CONTRIBUTIONS TO MULTIVARIATE DATA SCIENCE: ASSESSMENT AND IDENTIFICATION OF MULTIVARIATE DISTRIBUTIONS AND SUPERVISED LEARNING FOR GROUPS OF OBJECTS.

by

NGUYEN QUYNH HUONG TRAN

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN APPLIED MATHEMATICAL SCIENCES

2024

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Ravindra Khattree, Ph.D., Chair
Dorin Drignei, Ph.D.
Li Li, Ph.D.
Anuradha Roy, Ph.D.
Hon Yiu So, Ph.D.

To my family

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Prof. Ravindra Khattree, for his invaluable expertise, patience, and encouragement over the years. The interesting questions he raised always challenged me to grow as a researcher and writer. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level. These valuable experiences will hold a lasting place in my memory.

I sincerely thank my committee members, Prof. Dorin Drignei, Prof. Li Li, Prof. Anuradha Roy, and Prof. Hon Yiu So, for their thoughtful questions, comments, and feedback to improve this work. I appreciate their commitment to my academic development.

I am grateful to our Department of Mathematics and Statistics at Oakland University for providing me with graduate assistantships that supported me financially and enriched my graduate school experience.

Finally, I especially thank my parents, husband, and friends for their unconditional love, encouragement, and inspiration to pursue my dreams. I could only have done this with such a strong network of support.

Nguyen Quynh Huong Tran

ABSTRACT

CONTRIBUTIONS TO MULTIVARIATE DATA SCIENCE:
ASSESSMENT AND IDENTIFICATION OF MULTIVARIATE DISTRIBUTIONS
AND SUPERVISED LEARNING FOR GROUPS OF OBJECTS.

by

NGUYEN QUYNH HUONG TRAN

Adviser: Ravindra Khattree, Ph.D.

This dissertation considers three critical aspects of modern statistical analyses and machine learning, namely, (i) addressing the challenges posed by assessing the distributional assumptions of a univariate dataset, (ii) constructing graphical tests for the multivariate normality assumption, and (iii) exploring new algorithms for solving group classification problems, especially when the distance-based methods are not applicable.

First, we focus on the development and evaluation of graphical statistical tests for univariate datasets, aiming to assess any specific distributional assumption. Examination of normality is given special emphasis. Recognizing the impact of outliers on the normality assumption, this study also incorporates outlier detection methodologies and provides some graphical tools for their identification. T4 plot is introduced as an additional effective tool for this purpose.

Examination of multivariate normality is more challenging as, in this case, nonnormality may exhibit or mask itself in many different ways. Thus, in the second part, we emphasize graphical assessments of the multivariate normality assumption. This is done via MT3 and MT4 plots based on the derivatives of the cumulant generating function.

The final segment of the dissertation shifts the discussion towards machine learning algorithms devised specifically for group classification problems. This involves

the exploration of new methodologies that address the challenges inherent in classification and discrimination within complex datasets where other standard methods based on the classification of individual observation may not be very effective. For this, we rely on the data's eigenstructures. In the process, we also address the problem of dimensionality reduction. The problem of selection of copulas can be viewed as a corollary of the group classification problem. This has also been discussed in a separate chapter.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iv

ABSTRACT v

LIST OF TABLES xii

LIST OF FIGURES xv

LIST OF SYMBOLS xix

CHAPTER ONE
INTRODUCTION 1

1.1. Tests for Multivariate Normality 2

1.2. A Quick Review of Classification Rules 4

1.2.1. Naive Bayes 4

1.2.2. Logistic Regression 5

1.2.3. Discriminant Analysis 6

1.2.4. Support Vector Machines and Kernel Method 7

1.2.5. k-Nearest Neighbor 9

1.3. Copulas and Sklar's Theorem 9

CHAPTER TWO
ASSESSMENT OF DISTRIBUTIONAL ASSUMPTION: GRAPHICAL METHODS FOR UNIVARIATE DATA 12

2.1. Introduction 12

2.2. Univariate Normality Detection Using Score Spline Estimator 13

2.2.1. Illustrative Examples: Simulated Data 20

2.3. T_3 and T_4 plots 22

2.3.1. Two Extreme Pathological Examples 33

2.4. Assessing the Distributional Assumption: Any Continuous Distribution 38

2.5. A Quick Evaluation of Transformation Method in The Context of Formal Hypothesis Testing Methods 53

2.6. Two Examples: Cork Data and Water Potability Data 57

2.7. Concluding Remarks 75

CHAPTER THREE
EVALUATION OF MULTIVARIATE NORMALITY USING UNIVARIATE METHODS 82

3.1. Introduction 82

3.2. A Method Based on "$n \times p$-sample" 83

3.3. Illustrative Examples for "$n \times p$-sample" Method: Simulated Data 87

3.4. Two Examples for "$n \times p$-sample" Method: Cork Data and Water Potability Data Revisited 99

3.5. Outlier Detection Using Score Function 103

3.6. Tests Based on Best Linear Transformation 105

3.7. Illustrative Examples for Best Linear Transformation Method: Simulated Data 114

3.8. Two Examples for Best Linear Transformation Method: Cork Data and Water Potability Data Revisited 114

3.9. Concluding Remarks 115

CHAPTER FOUR
EVALUATION OF MULTIVARIATE NORMALITY VIA CUMULANT GENERATING FUNCTION 117

4.1. Introduction 117

4.2. Graphical Method Based on MT_3 Plot 118

4.2.1. Asymptotic Distribution of $\hat{\mathfrak{C}}^{(3)}(\mathbf{t})$: Multivariate Standard Normal Case 127

4.2.2. Asymptotic Distribution of $\hat{K}^{(3)}(\mathbf{t})$: Multivariate Non-standard Normal Case 139

4.2.3. Illustrative Examples: Simulated Data 140

4.2.4. Two Examples: Cork Data and Water Potability Data Revisited 141

4.3. MT_4 Plot 143

4.3.1. Illustrative Examples: Simulated Data 157

4.3.2. Two Examples: Cork Data and Water Potability Data revisited 160

4.4. Concluding Remarks 161

CHAPTER FIVE
GROUP DISCRIMINATION AND GROUP CLASSIFICATION 163

5.1. Introduction 163

5.2. Classification of An Observation by Majority Rule 164

5.3. Group Classification 169

5.3.1. Illustrative Examples: Simulated Data 170

5.4. Group Classification by Eigenstructures 173

5.4.1. Illustrative Examples: Simulated Data 178

5.5. An Example: Water Potability Dataset 178

5.6. Feature Selection Using Eigenstructures 180

5.6.1. Forward and Backward Selection Algorithms 184

5.6.2. Illustrative Examples: Simulated Data 188

5.6.3. Variable Selection for Water Potability Data 193

5.7. A Few Comments on Limitations 196

5.8. Conclusions and Remarks 197

CHAPTER SIX
ON CHOOSING COPULAS 199

6.1. Introduction 199

6.2. Some Approaches for Copula Selection 200

6.3. A Simulated Study 202

6.4. Choosing Copulas for ETF Data 211

6.5. Concluding Remarks 220

APPENDICES 221

A. Source Code for R-package *PlotNormTest* 222

A.1. Source code for score plot 223

TABLE OF CONTENTS — Continued

A.2. Source code for method of "$n \times p$" samples 228

A.3. Source code for "best" linear transformations method 229

A.4. Source code for method of using cumulant generating function 233

B. Details of Proofs for Chapter Four 267

C. Source Code of Chapter Two 270

D. Source Code of Chapter Three 311

E. Source Code of Chapter Four 329

F. Source Code of Chapter Five 340

F.1. Group classification methods 341

F.2. Variable selection methods 350

F.3. Synthetic data experiments 355

F.4. Water Potability revisited 372

G. Source Code of Chapter Six 376

G.1. Synthetic data experiments 377

G.2. Choosing copulas for ETF data 389

REFERENCES 397

LIST OF TABLES

Table 2.1	Some continuous distributions	46
Table 2.2	Empirical power of formal hypothesis tests on inverse cdf transformation	57
Table 2.3	Cork dataset	59
Table 2.4	Water Potability dataset	60
Table 3.1	Formal hypothesis tests on "$n \times p$" univariate observations, simulated data	99
Table 3.2	p-values for normality tests based on "$n \times p$-sample" method, Cork dataset	100
Table 3.3	p-values for normality tests based on "$n \times p$-sample" method, Water Potability (= Good) dataset	100
Table 3.4	Critical value for a size α test using the best linear transformation - minimizing skewness $\hat{\varkappa}_1^2/_p$	111
Table 3.5	Critical value for a size α test using the best linear transformation - minimizing kurtosis $\hat{\varkappa}_2^2/_p$	112
Table 3.6	Critical value for a size α test using the best linear transformation - minimizing average of skewness and kurtosis $\bar{\varkappa}^2/_p$	113
Table 3.7	Test statistics of best linear transformation method, simulated data	114
Table 3.8	Test statistics of best linear transformation method, Cork and Water Potability (= Good) datasets	115
Table 5.1	Misclassification rate using "Majority rule" and Optimal classification rule	171
Table 5.2	Misclassification rate using Eigenstructures	179

Table 5.3	Misclassification rates using Eigenstructures and "Majority rule" - Water Potability dataset	182
Table 5.4	Forward selection - different means and same covariance matrix, simulated example	189
Table 5.5	Backward selection - different means and same covariance matrix, simulated example	190
Table 5.6	Forward selection - same mean and different covariance matrices, simulated example	193
Table 5.7	Backward selection - same mean and different covariance matrices, simulated example	194
Table 5.8	Forward selection - Water Potability dataset	195
Table 5.9	Backward selection - Water Potability dataset	195
Table 6.1	Some popular copulas and their Kendall's tau correlation	205
Table 6.2	Estimated probabilities of correct classification for various copulas ($p = 2$, sample size $r = 500$, Kendall's $\tau = 0.7$)	207
Table 6.3	The highest of estimated probabilities of misclassification for various copulas ($p = 2$, sample size $r = 500$, Kendall's $\tau = 0.7$)	208
Table 6.4	Estimated probabilities of correct classification for various copulas ($p = 5$, sample size $r = 1000$)	209
Table 6.5	The highest estimated probabilities of misclassification for various copulas ($p = 5$, sample size $r = 1000$)	210
Table 6.6	Copulas classification for stock returns of FDVLX and FDGRX	214

Table 6.7	Ratios of eigenvalues of sub-datasets obtained from appending copula observations of the FDVLX- FDGRX dataset to each of the corresponding samples from fitted copulas	215
Table 6.8	Copulas classification for stock returns of IWF, LRGF, ONEO, OUSA, and PSQ	219
Table 6.9	Ratios of eigenvalues of sub-datasets obtained from appending copula observations of the IWF-LRGF -ONEO-OUSA-PSQ dataset to each of the corresponding samples from fitted copulas	220

LIST OF FIGURES

Figure 2.1	Estimator of score function for sample from $N(0,1)$	17
Figure 2.2	Score plots for simulated data	21
Figure 2.3	Score plots for normally distributed data with two outliers	23
Figure 2.4	Cumulant generating functions and their derivatives	26
Figure 2.5	T_3 and T_4 plots for sample from $N(0,1)$	32
Figure 2.6	T_3 plots for certain non-normal distributions	34
Figure 2.7	T_4 plots for certain non-normal distributions	35
Figure 2.8	Density and cumulant generating functions of McCullagh distribution	36
Figure 2.9	T_3 and T_4 plots for a sample generated from McCullagh distribution	37
Figure 2.10	Density and cumulant generating functions of Logistic $\left(\mu = 0, s = \sqrt{3}/\pi\right)$	38
Figure 2.11	T_3 and T_4 plots for a sample generated from Logistic $\left(\mu = 0, s = \sqrt{3}/\pi\right)$	39
Figure 2.12	Density functions of certain non-normal distributions	41
Figure 2.13	Graphical assessment for simulated data from Inverse Beta$(\alpha = 10, \beta = 3.5)$	42
Figure 2.14	Graphical assessment for simulated data from Gamma$(\alpha = 2, \beta = 2)$	43
Figure 2.15	Graphical assessment for simulated data from Inverse Gaussian$(\mu = 1, \lambda = 2)$	44
Figure 2.16	Graphical assessment for simulated data from Lomax$(\beta = 6, \theta = .25)$	45

Figure 2.17	Q - Q plots when fitting simulated data to test for distributional assumption	49
Figure 2.18	Score plots when fitting simulated data to test for distributional assumption	50
Figure 2.19	T_3 plots when fitting simulated data to test for distributional assumption	51
Figure 2.20	T_4 plots when fitting simulated data to test for distributional assumption	52
Figure 2.21	Density estimators from the given samples	54
Figure 2.22	Empirical Type I error of transformation method	56
Figure 2.23	Graphical assessment for Cork data set	64
Figure 2.24	Graphical assessment for Water Potability (= Good) data set	74
Figure 2.25	Fitting appropriate distributional assumptions for non-normal variables of Water Potability (= Good) data set	80
Figure 3.1	Empirical Type I error of "$n \times p$-sample" method	88
Figure 3.2	Graphical tests on "$n \times p$" univariate observations, data follow $N_5(\mathbf{0}_5, \mathbf{I}_5)$	90
Figure 3.3	Graphical tests on "$n \times p$" univariate observations, data follow non-standard normal distribution	91
Figure 3.4	Graphical tests on "$n \times p$" univariate observations, data follow Mixture normal distribution	92
Figure 3.5	Graphical tests on "$n \times p$" univariate observations, data follow Gaussian-copulas distribution	93

Figure 3.6	Graphical tests on "$n \times p$" univariate observations, data follow Lomax distribution	94
Figure 3.7	Graphical tests on "$n \times p$" univariate observations, data follow multivariate $t(5)$ distribution	96
Figure 3.8	Graphical tests on "$n \times p$" univariate observations, data follow multivariate $t(15)$ distribution	97
Figure 3.9	Graphical tests on "$n \times p$" univariate observations, data follow Kotz distribution	98
Figure 3.10	Graphical tests on "$n \times p$" univariate observations, Cork dataset	101
Figure 3.11	Graphical tests on "$n \times p$" univariate observations, Water Potability (= Good) dataset	102
Figure 3.12	Score plots on "$n \times p$" univariate observations, data follow $N_5(\mathbf{0}_5, \mathbf{I}_5)$ with two outliers	106
Figure 4.1	Plot of $L_3(\mathbf{t})$ for bivariate distributions	123
Figure 4.2	Plot of $L_3(\mathbf{t})$ for bivariate distributions with parameterization	124
Figure 4.3	MT_3 plots for bivariate distributed data - without probability bands	125
Figure 4.4	MT_3 plots for five-dimensional data - without probability bands	126
Figure 4.5	MT_3 plots for normally distributed data	141
Figure 4.6	MT_3 plots for non-normal data	142
Figure 4.7	MT_3 plot for Cork dataset	143
Figure 4.8	MT_3 plot for Water Potability (= Good) dataset	144
Figure 4.9	MT_4 plots for normally distributed data	158

Figure 4.10	MT_4 plots for non-normal data	159
Figure 4.11	MT_4 plot for Cork dataset	160
Figure 4.12	MT_4 plot for Water Potability (= Good) dataset	161
Figure 5.1	Comparison of summary statistics of the good water and the bad water populations	181
Figure 6.1	Scatter plots of certain bivariate copulas	203
Figure 6.2	Contour plots of various copulas	204
Figure 6.3	Scatterplots of stock return prices of FDVLX and FDGRX	212
Figure 6.4	Simulated samples from copulas estimated from stock returns of FDVLX and FDGRX	213
Figure 6.5	Scatterplots of daily return of IWF, LRGF, ONEO, OUSA and PSQ.	217
Figure 6.6	Scatterplots of copula observations from the daily return of IWF, LRGF, ONEO, OUSA, and PSQ	218

LIST OF SYMBOLS

$\mathrm{M}_Y(t)$	Moment generating function of random variable Y
$\mathfrak{C}_Y(t)$	Cumulant generating function of random variable Y
$\mu(t)$	Moment generating function of $N(0,1)$
$\hat{\mu}(t)$	Estimator of $\mu(t)$
$\mu^{(k)}(t)$	The k^{th} derivative of $\mu(t)$
$\hat{K}(t)$	Estimator of $\mathfrak{C}_Y(t)$
$\hat{K}^{(3)}(t)$	Third derivative of $\hat{K}(t)$
$\hat{K}^{(4)}(t)$	Fourth derivative of $\hat{K}(t)$
$\hat{T}^3(t)$	Test statistic for univariate T_3 plot
$\hat{T}^4(t)$	Test statistic for univariate T_4 plot
$\mathcal{Z}$	Test statistics for tests based on derivatives of $\mathfrak{C}_Y(t)$
$\mathbf{X}$	Random vector in p dimention $\mathbf{X} = [X_1, X_2, \ldots, X_p]'$
$\mathbf{t}$	Vector in p dimention $\mathbf{t} = [t_1, t_2, \ldots, t_p]'$
$\mathrm{M}_{\mathbf{X}}(\mathbf{t})$	Moment generating function of random vector $\mathbf{X}$
$\hat{\mathrm{M}}_{\mathbf{X}}(\mathbf{t})$	Estimator of $\mathrm{M}_{\mathbf{X}}(\mathbf{t})$
$\mathfrak{C}(\mathbf{t})$	Cumulant generating function of random vector $\mathbf{X}$
$\hat{\mathfrak{C}}(\mathbf{t})$	Estimator of $\mathfrak{C}(\mathbf{t})$
$\mu(\mathbf{t})$	Moment generating function of $N(\mathbf{0}_p, \mathbf{I}_p)$
$\hat{\mu}(\mathbf{t})$	Estimator of $\mu(\mathbf{t})$
$\hat{\mu}^{(j_1 j_2 j_3)}(\mathbf{t})$	Third derivatives of $\hat{\mu}(\mathbf{t})$ with respect to $t_{j_1}, t_{j_2}, t_{j_3}$

$\hat{\mathrm{K}}^{(j_1 j_2 j_3)}(\mathbf{t})$	Third derivatives of $\log[\hat{\mu}(\mathbf{t})]$ with respect to $t_{j_1}, t_{j_2}, t_{j_3}$
$\hat{\mu}^{(j_1 j_2 j_3 j_4)}(\mathbf{t})$	Fourth derivatives of $\hat{\mu}(\mathbf{t})$ with respect to $t_{j_1}, t_{j_2}, t_{j_3}, t_{j_4}$
$\hat{\mathrm{K}}^{(j_1 j_2 j_3 j_4)}(\mathbf{t})$	Fourth derivatives of $\log[\hat{\mu}(\mathbf{t})]$ with respect to $t_{j_1}, t_{j_2}, t_{j_3}, t_{j_4}$
$L_3(t)$	Test statistics for MT_3 plot
$L_4(t)$	Test statistics for MT_4 plot

CHAPTER ONE

INTRODUCTION

This dissertation addresses two main problems of multivariate data analysis, the assessment of normality assumption and the problem of classifying a group of objects.

In the context of statistical analysis and inference, normality assumption plays an essential role in ensuring the validity of conclusions drawn from data. Traditional statistical tests, while valuable, may not provide a comprehensive understanding of the underlying data structure. Graphical methods, on the other hand, offer a nuanced perspective by enabling us to visualize the distributional characteristics of multivariate data. Particular graphical approaches to build and develop efficient methods aiming to depict departure from normality have been addressed in this dissertation. We extend the traditional graphical methods for the problem of finding departure from normality in multivariate data to certain more effective yet somewhat computationally intensive methods. We thus introduce both indirect and direct methods to obtain a graphical assessment for a high-dimensional data set. The indirect approaches require transformations to univariate normal samples from multivariate data, while the other approach directly depicts the departure from normality using multivariate data. Related works are implemented in the R software and have been made available online.

We also consider the problem of *group classification and group discrimination*. It is a powerful and versatile analytical tool from a machine learning point of view. We provide an overview of fundamental concepts of classification for individual observation, including the common approaches, and evaluate these for the group classification problems. We then introduce our new classification method that utilizes the inherent data structure within the group. This approach is then applied to the problem of finding an

appropriate dependence structure for multivariate, non-normally distributed data, especially for copulas.

Before we proceed to our original contributions, we will provide a quick yet critical review of the existing literature on the related topics. This, in turn, also introduces notations and sets a theme for our work.

1.1 Tests for Multivariate Normality

According to a review by Henze [1], the classes of affine invariant and consistent tests for the hypothesis that a sample of size n of p features, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ were drawn from a multivariate normal distribution can be classified into the following six categories:

1. Graphical procedures, including Q-Q plot, contour plot of density, scatter plot, or Andrew plot. See Khattree and Naik [2] for more details.

2. Tests based on the measure of skewness and kurtosis, introduced by Mardia [3], Móri, Rohatgi and Szekely [4], Malkovich and Afifi [5], Koziol [6].

3. Tests based on angles and radii, introduced by Rayleigh [7], Koziol [8] and [9], Paulson, Roohan, and Sullo [10], Moore and Stubblebine [11], Tsai and Koziol [12], Mudholkar, McDermott and Srivastava [13].

