2023-05-01

Nonparametric Estimation of Elliptical Copulas

Panfeng Liang
University of Texas at El Paso

NONPARAMETRIC ESTIMATION OF ELLIPTICAL COPULAS

PANFENG LIANG

Doctoral Program in Computational Science

APPROVED:

Ori Rosen, Ph.D., Chair

Vanessa Lougheed, Ph.D.

Michael Pokojovy, Ph.D.

James M. Wood, Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School

NONPARAMETRIC ESTIMATION OF ELLIPTICAL COPULAS

by

PANFENG LIANG, M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Computational Science

THE UNIVERSITY OF TEXAS AT EL PASO

May 2023

Abstract

Elliptical copulas provide flexibility in modeling the dependence structure of a random vector. They are often parameterized with a correlation matrix and a scalar function, called generator. The estimation of the generator can be challenging, because it is a functional parameter. In this dissertation, we provide a rigorous approach to estimating the generator in a Bayesian framework, which is simpler, more robust, and outperforms existing estimation methods in the literature. Based on the proposed framework in this dissertation, other researchers may modify the model for other types of generators in their own research.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Copula

Suppose we are asked to compare the two bivariate data sets in Figure 1.1 in terms of the underlying dependence structures between the two respective variables. By eyeballing the scatter plots, the two data sets seem very different. The $\boldsymbol{X} = (X_1, X_2)$ data are more concentrated in the center while the $\boldsymbol{Y} = (Y_1, Y_2)$ data are more dense in the bottom left corner. Just by observing the two scatter plots, it is difficult to notice the similarity between their dependence structures, since the linear correlation between X_1 and X_2 (or between Y_1 and Y_2) depends on their marginal distributions.

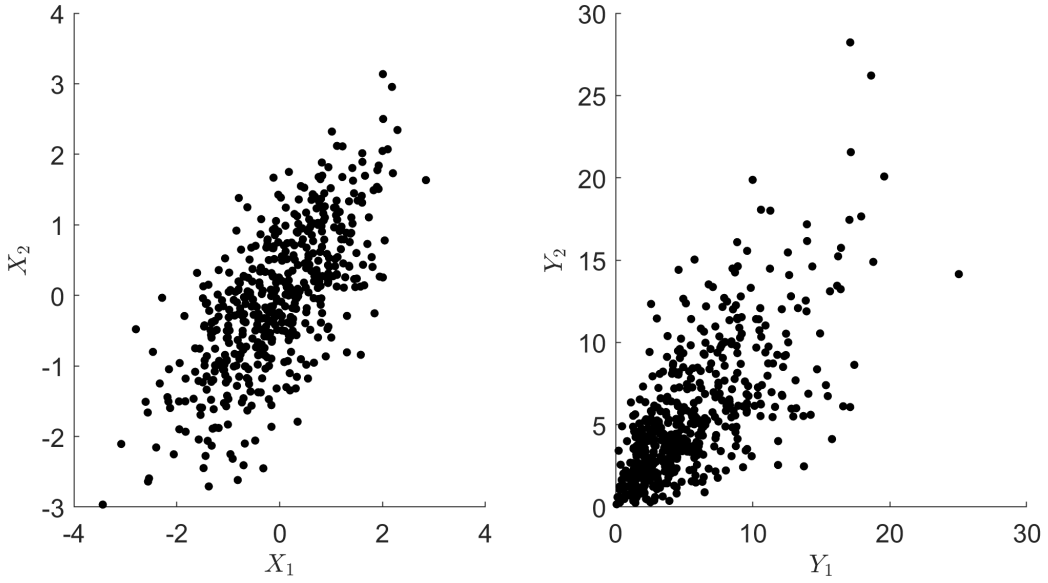


Figure 1.1: Scatter plots of $n = 500$ observations from two bivariate distributions.

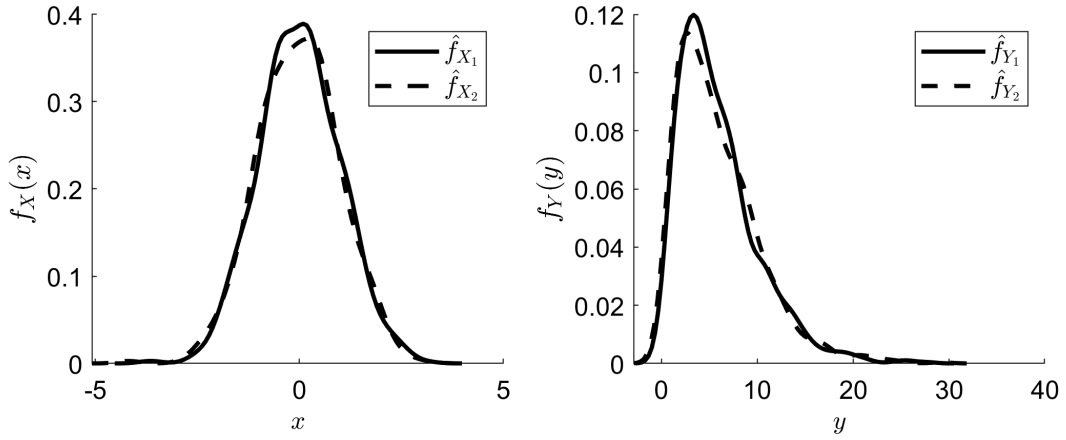


Figure 1.2: Kernel density estimations of (X_1, X_2) and of (Y_1, Y_2).

We obtain estimates of the marginal probability densities of (X_1, X_2) and (Y_1, Y_2) by kernel density estimation (KDE) and plot the results in Figure 1.2. We see that variables X_1 and X_2 have the same marginal distribution, and so do Y_1 and Y_2. But $\boldsymbol{X}$ and $\boldsymbol{Y}$ have very different marginals. The marginal distribution of $\boldsymbol{X}$ is bell-shaped whereas that of $\boldsymbol{Y}$ is right-skewed. The difference in the marginal distributions makes the comparison between the dependence structures of these two bivariate distributions difficult. If we could transform $\boldsymbol{X}$ and $\boldsymbol{Y}$, making their marginal distributions identical, it would be much easier to compare their dependence structures.

In the univariate case, the Probability Integral Transform (PIT) can achieve this goal. The PIT can transform any continuous distribution into the standard uniform distribution.

Lemma 1 (Probability Integral Transform) *Let F be the cumulative distribution function (CDF) of a continuous random variable X, that is, $X \sim F$. Then $U = F(X)$ follows a standard uniform distribution, that is, $U \sim \mathcal{U}(0, 1)$. The transformation $U = F(X)$ is called the Probability Integral Transform.*

The PIT can be generalized to the bivariate or multivariate cases. Genest (2018) refers to the multivariate PIT as the Copula Transform.

Theorem 2 (Copula Transform) *Let H be the joint distribution function of a d-dimensional random vector $\boldsymbol{X} = (X_1, \ldots, X_d)$ and $F_1, \ldots, F_d$ the continuous marginal CDFs. Then*

$$U_1 = F_1(X_1) \sim \mathcal{U}(0,1), \ldots, U_d = F_d(X_d) \sim \mathcal{U}(0,1),$$

and

$$\boldsymbol{U} = (U_1, \ldots, U_d) \sim C,$$

where C is the copula of H, i.e., a multivariate distribution with standard uniform marginals.

The copula transform converts the margins of a multivariate distribution into the standard uniform distribution. The transformed random vector $\boldsymbol{U} = (U_1, \ldots, U_d)$ follows a special multivariate distribution called copula.

If we apply the copula transform to a data set drawn from a multivariate distribution H, the transformed data set can be viewed as drawn from its copula. The copula transform standardizes the margins of the data sets, which reveals the dependence structure. Figure 1.3 is the scatter plots after applying the copula transform to $\boldsymbol{X}$ and $\boldsymbol{Y}$. We can easily tell that their dependence structures are exactly the same.

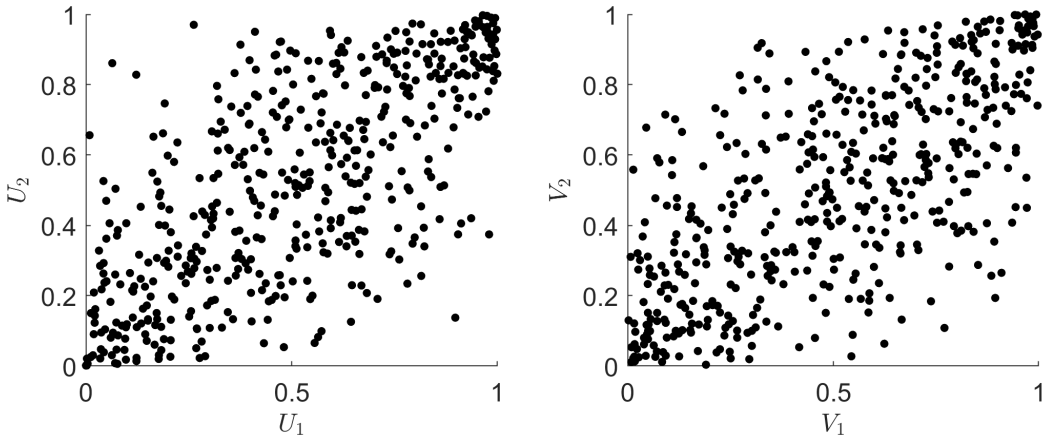


Figure 1.3: Scatter plots of $\boldsymbol{X}$ (left) and $\boldsymbol{Y}$ (right) after applying the copula transform, revealing the dependence structure.

The above example shows that a multivariate distribution has an underlying copula, which captures the dependence structure between the random variables. This copula can

be obtained through the copula transform. According to Sklar (1959), a multivariate distribution with continuous marginals can be uniquely defined by its copula and marginal distributions.

Theorem 3 (Sklar's theorem) *Given a vector* $\boldsymbol{Y} = (Y_1, \ldots, Y_d)$ *of* $d \geq 2$ *continuous random variables, its joint CDF can be represented by*

$$H(y_1, \ldots y_d) = C\big(F_1(y_1), \ldots F_d(y_d)\big), \tag{1.1}$$

where C *is a copula and* $F_1, \ldots, F_d$ *are marginal CDFs. If the* $F_1, \ldots, F_d$ *are continuous, then* C *is unique.*

The joint probability density function (PDF) of $\boldsymbol{Y}$ can be obtained by differentiating the CDF in Equation (1.1) through the chain rule:

$$\begin{aligned} h(y_1, \ldots, y_d) &= \frac{\partial^d H(y_1, \ldots, y_d)}{\partial y_1 \cdots \partial y_d} \\ &= \frac{\partial^d H(y_1, \ldots, y_d)}{\partial F_1(y_1) \cdots \partial F_d(y_d)} \times \prod_{i=1}^{d} \frac{\partial F_i(y_i)}{\partial y_i} \\ &= c\big(F_1(y_1), \ldots, F_d(y_d)\big) \cdot f_1(y_1) \cdot \cdots \cdot f_d(y_d), \end{aligned}$$

where c is the PDF of the copula and $f_i, i = 1, \ldots, d$, are the marginal PDFs of Y_i.

1.2 Archimedean Copulas

Archimedean copulas have been extensively used in both research and applications, such as Joe (2014), Genest and MacKay (1986), Genest and Rivest (1993), McNeil and Nešlehová (2009), McNeil (2008) and Müller and Scarsini (2005). This type of copulas has two main advantages. First, most Archimedean copulas have an explicit formula. Second, they only have a single parameter controlling the strength of dependence regardless of their dimensions, which may be advantageous in modeling the dependence of high-dimensional data.

A copula C is called Archimedean if it admits the representation

$$C(u_1, \ldots, u_d) = \psi^{-1}\Big(\psi(u_1) + \ldots + \psi(u_d)\Big),$$

where ψ is a continuous, strictly decreasing function from $[0, 1]$ to $[0, \infty)$ such that $\psi(1) = 0$.

Common Archimedean copulas include the Clayton copula, the Frank copula and the Gumbel copula.

- The **Clayton copula** has the copula function (CDF)

$$C(u_1, \ldots, u_d \mid \theta) = \left(-1 + \sum_{i=1}^{d} u_i^{-\theta}\right)^{-\frac{1}{\theta}}, \ \theta \in (0, \infty)$$

  with Kendall's tau given by $\tau = \theta/(\theta + 2)$.

- The **Frank copula** has the copula function

$$C(u_1, \ldots, u_d \mid \theta) = -\frac{1}{\theta} \log\left[1 + \frac{\prod_{i=1}^{d} (\exp(-\theta u_i) - 1)}{\exp(-\theta) - 1}\right], \ \theta \in (-\infty, \infty) \setminus \{0\},$$

  with Kendall's tau given by $\tau = 1 + \frac{4}{\theta}(D_1(\theta) - 1)$, where $D_1(\theta) = \frac{1}{\theta}\int_0^\theta \frac{t}{\exp(t)-1} dt$ is the Debye function.

- The **Gumbel copula** has the copula function

$$C(u_1, \ldots, u_d \mid \theta) = \exp\left\{-\left[\prod_{i=1}^{d} (-\log(u_i))^{\theta}\right]^{\frac{1}{\theta}}\right\}, \ \theta \in [1, \infty),$$

  with Kendall's tau given by $\tau = 1 - 1/\theta$.

Figure 1.4 consists of scatter plots of 500 observations from the three bivariate Archimedean copulas with Kendall's $\tau = 0.7$.

1.3 The Inversion Method

Unlike Archimedean copulas, many other copulas, such as the Gaussian copula, do not have simple analytical forms for their copula functions. In such cases, a common way to construct the copula is to use the Inversion Method.

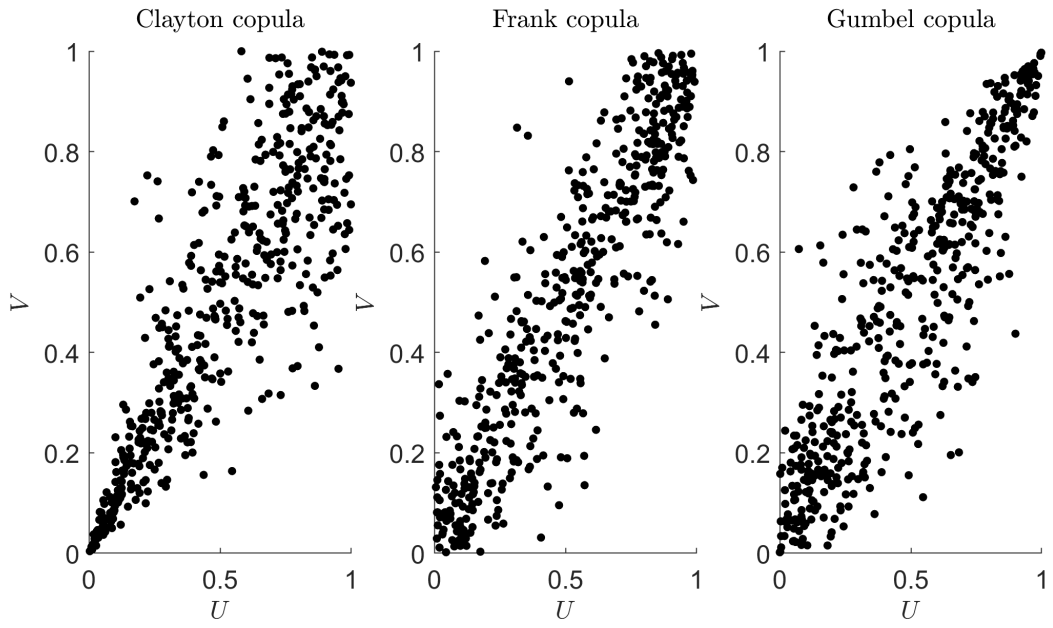


Figure 1.4: Scatter plots of $n = 500$ independent observations from bivariate Archimedean copulas with Kendall's $\tau = 0.7$.

Given a multivariate distribution F and marginal distributions $F_1, \ldots, F_d$, its copula function C can be obtained by inverting Equation (1.1) in Sklar's theorem

$$C(u_1, \ldots, u_d) = F\left[F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d)\right], \; u_1, \ldots, u_d \in (0, 1), \tag{1.2}$$

where $F_1^{-1}, \ldots, F_d^{-1}$ are the inverses of the marginal CDFs.

