2017-01-01

The Nonparametric Estimation Of Elliptical Distributions

Panfeng Liang
University of Texas at El Paso, mucvictor@gmail.com

THE NONPARAMETRIC ESTIMATION OF ELLIPTICAL DISTRIBUTIONS

PANFENG LIANG

Master's Program in Statistics

APPROVED:

Ori Rosen, Chair, Ph.D.

Vanessa Lougheed, Ph.D.

Michael Pokojovy, Ph.D.

Charles Ambler Ph.D.
Dean of the Graduate School

THE NONPARAMETRIC ESTIMATION OF ELLIPTICAL DISTRIBUTIONS

by

PANFENG LIANG

THESIS

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

MASTER OF SCIENCE

Master's Program in Statistics

THE UNIVERSITY OF TEXAS AT EL PASO

December 2017

Abstract

In practice, many multivariate datasets have identical marginal distributions. Elliptical distributions can be used to model many of those datasets. In this thesis, we will propose a Bayesian method using Markov chain Monte Carlo (MCMC) methods to estimate the density function underlying multivariate datasets assuming it is an elliptical distribution.

Table of Contents

Chapter 1

Introduction

1.1 Elliptical distribution

A continuous random vector $\boldsymbol{X} = (X_1, ..., X_p)$ has an elliptical distribution $\epsilon_p(\boldsymbol{\mu}, \Omega, g)$ with location vector $\boldsymbol{\mu}$, covariance matrix Ω and generator function g, if its density function is

$$f(\boldsymbol{x}; \boldsymbol{\mu}, \Omega, g) = c_p |\Omega|^{-1/2} g((\boldsymbol{x} - \boldsymbol{\mu})' \Omega^{-1} (\boldsymbol{x} - \boldsymbol{\mu})), \tag{1.1.1}$$

where

1. $c_p = \Gamma(p/2)\pi^{-p/2} / \int_0^\infty t^{p/2-1} g(t) dt$.
2. $\boldsymbol{\mu} \in \mathbb{R}^p$.
3. Ω is a $p \times p$ positive definite matrix.
4. g is a non-negative function on $[0, \infty)$ such that $\int_0^\infty t^{p/2-1} g(t) dt < \infty$.

If $\boldsymbol{X} \sim \epsilon_p(\boldsymbol{\mu}, \Omega, g)$, then $\boldsymbol{X} \sim \epsilon_p(\boldsymbol{\mu}^*, \Omega^*, g^*)$ if and only there are two positive numbers a and b such that $\mu^* = \mu,\ \Sigma^* = a\Sigma$ and $g^*(t) = bg(at)$, for all $t \in \mathbb{R}$ (see Gómez et al. (2003)).

1.2 Stochastic form

A p-variate vector $\boldsymbol{X}$ from $\epsilon_p(\boldsymbol{\mu}, \Omega, g)$ can also be expressed in the following stochastic representation (see Fang et al. (2002) and Fang et al. (2005))

$$\boldsymbol{X} = \boldsymbol{\mu} + RA\boldsymbol{U}, \tag{1.2.1}$$

where

1. A is the Cholesky factor of Ω ($\Omega = AA'$).

2. $\boldsymbol{U}$ is a random vector uniformly distributed on the p-dimensional unit sphere of $\mathbb{R}^p$.

3. R is a continuous nonnegative random variable, independent of $\boldsymbol{U}$, whose density function is

$$h(r) = \frac{2}{\int_0^\infty t^{p/2-1} g(t)dt} r^{p-1} g(r^2) \mathbb{1}_{[0,\infty]}(r). \tag{1.2.2}$$

Let $\boldsymbol{Y} = \Omega^{-1/2}(\boldsymbol{X} - \boldsymbol{\mu})$, then $\boldsymbol{Y}'\boldsymbol{Y} = (\boldsymbol{X} - \boldsymbol{\mu})'\Omega^{-1}(\boldsymbol{X} - \boldsymbol{\mu})$. By using the stochastic representation (1.2.1) of $\boldsymbol{X}$, it follows that $\boldsymbol{Y}'\boldsymbol{Y} = R\boldsymbol{U}'A'\Omega^{-1}A\boldsymbol{U}R = R\boldsymbol{U}'\boldsymbol{U}R = R^2$. Also, from (1.2.2), it follows that the density function (1.1.1) can be expressed as

$$f(r; A) = \frac{\Gamma(p/2)}{2\pi^{p/2}} |A|^{-1} r^{1-p} h(r). \tag{1.2.3}$$

In (1.2.3), the Cholesky factor A and the univariate density function $h(r)$ are unknown.We propose to estimate A parametrically and $h(r)$ nonparametrically.

Chapter 2

Dirichlet Process

2.1 Introduction

The Dirichlet process (DP) was originally defined in Ferguson (1973), as follows.

Definition 1 *Let γ be a non-null finite measure (nonnegative and finitely additive) on measurable space $(\mathscr{X}, \mathscr{A})$. We say P is a Dirichlet process with parameters γ if for every $k = 1, 2, \ldots,$ and measurable partition $(B_1, \cdots, B_k)$ of $\mathscr{X}$, the distribution of $(P(B_1), \cdots, P(B_k))$ is Dirichlet, $\mathscr{D}(\gamma(B_1), \cdots, \gamma(B_k))$.*

2.2 Metaphors

2.2.1 Stick-breaking process

Sethuraman (1994) gave a constructive definition of the Dirichlet process. The stick-breaking process consists of defining an infinite sequence of mixing proportions by

$$w_1 = \beta_1,\ w_k = \beta_k \prod_{j=1}^{k-1}(1-\beta_j), \quad k = 2,\ 3, \ldots,$$

where $\beta_1, \beta_2 \ldots \overset{iid}{\sim} \text{Beta}(1, \gamma)$. Sethuraman (1994) showed that the DP is actually an infinite sum of the form $P = \sum_{k=1}^{\infty} w_k \delta_{\boldsymbol{\phi}_k}$ that obeys the definition of the stick-breaking process.

Proof of the equivalence of the Stick-breaking process and the DP

Three lemmas will be used to prove that the stick-breaking process is a DP.

Lemma 1 *Let* $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_k)$ *and* $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots, \delta_k)$ *be k-dimensional vectors. Let* $\boldsymbol{U}$, $\boldsymbol{V}$ *be independent k-dimensional random vectors with Dirichlet distribution* $Dir(\boldsymbol{\gamma})$ *and* $Dir(\boldsymbol{\delta})$*, respectively. Let* W *be independent of* $(\boldsymbol{U}, \boldsymbol{V})$ *having a Beta distribution* $Beta(\sum \gamma_j, \sum \delta_j)$*. Then the distribution of* $W\boldsymbol{U} + (1 - W)\boldsymbol{V}$ *is the Dirichlet distribution* $Dir(\boldsymbol{\gamma} + \boldsymbol{\delta})$*.*

Lemma 2 *Let* $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_k)$*, and let* $\beta_j = \gamma_j / \sum \gamma_j$*. Then*

$$\sum \beta_j Dir(\boldsymbol{\gamma} + \boldsymbol{e}_j) = Dir(\boldsymbol{\gamma}),$$

where $\boldsymbol{e}_j$ *is a unit basis vector with the* j*th element equal to* 1 *and the other elements equal to* 0*. The left-hand side can also be written as* $E[Dir(\boldsymbol{\gamma} + \boldsymbol{X})]$*, where* $\boldsymbol{X}$ *is a random vector that takes the value* $\boldsymbol{e}_j$ *with probability* β_j*.*

Lemma 3 *Let* W*,*U *be a pair of random variables where* W *takes values in* $[-1, 1]$ *and* U *takes values in a linear space. Suppose that* V *is a random variable taking values in the same linear space as* U *and which is independent of* (W, U) *and satisfies the distributional equation*

$$V \stackrel{st}{=} U + WV.$$

Suppose that $P(|W| = 1) \neq 1$*. Then there is only one distribution for* V *that satisfies Lemma 3. "* $\stackrel{st}{=}$*" means having the same distribution.*

Assume G is a Dirichlet process by Ferguson's definition with concentration parameter α_0 and base distribution G_0, and denote it by $DP(\alpha_0, G_0)$. We generate an infinite list $(\phi_1, \phi_2, \ldots, \phi_k, \ldots) \sim G_0$ and form $G' = \sum_{k=1}^{\infty} \pi_k \delta_k$, where $\pi_1 = \beta_1$, $\pi_k = \beta_k \prod_{j=1}^{k-1}(1 - \beta_j)$, $k = 2, 3, \ldots,$ and $\beta_1, \beta_2 \ldots \stackrel{iid}{\sim} \text{Beta}(1, \alpha_0)$. Our goal is to show that $G' = G$ in distribution. The strategy is to show that for all partitions $(A_1, A_2, \ldots, A_k)$ of the sample space of G_0, G' has the finite Dirichlet marginals

$$(G'(A_1), G'(A_2), \ldots, G'(A_k)) \sim Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k)).$$

Let f be a deterministic function that transforms the random variables $\phi_k s$ and corresponding $\beta_k s$ into the sum of infinite terms

$$G' = f(\boldsymbol{\phi}, \boldsymbol{\beta}) = \sum_{k=1}^{\infty} \pi_k \delta_k.$$

The function f can be interpreted as the function used to construct a stick-breaking process. Let $\boldsymbol{\beta}^*$ denote $(\beta_2, \beta_3, \ldots)$, with the first element of the infinite sequence removed. Now,

$$G' = \pi_1 \delta_{\phi_1} + (1 - \pi) f(\boldsymbol{\beta}^*, \boldsymbol{\phi}^*) = \pi_1 \delta_{\phi_k} + (1 - \pi_1) G'',$$

where $G' \stackrel{d}{=} G''$ ("$\stackrel{d}{=}$" means equality in distribution).

Then, we have

$$G' \stackrel{st}{=} \pi_1 \delta_{\phi_k} + (1 - \pi_1) G'.$$

If we have a partition $(A_1, A_2, \ldots, A_k)$ of Φ, the sample space of G_0 in Ferguson's definition, then

$$\begin{pmatrix} G'(A_1) \\ \vdots \\ G'(A_k) \end{pmatrix} \stackrel{st}{=} \pi_1 \begin{pmatrix} \delta_{\phi_1}(A_1) \\ \vdots \\ \delta_{\phi_1}(A_k) \end{pmatrix} + (1 - \pi_1) \begin{pmatrix} G'(A_1) \\ \vdots \\ G'(A_k) \end{pmatrix}. \tag{2.2.1}$$

Note that

$$\delta_{\phi_1}(A_k) = \begin{cases} 1 & \phi_1 \in A_1 \\ 0 & \text{otherwise.} \end{cases}$$

By Lemma 3, if there is a distribution of $(G'(A_1), \ldots, G'(A_k)$ that satisfies (2.2.1), then it must be unique. If we can prove the $Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k))$ satisfies the Equation (2.2.1), then it must be the unique distribution of $(G'(A_1), \ldots, G'(A_k))$,

$$(G'(A_1), \ldots, G'(A_k) \sim Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k)).$$

This is a proof that $G' = DP(\alpha_0, G_0)$.

Let $\boldsymbol{Z} \sim Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k))$ and $\boldsymbol{X} = (\delta_{\phi_1}(A_1), \ldots, \delta_{\phi_1}(A_k))$, then $\boldsymbol{X} \sim Multinomial(G_0(A_1), G_0(A_2), \ldots, G_0(A_k))$. So $P(\delta_{\phi_1}(A_k) = 1) = G_0(A_k)$. We want

to show the distribution of $\pi_1 \boldsymbol{X} + (1-\pi_1)\boldsymbol{Z}$ is still $Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k))$. In other words, (2.2.1) is satisfied by $Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k))$.
Conditioning on $\boldsymbol{X} = \boldsymbol{e}_j$

$$(\pi_1 \boldsymbol{X} + (1-\pi_1)\boldsymbol{Z} | \boldsymbol{X} = \boldsymbol{e}_j) \stackrel{d}{=} \pi_1 Dir(\boldsymbol{e}_j) + (1-\pi_1)\boldsymbol{Z}.$$

Since, $\boldsymbol{Z} \sim Dir(\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k))$, denoted as $Dir(\boldsymbol{\alpha})$,
where $\boldsymbol{\alpha} = (\alpha_0 G_0(A_1), \alpha_0 G_0(A_2), \ldots, \alpha_0 G_0(A_k))$ by Lemma 1,

$$(\pi_1 \boldsymbol{X} + (1-\pi_1)\boldsymbol{Z} | \boldsymbol{X} = \boldsymbol{e}_j) \stackrel{d}{=} Dir(\boldsymbol{e}_j + \boldsymbol{\alpha}).$$

$P(\boldsymbol{X} = \boldsymbol{e}_j) = G_0(A_j) = \alpha_j / \sum \alpha_j$, where α_j is the jth element of $\boldsymbol{\alpha}$ and is equal to $\alpha_0 G_0(A_j)$. To obtain the distribution of $(\pi_1 \boldsymbol{X} + (1-\pi_1)\boldsymbol{Z})$, we sum over all the values of $\boldsymbol{X}$,

$$\sum_{j=1}^{K} \frac{\alpha_j}{\sum \alpha_j} Dir(\boldsymbol{e}_j + \boldsymbol{\alpha}) = Dir(\boldsymbol{\alpha}), \text{by Lemma 2.}$$

Thus, $\pi_1 \boldsymbol{X} + (1-\pi_1)\boldsymbol{Z} \sim Dir(\boldsymbol{\alpha})$. The distribution $Dir(\boldsymbol{\alpha})$ satisfies (2.2.1) and it is the distribution of $(G'(A_1), \ldots, G'(A_k))$. Hence, $G' \stackrel{d}{=} G$.

2.2.2 The Chinese Restaurant Process

Imagine a restaurant with countably infinitely many tables. Customers walk in and sit down at some table. The tables are chosen according to the following random process:

1. The first customer always chooses an unoccupied table.

2. The nth customer chooses an unoccupied table with probability $\frac{\alpha}{n-1+\alpha}$, and an occupied table with probability $\frac{c}{n-1+\alpha}$ where c is the number of people sitting at the table.

After infinitely many customers have entered the restaurant, the stochastic process underlying the CRP is a Dirichlet process.