4. Consistent procedures based on empirical characteristic functions, introduced and developed by Csörgő [14], Baringhaus and Henze [15], Henze and Zirkler [16].

5. Density-based tests introduced by Henze and Zirkler [16], Bowman and Foster [17].

6. Miscellaneous procedures introduced by Roy [18], Zhu, Fang and Bhatti [19], Zhu, Wong and Fang [20], Kuwana and Kariya [21], Kariya and George [22], Beirlant, Mason and Vynckier [23], Cox and Small [24].

In many cases, the assessment of multivariate normality can be done effectively via graphical method, as Q - Q plot. Henze [1] also pointed out that the multivariate normality

tests based on Mardia's skewness and kurtosis [3] are the "formal omnibus", and the test relying on characteristic function introduced by Henze and Zirkler [16] is able to identify evidence of non-normality from any alternative distribution as the sample size increases. We will now briefly summarize these three effective procedures.

Q-Q plot assesses the departure from normality of a multivariate data set by transformation to nearly independent univariate chi-square distributed samples, see Khattree and Naik [2]. With a sample of size n, say, $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, the values

$$d_i^2 = (\mathbf{X}_i - \bar{\mathbf{X}})' \boldsymbol{S}^{-1} (\mathbf{X}_i - \bar{\mathbf{X}}), i = 1, 2, \ldots, n$$

where $\bar{\mathbf{X}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{X}_i$ and $\boldsymbol{S} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})'$, are approximately distributed as a chi-square distribution with p degree of freedom, provided $\mathbf{X}_i \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $i = 1, 2, \ldots, n$ holds. Hence, plotting ordered d_i^2 values against quantiles of χ_p^2 should approximately resemble a 45° line passing through the origin.

Mardia [3] proposed measures of skewness and kurtosis for a multivariate distribution as follows,

$$\beta_{1,p} = \mathbb{E}\left[(\mathbf{X} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{X} - \boldsymbol{\mu})\right]^3$$

and

$$\beta_{2,p} = \mathbb{E}\left[(\mathbf{X} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{X} - \boldsymbol{\mu})\right]^2.$$

For the multivariate normal distribution $\beta_{1,p} = 0$ and $\beta_{2,p} = p(p+2)$. With a random samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, Mardia proposed the estimators $b_{1,p}$ and $b_{2,p}$ of $\beta_{1,p}$ and $\beta_{2,p}$ as

$$b_{1,p} = \frac{1}{n^2} \sum_{i,k=1}^{n} \left[\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)' \mathbf{S}^{-1} \left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\right]^3$$

and

$$b_{2,p} = \frac{1}{n} \sum_{i=1}^{n} \left[\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)' \mathbf{S}^{-1} \left(\mathbf{X}_i - \bar{\mathbf{X}}\right)\right]^2.$$

Under normality assumption, $\frac{nb_{1,p}}{6} \sim \chi^2_{p(p+1)(p+2)/6}$ and $\frac{b_{2,p} - p(p+2)}{\sqrt{8p(p+2)/n}} \sim N(0,1)$.

The Henze-Zirkler test statistic [16] based on the distance between empirical characteristic function and hypothesis characteristic function is given by

$$D_{n,\beta} = \int_{\mathbb{R}^p} \left| \Psi_n(\mathbf{t}) - \exp\left(\frac{-1}{2}\|\mathbf{t}\|^2\right) \right|^2 \varphi_\beta(\mathbf{t}) d\mathbf{t}$$

where $\beta > 0$ is a smoothing parameter, and $\Psi_n(\mathbf{t}) = \frac{1}{n}\sum_{i=1}^{n} \exp\left(i\mathbf{t}'\boldsymbol{S}^{-1}(\mathbf{X}_i - \bar{\mathbf{X}})\right), \mathbf{t} \in \mathbb{R}^p$ is the empirical characteristic function estimated from sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ and $\varphi_\beta = \left(2\pi\beta^2\right)^{-p/2} \exp\left(\frac{-\|\mathbf{t}\|^2}{2\beta^2}\right), \mathbf{t} \in \mathbb{R}^p$ is the characteristics function of $N_p(\mathbf{0}_p, \beta^2\mathbf{I}_p)$. A large value of the Henze-Zirkler test statistics $D_{n,\beta}$ is evidence of non-normality. They also propose the optimal choice of the smoothing parameter β as

$$\beta = \beta_p(n) = \frac{1}{\sqrt{2}}\left(\frac{n(2p+1)}{4}\right)^{1/p+4}.$$

See [16] for the details of the derivation of the asymptotic distribution of the Henze-Zirkler test.

1.2 A Quick Review of Classification Rules

In the vast landscape of supervised learning, one of the fundamental challenges is the task of classification of objects. For the simplicity of presentation, in this review, we will consider the classification of an observation into one of the two populations, let them be Π_1 and Π_2. Let $\mathbf{X}_{11}, \mathbf{X}_{12}, \ldots \mathbf{X}_{1n_1}$ and $\mathbf{X}_{21}, \mathbf{X}_{22}, \ldots \mathbf{X}_{2n_2}$ be multivariate samples of size n_1 and n_2 from Π_1 and Π_2, respectively. We now review some of the well-known classifiers.

1.2.1 Naive Bayes

Using Bayes' theorem, the posterior probability of the k^{th} population, given the observed value $\mathbf{X}$ is given by

$$f(\Pi_k|\mathbf{X}) = \frac{f(\Pi_k)f(\mathbf{X}|\Pi_k)}{f(\mathbf{X})}, k = 1, 2,$$

where $f(\Pi_k)$ and $f(\mathbf{X})$ are the *prior probabilities* of Π_k and $\mathbf{X}$. Under the assumption that conditional on Π_k, variables in $\mathbf{X} = \left(X_1, X_2, \ldots X_p\right)'$ are independent, the above posterior probability for the k^{th} population is

$$f(\Pi_k|\mathbf{X}) = \frac{f(\Pi_k)f(X_1|\Pi_k)f(X_2|\Pi_k)\ldots f(X_p|\Pi_k)}{f(\mathbf{X})}, k = 1, 2.$$

Moreover, as the denominator does not depend on k, the classification rule of the Naive Bayes classifier can be as: The membership of $\mathbf{X}$ is Π_1 if

$$f(\Pi_1)f(X_1|\Pi_1)f(X_2|\Pi_1)\ldots f(X_p|\Pi_1) \geq f(\Pi_2)f(X_1|\Pi_2)f(X_2|\Pi_2)\ldots f(X_p|\Pi_2).$$

If the probability distributions are unknown, the estimators $\hat{f}(\Pi_k)$ and $\hat{f}(X_j|\Pi_k)$ of $f(\Pi_k)$ and each of the conditional probability $f(X_j|\Pi_k)$, $j = 1, 2, \ldots, p$, $k = 1, 2$ can be obtained from random samples. Most of the time, we assume that variables follow a Gaussian distribution (continuous variables), Multinomial distribution (variables represent counts or frequencies), or Bernoulli distribution (variables are binary).

A drawback of the Naive Bayes method is that one or both estimators of conditional probabilities $\hat{f}(X_j|\Pi_k)$ may be zero. This might happen if some particular values for some of the variables have not been observed. In such a case, the estimated posterior probability will be zero, and thus, the resulting bias seriously affects the classification.

1.2.2 Logistic Regression

For a multivariate observation $\mathbf{X}$, let $\pi(\mathbf{X})$ and $1 - \pi(\mathbf{X})$ be the probabilities that $\mathbf{X}$ belongs to Π_1 and Π_2. Logistic regression assumes that the logarithm of the odds is a linear function of the vector of variables $\mathbf{X}$. That is,

$$\text{logit}[\pi(\mathbf{X})] = \log\left[\frac{\pi(\mathbf{X})}{1 - \pi(\mathbf{X})}\right] = \alpha + \boldsymbol{\beta}'\mathbf{X}.$$

The estimators $\hat{\alpha}$ and $\hat{\boldsymbol{\beta}}$ of α and $\boldsymbol{\beta}$ respectively can be obtained either by weighted least square or by maximum likelihood approaches, expressing the appropriate likelihood

function. The estimated logistic function is then

$$\log\left[\frac{\hat{\pi}(\mathbf{X})}{1-\hat{\pi}(\mathbf{X})}\right] = \hat{\alpha} + \hat{\boldsymbol{\beta}}'\mathbf{X}$$

and hence $\hat{\pi}(\mathbf{X}) = \dfrac{\exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}'\mathbf{X})}{1+\exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}'\mathbf{X})}$ is the estimated probability of $\mathbf{X}$ belonged to Π_1. Then, classification rule based on logistic regression is: recommend to classify $\mathbf{X}$ into Π_1 if $\dfrac{\exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}'\mathbf{X})}{1+\exp(\hat{\alpha} + \hat{\boldsymbol{\beta}}'\mathbf{X})} > \dfrac{1}{2}$.

1.2.3 Discriminant Analysis

The basic idea of discriminant analysis is to find a set of functions that best separates the transformed mean vectors of the groups. There are various techniques for classification based on discriminant functions. The linear discriminant analysis is applicable when the covariance variance matrices for populations are the same. Quadratic discriminant analysis makes no such assumption.

Specially, linear discriminant analysis aims to find a vector $\mathbf{a} \in \mathbb{R}^p$ such that $|\mathbf{a}'\boldsymbol{\mu}_1 - \mathbf{a}'\boldsymbol{\mu}_2|$ is maximized. The optimum choice is $\mathbf{a} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$. The linear discriminant function is then

$$\begin{aligned}\mathcal{D}(\mathbf{x}) &= \mathbf{a}'\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) \\ &= (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\Sigma^{-1}\mathbf{x} - \frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2).\end{aligned}$$

One classifies observation $\mathbf{X}$ to Π_1 if $\mathcal{D}(\mathbf{X}) \geq 0$.

In the case of unequal covariance matrices, and assuming the two populations to be multivariate normal, the quadratic discriminant rule is based on

$$Q(\mathbf{x}) = \frac{1}{2}\log\left(\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_2|}\right) - \frac{1}{2}(\boldsymbol{\mu}_1'\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2'\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\mu}_2) + \left(\boldsymbol{\mu}_1'\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\mu}_2'\boldsymbol{\Sigma}_2^{-1}\right)\mathbf{x} - \frac{1}{2}\mathbf{x}'\left(\boldsymbol{\Sigma}_1^{-1} - \boldsymbol{\Sigma}_2^{-1}\right)\mathbf{x}$$

and one classifies observation $\mathbf{X}$ to Π_1 if $Q(\mathbf{X}) \geq 0$.

When parameters $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ are unknown, one may use their estimators from the corresponding random samples and then

$$\hat{\mathcal{D}}(\mathbf{x}) = (\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2)' S_{pool}^{-1}\mathbf{x} - \frac{1}{2}(\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2)'\mathbf{S}_{pool}^{-1}(\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2),$$
$$\hat{Q}(\mathbf{x}) = \frac{1}{2}\log\left(\frac{|\mathbf{S}_1|}{|\mathbf{S}_2|}\right) - \frac{1}{2}(\bar{\mathbf{X}}_1'\mathbf{S}_1^{-1}\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2'\mathbf{S}_2^{-1}\bar{\mathbf{X}}_2) + \left(\bar{\mathbf{X}}_1'\mathbf{S}_1^{-1} - \bar{\mathbf{X}}_2'\mathbf{S}_2^{-1}\right)\mathbf{x} - \frac{1}{2}\mathbf{x}'\left(\mathbf{S}_1^{-1} - \mathbf{S}_2^{-1}\right)\mathbf{x}$$

where $\bar{\mathbf{X}}_k = \frac{1}{n_k}\sum_{i=1}^{n_k}\mathbf{X}_{ki}$ and $\mathbf{S}_k = \frac{1}{n_k - 1}\sum_{i=1}^{n_k}(\mathbf{X}_{ki} - \bar{\mathbf{X}}_k)(\mathbf{X}_{ki} - \bar{\mathbf{X}}_k)'$ are the sample means and sample covariance matrices of k^{th} population, $k = 1, 2$, and
$\mathbf{S}_{pool} = \frac{(n_1 - 1)\mathbf{S}_1 + (n_2 - 1)\mathbf{S}_2}{n_1 + n_2 - 2}$ is the pooled covariance matrix.

1.2.4 Support Vector Machines and Kernel Method

The linear discriminant analysis attempts to find a hyperplane that separates the means of the two populations as far as possible. The Support Vector Machine (SVM), on the other hand, focuses on finding a hyperplane that maximizes the distance from the two groups of sample points from Π_1 and Π_2, see [25]. Denote $n = n_1 + n_2$, and let y_i be the label of observation $\mathbf{X}_i, i = 1, 2, \ldots, n$, i.e

$$\begin{cases} y_i = 1, \mathbf{X}_i \in \Pi_1 \\ y_i = -1, \mathbf{X}_i \in \Pi_2. \end{cases}$$

SVM looks for the hyperplane $\mathbf{w} \in \mathbb{R}^p$ and some bias term $b \in \mathbb{R}$ such that

$$\begin{cases} \underset{\mathbf{w},b}{\text{minimize}} & \frac{1}{2}\mathbf{w}'\mathbf{w} \\ \text{subject to} & y_i(\mathbf{w}'\mathbf{X}_i - b) \geq 1, \forall i \in \{1, 2, \ldots n\} \end{cases} \tag{1.1}$$

The above constrained optimization problem can be reduced to minimizing the following Lagrange function with the use of $\alpha_i \geq 0$ as the Lagrange multipliers,

$$L(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}'\mathbf{w} - \sum_{i=1}^{n}\alpha_i\left[y_i(\mathbf{w}'\mathbf{X}_i + b) - 1\right]. \tag{1.2}$$

$\mathbf{w}$ and b must be solutions of the following system

$$\begin{cases} \dfrac{\partial L}{\partial b} = & -\sum_{i=1}^{n} \alpha y_i = 0 \\ \dfrac{\partial L}{\partial \mathbf{w}} = & \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{X}_i = 0. \end{cases} \tag{1.3}$$

Substitutes equations in the system (1.3) to (1.2) and obtain

$$\begin{aligned} L(\mathbf{w}, b) &= \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{X}_i\right)'\left(\sum_{i=1}^{n} \alpha_i y_i \mathbf{X}_i\right) - \sum_{i=1}^{n} \alpha_i \left\{ y_i \left[\left(\sum_{j=1}^{n} \alpha_j y_j \mathbf{X}_j\right)' \mathbf{X}_i + b \right] - 1 \right\} \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{X}_i' \mathbf{X}_j. \end{aligned}$$

Hence, maximizing $L(\mathbf{w}, b)$ with respect to $\mathbf{w}$ and b can be reduced to the problem of maximizing the following dual formulation

$$\begin{cases} \underset{\mathbf{w},b}{\text{maximizing}} & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{X}_i' \mathbf{X}_j \\ \text{subject to} & \alpha_i \geq 0, i = 1, 2, \ldots, n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{1.4}$$

which is a constrained quadratic programming problem.

The objective function in the optimization problem (1.4) depends on sample data only through the expression of $K(\mathbf{X}_i, \mathbf{X}_j) = \mathbf{X}_i' \mathbf{X}_j, i, j = 1, 2, \ldots n$. If data are not linearly separable, then with another appropriate choice of kernel $K(\mathbf{X}_i, \mathbf{X}_j)$, instead of $\mathbf{X}_i' \mathbf{X}_j$, one can transform data $\mathbf{X}_i, i = 1, 2, \ldots, n$ into another space where they are linearly separable. Below are some types of kernels $K(.)$ that have been widely used,

- Polynomial kernel $K(\mathbf{X}_i, \mathbf{X}_j) = \left(\mathbf{X}_i' \mathbf{X}_j + c\right)^d$, for some degree $d \in \mathbb{R}$.
- Gaussian kernel $K(\mathbf{X}_i, \mathbf{X}_j) = \exp\left(\dfrac{1}{2\sigma^2}(\mathbf{X}_i - \mathbf{X}_j)'(\mathbf{X}_i - \mathbf{X}_j)\right)$, for some $\sigma \in \mathbb{R}$.
- Hyperbolic tangent kernel (known as Sigmoid kernel)

  $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh\left(\kappa \mathbf{X}_i' \mathbf{X}_j + c\right)$, for some $\kappa > 0$ and $c \in \mathbb{R}$.

1.2.5 k-Nearest Neighbor

k-Nearest Neighbor (k-NN) is a non-parametric statistical classifier, which assumes that observations close to each other (under some specified distance metric) share the same posterior probability distribution, see [26]. Hence, k-NN classifiers classify an unknown membership observation $\mathbf{X}$ to the population to which the nearest neighbors belong. The k value in k-NN is the number of nearest neighbors that are considered when making a decision for a data point to be classified.

The choice of distance metric is crucial in determining the proximity. Let $\mathbf{X} = \left[X_1, X_2, \ldots, X_p\right]'$ and $\mathbf{Z} = \left[Z_1, Z_2, \ldots, Z_p\right]'$ are the two observations in a p-dimensional space. Some popular distance metrics between $\mathbf{X}$ and $\mathbf{Z}$ are defined as below:

- Euclidean distance: $d(\mathbf{X}, \mathbf{Z}) = \sqrt{\sum_{i=1}^{p}(X_i - Z_i)^2}$.
- Manhattan distance: $d(\mathbf{X}, \mathbf{Z}) = \sum_{i=1}^{p}|X_i - Z_i|$.
- Minkowski distance: $d(\mathbf{X}, \mathbf{Z}) = \left(\sum_{i=1}^{p}|X_i - Z_i|^k\right)^{1/k}, k \geq 1$.

k-NN may fail when the true distribution of the data to be classified is skewed, as "majority voting" among the k neighbors tends to bias for the dominant density in the region. The weighted nearest neighbor classifier was introduced as a modification to tackle this problem. See [27] for details.

1.3 Copulas and Sklar's Theorem

Copula is a multivariate cumulative distribution function, for which each variable is marginally uniformly distributed on the interval $[0, 1]$. More specifically, let $\mathbf{U} = [U_1, U_2, \ldots, U_p]' \in \mathbb{R}^p$, such that $U_i \sim U(0, 1), i = 1, 2, \ldots p$. Let $C(.)$ be the

cumulative distribution function of $\mathbf{U}$, i.e

$$C(u_1, u_2, \ldots, u_p) = P\left[U_1 \leq u_1, U_2 \leq u_2, \ldots, U_p \leq u_p\right]. \tag{1.5}$$

Then C is a p-dimensional copula.

Copulas are used to model the (nonlinear) dependence between variables, and this can be separated from any inference of their marginal distribution. This is stated formally in Sklar's theorem below.

Theorem 1.1 (Sklar's Theorem [28], [29])**.** *Let H be a p-dimensional cumulative distribution function of a random vector $\mathbf{X} = [X_1, X_2, \ldots, X_p]'$ with marginal cumulative distributions $F_1, F_2, \ldots, F_p$. Then there exists a p-dimensional copula C, such that for all $\mathbf{x} = [x_1, x_2, \ldots, x_p]' \in \mathbb{R}^p$,*

$$H(\mathbf{x}) = H(x_1, x_2, \ldots, x_p) = C\big(F_1(x_1), F_2(x_2), \ldots, F_p(x_p)\big).$$

Conversely, if C is a p-dimensional copula and $F_1, F_2, \ldots F_p$ are the marginal distribution functions of $X_1, \ldots, X_p$ respectively, then the function H defined above is a p-dimensional distribution function with marginal $F_1, F_2, \ldots F_p$.

Let $h(.)$ and $c(.)$ be the probability density functions of H and C respectively , then

$$\begin{aligned} h(x_1, x_2, \ldots, x_p) &= \frac{\partial^p}{\partial x_1 \partial x_2 \ldots \partial x_p} H(x_1, x_2, \ldots x_p) \\ &= c\big(F_1(x_1), F_2(x_2), \ldots, F_p(x_p)\big) f_1(x_1) f_2(x_2) \ldots f_p(x_p) \\ &= c\big(u_1, u_2, \ldots u_p\big) f_1(x_1) f_2(x_2) \ldots f_p(x_p), \end{aligned}$$

where the vector $[u_1, u_2, \ldots, u_p]'$ is defined as in equation (1.5). Hence, simulating a distribution of a random vector $\mathbf{X} \in \mathbb{R}^p$ can be done via simulating the marginal distributions $F_1, F_2, \ldots F_p$ and using the dependence structure $C(.)$ between $U_i \sim U(0, 1), i = 1, 2, \ldots, p$ separately. We will employ Sklar's theorem to assess the distributional assumption for the ETF dataset in Chapter 6.

The work in this dissertation has been driven by practical concentrations, focusing on the development of suitable statistical concepts needed in real-world applications. In that sense, this work truly falls under the domain of applied multivariate data science. As more and more advanced computational resources are made available and a better understanding of complex research problems is attained, a wide range of applications of our graphical multivariate normality tests and group classification approaches can be further implemented and used routinely.

