solutions. After CM-step 2 is complete, the convergence criteria based on Aitken's acceleration (described in Section 2.3.2) will be checked. A summary of this ECM algorithm is shown in Algorithm 2.

## 3.3 Model Variations

As seen from the joint distribution of Equation (3.3), component joint densities can be decomposed into a copula function of the $D$ marginal CDFs and a product of the $D$ marginal PDFs. Therefore, the proposed models may vary by copula functions and/or marginal densities. By varying the copula functions, different types of dependence structures are induced. For example, we can use a Gumbel copula in cases where data exhibits upper tail dependence, or use a Clayton copula for cases of lower tail dependence. For marginal distributions, we may use a normal distribution for modelling symmetric margins or a Gamma distribution for skewed variables. A summary of the variations considered in this thesis is listed below:

**Algorithm 2** ECM Algorithm.

```
Initialize $z_{ig}^{(0)}$, $\pi_g^{(0)}$, $\boldsymbol{\beta}_g^{(0)}$, and $\boldsymbol{\theta}_g^{(0)}$
while convergence criterion is not met do
    E-step: update $\hat{z}_{ig}^{(t+1)} = \frac{\hat{\pi}_g^{(t)} f_g(\boldsymbol{y_i}|\boldsymbol{x_i}, \hat{\boldsymbol{\Theta}}_g^{(t)})}{\sum_{g=1}^{G} \hat{\pi}_g^{(t)} f_g(\boldsymbol{y_i}|\boldsymbol{x_i}, \hat{\boldsymbol{\Theta}}_g^{(t)})}$
    M-step: update $\hat{\pi}_g^{(t+1)} = \frac{\sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)}}{n}$
    CM-step 1: For $g = 1, \dots, G$, update
        $\hat{\boldsymbol{\beta}}_g^{(t+1)} = \text{argmax}_{\boldsymbol{\beta}_g} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} \left[\log c_g(\cdot; \hat{\boldsymbol{\theta}}_g^{(t)}) + \sum_{d=1}^{D} \log f_{dg}(\cdot; \boldsymbol{\beta}_{dg})\right]$
    CM-step 2: For $g = 1, \dots, G$, update
        $\hat{\boldsymbol{\theta}}_g^{(t+1)} = \text{argmax}_{\boldsymbol{\theta}_g} \sum_{i=1}^{n} \hat{z}_{ig}^{(t+1)} \log c_g\{F_{1g}(\cdot; \hat{\boldsymbol{\beta}}_{1g}^{(t+1)}), \dots, F_{Dg}(\cdot; \hat{\boldsymbol{\beta}}_{Dg}^{(t+1)}); \boldsymbol{\theta}_g\}$
    check convergence criterion
end while
```

- Copula functions:

  1. Gaussian copula: elliptical shaped dependence without tail dependence.
  2. Gumbel copula: upper tail dependence.
  3. Clayton copula: lower tail dependence.
  4. Frank copula: no tail dependence.
  5. Independent copula: variables are mutually independent.

- Marginal distributions:

  1. Normal distribution: Symmetric marginals.
  2. Exponential distribution: Marginals with inverse growth.
  3. Gamma distribution: Skewed-type marginals.

Therefore by the possible variations within the CMixR model, the proposed model can be flexibly chosen to fit data with various marginals and different dependence structures.

## 3.4 Summary

In this chapter, we proposed copula-based mixtures of regression models for analyzing multivariate response data. Specifically, the dependence structure among the response variables is captured using a copula. To address optimization difficulties encountered in the maximization step, we employ an ECM algorithm to estimate the model parameters. Detailed computational procedures are also provided. In the next chapter, a simulation study for different models as listed in Section 3.3 as well as a data analysis on purple rock crabs are conducted.

# Chapter 4

# Data Analyses

In this chapter, a simulation study for the CMixR model is conducted in Section 4.1, with varying types of continuous margins and different types of copulas. Clustering performance by the proposed CMixR method is compared with other existing methods based on ARI values. In Section 4.2, the Leptograpsus variegatus crab data set is analysed using the CMixR model.

## 4.1 Simulation Study

The objective of this simulation study is to evaluate the performance of the proposed copula-based mixtures of regression (CMixR) models and compare it with other existing methods. We consider a total of 9 different scenarios as listed in Table 4.1 with corresponding parameters specified in Table 4.2. For each scenario, a Monte Carlo simulation with a total of 100 samples ($M = 100$) is generated. For each simulation sample, we generate data of size $n = 100$ from two-component mixture models. The simulation scenarios considered varied marginal distributions and dependence structures while assuming a prior knowledge of the number of clusters as 2. A comparison of the proposed CMixR model with flexible Mixture Modeling (`flexmix`), parsimonious Gaussian mixture models (`pgmm`), and mixture models for clustering and classification (`mixture`) is conducted in this chapter. The accuracy of clustering results is evaluated using the ARI value, as described in Section 2.4. To visually compare the clustering performance of each method, boxplots of ARI values across samples are presented. For the `flexmix` method, multiple initializations with randomly assigned clusters are applied. The initial values chosen for the CMixR method are from the `flexmix` results, which include the initial component memberships as well as the regression coefficients. In order to clarify the notation used, $Y_{dg}$ represents the $d$th response variable $Y_d$ in cluster $g$, $\boldsymbol{\beta_{dg}} = (\beta_{dg0}, \beta_{dg1}, \beta_{dg2})'$ denotes the regression coefficients of $Y_d|\boldsymbol{X}$ in cluster $g$, and $\sigma_{dg}$ refers to any additional parameter of $Y_d|\boldsymbol{X}$ in cluster $g$, e.g, the standard deviation of a normal density. Notations here are the same as described in tree diagram 3.1 in the previous chapter, with $G = 2, D = 2, P = 2$.

In case 1, $Y_1$, $Y_2$ have normal distributions as margins in both cluster 1 and cluster 2 but with different parameters. The dependence structures are both set to be the Gaussian copula but with different parameters. Under this setting, all methods exhibit strong performance with a median ARI exceeding 0.95. This result is not surprising as normal margins with a Gaussian copula form the multivariate normal distribution, and therefore allow for effective clustering via Gaussian mixture models. However, if the dependence structure is no longer elliptical, the Gaussian copula and the Gaussian mixture model will be limited. To see the performance with a non-elliptical dependence structure, copulas exhibiting tail dependence are employed in the subsequent scenarios.

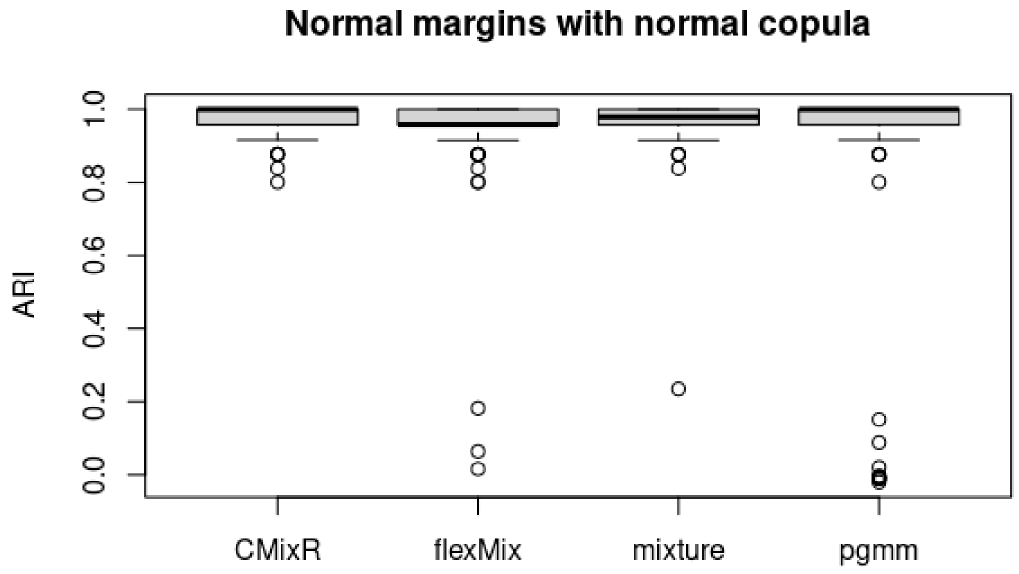

Figure 4.1: Boxplots of case 1: $Y_1$, $Y_2$ have normal margins with different parameters and normal copula is used to model dependence structure in both clusters with different parameters.

Table 4.1: Nine simulation scenarios used in the data generation and modelling process.

| | Case 1 | Case 2 | Case 3 | Case 4 | Case 5 | Case 6 | Case 7 | Case 8 | Case 9 |
|---|---|---|---|---|---|---|---|---|---|
| Marginal for $Y_{11}$ | Normal | Normal | Normal | Normal | Normal | Normal | Normal | Normal | Normal |
| Marginal for $Y_{12}$ | Normal | Normal | Normal | Normal | Normal | Normal | Normal | Normal | Normal |
| Marginal for $Y_{21}$ | Normal | Normal | Normal | Normal | Normal | Exponential | Gamma | Normal | Normal |
| Marginal for $Y_{22}$ | Normal | Normal | Normal | Normal | Normal | Exponential | Gamma | Normal | Normal |
| Copula in $g = 1$ | Normal | Clayton | Gumbel | Clayton | Clayton | Gumbel | Gumbel | Gumbel | Independent |
| Copula in $g = 2$ | Normal | Clayton | Gumbel | Frank | Gumbel | Gumbel | Gumbel | Independent | Independent |

Table 4.2: Parameter values used for the 9 simulation scenarios.

| Param | Case1 | Case2 | Case3 | Case4 | Case5 | Case6 | Case7 | Case8 | Case9 |
|---|---|---|---|---|---|---|---|---|---|
| $\beta_{0,11}$ | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\beta_{1,11}$ | 2 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |
| $\beta_{2,11}$ | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $\beta_{0,12}$ | 3 | 2 | 2 | 2 | 2 | -1 | -1 | 1 | 2 |
| $\beta_{1,12}$ | 2 | 2 | 2 | 2 | 2 | -1 | -1 | 1 | 2 |
| $\beta_{2,12}$ | 1 | 2 | 2 | 2 | 2 | -1 | -1 | 1 | 2 |
| $\beta_{0,21}$ | 4 | 1 | 1 | 1 | 1 | 0.5 | -0.2 | 1 | 1 |
| $\beta_{1,21}$ | 5 | 2 | 2 | 2 | 2 | 0.2 | -0.2 | 2 | 2 |
| $\beta_{2,21}$ | 6 | 3 | 3 | 3 | 3 | -0.2 | -0.2 | 3 | 3 |
| $\beta_{0,22}$ | 6 | 2 | 2 | 2 | 2 | 1 | 0.5 | 1 | 2 |
| $\beta_{1,22}$ | 5 | 3 | 3 | 3 | 3 | -0.5 | 0.5 | 2 | 3 |
| $\beta_{2,22}$ | 4 | 4 | 4 | 4 | 4 | 0 | 0 | 3 | 4 |
| $\sigma_{11}$ | 5 | 4 | 4 | 4 | 4 | 1 | 2 | 2 | 4 |
| $\sigma_{12}$ | 6 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| $\sigma_{21}$ | 7 | 3 | 3 | 3 | 3 | NA | 6 | 1 | 3 |
| $\sigma_{22}$ | 8 | 1 | 1 | 1 | 1 | NA | 8 | 1 | 1 |
| $\theta_1$ | 0.2 | 4 | 4 | 4 | 4 | 4 | 3 | 1 | NA |
| $\theta_2$ | 0.6 | 10 | 10 | 10 | 10 | 8 | 4 | 20 | NA |
| $\tau_1$ | 0.13 | 0.67 | 0.75 | 0.67 | 0.67 | 0.75 | 0.67 | 0 | 0 |
| $\tau_2$ | 0.41 | 0.83 | 0.9 | 0.67 | 0.9 | 0.88 | 0.75 | 0.95 | 0 |

In case 2, a lower tail dependence structure is imposed on $Y_1$ and $Y_2$ by using the Clayton copula. From Figure 4.2, it is evident that CMixR outperforms other methods and exhibits a narrower dispersion as well. Given that `flexmix` assumes independence between the response variables $Y_1$ and $Y_2$, it is reasonable to anticipate improved performance from CMixR, which accounts for the dependence structure. It can be seen that `pgmm` shows greater variation in performance, which may be due to this method being tailored for high-dimensional data. We only consider a 2-dimensional response vector here, so it may not be a high enough dimension response for good `pgmm` performance.