2.2.3 The Pólya Urn Scheme

Another way to visualize the Dirichlet process is called the modified Pólya urn scheme. We start with an empty urn. Then we proceed as follows:

1. Generate a random number from $U(0,1)$. If it is less than $\frac{\alpha}{\alpha+n}$, where n is the number of balls in the urn, then add a ball with new color into the urn.

2. Otherwise, pick out a ball from the urn and add it back with a new ball of the same color.

After infinitely many balls have been put into the urn, the distribution over infinitely many colors is the same with the distribution over tables in the Chinese restaurant process.

2.3 Mixture of Dirichlet process

The model

$$f(x) = \sum_{j=1}^{K} w_j f_j(x \mid \boldsymbol{\phi}_j) \tag{2.3.1}$$

is a mixture with K components, where f_j is the pdf of a distribution with parameters $\boldsymbol{\phi}_j$. Augmenting the data with random variables $Z_i, Z_i \in \{1, \ldots, K\}$, indicating which component gives rise to $\boldsymbol{X}_i$, we can write (2.3.1) in a hierarchical form as follows:

$$\begin{aligned}(\boldsymbol{X}_i \mid Z_i, \boldsymbol{\Phi}) &\sim f(x_i \mid \boldsymbol{\phi}_{z_i}) \\ Z_i \mid \boldsymbol{w} &\sim \text{Discrete}(w_1, \ldots, w_K),\end{aligned}$$

where $\boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_K)$ and $\boldsymbol{w} = (w_1, \ldots, w_K)$.
The priors on $\boldsymbol{\Phi}$ and $\boldsymbol{w}$ are

$$\begin{aligned}\boldsymbol{\phi}_j &\sim G_0 \\ \boldsymbol{w} &\sim \text{Dir}(\gamma/K, \ldots, \gamma/K).\end{aligned}$$

We show that this is a mixture of Dirichlet process (MDP) when $K \to \infty$.

$$P(z_1, \ldots, z_n \mid \boldsymbol{w}) = \prod_{j=1}^{K} w_j^{n_j},$$

where n_j is the number of observations assigned to component j.

$$\begin{aligned} P(\boldsymbol{Z}) &= \int P(z_1, \ldots, z_n \mid \boldsymbol{w}) \times P(\boldsymbol{w}) d\boldsymbol{w} \\ &= \int \prod_{j=1}^{K} w_j^{n_j} \times \frac{\Gamma(\gamma)}{[\Gamma(\frac{\gamma}{K})]^K} w_1^{\frac{\gamma}{K}-1} \cdots w_K^{\frac{\gamma}{K}-1} d\boldsymbol{w} \\ &= \frac{\Gamma(\gamma)}{\Gamma(n+\gamma)} \prod_{j+1}^{K} \frac{\Gamma(n_j + \frac{\gamma}{K})}{\Gamma(\frac{\gamma}{K})} \end{aligned}$$

$$P(\boldsymbol{w} \mid \boldsymbol{Z}) = \frac{P(\boldsymbol{Z} \mid \boldsymbol{w}) \times P(\boldsymbol{w})}{P(\boldsymbol{Z})} = \mathrm{Dir}(\gamma/K + n_{i1}, \ldots, \gamma/K + n_{ik}),$$

where n_{ik} is the number of observations previously (before the ith) assigned to component k.

$$\begin{aligned} P(z_i = k \mid z_1, \ldots, z_{i-1}) &= \int P(z_i = k \mid \boldsymbol{w}) \times P(\boldsymbol{w} \mid z_1, \ldots, z_{i-1}) d\boldsymbol{w} \\ &= \frac{\prod_{j=1}^{K} \Gamma(\gamma/K + n_{ij})}{\Gamma(\gamma + i)} \times \frac{\Gamma(\gamma + i - 1)}{\prod_{j=1}^{K} \Gamma(\gamma/K + n_{(i-1)j})} \times \\ &\int \frac{\Gamma(\gamma + i)}{\prod_{j=1}^{K} \Gamma(\gamma/K + n_{ij})} \prod_{j=1}^{K} w_j^{\gamma/K + n_{ij} - 1} d\boldsymbol{w} \\ &= \frac{n_{ik} + \gamma/K}{i - 1 + \gamma}. \end{aligned}$$

By taking the limit $K \to \infty$, for components with $n_{ik} > 0$,

$$P(z_i = k \mid z_1, \ldots, z_{i-1}) = \frac{n_{ik}}{i - 1 + \gamma},$$

which is identical to the probability of the ith customer sitting at an occupied table in the Chinese restaurant with n_{ik} customers already sitting at this table.

The probability of any particular unoccupied cluster approaches zero as $K \to \infty$. However,

the total probability assigned to all unoccupied clusters is still positive:

$$\begin{aligned} P(z_i \neq z_j \; \forall \; j < i \mid z_1, \ldots, z_{i-1}) &= 1 - \sum_{k:n_{ik}>0} P(z_i = k \mid z_1, \ldots, z_{i-1}) \\ &= 1 - \sum_{k:n_{ik}>0} \frac{n_{ik}}{i - 1 + \gamma} \\ &= \frac{\gamma}{i - 1 + \gamma}. \end{aligned}$$

Applying this result, the conditional probability of $(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1})$ where $\boldsymbol{\theta}_i = \boldsymbol{\phi}_{z_i}$ becomes

$$P(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1}) = \begin{cases} \frac{1}{i-1+\gamma} \sum_{j=1}^{i-1} \delta(\boldsymbol{\theta}_j = \boldsymbol{\theta}_i), & \text{if } \exists \, \boldsymbol{\theta}_j = \boldsymbol{\theta}_i \text{ for } j = 1, \ldots, i-1 \\ \frac{\gamma}{i-1+\gamma} G_0(\boldsymbol{\theta}_i), & \text{otherwise,} \end{cases} \tag{2.3.2}$$

where $\delta(\boldsymbol{\theta}_j = \boldsymbol{\theta}_i) = 1$ when $\boldsymbol{\theta}_j = \boldsymbol{\theta}_i$ and zero otherwise.

The parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{i-1}$ may not be unique since $\boldsymbol{\theta}_i = \boldsymbol{\phi}_{z_i}$. Since the observations are assumed to be exchangeable, we can regard any observation i as the last one and write the conditional probability of $\boldsymbol{\theta}_i$ given other $\boldsymbol{\theta}_j$ for $j \neq i$ as in (2.3.2).

Chapter 3

Methodology

In what follows we assume that we have $\epsilon_p(\boldsymbol{\mu}, \Omega, g)$ with $\boldsymbol{\mu} = \mathbf{0}$ and that Ω is a correlation matrix. Our goal is to estimate Ω without pre-specifying the generator function g.
Given a sample $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n$ from (1.1.1), and using (1.2.3), the likelihood function is given by

$$L(\Omega|\boldsymbol{X}) = \left[\frac{\Gamma(\frac{p}{2})}{2\pi^{p/2}}|A|^{-1}\right]^n \prod_{i=1}^{n} r_i^{1-p} h(r_i), \tag{3.0.1}$$

where $r_i = (\boldsymbol{X}_i'\Omega^{-1}\boldsymbol{X}_i)^{1/2}$.

3.1 Estimation of $h(r)$ and the correlation matrix Ω

We express $h(r)$ as an infinite mixture

$$h(r) = \sum_{j=1}^{\infty} w_j f(r|\alpha_j, \beta_j), \tag{3.1.1}$$

where

1. $f(r|\alpha_j, \beta_j)$ is the density function of a Gamma(α_j, β_j) distribution (with mean $\frac{\alpha_j}{\beta_j}$).
2. w_i is a mixing probability assigned to each component. Note that in order to satisfy $h(0) = 0$, we require α_j to be greater than 1.

We augment the data with latent indicators z_{ij}, such that

$$z_{ij} = \begin{cases} 1 & \text{if } r_i \text{ is from component } j \\ 0 & \text{otherwise.} \end{cases}$$

The augmented likelihood function is given by

$$L(\Omega|\boldsymbol{X}, Z) = \left[\frac{\Gamma(\frac{p}{2})}{2\pi^{p/2}}|A|^{-1}\right]^n \prod_{i=1}^{n}\left[r_i^{1-p}\prod_{j=1}^{\infty}[w_j f(r_i|\alpha_j, \beta_j)]^{z_{ij}}\right], \tag{3.1.2}$$

where Z is the matrix of all indicators.

3.1.1 Prior distributions

1. $\alpha_1,\ \alpha_2,\ \cdots \overset{iid}{\sim}$ Pareto$(1, c)$, where c is fixed, with pdf

$$\frac{c}{\alpha_j^{c+1}}, \quad \alpha_j > 1.$$

2. $\beta_1,\ \beta_2,\ \cdots \overset{iid}{\sim}$ Gamma(a, b), where a (shape) and b (rate) are fixed.

3. To model the infinite mixture, we use the stick-breaking representation of the Dirichlet process (Sethuraman (1994)), which means that the w_j satisfy

$$w_1 = v_1,\ w_j = v_j\prod_{l=1}^{j-1}(1 - v_l), \quad j = 2,\ 3, \ldots, \tag{3.1.3}$$

   where $v_1, v_2 \ldots \overset{iid}{\sim}$ Beta$(1, \gamma)$. In practice, we truncate the number of components and set it to k, so v_k is set to 1, such that $\sum_{j=1}^{k} w_j = 1$. Another possibility would be to use the slice sampler (Kalli et al. (2011)).

4. $\gamma \sim$ Gamma(η_1, η_2), where η_1 and η_2 are fixed hyperparameters.

5. When $p = 2$, only one correlation parameter has to be estimated, and we place the U$(-1, 1)$ prior on it. We discuss the case $p > 2$ in Section 3.1.4.

3.1.2 Posterior distribution

The posterior distribution is proportional to the likelihood times the prior distributions. In particular,

$$\begin{aligned} P(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{V}, \gamma, \rho | \boldsymbol{X}, Z) \propto & \left[\frac{\Gamma(\frac{p}{2})}{2\pi^{p/2}} |A|^{-1} \right]^n \prod_{i=1}^{n} \left[r_i^{1-p} \prod_{j=1}^{k} \left(w_j \frac{\beta_j^{\alpha_j}}{\Gamma(\alpha_j)} r_i^{\alpha_j - 1} e^{-\beta_j r_i} \right)^{z_{ij}} \right] \\ & \times \prod_{j=1}^{k} \frac{c}{\alpha_j^{c+1}} \\ & \times \prod_{j=1}^{k} \frac{b^a}{\Gamma(a)} \beta_j^{a-1} e^{-b\beta_j} \\ & \times \prod_{j=1}^{k-1} \gamma (1 - v_j)^{\gamma - 1} \\ & \times \gamma^{\eta_1 - 1} e^{-\eta_2 \gamma} \\ & \times \mathbb{1}_{(-1 \leq \rho \leq 1)}, \end{aligned} \tag{3.1.4}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)'$, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_k)'$, $\boldsymbol{v} = (v_1, \ldots, v_k)'$, and $\mathbb{1}$ is an indicator function. The conditional posterior distributions are given by

1. For $j = 1, 2, \ldots, k$,

$$P(\alpha_j | \beta_j, \rho, \boldsymbol{r}, \boldsymbol{z}_j) \propto \frac{\beta_j^{\alpha_j \sum_{i=1}^n z_{ij}} (\prod_{i=1}^n r_i^{z_{ij}})^{\alpha_j}}{\Gamma(\alpha_j)^{\sum_{i=1}^n z_{ij}}} \times \frac{1}{\alpha_j^{c+1}}, \tag{3.1.5}$$

   where $\boldsymbol{r} = (r_1, r_2, \ldots, r_n)'$ and $\boldsymbol{z}_j = (z_{1j}, \ldots, z_{nj})'$.

2. For $j = 1, 2, \ldots, k$,

$$P(\beta_j | \alpha_j, \rho, \boldsymbol{r}, \boldsymbol{z}_j) \propto \beta_j^{\alpha_j \sum_{i=1}^n z_{ij} + a - 1} e^{-\beta_j (\sum_{i=1}^n r_i z_{ij} + b)}, \tag{3.1.6}$$

   which means that $(\beta_j | \alpha_j, \rho, \boldsymbol{r}, \boldsymbol{z}_j) \sim \text{Gamma}(\alpha_j \sum_{i=1}^n z_{ij} + a, \sum_{i=1}^n r_i z_{ij} + b)$.

3.

$$\begin{aligned} P(v_j | Z) &\propto \left[\prod_{i=1}^{n} \prod_{m=j+1}^{k} w_m^{z_{im}} \right] \times (1 - v_j)^{\gamma - 1} \\ &= v_j^{\sum_{i=1}^n z_{ij}} (1 - v_j)^{\sum_{i=1}^n \sum_{m=j+1}^k z_{im} + \gamma - 1}, \end{aligned} \tag{3.1.7}$$

so $(v_j|Z) \sim \text{Beta}(\sum_{i=1}^{n} z_{ij} + 1, \sum_{i=1}^{n} \sum_{m=j+1}^{k} z_{im} + \gamma)$.

4. $(\gamma|\boldsymbol{v}) \sim \text{Gamma}(k + \eta_1 - 1, \eta_2 - \sum_{j=1}^{k-1} \log(1 - v_j))$.

5.
$$P(\rho|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{r}) \propto |A|^{-n} \prod_{i=1}^{n} \left[r_i^{1-p} \prod_{j=1}^{k} (f(r_i|\alpha_j, \beta_j))^{z_{ij}} \right] \times \mathbb{1}_{(-1 \leq \rho \leq 1)}. \tag{3.1.8}$$

3.1.3 Sampling scheme

1. Calculate $r_i^2 = \boldsymbol{X}_i' \Omega^{-1} \boldsymbol{X}_i$ at the beginning of each iteration of the Gibbs sampler using the current estimate of Ω.