CHAPTER TWO

ASSESSMENT OF DISTRIBUTIONAL ASSUMPTION: GRAPHICAL METHODS FOR UNIVARIATE DATA

2.1 Introduction

Normality assumption plays an important role in most modeling and related inferential problems. As a result, numerous test procedures to assess univariate normality have been developed over the last seventy years. Well-known methods address issues of non-normality relating to skewness and distributional behavior of the tails using the sample skewness and sample kurtosis [30]. Meanwhile, numerous tests are based on empirical distribution, among which the widely used ones are Shapiro-Wilk test [31], and Anderson-Darling test [32] as a modification of Kolmogorov-Smirnov statistic with larger weight to the tails of the distributions. Lin and Mudholkar [33] introduced a test statistic that relies on the independence of sample mean and sample variance. Murota and Takeuchi [34] make use of the characterization of the normality, exploiting its exponential form.

Often, it is more effective to assess and convey information about the departure from normality and the extent of that via graphical procedures. The Q-Q plot and P-P plots are the most common of these graphical procedures (see Khattree and Naik [35] and [36]). Other more advanced methods are based on the linearity of the score function developed by Bera and Ng in [37] and [38], and properties of the moment generating function derived by Ghosh [39]. However, one of the problems with all these procedures has been that in the case of symmetric yet non-normal distributions, these quite often result in plots erroneously confirming normality.

Section 2.2 considers the problem of graphical assessment of normality based on score functions, where we rely on the score spline estimator introduced by Bera and Ng [37]. Section 2.3 considers the approach of T_3 plots introduced by Ghosh [39], where we

further extend Ghosh's work to make the assessment more effective. In Section 2.4, we rely on the graphical methods developed in the previous two sections to assess the distributional assumptions for data coming from any continuous probability distribution. The key fact there is the well-known cdf-transformation that can transform any data to normally distributed data, in which case, the method introduced here (or any other previously developed methods) can be adopted. We conclude this chapter with a few important remarks.

2.2 Univariate Normality Detection Using Score Spline Estimator

First, we recall the definition of the score function.

Definition 2.1 (Score function)**.** Let $f(y)$ be a probability density function of a univariate random variable Y. The corresponding score function is defined as

$$\psi(y) = -\frac{d}{dy}\log f(y) = -\frac{f'(y)}{f(y)}.$$

From the above definition, Lemma 2.1 immediately follows.

Lemma 2.1. A probability density distribution is symmetric if and only if its score function is an odd function, that is if and only if $\psi(-y) = -\psi(y)$.

Proof. Without loss of generality, assume that the random variable $Y \sim f(y)$, with $f(.)$ is symmetric about 0. Then we note that $\psi(-y) = -\dfrac{f'(-y)}{f(-y)} = \dfrac{f'(y)}{f(y)} = -\psi(y)$.

The reserve also follows from

$$\psi(-y) = -\psi(y) \Rightarrow \frac{f'(-y)}{f(-y)} = \frac{f'(y)}{f(y)} \Rightarrow -\frac{f'(y)}{f(-y)} = \frac{f'(y)}{f(y)} \Rightarrow -f(-y) = f(y).$$

□

An important implication of the above result is that if the empirical score function of a standardized data set exhibits a behavior considerately different from that of an odd function, then the density function must be asymmetric. Thus one important step in

assessing the symmetry (and possibly normality) is to effectively estimate the score function.

Cox and Ng [37], [40] suggested estimating a score function $\psi_0(.)$ of an unknown cumulative distribution function $F_0(.)$ with density $f_0(.)$ by minimizing the mean-squared error

$$L(\psi) = \int_{-\infty}^{\infty} (\psi - \psi_0)^2 dF_0(y) = \int_{-\infty}^{\infty} \left[\psi(y) - \psi_0(y)\right]^2 f_0(y) dy.$$

To do this, we observed that $\psi_0(y)dF_0(y) = -\dfrac{f_0'(y)}{f_0(y)} f_0(y)dy = -f_0'(y)dy$. Now, the integration by parts yields:

$$\int_{-\infty}^{\infty} \psi\psi_0 dF_0(y) = -\psi(y) f_0(y)|_{-\infty}^{\infty} + \int_{-\infty}^{+\infty} \psi' dF_0(y)$$

where under normality the first term vanishes because $\lim_{y\to\pm\infty} f_0(y) = 0$. Therefore,

$$\begin{aligned}
\int_{-\infty}^{\infty} (\psi - \psi_0)^2 dF_0(y) &= \int_{-\infty}^{\infty} \psi^2 dF_0(y) - 2\int_{-\infty}^{\infty} \psi\psi_0 dF_0 + \int_{-\infty}^{\infty} \psi_0^2 dF_0(y) \\
&= \int_{-\infty}^{\infty} \psi^2 dF_0(y) - 2\int_{-\infty}^{\infty} \psi' dF_0 + \int_{-\infty}^{\infty} \psi_0^2 dF_0(y) \\
&= \int_{-\infty}^{\infty} (\psi^2 - 2\psi') dF_0(y) + \int_{-\infty}^{\infty} \psi_0^2 dF_0(y).
\end{aligned}$$

A more general result of the above equation can be found in Cox [40]. The second term in the right-hand side is independent of ψ. Thus, the above optimization problem reduces to

$$\min_{\psi} L(\psi) = \int_{-\infty}^{\infty} (\psi^2 - 2\psi') dF_0(y).$$

Note that in practice $F_0(y)$ is unknown. If the empirical distribution function based on a sample of size n, say F_n, is observed, the problem is ill-defined, since then there is no minimizer as we can make the objective function arbitrarily small. Thus, an alternative approach proposed by Cox [40] is to solve the problem for a given interval $[a, b]$,

$$\min_{\psi} L(\psi) = \int_a^b (\psi^2 - 2\psi) dF_n(y) + \lambda \int_a^b \left(\psi''(y)\right)^2 dy, \tag{2.1}$$

such that

$$\left\{\psi : \psi, \psi' \text{ are absolutely continuous and } \int_a^b \left[\psi''(y)\right]^2 dy < +\infty\right\},$$

where $\lambda > 0$ is a pre-chosen smoothing parameter. Cox [40] also showed that as $\lambda \to +\infty$, the solution of the problem in (2.1) approaches $\dfrac{y - \bar{Y}}{S^2}$ which is the maximum likelihood estimator of the score function $\psi_0(y)$ under $N(\mu, \sigma^2)$. Therefore, λ represents the trade-off between smoothness and fidelity.

We now consider the estimation problem. When data are given as a random sample, then for an ordered sample $a < y_1 < \cdots < y_n < b$, Ng [37] suggests minimizing

$$L(\psi) = \sum_{i=1}^{n} p_i \int_a^b \left(\psi^2(y) - 2\psi'(y)\right)\delta_{y_i}(y)dy + \lambda \int_a^b \left(\psi''(y)\right)^2 dy$$

where

$$\delta_\alpha(y) = \begin{cases} 1, \text{ if } y = \alpha \\ 0, \text{ otherwise} \end{cases}$$

and p_i's are the weight assigned to the ordered sample point y_i, $i = 1, 2 \ldots n$ so that $\sum_{i=1}^{n} p_i = 1$.

For the optimization problem, the following result is useful.

Lemma 2.2 (Euler - Lagrange equation)**.** The stationary values of the functional

$$I[f] = \int_a^b \mathcal{L}\left(y, f, f', f'', \ldots, f^{(k)}\right) dy$$

can be obtained from the Euler - Lagrange equation:

$$\frac{\partial \mathcal{L}}{\partial f} - \frac{d}{dy}\left(\frac{\partial \mathcal{L}}{\partial f'}\right) + \frac{d^2}{dy^2}\left(\frac{\partial \mathcal{L}}{\partial f''}\right) - \cdots + (-1)^k \frac{d^k}{dy^k}\left(\frac{\partial \mathcal{L}}{\partial f^{(k)}}\right) = 0.$$

For our problem, $\mathcal{L}(y, \psi, \psi', \psi'')) = \sum_{i=1}^{n} p_i \left(\psi^2(y) - 2\psi'(y)\right) \delta_{y_i}(y) + \lambda\left(\psi''(y)\right)^2$, and $I[\psi] = \int_a^b \mathcal{L}(y, \psi, \psi', \psi'')dy$, with $k = 2$. Accordingly, we have

$$\frac{\partial \mathcal{L}}{\partial \psi} = 2\sum_{i=1}^{n} p_i \psi(y)\delta_{y_i}(y),$$
$$\frac{d}{dy}\left(\frac{\partial \mathcal{L}}{\partial \psi'}\right) = -2\sum_{i=1}^{n} p_i \delta_{y_i}(y),$$
$$\frac{d^2}{dy^2}\left(\frac{\partial \mathcal{L}}{\partial \psi''}\right) = \frac{d}{dy^2}\left(2\psi''(y)\right) = 2\psi^{(4)}(y).$$

Euler - Lagrange equation thus gives:

$$\sum_{i=1}^{n} p_i \left[\psi(y)\delta_{y_i}(y) + \delta'_{y_i}(y)\right] + \lambda\psi^{(4)}(y) = 0.$$

Ng [37] showed that using the smooth spline algorithm, an estimate $\hat{\psi}(.)$ of $\psi_0(.)$ obtained by the above approach is a piecewise cubic function of the form:

$$\hat{\psi}(y) = a_i + b_i(y - y_i) + c_i(y - y_i)^2 + d_i(y - y_i)^3, \text{ for } y \in [y_i, y_{i+1}], i = 1 \ldots, n-1 \quad (2.2)$$

such that

$$\hat{\psi}^{(k)}(y_i+) - \hat{\psi}^{(k)}(y_i-) = \begin{cases} 0 & \text{if } k = 0, 1 \\ -\dfrac{p_i}{\lambda} & \text{if } k = 2 \\ -\dfrac{p_i\hat{\psi}(y_i)}{\lambda} & \text{if } k = 3. \end{cases} \quad (2.3)$$

Clearly, solving for (2.2) must return a_i as an estimate of $\psi_0(y_i)$, $i = 1, 2, \ldots n$.

Note that under normality assumption $\psi_0(y) = \dfrac{y - \mu}{\sigma^2}$. Thus a_i in (2.2) must be a linear function of y_i, $i = 1, 2, \ldots, n$. In Figure 2.1, we have illustrated $a_i = \hat{\psi}(y_i)$ as a function of y_i, $i = 1, 2, \ldots 200$ random samples from $N(\mu = 0, \sigma^2 = 1)$, with smoothing parameter $\lambda = 0.5$. Any departure from normality will manifest as non-linearity in this plot. We can, however, further refine such a plot by providing some probability bands, which necessitates an assessment of variance of $\hat{\psi}(y_i)$ for various y_i, $i = 1, 2, \ldots, n$ values.

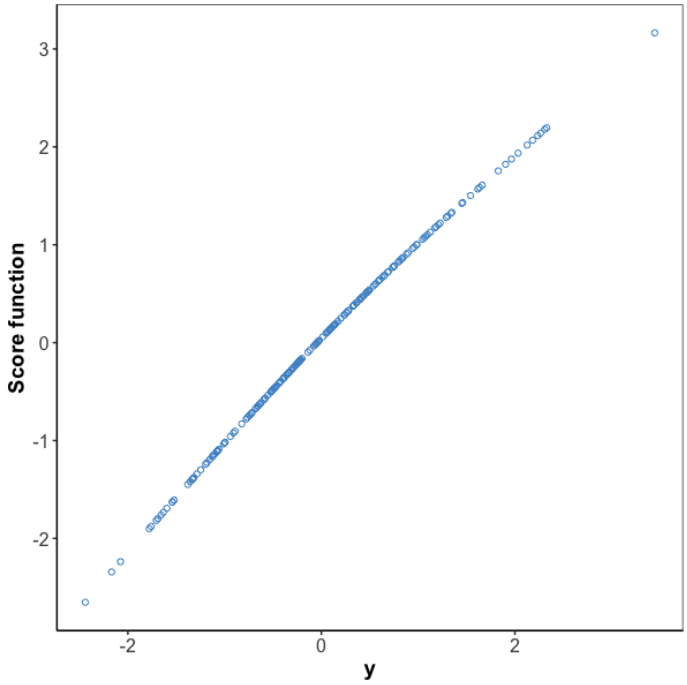


Figure 2.1: Estimator of score function for sample from $N(0,1)$, $n = 200$.

We compute these bands by the leave-one-out method. Especially under normality, we have the following lemma.

Lemma 2.3. Assume we have a random sample of size n from a normal distribution $N(\mu, \sigma^2)$. Then

(i) $\hat{\psi}_u(y) = \dfrac{n-3}{n-1}\dfrac{y-\bar{Y}}{S^2}$ is an unbiased estimator for score function $\psi_0(y) = \dfrac{y-\mu}{\sigma^2}$.

(ii) $\mathrm{Var}\left[\hat{\psi}_u(y)\right] = \dfrac{2}{n-5}\left(\dfrac{y-\mu}{\sigma^2}\right)^2 + \dfrac{n-3}{n(n-5)\sigma^2}$.

(iii) An unbiased estimator of $\mathrm{Var}[\hat{\psi}_u(y)]$ is

$$\hat{\mathrm{Var}}(\hat{\psi}_u(y)) = \frac{2(n-3)}{(n-1)^2}\left(\frac{y-\bar{Y}}{S^2}\right)^2 + \frac{n-3}{n(n-1)S^2},$$

where $\bar{Y} = \dfrac{1}{n}\sum_{i=1}^{n} Y_i$ and $S^2 = \dfrac{1}{n-1}\sum (Y_i - \bar{Y})^2$ are the usual unbiased estimators of μ and σ^2 respectively.

Proof. It is well known that under normality and under random sampling, $\bar{Y}$ and S^2 are independent and that $u = \dfrac{(n-1)S^2}{\sigma^2} \sim \chi^2_{n-1}$. Therefore, for any $r \geq \dfrac{-n}{2} + 1$, we obtain $\mathbb{E}(u^r) = \dfrac{C_{n-1+2r}}{C_{n-1}}$, where $C_k = 2^{\frac{k}{2}}\Gamma\left(\dfrac{k}{2}\right)$.

Now for a given $y \in \mathbb{R}$, which is independent of the sample, we have

$$\mathbb{E}\left[\frac{y-\bar{Y}}{S^2}\right] = \mathbb{E}\left(y-\bar{Y}\right)\mathbb{E}\left((S^2)^{-1}\right) = \frac{n-1}{n-3}\frac{y-\mu}{\sigma^2}$$

and thus, an unbiased estimator for the score function is:

$$\hat{\psi}_u(y) = \frac{n-3}{n-1}\frac{y-\bar{Y}}{S^2}.$$

It is also easy to verify that, for a given y,

$$\begin{aligned}
\operatorname{Var}\left[\hat{\psi}_u(y)\right] &= \mathbb{E}\left[\frac{n-3}{n-1}\frac{y-\bar{Y}}{S^2}\right]^2 - \left[\mathbb{E}\left(\frac{n-3}{n-1}\frac{y-\bar{Y}}{S^2}\right)\right]^2 \\
&= \left(\frac{n-3}{n-1}\right)^2 \mathbb{E}(y-\bar{Y})^2\,\mathbb{E}\,S^{-4} - \left[\frac{y-\mu}{\sigma^2}\right]^2 \\
&= \left(\frac{n-3}{n-1}\right)^2 \left[\mathbb{E}\left[(y-\mu)-(\bar{Y}-\mu)\right]\right]^2 \frac{(n-1)^2}{(n-3)(n-5)\sigma^4} - \left[\frac{y-\mu}{\sigma^2}\right]^2 \\
&= \frac{n-3}{\sigma^4(n-5)}\left[\frac{\sigma^2}{n} + (y-\mu)^2\right] - \left[\frac{y-\mu}{\sigma^2}\right]^2 \\
&= \frac{2}{n-5}\left(\frac{y-\mu}{\sigma^2}\right)^2 + \frac{n-3}{n(n-5)\sigma^2},
\end{aligned}$$

which clearly depends on unknown parameters μ and σ^2. Thus, to estimate the above variance unbiasedly, we observed that,

$$\begin{aligned}
&\mathbb{E}\left[\frac{2}{n-3}\left(\frac{n-3}{n-1}\frac{y-\bar{Y}}{S^2}\right)^2 + \frac{n-3}{n(n-1)S^2}\right] \\
=&\frac{2}{n-3}\,\mathbb{E}\left(\frac{n-3}{n-1}\frac{y-\bar{Y}}{S^2}\right)^2 + \frac{n-3}{n(n-1)}\,\mathbb{E}\,S^{-2} \\
=&\frac{2}{n-3}\frac{n-3}{\sigma^4(n-5)}\left(\frac{\sigma^2}{n} + (y-\mu)^2\right) + \frac{n-3}{n(n-1)}\frac{n-1}{n-3}\frac{1}{\sigma^2} \\
=&\frac{2}{n-5}\left(\frac{y-\mu}{\sigma^2}\right)^2 + \frac{n-3}{n(n-5)\sigma^2}.
\end{aligned}$$

Therefore, an unbiased estimator for $\text{Var}(\hat{\psi}_u(y))$ is

$$\hat{\text{Var}}(\hat{\psi}_u(y)) = \frac{2}{n-3}\left(\frac{n-3}{n-1}\frac{y-\bar{Y}}{S^2}\right)^2 + \frac{n-3}{n(n-1)S^2}.$$

□

The spline algorithm returns values $a_i = \hat{\psi}(y_i)$ (see (2.2)), which is an estimate of $\psi_0(y_i)$. Under normality, we can expect a_i to be close to $\hat{\psi}_u(y_i), i = 1, 2, \ldots n$.

To implement the algorithm for each y_i, we use only the other $(n-1)$ observations. The above expression for the estimated variance can be used to obtain the approximate 2-sigma bounds for each value $a_i = \hat{\psi}(y_i)$. It is so because in the above calculations, "y" is assumed to be "given" and hence is not part of the data used to estimate $\psi_0(y)$. Thus, to estimate such variances at y_i, the point y_i should not be considered in the calculation. This necessitates the use of the "leave-one-out" method, where an estimate of variance is obtained without the i^{th} data point y_i. Hence,

$$\hat{\psi}_u(y_i) = \frac{n-4}{n-2}\frac{y_i - \bar{Y}_{-i}}{S^2_{-i}},$$

and its estimated variance is

$$\hat{\text{Var}}(\hat{\psi}_u(y_i)) = \frac{2(n-4)}{(n-2)^2}\left(\frac{y_i - \bar{Y}_{-i}}{S^2_{-i}}\right)^2 + \frac{n-4}{(n-1)(n-2)S^2_{-i}},$$

where the subscript $-i$ indicates that the i^{th} observation is not used in the calculation. Accordingly, approximately 2-sigma bounds region for $a_i = \hat{\psi}(y_i)$ is given as

$$\hat{\psi}_u(y_i) \pm 2\sqrt{\hat{\text{Var}}(\hat{\psi}_u(y_i))}.$$

We will use these bands in the plots of $a_i = \hat{\psi}(y_i)$ versus y_i. In each figure of the score function, we will also plot $\hat{\psi}_u(y_i)$ together with the 2-sigma bands $\hat{\psi}_u(y_i) \pm 2\sqrt{\hat{\text{Var}}(\hat{\psi}_u(y_i))}$. Values of a_i are represented by the symbol $\circ$ while values of $\hat{\psi}_u(y_i)$ and the 2-sigma bands are represented by the dot-dashed lines.