Equation (1.2) can be differentiated to obtain the copula density

$$c(u_1, \ldots, u_d) = \frac{f\left[F_1^{-1}(u_1), \ldots, F_d^{-1}(u_d)\right]}{\prod_{k=1}^{d} f_k\left[F_k^{-1}(u_k)\right]}, \; u_1, \ldots, u_d \in (0, 1), \tag{1.3}$$

where f is the probability density of F and $f_1, \ldots, f_d$ are the marginal probability densities.

1.4 Gaussian Copula

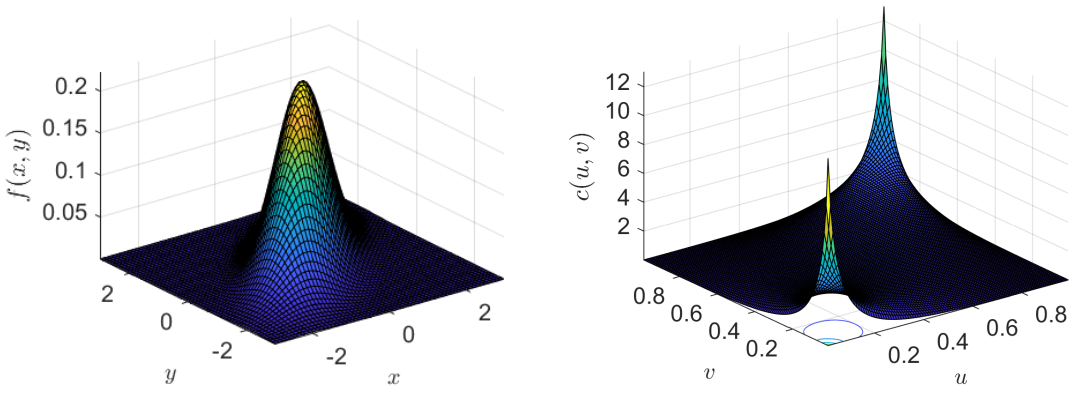
Gaussian copulas are another important family of copulas beside Archimedean copulas, related to multivariate Gaussian distributions. According to the Sklar's Theorem, a multivariate Gaussian distribution can be decomposed into a Gaussian copula and a collection

of marginal distributions that are also Gaussian. It is well known that a multivariate distribution with Gaussian margins is not necessarily a multivariate Gaussian distribution, because its copula might not be a Gaussian copula.

Gaussian copulas can be derived from multivariate Gaussian distributions by the inversion method. The resulting copula function can be written as

$$C^{\text{Gauss}}(u_1, \ldots, u_d \mid \Omega) = \Phi_\Omega \left[\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)\right], \; u_1, \ldots, u_d \in (0, 1),$$

where Φ^{-1} is the inverse CDF of a standard normal distribution and Φ_Ω is the joint CDF of a multivariate Gaussian distribution with mean vector zero and covariance matrix equal to the correlation matrix Ω. Figure 1.5 demonstrates the density of a bivariate Gaussian distribution $\mathcal{BVN}(\mathbf{0}, \rho = 0.7)$ and its underlying copula. Gaussian copulas belong to the family of elliptical copulas, which are the topic of this dissertation. We introduce elliptical copulas in subsequent sections of this chapter.



(a) Gaussian distribution (b) Gaussian copula

Figure 1.5: Surface plots of a bivariate Gaussian distribution and its copula.

1.5 Elliptical Distribution

Elliptical copulas are derived from elliptical distributions via the inversion method, which is why we introduce elliptical distributions first. A d-dimensional continuous random vector

$\boldsymbol{X} = (X_1, ..., X_d)$ has an elliptical distribution $\mathcal{E}_d(\boldsymbol{\mu}, \Sigma, g)$ if its density function takes the form of

$$f(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma, g) = \det(\Sigma)^{-1/2} g\left((\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right), \tag{1.4}$$

where

- $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean vector;
- Σ is the covariance matrix;
- g is the generator, which is a non-negative function defined on $[0, \infty)$.

Since the dissertation is focused on elliptical copulas, without loss of generality, we only consider elliptical distributions $\mathcal{E}(\mathbf{0}, \Omega, g)$, where Ω is a Pearson correlation matrix. In this case, the elliptical distribution has identical margins. In later chapters, we refer to the Pearson correlation matrix simply as a correlation matrix.

Stochastic form:

If a d-variate random vector $\boldsymbol{X}$ has elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$, it can be expressed in the following stochastic representation (see Fang et al. (2002) and Fang et al. (2005))

$$\boldsymbol{X} = RA\boldsymbol{U}, \tag{1.5}$$

where

- A is a Cholesky factor from the decomposition of Ω ($AA^T = \Omega$);
- $\boldsymbol{U}$ is a random vector uniformly distributed on the unit sphere of $\mathbb{R}^d$;
- R is a continuous nonnegative random variable with density function

$$h(r) = \frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1} g(r^2); \tag{1.6}$$

- R and $\boldsymbol{U}$ are independent of each other.

A typical example of an elliptical distribution is the multivariate Gaussian distribution, whose generator takes the form of $g(t) \propto \exp(-1/2)$. Figure 1.6 shows a scatter plot of 2000 observations from the bivariate Gaussian distribution, which forms an ellipse. This is the motivation for the term "elliptical" distributions.

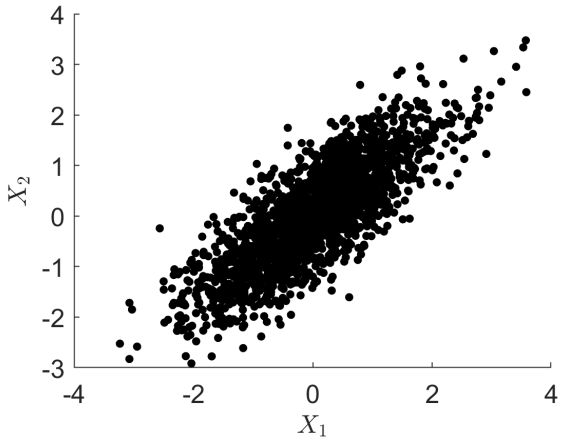


Figure 1.6: Scatter plot of $n = 2000$ observations from a bivariate normal distribution with $\boldsymbol{\mu} = (0, 0)$ and correlation $\rho = 0.8$.

Table 1.1, taken from Lemonte and Patriota (2011), includes common elliptical distributions and their generators. For a detailed introduction to elliptical distributions, see Fang and Zhang (1990).

1.6 Elliptical Copulas

Elliptical copulas, also called meta-elliptical copulas, were originally introduced by Fang et al. (2002). These copulas induce a dependence structure of elliptical distributions, the same way Gaussian copulas describe the dependence in multivariate Gaussian distributions.

The density of an elliptical copula can be derived from the density of an elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$ via the inversion method,

$$c(\boldsymbol{u} \mid \Omega, g) = \frac{\det(\Omega)^{-1/2} g\big(Q_1(\boldsymbol{u})^T \Omega^{-1} Q_1(\boldsymbol{u})\big)}{\prod_{j=1}^{d} f_1\big(Q_1(u_j)\big)}, \quad \boldsymbol{u} \in [0,1]^d, \tag{1.7}$$

Table 1.1: Different elliptical distributions and their generators.

Elliptical distribution	Distribution of R^2	Generating function g(t)
Normal	$R^2 \sim \chi^2_{(d)}$	c[1]$\exp(-t/2)$
Student	$R^2/d \sim F(d, \nu)$	$c\,(1 + t/\nu)^{(d+\nu)/2}$
Pearson type II	$R^2 \sim \text{Beta}(d/2, \nu + 1)$	$c\,(1 - t)^{\nu}$, $t \in [-1, 1]$, $\nu > -1$
Logistic	$R^2 \sim *$[2]	$c\,e^{-t}/(1 + e^{-t})^2$

[1] The c is normalizing constant.

[2] The distribution of R^2 is not a standard distribution and its kernel is $(r^2)^{(d-2)/2}e^{-r^2}/(1 + e^{-r^2})^2$.

where the numerator is the density of $\mathcal{E}(\mathbf{0}, \Omega, g)$ and f_1 and Q_1 in the denominator are respectively the marginal PDF and the marginal inverse CDF of $\mathcal{E}(\mathbf{0}, \Omega, g)$.

We denote the elliptical copula in (1.7) as $\mathcal{C}(\Omega, g)$, and call $\mathcal{E}(\mathbf{0}, \Omega, g)$ the "associated" elliptical distribution, which means that it is the distribution from which the copula is derived. As we see, the elliptical copula $\mathcal{C}(\Omega, g)$ has exactly the same parameters as its associated elliptical distribution. To some extent, estimating an elliptical copula is equivalent to estimating the associated elliptical distribution.

As mentioned before, a copula describes the dependence structure of a multivariate distribution. When it comes to elliptical copulas, the correlation matrix Ω describes the linear dependence. In this dissertation, we will only focus on the estimation of generators. The correlation matrix is assumed known or its estimate is obtained in a separate procedure.

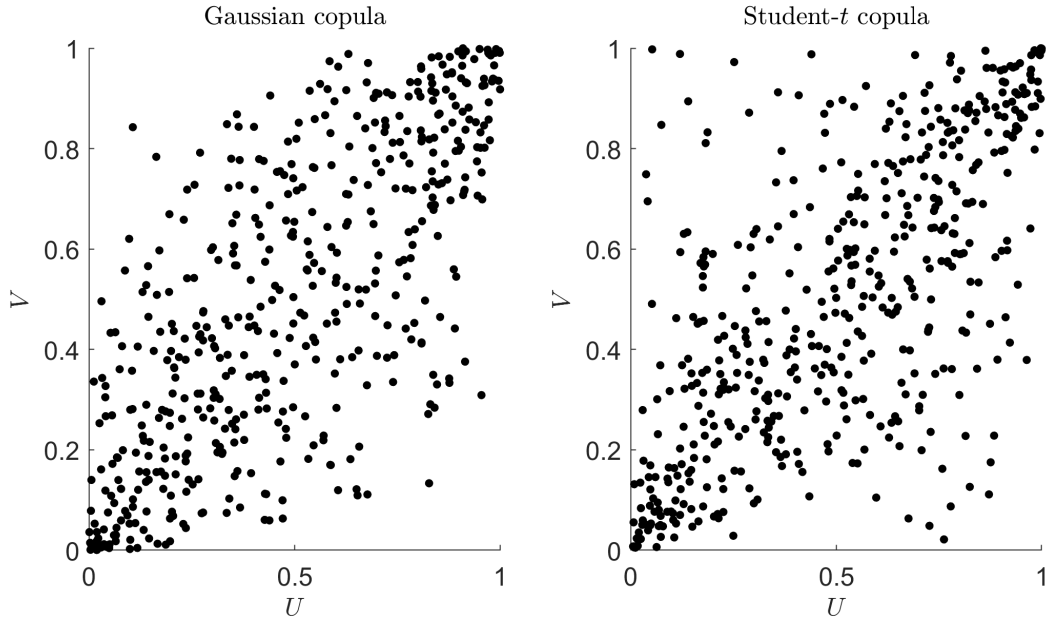


Figure 1.7: Scatter plots of $n = 500$ observations from two elliptical copulas with correlation $\rho = 0.7$.

Chapter 2

The Generator of an Elliptical Copula

2.1 Relation between the Generator and R

An elliptical copula is parameterized by the correlation matrix Ω and the generator g. The generator g can be expressed by the PDF of R. Recall that R is the random variable in the stochastic form of the associated elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$ and the PDF of R is denoted by $h(r)$.

Most of the relevant literature, such as Genest et al. (2007) and Liebscher (2005), mentions the relation between g of $h(r)$ as follows

$$g(t) = \frac{\Gamma(d/2)}{2\pi^{d/2}} t^{(1-d)/2} h(t^{1/2}),\ t > 0 \Longleftrightarrow h(r) = \frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1} g(r^2),\ r > 0. \tag{2.1}$$

The generator can also be expressed through the density of the squared radius R^2. Denoting the squared radius R^2 by Y, and the probability density of Y by f_Y, we have

$$g(t) = \frac{\Gamma(d/2)}{\pi^{d/2}} t^{1-\frac{d}{2}} f_Y(t),\ t > 0 \Longleftrightarrow f_Y(y) = \frac{\pi^{d/2}}{\Gamma(d/2)} y^{\frac{d}{2}-1} g(y),\ r > 0. \tag{2.2}$$

Equations (2.1) and (2.2) are equivalent. In fact, g can also be written in terms of the PDF of other functions of R, such as R^3, $\sqrt{R}$, or R^a, where a is power. Once the PDFs of these random variables are available, $h(r)$ can easily be obtained by a transformation.

The key to understanding the generator is understanding the role of R. The notation R actually means "radius". To explain this, we introduce a special elliptical distribution called spherical distribution.

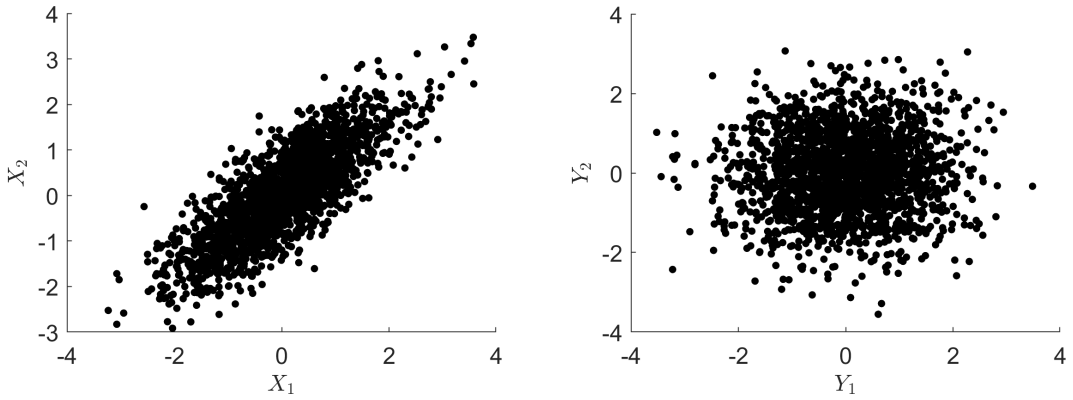
2.2 Spherical Distribution

When $\boldsymbol{\mu} = \mathbf{0}$ and Ω is the identity matrix I_d, the elliptical distribution $\mathcal{E}(\mathbf{0}, I_d, g)$ is called spherical distribution which can be denoted by $\mathcal{SP}(g)$. If a random vector $\boldsymbol{X}$ has elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$, then

$$\boldsymbol{Y} = A^{-1}\boldsymbol{X} \sim \mathcal{SP}(g),$$

where A is the Cholesky factor of Ω. If a random vector $\boldsymbol{Y}$ has spherical distribution $\mathcal{SP}(g)$, then

$$\boldsymbol{X} = A\boldsymbol{Y} \sim \mathcal{E}(\mathbf{0}, \Omega, g).$$



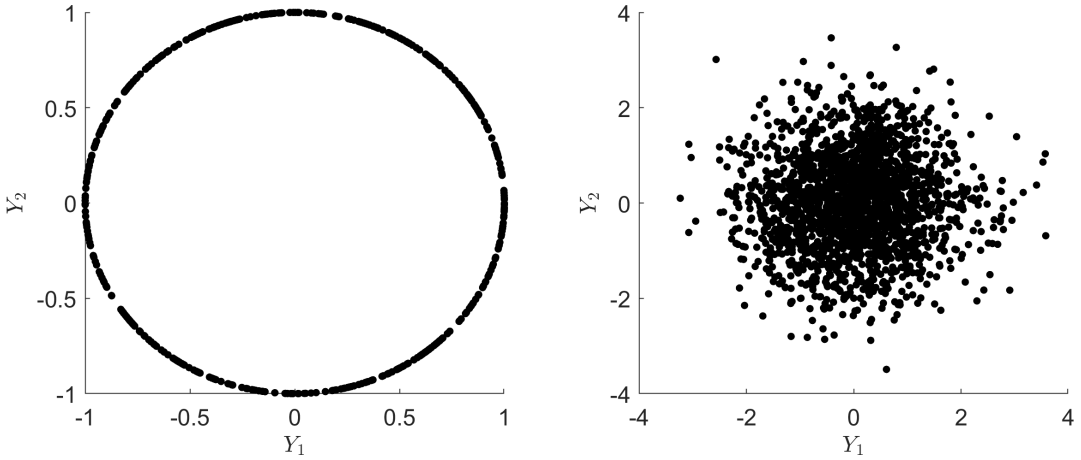
(a) Scatter plot of $\boldsymbol{X} \sim \mathcal{E}(\mathbf{0}, \Omega, g)$.