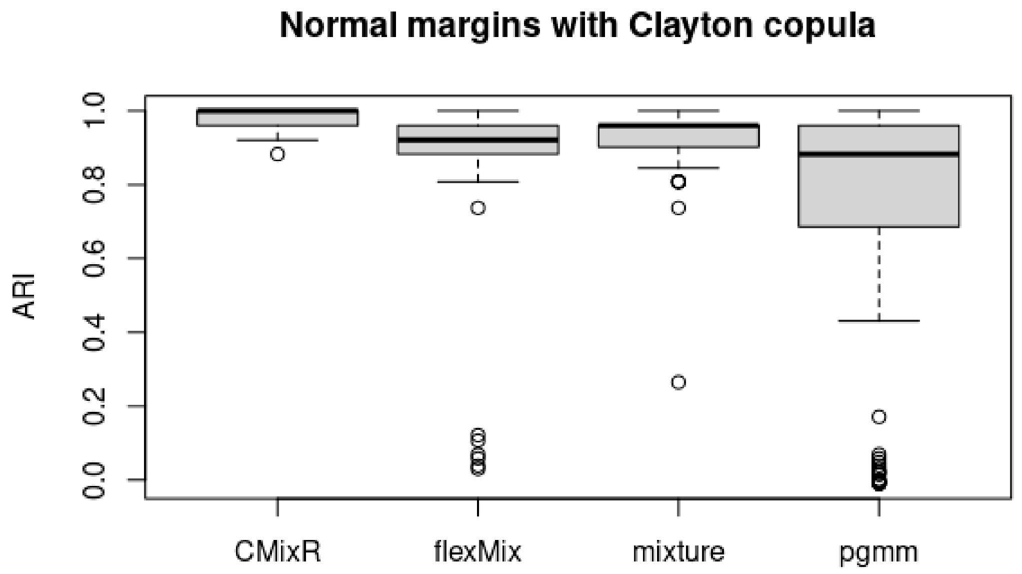

Figure 4.2: Boxplot of case 2: $Y_1$, $Y_2$ have normal margins with different parameters in two clusters and Clayton copula is used to model dependence structure in both clusters with different parameters.

In case 3, the Gumbel copula was employed to introduce an upper tail dependence structure on $Y_1$ and $Y_2$. As shown in Figure 4.3, both CMixR and `mixture` yielded highly comparable results with a median ARI of 1. However, CMixR exhibited a slightly higher mean ARI of 0.983 compared to the mean ARI of 0.966 obtained by `mixture`. At the same time, `flexmix` performs acceptably with a median ARI around 0.85, but much lower than the CMixR and `mixture`. Similar to case 2, `pgmm` has a median ARI value of around 0.83 but with a wide variation in performance.

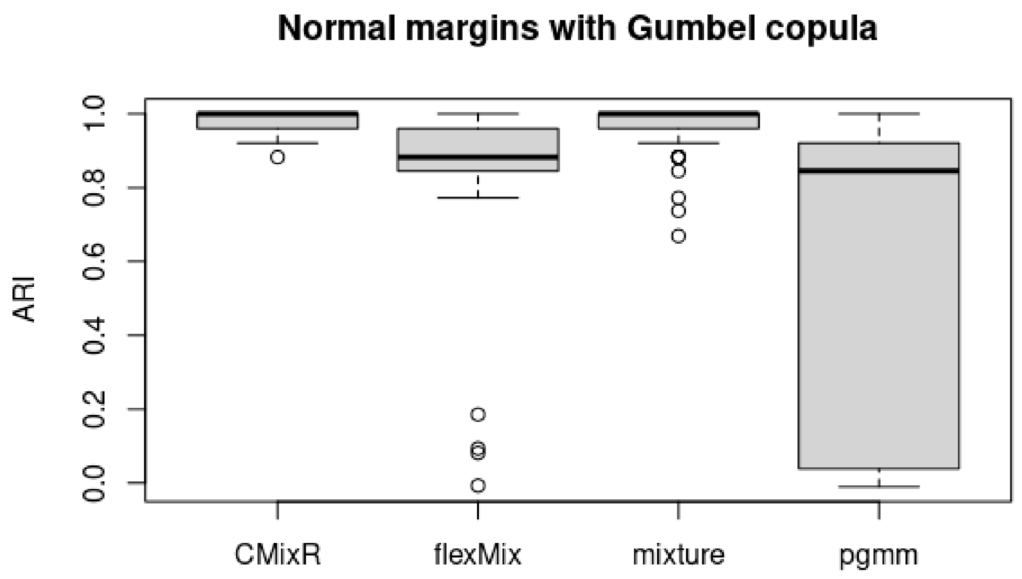

Figure 4.3: Boxplot of case 3: $Y_1$, $Y_2$ have normal margins with different parameters in two clusters and Gumbel copula is used to model dependence structure in both clusters with different parameters.

In case 4, we investigate the impact of different dependence structures between two clusters by using different types of copulas in two clusters. Copula $C_1$ is set to Clayton copula and copula $C_2$ is set to Frank copula, while $Y_1$, $Y_2$ have normal margins with different parameters. As can be seen from Figure 4.4, CMixR has the overall best performance with a higher median ARI value. `flexmix` and `mixture` also perform well with median ARI around 0.9. On the other hand, `pgmm` still has a wide range of dispersion, similar to previous cases.

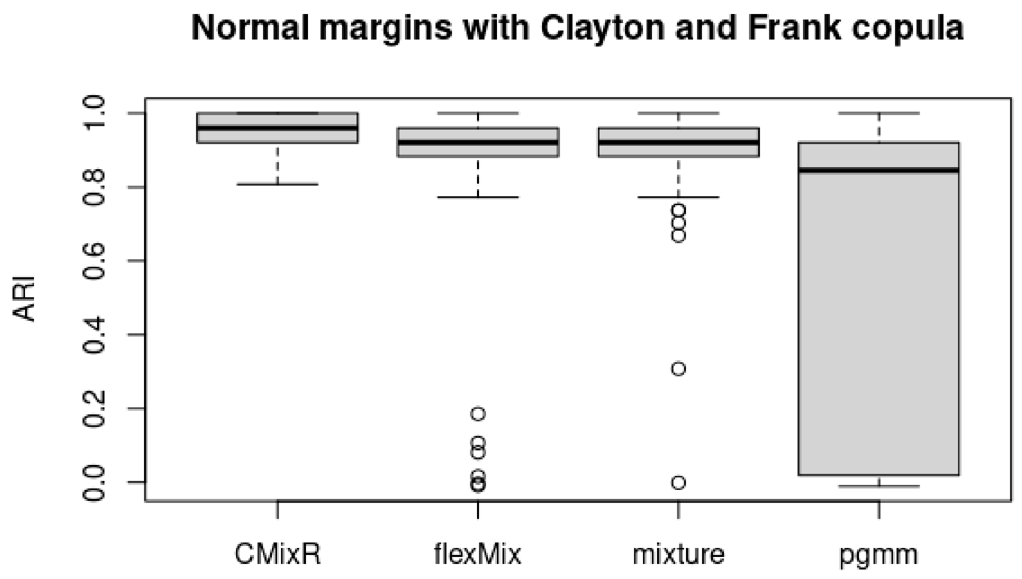

Figure 4.4: Boxplot of case 4: $Y_1$, $Y_2$ have normal margins with different parameters in two clusters. Cluster 1 has Clayton copula to model dependence structure, while cluster 2 uses Frank copula.

Regarding case 5, the copula $C_1$ remains as the Clayton copula as case 4, while the copula $C_2$ is replaced with the Frank copula. As shown in Figure 4.5, both CMixR and `mixture` exhibit superior performance compared to the other two methods. When comparing CMixR to `mixture`, it can be observed that they share an identical median ARI of 0.960; however, CMixR achieves a higher mean ARI of 0.962 in contrast to a mean ARI of 0.945 for `mixture`. At the same time, `flexmix` has a slightly lower ARI with a median value of 0.921. Again, `pgmm` has a wide variation in performance.

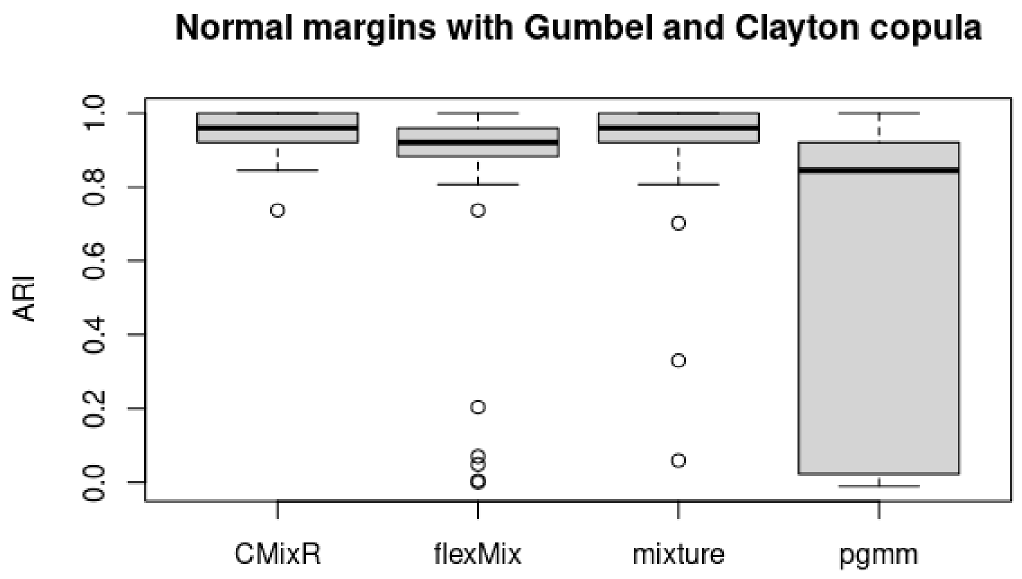

Figure 4.5: Boxplot of case 5: $Y_1$, $Y_2$ have normal margins with different parameters in two clusters. Cluster 1 uses Clayton copula to model dependence structure, while cluster 2 uses Gumbel copula.