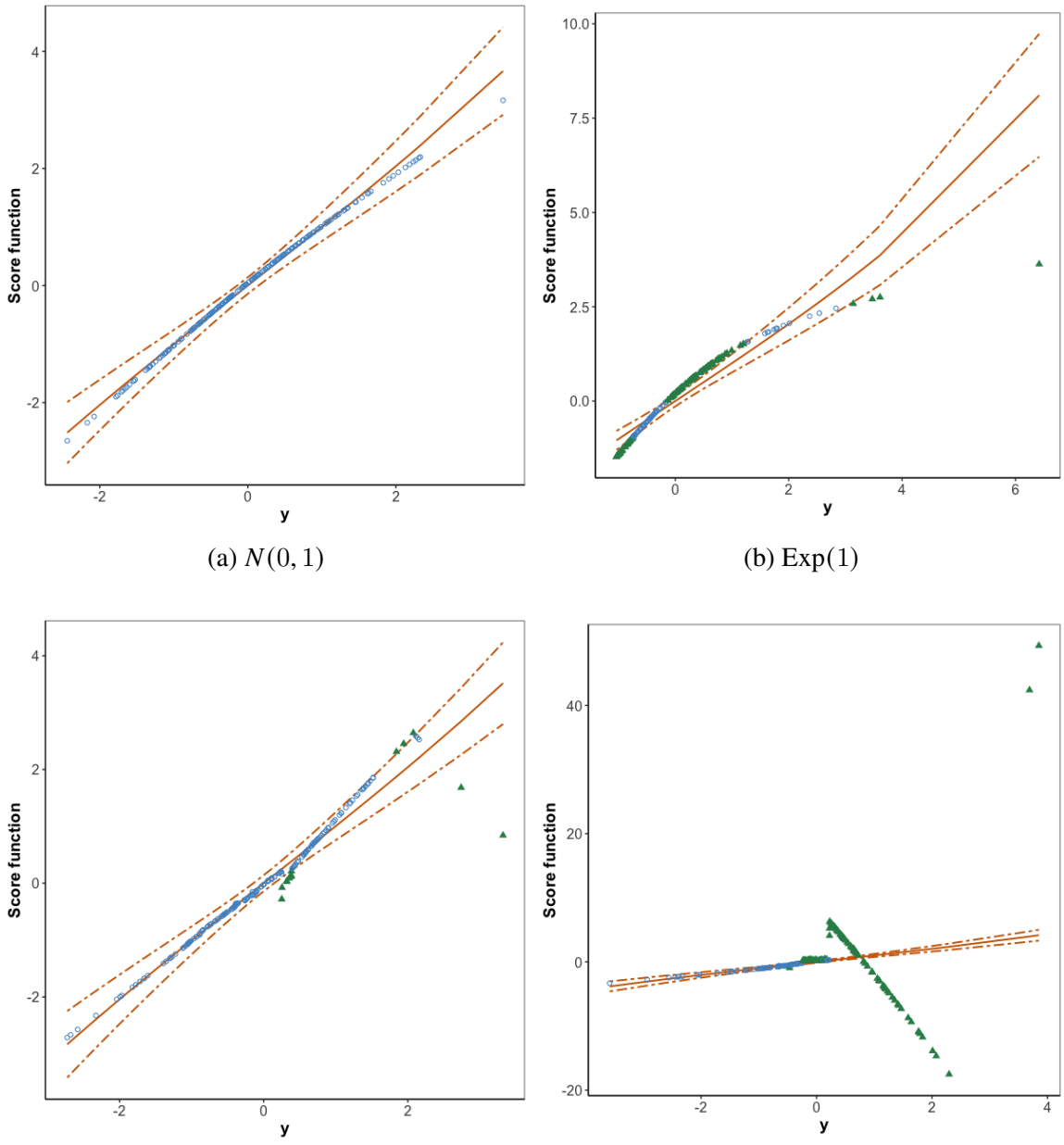
We know that with a sample of size n, $Y_1, Y_2, \ldots, Y_n$ from $N(\mu, \sigma^2)$, $\frac{Y_i - \mu}{\sigma} \sim N(0,1)$. As $\bar{Y}$ and S^2 are consistent estimators of μ and σ^2, the standardized data $\tilde{Y}_1 = \frac{Y_1 - \bar{Y}}{S}, \tilde{Y}_2 = \frac{Y_2 - \bar{Y}}{S}, \ldots, \tilde{Y}_n = \frac{Y_n - \bar{Y}}{S}$ can be seen asymptotically as a random sample of size n from $N(0,1)$. Hence, in the following illustrations, we will standardize the data before applying the score function recovery process and before obtaining the 2-sigma bands. For our work, we will also fix the smoothing parameter as $\lambda = .5$.

2.2.1 Illustrative Examples: Simulated Data

To illustrate our approach, we consider four simulated data sets from different distributions, each of size $n = 200$. Especially, distributions taken are $N(0,1)$, $\text{Exp}(1)$, $t(5)$ and Double exponential distribution ($\mu = 0, \sigma = 1, \beta = .5$). Note that the last two distributions are non-normal, yet symmetric. The objective is to assess normality for each of these cases. Figure 2.2a shows that the graph effectively confirms normality when it is indeed present, by the linearity of the plots where all points are within the 2-sigma bands. In contrast, in all the other three cases in Figure 2.2, a departure from linearity in the plots, as well as the presence of many points outside the probability bands, give a strong indication of non-normality.

Figures 2.3 illustrates the situations when each of the normally distributed ($N(0,1)$) samples ($n = 200$) contains two outliers. More specifically, Figure 2.3a is a case when the two outlying values are -1.9611 and 4.9429, which were randomly generated from $N(\mu = 1, \sigma = 2.5)$. In Figure 2.3b the two outlying values (-6.0353 and 0.0555) are from two different populations, $N(\mu = 0, \sigma = 5)$ and $N(\mu = 0, \sigma = .2)$, respectively. In Figure 2.3c, the two outliers are from $N(\mu = 1, \sigma = 2.5)$ and $N(\mu = -1, \sigma = 2.5)$, which have values -4.7014 and 4.9429. Finally, in Figure 2.3d, the two introduced outlying values 4.1701 and 7.8564 are from Gamma($\alpha = 2, \beta = 7.5$).[1]

[1]Gamma pdf is taken here as well as elsewhere is $f(y) = \frac{1}{\beta^\alpha \Gamma(\alpha)} y^{\alpha-1} \exp\left(\frac{-y}{\beta}\right), y > 0, \alpha > 0, \beta > 0.$



(a) $N(0, 1)$ (b) Exp(1)

(c) $t(5)$ (d) Double exponential distribution

Figure 2.2: Score plots for simulated data ($n = 200$).

Each of the pictures in Figure 2.3 shows a violation of linearity. It is clear that in the case of Figure 2.3a and Figure 2.3c, due to the high variance, $\sigma = 2.5$, some of the outliers may not be identifiable. In Figure 2.3b, the two populations have the same center, namely zero. Thus, the observation, which happened to be close to the center, is not identified. In Figure 2.3d, we have outliers coming from a highly asymmetric non-normal population with $\mu = \alpha\beta = 7.5$ and $\sigma^2 = \alpha\beta^2 = 7.5$, both of which are quite large. As a result, their presence affects the scores of many observations, and we see several observations falling outside the bands.

2.3 T_3 and T_4 plots

In this section, we consider the T_3 plot suggested by Ghosh [39] and then introduce the T_4 plot. The basic premise lies in the following important characterization of normal distribution.

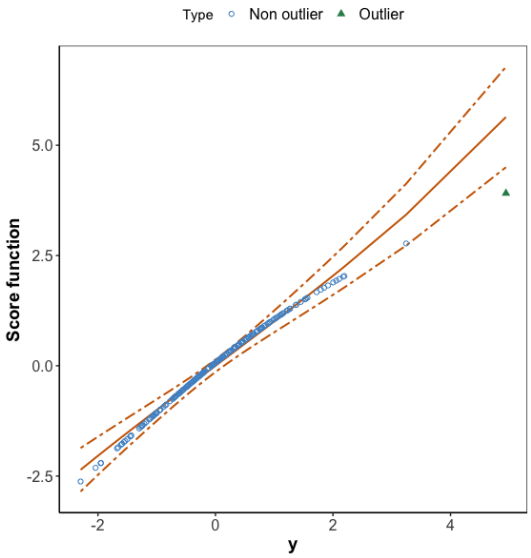
Lemma 2.4 ([41])**.** Let Y be a random variable. Let $\mathrm{M}_Y(t)$ be the moment generating function of Y. Then

$$\frac{\partial^3}{\partial t^3}\left[\log(\mathrm{M}_Y(t))\right] = 0 \text{ iff } Y \sim N(\mu, \sigma^2).$$

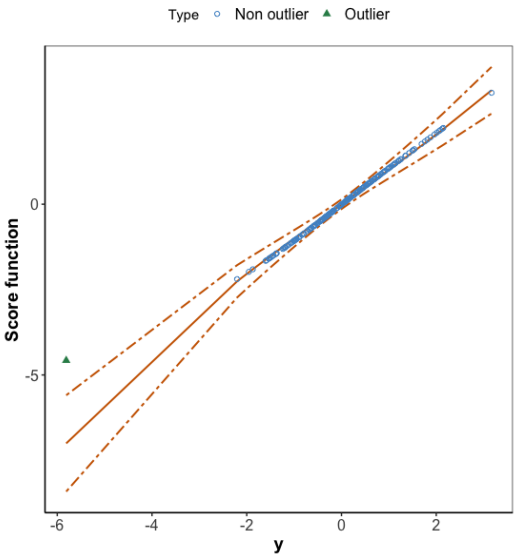
The cumulant generating function $\mathfrak{C}_Y(t)$ is the natural logarithm of the moment generating function. The k^{th} cumulants alternatives to the k^{th} moments $m_k = \mathrm{M}_Y^{(k)}(0)$, and are defined as $\kappa_k = \mathfrak{C}_Y^{(k)}(0)$. In order to construct plots, we first observe that the cumulant generating function of a normally distributed random variable Y with mean μ and variance σ^2 is given by

$$\mathfrak{C}_Y(t) = \log[\mathrm{M}_Y(t)] = \log\left[\mathbb{E}(e^{tY})\right] = \log\left[\exp\left(\mu t + \frac{t^2\sigma^2}{2}\right)\right] = \mu t + \frac{t^2\sigma^2}{2}.$$

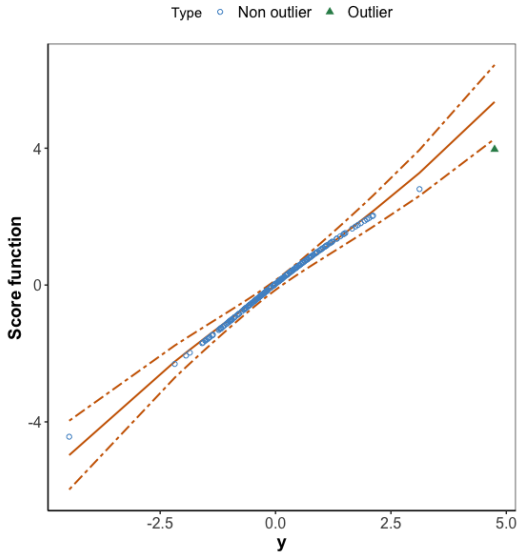
The above is a quadratic function in t, and its third and higher derivatives with respect to t all vanish. The above characterization also ensures the converse to be true. We



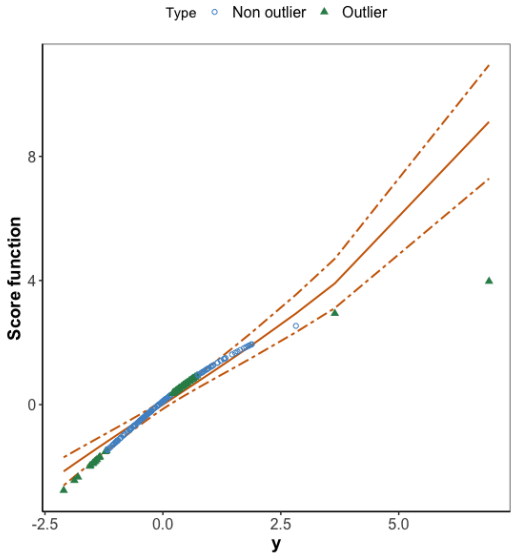
(a) 2 outliers from $N(1, 2.5^2)$



(b) 2 outliers from $N(0, 5^2)$ and $N(0, .2^2)$



(c) 2 outliers from $N(1, 2.5^2)$ and $N(-1, 2.5^2)$



(d) 2 outliers from Gamma($\alpha = 7.5, \beta = 1$)

Figure 2.3: Score plots for normally distributed data with two outliers ($n = 200$).

will denote the moment generating function of the standardized normally distributed variable $Z = \frac{Y-\mu}{\sigma}$ by $\mu(t)$. Clearly $Z \sim N(0,1)$ and $\mu(t) = \exp\left(\frac{t^2}{2}\right)$.

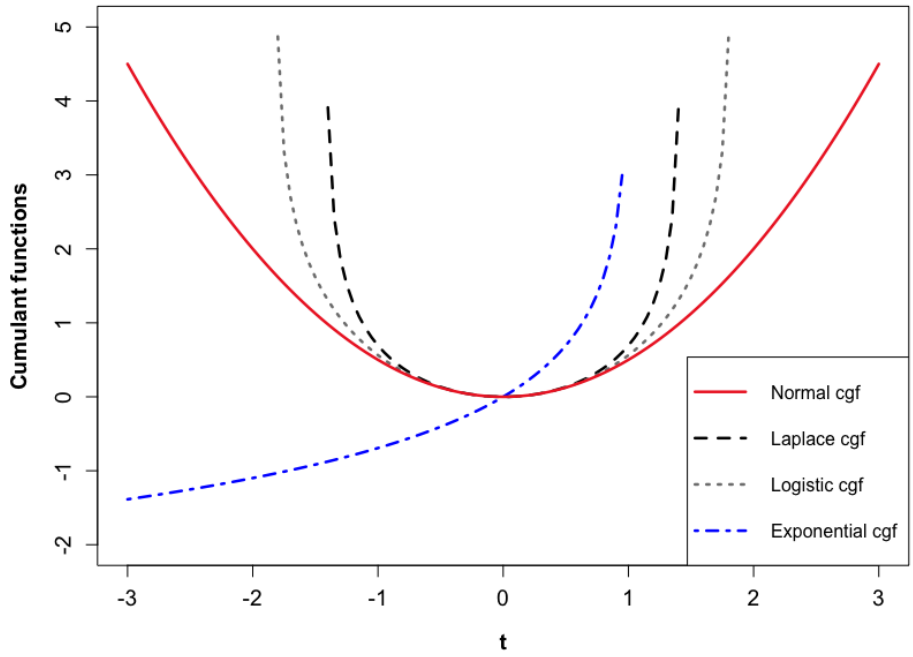
Figure 2.4a presents graphs of the cumulant generating functions of $N(0,1)$, which is a parabola, and of other density functions, namely Laplace $(0,1)$, Logistic $(0,1)$ and Exp(1) distributions. Figure 2.4b presents the plots of their third derivatives, in which the cumulant generating function of normal distribution results in a horizontal line passing through the origin. We also note that at point $t = 0$, the three curves of Normal, Logistic, and Laplace distribution all pass through the same point (the origin). This happens because the Laplace and Logistic distributions are symmetric. Hence, their corresponding behaviors around $t = 0$ are close to normal. This phenomenon does not occur in the case of Exponential distribution, which is asymmetric. Figure 2.4c is about the fourth derivative, in which the difference of symmetric but not normal families from the horizontal line becomes more prominent. Hence, using the third derivative of the cumulant generating function can assess non-normality due to asymmetry (as in the case of Exponential distribution). The fourth derivative of this function reveals evidence of non-normality due to kurtosis; hence, it can be applied when the underlying distribution is symmetric (as in the case of Laplace and Logistic distribution).

Given a random sample of size n, $Y_1, Y_2, \ldots Y_n$ from a normal distribution with mean μ and variance σ^2, one can provide an empirical estimator $\hat{M}_Y(t)$ of the moment generating function $\mathrm{M}_Y(t)$ as

$$\hat{M}_Y(t) = \frac{1}{n}\sum_{i=1}^{n} \exp(tY_i)$$

and hence the empirical studentized mgf is as follows

$$\hat{\mu}(t) = \frac{1}{n}\sum_{i=1}^{n} \exp\left(\frac{Y_i - \bar{Y}}{S}t\right),$$



(a) Cumulant generating functions.

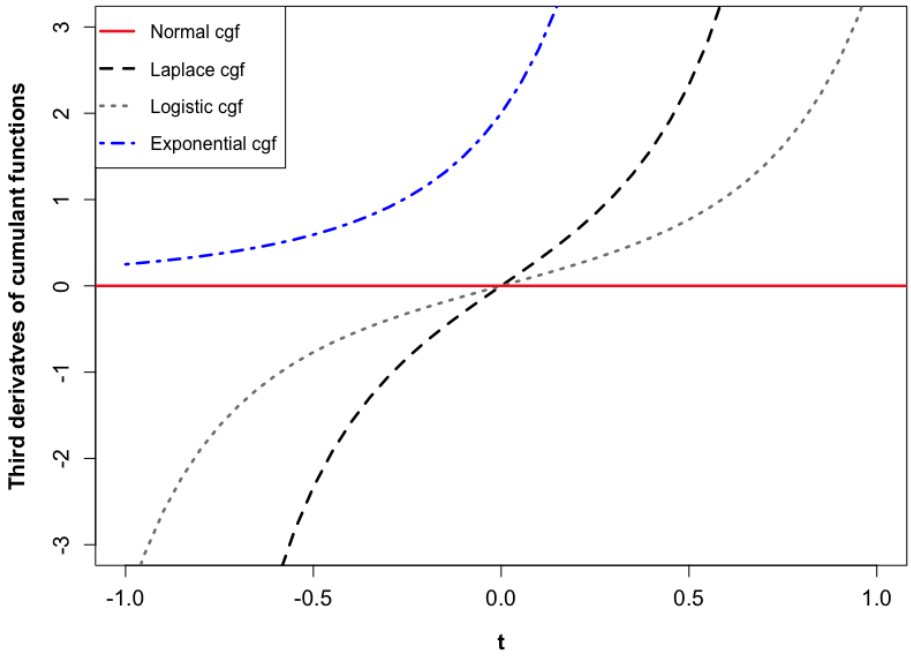
Figure 2.4: Cumulant generating functions and their derivatives.

where $\bar{Y}$ and S^2 are the usual unbiased estimators of μ and σ^2. As shown in [34], this studentized empirical moment generating function converges weakly to $\mu(t)$ as $n \to \infty$. Utilizing the above characterization of normal distribution, Ghosh has [39] provided an innovative approach to graphically assess the normality of the data. In the following, we first summarize her approach.

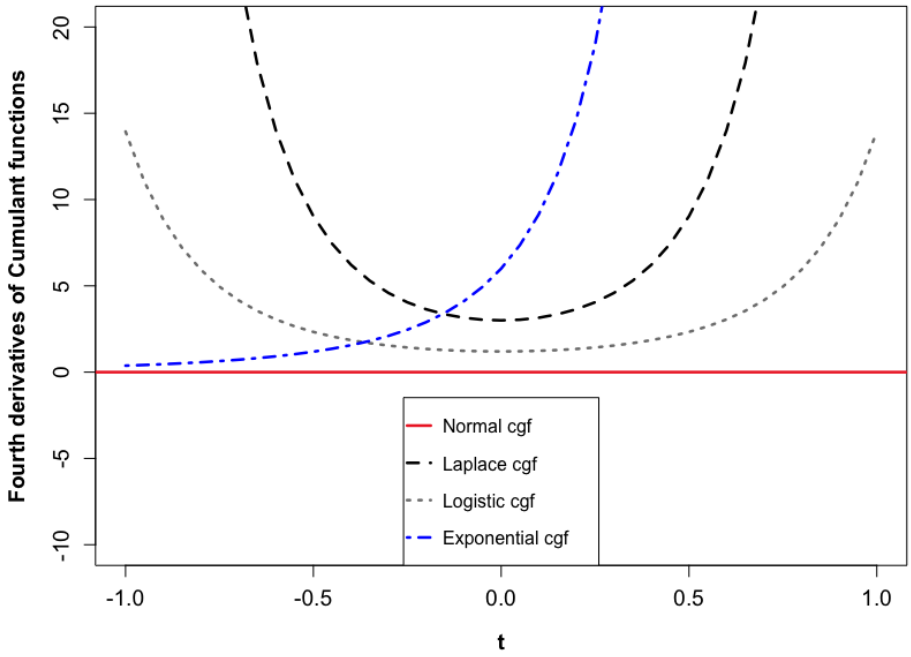
As indicated earlier, the mgf of standard normal distribution $N(0,1)$ is $\mu(t) = \exp\left(\frac{t^2}{2}\right)$. We further denote the k^{th} derivative of $\mu(t)$ as $\mu^{(k)}(t)$.

Lemma 2.5 (Derivatives of the univariate standard normal mgf)**.** Let random variable Y follows a standard normal distribution $N(0,1)$. Then

$$\mathbb{E}\left[Y^k \exp(tY)\right] = \mu^{(k)}(t).$$



(b) Third derivatives of cumulant generating functions.



(c) Fourth derivatives of cumulant generating functions.

Figure 2.4: Cumulant generating functions and their derivatives.