(b) Scatter plot of $\boldsymbol{Y} = A^{-1}\boldsymbol{X}$.

Figure 2.1: Scatter plots of $n = 2000$ observations from a bivariate elliptical distribution and its associated spherical distribution.

Figure 2.1 shows the relation between $\mathcal{E}(\mathbf{0}, \Omega, g)$ and $\mathcal{SP}(g)$. Instead of an ellipse, a bivariate spherical distribution forms a circle centered at the origin. If a random vector $\boldsymbol{Y}$ follows $\mathcal{SP}(g)$, there is no linear dependence between its entries. The operation $A^{-1}\boldsymbol{X}$ removes the linear correlation from the entries of $\boldsymbol{X}$, whereas the operation $A\boldsymbol{Y}$ restores linear dependence.

If the random vector $\boldsymbol{X}$ has elliptical distribution, it has the stochastic representation $\boldsymbol{X} = RA\boldsymbol{U}$, as shown in Equation (1.5). The vector $\boldsymbol{Y} = A^{-1}\boldsymbol{X}$ has spherical distribution and its stochastic form is $\boldsymbol{Y} = R\boldsymbol{U}$. It is not difficult to see that the R in stochastic form (1.5) is equal to $(\boldsymbol{Y}^T\boldsymbol{Y})^{-1/2}$, which is the Euclidean distance from $\boldsymbol{Y}$ to the origin. The two-dimensional spherical distribution forms a circle and R can be intuitively viewed as its "radius". For a detailed introduction to spherical distributions, see Kelker (1970), Cambanis et al. (1981) and Steerneman and van Perlo-ten Kleij (2005). For now, we only need to remember that R is a radius.



(a) Uniform distribution on a unit sphere.

(b) Spherical distribution.

Figure 2.2: Scatter plots of $n = 500$ observations from a uniform distribution on sphere and a spherical distribution.

A spherical distribution should not be confused with another distribution related to spheres, which is the uniform distribution on a sphere. In most situations, the dimension of a sphere is at least three, but for simplicity, we use the two-dimensional case to demonstrate the difference. Figure 2.2 displays scatter plots of a sample from a uniform distribution on the unit sphere and a sample from a spherical distribution. The difference is obvious: the uniform distribution on a sphere forms a hollow circle while the spherical distribution forms a solid circle. Also, all points in plot (a) are equally distant from the origin while

the distances of the points in plot (b) to the origin are different.

A spherical distribution is determined by $h(r)$, which is the distribution of R in the stochastic form. Equivalently, it can be determined from the generator g because g and $h(r)$ have a one-to-one relation. Drawing a sample from a spherical distribution is a two-step procedure:

1. Draw a radius R from its distribution $h(r)$;

2. Draw a point uniformly from a sphere with radius equal to R.

A spherical distribution is made of uniform distributions on spheres with a random radius distributed with density function $h(r)$. Thus, the solid circle in Figure 2.2 (a) is actually formed by many hollow circles with all different radii. Now we have a better understanding of R and its distribution $h(r)$.

2.3 Nonlinear Dependence

Generalizing our statements in the previous section to the d-dimensional case, if a d-dimensional random vector $\boldsymbol{Y}$ has spherical distribution, then its entries must satisfy

$$\sqrt{Y_1^2 + Y_2^2 + \cdots + Y_d^2} = R \sim h(r). \tag{2.3}$$

The dependence described in Equation (2.3) is nonlinear. In fact, the correlations between entries of $\boldsymbol{Y}$ are all zero, which means that these entries are linearly independent.

A spherical distribution has only nonlinear dependence, which is described by the distribution of R. The generator is derived from the distribution of R, and so it also contains information of nonlinear dependence.

We now take a look at the parameters of an elliptical copula $\mathcal{C}(\Omega, g)$. The correlation matrix informs us about the linear dependence, while the generator expresses nonlinear dependence. This is how a copula describes the entire dependence structure.

2.4 Condition 1

The generator cannot be an arbitrary function. Equations (2.1) and (2.2) show that $h(r)$ and the generator are related. Since $h(r)$ is a probability density, it must integrate to one, and so the generator g must satisfy

$$\int_0^\infty \frac{2\pi^{d/2}}{\Gamma(d/2)} r^{d-1} g(r^2) dr = 1 \iff \int_0^\infty r^{d-1} g(r^2) dr = \frac{\Gamma(d/2)}{2\pi^{d/2}}. \tag{2.4}$$

If we derive this condition in terms of the PDF of $Y = R^2$, it can be expressed as

$$\int_0^\infty \frac{\pi^{d/2}}{\Gamma(d/2)} y^{d/2-1} g(y) dy = 1 \iff \int_0^\infty y^{d/2-1} g(y) dy = \frac{\Gamma(d/2)}{\pi^{d/2}}. \tag{2.5}$$

Equations (2.4) and (2.5) are equivalent. We refer to both of them as Condition 1. Beside Condition 1, g needs to satisfy other conditions but those are relatively weak and will not be discussed in this dissertation. See Derumigny and Fermanian (2022) for other constraints on g.

2.5 Normalization of a Copula Generator

Given a function that does not satisfy Equation (2.4) or Equation (2.5), it can be transformed to satisfy Condition 1. We call this process "normalization", because it amounts to normalizing the corresponding PDF of R. Algorithm 1 describes the steps to normalize a generator.

Algorithm 1 Normalization of a Generator

Input: A function g that is not normalized.

1: Compute $\mathcal{I}_1 = \int_0^\infty t^{d/2-1} g(t) dt$, where the integrand is the kernel of the PDF of R^2;

2: Denote $s_d = \frac{\pi^{d/2}}{\Gamma(d/2)}$;

3: Calculate $\tilde{g}(t) = \frac{1}{s_d \mathcal{I}_1} g(t)$.

Output: A normalized $\tilde{g}$ satisfying Condition 1 in Equation (2.4).

2.6 Identifiability

The elliptical copula $\mathcal{C}(\Omega, g)$ is parameterized in the same way as its associated elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$. However, generators are not unique and identifiable for elliptical copulas. This problem has been discussed by Derumigny and Fermanian (2022).

Here we give a simplified explanation. For a given generator g, the marginal CDF and PDF of $\mathcal{E}(\mathbf{0}, \Omega, g)$ are given by

$$\begin{aligned} F_g(x) &= \frac{1}{2} + \frac{\pi^{(d-1)/2}}{\Gamma\left(\frac{d-1}{2}\right)} \int_0^x \int_{u^2}^{\infty} (y - u^2)^{(d-1)/2-1} g(y) dy du, \\ f_g(x) &= \frac{\pi^{(d-1)/2}}{\Gamma\left(\frac{d-1}{2}\right)} \int_{x^2}^{\infty} (y - x^2)^{(d-1)/2-1} g(y) dy. \end{aligned} \tag{2.6}$$

Thus, an elliptical distribution's generator not only contains information about the non-linear dependence, but also information about the marginal distributions. This can cause an identifiability problem when using g to parameterize an elliptical copula. In particular, assume that two elliptical distributions, $\mathcal{E}(\mathbf{0}, \Omega, g_a)$ and $\mathcal{E}(\mathbf{0}, \Omega, g_b)$, share the same dependence structure but have different marginal distributions. In such a case, these distributions have different generators, g_a and g_b, but their copulas are the same, which can be denoted by either $\mathcal{C}(\Omega, g_a)$ or $\mathcal{C}(\Omega, g_b)$. It should be mentioned that the correlation matrix Ω is unique and identifiable for elliptical copulas, because the correlation matrix contains only information about the linear dependence.

Derumigny and Fermanian (2022) propose a transformation for changing the generator of an elliptical copula without modifying the dependence structure. Specifically, let Ω be a positive definite correlation matrix and g a generator. Then, for any positive value a, $\mathcal{C}(\Omega, g) = \mathcal{C}(\Omega, g_a)$ by setting

$$g_a(t) = a^{d/2} g(at). \tag{2.7}$$

In other words, the dependence structure induced by g is invariant under the transformation (2.7), although the marginal distribution F_g and f_g have changed to F_{g_a} and f_{g_a} because of their relation to the generator described in Equation (2.6).

2.7 Standardization of the Copula Generator

It was shown in the previous section that the generator of an elliptical copula is not unique or identifiable because it contains information on both the dependence and the marginal distribution. The transformation (2.7) can modify the corresponding marginal distribution associated with a generator without changing the dependence structure. This transformation is able to make the generator unique and constrain the marginal distribution to a have a specific form.

According to Derumigny and Fermanian (2022), for an elliptical copula, there "most often" exists a unique normalized generator g satisfying

$$\frac{\pi^{\frac{d-1}{2}}}{\Gamma(\frac{d-1}{2})} \int_0^\infty s^{\frac{d-3}{2}} g(s) ds = b, \tag{2.8}$$

where b is a positive number that we call a standardization constant. In other words, each elliptical copula has only *one* generator satisfying the constraint in Equation (2.8).

The quantity b is in fact $f_g(0)$. Recall that f_g is the marginal PDF derived from g via Equation (2.6). When fixing $f_g(0) = b$, f_g is specified. We do not need to fix the values of f_g everywhere to specify the marginal distribution. For a detailed proof, see Proposition 3 and Appendix A and in Derumigny and Fermanian (2022).

Algorithm 2 Standardizing the Generator

Input: A generator $\tilde{g}$ that is already normalized by Algorithm 1, such that $\mathcal{I}_1 = \int_0^\infty t^{d/2-1} \tilde{g}(t) dt = \Gamma(d/2)/\pi^{d/2}$.

1: Compute and $\mathcal{I}_2 = \int_0^\infty t^{(d-3)/2} \tilde{g}(t) dt$;

2: Set $\beta = \left(b/(s_{d-1}\, \mathcal{I}_2)\right)^2$, where b is any positive constant, and $s_{d-1} = \pi^{(d-1)/2}/\Gamma\left((d-1)/2\right)$;

3: Calculate $\tilde{g}_s = \beta^{d/2}\, \tilde{g}(\beta\, t)$;

Output: A modified version $\tilde{g}_s$ satisfying the normalization and identification constraints.

We provide Algorithm 2 for standardizing the generator of an elliptical copula. It has been modified from the algorithm in Derumigny and Fermanian (2022).

2.8 Literature Review

Though elliptical copulas were introduced a while ago, research on the estimation of elliptical copulas is fairly limited. Related articles are mostly focused on the application of a specific type of elliptical copula, such as a Gaussian copula or a Student-t copula. In other words, these articles assume a known generator g. Genest et al. (2007) assume that the generator g is unknown but is one option from a fixed list, such as those listed in Table 1.1. Genest et al. (2007) provide goodness-of-fit tests to select the best generator from a list for a given data set but there is no rigorous way to estimate the generator of an elliptical copula.

Recently, an iterative procedure was proposed by Derumigny and Fermanian (2022). This iterative procedure estimates the generator of an elliptical copula on a pre-specified grid. The algorithm interactively "translates" the copula data into pseudo data having an elliptical distribution and then estimates the generator of the elliptical distribution using the procedure from Liebscher (2005). There are two major deficiencies in their algorithm. Firstly, it does not show very good results in some simulation studies. Secondly, Derumigny and Fermanian (2022) do not evaluate the likelihood through the density of elliptical copulas, rather, they estimate an elliptical distribution from the transformed data. This makes the assessment of their algorithm difficult. Besides, there is no proof of convergence of their algorithm, though it seems to have converged in their experiments.

Liebscher (2005) provides an estimator of the generator g of elliptical distributions, which is constructed indirectly through the estimator of $h(r)$. We have introduced Liebscher's model in Chapter 3. This method works on elliptical *distributions*. Estimation of the generator of an elliptical *copula* and an elliptical *distribution* can be quite different due to two factors. Firstly, the generator of elliptical copulas is not unique, as mentioned in Section 2.6. Secondly, the estimation of elliptical copulas involves the evaluation of the marginal quantile function and the marginal probability density function of its associated elliptical distribution, both of which have no closed form in many cases.

Chapter 3

Methodology

3.1 Modeling the Generator

To estimate the generator, we first need to construct a model for it. In Section 2.4, we have mentioned in Section 2.4 that to be a generator, a function must satisfy Condition 1 described in Equation (2.4). The model for g can be constructed through the density of R or the density of R^2 because of the relations described in equations (2.1) and (2.2). In this case, the model automatically satisfies Condition 1.

We model the generator g via $f_Y(y)$, which is the density of the squared radius, i.e., $Y = R^2$. Assume that we have a model for $f_Y(y)$, denoted by $m(y)$, then a model for g, denoted by $g_m(t)$, can be written as

$$g_m(t) = \frac{\Gamma(d/2)}{\pi^{d/2}} t^{1-d/2} m(t). \tag{3.1}$$

We refer to model (3.1) as the "naive model". The deficiency of the naive model is discussed in Liebscher (2005). The $\lim_{t\to 0} g_m(t)$ goes to ∞ when $m(y)$ is bounded away from 0 in a neighborhood of zero because $\lim_{t\to 0} t^{1-d/2} = \infty$ when the dimension d is greater than 2. This implies that the copula density (1.7) goes to infinity at $\boldsymbol{u} = (1/2, 1/2, \ldots, 1/2)$. In many cases, we do not want such a strong constraint on the model.

3.1.1 A Model Based on a Mixture of Noncentral F-Distributions

In order to make g_m well-behaved near zero, we can use some methods to cancel out the term $t^{1-d/2}$. One solution is to use noncentral F-distributions in modeling $m(y)$. The PDF

of a noncentral F-distribution is

$$f_{NCF}(x \mid \nu_1, \nu_2, \lambda) = \sum_{k=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^k}{B(\frac{\nu_2}{2}, \frac{\nu_1}{2}+k)k!} \left(\frac{\nu_1}{\nu_2}\right)^{\nu_1/\nu_2+k} \left(\frac{\nu_2}{\nu_2+\nu_1 x}\right)^{(\nu_1+\nu_2)/2+k} x^{\nu_1/2-1+k}, \tag{3.2}$$

where ν_1 and ν_2 are degrees of freedom and λ is the noncentrality parameter.

Assume that model $m(y)$ is $f_{NCF}(x \mid \nu_1, \nu_2, \lambda)$ with its first degree of freedom ν_1 equal to the copula dimension d, then the model for the generator becomes

$$g_m(t \mid \nu_2, \lambda) = \frac{\Gamma(d/2)}{\pi^{p/2}} \sum_{k=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^k}{B(\frac{\nu_2}{2}, \frac{d}{2}+k)k!} \left(\frac{d}{\nu_2}\right)^{\frac{d}{\nu_2}+k} \left(\frac{\nu_2}{\nu_2+dt}\right)^{\frac{d+\nu_2}{2}+k} t^k, \tag{3.3}$$

which is well-behaved near zero because the term $t^{1-d/2}$ is cancelled out by a term from the density of the noncentral F-distribution.

In order to improve flexibility, we use a mixture of noncentral F-distributions as the model for f_Y, and denote it by m_1

$$m_1(y) = \sum_{j=1}^{K} w_j f_{NCF}(y \mid \nu_{1j} = d, \nu_{2j}, \lambda_j), \tag{3.4}$$

where the w_js are mixing proportions, and (ν_{2j}, λ_j) are parameters for the jth component. Every component has its first degree of freedom ν_{1j} equal to the dimension d. The corresponding model for the generator is

$$g_{m_1}(t) = \frac{\Gamma(d/2)}{\pi^{d/2}} t^{1-d/2} m_1(t) = \sum_{j=1}^{K} w_j \frac{\Gamma(d/2)}{\pi^{d/2}} t^{1-d/2} f_{NCF}(y \mid \nu_{1j} = d, \nu_{2j}, \lambda_j). \tag{3.5}$$

The g_{m_1} in (3.5) can be viewed as a mixture of generators whose components are the g_m in Equation (3.3).