To examine the impact of the marginal distributions, alternative distributions other than the normal distribution are employed. In case 6, $Y_1$ has a normal distribution, while $Y_2$ is generated from an exponential distribution. In case 7, $Y_1$ is still generated from a normal distribution, while $Y_2$ is generated from a Gamma distribution. Given that the `pgmm` and `mixture` are designed for parsimonious Gaussian mixture models, we solely compare CMixR with `flexmix`. Notably, in both cases, CMixR with non-normal margins exhibits slightly diminished performance compared to previous scenarios; nevertheless, it still achieves an ARI of approximately 0.75 which is better than the ARI obtained by `flexmix` (less than 0.5).

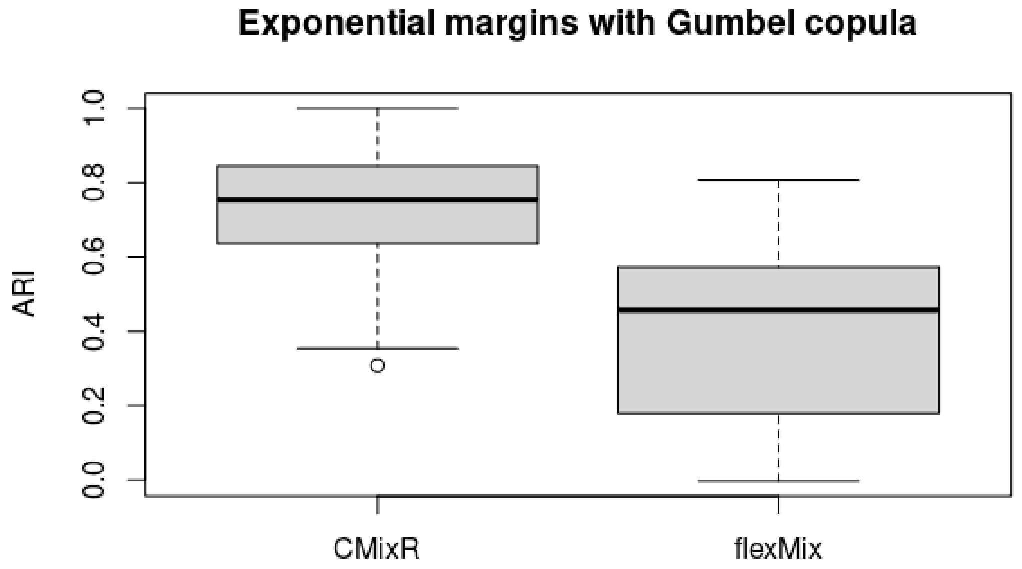

Figure 4.6: Boxplot of case 6: $Y_1$ has normal margins with different parameters in two clusters, $Y_2$ has exponential margins with different parameters in two clusters, and the Gumbel copula is used in both clusters.

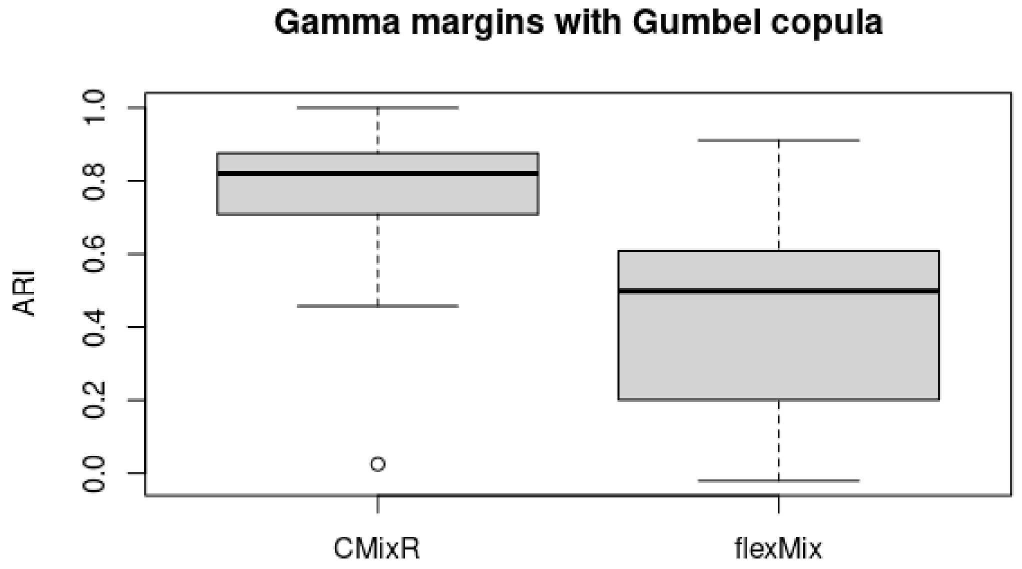

Figure 4.7: Boxplot of case 7: $Y_1$ has normal margins with different parameters in two clusters, $Y_2$ has Gamma margins with different parameters in two clusters, and the Gumbel copula is used in both clusters.

Case 8 is an interesting simulation setting because we set responses $Y_1$ and $Y_2$ to have the exactly same margins in both clusters 1 and 2. Specifically, we set $\boldsymbol{\beta}_{11} = \boldsymbol{\beta}_{12}, \sigma_{11} = \sigma_{12}$ and $\boldsymbol{\beta}_{21} = \boldsymbol{\beta}_{22}, \sigma_{21} = \sigma_{22}$. Copula $C_1$ is the independence copula, which means that $Y_1$ and $Y_2$ are conditionally independent in cluster 1. Copula $C_2$ is Gumbel copula with parameter $\theta_2 = 20$, which means that $Y_1$ and $Y_2$ are dependent with Kendall's $\tau = 0.95$ in cluster 2. This is the case when we expect the best performance from CMixR. Results from Figure 4.8 show that CMixR has superior performance and can be used to separate 2 clusters with hidden dependence in one cluster. On the other hand, `flexmix` and `pgmm` yield ARIs around 0, meaning that their results have approximately the same accuracy as randomly assigning clusters. `mixture` has some ARI values above 0.5, but the median ARI is low and performance is extremely widespread.

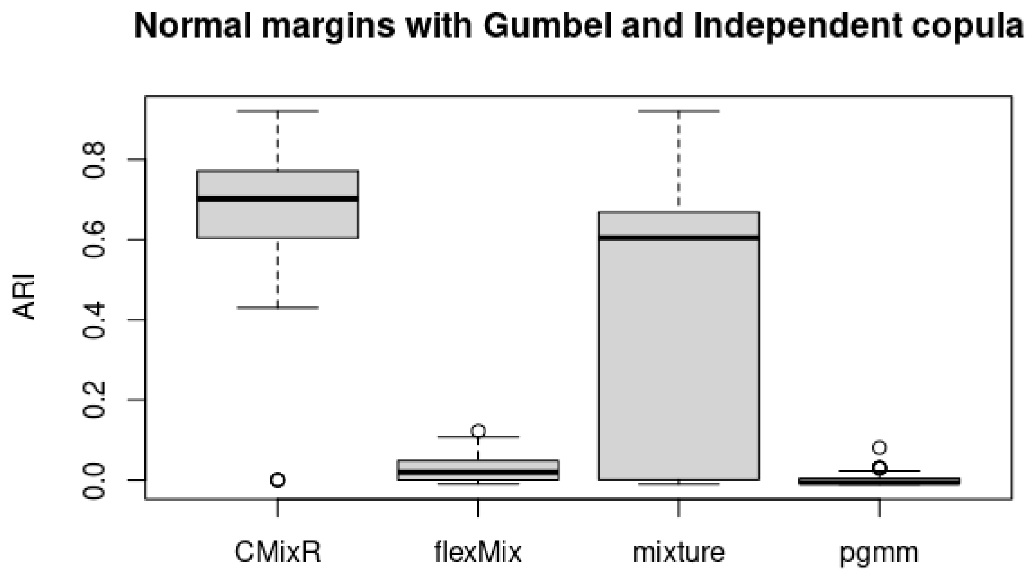

Figure 4.8: Boxplot of case 8: Independent $Y_1$,$Y_2$ with normal margins in cluster 1 and $Y_1$,$Y_2$ with normal margins and Gumbel copula in cluster 2. Moreover, the two clusters have identical parameters for the margins.

In case 8, there is an ARI value of 0 for the CMixR method, indicating a need for further investigation. From the residual dependence plot shown in Figure 4.9, it can be seen that the dependence structures used to generate the clusters are mixed in the result. Considering that the flexmix method yielded unsatisfactory results in this case (ARI $\approx 0$) and serves to initialize CMixR, there is a suspicion that the low ARI value in CMixR is a result of poor initialization. To check this conclusion, multiple randomizations were used for the initial cluster assignment for this particular simulated dataset to see if the ARI result improved. The highest ARI amongst all the initializations was 0.845, which is significantly better than the ARI value of 0 based on the flexmix initialization. Therefore, alternative initialization methods are needed to improve the performance of CMixR.

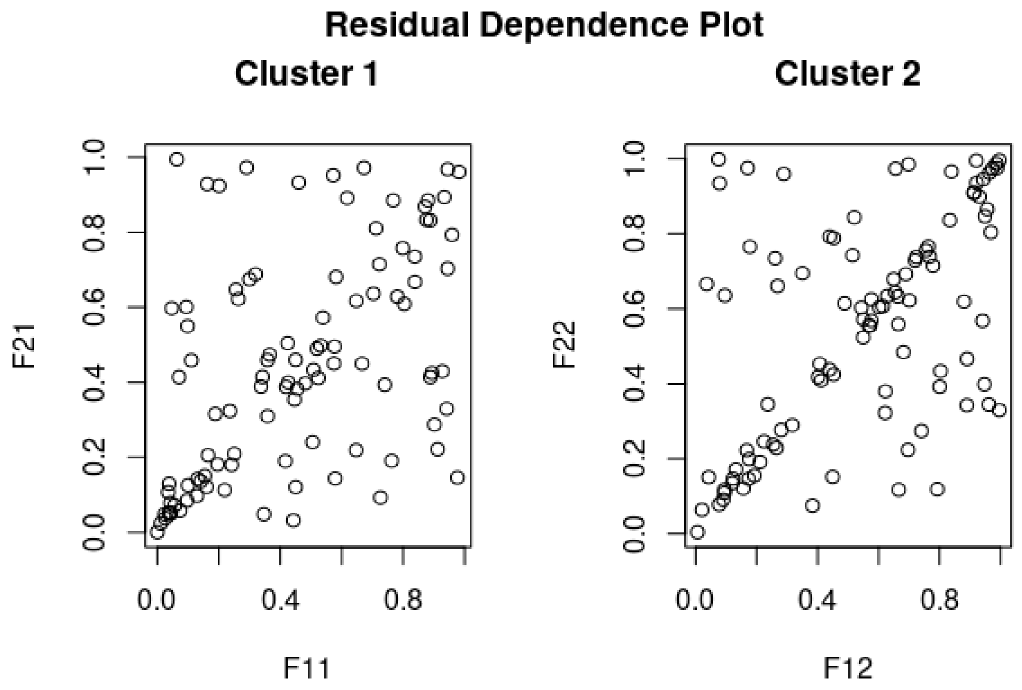

Figure 4.9: Plot for residual dependence structure of the case that CmixR performs not well in case 8.