2. The conditional distribution of α_j (Equation (3.1.5)) is not a standard distribution, so we use a Metropolis step to sample from it. The proposal distribution is normal with mean $\alpha_j^{(c)}$ and variance δ_1, where δ_1 is a fixed tuning parameter and $\alpha_j^{(c)}$ is the current value of α_j. The proposed value $\alpha_j^{(p)}$ is accepted with probability $\min\left\{1, \frac{p(\alpha_j^{(p)}|\beta_j^{(c)}, \rho^{(c)}, \boldsymbol{r}^{(c)}, \boldsymbol{z}_j^{(c)})}{p(\alpha_j^{(c)}|\beta_j^{(c)}, \rho^{(c)}, \boldsymbol{r}^{(c)}, \boldsymbol{z}_j^{(c)})}\right\}$, provided $\alpha_j^{(p)} > 1$.

3. Sample β_j from $\text{Gamma}(\alpha_j \sum_{i=1}^{n} z_{ij} + a, \sum_{i=1}^{n} r_i z_{ij} + b)$, $j = 1, \ldots, k$.

4. Sample v_j, $j = 1, \ldots, k-1$, from $\text{Beta}(\sum_{i=1}^{n} z_{ij} + 1, \sum_{i=1}^{n} \sum_{m=j}^{k} z_{im} + \gamma)$, and calculate w_j's from the v_j's, according to Equation (3.1.3).

5. Sample γ from $\text{Gamma}(k + \eta_1 - 1, \eta_2 - \sum_{j=1}^{k-1} \log(1 - v_j))$.

6. The conditional posterior distribution of ρ is not a standard distribution, so we use a Metropolis step to sample from it. The proposal distribution is $\text{U}(\rho^{(c)} - \delta_2, \rho^{(c)} + \delta_2)$, where δ_2 is a fixed tuning parameter and $\rho^{(c)}$ is the current value of ρ. The proposed value $\rho^{(p)}$ is accepted with probability $\min\left\{1, \frac{p(\rho^{(p)}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{r})}{p(\rho^{(c)}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{r})}\right\}$, provided $\rho^{(p)} \in [-1, 1]$.

3.1.4 The case $p > 2$

The discussion above is for the case in which $p = 2$. When $p > 2$, the positive definiteness of Ω makes it difficult to work directly with its individual entries. Instead, we parameterize the

Cholesky factor of Ω using hyperspherical coordinates (Rapisarda et al. (2007)), described also in Pourahmadi and Wang (2015). The Cholesky factor is expressed as:

$$A = \begin{bmatrix} 1 & c_{12} & c_{13} & c_{14} & \cdots & c_{1p} \\ 0 & s_{12} & c_{23}s_{13} & c_{24}s_{14} & \cdots & c_{2p}s_{1p} \\ 0 & 0 & s_{23}s_{13} & c_{34}s_{24}s_{14} & \cdots & c_{3p}s_{2p}s_{1p} \\ 0 & 0 & 0 & \prod_{j=1}^{3} s_{j4} & \cdots & c_{4p}\prod_{j=1}^{3} s_{jp} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & c_{p-1,p}\prod_{j=1}^{p-2} s_{jp} \\ 0 & 0 & 0 & 0 & \cdots & \prod_{j=1}^{p-1} s_{jp} \end{bmatrix},$$

where $c_{ij} = \cos(\theta_{ij})$, $s_{ij} = \sin(\theta_{ij})$, and the θ_{ij}'s are some angles. The matrix A is unique if its diagonal entries are positive, or equivalently if the θ_{ij}'s are in the interval $(0, \pi)$. Therefore, we assume a priori that $\theta_{ij} \overset{iid}{\sim} \mathrm{U}(0, \pi)$. Let $\boldsymbol{\theta} = (\theta_{12}, \theta_{13}, \ldots, \theta_{p-1,p})'$, then

$$P(\boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{r}, Z, A) \propto |A|^{-n} \prod_{i=1}^{n} \left[r_i^{1-p} \prod_{u=1}^{k} (f(r_i|\alpha_i, \beta_i))^{z_{ij}} \right] \left| \frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{\theta}} \right|, \tag{3.1.9}$$

where $\boldsymbol{\rho} = (\rho_{12}, \rho_{13}, \ldots, \rho_{p-1,p})'$ and $\left| \frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{\theta}} \right|$ is the Jacobian of the transformation from $\boldsymbol{\rho}$ to $\boldsymbol{\theta}$ (Pourahmadi and Wang (2015)). The Jacobian is given by

$$\left| \frac{\partial \boldsymbol{\rho}}{\partial \boldsymbol{\theta}} \right| = \prod_{i=2}^{p} \prod_{j=1}^{i-1} \frac{\partial \rho_{ij}}{\partial \theta_{ij}} = (-1)^{\sum_{h=0}^{p-1} h} \times \prod_{j=1}^{p-1} \left(\prod_{i=j+1}^{p} s_{ij} \right)^{p-j}. \tag{3.1.10}$$

Since the pdf (3.1.9) is not standard, we sample the θ_{ij} via independent Metropolis steps. The proposal distribution for θ_{ij} is $\mathrm{U}(\theta_{ij}^{(c)} - \delta_3, \theta_{ij}^{(c)} + \delta_3)$, where $\theta_{ij}^{(c)}$ is the current value of θ_{ij}. The proposed value $\theta_{ij}^{(p)}$ is accepted with probability $\min\left\{1, \frac{p(\theta_{ij}^{(p)}|\boldsymbol{\alpha}^{(c)}, \boldsymbol{\beta}^{(c)}, \boldsymbol{r}^{(c)}, Z, A)}{p(\theta_{ij}^{(c)}|\boldsymbol{\alpha}^{(c)}, \boldsymbol{\beta}^{(c)}, \boldsymbol{r}^{(c)}, Z, A)}\right\}$, where $p(\theta_{ij}|\boldsymbol{\alpha}^{(c)}, \boldsymbol{\beta}^{(c)}, \boldsymbol{r}^{(c)}, Z, A)$ is given in Equation (3.1.9).

Chapter 4

Results Based on Simulated Data

Table 4.1 shows different elliptical distributions and their corresponding generating functions, see Genest, Favre, Béliveau, and Jacques (2007) and Lemonte (2011). By the

Table 4.1: Different elliptical distributions and their generating functions.

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

transformation from R^2 to R, the theoretical density function of R is given by

$$h(r) = 2r f_{R^2}(r^2), \tag{4.0.1}$$

where f_{R^2} is the density function of R^2 in Table 4.1. The density function of R can also be derived by Equation (1.2.2).

4.0.1 Data Generation

By using Equation (1.2.1) and the distributions of R^2 in Table 4.1, we can simulate a data set from a specific elliptical distribution as follows:

1. Generate R^2 from the distribution in Table 4.1 and take the square root or generate R directly according to the density function $h(r)$ in Equation (4.0.1).

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

2. Generate $\boldsymbol{U}$ on the unit sphere S_p by

   (a) generating $(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_p) \overset{iid}{\sim} \mathrm{N}(0,1)$;

   (b) $\boldsymbol{U} = (\boldsymbol{Z}_1/|\boldsymbol{Z}|, \ldots, \boldsymbol{Z}_p/|\boldsymbol{Z}|)$ where $|\boldsymbol{Z}| = \sqrt{\boldsymbol{Z}_1^2 + \ldots + \boldsymbol{Z}_p^2}$.

3. Compute the Cholesky factor A of Ω.

4. Deliver $\boldsymbol{X} = RA\boldsymbol{U}$.

4.0.2 Simulation Results

In the bivariate case, we use $\rho = .3$ and the functions in Table 4.1 to generate $N = 20$ data sets, each of size $n = 500$ to study the estimation method proposed in Section 3.1. For each sample, the total number of iterations used in the MCMC sampling scheme is 5000 with a burn-in period of 1000.

For each data set, the correlation ρ is estimated as its posterior mean, approximated by

$$\hat{\rho}^{(i)} = \frac{1}{T}\sum_{t=1}^{T} \rho_t^{(i)},\ i = 1, \ldots,\ N, \tag{4.0.2}$$

where $\rho_t^{(i)}$ is the value of ρ generated at the tth iteration of the ith dataset, and T is the number of iterations after burn-in ($T = 4000$). Figure 4.1 (left) is a density histogram of $\hat{\rho}^{(i)},\ i = 1, \ldots,\ N$, based on random samples generated from the bivariate normal distribution, $N_2(\mathbf{0}, \Omega_2)$. The mean $(1/N)\sum_{i=1}^{N} \hat{\rho}^{(i)} = .299$ is close to the the theoretical value $\rho = .3$.

Let $r_1, r_2, \ldots, r_G$ ($G = 100$) denote an equally-spaced grid of G values of r. The density of $h(r)$ is estimated based on the ith data set at $r_g,\ g = 1,\ \cdots,\ G$, as follows,

$$\hat{h}^{(i)}(r_g) = \frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{k} w_{ij}^t \cdot f(r_g|\alpha_{ij}^t, \beta_{ij}^t),\ i = 1, \ldots,\ N, \tag{4.0.3}$$

where $w_{ij}^t,\ \alpha_{ij}^t,\ \beta_{ij}^t$ are the values of the jth weight, shape and rate parameters, respectively, of the gamma density function of the ith data set based on the tth iteration. Figure 4.1

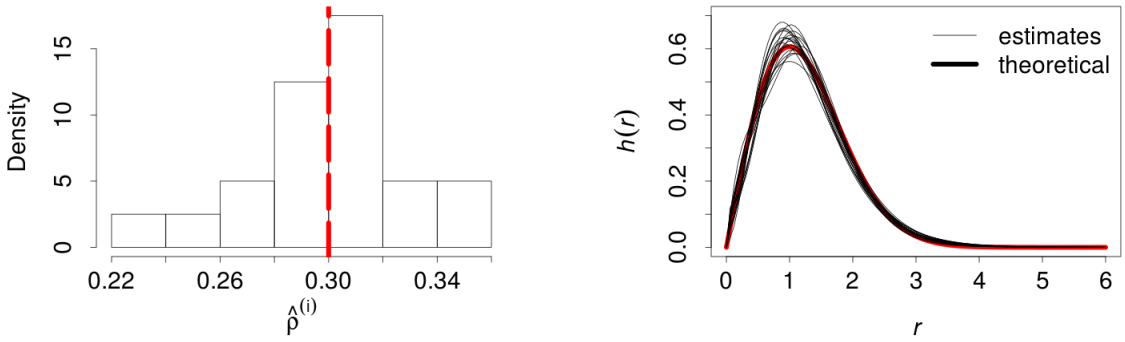

Figure 4.1: Density histogram of $\hat{\rho}^{(i)}$, and the theoretical pdf (4.0.1) $h(r)$ (thick line) along with the its estimates $\hat{h}^{(i)}(r),\ i = 1, \ldots, N$, based on random samples of size $n = 500$ from the bivariate normal distribution $N_2(\mathbf{0}, \Omega_2)$.

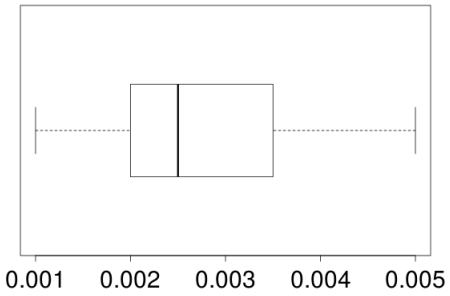

Figure 4.2: Box plot of the Euclidean distances $S^{(i)},\ i = 1, \ldots, N$, based on random samples of size $n = 500$ from the bivariate normal distribution $N_2(\mathbf{0}, \Omega_2)$.

(right) displays the theoretical pdf (4.0.1) $h(r)$ along with the its estimates based on the simulated data sets. The plot shows close agreement between the theoretical density and its estimates. Figure 4.2 shows a box plot of the Euclidean distances between $h(r)$ and its estimates computed for the ith data set as

$$S^{(i)} = \sqrt{\frac{1}{G}\sum_{g=1}^{G}\left(h(r_g) - \hat{h}^{(i)}(r_g)\right)^2},\ i = 1, \ldots,\ N.$$

Figures 4.3 and 4.4 show the plots of the bivariate t and logistic distributions, respectively. In the trivariate case, we use the correlation matrix

$$\Omega_3 = \begin{bmatrix} 1 & 0.5 & 0.6 \\ 0.5 & 1 & 0.7 \\ 0.6 & 0.7 & 1 \end{bmatrix}.$$

The simulation results are shown in Figures 4.5 to 4.7.

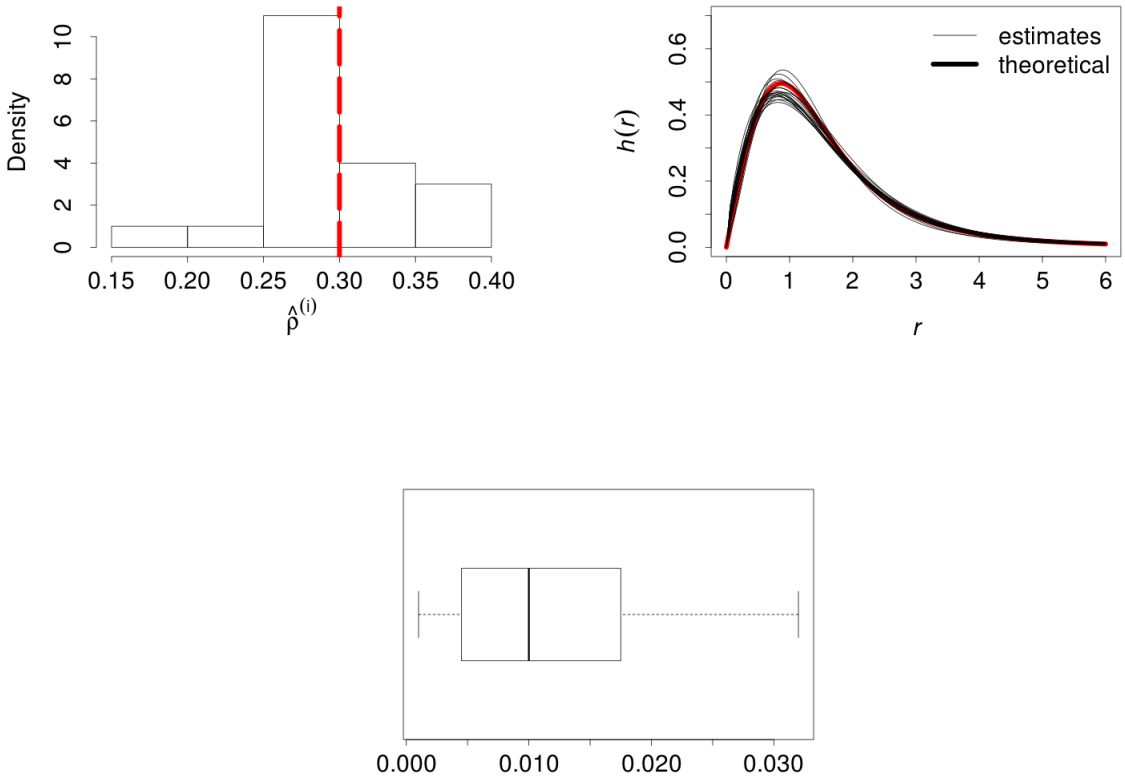

Figure 4.3: Simulation results for the bivariate t distribution with 3 degrees of freedom.