Proposition 4 *A mixture of generators is also a generator.*

Proof: To be a generator, a function $g(t)$ has to satisfy Condition 1 in Equation (2.4)

$$\int_0^{\infty} \frac{\pi^{d/2}}{\Gamma(d/2)} z^{d/2-1} g(z) dz = 1.$$

Now assume g is a mixture of generators,

$$g(t) = \sum_{j=1}^{K} w_j g_j(t),$$

where the g_j all satisfy

$$\int_0^\infty \frac{\pi^{d/2}}{\Gamma(d/2)} z^{d/2-1} g_j(z) = 1.$$

Then we have

$$\begin{aligned}
&\int_0^\infty \frac{\pi^{d/2}}{\Gamma(d/2)} z^{d/2-1} g(z) dz \\
&= \int_0^\infty \left(\frac{\pi^{d/2}}{\Gamma(d/2)} z^{d/2-1} \sum_{j=1}^{K} w_j g_j(z) \right) dz \\
&= \int_0^\infty \left(\sum_{j=1}^{K} w_j \frac{\pi^{d/2}}{\Gamma(d/2)} z^{d/2-1} g_j(z) \right) dz \\
&= \sum_{j=1}^{K} w_j \int_0^\infty \frac{\pi^{d/2}}{\Gamma(d/2)} z^{d/2-1} g_j(z) dz = \sum_{j=1}^{K} w_j = 1.
\end{aligned}$$

Thus, g also satisfies Condition 1 and is a valid generator.

3.1.2 A Model Based on a Mixture of Truncated Normal Distributions

Another way to fix the deficiency of the naive model in Equation (3.1) is to build the generator model through the probability density of $Y^{d/2}$. It can also dissolve the term $t^{1-d/2}$ and make the model behave well around zero.

If we have a model for the PDF of $Y^{d/2}$, denoted by m_2, then the model for the generator g can be

$$g_{m_2}(t) = \frac{\Gamma(d/2)d}{2\pi^{d/2}} m_2(t^{\frac{d}{2}}). \tag{3.6}$$

However, Liebscher (2005) pointed that this model became wiggle for large value of t, since the power transformation magnified small differences. Liebscher (2005) provided a way to create a model for the generator g, which can overcome disadvantages mentioned above.

Now assume we have a transformation function Ψ and $\tilde{Y} = \Psi(Y)$. The model for g is built through the density of $\tilde{Y}$. Liebscher (2005) gave an example of the transformation function Ψ, which is

$$\Psi(Y) = -a + (a^{d/2} + Y^{d/2})^{2/d}, \tag{3.7}$$

where a is a positive constant, such as 1.

We model the probability density function of $\tilde{Y}$ via a finite mixture

$$m_3(\tilde{y}) = \sum_{j=1}^{K} w_j f_j(\tilde{y} \mid \boldsymbol{\theta}_j), \tag{3.8}$$

where the w_js are mixing proportions and $f_j,\ j = 1, \ldots, K$ are mixture components with the parameter vector $\boldsymbol{\theta}_j$. The density of $Y = R^2$ is

$$f_Y(y) = f_{\tilde{y}}(\tilde{y}) \times \left|\frac{d\tilde{y}}{dy}\right| = (a^{d/2} + y^{d/2})^{2/d-1} y^{d/2-1} \sum_{j=1}^{K} w_j f_j\big(-a + (a^{d/2} + y^{d/2})^{2/d}\big).$$

The corresponding generator is

$$\begin{aligned} g_{m_3}(t) &= \frac{\Gamma(d/2)}{\pi^{d/2}} t^{1-d/2} f_Y(t) \\ &= \frac{\Gamma(d/2)}{\pi^{d/2}} (a^{d/2} + t^{d/2})^{2/d-1} \sum_{j=1}^{K} w_j f_j\big(-a + (a^{d/2} + t^{d/2})^{2/d}\big) \\ &= \sum_{j=1}^{K} w_j \frac{\Gamma(d/2)}{\pi^{d/2}} (a^{d/2} + t^{d/2})^{2/d-1} f_j\big(-a + (a^{d/2} + t^{d/2})^{2/d}\big) \\ &= \sum_{j=1}^{K} w_j g_j(t), \end{aligned} \tag{3.9}$$

where

$$g_j(t) = \frac{\Gamma(d/2)}{\pi^{d/2}} (a^{d/2} + t^{d/2})^{2/d-1} f_j\big(-a + (a^{d/2} + t^{d/2})^{2/d}\big) \tag{3.10}$$

is a generator itself. It is easy to prove that g_j in Equation (3.10) satisfies Condition 1.

The mixture in Equation 3.8 can be a mixture of any continuous distribution with its support on $(0, \infty)$. What we use is a mixture of truncated normal distributions. Assume we have

$$m_3(\tilde{y}) = \sum_{j=1}^{K} w_j \phi_{j(0,\infty)}(\tilde{y} \mid \mu_j, \sigma_j = 1), \tag{3.11}$$

where the $\phi_{j_{(0,\infty)}}$ is a normal distribution with its mean equal to μ_j and its standard deviation equal to one, truncated to $(0, \infty)$. From Equation (3.11), the model for the generator is

$$g_{m_3}(t) = \sum_{j=1}^{K} w_j g_j(t), \tag{3.12}$$

where

$$g_j(t) = \frac{\Gamma(d/2)}{\pi^{d/2}}(a^{d/2} + t^{d/2})^{2/d-1}\phi_{j_{(0,\infty)}}\big(- a + (a^{d/2} + t^{d/2})^{2/d} \mid \mu_j, \sigma_j = 1\big). \tag{3.13}$$

3.2 Priors

3.2.1 Prior for Mixing Proportions

For the mixing proportions, we use the Dirichlet distribution as prior,

$$(w_1, \ldots, w_K) \sim \text{Dir}(\alpha_1, \ldots, \alpha_K),$$

where $\alpha_1 = \ldots = \alpha_K = 1/K$.

In this dissertation, Metropolis algorithm is used to sample the mixing proportions from the posterior distribution, so it would be convenient to transform $w_1, \ldots, w_K$ to unconstrained variables. A common way to do this is using multi-logits,

$$w_j = \frac{\exp(v_j)}{1 + \sum_{\ell=1}^{K-1} \exp(v_\ell)},\ j = 1, \ldots, K-1$$
$$w_K = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(v_\ell)} = 1 - \sum_{\ell=1}^{K-1} w_\ell,$$

where $v_1, \ldots, v_{K-1} \in (-\infty, \infty)$. The Jacobian for this transfomation is

$$J = \begin{bmatrix} \frac{\partial w_1}{\partial v_1} & \frac{\partial w_1}{\partial v_2} & \cdots & \frac{\partial w_1}{\partial v_{K-1}} \\ \frac{\partial w_2}{\partial v_1} & \frac{\partial w_2}{\partial v_2} & \cdots & \frac{\partial w_2}{\partial v_{K-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial w_{K-1}}{\partial v_1} & \frac{\partial w_{K-1}}{\partial v_2} & \cdots & \frac{\partial w_{K-1}}{\partial v_{K-1}} \end{bmatrix},$$

where diagonal elements are

$$\begin{aligned}\frac{\partial w_j}{\partial v_j} &= \frac{\exp(v_j)\left(1+\sum_{\ell=1}^{K-1}\exp(v_\ell)\right)-\exp(2v_j)}{\left(1+\sum_{\ell=1}^{K-1}\exp(v_\ell)\right)^2} \\ &= \frac{\exp(v_j)\left(1+\sum_{\ell=1}^{K-1}\exp(v_\ell)-\exp(v_j)\right)}{\left(1+\sum_{\ell=1}^{K-1}\exp(v_\ell)\right)^2} \\ &= \frac{\exp(v_j)\left(1+\sum_{\ell\neq j}^{K-1}\exp(v_\ell)\right)}{\left(1+\sum_{\ell=1}^{K-1}\exp(v_\ell)\right)^2}\end{aligned}$$

and off diagonal elements are

$$\frac{\partial w_j}{\partial v_k} = \frac{-\exp(v_j+v_k)}{\left(1+\sum_{\ell=1}^{K-1}\exp(v_\ell)\right)^2}, \quad j \neq k.$$

3.2.2 Priors for Mixture Components

For the mixture of noncentral F-distributions, we use uniform priors for both degrees of freedom and noncentrality parameters,

$$\nu_{2j} \sim \mathcal{U}(0, \delta_0)$$

and

$$\lambda_j \sim \mathcal{U}(0, \delta_0),$$

where the upper bound δ_0 is a positive number, such as 50.

For the mixture of truncated normal distributions, we use uniform priors on μ_js,

$$\mu_j \sim \mathcal{U}(L, U),$$

where L and U are lower and upper bounds, respectively. In our experiments in Chapter 4, we set $L = -5$ and $U = 5$.

3.3 Likelihood

Given N independent observations $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_N$ from an elliptical copula $\mathcal{C}(\Omega, g)$ in Equation (1.7), where the correlation matrix Ω is known or its estimate is obtained from a separate procedure, the likelihood function is given by

$$L(g \mid \boldsymbol{U}_1, \ldots, \boldsymbol{U}_N) = \prod_{i=1}^{N} \left(\frac{\det(\Omega)^{-1/2} g\big(Q_g(\boldsymbol{U}_i)^T \Omega^{-1} Q_g(\boldsymbol{U}_i)\big)}{\prod_{j=1}^{d} f_g\big(Q_g(U_{ij})\big)} \right), \tag{3.14}$$

where

$$F_g(x) = \frac{1}{2} + \frac{\pi^{(p-1)/2}}{\Gamma\left(\frac{p-1}{2}\right)} \int_0^x \int_{u^2}^{\infty} (y - u^2)^{(p-1)/2-1} g(y) dy du, \tag{3.15}$$

$$Q_g(u) = F_g^{-1}(u), \tag{3.16}$$

and

$$f_g(x) = \frac{\pi^{(p-1)/2}}{\Gamma\left(\frac{p-1}{2}\right)} \int_{x^2}^{\infty} (y - x^2)^{(p-1)/2-1} g(y) dy. \tag{3.17}$$

Recall that F_g and f_g are the marginal CDF and marginal PDF of the associated elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$, and Q_g is the inverse function of F_g.

The likelihood function (3.14) can hardly be further simplified no matter what model is used for the generator g. Though F_g, Q_g and f_g can be derived from a given g, these functions are expressed as complex integrals. Due to the same reason, it is challenging to directly evaluation the likelihood. If we use numeric integration to evaluate F_g and f_g and root finding method to obtain the inverse function $Q_g = F_g^{-1}$, it can be very time consuming and often not accurate. Instead, we propose to use B-spline smoothing to approximate F_g and f_g.

3.4 Evaluation of Marginal Generator

As we mentioned, the f_g is the marginal PDF of the associated elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$. The marginal distribution of an elliptical distribution is also elliptical, see Theorem 6 of Gómez et al. (2003). Thus, the marginal elliptical distribution has its own

generator, denoted by g_1. The marginal PDF f_g can be computed from g_1 via

$$f_g(x) = g_1(x^2). \tag{3.18}$$

To evaluate f_g, we can evaluate g_1 first. The marginal generator is defined as

$$g_1(t) = \frac{\pi^{(d-1)/2}}{\Gamma\big((d-1)/2\big)} \int_0^\infty g(t+s) s^{(d-3)/2} ds. \tag{3.19}$$

By using the substitution $y = t + s$, g_1 can be written as

$$g_1(t) = \frac{\pi^{(d-1)/2}}{\Gamma\big((d-1)/2\big)} \int_t^\infty g(y)(y-t)^{(d-3)/2} dy. \tag{3.20}$$

This integral can approximated via Riemann sum. In particular, denote the integrand in Equation (3.20) as $f(y) = g(y)(y-t)^{(d-3)/2}$. Assume we need to evaluate g_1 at the knots $0 = t_1, \ldots, t_n = T_g$, where T_g is a value to which we truncate the generator g, such that at T_g, $g(T_g)$ is about zero. Knots $t_1, \ldots t_n$ are evenly spaced by Δt. We have

$$\begin{aligned} g_1(t_j) &\propto \int_{t_j}^\infty g(y)(y-t_j)^{(d-3)/2} dy \\ &\approx \int_{t_j}^{t_n} g(y)(y-t_j)^{(d-3)/2} dy \\ &= \int_{t_j}^{t_n} f(y) dy. \end{aligned}$$

The value of $\int_{t_j}^{t_n} f(y)dy$ can be approximated with Right Riemann sum:

$$\begin{aligned} \int_{t_j}^{t_n} f(y)dy &\propto \Delta t \Big(f(t_j + \Delta t) + f(t_j + 2\Delta t) + \ldots + f(t_n) \Big) \\ &= \Delta t \Big(f(t_{j+1}) + f(t_{j+2}) + \ldots + f(t_n) \Big) \\ &= \Delta t \sum_{i=j+1}^{n} f(t_i). \end{aligned}$$

We will need to evaluate the integrand $f(y)$ at t_{j+1} to t_n, in total $n - j$ knots. In this way, we can obtain $g_1(t_1)$ to $g_1(t_{n-1})$. To obtain $g_1(t_{n-1})$, we only need to evaluate $f(t_n)$.

The integrand for calculating $g_1(t_j)$ is

$$f(t_i) = g(t_i)(t_i - t_j)^{\frac{d-3}{2}}, i = j+1, \ldots, n.$$

Note that the integrand's second term depends on t_j. Thus, the integrand will change with t_j at which we want to evaluate g_1. Since $t_1 = 0$ and we partition $[t_1, t_n]$ evenly with space Δt, we have

$$\begin{aligned} t_2 &= \Delta t \\ t_3 &= 2\Delta t \\ &\dots \\ t_{n-j+1} &= (n-j)\Delta t. \end{aligned}$$

Thus,

$$\begin{aligned} f(t_{j+1}) &= g(t_{j+1})(\Delta t)^{\frac{d-3}{2}} = g(t_{j+1})(t_2)^{\frac{d-3}{2}} \\ f(t_{j+2}) &= g(t_{j+2})(2\Delta t)^{\frac{d-3}{2}} = g(t_{j+2})(t_3)^{\frac{d-3}{2}} \\ &\dots \quad \dots \quad \dots \\ f(t_{n-1}) &= g(t_{n-1})t_{n-j}^{\frac{d-3}{2}} \\ f(t_n) &= g(t_n)\big((n-j)\Delta t\big)^{\frac{d-3}{2}} = g(t_n)(t_{n-j+1})^{\frac{d-3}{2}}. \end{aligned} \tag{3.21}$$

From the Equations in (3.21) above, we can see that we need the value of $g(t_{j+1})$ to $g(t_n)$ and knots t_2 to t_{n-j+1}.

Table 3.1: Approximation of g_1 via Riemman sum.

Evaluation of g_1	Integrand	Evaluation of g	Knots	Number of knots
$g_1(t_1)$	$f(t_2)$ to $f(t_n)$	$g(t_2)$ to $g(t_n)$	$t_2^{(d-3)/2}$ to $t_n^{(d-3)/2}$	n-1 points
$g_1(t_2)$	$f(t_3)$ to $f(t_n)$	$g(t_3)$ to $g(t_n)$	$t_2^{(d-3)/2}$ to $t_{n-1}^{(d-3)/2}$	n-2 points
...	...	...	...	...
$g_1(t_j)$	$f(t_{j+1})$ to $f(t_n)$	$g(t_{j+1})$ to $g(t_n)$	$t_2^{(d-3)/2}$ to $t_{n-j+1}^{(d-3)/2}$	n-j points
...	...	...	...	...
$g_1(t_{n-1})$	$f(t_n)$	$g(t_n)$	$t_2^{(d-3)/2}$	1 points

The process of evaluating g_1 at knots t_1 to t_{n-1} is summarized in Table 3.1. The first column needs to be evaluated. The second column is the integrand. The third and forth

columns are values required to compute the integrand and the last column is the total number of knots at which the integrand needs to be evaluated.

With the approximation of $g_1(t_1), \ldots, g_1(t_{n-1})$, we can use (3.18) to compute the approximation of

$$f_g(-\sqrt{t_{n-1}}), \ldots, f_g(-\sqrt{t_1}), f_g(\sqrt{t_1}), \ldots, f_g(\sqrt{t_{n-1}}). \tag{3.22}$$

The sequence in (3.22) can be used to obtain a spline function approximation of f_g.