Concerning case 9, the margins are independently simulated such that they are normal in both clusters. The Gumbel copula is used within CMixR to characterize the dependence structure for estimation, allowing us to assess the applicability of CMixR under conditions of independent dependence structure. This is the case flexmix is designed for and we expect comparable performance from CMixR and flexmix. From Figure 4.10, it can be seen that CMixR, flexmix, and mixture all provide highly comparable results. To be more specific, all methods have a median ARI of

0.96 but `flexmix` has a narrower spread than the other three methods.

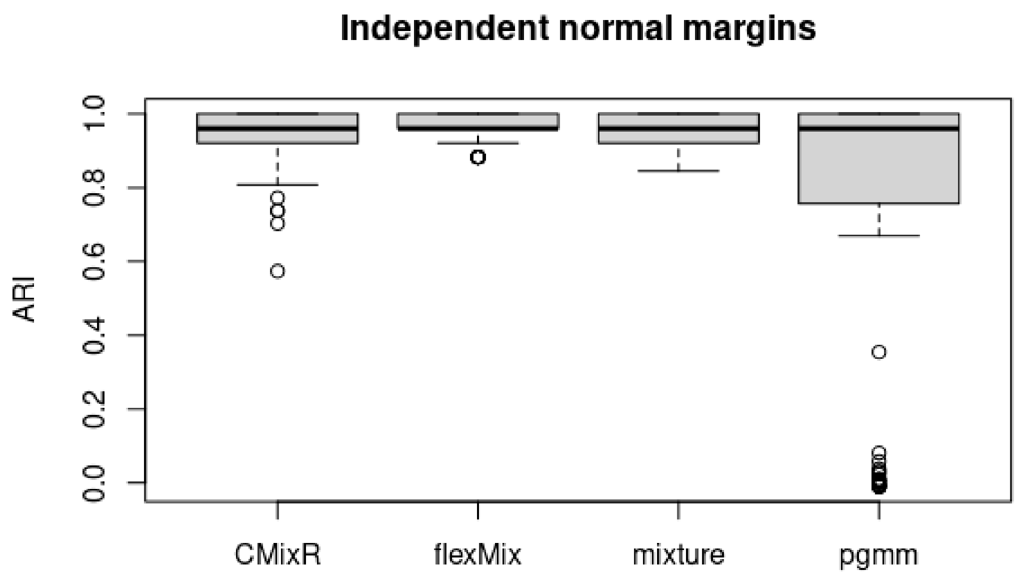

Figure 4.10: Boxplot of case 9: Independent $Y_1$,$Y_2$ with normal margins of different parameters in both clusters. Gumbel copula is used to model dependence structure in both clusters.

In summary, the proposed CMixR model demonstrates a better performance compared to the other three methods, especially when there exists a difference in the dependence structures of the clusters. Next, we apply this approach to a data set.

## 4.2 Crab data set

The famous crab data set, which is available in the R package `MASS`, is analyzed here. These data represent the outcomes of 200 crabs of the species Leptograpsus variegatus, with an equal distribution of 100 males and 100 females. A total of five morphological measurements were recorded, namely frontal lobe size (FL), rear width (RW), carapace length (CL), carapace width (CW), and body depth (BD). All variables in this dataset are continuous and measured in millimetres (mm). Summary statistics for all variables are shown in Table 4.3.

Following the approach employed by Dang et al. (2017), variables FL, RW, and CW that reflect width measurements are taken to be the response variables, with CL and BD as the covariates. Since the response variables under consideration are size, width, and width, respectively, it seems reasonable to employ normal margins. Histograms with density curves of these 3 response variables are shown in Figure 4.11. There appears to be a bimodal distribution for $Y_2$ and $Y_3$, and thus the number of components is set to be 2 in this thesis. Further investigation into determining the optimal number of clusters is considered for future work. Since there is no obvious tail dependence, we select the copula from the Gaussian copula and the Frank copula based on their Bayesian Information Criterion (BIC) values. There are a total of 28 parameters considered in this model, the Frank copula gives BIC= 1194.7, which is slightly less than BIC= 1194.9 for the Gaussian copula. Therefore, the Frank copula is used as the dependence structure for our model.

Table 4.3: Summary statistics for all variables: frontal lobe size (FL), rear width (RW), carapace width (CW), carapace length (CL), and body depth (BD) measured in mm. The first three are responses while the rest two are covariates.

| | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|---|
| FL | 7.20 | 12.90 | 15.55 | 15.58 | 18.05 | 23.10 |
| RW | 6.50 | 11.00 | 12.80 | 12.74 | 14.30 | 20.20 |
| CW | 17.10 | 31.50 | 36.80 | 36.41 | 42.00 | 54.60 |
| CL | 14.70 | 27.27 | 32.10 | 32.11 | 37.23 | 47.60 |
| BD | 6.10 | 11.40 | 13.90 | 14.03 | 16.60 | 21.60 |

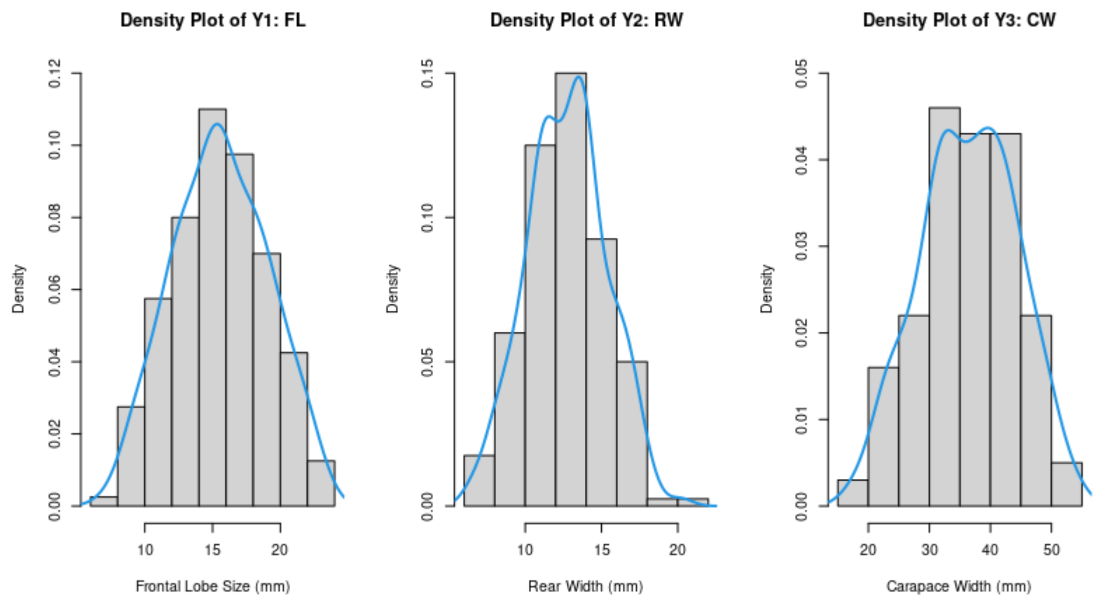

Figure 4.11: Histograms with density curves of response variables $Y_1$: frontal lobe size (FL), $Y_2$: rear width (RW), and $Y_3$: carapace width (CW).

Table 4.4: MAP tables in order of using CMixR (ARI=0.83), flexmix (ARI=0.81), mixture (ARI=0.72), and pgmm (ARI=0.77).

| | 1 | 2 |
|---|---|---|
| 1 | 98 | 2 |
| 2 | 7 | 93 |

| | 1 | 2 |
|---|---|---|
| 1 | 94 | 6 |
| 2 | 4 | 96 |

| | 1 | 2 |
|---|---|---|
| 1 | 0 | 100 |
| 2 | 85 | 15 |

| | 1 | 2 |
|---|---|---|
| 1 | 92 | 8 |
| 2 | 4 | 96 |

The resulting ARI values and MAP tables using all four methods are displayed in Table 4.4. Based on the ARI values, CMixR performs slightly better than other methods. The covariate effects with standard errors are presented in Table 4.5, where the standard errors are calculated by numerical differentiation. The Hessian matrix of the observed log-likelihood at EM estimates, $H(\hat{\boldsymbol{\beta}})$, is numerically obtained using the R package `numDeriv`. The results reveal a positive relationship between the covariate $X_1$ (CL) and all response variables. Regarding covariate $X_2$ (BD), it exhibits a positive relationship with $Y_1$ (FL) but a negative relationship with $Y_3$ (CW). Notably, the relationship between $X_2$ and $Y_2$ (RW) is interesting, as there is no significant relationship within cluster 1 and a slightly negative relationship within cluster 2, indicating a weak relationship between the body depth and the rear width of crabs.

Table 4.5: Regression results for the three responses, each $Y_{dg}$ follows normal distribution with mean $\beta_0 + \beta_0 x_1 + \beta_0 x_2$.

| | $\beta_0$ (intercept) | $\beta_1$ ($x_1$) | $\beta_2$ ($x_2$) |
|---|---|---|---|
| $Y_{11}$ | 1.33<br>(0.16) | 0.12<br>(0.02) | 0.73<br>(0.05) |
| $Y_{12}$ | 0.26<br>(0.37) | 0.19<br>(0.06) | 0.67<br>(0.12) |
| $Y_{21}$ | 2.80<br>(0.20) | 0.26<br>(0.03) | 0.06<br>(0.07) |
| $Y_{22}$ | 0.67<br>(0.23) | 0.43<br>(0.04) | -0.05<br>(0.01) |
| $Y_{31}$ | -0.14<br>(0.25) | 1.52<br>(0.02) | -0.89<br>(0.03) |
| $Y_{32}$ | 0.62<br>(0.27) | 1.36<br>(0.04) | -0.55<br>(0.02) |

Note that the recorded ARI value uses the sexes of the crabs, male and female, as the true clusters. This data set does include another factor variable called "species colour" with blue and orange. In a study conducted by Dang et al. (2017), the `mixture` approach selected 4 clusters, namely blue female, blue male, orange female, and orange male, based on some model selection criteria. It was observed that the ARI for the `mixture` approach increased to 0.78 when the number of clusters was 4 rather than being fixed at 2. However, in order to conduct a fair comparison, all methods employed in this thesis are set to use a two-component mixture model. As mentioned earlier, further investigation into model selection for the proposed CMixR method is considered as future work. In general, the CMixR shows a competitive outcome when the predetermined number of clusters is known.