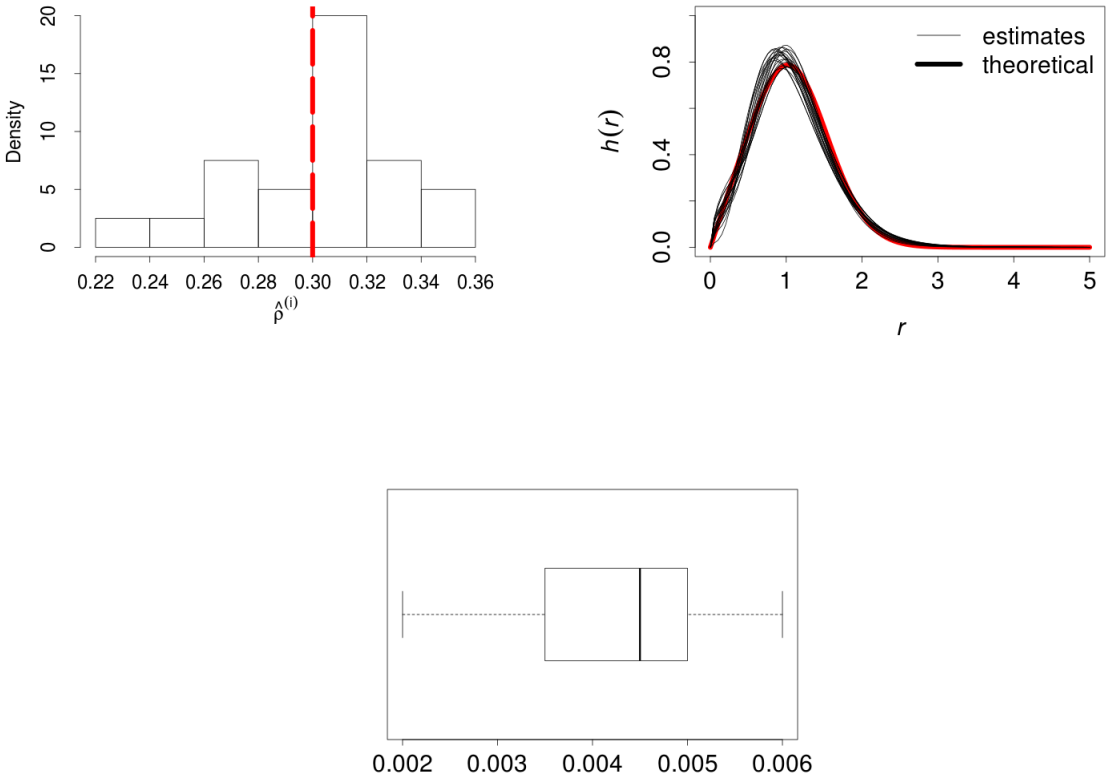

Figure 4.4: Simulation results for the bivariate logistic distribution.

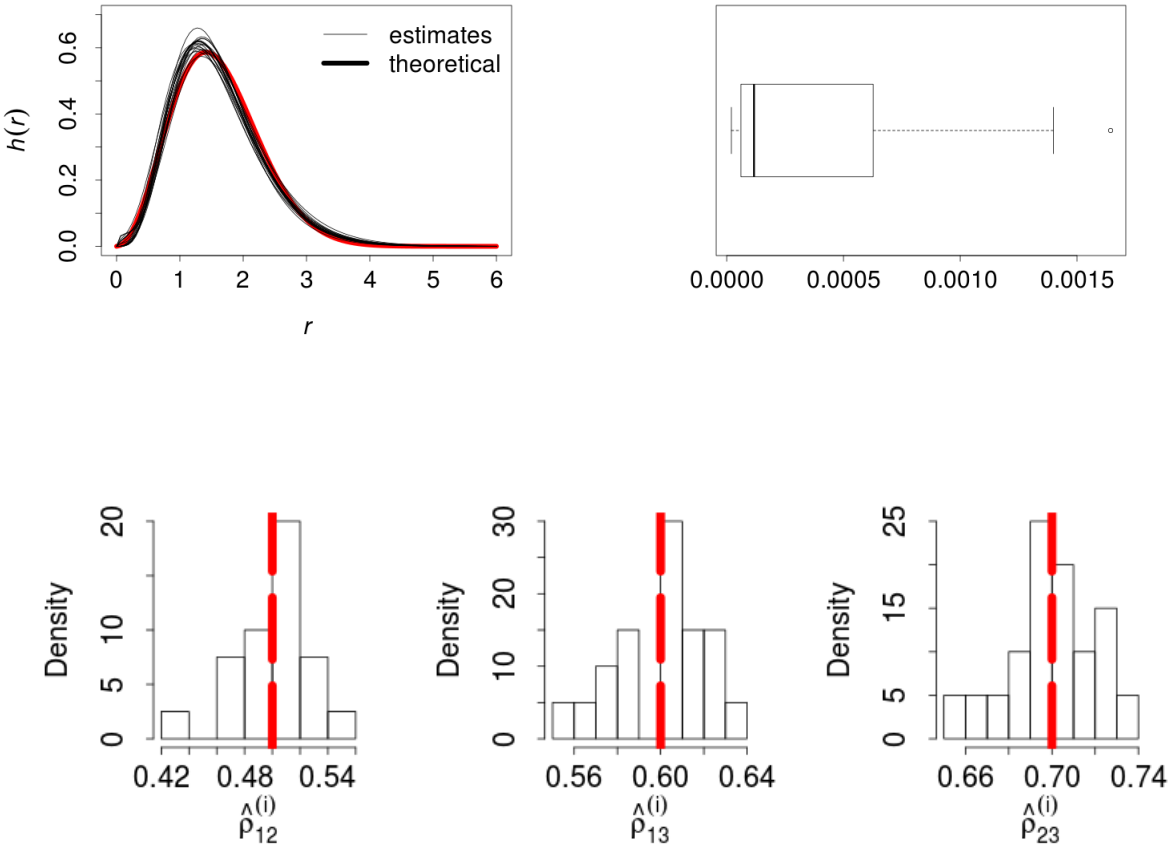

Figure 4.5: Simulation results for the trivariate normal distribution $N_3(\mathbf{0}, \Omega_3)$.

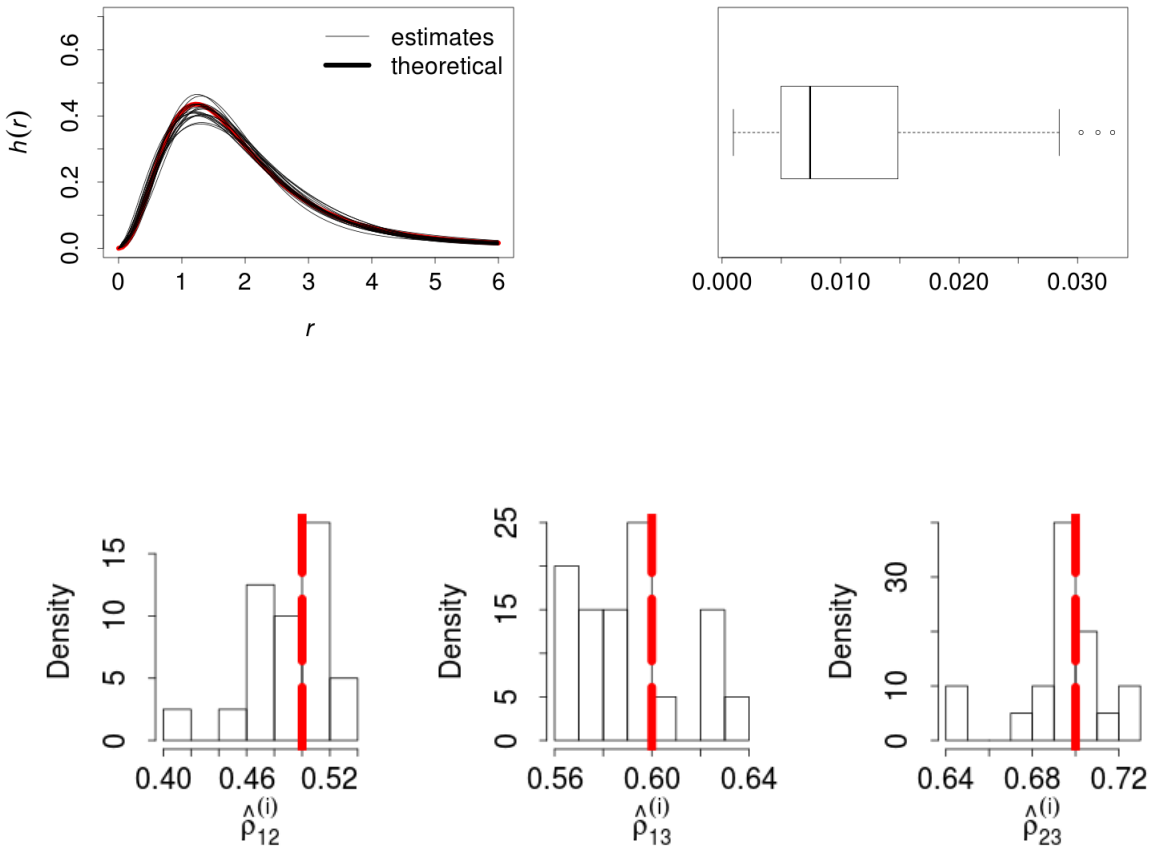

Figure 4.6: Simulation results for the trivariate t distribution with 3 degrees of freedom.

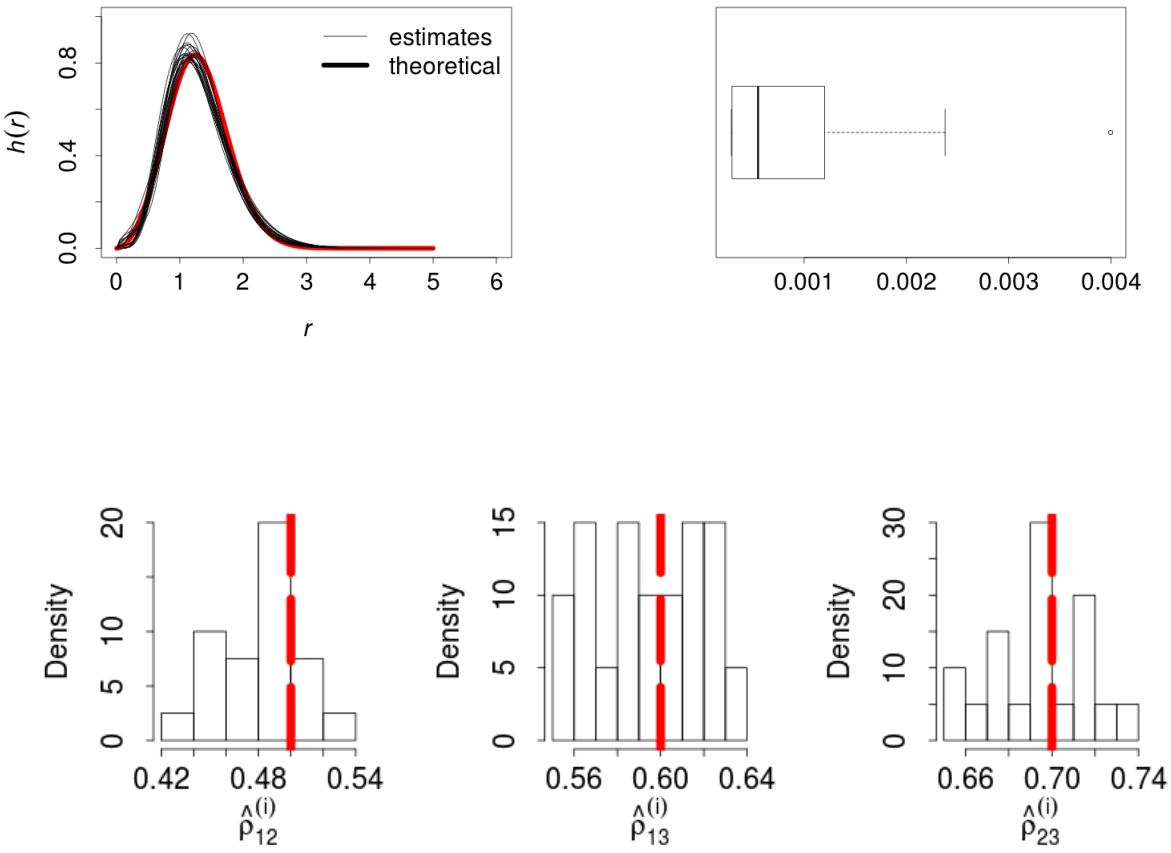

Figure 4.7: Simulation results for the trivariate logistic distribution.

Chapter 5

Results Based on Real Data

Table 5.1 shows the first few records out of 744 of three atmospheric features that are closely related to PM2.5 air pollution in Beijing. These measurements consist of hourly collected Pressure (hPa), Temperature (degree Celsius), and Humidity (%). The time interval is from January 1, 2011 to January 31, 2011.