3.5 B-spline

Assume that we have a sequence of samples $(x_i, y_i),\ i = 1, \ldots, n$ from an unknown function $y = f(x)$, and we want to obtain a smoothing estimator $\hat{f}(x)$ for the function, it is popular

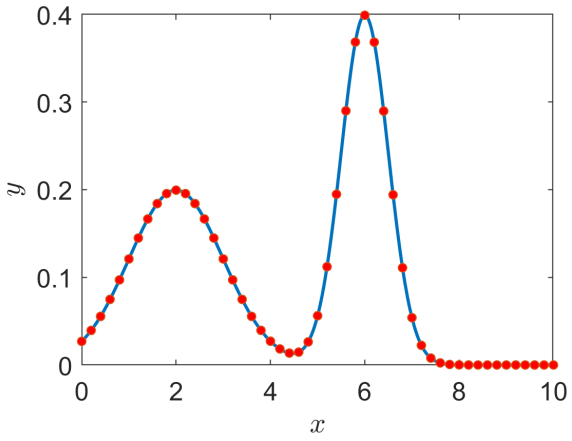


Figure 3.1: A sequence of samples (red) from an unknown function (blue).

to model the unknown function f as a linear combination of K basis functions $B_1, \ldots, B_K$, so that

$$\hat{y} = \sum_{j=1}^{K} \alpha_j B_j(x). \tag{3.23}$$

In such case, the model in Equation (3.23) can be written as

$$\boldsymbol{Y} = B\boldsymbol{\alpha} + \boldsymbol{\epsilon}, \tag{3.24}$$

where $\boldsymbol{Y} = (y_1, \ldots, y_n)$, $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$ are the vector of error terms, the $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$ is the vector of coefficients, and B is a n by K design matrix

$$B = \begin{bmatrix} B_1(x_1) & B_2(x_1) \ldots & B_K(x_1) \\ B_1(x_2) & B_2(x_2) \ldots & B_K(x_2) \\ \vdots & \vdots & \vdots \\ B_1(x_n) & B_2(x_n) \ldots & B_K(x_n) \end{bmatrix}$$

with its ith row equal to basis functions $B_1, \ldots, B_K$ evaluated at x_i.

There are a few options for the basis functions. Here we consider the commonly used B-splines. The name"B-spline" is short for basis spline. A mth order spline is a piecewise polynomial function of degree $m-1$, that is continuous and has continuous derivatives of order $1, \ldots, m-1$, at its knot points. See de Boor (1978) and Eilers and Marx (2021) for more details. Figure 3.2 visualized the B-splines with order= 4 and degree= 3, which are called cubic B-splines.

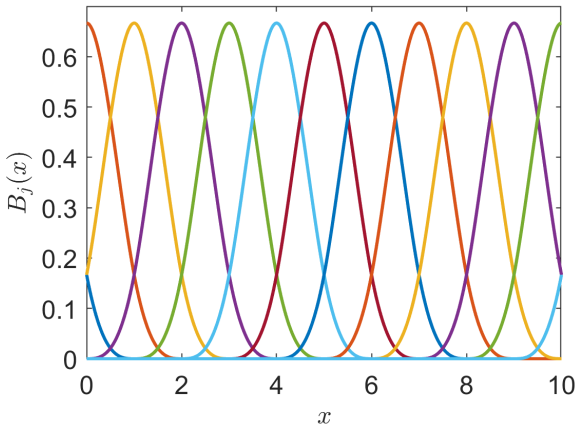


Figure 3.2: Example of Cubic B-slines.

B-splines can be constructed by the Cox-de Boor recursion formula, which was introduced by de Boor (1978). Given a knot sequence $t_0, t_1, \ldots, t_n$, the B-splines of order 1 are defined by

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}.$$

The basis functions for order 1 can be viewed indicator functions. If x is between knots t_i and t_{i+1}, the corresponding basis function, which is indexed by i, will be equal to 1, while other basis functions are equal to zero. The higher-order B-splines are defined by recursion

$$B_{i,k+1}(x) = w_{i,k}(x)B_{i,k}(x) + \Big(1 - w_{i+1,k}(x)\Big)B_{i+1,k}(x),$$

where

$$w_{i,k}(x) = \begin{cases} \frac{x-t_i}{t_{i+k}-t_i} & \text{if } t_{i+k} \neq t_i \\ 0 & \text{otherwise} \end{cases}.$$

Thus, in order to compute basis functions of higher orders, we have to compute those for *all* lower orders first.

3.6 B-spline Smoothing with Constraints

In order to estimate the coefficient vector $\boldsymbol{\alpha}$, we minimize the objective function

$$\begin{aligned} S &= (\boldsymbol{Y} - B\boldsymbol{\alpha})^T(\boldsymbol{Y} - B\boldsymbol{\alpha}) \\ &= \boldsymbol{\alpha}^T B^T B\boldsymbol{\alpha} - 2\boldsymbol{\alpha}^T B^T \boldsymbol{Y} + \boldsymbol{Y}^T\boldsymbol{Y}. \end{aligned} \tag{3.25}$$

To minimize S, we solve linear system

$$\frac{\partial S}{\partial \boldsymbol{\alpha}} = 2\boldsymbol{\alpha}^T B^T B - 2B^T\boldsymbol{Y} \Longleftrightarrow B^T B\boldsymbol{\alpha} = B^T\boldsymbol{Y}.$$

Beware that the B is not a square matrix and does not have inverse. The solution is

$$\boldsymbol{\alpha} = (B^T B)^{-1}B^T\boldsymbol{Y}.$$

By so far, the fitting is not smooth because smoothness is not regulated yet. For B-spline smoothing, the term of smoothing penalty needs to be added onto the objective funtion (3.25),

$$S = (\boldsymbol{Y} - B\boldsymbol{\alpha})^T(\boldsymbol{Y} - B\boldsymbol{\alpha}) + \lambda(D\boldsymbol{\alpha})^T(D\boldsymbol{\alpha}) \tag{3.26}$$

where the λ is the fixed smoothing parameter, such as 1000, and D is the matrix that forms differences of $\boldsymbol{\alpha}$, usually second differences. Then we solve the new linear system

$$\frac{\partial S}{\partial \boldsymbol{\alpha}} = 2\boldsymbol{\alpha}^T B^T B - 2B^T\boldsymbol{Y} + 2\boldsymbol{\alpha}^T\lambda D^T D = 0$$

which has solution

$$\boldsymbol{\alpha} = (B^T B + \lambda D^T D)^{-1} B^T \boldsymbol{Y}.$$

If we want to enforce positiveness of the fitting, another penalty term $\kappa \boldsymbol{\alpha}^T V \boldsymbol{\alpha}$ has to be added onto the objective function in (3.26), where the V is diagonal matrix with

$$v_j = I(\alpha_j \leq 0).$$

Given V, $\boldsymbol{\alpha}$ is computed as

$$\boldsymbol{\alpha} = (B^T B + \lambda D^T D + \kappa V)^{-1} B^T \boldsymbol{Y}.$$

Beware that the value of V is not given a priori, but updated with $\boldsymbol{\alpha}$ iteratively until convergence. For details about B-spline smoothing with shape constraints, see Chapter 8 of Eilers and Marx (2021).

3.7 Approximation of f_g and Q_g

We use B-spline smoothing on the sequence of samples in (3.22) to obtain a B-spline function as an approximation for f_g, which can simplify the evaluation of copula likelihood in (3.14).

To approximate the marginal quantile function Q_g, first we need the marginal cumulative distribution function F_g, which is the integral of f_g. Since f_g is already approximated with a B-spline function, it is not difficult to obtain its integral. After that, we have the value of F_g at a grid

$$F_g(-\sqrt{t_{n-1}}), \ldots, F_g(-\sqrt{t_1}), F_g(\sqrt{t_1}), \ldots, F_g(\sqrt{t_{n-1}}). \tag{3.27}$$

Now we flip x-axis and y-axis of the sample (3.27) and apply interpolation to obtain the functional approximation of Q_g. For simplicity, we used linear interpolation here, the same as the one used in Derumigny and Fermanian (2022). Now the evaluation of copula likelihood (3.14) becomes practical in speed as well as accuracy.

3.8 Sampling Scheme

Since the likelihood function (3.14) is complex, the joint posterior function as well as conditional posterior functions will are complicated. Thus, we use Metropolis algorithm to sample all parameters, including mixing proportions and parameters in mixture components.

Recall that the mixing proportions $(w_1, \ldots, w_K)$ are transformed into unconstrained $(v_1, \ldots, v_{K-1})$, so we operate on the unconstrained variables instead. For the transformed mixing proportions, the proposal distributions are all uniform. The new value $v_j^{(p)}$ is proposed by $v_j^{(p)} = \mathcal{U}(v_j^{(c)} - \delta_v, v_j^{(c)} + \delta_v)$, where δ_v is a fixed tuning parameter and $v_j^{(c)}$ is the current value of v_j.

We sample $\boldsymbol{v} = (v_1, \ldots, v_{K-1})$ jointly. First, we propose $\boldsymbol{v}^{(p)} = (v_1^{(p)}, \ldots, v_{K-1}^{(p)})$. The proposed value $\boldsymbol{v}^{(p)}$ is accepted with probability $\min\left\{1, \frac{p_{\boldsymbol{v}}(\boldsymbol{v}^{(p)}|\cdots)}{p_{\boldsymbol{v}}(\boldsymbol{v}^{(c)}|\cdots)}\right\}$, where $p_{\boldsymbol{v}}$ is the posterior density of $\boldsymbol{v}^{(p)} = (v_1^{(p)}, \ldots, v_{K-1}^{(p)})$ conditioned on the current values of other parameters. Before starting the MCMC loop, we need to tune the value of δ_v by running 500 iterations and see the acceptance rate. If the acceptance rate is between 0.3 and 0.6, then the value of δ_v is assumed to be good.

For other parameters, such as the μ_js in the truncated normal mixture, the Metropolis algorithm is slightly different. Because those parameters have upper and lower bounds, if the proposed values are not inside of the range set by the prior, the algorithm keeps proposing until a proper value is obtained. Other steps of Metropolis algorithm are the same with those of drawing mixing proportions. Besides, proposal distributions are also uniform.

Chapter 4

Simulation Study

4.1 Data Generation

To sample from an elliptical copula, we can sample from its associated elliptical distribution first, then apply the copula transform. The steps of sampling from an elliptical copula $\mathcal{C}(\Omega, g)$ is described in Algorithm 3.

Algorithm 3 Generating data from elliptical copula $\mathcal{C}(\Omega, g)$

Input: The correlation matrix Ω and the generator g.

1: Compute $h(r)$ from g using Equation (2.1);

2: Sample R from the distribution $h(r)$;

3: Sample $\boldsymbol{U}_s$ uniformly from the unit sphere in $\mathbb{R}^d$;

4: Compute the Cholesky factor A such that $AA^T = \Omega$;

5: Compute $\boldsymbol{X} = RA\boldsymbol{U}_s$, which is a sample from elliptical distribution $\mathcal{E}(\mathbf{0}, \Omega, g)$;

6: Compute the marginal CDF F_g from g via Equation (2.6);

7: Deliver $\boldsymbol{U} = F_g(\boldsymbol{X})$.

Output: A sample $\boldsymbol{U}$ from $\mathcal{C}(\Omega, g)$.

4.2 Bivariate Cases

For the simulation study on 2-dimensional cases, we used five elliptical copulas from Derumigny and Fermanian (2022). They have generators (before normalization and standardization):

1. $g(t) = 1/(1+t^2)$;

2. $g(t) = \exp(-t)$;

3. $g(t) = \exp(-t)+\text{bump}(t)$, where $\text{bump}(x) = \mathbf{1}\{x \in [1, 1+\pi\}(x-1)(1+\pi-x)\sin(x-1)$ is a smooth function supported on $[1, 1+\pi]$;

4. $g(t) = t/(1+t^3)$;

5. $g(t) = t^2\exp(-t^2)$.

Their correlation matrix is

$$\Omega = \begin{bmatrix} 1 & 0.2 \\ 0.2 & 1 \end{bmatrix},$$

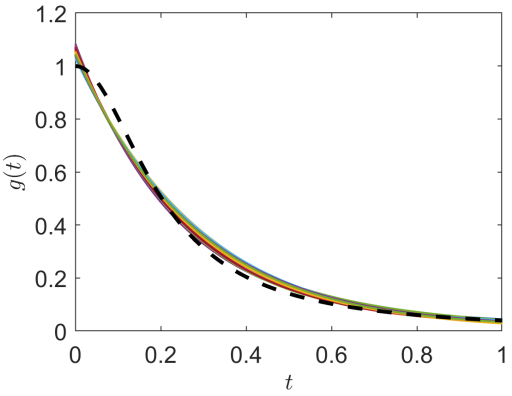
which is assumed known.

For each copula, we generated $N = 1000$ independent samples using steps in Algorithm 3. The MCMC had 10000 iterations with the first 5000 as burn-in stage. Each iteration returned an estimate of the generator g. In total, we got 5000 estimates from the MCMC loop. Then we computed the posterior mean of these estimates and denoted it as $\hat{g}$.

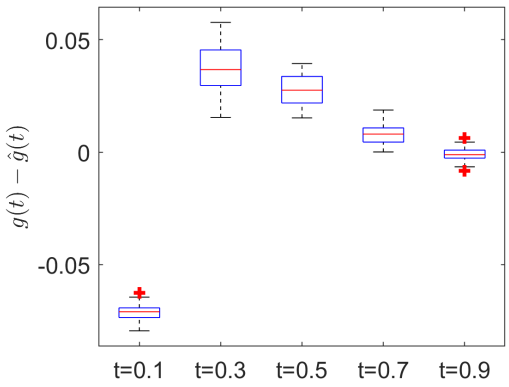
Each simulation study was repeated for 40 times on independent data sets generated from the same copula. Thus, for each copula, we have 40 posterior means of estimates for its generator. We plotted the 40 $\hat{g}$s together with the true generator $g(t)$. We call this plot as "spaghetti plot".

We also evaluated these 40 $\hat{g}$s and the true generator g at a grid $(t_1, t_2, t_3, t_4, t_5)$ and calculated the distance between $\hat{g}(t_\ell)$ and $g(t_\ell)$, $\ell = 1, \ldots, 5$. Box plot of the distances was created at each value of t_ℓ.

In Simulation 1 - 3, we used the model based on a mixture of truncated normal distributions (3.12), and in Simulation 4 - 5, we used the model based on a mixture of noncentral F distributions (3.5).