# Chapter 5

# Discussion

Finite mixture models have been widely used across various disciplines such as biology, economics, engineering, and environmental science. In addition to the conventional Gaussian mixture models, extensions have been done in previous studies. For example, in order to address the computational challenge arising from the quadratic increase in the number of covariance parameters with data dimension, McNicholas and Murphy (2008) proposed parsimonious Gaussian mixture models (PGMM) as a means to have the number of covariance parameters increase linearly with dimension. Leisch (2004) proposed a general framework for finite mixtures of regression models to consider the covariate effects; however, it assumes that any multivariate response variables are mutually independent. To consider dependent continuous responses as well as the regression relationships, we applied the copula framework to the finite mixtures of regression models and proposed the copula-based mixtures of regression (CMixR) models.

The proposed CMixR models account for heterogeneous regression data with multivariate dependent response variables. These models are used to cluster observations based on commonalities and can be used to interpret changes in covariate effects across clusters. Due to the higher computational cost in the discrete case, we focused only on continuous response variables in this thesis. For continuous variables, Sklar's theorem states that the copula can fully describe the dependence structure and is independent of the marginal distributions. Using a copula framework enables us to capture a wide range of shapes for the dependence structure and have different distributions as margins, yielding more flexible models. Estimation of the CMixR model was conducted through an ECM algorithm, which was chosen to address the computational challenges in the maximization step of the EM algorithm.

As illustrated in the simulation study, the dependence structure can be easily and flexibly modified by varying the selected copula. Additionally, a range of continuous margin types can be employed, ranging from normal to gamma distributions. Through performance assessments based on ARI values in the simulation study, it is evident that the CMixR model exhibits similar or superior clustering performance in all the considered simulation scenarios when compared to existing methods such as `flexmix`, `pgmm`, and `mixture`. In the case of normal margins with normal copulas, all methods exhibit strong performance, as they are all effective for clustering via Gaussian mixture models. When the dependence is no longer normal, the CMixR model exhibits its advantages and `mixture` is the only method that shows similar performance. However, it is worth noting that `pgmm` shows a large variability in performance, which may be due to the lower-dimensional response data in the simulation study. One important advantage of the CMixR model is its ability to handle situations where responses are dependent in certain clusters but independent in other clusters, while the marginal distributions are identical for all clusters. In this situation, the CMixR method is the only one that can be used.

Furthermore, when applied to real-world data of purple crabs with five morphological measurements, CMixR shows its effectiveness and applicability. Through the exploratory data analysis, we determined the number of clusters as 2 and used normal distributions as margins along with Frank copulas to capture the dependence structure. For the covariate effects, carapace length exhibits a positive relationship with lobe size, rear width, and carapace width. Additionally, body depth demonstrates a positive effect on lobe size but a negative effect on carapace width. Interestingly, it yields a weak effect on rear width: no significant effect in cluster 1 while a slight negative effect in cluster 2. To ensure a fair comparison of the clustering performance, all methods were set to use a two-component mixture model without using any model selection criteria. The CMixR method shows a slightly better performance in terms of ARI when compared with `flexmix`, `pgmm`, and `mixture`. However, further investigation into model selection for the proposed CMixR method should be considered, as discussed in the next section.

## 5.1 Future work

There are many avenues for future research with this work. First, model selection based on Akaike information criterion (AIC), Bayesian information criterion (BIC), or another suitable criterion should be investigated. Model selection for the CMixR would be used to determine the optimal number of clusters, and the most appropriate marginal distributions for the response variables, and select an appropriate copula to define the dependence structures. Second, rotational copulas as mentioned by Kosmidis and Karlis (2016) could be introduced for non-elliptical dependence structure. For example, the $180^\circ$ rotated version of the Clayton copula exhibits upper-tail dependence, rather than the standard lower-tail dependence. This would add flexibility while restricting all dependence structures to the same family. Third, alternative marginal distributions such as the Student $t$ distribution and the beta distribution could be implemented to enhance the flexibility of the current CMixR model. Fourth, while continuous marginal distributions have been the focus of this thesis, there is a need for further investigation of discrete and mixed-type response data. The bivariate discrete Poisson case has been investigated by Bermúdez and Karlis (2022), but more work needs to be done to extend their models to the computationally demanding, high-dimensional discrete cases. Fifth, the proposed CMixR models incorporate Archimedean copulas, such as the Clayton and Gumbel copulas, and may encounter limitations when applied to high-dimensional data, as Archimedean copulas still rely on a single parameter to determine all dependencies. Therefore, other more suitable copulas like vine copula (Nagler et al., 2019) could be investigated. Finally, the number of parameters of the Gaussian copula increases quadratically with the dimension, which poses challenges in high-dimensional cases. Restricted covariances, such as those used by the PGMM could be explored for the CMixR model.

# Bibliography

Aitken, A. (1926). Iii.—a series formula for the roots of algebraic and transcendental equations. *Proceedings of the Royal Society of Edinburgh*, 45(1):14–22.

Andrews, J. L. and McNicholas, P. D. (2011). Mixtures of modified t-factor analyzers for model-based clustering, classification, and discriminant analysis. *Journal of Statistical Planning and Inference*, 141(4):1479–1486.

Bermúdez, L. and Karlis, D. (2022). Copula-based bivariate finite mixture regression models with an application for insurance claim count data. *TEST*, 31(4):1082–1099.

Caliński, T. and Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27.

Dang, U. J., Punzo, A., McNicholas, P. D., Ingrassia, S., and Browne, R. P. (2017). Multivariate response and parsimony for gaussian cluster-weighted models. *Journal of Classification*, 34:4–34.

Davies, D. L. and Bouldin, D. W. (1979). A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, pages 224–227.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22.

Frühwirth-Schnatter, S. and Pyne, S. (2010). Bayesian inference for finite mixtures of univariate and multivariate skew-normal and skew-t distributions. *Biostatistics*, 11(2):317–336.

Hubert, L. and Arabie, P. (1985). Comparing partitions. *Journal of classification*, 2:193–218.

Joe, H. (1997). *Multivariate models and multivariate dependence concepts*. CRC press.

Karlis, D. and Santourian, A. (2009). Model-based clustering with non-elliptically contoured distributions. *Statistics and Computing*, 19:73–83.

Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Kolev, N. and Paiva, D. (2009). Copula-based regression models: A survey. *Journal of statistical planning and inference*, 139(11):3847–3856.

Kosmidis, I. and Karlis, D. (2016). Model-based clustering using copulas with applications. *Statistics and computing*, 26:1079–1099.

Leisch, F. (2004). Flexmix: A general framework for finite mixture models and latent class regression in r. *Journal of Statistical Software*, 11(8):1–18.

Masarotto, G. and Varin, C. (2012). Gaussian copula marginal regression. *Electronic Journal of Statistics*, 6:1517 – 1549.

McLachlan, G. and Chang, S. (2004). Mixture modelling for cluster analysis. *Statistical methods in medical research*, 13(5):347–361.

McNicholas, P. D. (2016). *Mixture model-based classification*. CRC press.

McNicholas, P. D. and Murphy, T. B. (2008). Parsimonious gaussian mixture models. *Statistics and Computing*, 18:285–296.

McNicholas, P. D., Murphy, T. B., McDaid, A. F., and Frost, D. (2010). Serial and parallel implementations of model-based clustering via parsimonious gaussian mixture models. *Computational Statistics & Data Analysis*, 54(3):711–723.

Meng, X.-L. and Rubin, D. B. (1993). Maximum likelihood estimation via the ecm algorithm: A general framework. *Biometrika*, 80(2):267–278.

Nagler, T., Bumann, C., and Czado, C. (2019). Model selection in sparse high-dimensional vine copula models with an application to portfolio risk. *Journal of Multivariate Analysis*, 172:180–192.

Pitt, M., Chan, D., and Kohn, R. (2006). Efficient bayesian inference for gaussian copula regression models. *Biometrika*, 93(3):537–554.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.

Saxena, A., Prasad, M., Gupta, A., Bharill, N., Patel, O. P., Tiwari, A., Er, M. J., Ding, W., and Lin, C.-T. (2017). A review of clustering techniques and developments. *Neurocomputing*, 267:664–681.

Sklar, M. (1959). Fonctions de repartition an dimensions et leurs marges. *Publ. inst. statist. univ. Paris*, 8:229–231.