The data set has first been scaled by

$$\frac{X - \bar{X}}{S_X},$$

where $\bar{X}$ and S_X are the sample mean and the standard deviation, respectively. The sample correlation matrix is

$$\begin{bmatrix} 1 & -0.3309 & -0.0169 \\ -0.3309 & 1 & -0.7098 \\ -0.0169 & -0.7098 & 1 \end{bmatrix}.$$

Figure 5.1 shows histograms of these three columns. We can see that the three plots are reasonably symmetric, indicating that they may have the same marginal distribution. Thus, we assume a three-dimensional elliptical distribution for the data and use the method in Chapter 3 to estimate the pdf, $h(r)$.

Figure 5.2 displays a histogram of

$$r_i = (\boldsymbol{X}_i' \hat{\Omega}^{-1} \boldsymbol{X}_i)^{1/2},$$

where $\boldsymbol{X}$ is the atmospheric data and $\hat{\Omega}$ is the correlation matrix estimated by Equation (4.0.2).

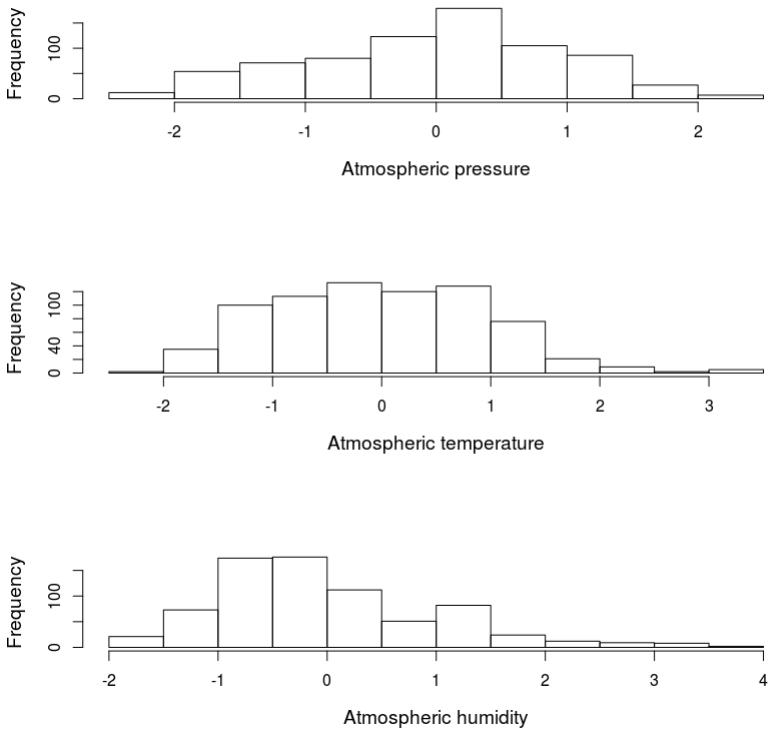

Figure 5.1: Histograms of atmospheric data

Table 5.1: Scaled Atmospheric Data in January 2011 frome Beijing (UC Irvine Machine Learning Repository)

	Pressure	Temperature	Humidity
1	0.0648	-0.1440	0.3991
2	0.0648	-0.4511	0.6775
3	0.0648	-0.4511	0.6775
4	-0.2653	-0.7582	1.0115
5	0.0648	-0.7582	0.6775
6	0.0648	-0.7582	0.6775
7	0.0648	-0.7582	1.0115
8	0.0648	-0.4511	0.6775
...	...	...	...

The estimated $h(r)$ and the theoretical $h(r)$ of the trivariate normal distribution are in close agreement. The estimated correlation matrix is

$$\hat{\Omega} = \begin{bmatrix} 1 & -0.3271 & -0.0074 \\ -0.3271 & 1 & -0.7152 \\ -0.0074 & -0.7152 & 1 \end{bmatrix}.$$

Table 5.2 shows part of another data set related to red wine quality. The measurements are fixed acidity, volatile acidity and PH in 500 samples of the red wine from the north of Portugal. We can see the possibility of identical marginal distributions in Figure 5.3. Figure 5.4 shows that the estimated $h(r)$ is close to $h(r)$ of a trivariate normal distribution. The estimated correlation matrix is

$$\hat{\Omega} = \begin{bmatrix} 1 & -0.3564 & -0.6789 \\ -0.3564 & 1 & 0.2564 \\ -0.6789 & 0.2564 & 1 \end{bmatrix},$$

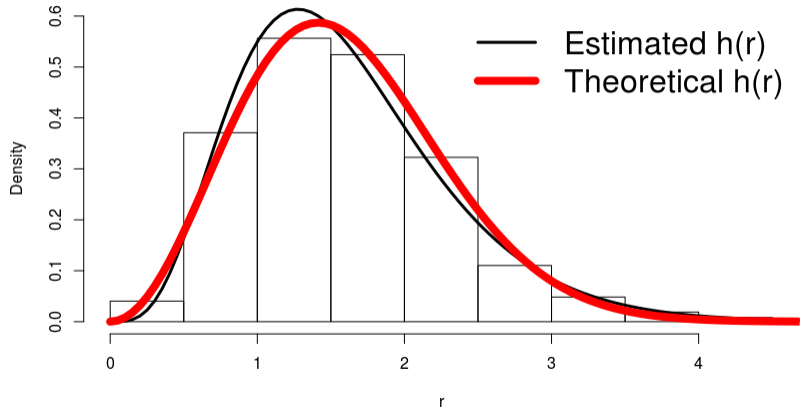

Figure 5.2: Estimated and theoretical $h(r)$ corresponding to a trivariate normal overlaid on the histogram of r

which is very close to the sample correlation matrix

$$\begin{bmatrix} 1 & -0.3483 & -0.6713 \\ -0.3483 & 1 & 0.2659 \\ -0.6713 & 0.2659 & 1 \end{bmatrix}.$$

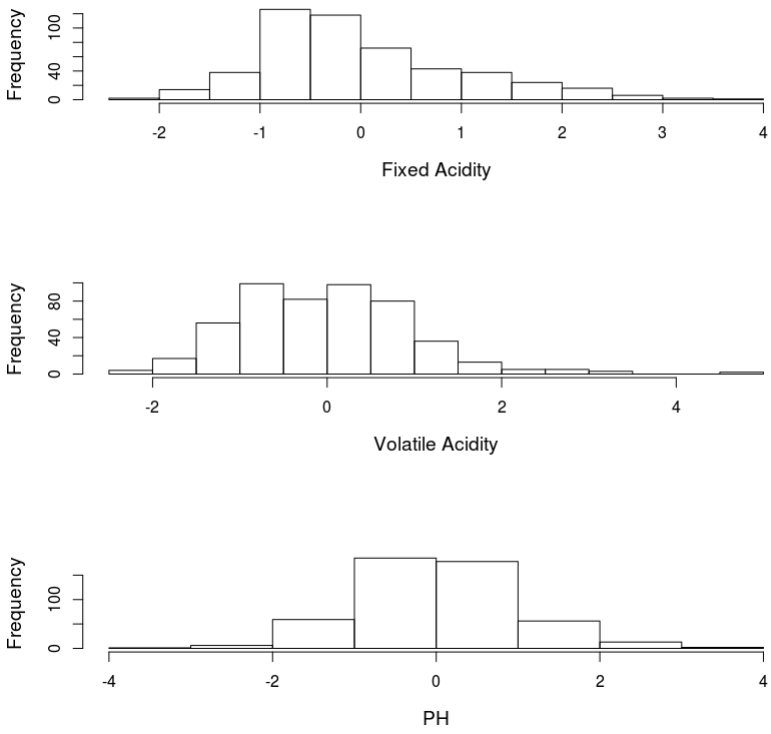

Figure 5.3: Histograms of red wine data

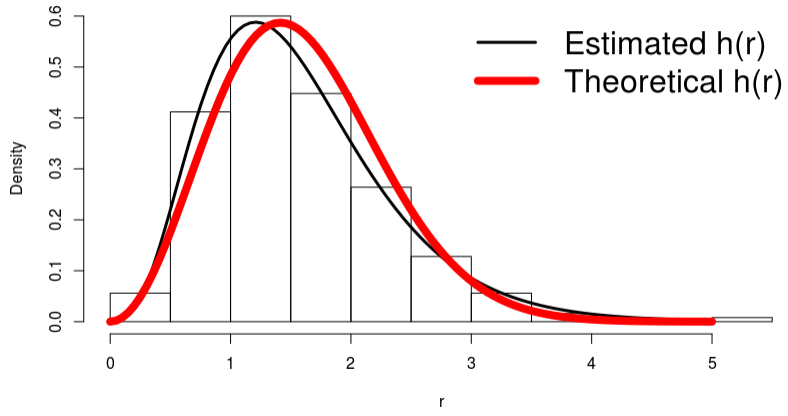

Figure 5.4: Estimated and theoretical $h(r)$ corresponding to a trivariate normal overlaid on the histogram of r

Table 5.2: Scaled Red Wine Data (UC Irvine Machine Learning Repository)

	Fixed acidity	Volatile acidity	PH
1	-0.6828	0.9459	1.3998
2	-0.4705	1.9676	-0.5986
3	-0.4705	1.2864	-0.2118
4	1.3342	-1.4382	-0.8565
5	-0.6828	0.9459	1.3998
6	-0.6828	0.7188	1.3998
7	-0.4174	0.3782	0.0460
8	-0.7359	0.6620	0.6262
...	...	...	...
500	0.0072	0.8891	1.2064

Chapter 6

Concluding Remarks

6.1 Significance of the Result

We are able to estimate the density function of an elliptical distribution without specifying its generator function. Given a multivariate data set, we can use an elliptical distribution to model the data and estimate the density function using the methods of Chapter 3. The estimated pdf will be very useful in further data analysis.

6.2 Future Work

Elliptical distributions may be used to model many multivariate data sets. However, their limitation is that the scaled components must have identical marginal distributions. If $\boldsymbol{X}^* = (x_1^*, \cdots, x_p^*)$ has an elliptical distribution $E_p(\mu, \Omega, g)$, then, the scaled components $x_k^*/\sqrt{\sigma_{kk}},\ k = 1, \cdots, p$, must have same marginal distributions, where σ_{kk} is the kth elements of the diagonal of Ω. The atmospheric data and the red wine data used in Chapter 5 satisfy this requirement approximately. This feature of elliptical distributions limits their use since many multivariate data sets do not have identical margins. In other words, given a data set, we cannot in general assume an elliptical distribution.

To address this problem, we will extend the estimation method to meta-elliptical copulas (see Fang et al. (2005)).

Given a vector $\boldsymbol{X} = (X_1, \ldots, X_p)$ of $p \geq 2$ continuous random variables, the copula approach expresses the joint distribution of $\boldsymbol{X}$

$$Pr(X_1 \leq x_1, \ldots, X_p \leq x_p) = C(F_1(x_1), F_2(x_2), \ldots, F_p(x_p))$$

in terms of its marginal distributions

$$F_k(x) = Pr(X_k \leq x), k \in \{1, \ldots, p\}$$

and a copula C, i.e., the cumulative distribution function of a vector $(U_1, \ldots, U_p)$ of dependent uniform random variables on the interval $(0, 1)$. The copula approach does not require identical marginal distributions so that it is more practical in the modeling of multivariate data.

References

Blackwell, D. and MacQueen, J. B. (1973), "Ferguson Distributions Via Polya Urn Schemes," *The Annals of Statistics*, 1, 353–355.

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), "Correction to "The meta-elliptical distributions with given marginals" (2002V82 p1-16)," *Journal of Multivariate Analysis*, 94, 222–223.

Ferguson, T. S. (1973), "A Baysian analysis of some nonparametric problems," *The Annals of Statistics*, 1, 209–230.

Genest, C., Favre, A.-C., Béliveau, J., and Jacques, C. (2007), "Metaelliptical copulas and their use in frequency analysis of multivariate hydrological data," *Water Rresources Research*, 43, 1–12.

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

Kalli, M., Griffin, J. E., and Walker, S. G. (2011), "Slice Sampling Mixture Models," *Statistics and Computing*, 21, 93–105.

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

Lo, A. Y. (1984), "On a class of Baysian nonparametric estimates: I. Density estimates," *The Annals of Statistics*, 12, 351–357.

Pourahmadi, M. and Wang, X. (2015), "Distribution of random correlation matrices: hyperspherical parameterization of the Cholesky factor," *Statistics and Probability Letters*, 106, 5–12.

Rapisarda, F., Brigo, D., and Mercurio, F. (2007), "Parameterizing correlations: a geometric interpretation," *IMA Journal of Management Mathematics*, 18, 55–73.

Sethuraman, J. (1994), "A constructive definition of Dirichlet priors," *Statistica Sinica*, 639–650.