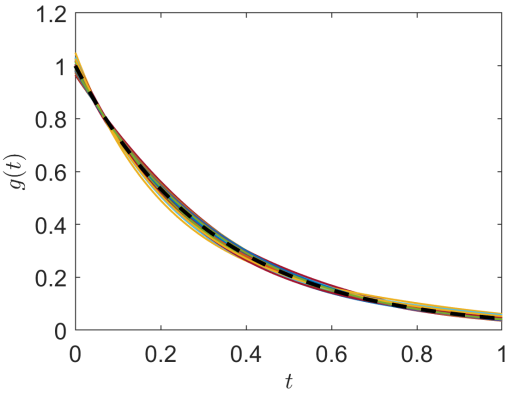


(a) The fitted and true $g(t)$

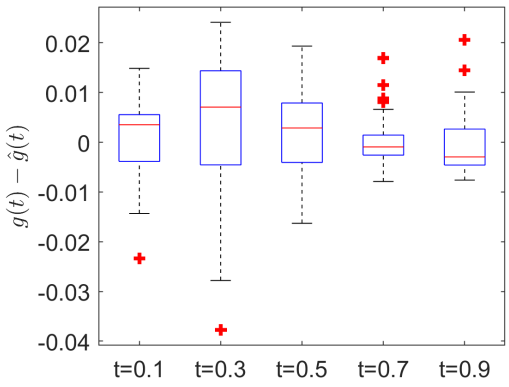


(b) Distance between the true and fitted $g(t)$

Figure 4.1: Simulation 1: $g(t) = 1/(1+t^2)$

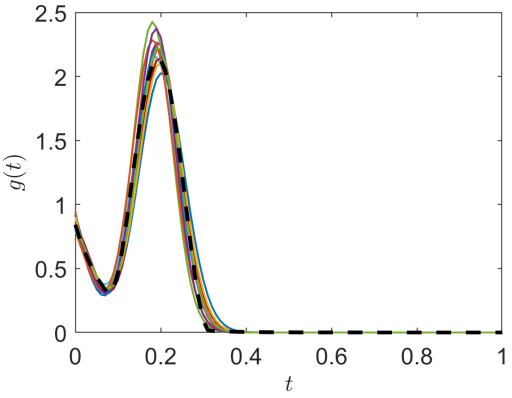


(a) The fitted and true $g(t)$

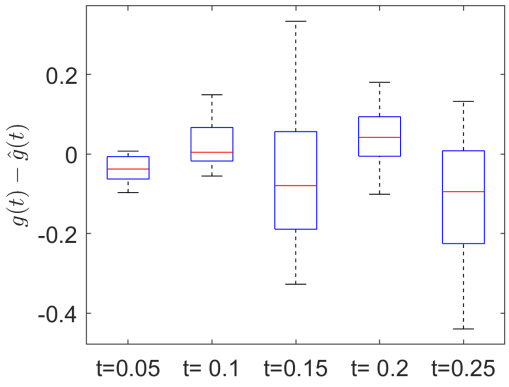


(b) Distance between the true and fitted $g(t)$

Figure 4.2: Simulation 2: $g(t) = \exp(-t)$

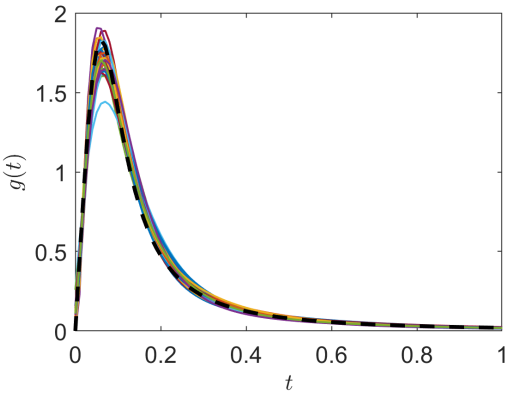


(a) The fitted and true $g(t)$

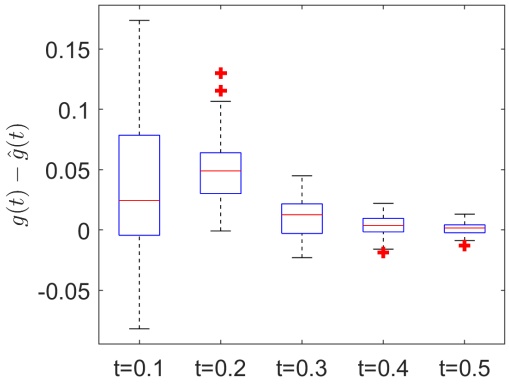


(b) Distance between the true and fitted $g(t)$

Figure 4.3: Simulation 3: $g(t) = \exp(-t) + \text{bump}$

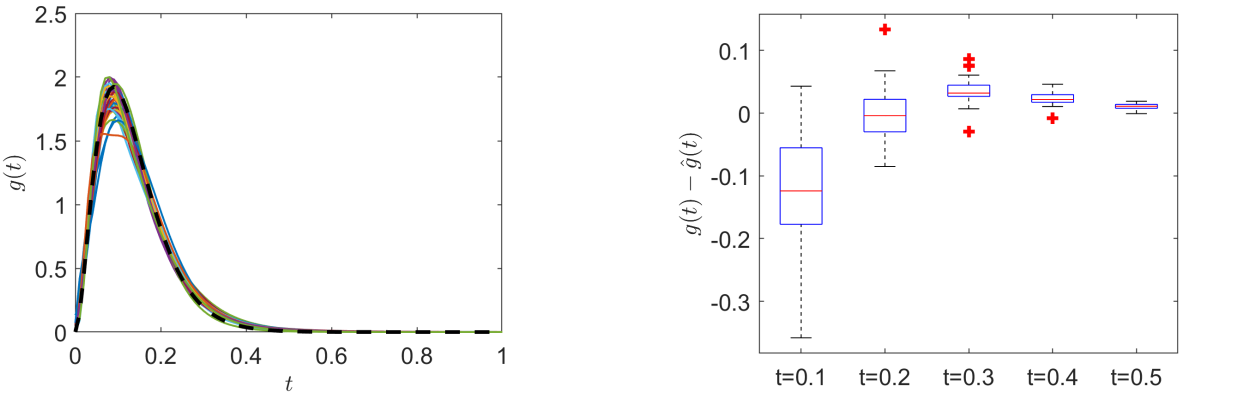


(a) The fitted and true $g(t)$



(b) Distance between the true and fitted $g(t)$

Figure 4.4: Simulation 4: $g(t) = t/(1 + t^3)$



(a) The fitted and true $g(t)$

(b) Distance between the true and fitted $g(t)$

Figure 4.5: Simulation 5: $g(t) = t^2 \exp(-t^2)$

The results are shown in Figure 4.1 to Figure 4.5. In each figure, The (a) is the spaghetti plot, which includes the true g (black dashed line) and the estimates $\hat{g}$s (solid colored lines). The (b) is box plot of the distances between the value of the true generator and $\hat{g}$ evaluated at the grid. It has to be mentioned that the plotted generators and their estimates are normalized and standardized.

From the figure (a) of each simulation, we can see the $\hat{g}$ are close to the true generator almost everywhere. From the box plots, we can see the distances are all very small, including outliers, which indicates goodness of fit. The distance between $\hat{g}$ and $g(t)$ can be larger when $g(t)$ take sharp turns. But overall, the fittings are satisfying.

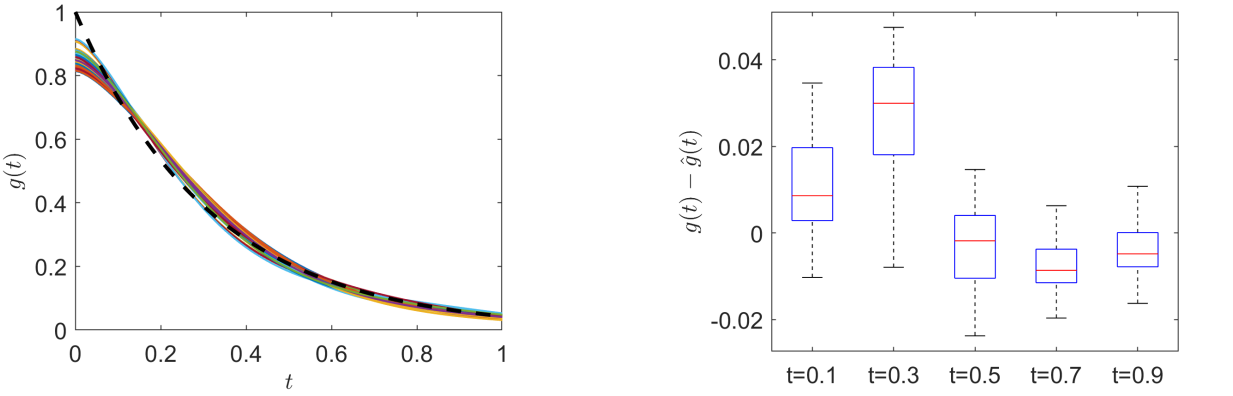
4.3 Higher Dimensions

4.3.1 Gaussian Copula

We generated $N = 1000$ observations from a 3-dimensional Gaussian copula with correlation matrix

$$\Omega = \begin{bmatrix} 1 & 0.5 & 0.6 \\ 0.5 & 1 & 0.7 \\ 0.6 & 0.7 & 1 \end{bmatrix}, \tag{4.1}$$

and also assumed the correlation matrix was known. The model used is based on a mixture of truncated normal distributions. The MCMC settings are the same with that in Section 4.2. From the spaghetti plot and box plot in Figure 4.6, we can see the fitting is also very good.



(a) The fitted and true $g(t)$

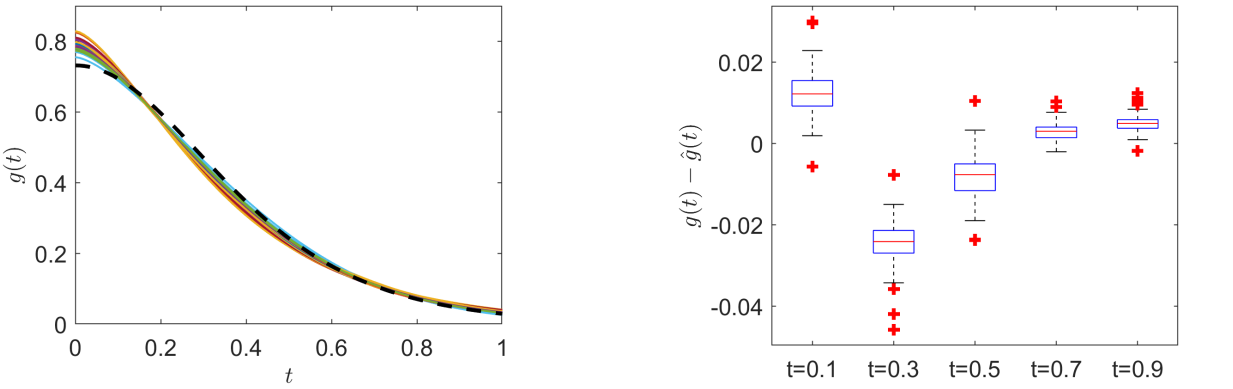
(b) Distance between the true and fitted $g(t)$

Figure 4.6: Gaussian copula: $g(t) = \exp(-t/2)$

4.3.2 Logistic Copula

We also tested our method ion a less common elliptical copula, which is a 3-dimensional logistic copula with unnormalized generator $g(t) = e^{-t}/(1 + e^{-t})$. The correlation matrix is the same one in (4.1) and is assumed known. The model is also based on a mixture of

truncated normal distributions and MCMC settings are the same. The results in Figure 4.7 look good, though the boxplot has many outliers, which are all very small numbers.



(a) The fitted and true $g(t)$

(b) Distance between the true and fitted $g(t)$

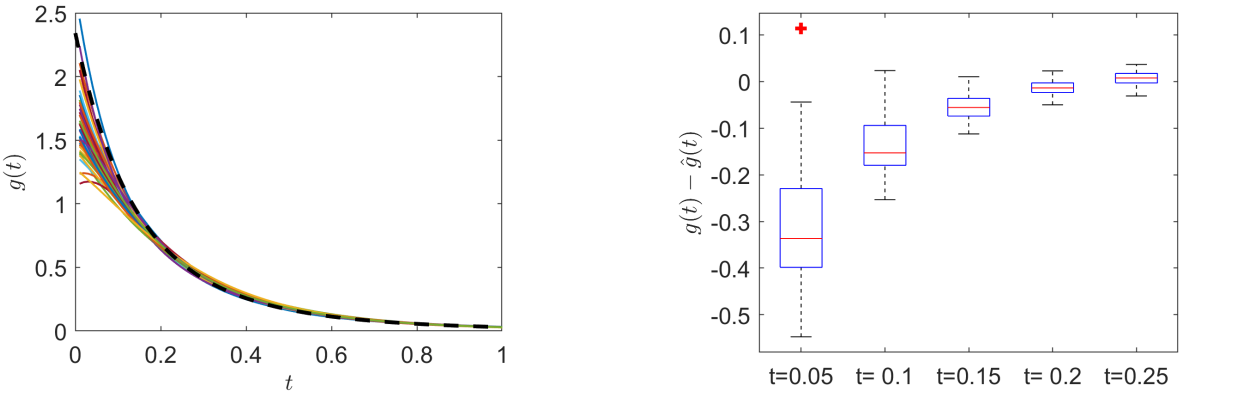
Figure 4.7: Logistic copula: $g(t) = e^{-t}/(1 + e^{-t})$

4.3.3 Student-t Copula

We generated $N = 1000$ observations from a Student-t copula with degree of freedom $\nu = 5$ and correlation matrix

$$\Omega = \begin{bmatrix} 1 & 0.2 & 0 & 0 & 0 \\ 0.2 & 1 & 0.2 & 0 & 0 \\ 0 & 0.2 & 1 & 0.2 & 0 \\ 0 & 0 & 0.2 & 1 & 0.2 \\ 0 & 0 & 0 & 0.2 & 1 \end{bmatrix}.$$

Student-t copulas have their generators of the form $g(t) \approx (1 + t/\nu)^{-(d+\nu)/2}$. We use the model based on the mixture of noncentral F distributions. Again, MCMC settings are the same with that in Section 4.2. By looking at the Figure 4.8, we can see that the distance between the true and estimated generators is a little larger when t is around zero. Besides, the fitting is still acceptable.



(a) The fitted and true $g(t)$

(b) Distance between the true and fitted $g(t)$

Figure 4.8: Student-t copula: $g(t) = (1 + t/\nu)^{-(\nu+d)/2}$

Chapter 5

Application

5.1 Meta-elliptical Distribution

A typical application of elliptical copulas is to construct meta-elliptical distributions, which are also introduced by Fang et al. (2002). Elliptical distributions require their marginal distributions to be also elliptical. However, many real world data do not satisfy this condition. For modeling multi-dimensional data sets with heterogeneous margins, a good choice would be meta-elliptical distributions. A meta-elliptical distribution consists of an elliptical copula and a collection of arbitrary chosen continuous marginals. In other words, the dependence structure is induced by an elliptical copula while its marginal distributions are not necessarily elliptical.

A d-dimensional meta-elliptical distribution can be denoted as $\mathcal{ME}(\Omega, g; F_1, \ \ldots, \ F_d)$, where Ω is a correlation matrix, g is a copula generator and $F_1, \ldots, F_d$ are its marginal CDFs. Its CDF is in the form of

$$F_{\mathcal{ME}}(x_1, \ldots, x_d) = C_{\mathcal{E}}\Big(F_1(x_1), \ldots, F_d(x_d)\Big),$$

where $C_{\mathcal{E}}$ is the CDF of the elliptical copula $\mathcal{C}_{\mathcal{E}}(\Omega, g)$. The PDF of $\mathcal{ME}$ is given by

$$f_{\mathcal{ME}}(x_1, \ldots, x_d) = c_{\mathcal{E}}\Big(F_1(x_1), \ldots, F_d(x_d)\Big) \prod_{\ell=1}^{d} f_\ell(x_\ell),$$

where the $c_{\mathcal{E}}$ is the PDF of $\mathcal{C}_{\mathcal{E}}(\Omega, g)$ and f_ℓs are marginal PDFs.

When we estimate a meta-elliptical distribution, its marginal distributions and copula can be estimated separately. Assume we observe a data set $\boldsymbol{x}_1, \ \ldots, \ \boldsymbol{x}_n$ from meta-elliptical

distribution $\mathcal{ME}(\Omega, g; F_1, \ldots, F_d)$,

$$\begin{bmatrix} \boldsymbol{x}_1' \\ \boldsymbol{x}_2' \\ \vdots \\ \boldsymbol{x}_n' \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1d} \\ x_{21} & x_{22} & \ldots & x_{2d} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1}, & x_{n2}, & \ldots, & x_{nd} \end{bmatrix}, \tag{5.1}$$

we often estimate its marginal distributions first. There are many ways to estimate $F_1, \ldots, F_d$, such as normal mixtures, Gamma mixtures, kernel densities, empirical CDFs, etc. After we obtained estimates of marginal CDFs, denoted as $\hat{F}_1, \ldots, \hat{F}_d$, we apply the copula transform in Theorem 2 on data (5.1),

$$u_{ij} = \hat{F}_j(x_{ij}), \;\; j = 1, \ldots, d, \; i = 1, \ldots, n, \tag{5.2}$$

to obtain the pseudo data

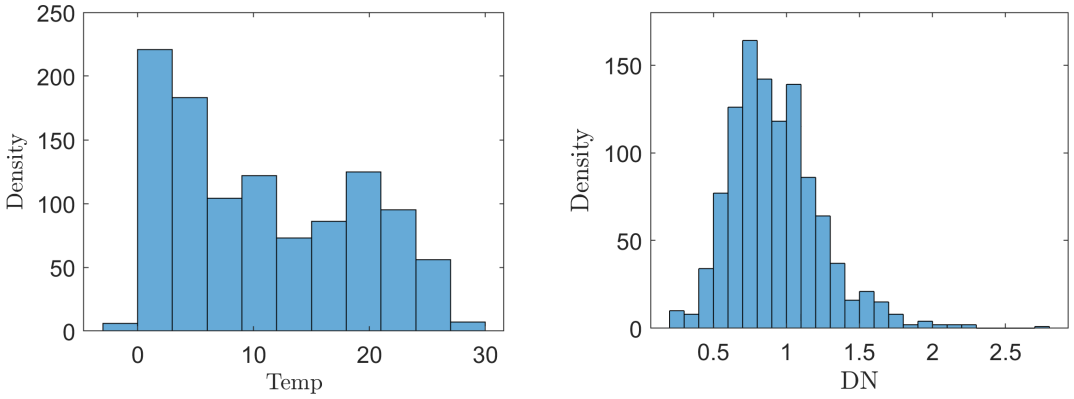
$$\begin{bmatrix} \boldsymbol{u}_1' \\ \boldsymbol{u}_2' \\ \vdots \\ \boldsymbol{u}_n' \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1d} \\ u_{21} & u_{22} & \ldots & u_{2d} \\ \ldots & \ldots & \ldots & \ldots \\ u_{n1} & u_{n2} & \ldots & u_{nd} \end{bmatrix}, \tag{5.3}$$

which can be called "copula data". The copula data (5.3) can be viewed as a data set from an elliptical copula $\mathcal{C}_{\mathcal{E}}(\Omega, g)$, which is the copula of $\mathcal{ME}(\Omega, g; F_1, \ldots, F_d)$. The estimation of copula is often performed on the copula data.