# Appendix A

# R Code

## A.1 Simulation Study

```
rm(list = ls())
require(copula)
require(flexmix)
require(mclust)
require(mixture)
require(pgmm)

# EM algorithm
## E-step
E_step=function(y, x, beta_mat, sigma_vec, theta_vec, pi_vec){
  y1 = y[,1]; y2 = y[,2]
  x1 = x[,1]; x2 = x[,2]
  beta11 = beta_mat[,1]    #beta for Y1, g=1
  beta12 = beta_mat[,2]    #beta for Y1, g=2
  beta21 = beta_mat[,3]    #beta for Y2, g=1
  beta22 = beta_mat[,4]    #beta for Y2, g=2

  mu11 = beta11[1] + beta11[2]*x1 + beta11[3]*x2   #Y1, g=1
  mu12 = beta12[1] + beta12[2]*x1 + beta12[3]*x2   #Y1, g=2
  mu21 = beta21[1] + beta21[2]*x1 + beta21[3]*x2   #Y2, g=1
  mu22 = beta22[1] + beta22[2]*x1 + beta22[3]*x2   #Y2, g=2

  F11 = pnorm(y1, mean = mu11, sd = sigma_vec[1])   #Y1, g=1
  F12 = pnorm(y1, mean = mu12, sd = sigma_vec[2])   #Y1, g=2
  F21 = pnorm(y2, mean = mu21, sd = sigma_vec[3])   #Y2, g=1
  F22 = pnorm(y2, mean = mu22, sd = sigma_vec[4])   #Y2, g=2

  cop1 = gumbelCopula(theta_vec[1])
  cop2 = gumbelCopula(theta_vec[2])
  c1 = dCopula(cbind(F11,F21), cop1)      #copula density of (Y1, Y2), g=1
  c2 = dCopula(cbind(F12,F22), cop2)      #copula density of (Y1, Y2), g=2

  f11 = dnorm(y1, mean = mu11, sd = sigma_vec[1])   #marginal density of Y1, g=1
  f12 = dnorm(y1, mean = mu12, sd = sigma_vec[2])   #marginal density of Y1, g=2
  f21 = dnorm(y2, mean = mu21, sd = sigma_vec[3])   #marginal density of Y2, g=1
  f22 = dnorm(y2, mean = mu22, sd = sigma_vec[4])   #marginal density of Y2, g=2
  comp_dens1 = c1 * pi_vec[1] * f11 * f21
  comp_dens2 = c2 * pi_vec[2] * f12 * f22
  sum_dens = comp_dens1 + comp_dens2
  z1.vec = comp_dens1 / sum_dens
  z2.vec = comp_dens2 / sum_dens
  zhat_mat = cbind(z1.vec,z2.vec)
  return(zhat_mat)
}

## M-step1 for pi1 and pi2, returns pi_vec
M_step1 = function(zhat_mat){
  n = nrow(zhat_mat)
  sum_z1 = sum(zhat_mat[,1])
  sum_z2 = sum(zhat_mat[,2])
  pi1 = sum_z1/n
  pi2 = sum_z2/n
  pi_vec = c(pi1,pi2)
  return(pi_vec)
}
```

```
## M-step2: CM-step1 and CM-step2
optFunc = function(param_g, theta_g, zhat_g){
  beta_1g = param_g[1:3]
  beta_2g = param_g[4:6]
  sigma_1g = param_g[7]
  sigma_2g = param_g[8]
  mu_1g = beta_1g[1] + beta_1g[2]*x1 + beta_1g[3]*x2
  mu_2g = beta_2g[1] + beta_2g[2]*x1 + beta_2g[3]*x2
  f1g = dnorm(y1, mean = mu_1g, sd = sigma_1g, log = TRUE)
  f2g = dnorm(y2, mean = mu_2g, sd = sigma_2g, log = TRUE)
  F1g = pnorm(y1, mean = mu_1g, sd = sigma_1g)    #Y1
  F2g = pnorm(y2, mean = mu_2g, sd = sigma_2g)    #Y2
  cop_g = gumbelCopula(theta_g)
  logcg = dCopula(cbind(F1g,F2g), cop_g, log = TRUE)
  zlog_cg_fg = zhat_g * (logcg + f1g + f2g)
  arg = sum(zlog_cg_fg)
  return(arg)
}

CM_step1 = function(y, x, zhat_mat, beta_mat_initial,
                    sigma_vec_initial, theta_vec){
  y1 = y[,1]; y2 = y[,2]
  x1 = x[,1]; x2 = x[,2]
  beta_mat = matrix(NA, 3, 4)
  sigma_vec = rep(NA, 4)

  # g=1
  init1 = c(beta_mat_initial[,1], beta_mat_initial[,3],
            sigma_vec_initial[1], sigma_vec_initial[3])
  res1 = optim(init1, optFunc, theta_g = theta_vec[1],
               zhat_g=zhat_mat[,1], method = "L-BFGS-B",
               lower = c(-Inf, -Inf, -Inf, -Inf, -Inf, -Inf,
                          .Machine$double.eps, .Machine$double.eps),
               upper = c(Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf),
               control = list(fnscale = -1))
  beta_mat[,1] = res1$par[1:3]
  beta_mat[,3] = res1$par[4:6]
  sigma_vec[1] = res1$par[7]
  sigma_vec[3] = res1$par[8]

  # g=2
  init2 = c(beta_mat_initial[,2], beta_mat_initial[,4],
            sigma_vec_initial[2], sigma_vec_initial[4])
  res2 = optim(init2, optFunc, theta_g = theta_vec[2],
               zhat_g=zhat_mat[,2], method = "L-BFGS-B",
               lower = c(-Inf, -Inf, -Inf, -Inf, -Inf, -Inf,
               .Machine$double.eps, .Machine$double.eps),
               upper = c(Inf, Inf, Inf, Inf, Inf, Inf, Inf, Inf),
               control = list(fnscale = -1))
  beta_mat[,2] = res2$par[1:3]
  beta_mat[,4] = res2$par[4:6]
  sigma_vec[2] = res2$par[7]
  sigma_vec[4] = res2$par[8]
  return(list("beta_mat" = beta_mat, "sigma_vec" = sigma_vec))
}

optFunc2 = function(theta_g, F1g, F2g, zhat_g){
  cop_g = gumbelCopula(theta_g)
  logcg = dCopula(cbind(F1g,F2g), cop_g, log = TRUE)
  zlogcg = zhat_g * logcg
  arg = sum(zlogcg)
  return(arg)
}
```

```
CM_step2 = function(y, x, zhat_mat, beta_mat,
                    sigma_vec, theta_vec_initial){
  y1 = y[,1]; y2 = y[,2]
  x1 = x[,1]; x2 = x[,2]
  beta11 = beta_mat[,1]    #beta for Y1, g=1
  beta12 = beta_mat[,2]    #beta for Y1, g=2
  beta21 = beta_mat[,3]    #beta for Y2, g=1
  beta22 = beta_mat[,4]    #beta for Y2, g=2

  mu11 = beta11[1] + beta11[2]*x1 + beta11[3]*x2   #Y1, g=1
  mu12 = beta12[1] + beta12[2]*x1 + beta12[3]*x2   #Y1, g=2
  mu21 = beta21[1] + beta21[2]*x1 + beta21[3]*x2   #Y2, g=1
  mu22 = beta22[1] + beta22[2]*x1 + beta22[3]*x2   #Y2, g=2

  F11 = pnorm(y1, mean = mu11, sd = sigma_vec[1])   #Y1, g=1
  F12 = pnorm(y1, mean = mu12, sd = sigma_vec[2])   #Y1, g=2
  F21 = pnorm(y2, mean = mu21, sd = sigma_vec[3])   #Y2, g=1
  F22 = pnorm(y2, mean = mu22, sd = sigma_vec[4])   #Y2, g=2

  theta_vec = rep(NA, 2)

  # g=1
  res1 = optim(theta_vec_initial[1], optFunc2, F1g = F11, F2g=F21,
               zhat_g=zhat_mat[,1], method = "L-BFGS-B",
               lower = 1 + .Machine$double.eps,
               upper = Inf,
               control = list(fnscale = -1))
  theta_vec[1] = res1$par

  # g=2
  res2 = optim(theta_vec_initial[2], optFunc2, F1g = F12, F2g=F22,
               zhat_g=zhat_mat[,2], method = "L-BFGS-B",
             lower = 1 + .Machine$double.eps,
             upper = Inf,
             control = list(fnscale = -1))
  theta_vec[2] = res2$par
  return(theta_vec)
}

loglik = function(y, x, pi_vec, beta_mat, sigma_vec, theta_vec){
  y1 = y[,1]; y2 = y[,2]
  x1 = x[,1]; x2 = x[,2]

  beta11 = beta_mat[,1]    #beta for Y1, g=1
  beta12 = beta_mat[,2]    #beta for Y1, g=2
  beta21 = beta_mat[,3]    #beta for Y2, g=1
  beta22 = beta_mat[,4]    #beta for Y2, g=2

  mu11 = beta11[1] + beta11[2]*x1 + beta11[3]*x2   #mu for Y1, g=1
  mu12 = beta12[1] + beta12[2]*x1 + beta12[3]*x2   #mu for Y1, g=2
  mu21 = beta21[1] + beta21[2]*x1 + beta21[3]*x2   #mu for Y2, g=1
  mu22 = beta22[1] + beta22[2]*x1 + beta22[3]*x2   #mu for Y2, g=2

  F11 = pnorm(y1, mean = mu11, sd = sigma_vec[1])    #Y1, g=1
  F12 = pnorm(y1, mean = mu12, sd = sigma_vec[2])    #Y1, g=2
  F21 = pnorm(y2, mean = mu21, sd = sigma_vec[3])    #Y2, g=1
  F22 = pnorm(y2, mean = mu22, sd = sigma_vec[4])    #Y2, g=2

  cop1 = gumbelCopula(theta_vec[1])
  cop2 = gumbelCopula(theta_vec[2])
  c1 = dCopula(cbind(F11,F21), cop1)
  c2 = dCopula(cbind(F12,F22), cop2)

  comp1 = pi_vec[1] * c1 * dnorm(y1, mean = mu11, sd = sigma_vec[1])
```

```
          * dnorm(y2, mean = mu21, sd = sigma_vec[3])
  comp2 = pi_vec[2] * c2 * dnorm(y1, mean = mu12, sd = sigma_vec[2])
          * dnorm(y2, mean = mu22, sd = sigma_vec[4])
  sum_comp = comp1 + comp2
  loglik = sum(log(sum_comp))
  return(loglik)
}

## MC simulation
set.seed(1)
n = 100
M = 100
max.iter = 1000
data = array(rep(NA, 100*4*M), dim=c(100, 4, M))
beta_mat_MC <- array(rep(NA, 3*4*M), dim=c(3, 4, M))
sigma_vec_MC <- matrix(rep(NA, M*4), nrow = M)
theta_vec_MC <- matrix(rep(NA, M*2), nrow = M)
pi_vec_MC <- matrix(rep(NA, M*2), nrow = M)
ARI_MC <- rep(NA, M)
MAP_MC <- array(rep(NA, 2*2*M), dim=c(2, 2, M))
ARI_MC_flex <- rep(NA, M)
MAP_MC_flex <- array(rep(NA, 2*2*M), dim=c(2, 2, M))
ARI_MC_mix <- rep(NA, M)
MAP_MC_mix <- array(rep(NA, 2*2*M), dim=c(2, 2, M))
ARI_MC_pgmm <- rep(NA, M)
MAP_MC_pgmm <- array(rep(NA, 2*2*M), dim=c(2, 2, M))

true_beta = c(1,1,1, 1,1,1, 1,2,3, 1,2,3)
true_sigma = c(2,2,1,1)
true_theta = c(1, 20)

# Data generation
```

```
for (m in 1:M) {
  x1 = runif(n, min = 0, max = 10)
  x2 = runif(n, min = 1, max = 11)
  mu11 = true_beta[1] + true_beta[2]*x1[1:(n/2)]
         + true_beta[3]*x2[1:(n/2)]      #Y1, g=1
  mu12 = true_beta[4] + true_beta[5]*x1[(n/2+1):n]
         + true_beta[6]*x2[(n/2+1):n]  #Y1, g=2
  mu21 = true_beta[7] + true_beta[8]*x1[1:(n/2)]
         + true_beta[9]*x2[1:(n/2)]      #Y2, g=1
  mu22 = true_beta[10] + true_beta[11]*x1[(n/2+1):n]
         + true_beta[12]*x2[(n/2+1):n]  #Y2, g=2
  cop1 <- rCopula(n/2, gumbelCopula(true_theta[1]))
  cop2 <- rCopula(n/2, gumbelCopula(true_theta[2]))
  y11 = qnorm(cop1[,1], mean = mu11, sd = true_sigma[1])
  y12 = qnorm(cop2[,1], mean = mu12, sd = true_sigma[2])
  y21 = qnorm(cop1[,2], mean = mu21, sd = true_sigma[3])
  y22 = qnorm(cop2[,2], mean = mu22, sd = true_sigma[4])
  y1 = c(y11,y12)
  y2 = c(y21,y22)
  x = cbind(x1,x2)
  y = cbind(y1,y2)
  data[,,m] = cbind(x1,x2,y1,y2)
}

for (m in 1:M) {
  x = data[,,m][,1:2]
  y = data[,,m][,3:4]
  # Other methods: performance evaluation
  flex.clust = stepFlexmix(~x[,1]+x[,2], k=2, nrep=10,
                           model=list(FLXMRglm(y[,1]~x[,1]+x[,2]),
                                      FLXMRglm(y[,2]~x[,1]+x[,2])))
  flex.clust = flex.clust@cluster
```

```
pgmm.clust = pgmmEM(scale(cbind(x,y)),rG=2:2,zstart=1,loop=10, relax=TRUE)
pgmm.clust = pgmm.clust$map
mix.clust = pcm(cbind(y,x), G=2:2, pcmfamily = c(gpcm,ghpcm,vgpcm,tpcm,stpcm))
mix.clust = mix.clust$best_model$map

ARI_MC_flex[m] = adjustedRandIndex(c(rep(1,50),rep(2,50)),flex.clust)
ARI_MC_pgmm[m] = adjustedRandIndex(c(rep(1,50),rep(2,50)),pgmm.clust)
ARI_MC_mix[m] = adjustedRandIndex(c(rep(1,50),rep(2,50)),mix.clust)

MAP_MC_flex[,,m] = table(c(rep(1,50),rep(2,50)),flex.clust)
MAP_MC_pgmm[,,m] = table(c(rep(1,50),rep(2,50)),pgmm.clust)
MAP_MC_mix[,,m] = table(c(rep(1,50),rep(2,50)),mix.clust)

ll_vec = c(-Inf, rep(NA, max.iter-1))
aitken = rep(NA, max.iter)
# Initialization for CMixR
zhat_mat = cbind(flex.clust-1, 2-flex.clust)
beta_mat_initial = matrix(NA, 3, 4)
beta_mat_initial[,1] = flex.clust@components$Comp.1[[1]]@parameters$coef
beta_mat_initial[,2] = flex.clust@components$Comp.2[[1]]@parameters$coef
beta_mat_initial[,3] = flex.clust@components$Comp.1[[2]]@parameters$coef
beta_mat_initial[,4] = flex.clust@components$Comp.2[[2]]@parameters$coef
sigma_vec_initial = rep(NA, 4)
sigma_vec_initial[1] = flex.clust@components$Comp.1[[1]]@parameters$sigma
sigma_vec_initial[2] = flex.clust@components$Comp.2[[1]]@parameters$sigma
sigma_vec_initial[3] = flex.clust@components$Comp.1[[2]]@parameters$sigma
sigma_vec_initial[4] = flex.clust@components$Comp.2[[2]]@parameters$sigma

pi_vec = M_step1(zhat_mat)
cmstep1 = CM_step1(y, x, zhat_mat, beta_mat_initial = beta_mat_initial,
                   sigma_vec_initial = sigma_vec_initial, theta_vec = c(2,2))
beta_mat = cmstep1$beta_mat
```

```
sigma_vec = cmstep1$sigma_vec
theta_vec = CM_step2(y, x, zhat_mat, beta_mat, sigma_vec,
                     theta_vec_initial=c(2,2))
for(i in 1:max.iter){
  zhat_mat = E_step(y, x, beta_mat, sigma_vec, theta_vec, pi_vec)
  pi_vec = M_step1(zhat_mat)
  cmstep1 = CM_step1(y, x, zhat_mat, beta_mat_initial = beta_mat,
                     sigma_vec_initial = sigma_vec, theta_vec)
  beta_mat = cmstep1$beta_mat
  sigma_vec = cmstep1$sigma_vec
  theta_vec = CM_step2(y, x, zhat_mat, beta_mat, sigma_vec,
                       theta_vec_initial=theta_vec)
  ll_vec[i+1] = loglik(y, x, pi_vec, beta_mat, sigma_vec, theta_vec)
  if(i>2){
    aitken[i] = (ll_vec[i+1]-ll_vec[i])/(ll_vec[i]-ll_vec[i-1])
    ll_inf = ll_vec[i]+(ll_vec[i+1]-ll_vec[i])/(1-aitken[i])
    stop = ll_inf-ll_vec[i+1]
    if(stop>0 & stop < 1e-6) {break}
  }
}

beta_mat_MC[, , m] = beta_mat
sigma_vec_MC[m,] = sigma_vec
theta_vec_MC[m,] = theta_vec
pi_vec_MC[m,] = pi_vec

# Performance evaluation for CMixR
my.clust = rep(NA,n)
for (a in 1:n) {
  if (zhat_mat[a,1] > zhat_mat[a,2]) {my.clust[a] = 1} else {my.clust[a] = 2}
}
ARI_MC[m] = adjustedRandIndex(c(rep(1,50),rep(2,50)),my.clust)
```

```
  MAP_MC[,,m] = table(c(rep(1,50),rep(2,50)),my.clust)
}
```