Appendix A

R Code for Simulated Data

A.1 Data generation

```
############################data generation####################
#################################################################

library(MASS)
library(MCMCpack)
library(mvtnorm)

#Two dimension
#Generate bivariate logistic
rho = .3
omega = matrix(c(1, rho, rho, 1), nrow = 2)
n = 500
#Generate bivariate logistic
density.ker <- function(r, p) {
r ^ (p - 1) * exp(-r ^ 2) / (1 + exp(-r ^ 2)) ^ 2
}
M = 5
for (i in 1:20) {
#Rejection Sampling
R = vector(mode = "double")
```

```
for (j in 1:30000) {
y = rchisq(1, 4)
u = runif(1)
if (u < (density.ker(y, 2) / (10 * dchisq(y, 4)))) {
R = c(R, y)
}
}
R = R[1:n]
theta = runif(n, 0, 2 * pi)
U = matrix(c(cos(theta), sin(theta)), nrow = 2, byrow = TRUE)
X = R * t(t(chol(omega)) %*% U)
write.table(X,
row.names = FALSE,
file = paste("bvlogistic", toString(i),
".txt", sep = ""))
}


#Generate bivariate normal
for (i in 1:20) {
X = mvrnorm(n, c(0, 0), omega)
write.table(X,
row.names = FALSE,
file = paste("bvn", toString(i), ".txt",
sep = ""))
}

#Generate bivariate T
```

```
for (i in 1:20) {
X = rmvt(n, sigma = omega, df = 3)
write.table(X,
row.names = FALSE,
file = paste("bvt", toString(i), ".txt",
sep = ""))
}



#Three dimension
#Generate trivariate normal
for (i in 1:20) {
X = mvrnorm(n, c(0, 0, 0), omega)
write.table(X,
row.names = FALSE,
file = paste("trivn", toString(i), ".txt", sep = ""))
}
#Generate trivariate T
for (i in 1:20) {
X = rmvt(n, sigma = omega, df = 3)
write.table(X,
row.names = FALSE,
file = paste("trivt", toString(i), ".txt", sep = ""))
}


#Generate trivariate logistic
```

```
rho12 = .5
rho13 = .6
rho23 = .7
dimention = 3
omega = matrix(c(1, rho12, rho13, rho12, 1,
rho23, rho13, rho23, 1),
nrow = dimention)
n = 500
density.ker <- function(r, p) {
r ^ (p - 1) * exp(-r ^ 2) / (1 + exp(-r ^ 2)) ^ 2
}
M = 5
for (i in 1:20) {
#Rejection Sampling

R = vector(mode = "double")
for (j in 1:30000) {
y = rchisq(1, 4)
u = runif(1)
if (u < (density.ker(y, 3) / (10 * dchisq(y, 4)))) {
R = c(R, y)
}
}
R = R[1:n]

A = t(chol(omega))
phi = runif(n, 0, 2 * pi)
yu = runif(n, -1, 1)
```

```
U = matrix(c(sqrt(1 - yu ^ 2) * cos(phi),
sqrt(1 - yu ^ 2) * sin(phi), yu),
nrow = 3,
byrow = TRUE)
X = R * t(A %*% U)
write.table(
X,
row.names = FALSE,
file = paste("trivlogistic", toString(i), ".txt", sep = "")
)
}
```

A.2 MCMC on two dimensional simulated data

```
######################## MCMC for two-dimension ################
#######################################################################
for (num in 14:20) {
#Change the file name for data from other distributions
X = read.table(paste("bvlogistic", toString(num),
".txt", sep = ""),
header = TRUE)

X = data.matrix(X, rownames.force = NA)
n = nrow(X)
dimension = ncol(X)

#Priors
rho.est = 0
```

```
k = 30 # Indicating infinite number of components
v = rbeta(k - 1, 1, 1)
v = c(v, 1)
#compute P
p.est = vector(mode = "double", length = k)
sum = 0
for (i in 1:k) {
p.est[i] = (1 - sum) * v[i]
sum = sum + p.est[i]
}


rpareto <- function(n, c) {
if (c <= 0)
stop("c must be positive")
rp <- runif(n) ^ (-1 / c)
rp
}

#Prior of alpha
c = 3 # how to set it up
alpha.est = c(rep(1.1, k))

#Prior of lamda
alpha.null = 1
lamda.null = 1
```

```
lamda.est = c(rep(1, k))

#number of iteration
n.iteration = 2000
#container for recording result of iteration
p.record = matrix(NA, nrow = k , ncol = n.iteration)
alpha.record = matrix(NA, nrow = k, ncol = n.iteration)
lamda.record = matrix(NA, nrow = k, ncol = n.iteration)
e.record = matrix(NA, nrow = k, ncol = n.iteration)
e2.record = vector(mode = "double", length = n.iteration)
rho.record = vector(mode = "double", length = n.iteration)

# this is a matrix used in calculation of parameter
m = matrix(c(rep(1, (k - 1) ^ 2)), nrow = k - 1)
m[lower.tri(m) == FALSE] = 0

for (i in 1:n.iteration) {
#update R on each iteration
omega.est = matrix(c(1, rho.est, rho.est, 1), nrow = 2)
A = chol(omega.est)
R.sqr = diag(X %*% solve(omega.est) %*% t(X))
R.est = sqrt(R.sqr)

#f(x| mu_i, phi_i) k x n matrix
density.individual = apply(as.matrix(R.est),
1,
dgamma,
shape = alpha.est,
```

```
rate = lamda.est)
#whole mixture pdf f(x)=p1*f(x|mu1,phi1) + p2*f(x|mu2,phi2)
density.mixture = p.est %*% density.individual
#probability vector of Zi vector
prob = t(t(density.individual * as.vector(p.est)) /
as.vector(density.mixture))


zn = apply(prob, 2, rmultinom, n = 1, size = 1)

r = apply(zn, 1, sum)

v = rbeta(k - 1, 1 + r[1:k - 1], r[1:k - 1] %*% m + 1)
# m is a kinda of lower triangular matrix
v = c(v, 1)
sum = 0
for (j in 1:k) {
p.est[j] = (1 - sum) * v[j]
sum = sum + p.est[j]
}
p.record[, i] = p.est

################################## Conjugate

lamda.est = rgamma(k, alpha.null +
alpha.est * (apply(zn, 1, sum)),
rate = lamda.null + zn %*% R.est)
lamda.record[, i] = lamda.est
```

```
##############################
sigma = 1
alpha.est.test = rnorm(k , alpha.est, sigma)

A1 = alpha.est.test * apply(zn, 1, sum) * log(lamda.est) +
alpha.est.test * zn %*% log(R.est) -
apply(zn, 1, sum) * lgamma(alpha.est.test) +
(-3 - 1) * log(alpha.est.test)

B1 = alpha.est * apply(zn, 1, sum) * log(lamda.est) +
alpha.est * zn %*% log(R.est) -
apply(zn, 1, sum) * lgamma(alpha.est) +
(-3 - 1) * log(alpha.est)

logR = A1 - B1
r = exp(logR)
r[is.nan(r)] = 0
e = pmin(1, r)
u = runif(k, 0, 1)
alpha.est[u <= e & alpha.est.test > 1] =
alpha.est.test[u <= e & alpha.est.test > 1]

#acceptance rate
e.record[, i] = e

alpha.record[, i] = alpha.est
```

```
############## about rho metropolis

rho.est.prop = runif(1, rho.est - 0.05, rho.est + 0.05)

if (rho.est.prop > -1 && rho.est.prop < 1)
{
omega.est.prop = matrix(c(1, rho.est.prop, rho.est.prop, 1),
nrow = 2)
A.prop = chol(omega.est.prop)
R.sqr.prop = diag(X %*% solve(omega.est.prop) %*% t(X))
R.est.prop = sqrt(R.sqr.prop)

A2 = -n * log(abs(det(A.prop))) -
sum(log(R.est.prop)) +
sum(diag(t(zn) %*%
log(
apply(
as.matrix(R.est.prop),
1,
dgamma,
shape =
alpha.est,
rate = lamda.est
)
)))

B2 = -n * log(abs(det(A))) -
```

```
sum(log(R.est)) +
sum(diag(t(zn) %*%
log(
apply(
as.matrix(R.est),
1,
dgamma,
shape =
alpha.est,
rate = lamda.est
)
)))
#logR=A-B
r2 = exp(A2 - B2)
r2[is.nan(r2)] = 0
e2 = pmin(1, r2)
u2 = runif(1)
rho.est[u2 <= e2] = rho.est.prop[u2 <= e2]
#acceptance rate
e2.record[i] = e2
} else{
e2.record[i] = 0
}

rho.record[i] = rho.est
}
```

```
save(
rho.record,
e2.record,
alpha.record,
e.record,
lamda.record,
p.record,
R.est,
n.iteration,
#Change the file name for data from other distributions
file = paste("resultbvlogistic", toString(num),
".RData", sep = "")
)

rm(list = ls())
}
```

A.3 MCMC on three dimensional simulated data

```
########################MCMC for three-dimension ##############
for (num in 1:20) {
X = read.table(paste("trivlogistic", toString(num),
".txt", sep = ""), header = TRUE)

X = data.matrix(X, rownames.force = NA)
n = nrow(X)
dimension = ncol(X)
#########################################################################
```

```
#functions
angel2correlation <- function(a12, a13, a23) {
matrix(
c(
1,
cos(a12),
cos(a13),
cos(a12),
1,
cos(a12) * cos(a13) +
sin(a12) * sin(a13) * cos(a23),
cos(a13),
cos(a12) * cos(a13) +
sin(a12) * sin(a13) * cos(a23),
1
),
nrow = dimension
)
}

angel2chol <- function(a12, a13, a23) {
matrix(c(
1,
0,
0,
cos(a12),
sin(a12),
0,
```

```
cos(a13),
sin(a13) * cos(a23),
sin(a23) * sin(a13)
), nrow = dimension)
}

loglikelihood <- function(a12, a13, a23) {
cholesky = angel2chol(a12, a13, a23)
correlation = angel2correlation(a12, a13, a23)
R = sqrt(diag(X %*% chol2inv(cholesky) %*% t(X)))

#???????
(-n * log(abs(det(cholesky))) +
(1 - dimension) * sum(log(R)) +
sum(diag(t(zn) %*%
log(
apply(as.matrix(R),
1,
dgamma,
shape =  alpha.est, rate = lamda.est)
)))
+ log(abs((
sin(a12) * sin(a13)
) ^ 2 * sin(a23))))

}
```

```
#Priors
theta12 = runif(1, 0, pi)
theta13 = runif(1, 0, pi)
theta23 = runif(1, 0, pi)

k = 30 # Indicating infinite number of components
eta1 = 1
eta2 = 1
h.gamma = rgamma(1, eta1, eta1)
v = rbeta(k - 1, 1, h.gamma)
v = c(v, 1)
#compute P
p.est = vector(mode = "double", length = k)


sum = 0
for (i in 1:k) {
p.est[i] = (1 - sum) * v[i]
sum = sum + p.est[i]
}


rpareto <- function(n, c) {
if (c <= 0)
stop("c must be positive")
rp <- runif(n) ^ (-1 / c)
rp
}
```

```
#Prior of alpha
c = 3 # how to set it up
alpha.est = rpareto(k, c)


#Prior of lamda, which is lamda
alpha.null = 1  #too small?
lamda.null = 1

lamda.est = rgamma(k, alpha.null, rate = lamda.null)

#number of iteration
n.iteration = 2000


#container for recording result of iteration
p.record = matrix(NA, nrow = k , ncol = n.iteration)
alpha.record = matrix(NA, nrow = k, ncol = n.iteration)
lamda.record = matrix(NA, nrow = k, ncol = n.iteration)
e.record = matrix(NA, nrow = k, ncol = n.iteration)
e.theta12.record = vector(mode = "double", length = n.iteration)
e.theta13.record = vector(mode = "double", length = n.iteration)
e.theta23.record = vector(mode = "double", length = n.iteration)
theta12.record = vector(mode = "double", length = n.iteration)
theta13.record = vector(mode = "double", length = n.iteration)
theta23.record = vector(mode = "double", length = n.iteration)
```

```
# this is a matrix used in calculation of parameter
m = matrix(c(rep(1, (k - 1) ^ 2)), nrow = k - 1)
m[lower.tri(m) == FALSE] = 0

i = 1
errornum = 0
for (i in 1:n.iteration) {
#update omega and R
omega.chol.est = angel2chol(theta12, theta13, theta23)
omega.est = angel2correlation(theta12, theta13, theta23)
R.sqr = diag(X %*% chol2inv(omega.chol.est) %*% t(X))
R.est = sqrt(R.sqr)
#f(x| mu_i, phi_i) k x n matrix
density.individual = apply(as.matrix(R.est),
1,
dgamma,
shape = alpha.est,
rate = lamda.est)
#whole mixture pdf f(x)=p1*f(x|mu1,phi1) + p2*f(x|mu2,phi2)
density.mixture = p.est %*% density.individual
#probability vector of Zi vector
prob = t(t(density.individual * as.vector(p.est)) /
as.vector(density.mixture))

zn = apply(prob, 2, rmultinom, n = 1, size = 1)

#apply(zn, 1, sum)
```

```
h.gamma = rgamma(1, k + eta1 - 1, eta2 - sum(log(1 - v[1:k - 1])))

r = apply(zn, 1, sum)
v = rbeta(k - 1, 1 + r[1:k - 1], r[1:k - 1] %*% m + h.gamma)
# m is a kinda of lower triangular matrix
v = c(v, 1)
sum = 0
for (j in 1:k) {
p.est[j] = (1 - sum) * v[j]
sum = sum + p.est[j]
}


p.record[, i] = p.est

################################## Conjugate

lamda.est = rgamma(k, alpha.null +
alpha.est * (apply(zn, 1, sum)),
rate = lamda.null + zn %*% R.est)
lamda.record[, i] = lamda.est

################################## alpha
sigma = 1
alpha.est.test = rnorm(k , alpha.est, sigma)

A1 = alpha.est.test * apply(zn, 1, sum) * log(lamda.est) +
alpha.est.test * zn %*% log(R.est) -
```

```
apply(zn, 1, sum) * lgamma(alpha.est.test) +
(-3 - 1) * log(alpha.est.test)

B1 = alpha.est * apply(zn, 1, sum) * log(lamda.est) +
alpha.est * zn %*% log(R.est) -
apply(zn, 1, sum) * lgamma(alpha.est) +
(-3 - 1) * log(alpha.est)

logR = A1 - B1
r = exp(logR)
r[is.nan(r)] = 0
e = pmin(1, r)
u = runif(k, 0, 1)
alpha.est[u <= e & alpha.est.test > 1] =
alpha.est.test[u <= e & alpha.est.test > 1]
#acceptance rate
e.record[, i] = e
alpha.record[, i] = alpha.est

############## about rho metropolis, based on theta's
################ for theta12
theta12.prop = runif(1, theta12 - 0.1, theta12 + 0.1)

if (theta12.prop < 0 || theta12.prop > pi) {
e.theta12 = 0
errornum = (errornum + 1)
} else{
r.theta12 = exp(
```

```
loglikelihood(theta12.prop, theta13, theta23)
- loglikelihood(theta12, theta13, theta23)
)

if (is.nan(r.theta12)) {
r.theta12 = 0
errornum = (errornum + 1)
}
e.theta12 = min(1, r.theta12)
}

u = runif(1)
if (u <= e.theta12) {
theta12 = theta12.prop
}
#acceptance rate
e.theta12.record[i] = e.theta12
theta12.record[i] = theta12

##########################theta13
theta13.prop = runif(1, theta13 - 0.1, theta13 + 0.1)

if (theta13.prop < 0 || theta13.prop > pi) {
e.theta13 = 0
errornum = (errornum + 1)
} else{
r.theta13 = exp(
loglikelihood(theta12, theta13.prop, theta23)
```

```
- loglikelihood(theta12, theta13, theta23)
)

if (is.nan(r.theta13)) {
r.theta13 = 0
errornum = (errornum + 1)
}
e.theta13 = min(1, r.theta13)
}

u = runif(1)
if (u <= e.theta13) {
theta13 = theta13.prop
}
#acceptance rate
e.theta13.record[i] = e.theta13
theta13.record[i] = theta13



############################## theta23
theta23.prop = runif(1, theta23 - 0.1, theta23 + 0.1)

if (theta23.prop < 0 || theta23.prop > pi) {
e.theta23 = 0
errornum = (errornum + 1)
} else{
r.theta23 = exp(
loglikelihood(theta12, theta13, theta23.prop)
```

```
- loglikelihood(theta12, theta13, theta23)
)

if (is.nan(r.theta23)) {
r.theta23 = 0
errornum = (errornum + 1)
}
e.theta23 = min(1, r.theta23)
}

u = runif(1)
if (u <= e.theta23) {
theta23 = theta23.prop
}
#acceptance rate
e.theta23.record[i] = e.theta23
theta23.record[i] = theta23

}

save(
theta12.record,
theta13.record,
theta23.record,
e.theta12.record,
e.theta13.record,
e.theta23.record,
alpha.record,
```

```
e.record,
lamda.record,
p.record,
R.est,
n.iteration,
file = paste("resulttrivlogistic",
toString(num), ".RData", sep = "")
)

rm(list = ls())
}
```