5.2 Water Quality Data

The water quality data was collected by SRBC (Susquehanna River Basin Commission), which contain parameters of the water at Susquehanna River Basin in the United States, such as temperature (Celsius), flow (cubic feet/second), dissolved oxygen (mg/L), dissolved nitrogen (mg/L), from 1985 to 2021. The data are available at (https://www.srbc.net/portals/water-quality-projects/sediment-nutrient-assessment/sites/susquehanna-river-towanda.html).

Li et al. (2019) used the subset of the data (from 1988 to 2018) to perform a copula-based analysis for time series. Thus, this is a time series data set, but we ignore the sequential dependence at this moment. We use elliptical copula to model the dependence between the water temperature (Temp) and the dissolved nitrogen (DN). In such case, the joint distribution of Temp and DN is assumed to be a meta-elliptical distribution. Nitrogen is one of the most common elements existing in many chemical compounds, such as nitrogen gas and protein. The dissolved nitrogen in water is mainly from the decomposition of organic compounds, including fish drops, died water plants and water creatures, etc. The rate of dissolving is related to the water temperature. Mostly, when the temperature goes up, so does the dissolving rate. This is because that heating up a solvent makes the molecules move faster, which increases the frequency of the solvent molecules collide with the solute.



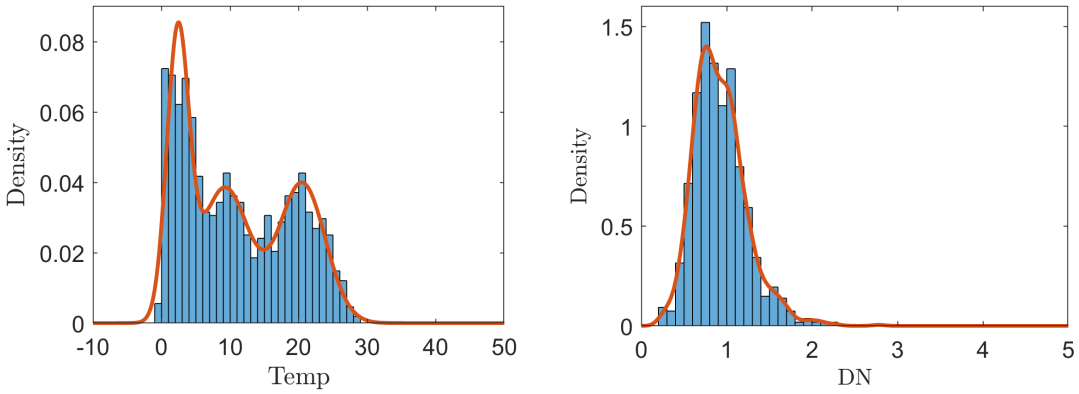
(a) Water temperature (Celsius). (b) Dissolved nitrogen (mg/L).

Figure 5.1: Histograms of water temperature and dissolved nitrogen.

The histograms in Figure 5.1 show heterogeneity of marginal distributions. The marginal distribution of the water temperature is multi-modal while the one of the dissolved nitrogen is uni-modal with slight skewness on the right side.

5.3 Estimation and Results

We estimate marginal distributions of Temp and DN first. We used 3-component normal mixture to approximate the marginal distribution of the water temperature while the dissolved nitrogen's distribution is estimated by kernel densities. The estimation was performed with functions "fitgmdist" and "fitdist" in Matlab, respectively. In Figure 5.2, we can see the curves of fitted PDFs above the histograms, which shows very good fitting results.



(a) Water temperature (Celsius). (b) Dissolved nitrogen (mg/L).

Figure 5.2: Fitted marginal PDFs of water temperature and the dissolved nitrogen.

In the next step, we use the copula transform to obtain the copula data. We denoted the estimates of marginal PDFs as $\hat{f}_T$ and $\hat{f}_N$, and marginal CDFs as $\hat{F}_T$ and $\hat{F}_N$. The copula data can be computed by

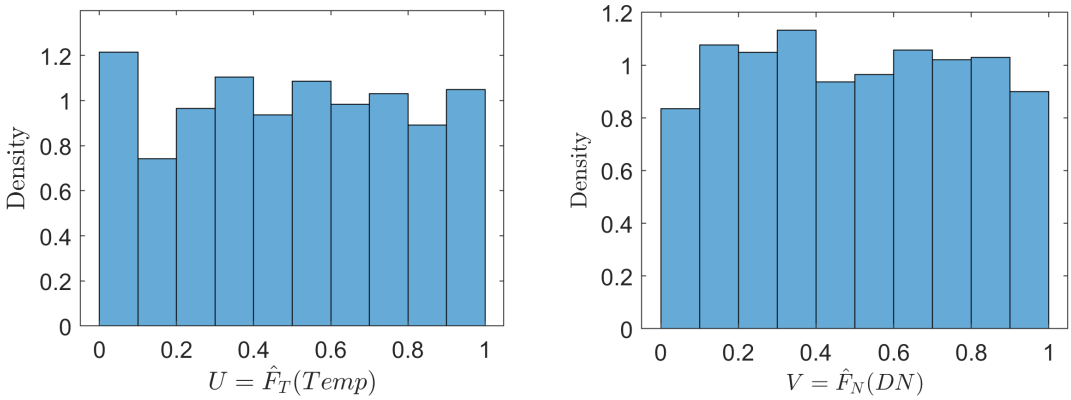
$$U = \hat{F}_T(Temp)$$

and

$$V = \hat{F}_N(DN).$$

We plotted the marginal histograms of copula data in Figure 5.3. They have standard uniform distributions as we expected. The bivariate histogram is demonstrated in Figure 5.4.

We can see that it has radial symmetry, which indicates that elliptical copulas are suitable to fit this copula data set.



(a) Water temperature.

(b) Dissolved nitrogen.

Figure 5.3: Marginal histograms of copula data.

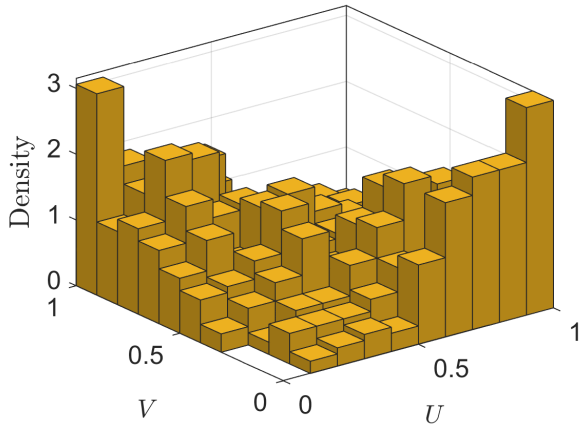


Figure 5.4: Histogram of copula data.

We use the sample correlation as the estimate of Ω, which is

$$\hat{\Omega} = \begin{bmatrix} 1 & -0.46 \\ -0.46 & 1 \end{bmatrix}.$$

Thus, in the MCMC loop, we only estimate the generator of the copula. The generator is estimated with the model based on the mixture of truncated normal distributions. We use

five components and truncated the generator to 15, which assumes the generator vanish at $t = 15$. The prior setting are the described in Section 3.2, the same as that for the simulation studies. The MCMC loop contains 10000 iterations with the first 5000 as burnin. The results are demonstrated in Figure 5.5 and Figure 5.6. The estimated copula PDF and generator are the posterior means after the burnin stage. We can see that the results look very good. By looking at the plot of the estimated generator, the elliptical copula of the water temperature and the dissolved nitrogen is a Gaussian copula.

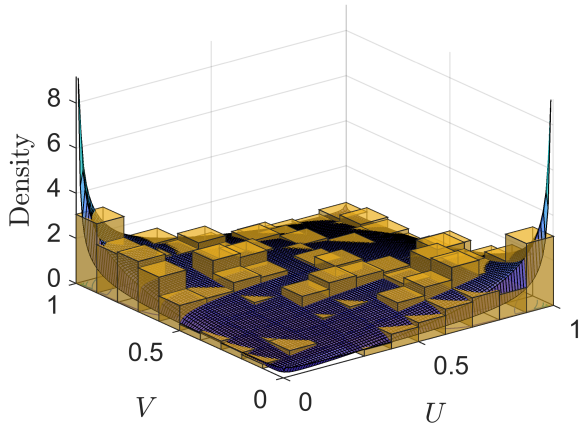


Figure 5.5: Fitted copula density on the histogram of copula data.

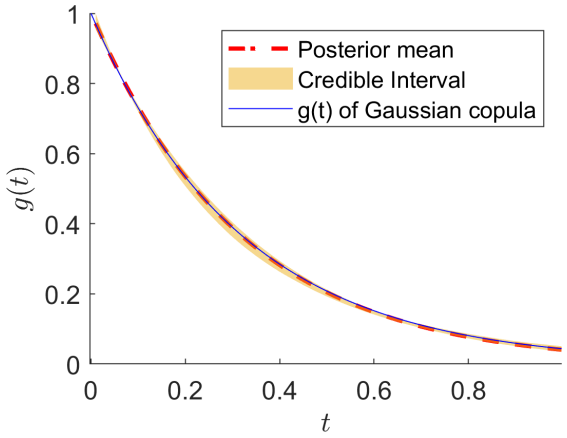


Figure 5.6: Fitted generator with 95% credible interval.

Chapter 6

Discussion

In this dissertation, we provided a method to estimate the generator of elliptical copulas in Bayesian framework. It is more convenient and rigorous than old approaches in the literature. Our method is inspired by the iterative algorithm in Derumigny and Fermanian (2022) and has better performance in the simulation studies. In the application of the water quality data, our approach also shows excellent results.

The major contribution of our work is that we provided a framework, instead of a single model, to estimate the generators of elliptical copulas. The framework enables researchers to construct new models or extend the two models we provided according to some certain situations. Besides, we offered a steady way to evaluate the copula likelihood, including evaluation of the marginal PDF f_g and the inverse CDF Q_g. Although we still need to find out at which point the generator starts to vanish into the x-axis. It can be a difficult task in practice and may require some trial and error.

For future work, we might add more flexible models in our existing framework. Besides, it would be nice to find a better way to evaluate f_g and Q_g, which can avoid truncating the generator g. We shall also apply our method on real world data of higher dimensions and evaluate the results with tests of goodness of fit introduced by Genest et al. (2009).

References

Cambanis, S., Huang, S., and Simons, G. (1981), "On the theory of elliptically contoured distributions," *Journal of Multivariate Analysis*, 11, 368–385.

de Boor, C. (1978), *A Practical Guide to Splines*, Applied Mathematical Sciences, Springer.

Derumigny, A. and Fermanian, J.-D. (2022), "Identifiability and Estimation of Meta-Elliptical Copula Generators," *Journal of Multivariate Analysis*, 190, 104962.

Eilers, P. and Marx, B. (2021), *Practical Smoothing: The Joys of P-splines*, Cambridge University Press.

Fang, H.-B., Fang, K.-T., and Kotz, S. (2002), "The Meta-elliptical Distributions with Given Marginals," *Journal of Multivariate Analysis*, 82, 1–16.

— (2005), "Corrigendum to "The meta-elliptical distributions with given marginals" [J. Multivariate Anal. 82 (2002) 1–16]," *Journal of Multivariate Analysis*, 2.

Fang, K. T. and Zhang, Y. T. (1990), *Generalized multivariate analysis*, Springer-Verlag, Berlin; Science Press Beijing, Beijing.

Genest, C. (2018), "Part 1: A Gentle Introduction to Copula Modeling and Rank-Based Inference," Joint Statistical Meetings, Vancouver Convention Centre.

Genest, C., Favre, A.-C., Béliveau, J., and Jacques, C. (2007), "Metaelliptical Copulas and Their Use in Frequency Analysis of Multivariate Hydrological Data," *Water Resources Research*, 43.

Genest, C. and MacKay, R. J. (1986), "Copules archimédiennes et familles de lois bidimensionnelles dont les marges sont données," *Statistical Society of Canada*, 14, 145–159.

Genest, C. and Rivest, L.-P. (1993), "Statistical Inference Procedures for Bivariate Archimedean Copulas," 88, 1034–1043.

Genest, C., Rémillard, B., and Beaudoin, D. (2009), "Goodness-of-fit tests for copulas: A review and a power study," *Insurance: Mathematics and Economics*, 44, 199–213.

Gómez, E., Gómez-villegas, M. A., and Marín, J. M. (2003), "A survey on continuous elliptical vector distributions," *Revista Matemática Complutense*, 16, 345 – 361.

Joe, H. (2014), *Dependence Modeling with Copulas*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability, Taylor & Francis.

Kelker, D. (1970), "Distribution Theory of Spherical Distributions and a Location-Scale Parameter Generalization," *Sankhyā: The Indian Journal of Statistics*, 13.

Lemonte, A. J. and Patriota, A. G. (2011), "Multivariate elliptical models with general parameterization," *Stat. Methodol.*, 8, 389–400.

Li, F., Tang, Y., and Wang, H. J. (2019), "Copula-based semiparametric analysis for time series data with detection limits," *Canadian Journal of Statistics*, 47.

Liebscher, E. (2005), "A Semiparametric Density Estimator Based on Elliptical Distributions," *Journal of Multivariate Analysis*, 92, 205–225.

McNeil, A. J. (2008), "Sampling nested Archimedean copulas," *Journal of Statistical Computation and Simulation*, 78, 567–581.

McNeil, A. J. and Nešlehová, J. (2009), "Multivariate Archimedean copulas, d-monotone functions and ℓ_1-norm symmetric distributions," *The Annals of Statistics*, 37.

Müller, A. and Scarsini, M. (2005), "Archimedean Copulae and Positive Dependence," *Journal of Multivariate Analysis*, 93, 434–445.

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

Steerneman, A. and van Perlo-ten Kleij, F. (2005), "Spherical distributions: Schoenberg (1938) revisited," *Expositiones Mathematicae*, 23, 281–287.