It thus follows that for the standard normal distribution,

$$\mu^{(1)}(t) = t\mu(t),$$
$$\mu^{(2)}(t) = (1 + t^2)\mu(t),$$
$$\mu^{(3)}(t) = (3t + t^3)\mu(t),$$
$$\mu^{(4)}(t) = (3 + 6t^2 + t^4)\mu(t),$$
$$\mu^{(5)}(t) = (15t + 10t^3 + t^5)\mu(t),$$
$$\mu^{(6)}(t) = (15 + 45t^2 + 15t^4 + t^6)\mu(t),$$
$$\mu^{(7)}(t) = (105t + 105t^3 + 21t^5 + t^7)\mu(t),$$
$$\mu^{(8)}(t) = (105 + 420t^2 + 210t^4 + 28t^6 + t^8)\mu(t).$$

The k^{th} derivatives of $\hat{\mu}(t)$ with respect to t is

$$\hat{\mu}^{(k)}(t) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \bar{Y}}{S}\right)^k \exp\left(\frac{t(Y_i - \bar{Y})}{S}\right).$$

Accordingly, an empirical estimator for the cumulant generating function of the standardized data, relying on $\hat{\mu}(t)$ is $\hat{K}(t) = \log[\hat{\mu}(t)]$, and an estimator for the k^{th} derivative can be defined as $\hat{K}^{(k)}(t) = \frac{\partial^k}{\partial t^k}\left[\hat{K}(t)\right] = \frac{\partial^k}{\partial t^k}\left[\log\left(\hat{\mu}(t)\right)\right]$. Some of the expressions of $\hat{K}^{(k)}(t)$ are derived as follows and are easy to verify.

$$\hat{K}^{(1)}(t) = \frac{\hat{\mu}^{(1)}(t)}{\hat{\mu}(t)}, \tag{2.4}$$
$$\hat{K}^{(2)}(t) = \frac{\hat{\mu}^{(2)}}{\hat{\mu}(t)} - \frac{\left[\hat{\mu}^{(1)}(t)\right]^2}{\hat{\mu}(t)},$$
$$\hat{K}^{(3)}(t) = \frac{\hat{\mu}^{(3)}(t)}{\hat{\mu}(t)} - 3 \times \frac{\hat{\mu}^{(2)}(t)}{\left[\hat{\mu}(t)\right]}\frac{\hat{\mu}'(t)}{\hat{\mu}(t)} + 2\frac{\left[\hat{\mu}^{(1)}(t)\right]^3}{\left[\hat{\mu}(t)\right]^3},$$
$$\hat{K}^{(4)}(t) = \frac{\hat{\mu}^{(4)}(t)}{\hat{\mu}(t)} - 4\frac{\hat{\mu}^{(3)}(t)}{\hat{\mu}(t)}\frac{\hat{\mu}^{(1)}(t)}{\hat{\mu}(t)} - 3\left[\frac{\hat{\mu}^{(2)}(t)}{\hat{\mu}(t)}\right]^2 + 12\frac{\hat{\mu}^{(2)}(t)}{\hat{\mu}(t)}\left[\frac{\hat{\mu}^{(1)}(t)}{\hat{\mu}(t)}\right]^2 - 6\left[\frac{\hat{\mu}^{(1)}(t)}{\hat{\mu}(t)}\right]^4.$$

The following result gives the asymptotic distribution of the vector of estimators of the derivatives of the moment generating function and, hence of the cumulant generating function under normality [34].

Lemma 2.6. Under normality, the ordered sequence

$$\mathcal{Z}_k(t) = \begin{bmatrix} Z_0(t) \\ Z_1(t) \\ \vdots \\ Z_k(t) \end{bmatrix} = \sqrt{n} \begin{bmatrix} \hat{\mu}(t) - \mu(t) \\ \hat{\mu}^{(1)}(t) - \mu^{(1)}(t) \\ \vdots \\ \hat{\mu}^{(k)}(t) - \mu^{(k)}(t) \end{bmatrix}, t \in [-a, a], a > 0 \tag{2.5}$$

converges weakly to a stochastic process $\mathcal{E}_k(t) = [\xi_0(t), \xi_1(t), \ldots, \xi_k(t)]'$ with zero mean and covariance function given by

$$\begin{aligned} K_{ij}(t,s) =& \operatorname{Cov}(\xi_i(t), \xi_j(s)) \\ =& \mu^{(i+j)}(t+s) - \mu^{(i)}(t)\mu^{(j)}(s) \\ & - \frac{1}{2}\left[j\mu^{(j)}(s) + s\mu^{(j+1)}(s)\right]\left[\mu^{(i+2)}(t) - \mu^{(i)}(t)\right] \\ & - \frac{1}{2}\left[i\mu^{(i)}(t) + t\mu^{(i+1)}(t)\right]\left[\mu^{(j+2)}(s) - \mu^{(j)}(s)\right] \\ & - \left[s\mu^{(j)}(s) + j\mu^{(j-1)}(s)\right]\mu^{(i+1)}(t) - \left[t\mu^{(i)}(t) + i\mu^{(i-1)}(t)\right]\mu^{(j+1)}(s) \\ & + \frac{1}{2}\left[j\mu^{(j)}(s) + s\mu^{(j+1)}(s)\right]\left[i\mu^{(i)}(t) + t\mu^{(i+1)}(t)\right] \\ & + \left[s\mu^{(j)}(s) + j\mu^{(j-1)}(s)\right]\left[t\mu^{(i)}(t) + i\mu^{(i-1)}(t)\right]. \end{aligned}$$

Theorem 2.1 ([39] Asymptotic distribution of the estimators of the derivatives of the cumulant generating function)**.** *Under normality and for any $k \geq 3$, the process $\hat{T}^k(t) = \sqrt{n}\hat{K}^{(k)}(t)$ converges weakly to a zero mean Gaussian process $\mathcal{T}_k(t)$ with*

$$\operatorname{Cov}\left(\mathcal{T}_k(t), \mathcal{T}_k(s)\right) = a_k(t)' \operatorname{Cov}\left(\mathcal{E}_k(t), \mathcal{E}_k(s)\right) a_k(s),$$

where

$$a_k(t) = \left[\frac{d}{d\hat{\mu}} g_k\big(U_k(t)\big)\Big|_{\xi(t)}, \frac{d}{d\hat{\mu}^{(1)}} g_k\big(U_k(t)\big)\Big|_{\xi(t)}, \ldots, \frac{d}{d\hat{\mu}^{(k)}} g_k\big(U_k(t)\big)\Big|_{\xi(t)} \right]',$$

and the function $g_k(.)$ *is defined as*

$$g_k\big(U_k(t)\big) = \frac{d^k}{dt^k} \log \hat{\mu}_Y(t)$$

with $U_k(t) = \left[\hat{\mu}(t), \hat{\mu}^{(1)}(t), \ldots \hat{\mu}^{(k)}(t)\right]'$ *and* $\xi(t) = \left[\mu(t), \mu^{(1)}(t), \ldots \mu^{(k)}(t)\right]'$.

A detailed proof of Theorem 2.1 can be found in [39] and the generalization to the case of multivariate data will be presented in Chapter 4. We will give only a general idea of this proof here. Applying Taylor's expansion, we can show that

$$\hat{T}^k(t) = \sqrt{n}\hat{K}^{(k)}(t) \approx a_k(t)'\mathcal{Z}_k(t).$$

Together with the aid of Lemma 2.6, we conclude $\hat{T}^k(t)$ converges weakly to a Gaussian process $\mathcal{T}_k(t) = a_k(t)'\mathcal{E}_k(t)$ as $n \to +\infty$, for $t \in [-a, a]$. The asymptotic distribution of the statistic $\hat{T}^k(t)$ is a Gaussian process with 0 mean and covariance matrix

$$\mathrm{Cov}\left(\hat{T}^k(t), \hat{T}^k(s)\right) \approx \mathrm{Cov}\left(\mathcal{T}_k(t), \mathcal{T}_k(s)\right) = a_k(t)'\,\mathrm{Cov}\left(\mathcal{Z}_k(s), \mathcal{Z}_k(t)\right) a_k(s). \tag{2.6}$$

Thus, the standardized statistic $\tilde{T}^k(t) = \dfrac{\hat{T}^k(t)}{\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}}$ approaches a Gaussian process $\dfrac{\mathcal{T}_k(t)}{\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}}$ with 0 mean and variance of 1, and covariance matrix defined by the covariance function

$$\mathrm{Cov}\left(\tilde{T}^k(t), \tilde{T}^k(s)\right) = \frac{\mathrm{Cov}\left(\mathcal{T}_k(t), \mathcal{T}_k(s)\right)}{\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}\sqrt{\mathrm{Var}(\mathcal{T}_k(s))}}.$$

Specifically, we have for $k = 3, 4$

$$\mathcal{T}_3(t) = a_3(t)'\mathcal{Z}_3(t)$$

where $a_3(t) = \left[3t - t^3, 3t^2 - 3, -3t, 1\right]' \exp\left(\frac{-t^2}{2}\right)$[1], and

$$\mathcal{T}_4(t) = a_4(t)'\mathcal{Z}_4(t)$$

where $a_4(t) = \left[t^4 - 6t^2 + 3, -4t^3 + 12t, 6t^2 - 6, -4t, 1\right]' \exp\left(\frac{-t^2}{2}\right)$.

As we will see later, we may also want to provide probability bands on the plots. For that, we will use the following result.

Lemma 2.7 (Borell-TIS inequality for Gaussian process [42])**.** Let $\{f_t\}_{t\in T}$ be a zero mean Gaussian process on a topological space T. Let $\sigma_T^2 = \sup_{t\in T} \mathbb{E}|f_t|^2$, then for each $u > 0$,

$$P\left(\sup_{t\in T}|f_t| > \mathbb{E}\left(\sup_{t\in T}|f_t|\right) + u\right) \leq \exp\left(\frac{-u^2}{2\sigma_T^2}\right).$$

In our context, Lemma 2.7 implies that using the standardized process $\tilde{T}^k(t) = \dfrac{T^k(t)}{\sqrt{\mathrm{Var}\left(\mathcal{T}_k(t)\right)}}$ whose variance is 1, we can derive the $1-\alpha$ confidence region for $\hat{T}^k(t)$ when data are from normal distribution. More specifically, we have,

$$P\left(\sup_{t\in[-a,a]}|\tilde{T}^k(t)| \leq \mathbb{E}\left(\sup_{t\in-[a,a]}\left|\tilde{T}^k(t)\right|\right) + u\right) \geq 1 - \exp\left(\frac{-u^2}{2}\right)$$

Using above result with an appropriate choice of u such that $e^{-u^2/2} \leq \alpha$, a $1-\alpha$ confidence region for $\sup_{t\in[-a,a]}\left|\hat{T}^k(t)\right|$ can be easily derived as follow.

We have

$$\sup_{[-a,a]}\left|\tilde{T}^k(t)\right| \leq \mathbb{E}\left(\sup_{t\in-[a,a]}\left|\tilde{T}^k(t)\right|\right) + u_\alpha$$

$$\Rightarrow \left|\tilde{T}^k(t)\right| \leq \mathbb{E}\left(\sup_{t\in-[a,a]}\left|\tilde{T}^k(t)\right|\right) + u_\alpha, \forall t \in [-a,a]$$

[1]There is a typo in Ghosh's work where she reports $a_{31}(t) = -3\dfrac{\mu^{(2)}(t) - \left(\mu(t)\right)^2}{\left(\mu(t)\right)^3}$ while in fact $a_{31}(t) = -3\dfrac{\mu^{(2)}(t) - 2\left(\mu(t)\right)^2}{\left(\mu(t)\right)^3}$. It is difficult to say if this error also has an effect on her plot, which she presented in her paper.

which then implies that

$$-\left[\mathbb{E}\left(\sup_{t\in-[a,a]}\left|\tilde{T}^k(t)\right|\right)+u\right]\leq\tilde{T}^k(t)\leq\left[\mathbb{E}\left(\sup_{t\in[-a,a]}\left|\tilde{T}^k(t)\right|\right)+u\right],\forall t\in[-a,a]$$

and hence, $\forall t\in[-a,a]$ we have

$$-\left[\mathbb{E}\left(\sup_{t\in[-a,a]}\left|\frac{\hat{T}^k(t)}{\sqrt{\mathrm{Var}\left(\mathcal{T}_k(t)\right)}}\right|\right)+u\right]\leq\frac{\hat{T}^k(t)}{\sqrt{\mathrm{Var}\left(\mathcal{T}_k(t)\right)}}\leq\left[\mathbb{E}\left(\sup_{t\in[-a,a]}\left|\frac{\hat{T}^k(t)}{\sqrt{\mathrm{Var}\left(\mathcal{T}_k(t)\right)}}\right|\right)+u\right].$$

Let $m_k=\mathbb{E}\left(\sup_{t\in[-a,a]}\left|\frac{\hat{T}^k(t)}{\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}}\right|\right)$. As discussed above, the statistic $\frac{\hat{T}^k(t)}{\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}}$ asymptotically approaches $\tilde{\mathcal{T}}_k(t)=\frac{\mathcal{T}_k(t)}{\sqrt{\mathrm{Var}\left(\mathcal{T}_k(t)\right)}}$. Hence, an estimator of m_k, say $\hat{m}_k$, can be obtained by simulating the Gaussian process $\tilde{\mathcal{T}}_k(t)$. The $(1-\alpha)\times100\%$ confidence bands for $\hat{T}^k(t)$ are thus $\pm(\hat{m}_k+u_\alpha)\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}, t\in[-a,a]$. A nonlinear plot of $\hat{T}^k(t)$ away from the horizontal line $y=0$, especially when it crosses the bands $\pm(\hat{m}_k+u_\alpha)\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}, t\in[-a,a]$ supports a departure from normality.

We may point out that Ghosh [39] derived the critical bands for $\hat{T}^3(t)$ in the case $k=3$ as

$$\hat{T}^3(t)\pm\left(z_\alpha+1.38-\frac{5.92}{\sqrt{n}}+\frac{12.8}{n}\right)/\sqrt{K_3(t,t)}$$

where $K_3(t,t)=\sum_{j=1}^{3}\sum_{l=1}^{3}a_{3j}a_{3l}K_{jl}(t,t)$ is the variance of $\hat{T}^3(t)$ and a_{3j} is the j^{th} element of the vector $a_3(t)$ defined earlier. She had empirically obtained the second expression by finding the upper and lower bands as a function of sample size n.

We will now concentrate only on the case $k=3$ or 4, and thus, on T_3 and T_4 plots (which will plot the quantities $\hat{T}^3(t)$ and $\hat{T}^4(t)$, respectively), where we will use our newly developed probability bands $\pm(\hat{m}_k+u_\alpha)\sqrt{\mathrm{Var}(\mathcal{T}_k(t))}, t\in[-a,a]$.

Rule of thumb. Evidence of non-normality is curves crossing the probability bands or curves with high gradients.

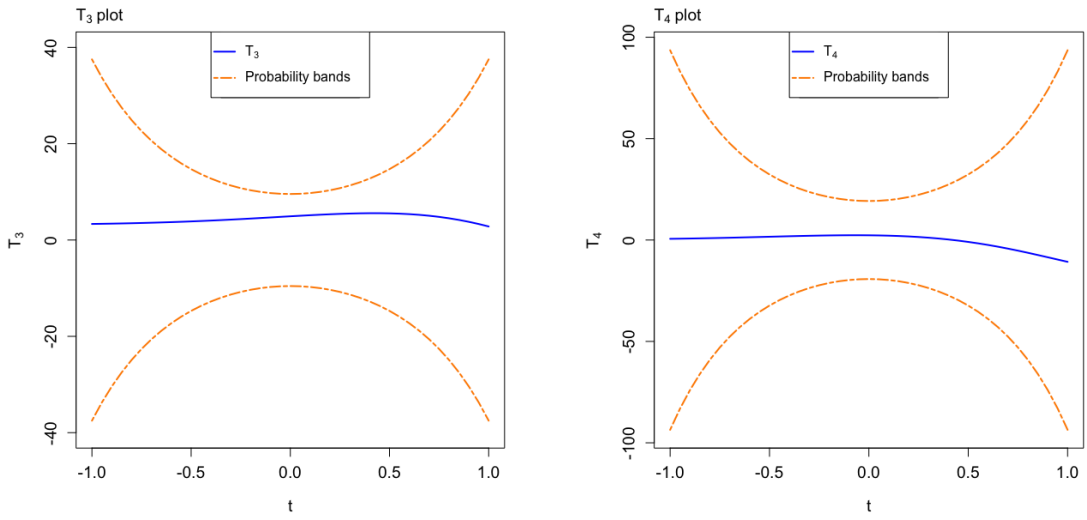


Figure 2.5: T_3 and T_4 plots for sample from $N(0,1)$, $n = 200$.

In Figure 2.5, we have plotted $\hat{T}^3(t)$ and $\hat{T}^4(t)$ using our approach along with the corresponding 95% confidence bounds for a simulated data of size $n = 200$ from $N(0,1)$. Clearly, as data follow $N(0,1)$, the statistics curves $\hat{T}^k(t)$, $k = 3, 4$ are in 95% confidence region and also have nearly zero gradient.

Figure 2.6 presents T_3 plots for the random sample of size $n = 200$ from certain non-normal populations. T_4 plots for the same six cases are presented in Figure 2.7. As a reference for comparison, we have also included in each of the plots the $\hat{T}^3$ and $\hat{T}^4$ curves for normally distributed data (shown by the dotted curve).

It is clear in Figure 2.6 that $\hat{T}^3$ curves are not only highly non-linear in each case, they also violate the probability bands in most cases. Despite the symmetry of t-distribution in Figure 2.6a and of Power Exponential family with $\beta = 1/2$ in Figure 2.6c, we observe a dramatic non-linearity in each case. Figure 2.6b for the case of Power Exponential family with a large value of shape parameter $\beta = 5$ has a graph of $\hat{T}^3(t)$ falling within the bands. The curve is relatively close to a flat line, thereby giving weak support for nonnormality. Cauchy distribution is symmetric, but its moment generating

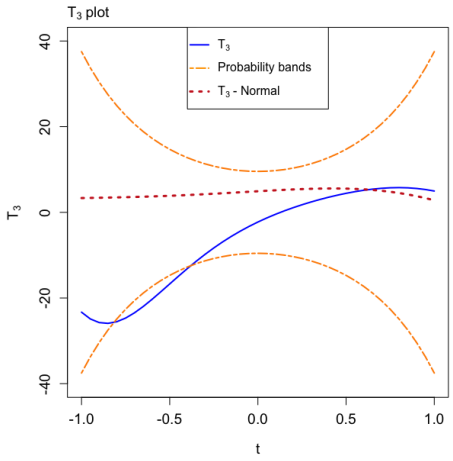
function does not exist. Accordingly, T_3 plot in Figure 2.6d has excessive large value for $\hat{T}_3(.)$ function (notice that even the vertical axis goes from -1000 to 1000). The exponential distribution is highly skewed, and this fact shows up on the corresponding T_3 plot (Figure 2.6f), where just like the Cauchy case, the non-linearity is so strong that the vertical axis had to be considerably shrunk. Finally, although the $\hat{T}^3(t)$ curve of the sample from mixture normal distribution lies within the two 95% probability bands, its non-zero gradient exposes the asymmetry and hence non-normality.

T_4 plots based on the fourth derivative of the cumulant generating function are presented in Figure 2.7. We also see the effects of non-linearity much more pronounced. One can rightly expect that, since unlike T_3 plots, T_4 plots, being a function of up to fourth derivatives, should be able to also account for differences in the kurtosis. Clearly, while t-distribution and power exponential distribution are symmetric, they have very different kurtosis compared to normal distribution. The kurtosis of exponential distribution is quite different ($= 9 - 3 = 6$). This effect of kurtosis is clearly exhibited in the highly non-linear $\hat{T}^4(t)$ curves, which in Figures 2.7a, 2.7d and 2.7f also cross the bands. Again, the exponential and Cauchy cases show a dramatic high and low (note the different scaling of the vertical axis). Moreover, unlike T_3 plots, T_4 plots of power exponential family in Figure 2.7b and mixture normal in Figure 2.7e are curved and always or mostly negative. Plots for larger sample sizes also cross the probability bands.