A.4 Plot generation

```
############## Plot generation #################
rho=vector(mode="double", length = 20)
est.density=matrix(NA, nrow = 100, ncol = 20)
#Plot theoretical pdf and estimated one on the same plot
y=seq(0,6, length.out = 100)
#for mvnormal
theoretical.density2<-function(x){
dchisq(x^2, 2)*2*x
}
theoretical=sapply(y, theoretical.density2)
for(i in 1:20){
load(paste("resultbvn", toString(i), ".RData", sep=""))
rho[i]=mean(rho.record[1000:n.iteration])
#Estimated pdf from mcmc for R
```

```
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration])
*p.record[ ,1000:n.iteration]
test=apply(test, 2, sum)
mean(test)

}
est.density[,i]=sapply(y, est.mixture)
}
#Euclidean distance
ds=vector(mode = "double", length = 20)
for(i in 1:20){
ds[i]=sqrt((sum(theoretical-est.density[, i])^2)/100)
}
ds=round(ds, digits = 3)

png("~/Dropbox/panfeng/images/BivariateNorm/BvNRho.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
hist(rho, freq = FALSE, main = "",
xlab = expression(hat(rho)^(i))
)
abline(v=0.3, lty=5,col="red", lwd=8)
dev.off()
```

```
png("~/Dropbox/panfeng/images/BivariateNorm/BvNDensity.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
plot(y, theoretical, type = "l", lty=1, lwd=8, col="red",
xlab = expression(italic(r)), ylab = expression(italic(h(r))),
ylim = c(0,0.7), xlim = c(0,6))
par(new=TRUE)
for(i in 1:20){
lines(y, est.density[,i])
}

legend('topright', c('estimates', 'theoretical') ,
lty=c(1,1), lwd = c(1, 8), bty='n', cex=2.5)
dev.off()


png("~/Dropbox/panfeng/images/BivariateNorm/BvNEucBox.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
boxplot(ds, horizontal = TRUE)
dev.off()
```

```
#########Bivariate T#####################################
#Plot histogram of rhos from the 20 simulations
rho=vector(mode="double", length = 20)
est.density=matrix(NA, nrow = 100, ncol = 20)
#Plot theoretical pdf and estimated one on the same plot
y=seq(0,6, length.out = 100)
theoretical.density2<-function(x){

df(x^2/2, 2, 3) *x
}
theoretical=sapply(y, theoretical.density2)
for(i in 1:20){
load(paste("resultbvt", toString(i), ".RData", sep=""))
rho[i]=mean(rho.record[1000:n.iteration])

#Estimated pdf from mcmc for R
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration])
*p.record[ ,1000:n.iteration]
test=apply(test, 2, sum)
mean(test)
```

```
}
est.density[,i]=sapply(y, est.mixture)
}
#Euclidean distance
ds=vector(mode = "double", length = 20)
for(i in 1:20){
ds[i]=sqrt((sum(theoretical-est.density[, i])^2)/100)
}
ds=round(ds, digits = 3)

png("~/Dropbox/panfeng/images/BivariateT/BvTRho.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
hist(rho, freq = FALSE, main = "",
xlab = expression(hat(rho)^(i))
)
abline(v=0.3, lty=5,col="red", lwd=8)
dev.off()

png("~/Dropbox/panfeng/images/BivariateT/BvTDensity.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
```

```
plot(y, theoretical, type = "l", lty=1, lwd=8, col="red",
xlab = expression(italic(r)), ylab = expression(italic(h(r))),
ylim = c(0,0.7), xlim = c(0,6))
par(new=TRUE)
for(i in 1:20){
lines(y, est.density[,i])
}

legend('topright', c('estimates', 'theoretical') ,
lty=c(1,1), lwd = c(1, 8), bty='n', cex=2.5)
dev.off()



png("~/Dropbox/panfeng/images/BivariateT/BvTEucBox.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
boxplot(ds,horizontal = TRUE)
dev.off()

#########################Bivariate Logistic#################
#Plot histogram of rhos from the 20 simulations
rho=vector(mode="double", length = 20)
est.density=matrix(NA, nrow = 100, ncol = 20)
```

```
#Plot theoretical pdf and estimated one on the same plot
y=seq(0,5, length.out = 100)
theoretical.density2<-function(r){
p=2
2*pi^(p/2)/gamma(p/2)*r^(p-1)*exp(-r^2)/(1+exp(-r^2))^2/1.57
}

theoretical=sapply(y, theoretical.density2)

for(i in 1:20){
load(paste("resultbvlogistic", toString(i), ".RData", sep=""))
rho[i]=mean(rho.record[1000:n.iteration])

#Estimated pdf from mcmc for R
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration]) *p.record[ ,1000:n.iteration]
test=apply(test, 2, sum)
mean(test)

}

est.density[,i]=sapply(y, est.mixture)

}
#Euclidean distance
ds=vector(mode = "double", length = 20)
for(i in 1:20){
```

```
ds[i]=sqrt((sum(theoretical-est.density[, i])^2)/100)
}
ds=round(ds, digits = 3)

png("~/Dropbox/panfeng/images/BivariateLogistic/
BvLogisticRho.png",
width=800, height=600)
par(mgp=c(5,2,0),
mar=c(rep(10,4)),
cex.lab=2,
cex.axis=2)
hist(rho, freq = FALSE, main = "",
xlab = expression(hat(rho)^(i))
#ylab = expression(italic(h(r))),
)
abline(v=0.3, lty=5,col="red", lwd=8)
dev.off()


#Plot of estimated density of R
png("~/Dropbox/panfeng/images/BivariateLogistic/
BvLogisticDensity.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
```

```
plot(y, theoretical, type = "l", lty=1, lwd=8, col="red",
xlab = expression(italic(r)), ylab = expression(italic(h(r))),
ylim = c(0,1), xlim = c(0,5))
par(new=TRUE)
lines(y, est.density[,1])
for(i in 1:20){
lines(y, est.density[,i])
}
legend('topright', c('estimates', 'theoretical') ,
lty=c(1,1), lwd = c(1, 8), bty='n', cex=2.5)
dev.off()

png("~/Dropbox/panfeng/images/BivariateLogistic/
BvLogisticEucBox.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
boxplot(ds, horizontal = TRUE)
dev.off()


#########################################################

#Three dimension
#Plot histogram of rhos from the 20 simulations
rho12=vector(mode="double", length = 20)
```

```
rho13=vector(mode="double", length = 20)
rho23=vector(mode="double", length = 20)
est.density=matrix(NA, nrow = 100, ncol = 20)

#Plot theoretical pdf and estimated one on the same plot
y=seq(0,6, length.out = 100)
#for mvnormal
theoretical.density2<-function(x){
dchisq(x^2, 3)*2*x
}
theoretical=sapply(y, theoretical.density2)
for(i in 1:20){
load(paste("resulttrivn", toString(i), ".RData", sep=""))
rho12[i]=mean(cos(theta12.record[1000:n.iteration]))
rho13[i]=mean(cos(theta13.record[1000:n.iteration]))
rho23[i]=mean(cos(theta12.record[1000:n.iteration])*
cos(theta13.record[1000:n.iteration])+
sin(theta12.record[1000:n.iteration])*
sin(theta13.record[1000:n.iteration])*
cos(theta23.record[1000:n.iteration]))

#Estimated pdf from mcmc for R
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration])
*p.record[ ,1000:n.iteration]
test=apply(test, 2, sum)
mean(test)
```

```
}

est.density[,i]=sapply(y, est.mixture)

}
#Euclidean distance
ds=vector(mode = "double", length = 20)
for(i in 1:20){
ds[i]=sqrt((sum(theoretical-est.density[, i])^2)/100)
}
ds=round(ds, digits = 6)



png("~/Dropbox/panfeng/images/TrivariateNorm/TrivNRhos.png",
width=800, height=350)
par(mgp=c(6,2,0),
mar=c(10,10,10,3),
cex.lab=2.5,
cex.axis=2.5)
#Estimated rhos
par(mfrow=c(1,3))
hist(rho12, freq = FALSE, main = "",
xlab = expression(hat(rho)[12]^(i)))
abline(v=0.5, lty=5,col="red", lwd=8)
hist(rho13, freq = FALSE, main = "",
xlab = expression(hat(rho)[13]^(i)))
```

```
abline(v=0.6, lty=5,col="red", lwd=8)
hist(rho23, freq = FALSE, main = "",
xlab = expression(hat(rho)[23]^(i)))
abline(v=0.7, lty=5,col="red", lwd=8)
par(mfrow=c(1,1))
dev.off()

#Plot of estimated density of R
png("~/Dropbox/panfeng/images/TrivariateNorm/TrivNDensity.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
plot(y, theoretical, type = "l", lty=1, lwd=8,col="red",
xlab = expression(italic(r)), ylab = expression(italic(h(r))),
ylim = c(0,0.7), xlim = c(0,6))
par(new=TRUE)
for(i in 1:20){
lines(y, est.density[,i])
}
legend('topright', c('estimates', 'theoretical') ,
lty=c(1,1), lwd = c(1, 8), bty='n', cex=2.5)
dev.off()

png("~/Dropbox/panfeng/images/TrivariateNorm/TrivNEucBox.png",
width=800, height=600)
par(mgp=c(6,2,0),
```

```
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
boxplot(ds, horizontal = TRUE)
dev.off()

#####################################

########Trivariate T#################################
#for mvt

rho12=vector(mode="double", length = 20)
rho13=vector(mode="double", length = 20)
rho23=vector(mode="double", length = 20)
est.density=matrix(NA, nrow = 100, ncol = 20)

#Plot theoretical pdf and estimated one on the same plot
y=seq(0,6, length.out = 100)
theoretical.density2<-function(x){

df(x^2/3, 3, 3) *x*(2/3)
}
theoretical=sapply(y, theoretical.density2)
for(i in 1:20){
```

```
load(paste("resulttrivt", toString(i), ".RData", sep=""))
rho12[i]=mean(cos(theta12.record[1000:n.iteration]))
rho13[i]=mean(cos(theta13.record[1000:n.iteration]))
rho23[i]=mean(cos(theta12.record[1000:n.iteration])*
cos(theta13.record[1000:n.iteration])+
sin(theta12.record[1000:n.iteration])*
sin(theta13.record[1000:n.iteration])*
cos(theta23.record[1000:n.iteration]))

#Estimated pdf from mcmc for R
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration])
*p.record[ ,1000:n.iteration]
test=apply(test , 2, sum)
mean(test)

}

est.density[,i]=sapply(y, est.mixture)

}

#Euclidean distance
ds=vector(mode = "double", length = 20)
for(i in 1:20){
ds[i]=sqrt((sum(theoretical-est.density[, i])^2)/100)
}
```

```
ds=round(ds, digits = 6)



png("~/Dropbox/panfeng/images/TrivariateT/TrivTRhos.png",
width=800, height=350)
par(mgp=c(6,2,0),
mar=c(10,10,10,3),
cex.lab=2.5,
cex.axis=2.5)
#Estimated rhos
par(mfrow=c(1,3))
hist(rho12, freq = FALSE, main = "",
xlab = expression(hat(rho)[12]^(i)))
abline(v=0.5, lty=5,col="red", lwd=8)
hist(rho13, freq = FALSE, main = "",
xlab = expression(hat(rho)[13]^(i)))
abline(v=0.6, lty=5,col="red", lwd=8)
hist(rho23, freq = FALSE, main = "",
xlab = expression(hat(rho)[23]^(i)))
abline(v=0.7, lty=5,col="red", lwd=8)
par(mfrow=c(1,1))
dev.off()

#Plot of estimated density of R
png("~/Dropbox/panfeng/images/TrivariateT/TrivTDensity.png",
width=800, height=600)
par(mgp=c(6,2,0),
```

```
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
plot(y, theoretical, type = "l", lty=1, lwd=8,col="red",
xlab = expression(italic(r)), ylab = expression(italic(h(r))),
ylim = c(0,0.7), xlim = c(0,6))
par(new=TRUE)
for(i in 1:20){
lines(y, est.density[,i])
}
legend('topright', c('estimates', 'theoretical') ,
lty=c(1,1), lwd = c(1, 8), bty='n', cex=2.5)
dev.off()

png("~/Dropbox/panfeng/images/TrivariateT/TrivTEucBox.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
boxplot(ds,horizontal = TRUE)
dev.off()



##################################################
#for trivariate logistic
rho12=vector(mode="double", length = 20)
rho13=vector(mode="double", length = 20)
```

```
rho23=vector(mode="double", length = 20)
est.density=matrix(NA, nrow = 100, ncol = 20)

#Plot theoretical pdf and estimated one on the same plot
y=seq(0,5, length.out = 100)
theoretical.density2<-function(r){
p=3
r^(p-1)*exp(-r^2)/(1+exp(-r^2))^2/0.268
}


theoretical=sapply(y, theoretical.density2)


for(i in 1:20){
load(paste("resulttrivlogistic", toString(i), ".RData", sep=""))
rho12[i]=mean(cos(theta12.record[1000:n.iteration]))
rho13[i]=mean(cos(theta13.record[1000:n.iteration]))
rho23[i]=mean(cos(theta12.record[1000:n.iteration])*
cos(theta13.record[1000:n.iteration])+
sin(theta12.record[1000:n.iteration])*
sin(theta13.record[1000:n.iteration])*
cos(theta23.record[1000:n.iteration]))

#Estimated pdf from mcmc for R
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration])
```

```
*p.record[ ,1000:n.iteration]
test=apply(test, 2, sum)
mean(test)

}

est.density[,i]=sapply(y, est.mixture)

}
#Euclidean distance
ds=vector(mode = "double", length = 20)
for(i in 1:20){
ds[i]=sqrt((sum(theoretical-est.density[, i])^2)/100)
}
ds=round(ds, digits = 6)



png("~/Dropbox/panfeng/images/TrivariateLogistic/
TrivLogisticRhos.png",
width=800, height=350)
par(mgp=c(6,2,0),
mar=c(10,10,10,3),
cex.lab=2.5,
cex.axis=2.5)
#Estimated rhos
par(mfrow=c(1,3))
hist(rho12, freq = FALSE, main = "",
```

```
xlab = expression(hat(rho)[12]^(i)))
abline(v=0.5, lty=5,col="red", lwd=8)
hist(rho13, freq = FALSE, main = "",
xlab = expression(hat(rho)[13]^(i)))
abline(v=0.6, lty=5,col="red", lwd=8)
hist(rho23, freq = FALSE, main = "",
xlab = expression(hat(rho)[23]^(i)))
abline(v=0.7, lty=5,col="red", lwd=8)
par(mfrow=c(1,1))
dev.off()

#Plot of estimated density of R
png("~/Dropbox/panfeng/images/TrivariateLogistic/
TrivLogisticDensity.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
plot(y, theoretical, type = "l", lty=1, lwd=8,col="red",
xlab = expression(italic(r)), ylab = expression(italic(h(r))),
ylim = c(0,1), xlim = c(0,6))
par(new=TRUE)
for(i in 1:20){
lines(y, est.density[,i])
}
legend('topright', c('estimates', 'theoretical') ,
lty=c(1,1), lwd = c(1, 8), bty='n', cex=2.5)
```

```
dev.off()

png("~/Dropbox/panfeng/images/TrivariateLogistic/
TrivLogisticEucBox.png",
width=800, height=600)
par(mgp=c(6,2,0),
mar=c(rep(10,4)),
cex.lab=2.5,
cex.axis=2.5)
boxplot(ds, horizontal = TRUE)
dev.off()
```