## A.2 Crab data set

```
rm(list = ls())
library(flexmix)
library(mclust)
library(mixture)
library(pgmm)
library(copula)

library(MASS)
data("crabs")

y <- as.matrix(crabs[,c(4,5,7)])
x <- as.matrix(crabs[,c(6,8)])
class <- crabs$sex


par(mfrow = c(1, 3))

hist(y[,1], prob = TRUE, ylim = c(0,0.12), xlab = "Frontal Lobe Size (mm)",
     main = "Density Plot of Y1: FL")
lines(density(y[,1]), col = 4, lwd = 2)

hist(y[,2], prob = TRUE, ylim = c(0,0.15), xlab = "Rear Width (mm)",
     main = "Density Plot of Y2: RW")
lines(density(y[,2]), col = 4, lwd = 2)

hist(y[,3], prob = TRUE, ylim = c(0,0.05), xlab = "Carapace Width (mm)",
     main = "Density Plot of Y3: CW")
lines(density(y[,3]), col = 4, lwd = 2)

# EM algorithm
## E-step
E_step=function(y, x, beta_mat, sigma_vec, theta_vec, pi_vec){
  y1 = y[,1]; y2 = y[,2]; y3 = y[,3]
  x1 = x[,1]; x2 = x[,2]
  beta11 = beta_mat[,1]    #beta for Y1, g=1
  beta12 = beta_mat[,2]    #beta for Y1, g=2
  beta21 = beta_mat[,3]    #beta for Y2, g=1
  beta22 = beta_mat[,4]    #beta for Y2, g=2
  beta31 = beta_mat[,5]    #beta for Y3, g=1
  beta32 = beta_mat[,6]    #beta for Y3, g=2

  mu11 = beta11[1] + beta11[2]*x1 + beta11[3]*x2   #Y1, g=1
  mu12 = beta12[1] + beta12[2]*x1 + beta12[3]*x2   #Y1, g=2
  mu21 = beta21[1] + beta21[2]*x1 + beta21[3]*x2   #Y2, g=1
  mu22 = beta22[1] + beta22[2]*x1 + beta22[3]*x2   #Y2, g=2
  mu31 = beta31[1] + beta31[2]*x1 + beta31[3]*x2   #Y3, g=1
  mu32 = beta32[1] + beta32[2]*x1 + beta32[3]*x2   #Y3, g=2

  F11 = pnorm(y1, mean = mu11, sd = sigma_vec[1])   #Y1, g=1
  F12 = pnorm(y1, mean = mu12, sd = sigma_vec[2])   #Y1, g=2
  F21 = pnorm(y2, mean = mu21, sd = sigma_vec[3])   #Y2, g=1
  F22 = pnorm(y2, mean = mu22, sd = sigma_vec[4])   #Y2, g=2
  F31 = pnorm(y3, mean = mu31, sd = sigma_vec[5])   #Y3, g=1
  F32 = pnorm(y3, mean = mu32, sd = sigma_vec[6])   #Y3, g=2

  cop1 = frankCopula(theta_vec[1], dim = 3)
  cop2 = frankCopula(theta_vec[2], dim = 3)
  c1 = dCopula(cbind(F11,F21,F31), cop1)   #copula density of (Y1,Y2,Y3), g=1
  c2 = dCopula(cbind(F12,F22,F32), cop2)   #copula density of (Y1,Y2,Y3), g=2

  f11 = dnorm(y1, mean = mu11, sd = sigma_vec[1])  #marginal density of Y1, g=1
  f12 = dnorm(y1, mean = mu12, sd = sigma_vec[2])  #marginal density of Y1, g=2
  f21 = dnorm(y2, mean = mu21, sd = sigma_vec[3])  #marginal density of Y2, g=1
  f22 = dnorm(y2, mean = mu22, sd = sigma_vec[4])  #marginal density of Y2, g=2
  f31 = dnorm(y3, mean = mu31, sd = sigma_vec[5])  #marginal density of Y3, g=1
  f32 = dnorm(y3, mean = mu32, sd = sigma_vec[6])  #marginal density of Y3, g=2
  comp_dens1 = c1 * pi_vec[1] * f11 * f21 * f31
  comp_dens2 = c2 * pi_vec[2] * f12 * f22 * f32
  sum_dens = comp_dens1 + comp_dens2
  z1.vec = comp_dens1 / sum_dens
  z2.vec = comp_dens2 / sum_dens
  zhat_mat = cbind(z1.vec,z2.vec)
  return(zhat_mat)
}

## M-step1 for pi1 and pi2, returns pi_vec
M_step1 = function(zhat_mat){
  n = nrow(zhat_mat)
  sum_z1 = sum(zhat_mat[,1])
  sum_z2 = sum(zhat_mat[,2])
  pi1 = sum_z1/n
  pi2 = sum_z2/n
  pi_vec = c(pi1,pi2)
  return(pi_vec)
}

optFunc = function(param_g, theta_g, zhat_g, y1, y2, y3, x1, x2){

  beta_1g = param_g[1:3]
  beta_2g = param_g[4:6]
  beta_3g = param_g[7:9]
  sigma_1g = param_g[10]
  sigma_2g = param_g[11]
  sigma_3g = param_g[12]
  mu_1g = beta_1g[1] + beta_1g[2]*x1 + beta_1g[3]*x2
  mu_2g = beta_2g[1] + beta_2g[2]*x1 + beta_2g[3]*x2
  mu_3g = beta_3g[1] + beta_3g[2]*x1 + beta_3g[3]*x2
  f1g = dnorm(y1, mean = mu_1g, sd = sigma_1g, log = TRUE)
  f2g = dnorm(y2, mean = mu_2g, sd = sigma_2g, log = TRUE)
  f3g = dnorm(y3, mean = mu_3g, sd = sigma_3g, log = TRUE)
  F1g = pnorm(y1, mean = mu_1g, sd = sigma_1g)      #Y1
  F2g = pnorm(y2, mean = mu_2g, sd = sigma_2g)      #Y2
  F3g = pnorm(y3, mean = mu_3g, sd = sigma_3g)      #Y2

  cop_g = frankCopula(theta_g, dim = 3)
  logcg = dCopula(cbind(F1g,F2g,F3g), cop_g, log = TRUE)
  zlog_cg_fg = zhat_g * (logcg + f1g + f2g + f3g)
  arg = sum(zlog_cg_fg)
  return(arg)
}

## M-step2: CM-step1 and CM-step2
CM_step1 = function(y, x, zhat_mat, beta_mat_initial,
                    sigma_vec_initial, theta_vec){

  y1 = y[,1]; y2 = y[,2]; y3 = y[,3]
  x1 = x[,1]; x2 = x[,2]

  beta_mat = matrix(NA, 3, 6)
  sigma_vec = rep(NA, 6)

  # g=1
```

```
init1 = c(beta_mat_initial[,1], beta_mat_initial[,3], beta_mat_initial[,5],
          sigma_vec_initial[1], sigma_vec_initial[3], sigma_vec_initial[5])
res1 = optim(init1, optFunc, theta_g = theta_vec[1], zhat_g=zhat_mat[,1],
             y1=y1, y2=y2, y3=y3, x1=x1, x2=x2, method = "L-BFGS-B",
             lower = c(-Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf,
                       .Machine$double.eps, .Machine$double.eps,
                       .Machine$double.eps),
             upper = c( Inf,  Inf,  Inf,  Inf,  Inf,  Inf,  Inf,  Inf,  Inf,
                       Inf, Inf, Inf),
             control = list(fnscale = -1))
beta_mat[,1] = res1$par[1:3]
beta_mat[,3] = res1$par[4:6]
beta_mat[,5] = res1$par[7:9]
sigma_vec[1] = res1$par[10]
sigma_vec[3] = res1$par[11]
sigma_vec[5] = res1$par[12]

# g=2
init2 = c(beta_mat_initial[,2], beta_mat_initial[,4], beta_mat_initial[,6],
         sigma_vec_initial[2], sigma_vec_initial[4], sigma_vec_initial[6])
res2 = optim(init2, optFunc, theta_g = theta_vec[2], zhat_g=zhat_mat[,2],
             y1=y1, y2=y2, y3=y3, x1=x1, x2=x2, method = "L-BFGS-B",
             lower = c(-Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf, -Inf,
                       -Inf,.Machine$double.eps,
                       .Machine$double.eps, .Machine$double.eps),
             upper = c( Inf,  Inf,  Inf,  Inf,  Inf,  Inf,  Inf,  Inf,
                        Inf, Inf, Inf, Inf),
             control = list(fnscale = -1))
beta_mat[,2] = res2$par[1:3]
beta_mat[,4] = res2$par[4:6]
beta_mat[,6] = res2$par[7:9]
sigma_vec[2] = res2$par[10]
```

```
  sigma_vec[4] = res2$par[11]
  sigma_vec[6] = res1$par[12]

  return(list("beta_mat" = beta_mat, "sigma_vec" = sigma_vec))
}

optFunc2 = function(theta_g, F1g, F2g, F3g, zhat_g){
  cop_g = frankCopula(theta_g, dim = 3)
  logcg = dCopula(cbind(F1g,F2g,F3g), cop_g, log = TRUE)
  zlogcg = zhat_g * logcg
  arg = sum(zlogcg)