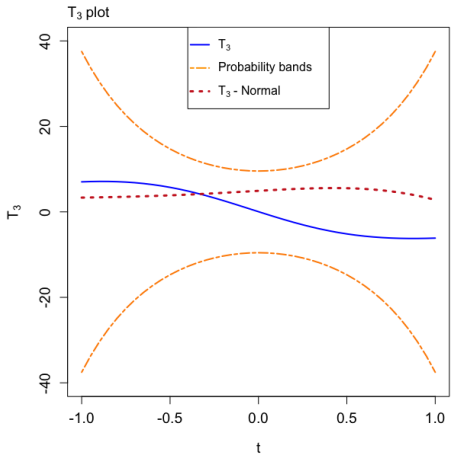
2.3.1 Two Extreme Pathological Examples

McCullagh [43] pointed out that although the moment generating function mathematically characterizes a distribution, it may not do so numerically. He considers the following density function

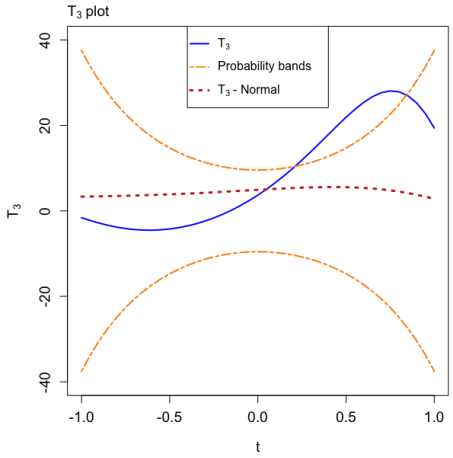
$$f(y) = \phi(y)\left[1 + \frac{1}{2}\sin 2\pi y\right], y \in (-\infty, \infty),$$



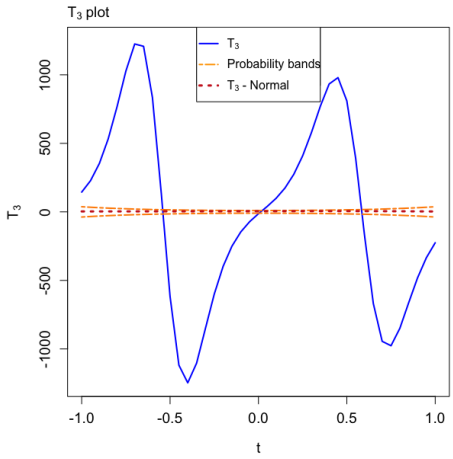
(a) $t(15)$



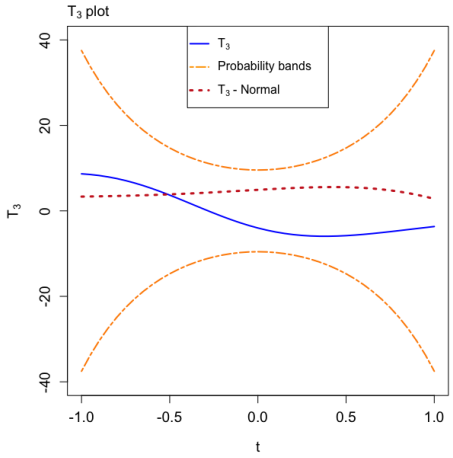
(b) Standard power exponential family with $\beta = 5$



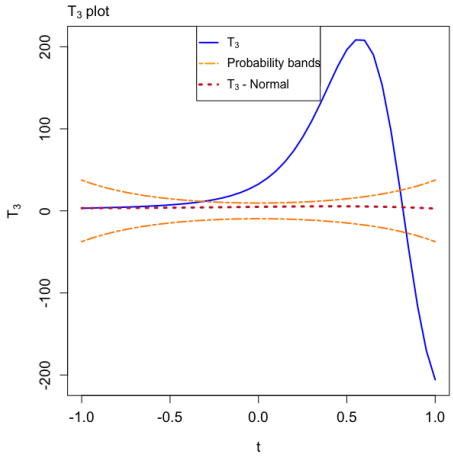
(c) Standard power exponential family with $\beta = 1/2$



(d) Cauchy$(y_0 = 0, \gamma = 1)$

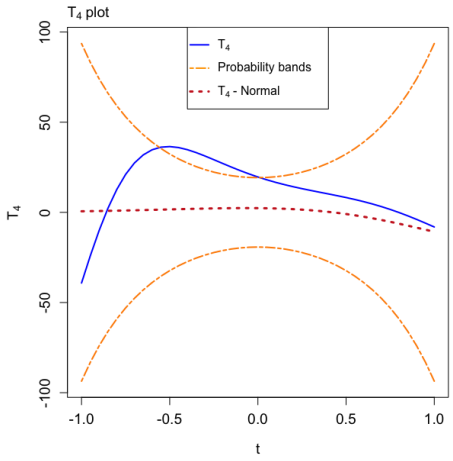


(e) Mixture normal: $.3N(\mu = 0, 1) + .7N(4, 3)$

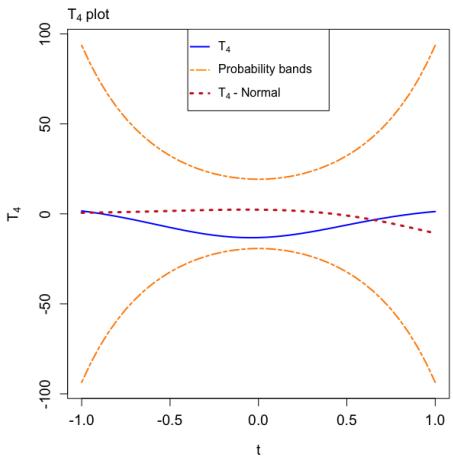


(f) Exp$(\lambda = 1)$

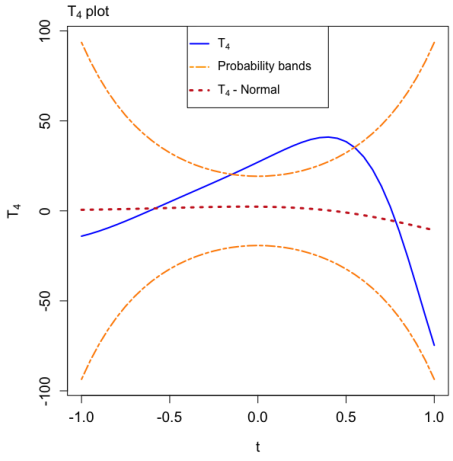
Figure 2.6: T_3 plot for certain non-normal distributions, $n = 200$.



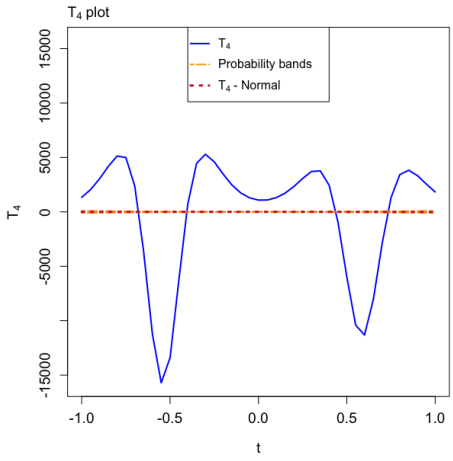
(a) $t(15)$



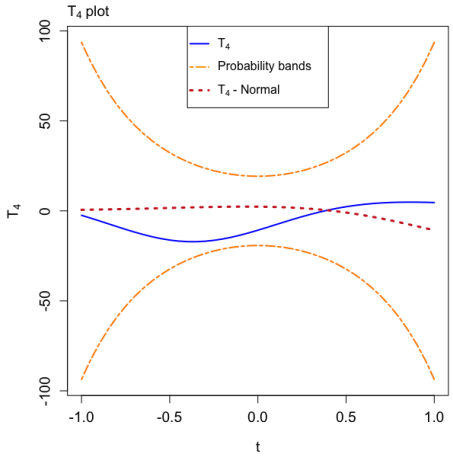
(b) Standard power exponential family with $\beta = 5$



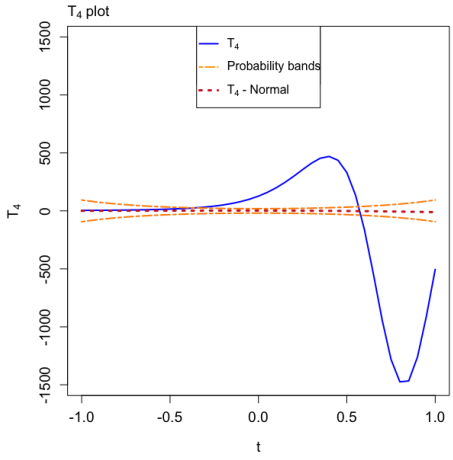
(c) Standard power exponential family with $\beta = 1/2$



(d) Cauchy$(y_0 = 0, \gamma = 1)$

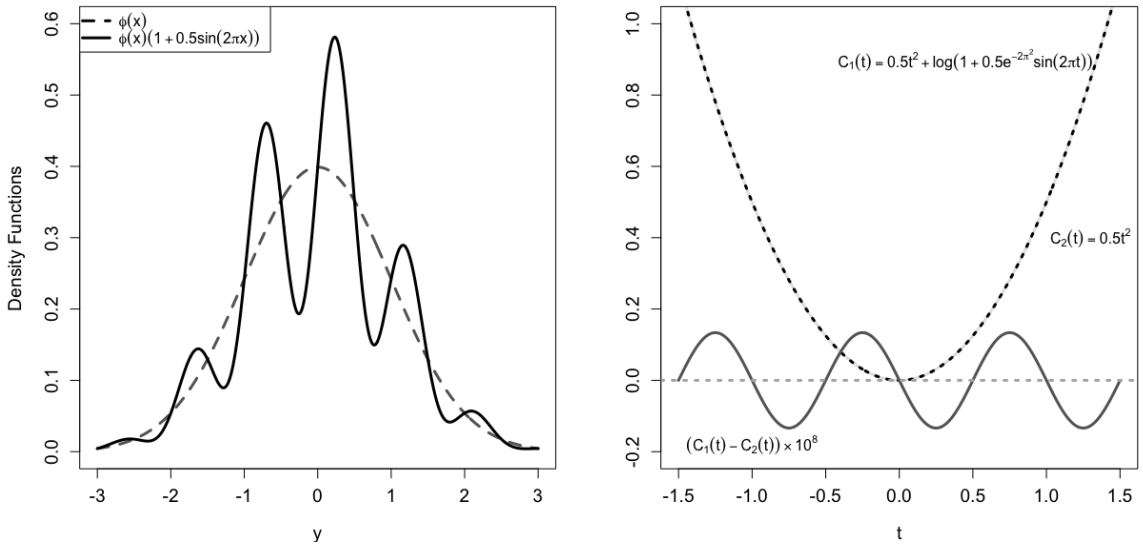


(e) Mixture normal: $.3N(\mu = 0, 1) + .7N(4, 3)$



(f) Exp$(\lambda = 1)$

Figure 2.7: T_4 plot for certain non-normal distributions, $n = 200$.



(a) (b)

Figure 2.8: Density and cumulant generating functions of McCullagh distribution.

where $\phi(y) = \frac{1}{2}\exp\left(\frac{-1}{2}y^2\right)$ is the density function of $N(0,1)$. It is straightforward to derive the cumulant generating function of the above as

$$C(t) = \frac{t^2}{2} + \log\left[1 + \frac{1}{2}\exp(-2\pi^2)\sin(2\pi t)\right], t \in (-\infty, \infty)\,.$$

We will tentatively call the above distribution the McCullagh distribution. As shown in Figure 2.8a, the McCullagh density function is visibly distinct from standard normal distribution, while the two cumulant generating functions are numerically indistinguishable in Figure 2.8b, with the maximum difference being $.2 \times 10^{-8}$. Hence, we admit that relying on the cumulant generating function (or the moment generating function) may sometimes fail to exhibit evidence of non-normality. This is also shown as an example in Figure 2.9, where data are generated from McCullagh distribution with the sample size $n = 500$.

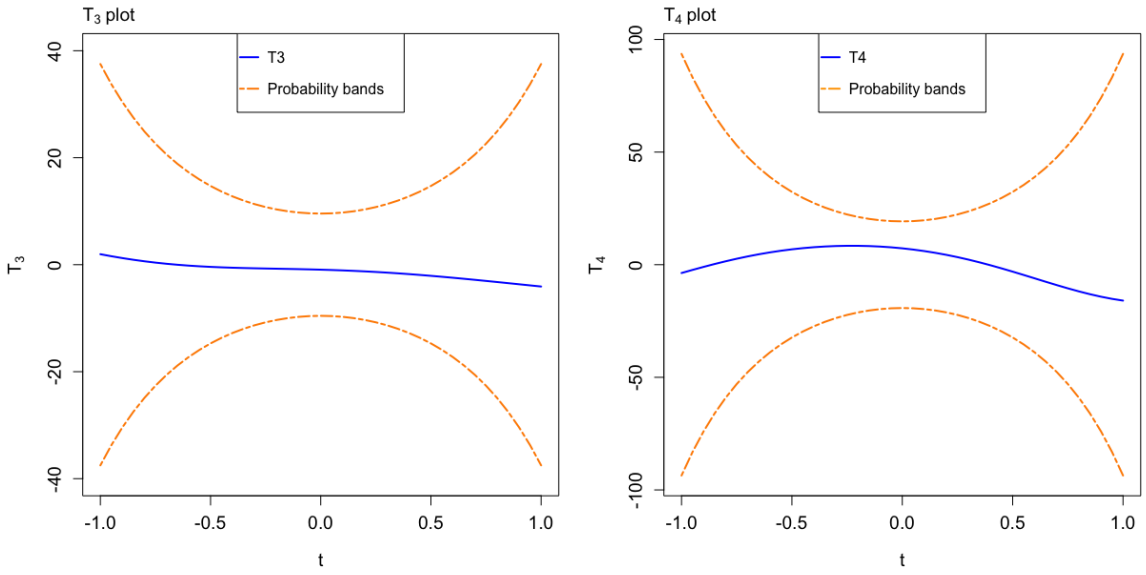
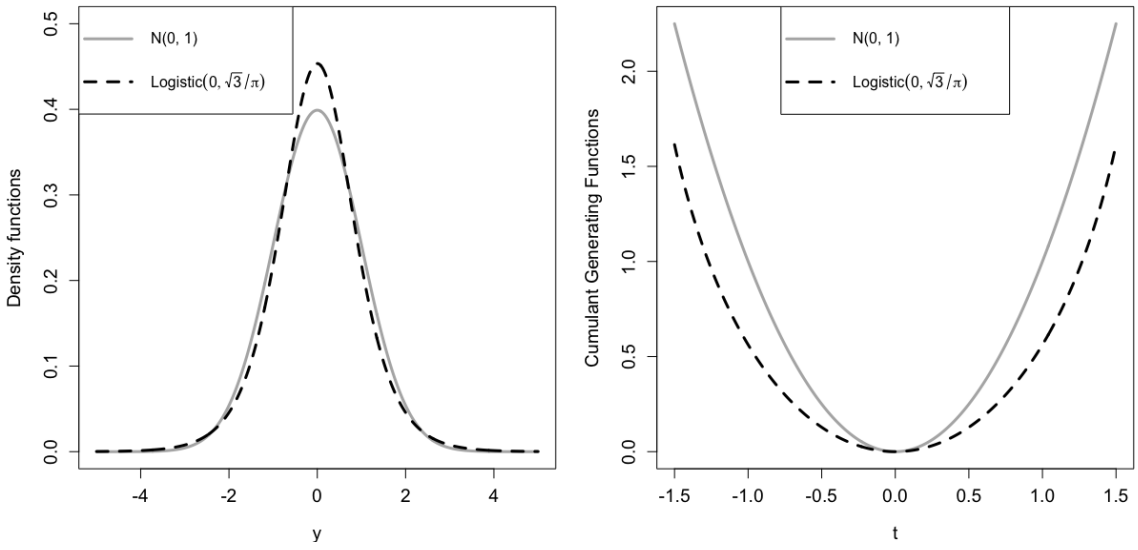


Figure 2.9: T_3 and T_4 plots for a sample generated from McCullagh distribution, $n = 500$.

On the other hand, other extreme is also possible. Specifically, in the scenario that the sample is from a symmetric but nonnormal distribution, T_3 and T_4 plots may be able to distinguish between the two populations. For this, we consider a Logistic distribution with location and scale parameters being $\mu = 0$ and $s = \sqrt{3}/\pi$ respectively. The absolute difference between the cumulative distribution functions of Logistic $\left(\mu = 0, s = \sqrt{3}/\pi\right)$ and $N(0, 1)$ ranges from -0.022 to 0.022 [44]. Despite the similarity in shape between the two density functions as shown in Figure 2.10a, tails are subtly different for the two. Especially, the infection points of $N(0, 1)$ are at ± 1 but those of Logistic $\left(\mu = 0, s = \sqrt{3}/\pi\right)$ distribution are at $\pm\sqrt{3}/\pi \log(2 + \sqrt{3}) \approx \pm.53$. The cumulant generating function of Logistic $(\mu = 0, s = \sqrt{3}/\pi)$ distribution is

$$C(t) = \log\left[B\left(1 - \frac{\sqrt{3}}{\pi}t, 1 + \frac{\sqrt{3}}{\pi}t\right)\right], t \in \left(-\frac{\pi}{\sqrt{3}}, \frac{\pi}{\sqrt{3}}\right).$$

It is shown in Figure 2.10b, also with the cumulant generating function of $N(0, 1)$, $t^2/2$. Departure between the two curves for values of t away from zero is self-evident. This



(a) (b)

Figure 2.10: Density and cumulant generating functions of Logistic $\left(\mu = 0, s = \sqrt{3}/\pi\right)$.

shows the other extreme when other graphical tests may fail to distinguish Logistic distribution from $N(0, 1)$, but cumulant generating function-based methods will be able to distinguish the two. For T_3 and T_4 plots, this is vividly seen in Figure 2.11.

2.4 Assessing the Distributional Assumption: Any Continuous Distribution

So far, our discussion has pertained to the assessment of the normality of a given dataset. The natural question is: how does one assess if a random sample comes from a specific family of distribution, albeit with unknown parameters? While there are certain special procedures for specific probability distributions, not much work has been done to address this general problem. The Q-Q plot is one of the graphical tools that has been used in certain commercial software (such as SAS and R) to assess only certain specific distributional assumptions. However, they usually require the knowledge of the parameter values that must be specified to compute the quantiles. In this section, we present a general approach based on a simple, well-known result about cumulative distribution function. We

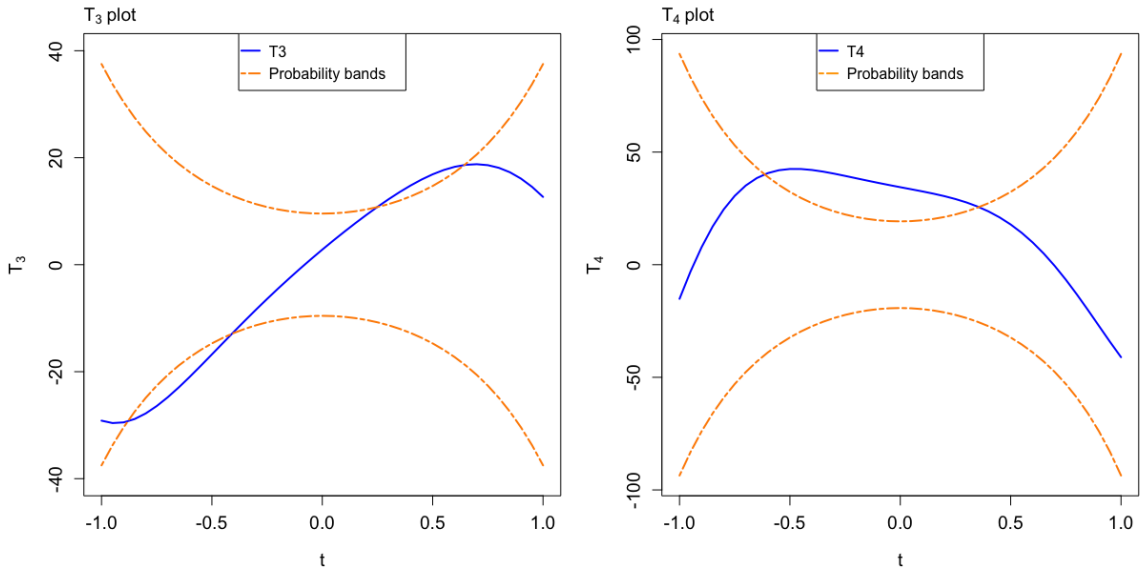


Figure 2.11: T_3 and T_4 plots for a sample generated from Logistic $\left(\mu = 0, s = \sqrt{3}/\pi\right)$, $n = 500$.

state the result below for the continuous distributions, although it can be suitably modified for the discrete probability distributions as well.

Lemma 2.8 (Probability integral transformation, Casella and Berger [45])**.** Let a continuous random variable Y have the cdf $F_Y(y)$. Define a random variable $U = F_Y(Y)$. Then U is uniformly distributed on $(0, 1)$, i.e, $U \sim U(0, 1)$.

Note that, since the cdf $F_Y(.)$ is assumed to be continuous, it admits its inverse. Thus, any uniform random variable U can be transformed to a random variable W with a cumulative probability distribution function $G_W(w)$ via $W = G_W^{-1}(U)$. Suppose $Y_1, \ldots Y_n$ is a random sample from a continuous univariate distribution with cdf $F_Y(y)$. In view of the above result, we can transform the data to a new sample $\tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_n$ with cdf $G_{\tilde{Y}}$ by

$$Y \sim F_Y(y) \xrightarrow{F_Y} U \xrightarrow{G^{-1}} \tilde{Y} \sim G_{\tilde{Y}}(\tilde{y}). \tag{2.7}$$

Here $\tilde{Y}$ is also assumed to be continuous and thus, $G_{\tilde{Y}}^{-1}$ exists. Clearly, when $G(.) = \Phi(.)$, the cumulative distribution function of standard normal distribution $N(0, 1)$, the sample is

transformed into a standard normal random sample. Thus, numerous available procedures for assessing normality can be used to test the normality of the transformed data.

This is the very premise of assessing the correctness of any distributional assumption for a given data set. Clearly, under the assumption of distribution $F(.)$ of our data, parameters of F may still be unknown. We thus estimate them under $F(.)$ using our sample. Let $f(.)$ be the probability density function corresponding to cdf $F(.)$. Substituting the maximum likelihood estimates of unknown parameters $\boldsymbol{\theta}$, say $\hat{\boldsymbol{\theta}}$, we have the estimated pdf $\hat{f}(.)$ and estimated cdf $\hat{F}(.)$. Now define $U_i = \hat{F}(Y_i)$ and let $\tilde{Y}_i = \Phi^{-1}(U_i) = \Phi^{-1}(\hat{F}(Y_i)), i = 1, 2, \ldots, n$. Clearly, under the assumption $Y_i \sim F(.)$, we anticipate $\tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_n$ to be nearly normal, on which many graphical procedures described in the previous sections as well as other formal testing procedures for testing normality can be implemented. If the data Y_i, $i = 1, \ldots n$ were not from $F(.)$, then clearly $\hat{F}(.)$ would not estimate the true cdf and accordingly, this effect must manifest on the samples $\tilde{Y}_1, \tilde{Y}_2 \ldots, \tilde{Y}_n$ as being samples from some non-normal distribution.