Appendix B

R Code for real Data

B.1 Data subset

```
data=read.csv("BeijingPM20100101_20151231.csv")

#subset of atmospheric data
january2011=data.frame("press"=data$PRES[data$month==1
& data$year==2011],
"temp"=data$TEMP[data$month==1 & data$year==2011],
"humi"=data$HUMI[data$month==1 & data$year==2011])

cov(january2011)
apply(january2011, 2, mean)
#scale dataset
january2011=data.frame(round(scale(january2011),4))
write.table(january2011, "january2011.dat")



#subset wine data
data=read.csv("winequality-red.csv")
newdata=data.frame("fixed_acidity"=data$fixed_acidity,
"volatile_acidity"=data$volatile_acidity,
"PH"=data$pH)
```

```
write.table(round(scale(newdata[1:500,]),4), "redwine.dat")


########### Marginal histograms ##########
par(mfrow=c(3,1))
par(cex.lab=1.5)
par(cex.axis=1,mar=c(5,5,4,2))

#marginal histogram of data
dat=read.table("january2011.dat")
hist(dat$press,xlab = "Atmospheric pressure", main = "")
hist(dat$temp,xlab = "Atmospheric temperature", main = "")
hist(dat$humi,xlab = "Atmospheric humidity", main = "")



data=read.table("redwine.dat")
hist(data$fixed_acidity, xlab = "Fixed Acidity", main = "")
hist(data$volatile_acidity,xlab = "Volatile Acidity", main = "")
hist(data$PH,xlab = "PH", main = "")
```

B.2 MCMC

```
library(MASS)
library(MCMCpack)
library(mvtnorm)
num=1
##############################data reading################
#change file name when read in red wine data
```

```
X=read.table("january2011.dat", header = TRUE)

X=data.matrix(X, rownames.force = NA)
n=nrow(X)
dimension=ncol(X)
#############################################################
#functions
angel2correlation<-function(a12, a13, a23){
matrix(
c(1, cos(a12), cos(a13),
cos(a12), 1, cos(a12)*cos(a13)+
sin(a12)*sin(a13)*cos(a23),
cos(a13), cos(a12)*cos(a13)+
sin(a12)*sin(a13)*cos(a23), 1
),nrow = dimension )
}

angel2chol<-function(a12, a13, a23){
matrix(
c(1,0, 0,
cos(a12), sin(a12), 0,
cos(a13), sin(a13)*cos(a23), sin(a23)*sin(a13)
),nrow = dimension )
}

loglikelihood<-function(a12, a13, a23){
cholesky=angel2chol(a12, a13, a23)
correlation=angel2correlation(a12, a13, a23)
```

```
R=sqrt(diag(X%*% chol2inv(cholesky) %*% t(X)))

#???????
(-n*log(abs(det(cholesky)))+
(1-dimension)*sum(log(R))+
sum(diag(t(zn)%*%
log(apply(
as.matrix(R),
1,
dgamma,
shape =  alpha.est,
rate = lamda.est))))
+log(abs((sin(a12)*sin(a13))^2*sin(a23))))

}


#Priors
theta12=runif(1, 0, pi)
theta13=runif(1, 0, pi)
theta23=runif(1, 0, pi)

k=30 # Indicating infinite number of components
eta1=1
eta2=1
h.gamma=rgamma(1,eta1,eta1)
v=rbeta(k-1, 1, h.gamma)
v=c(v,1)
```

```
#compute P
p.est=vector(mode="double", length = k)



sum=0
for (i in 1:k) {
p.est[i]=(1-sum)*v[i]
sum=sum+p.est[i]
}



rpareto <- function(n,c){
if(c<=0) stop("c must be positive")
rp <- runif(n)^(-1/c)
rp}

#Prior of alpha
c=3 #
alpha.est=rpareto(k, c)



#Prior of lamda
alpha.null=1
lamda.null=1

lamda.est=rgamma(k, alpha.null, rate=lamda.null)

#number of iteration
```

```
n.iteration= 2000


#container for recording result of iteration
p.record=matrix(NA, nrow =k , ncol =n.iteration )
alpha.record=matrix(NA, nrow = k, ncol = n.iteration)
lamda.record=matrix(NA, nrow = k, ncol = n.iteration)
e.record=matrix(NA, nrow = k, ncol = n.iteration)
e.theta12.record=vector(mode="double", length = n.iteration)
e.theta13.record=vector(mode="double", length = n.iteration)
e.theta23.record=vector(mode="double", length = n.iteration)
theta12.record=vector(mode="double", length = n.iteration)
theta13.record=vector(mode="double", length = n.iteration)
theta23.record=vector(mode="double", length = n.iteration)

# this is a matrix used in calculation of parameter
m=matrix(c(rep(1, (k-1)^2)), nrow = k-1)
m[lower.tri(m)== FALSE] =0

i=1
errornum=0
for (i in 1: n.iteration) {

#update omega and R
omega.chol.est=angel2chol(theta12,theta13,theta23)
omega.est=angel2correlation(theta12,theta13,theta23)
R.sqr=diag(X%*% chol2inv(omega.chol.est) %*% t(X))
R.est=sqrt(R.sqr)
```

```
#f(x| mu_i, phi_i) k x n matrix
density.individual= apply(as.matrix(R.est), 1,
dgamma, shape=alpha.est,
rate=lamda.est)
#whole mixture pdf f(x)=p1*f(x|mu1,phi1) + p2*f(x|mu2,phi2)
density.mixture=p.est %*% density.individual
#probability vector of Zi vector
prob=t(t(density.individual*as.vector(p.est) ) /
as.vector(density.mixture))

zn=apply(prob, 2, rmultinom, n=1, size=1)

#apply(zn, 1, sum)
########################## continue
h.gamma=rgamma(1, k+eta1-1, eta2-sum(log(1-v[1:k-1])))

r=apply(zn, 1, sum)
v=rbeta(k-1,1+r[1:k-1],r[1:k-1] %*% m+h.gamma)
# m is a kinda of lower triangular matrix
v=c(v,1)
sum=0
for (j in 1:k) {
p.est[j]=(1-sum)*v[j]
sum=sum+p.est[j]
}
```

```
p.record[, i]=p.est

################################## Conjugate

lamda.est=rgamma(k, alpha.null+alpha.est*(apply(zn, 1, sum)),
rate=lamda.null+ zn %*% R.est)
lamda.record[, i]=lamda.est

################################## alpha
sigma=1
alpha.est.test=rnorm(k ,alpha.est, sigma)

A1=alpha.est.test * apply(zn, 1, sum) * log(lamda.est) +
alpha.est.test * zn %*% log(R.est) -
apply(zn, 1, sum)*lgamma(alpha.est.test)+
(-3-1)*log(alpha.est.test)

B1=alpha.est * apply(zn, 1, sum) * log(lamda.est) +
alpha.est * zn %*% log(R.est) -
apply(zn, 1, sum)*lgamma(alpha.est)+
(-3-1)*log(alpha.est)

logR=A1-B1
r=exp(logR)
r[is.nan(r)]=0
e=pmin(1, r)
```

```
u=runif(k, 0, 1)
alpha.est[u<=e & alpha.est.test>1]=
alpha.est.test[u<=e & alpha.est.test>1]
#acceptance rate
e.record[, i]=e
alpha.record[, i]=alpha.est

############### about rho metropolis, based on theta's
################# for theta12
theta12.prop=runif(1, theta12-0.1, theta12+0.1)

if(theta12.prop <0 || theta12.prop>pi){
e.theta12=0
errornum=(errornum+1)
}else{

r.theta12=exp(loglikelihood(theta12.prop,theta13,theta23)
-loglikelihood(theta12,theta13,theta23))

if(is.nan(r.theta12)){
r.theta12=0
errornum=(errornum+1)
}
e.theta12=min(1, r.theta12)
}

u=runif(1)
if(u<= e.theta12){
```

```
theta12=theta12.prop
}
#acceptance rate
e.theta12.record[i]=e.theta12
theta12.record[i]=theta12

############################theta13
theta13.prop=runif(1, theta13-0.1, theta13+0.1)

if(theta13.prop <0 || theta13.prop>pi){
e.theta13=0
errornum=(errornum+1)
}else{

r.theta13=exp(loglikelihood(theta12,theta13.prop,theta23)
-loglikelihood(theta12,theta13,theta23))

if(is.nan(r.theta13)){
r.theta13=0
errornum=(errornum+1)
}
e.theta13=min(1, r.theta13)
}

u=runif(1)
if(u<= e.theta13){
theta13=theta13.prop
}
```

```
#acceptance rate
e.theta13.record[i]=e.theta13
theta13.record[i]=theta13


################################ theta23
theta23.prop=runif(1, theta23-0.1, theta23+0.1)

if(theta23.prop <0 || theta23.prop>pi){
e.theta23=0
errornum=(errornum+1)
}else{

r.theta23=exp(loglikelihood(theta12,theta13,theta23.prop)
-loglikelihood(theta12,theta13,theta23))

if(is.nan(r.theta23)){
r.theta23=0
errornum=(errornum+1)
}
e.theta23=min(1, r.theta23)
}

u=runif(1)
if(u<= e.theta23){
theta23=theta23.prop
}
#acceptance rate
```

```
e.theta23.record[i]=e.theta23
theta23.record[i]=theta23

}

save(X,
theta12.record,
theta13.record,
theta23.record,
e.theta12.record,
e.theta13.record,
e.theta23.record,
alpha.record,
e.record,
lamda.record,
p.record,
R.est,
n.iteration,
#change file name when save red wine data
file = "january2011.RData")
```

B.3 Plot generation

```
#results based on real data
load("redwine.RData")
rho12=mean(cos(theta12.record[1000:n.iteration]))
rho13=mean(cos(theta13.record[1000:n.iteration]))
rho23=mean(cos(theta12.record[1000:n.iteration])*
```

```
cos(theta13.record[1000:n.iteration])+
sin(theta12.record[1000:n.iteration])*
sin(theta13.record[1000:n.iteration])*
cos(theta23.record[1000:n.iteration]))

#Estimated pdf from mcmc for R
est.mixture<-function(x){
test=dgamma(x, alpha.record[ ,1000:n.iteration],
lamda.record[ ,1000:n.iteration])
*p.record[ ,1000:n.iteration]
test=apply(test, 2, sum)
mean(test)

}

y=seq(0,5, length.out = 100)
est.density=sapply(y, est.mixture)



correlation=matrix(c(1,rho12,rho13,rho12,1,
rho23, rho13,rho23,1),
ncol = 3)
R.sqr=diag(X%*% solve(correlation) %*% t(X))
R.est=sqrt(R.sqr)

par(mfrow=c(1,1))
#estimated pdf over histogram
hist(R.est, freq = FALSE,ylim=c(0,0.6), xlab = "r", main = "")
```

```
lines(y,est.density, lwd=3)

#Theoretical lines
lines(y, dchisq(y^2,3)*2*y, col="red", lwd=8)
legend('topright', c('Estimated h(r)', 'Theoretical h(r)') ,
lty=c(1,1), col=c("black", "red"),
lwd = c(3, 8), bty='n', cex=2)
```

Curriculum Vitae

Panfeng Liang was from People's Republic of China. He went to Minzu University of China in the fall of 2009 and received his bachelor's degree in Computer Science in the Fall of 2013. He entered 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.

Contact Information: pliang@miners.utep.edu