  return(arg)
}

CM_step2 = function(y, x, zhat_mat, beta_mat, sigma_vec, theta_vec_initial){

  y1 = y[,1]; y2 = y[,2]; y3 = y[,3]
  x1 = x[,1]; x2 = x[,2]
  beta11 = beta_mat[,1]    #beta for Y1, g=1
  beta12 = beta_mat[,2]    #beta for Y1, g=2
  beta21 = beta_mat[,3]    #beta for Y2, g=1
  beta22 = beta_mat[,4]    #beta for Y2, g=2
  beta31 = beta_mat[,5]    #beta for Y3, g=1
  beta32 = beta_mat[,6]    #beta for Y3, g=2

  mu11 = beta11[1] + beta11[2]*x1 + beta11[3]*x2   #Y1, g=1
  mu12 = beta12[1] + beta12[2]*x1 + beta12[3]*x2   #Y1, g=2
  mu21 = beta21[1] + beta21[2]*x1 + beta21[3]*x2   #Y2, g=1
  mu22 = beta22[1] + beta22[2]*x1 + beta22[3]*x2   #Y2, g=2
  mu31 = beta31[1] + beta31[2]*x1 + beta31[3]*x2   #Y3, g=1
  mu32 = beta32[1] + beta32[2]*x1 + beta32[3]*x2   #Y3, g=2
  F11 = pnorm(y1, mean = mu11, sd = sigma_vec[1])   #Y1, g=1
  F12 = pnorm(y1, mean = mu12, sd = sigma_vec[2])   #Y1, g=2
  F21 = pnorm(y2, mean = mu21, sd = sigma_vec[3])   #Y2, g=1
  F22 = pnorm(y2, mean = mu22, sd = sigma_vec[4])   #Y2, g=2
  F31 = pnorm(y3, mean = mu31, sd = sigma_vec[5])   #Y3, g=1
  F32 = pnorm(y3, mean = mu32, sd = sigma_vec[6])   #Y3, g=2

  theta_vec = rep(NA, 2)

  # g=1
  res1 = optim(theta_vec_initial[1], optFunc2, F1g=F11, F2g=F21, F3g=F31,
               zhat_g=zhat_mat[,1], method = "L-BFGS-B",
               lower = .Machine$double.eps,
               upper = Inf,
               control = list(fnscale = -1))
  theta_vec[1] = res1$par

  # g=2
  res2 = optim(theta_vec_initial[2], optFunc2, F1g=F12, F2g=F22, F3g=F32,
               zhat_g=zhat_mat[,2], method = "L-BFGS-B",
               lower = .Machine$double.eps,
               upper = Inf,
               control = list(fnscale = -1))
  theta_vec[2] = res2$par

  return(theta_vec)
}

loglik = function(y, x, pi_vec, beta_mat, sigma_vec, theta_vec){
  y1 = y[,1]; y2 = y[,2]; y3 = y[,3]
  x1 = x[,1]; x2 = x[,2]
  beta11 = beta_mat[,1]    #beta for Y1, g=1
```

```
beta12 = beta_mat[,2]    #beta for Y1, g=2
beta21 = beta_mat[,3]    #beta for Y2, g=1
beta22 = beta_mat[,4]    #beta for Y2, g=2
beta31 = beta_mat[,5]    #beta for Y3, g=1
beta32 = beta_mat[,6]    #beta for Y3, g=2

mu11 = beta11[1] + beta11[2]*x1 + beta11[3]*x2   #Y1, g=1
mu12 = beta12[1] + beta12[2]*x1 + beta12[3]*x2   #Y1, g=2
mu21 = beta21[1] + beta21[2]*x1 + beta21[3]*x2   #Y2, g=1
mu22 = beta22[1] + beta22[2]*x1 + beta22[3]*x2   #Y2, g=2
mu31 = beta31[1] + beta31[2]*x1 + beta31[3]*x2   #Y3, g=1
mu32 = beta32[1] + beta32[2]*x1 + beta32[3]*x2   #Y3, g=2

F11 = pnorm(y1, mean = mu11, sd = sigma_vec[1])  #Y1, g=1
F12 = pnorm(y1, mean = mu12, sd = sigma_vec[2])  #Y1, g=2
F21 = pnorm(y2, mean = mu21, sd = sigma_vec[3])  #Y2, g=1
F22 = pnorm(y2, mean = mu22, sd = sigma_vec[4])  #Y2, g=2
F31 = pnorm(y3, mean = mu31, sd = sigma_vec[5])  #Y3, g=1
F32 = pnorm(y3, mean = mu32, sd = sigma_vec[6])  #Y3, g=2

cop1 = frankCopula(theta_vec[1], dim = 3)
cop2 = frankCopula(theta_vec[2], dim = 3)
c1 = dCopula(cbind(F11,F21,F31), cop1)  # copula density of (Y1,Y2,Y3), g=1
c2 = dCopula(cbind(F12,F22,F32), cop2)  # copula density of (Y1,Y2,Y3), g=2

comp1 = pi_vec[1] * c1 * dnorm(y1, mean = mu11, sd = sigma_vec[1])
                       * dnorm(y2, mean = mu21, sd = sigma_vec[3])
                       * dnorm(y3, mean = mu31, sd = sigma_vec[5])
comp2 = pi_vec[2] * c2 * dnorm(y1, mean = mu12, sd = sigma_vec[2])
                       * dnorm(y2, mean = mu22, sd = sigma_vec[4])
                       * dnorm(y3, mean = mu32, sd = sigma_vec[6])
sum_comp = comp1 + comp2
```

```
  loglik = sum(log(sum_comp))
  return(loglik)
}


## EM algorithm
max.iter = 1000

ll_vec = c(-Inf, rep(NA, max.iter-1))
aitken = rep(NA, max.iter)
beta_mat=matrix(1, nrow=3,ncol=6)
sigma_vec=rep_len(2,6)
theta_vec=rep_len(0.1,2)

y.kmeans <- kmeans(cbind(y,x), 2, iter.max = 100, nstart = 30)
y.kmeans.cluster <- y.kmeans$cluster
zhat_mat = cbind(y.kmeans.cluster-1, 2-y.kmeans.cluster)
pi_vec = M_step1(zhat_mat)

cmstep1 = CM_step1(y, x, zhat_mat, beta_mat_initial = beta_mat,
                   sigma_vec_initial = sigma_vec, theta_vec = theta_vec)
beta_mat = cmstep1$beta_mat
sigma_vec = cmstep1$sigma_vec
theta_vec = CM_step2(y, x, zhat_mat, beta_mat, sigma_vec,
                     theta_vec_initial=theta_vec)


for(i in 1:max.iter){
  zhat_mat = E_step(y, x, beta_mat, sigma_vec, theta_vec, pi_vec)
  pi_vec = M_step1(zhat_mat)
  cmstep1 = CM_step1(y, x, zhat_mat, beta_mat_initial = beta_mat,
                     sigma_vec_initial = sigma_vec, theta_vec)
  beta_mat = cmstep1$beta_mat
  sigma_vec = cmstep1$sigma_vec
  theta_vec = CM_step2(y, x, zhat_mat, beta_mat, sigma_vec,
                     theta_vec_initial = theta_vec)
  ll_vec[i+1] = loglik(y, x, pi_vec, beta_mat, sigma_vec, theta_vec)

  if(i>2){
    aitken[i] = (ll_vec[i+1]-ll_vec[i])/(ll_vec[i]-ll_vec[i-1])
    ll_inf = ll_vec[i]+(ll_vec[i+1]-ll_vec[i])/(1-aitken[i])
    stop = ll_inf-ll_vec[i+1]
    if(stop>0 & stop < 1e-6) {break}
  }
}

# Performance evaluation
my.clust = rep(NA,200)
for (a in 1:200) {
  if (zhat_mat[a,1] > zhat_mat[a,2]) {my.clust[a] = 1} else {my.clust[a] = 2}
}
ARI = adjustedRandIndex(as.numeric(class),my.clust)
MAP = table(as.numeric(class),my.clust)

flex.clust = stepFlexmix(~x[,1]+x[,2], k = 2, nrep = 10,
                     model = list(FLXMRglm(y[,1]~x[,1]+x[,2]),
                                  FLXMRglm(y[,2]~x[,1]+x[,2]),
                                  FLXMRglm(y[,3]~x[,1]+x[,2])))
flex.clust = flex.clust@cluster
ARI_flex = adjustedRandIndex(as.numeric(class), flex.clust)
MAP_flex = table(as.numeric(class), flex.clust)

pgmm.clust = pgmmEM(scale(cbind(x,y)), rG=2:2, zstart=1, loop=10, relax = TRUE)
pgmm.clust = pgmm.clust$map
```

```
MAP_pgmm = table(as.numeric(class), pgmm.clust)
ARI_pgmm = adjustedRandIndex(as.numeric(class), pgmm.clust)

gmm.clust = Mclust(cbind(y,x), 2)
gmm.clust = gmm.clust$classification
ARI_mclust = adjustedRandIndex(as.numeric(class), gmm.clust)
MAP_mclust = table(as.numeric(class), gmm.clust)

mix.zhat = pcm(cbind(y,x), G=2:2)
mix.zhat = mix.zhat$best_model$map
ARI_mix = adjustedRandIndex(as.numeric(class), mix.clust)
MAP_mix = table(as.numeric(class), mix.clust)
```