We must emphasize that it is advantageous to work with normal distribution instead of directly working with uniform distribution because methods for normal distribution are plentiful. Further, T_3 and T_4 plots have more meaning in the context of the normal distribution as derivatives of the cumulant generating function for normal distribution are zero. This is not the case for uniform distribution.

Table 2.1 lists some of the distributions that we will discuss. We will evaluate how effective our approach is in these non-normal distribution situations. Specifically, we consider Inverse Beta($\alpha = 10, \beta = 3.5$), Gamma($\alpha = 2, \beta = 2$), Inverse Gaussian($\mu = 1, \lambda = 2$) and Lomax($\beta = 6, \theta = .25$) distributions. Figure 2.12 plots the density functions of these distributions. Figures 2.13, 2.14, 2.15, 2.16 consider simulated random samples of size $n = 200$ from each of the above four distributions. In each case, we estimate the corresponding parameters and then use these to obtain

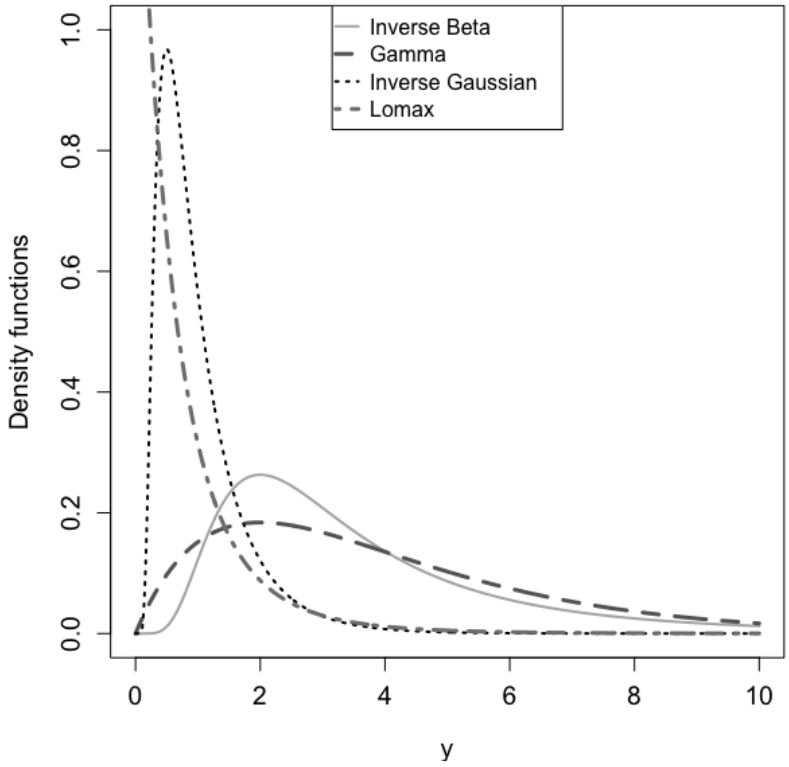


Figure 2.12: Density function of certain non-normal distributions.

estimated cdf $\hat{F}(.)$. Applying the procedure given above, we obtain the samples $\tilde{Y}_1, \tilde{Y}_2, \ldots, \tilde{Y}_n$. We assess the distributional assumptions via the Q - Q plot, score plot, T_3 plot, and T_4 plot. As shown in Figures 2.13, 2.14, 2.15, 2.16, transformed data confirms the normality, thereby also confirms to the original distribution.

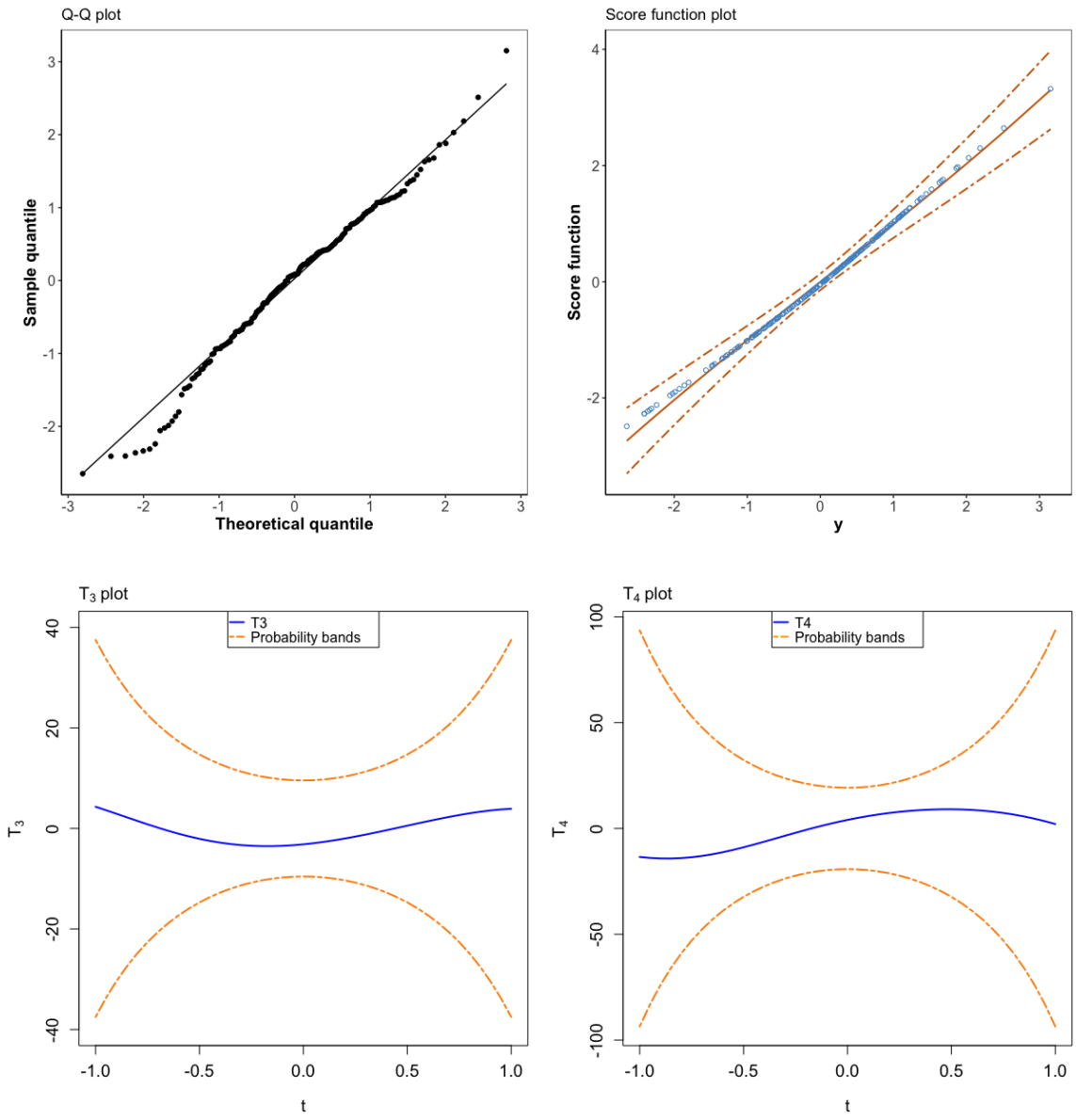


Figure 2.13: Graphical assessment for simulated data from Inverse Beta($\alpha = 10, \beta = 3.5$).

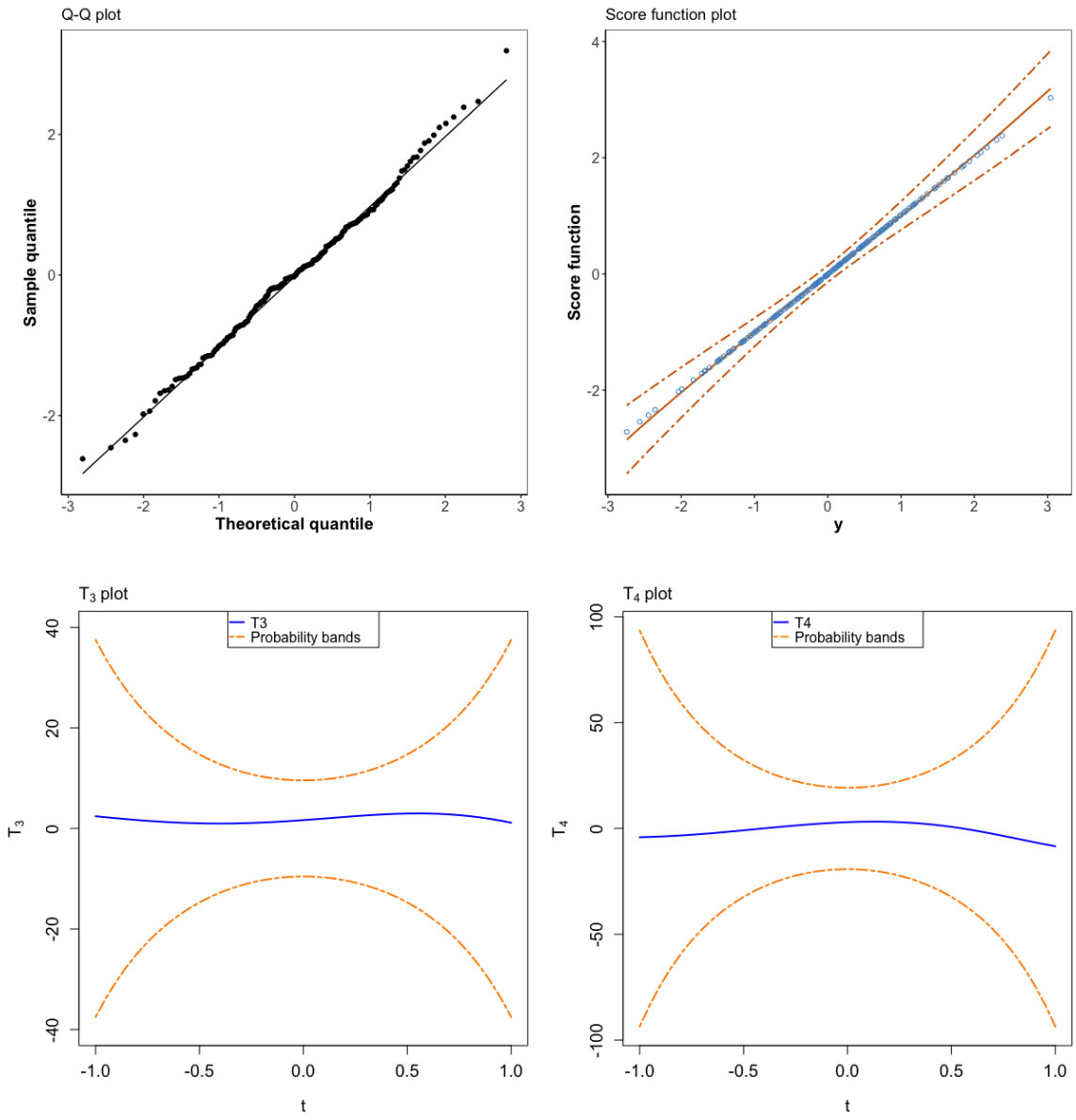


Figure 2.14: Graphical assessment for simulated data from Gamma($\alpha = 2, \beta = 2$).

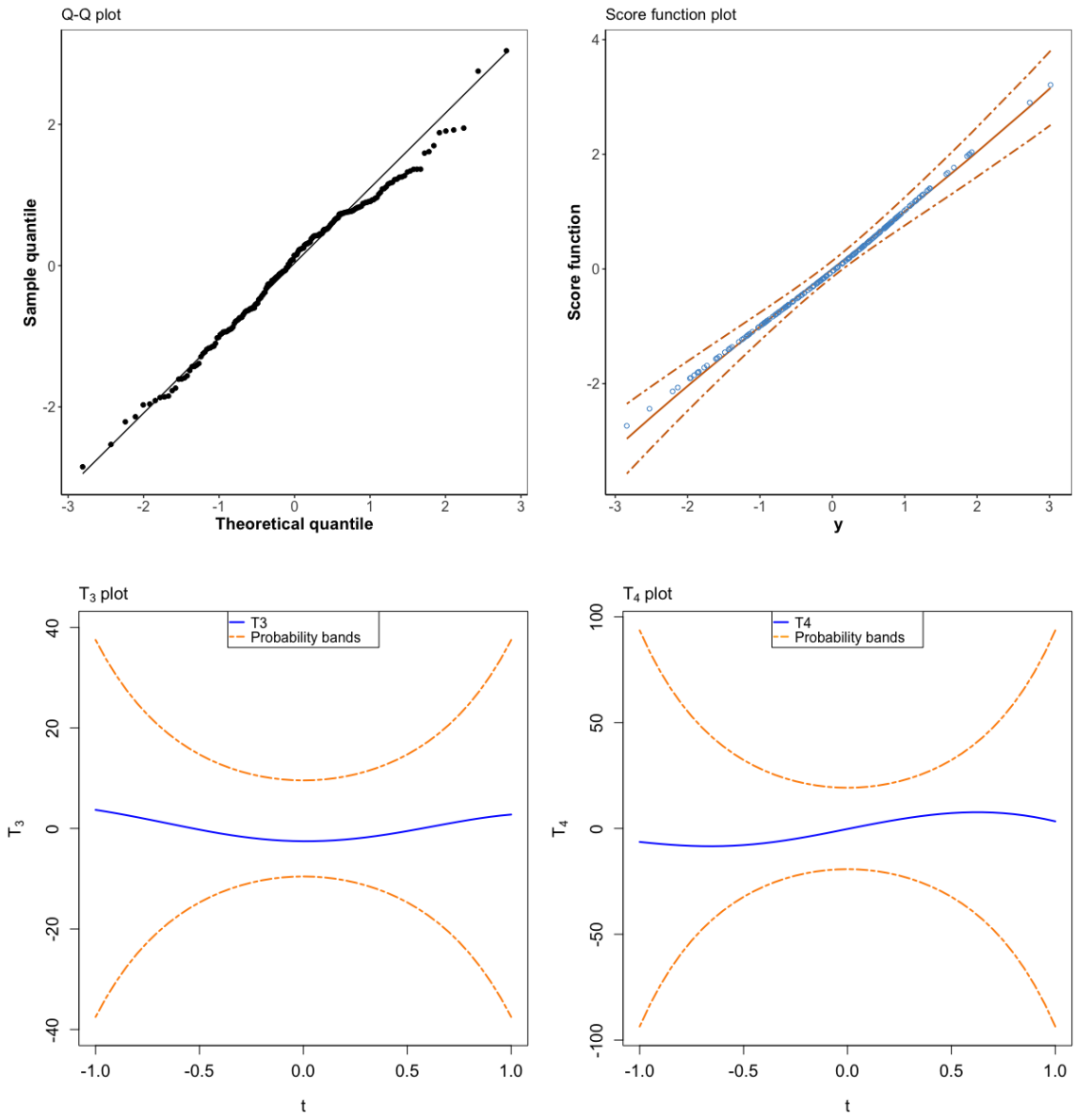


Figure 2.15: Graphical assessment for simulated data from Inverse Gaussian($\mu = 1, \lambda = 2$).

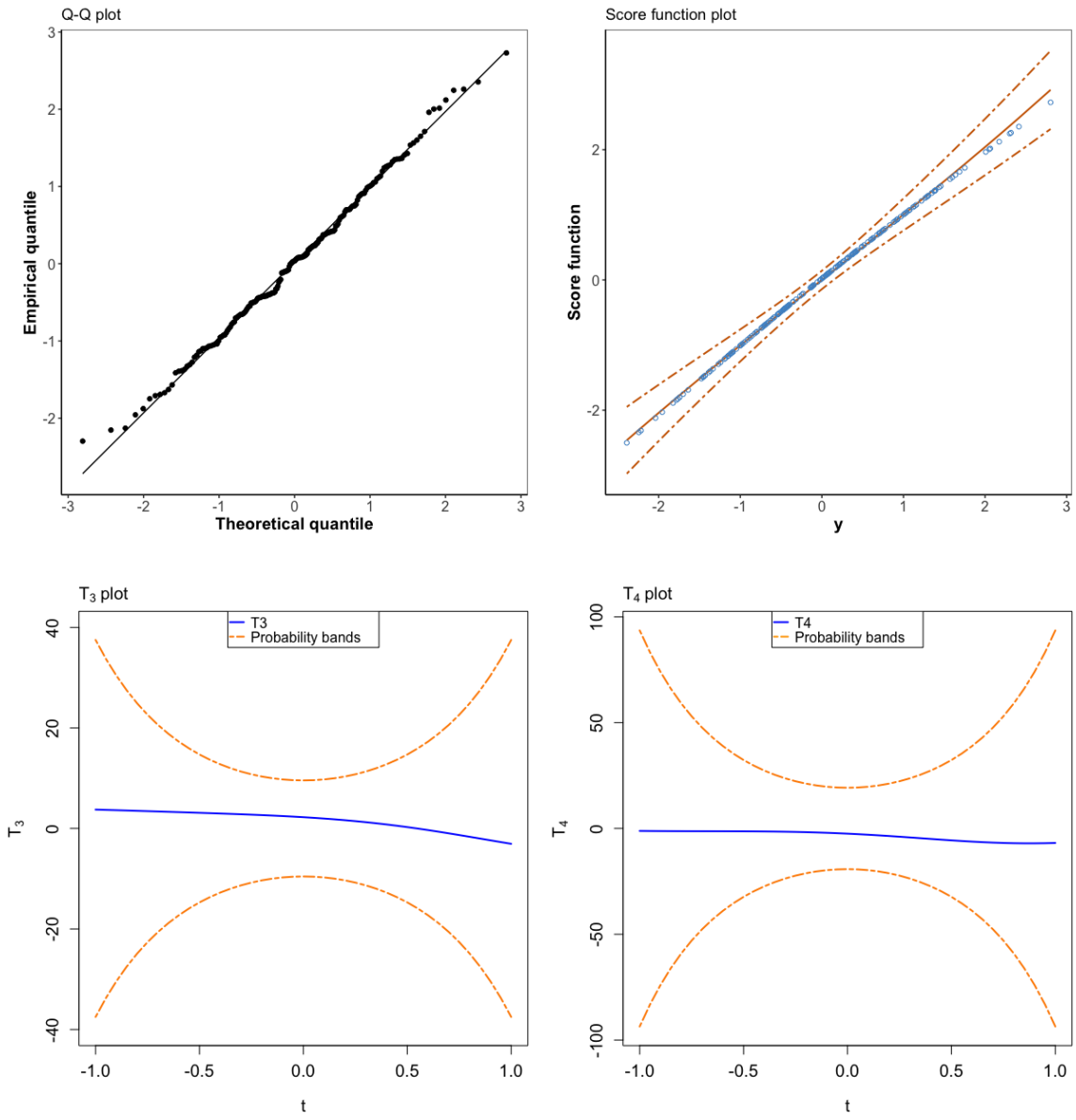


Figure 2.16: Graphical assessment for simulated data from Lomax($\beta = 6, \theta = .25$).

Table 2.1 Some continuous distributions.

Distribution	**pdf**	**cdf**
Exp(λ)	$\lambda \exp(-\lambda y)$	$1 - \exp(-\lambda y)$
Gamma(α, β)	$\frac{1}{\beta^\alpha \Gamma(\alpha)} y^{\alpha-1} \exp\left(-\frac{y}{\beta}\right)$	$\frac{1}{\Gamma(\alpha)} \gamma(\alpha, \frac{y}{\beta})$
Logistic(μ, σ)	$\frac{\exp\left(-\frac{y-\mu}{\sigma}\right)}{\sigma\left(1+\exp\left(-y-\mu/\sigma\right)\right)^2}$	$\frac{1}{1+\exp\left(-y-\mu/\sigma\right)}$
Lomax(θ, β)	$\frac{\theta\beta}{(1+\theta y)^{\beta+1}}$	$1 - (1+y\theta)^{-\beta}$
Inverse Gaussian (λ, μ)	$\left(\frac{\lambda}{2\pi y^3}\right)^{1/2} \exp\left(\frac{-\lambda(y-\mu)^2}{2\mu^2 y}\right)$	$\Phi\left[\sqrt{\frac{\lambda}{y}}\left(\frac{y}{\mu}-1\right)\right] + \exp\left(\frac{2\lambda}{\mu}\right)\Phi\left[-\sqrt{\frac{\lambda}{y}}\left(\frac{y}{\mu}+1\right)\right]$
Power Exponential (μ, σ^2, β)	$\frac{1}{\sigma\Gamma(1+\frac{1}{2\beta})2^{1+1/2\beta}} \exp\left\{\frac{-1}{2}\left[\left(y-\mu/\sigma\right)^2\right]^\beta\right\}$	Do not have closed form
Cauchy(y_0, γ)	$\frac{1}{\pi\gamma\left[1+\left(y-y_0/\gamma\right)^2\right]}$	$\frac{1}{\pi}\arctan\left(\frac{y-y_0}{\gamma}\right) + \frac{1}{2}$
Beta Prime(α, β)	$\frac{y^{\alpha-1}(1+y)^{-\alpha-\beta}}{B(\alpha,\beta)}$	$I_{y/1+y}(\alpha,\beta)$ [2]

What if the data actually came from some distributions other than the assumed one? Will our transformation approach then give the correct assessment? To check this, we generate data from each of the four distributions described above and fit the other three distributions. The previous four graphical methods, namely the Q-Q plot, score plot, T_3 plot, and T_4 plot, are used to test the validity of our fitted distribution. Do these graphs appropriately reject our (incorrect) hypotheses? If so, which ones is/are better and more sensitive to the current assessment?