Chapter 7

Appendix: Matlab Code

7.1 Utilities

```
function [T_grid, g1_grid] = get_g1(g, dim, g_trunc)
% Return g1 values on grid T_d, computed by Riemann Summ
% input:
%   g: generator
% output:
%   T_grid: grid
%   g1_grid: evaluation of g1 on T_grid

% make grid
dx=0.01;
T_grid_raw=(0: dx: g_trunc)';

T_grid_plus = T_grid_raw.^((dim-3)/2); % evaluate first term on grid
g_grid=g(T_grid_raw);  % evaluate g on grid

n1=length(T_grid_raw);
g1_grid_raw = zeros(n1-1, 1); % we can only get g1(1:n1-1)

for j = 1:(n1-1)
g1_grid_raw(j)=dot(g_grid((j+1):n1), T_grid_plus(2:(n1-j+1)));
end

T_grid=T_grid_raw(1:n1-1); % only n1-1 points are valid!
g1_grid = g1_grid_raw*dx * (pi^((dim-1)/2) / gamma((dim-1)/2));
```

```
25
26  end
```

Listing 7.1: Evaluation of g_1

```
1   function [sp] = BsplineFit_positive(x, y, xl, xr, nseg, bdeg)
2   % x: covariate values of the sampe
3   % y: response values of the same
4   % xl: left bound of segmentation
5   % xr: right boundary for segmentation
6   % nseg: the number of segments between xl and xr
7   % bdeg: degree, which is equal to order -1
8   [B, knots] = bbase(x, xl, xr, nseg, bdeg);
9   % number of knots
10  nknots=length(knots);
11  order=bdeg+1;
12  % number of basis functions
13  nbasis=nknots-order;
14
15  lambda=1000;
16  D = diff(eye(nbasis), 2);
17  BTB=B'*B; %nbasis by nbasis
18  DTD=D'*D; %nbasis by nbasis
19  BTY=B'*y; % nbasis by N times N by 1= nbasis by 1
20
21  % get coefficients
22  kappa = 1e8;
23  D0=eye(nbasis);
24  a=ones(nbasis, 1);
25  epsilon=1.0e-4;
26
27  for i=1:100
28  v_vec=(sign(D0*a)<=0);
29  V=diag(v_vec);
30
```

```
% update a
a_new=(BTB + lambda * DTD+kappa*D0'*V*D0)\BTY;
dz=max(abs(a_new-a));

if dz< epsilon
%fprintf(strcat("Converged after ", num2str(i), " iternations!\n"));
break;
end
if  i==100 && dz> epsilon
fprintf("Warning: Divergent after 100 iterations!\n");
end
a=a_new;
end
a=max(a, realmin);
sp=spmak(knots, a');
end
```

Listing 7.2: Bspline smoothing with constraint of positiveness.

```
function [f1_sp, f1] = get_f1(g, dim, g_trunc)
% return f1
% g: generator
% dim: dimension

[T_grid, g1_grid] = get_g1(g, dim, g_trunc);

% get sample of f1 on grid
f1x=[-flipud(sqrt(T_grid)); sqrt(T_grid)];
f1y=[flipud(g1_grid); g1_grid];

% get spline function of f1
xl=min(f1x);
xr=max(f1x);

n1=length(f1x);
```

```
%nseg=n1;
nseg=n1/10;
%nseg=1000;
bdeg=3;

% get b form of f1, f1_sp must be positive between -sqrt(g_trunc) and
 sqrt(g_trunc)
f1_sp=BsplineFit_positive(f1x, f1y, xl, xr, nseg, bdeg);

% normalize f1_sp
f1inte=diff(fnval(fnint(f1_sp), [xl; xr]));
f1_sp.coefs=f1_sp.coefs./f1inte;

f1=@(x) fnval(f1_sp, x);

end
```

Listing 7.3: Approximation of f_g.

```
function [g_n] = g_normalize(g, dim)
% return normalized copula generator.

I1=integral(@(s) s.^(dim/2-1).*g(s), 0, Inf);
SD=pi^(dim/2)/gamma(dim/2);
g_n=@(t) 1/SD/I1*g(t);

end
```

Listing 7.4: Normalization of generator.

```
function [g_s, A, b] = g_standardize(g, dim, b)
% g: a normalized generator
% g take column input, return column output
% dim: copula dimension
% return standardized copula generator, which is unique.
```

```
6
7   %make this interation faster and more robust!
8   I2=integral(@(s) s.^((dim-3)/2).*g(s), 0, Inf);
9
10  A=(b/I2/sd_EllipCop(dim-1))^2;
11  g_s=@(t) A^(dim/2).*g(A.*t);
12
13  end
```

Listing 7.5: Standardization of generator.

7.2 Model Based on Noncentral F Mixture

```
1   function gt= g_ncF(t, dim, v2, DELTA)
2   % return g(t) from a noncentral F distribution
3   % v2: seconde degree of freedom
4   % DELTA: noncentrality parameters
5
6   gt=gamma(dim/2)/(pi^(dim/2)).*(t.^(1-dim/2)).*ncfpdf(t, dim, v2, DELTA)
     ;
7
8   end
```

Listing 7.6: Generator based on noncentral F-distribution.

```
1   function [gmix] = g_ncFMix(t, dim, Weights, v2s, DELTAs)
2   % Return the generator from a noncentral F Mixture
3   % t: ROW or COLUMN vector
4   % Weights: mixing proportions
5
6   ndat=length(t);
7   K=length(Weights);
8
9   if (size(t, 1)>size(t, 2))
10  % if t is column vector
```

```
% repeat by columns
tMat=repmat(t, 1, K); % n by K

% parameters are always columns
% transpose and repeat byt rows
WeightsMat=repmat(Weights', ndat, 1); % n by K
v2sMat=repmat(v2s', ndat, 1);
deltasMat=repmat(DELTAs', ndat, 1);

g_mat=g_ncF(tMat, dim, v2sMat, deltasMat);

gmix= sum(WeightsMat.*g_mat, 2);
else
% if t is row vector, parameters are still columns!
% repeat by rows
tMat=repmat(t, K, 1); % K by n

WeightsMat=repmat(Weights, 1, ndat); % K by n
v2sMat=repmat(v2s, 1, ndat);
deltasMat=repmat(DELTAs,1, ndat);

g_mat=g_ncF(tMat, dim, v2sMat, deltasMat);

gmix= sum(WeightsMat.*g_mat, 1);
end

end
```

Listing 7.7: Generator based on noncentral F mixture.

```
function [g_s, f1, Q1] = getFuncF(dim, Weights, v2s, DELTAs, g_trunc, b
 )
% Return related functions according to dim and mixParameters

g =@(t) g_ncFMix(t, dim, Weights, v2s, DELTAs);
```

```
g_s = g_standardize(g, dim, b);

[f1_sp, f1] = get_f1(g_s, dim, g_trunc);
cdf1 = get_cdf1(f1_sp, g_trunc);
Q1 = get_Q1(cdf1, g_trunc);

end
```

Listing 7.8: Function conversions for model based on noncentral F mixture.

```
function [logPost] = logPostCopulaNCF(U, dim, Omega, Etas, v2s, DELTAs,
  g_trunc, b)
% Return log Likelihood for Copula data at mixParameters
% U: Copula data ndat by dim

ndat=size(U,1);
% Transform back to constrained
Weights=Etas_to_Weights(Etas);
K=length(Weights); % number of components
%sum(Weights)

[g, f1, Q1] = getFuncF(dim, Weights, v2s, DELTAs, g_trunc, b);
% Get pseudo data
Z=Q1(U);
Rsq=zeros(ndat, 1); % ndat by 1 vector
for i=1:ndat
Rsq(i)=Z(i, :)/Omega*Z(i, :)';
end

logNum=reallog(g(Rsq));

f1Z=f1(Z);
logDen=reallog(f1Z); % n by dim
logLike=sum(logNum, 'all')-sum(logDen, 'all');

```

```
logPWeights = dirichletlogpdf(Weights, 1/K*ones(K, 1));
JWeights = Jacobian_WtoEta(Etas);
logJWeights=reallog(abs(det(JWeights)));

logPost=logLike+logPWeights+logJWeights;

end
```

Listing 7.9: Log posterior distribution for the model based on noncentral F mixture.

```
%% Initialize parameters
% Weights trans
Etas=normrnd(0, 1, K-1, 1);
Weights=Etas_to_Weights(Etas);

% v2 trans and DELTA trans
v2s=unifrnd(v2L, v2U, K, 1);
DELTAs=unifrnd(DELTAL, DELTAU, K, 1);

% hold values after burn in, each column is one sample
weights_chain=zeros(K, nloop-nwarmup);
v2_chain=zeros(K, nloop-nwarmup);
DELTA_chain=zeros(K, nloop-nwarmup);
e_chain=zeros(3, nloop-nwarmup);

for i=1:nloop

if mod(i, 1)==0
fprintf("loop: %d\n", i)
end

% Draw v2_trans
v2s1=unifrnd(v2s-v2_tuning, v2s+v2_tuning);
% repeat untile get proposal in range
while (any(v2s1<v2L) || any(v2s1>v2U))
```

```
v2s1=unifrnd(v2s-v2_tuning, v2s+v2_tuning);
end

A=logPostCopulaNCF(U, dim, Omega, Etas, v2s1, DELTAs, g_trunc, b);
B=logPostCopulaNCF(U, dim, Omega, Etas, v2s, DELTAs, g_trunc, b);
logRate=A-B ;
Rate=exp(logRate);
e1=min(1,Rate);
u=rand;
if u<=e1
v2s=v2s1;
end

% Draw DELTA_trans
DELTAs1=unifrnd(DELTAs-DELTA_tuning, DELTAs+DELTA_tuning);
% repeat untile get proposal in range
while (any(DELTAs1<DELTAL) || any(DELTAs1>DELTAU))
DELTAs1=unifrnd(DELTAs-DELTA_tuning, DELTAs+DELTA_tuning);
end

A=logPostCopulaNCF(U, dim, Omega, Etas, v2s, DELTAs1, g_trunc, b);
B=logPostCopulaNCF(U, dim, Omega, Etas, v2s, DELTAs, g_trunc, b);
logRate=A-B;
Rate=exp(logRate);
e2=min(1,Rate);
u=rand;
if u<=e2
DELTAs=DELTAs1;
end

% Draw Etas
Etas1=unifrnd(Etas-Eta_tuning, Etas+Eta_tuning);

A=logPostCopulaNCF(U, dim, Omega, Etas1, v2s, DELTAs, g_trunc, b);
```

```
B=logPostCopulaNCF(U, dim, Omega, Etas, v2s, DELTAs, g_trunc, b);
logRate=A-B;
Rate=exp(logRate);
e3=min(1,Rate);
u=rand;
if u<=e3
Etas=Etas1; %update eta
Weights=Etas_to_Weights(Etas);
end

% record value after burn in
% each COLUMN is a sample
if i>nwarmup
v2_chain(:, i-nwarmup)=v2s;
DELTA_chain(:, i-nwarmup)=DELTAs;
weights_chain(:, i-nwarmup)=Weights;
e_chain(:, i-nwarmup)=[e1; e2; e3];
end

end
```

Listing 7.10: MCMC loop for noncentral F mixture.

7.3 Model Based on Truncated Normal Mixture

```
function gt= g_trcnorm_Lieb(t, dim, a, mu, sigma)
% a: any positive value for the transformation function
const=gamma(dim/2)/(pi^(dim/2));
y=-a+(a^(dim/2)+t.^(dim/2)).^(2/dim);
term1=(a^(dim/2)+t.^(dim/2)).^(2/dim-1).*t.^(dim/2-1);
gt=const.*term1.*trcnormpdf(y, mu, sigma);

end
```

Listing 7.11: Generator based on truncated normal distribution.

```
function [gmix] = g_trcnormMix_Lieb(t, dim, a, Weights, Mu)
% Return the generator from a Gamma Mixture
% t: ROW or COLUMN vector
% Weights: mixing proportions

Sigma=1;

ndat=length(t);
K=length(Weights);

if (size(t, 1)>size(t, 2))
% if t is column vector
% repeat by columns
tMat=repmat(t, 1, K); % n by K

% parameters are always columns
% transpose and repeat byt rows
WeightsMat=repmat(Weights', ndat, 1); % n by K
MuMat=repmat(Mu', ndat, 1);

g_mat=g_trcnorm_Lieb(tMat, dim, a, MuMat, Sigma);

gmix= sum(WeightsMat.*g_mat, 2);
else
% if t is row vector, parameters are still columns!
% repeat by rows
tMat=repmat(t, K, 1); % K by n

WeightsMat=repmat(Weights, 1, ndat); % K by n
MuMat=repmat(Mu, 1, ndat);

g_mat=g_trcnorm_Lieb(tMat, dim, a, MuMat, Sigma);

gmix= sum(WeightsMat.*g_mat, 1);
```

```
end

end
```

Listing 7.12: Generator based on truncated normal mixture.

```
function [logPost] = logPostCopulaTrcnorm_Lieb(U, dim, a, Omega, Etas,
 Mu, g_trunc, b)
% Return log Likelihood for Copula data at mixParameters
% U: Copula data ndat by dim
% mixParametersTrans 3K-1 include:
% etas (K-1), AlphaTrans(K)

ndat=size(U,1);

% Transform back to constrained
Weights=Etas_to_Weights(Etas);
K=length(Weights); % number of components

[g, f1, Q1] = getFuncTrcnorm_Lieb(dim, a, Weights, Mu, g_trunc, b);

% Get pseudo data
Z=Q1(U);
Rsq=zeros(ndat, 1); % ndat by 1 vector
for i=1:ndat
Rsq(i)=Z(i, :)/Omega*Z(i, :)'; %make this more efficient!
end

logNum=reallog(g(Rsq));

f1Z=f1(Z);
logDen=reallog(f1Z); % n by dim
logLike=sum(logNum, 'all')-sum(logDen, 'all');

logPWeights = dirichletlogpdf(Weights, 1/K*ones(K, 1));
```

```
JWeights = Jacobian_WtoEta(Etas);
logJWeights=reallog(abs(det(JWeights)));

logPost=logLike+logPWeights+logJWeights;

end
```

Listing 7.13: Log posterior distribution for the model based on truncated normal mixture.

```
function [g_s, f1, Q1] = getFuncTrcnorm_Lieb(dim, a, Weights, Mu,
 g_trunc, b)
% Return related functions according to dim and mixParameters

g =@(t) g_trcnormMix_Lieb(t, dim, a, Weights, Mu);
g_s = g_standardize(g, dim, b);

[f1_sp, f1] = get_f1(g_s, dim, g_trunc);
cdf1 = get_cdf1(f1_sp, g_trunc);
Q1 = get_Q1(cdf1, g_trunc);

end
```

Listing 7.14: Function conversions for model based on truncated normal mixture.

```
%% Initialize parameters
% Weights trans
Etas=normrnd(0, 1, K-1, 1);
Weights=Etas_to_Weights(Etas);
Mu=unifrnd(Mu_L, Mu_U, K, 1);

% MCMC chains, each column is one sample
weights_chain=zeros(K, nloop-nwarmup);
Mu_chain=zeros(K, nloop-nwarmup);
e_chain=zeros(2, nloop-nwarmup);

```

```
for i=1:nloop

if mod(i, 100)==0
fprintf("loop: %d\n", i)
end

% Draw Mu
Mu1=unifrnd(Mu-Mu_tuning, Mu+Mu_tuning);
% repeat untile get proposal in range
while (any(Mu1<Mu_L) || any(Mu1>Mu_U))
Mu1=unifrnd(Mu-Mu_tuning, Mu+Mu_tuning);
end

A=logPostCopulaTrcnorm_Lieb(U, dim, a, Omega, Etas, Mu1, g_trunc, b);
B=logPostCopulaTrcnorm_Lieb(U, dim, a, Omega, Etas, Mu, g_trunc, b);


logRate=A-B;
Rate=exp(logRate);
e1=min(1,Rate);
u=rand;
if u<=e1
Mu=Mu1;
end

% Draw Etas
Etas1=unifrnd(Etas-Eta_tuning, Etas+Eta_tuning);

A=logPostCopulaTrcnorm_Lieb(U, dim, a, Omega, Etas1, Mu, g_trunc, b);
B=logPostCopulaTrcnorm_Lieb(U, dim, a, Omega, Etas, Mu, g_trunc, b);
logRate=A-B;
Rate=exp(logRate);
e2=min(1,Rate);
u=rand;
```

```
46  if u<=e2
47  Etas=Etas1; %update eta
48  Weights=Etas_to_Weights(Etas);
49  end
50
51  % record value after burn in
52  % each COLUMN is a sample
53  if i>nwarmup
54  Mu_chain(:, i-nwarmup)=Mu;
55  weights_chain(:, i-nwarmup)=Weights;
56  e_chain(:, i-nwarmup)=[e1; e2];
57
58  end
59
60  end
```

Listing 7.15: MCMC loop for model based on truncated normal mixture.

Curriculum Vitae

Panfeng Liang is from People's Republic of China. He went to Minzu University of China in September 2009 and received his bachelor's degree in Computer Science in July 2013. He came to The University of Texas at El Paso in August 2015. While pursuing his master's degree in Statistics he worked as a Teaching Assistant and Research Assistant. In the Spring of 2018, he started the PhD program in Computational Science in The University of Texas at El Paso.

During his PhD years, he worked with his advisor Dr. Ori Rosen, doing research on statistical copulas. Beside of his research, he worked as a research assistant in the statistical consulting lab for three years and obtained much experience of doing statistical analysis on medical data with medical doctors.

After his graduation, he plans to stay in the statistical consulting lab and learn more statistical methods used on real world data research.

Email: pliang@miners.utep.edu