Figures 2.17, 2.18, 2.19 and 2.20 present the plots for each of the four graphical methods (Q-Q plot, score plot, T_3 plot, and T_4 plot). As earlier, the four graphical methods rely on the same transformed sample $\tilde{Y}_1, \tilde{Y}_2 \ldots, \tilde{Y}_n$ for each pair of the true distribution and the fitted distribution.

To get a better idea of our interpretation, we first summarize certain relationships between various continuous distributions. First, it is well known that if a random variable $Y \sim \text{Lomax}(\theta, \beta)$, then $\beta Y \rightarrow \text{Gamma}(1, 1/\theta)$ as $\beta \rightarrow \infty$, (Lun and Khattree [46]). The Inverse Beta$(\alpha, \beta = 1)$ is essentially a reparameterization of Lomax$(\alpha, 1)$. Hence, the specified parameters $(\alpha = 10, \beta = 3.5)$ for Inverse Beta distribution and $(\beta = 6, \theta = .25)$ for Lomax distribution help avoid the connection between these two distributions. On the other hand, Inverse Gaussian distribution has a very different behavior compared to the previous three. Thus one can expect that a departure from fitted distribution will be exhibited differently in different graphs, depending on the pair of actually true and fitted distributions.

In a relatively subjective comparison, by looking at the plots in Figures 2.17, 2.18, 2.19 and 2.20, we observe that

(i) When data are from the Inverse Beta distribution, and the other three distributions are being fitted, all of the four graphical methods are able to show strong evidence to reject the null hypothesis of any of the other three distributions, except when the

score plot is used, and the Inverse Gaussian distribution is fitted. The evidence of wrongly fitted distribution is especially strong in T_3 and T_4 plots as shown by the non-linearity of the curve-statistic $\hat{T}^3(t)$ and $\hat{T}^4(t)$. Also not shown in the curves is the fact that the magnitude of these curve statistics was so large that the y-axis had to be re-scaled (compared to other graphs in the same frame) to fit the whole graphs.

(ii) When data are actually from Gamma distribution, all four approaches are able to give evidence of wrongly fitted distribution when either the Inverse Beta or the Inverse Gaussian distribution is used. When the Lomax distribution is fitted, none of the methods can indicate the departure. However, we also notice that fitting Lomax distribution into Gamma-simulated data results in a very high value of $\hat{\beta} \approx 695.94$, confirming the limiting relationship of Lomax distribution and Gamma (in fact, exponential) distribution mentioned in Lun and Khattree [46].

(iii) Score plots perform poorly for data generated from Inverse Gaussian distribution. It appears to accept all fits. On the other hand, the Q-Q plot, T_3, and T_4 plots show some evidence to indicate the incorrectness of fit when either the Gamma or the Lomax distributions are assumed. However, none of the four approaches can discriminate between the Inverse Beta distribution and Inverse Gaussian distribution.

(iv) When data are generated from Lomax distribution, wrongly fitting the Gamma or Inverse Gaussian model to Lomax data is detected by all four approaches. However, none of the four plots can discriminate between Lomax and Inverse Beta distributions.

In an overall sense, the T_3 and T_4 plots are relatively more effective in correctly identifying the actual distribution and disputing the wrong fit.

Figure 2.21 presents the estimates of the density functions when each of the four considered families is fitted to each sample. In each of the sub-figures, the solid line is the

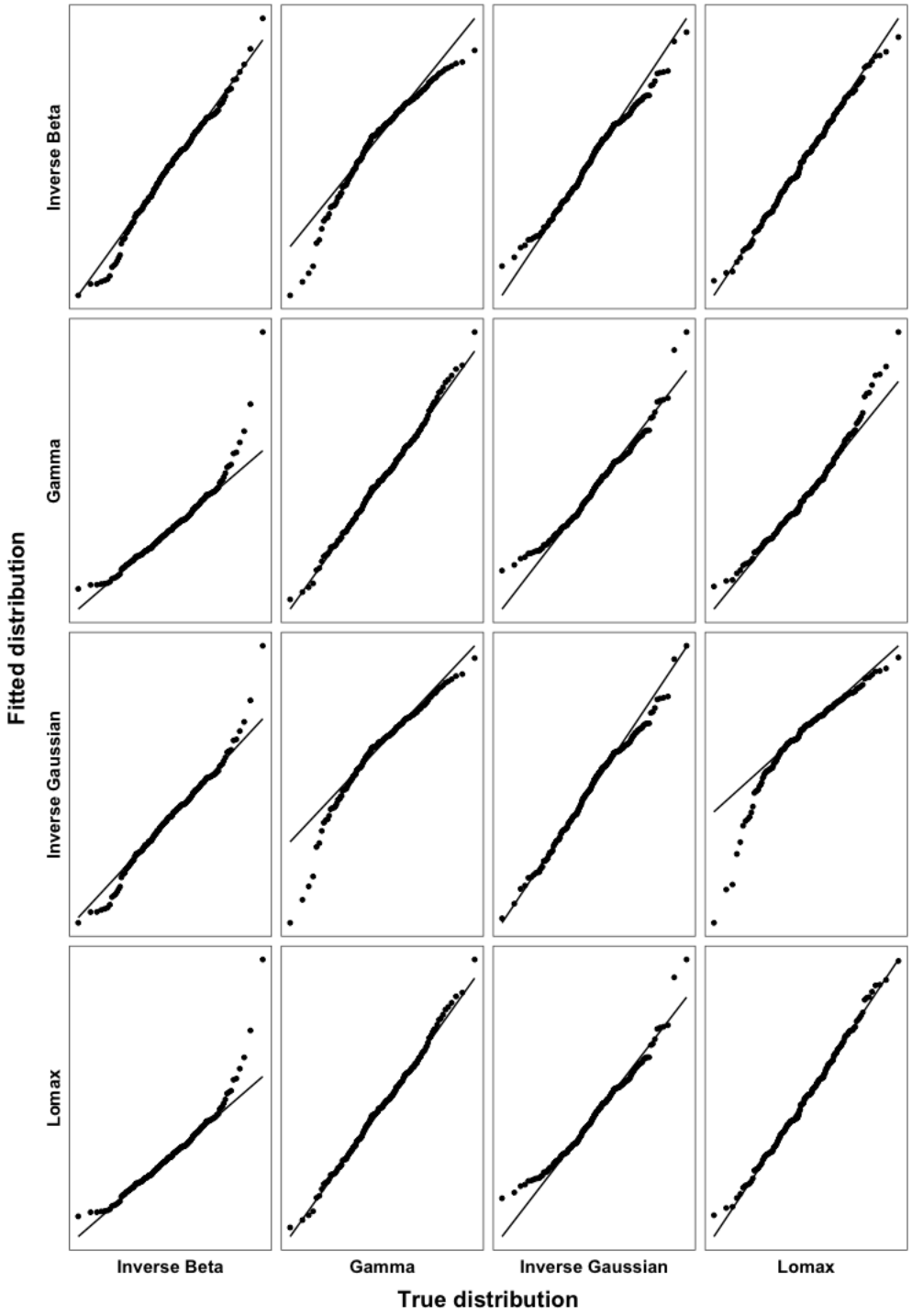


Figure 2.17: Q - Q plots when fitting simulated data to test for distributional assumption.

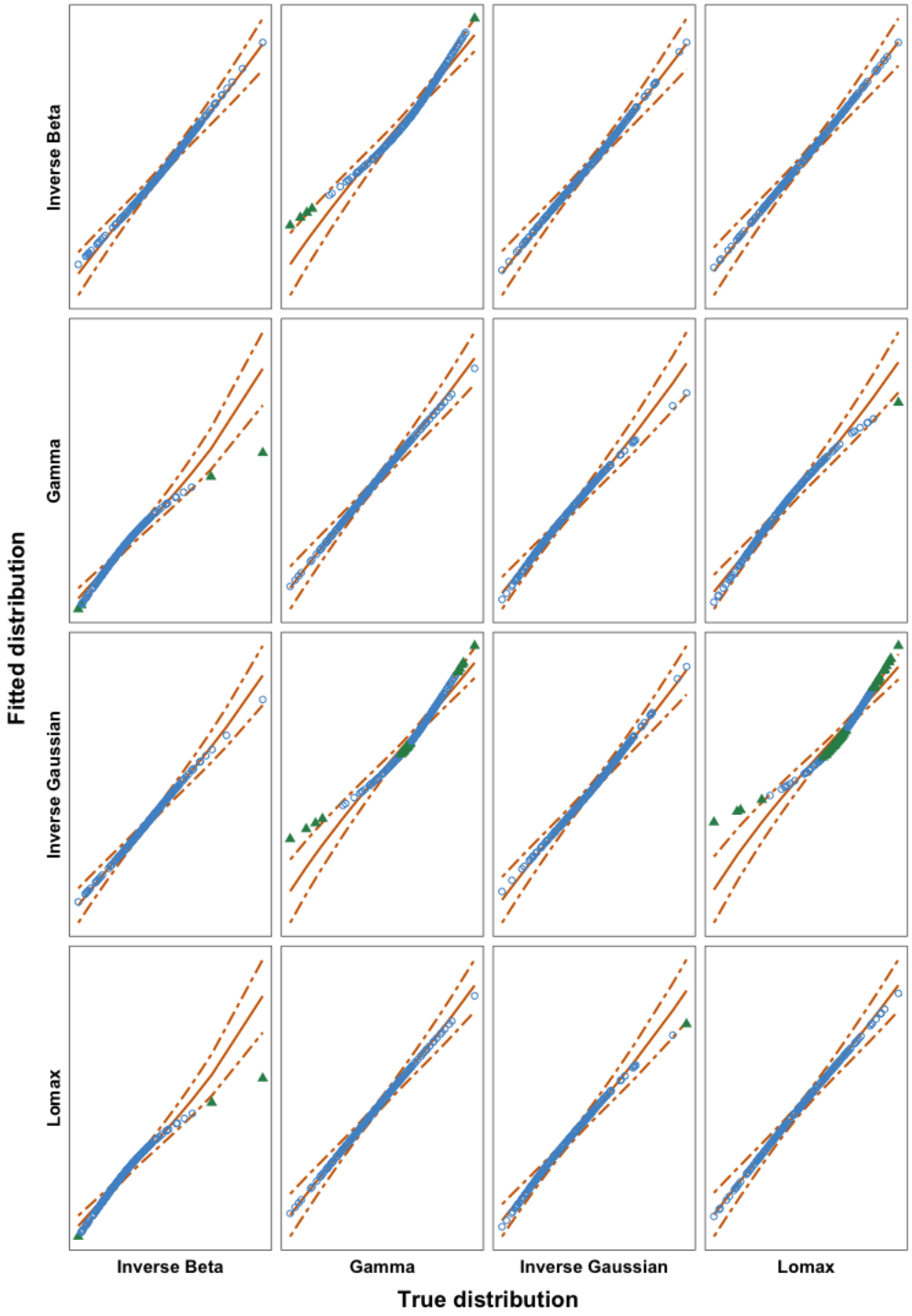


Figure 2.18: Score plots when fitting simulated data to test for distributional assumption.

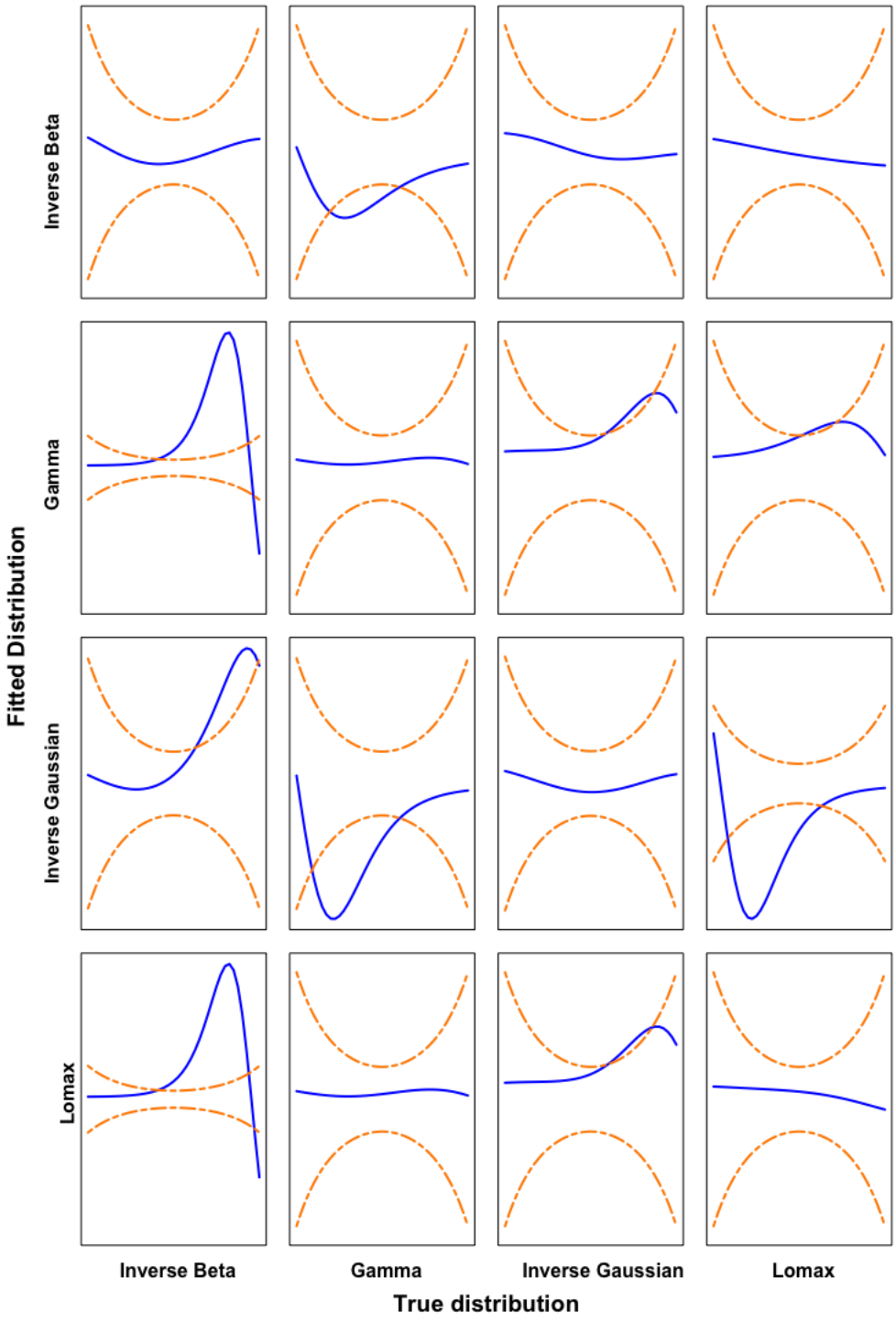


Figure 2.19: T_3 plots when fitting simulated data to test for distributional assumption.

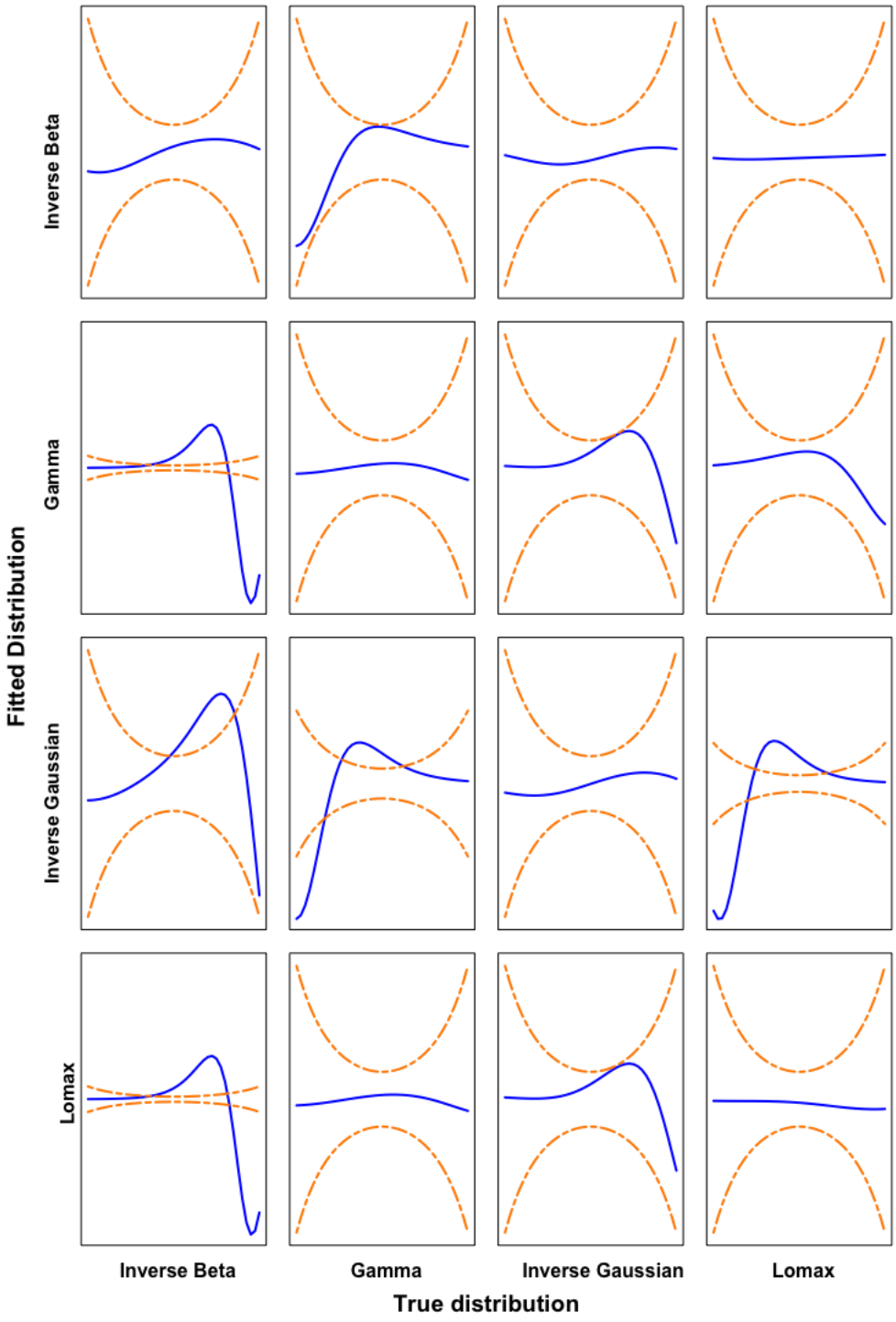


Figure 2.20: T_4 plots when fitting simulated data to test for distributional assumption.

true density (in columns) from which the sample is generated, and the dotted line is the estimated density when the particular distribution (in rows) is fitted to the data. Whenever the true density curve and the fitted density curve have a major region where they overlap, the two distributions will be difficult to distinguish from each other in terms of which distribution the data come from. And those were the cases that T_3 and T_4 curves were unable to distinguish in Figures 2.19 and 2.20.

2.5 A Quick Evaluation of Transformation Method in The Context of Formal Hypothesis Testing Methods

Here, we will examine how effective some of the formal hypothesis testing methods are when we have data from the previous four probability distributions, and we will try to fit each of the four. We will work with the transformed data $\tilde{Y}_1, \tilde{Y}_2, \ldots \tilde{Y}_n$, and we will consider the following four test statistics.

(i) Mardia's skewness test statistic [47], [48]

$$\sqrt{\hat{\kappa}_1} = \frac{\frac{1}{n}\sum_{i=1}^{n}(\tilde{Y}_i - \bar{\tilde{Y}})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(\tilde{Y}_i - \bar{\tilde{Y}})^2\right)^{3/2}} \sim N(0, 6/n) \text{ under normality.}$$

(ii) Mardia's kurtosis test statistic [47], [48]

$$\hat{\kappa}_2 = \frac{\frac{1}{n}\sum_{i=1}^{n}(\tilde{Y}_i - \bar{\tilde{Y}})^4}{\left(\frac{1}{n}\sum_{i=1}^{n}(\tilde{Y}_i - \bar{\tilde{Y}})^2\right)^{3/2}} - 3 \sim N(0, 24/n) \text{ under normality.}$$

(iii) Anderson - Darling test statistic [49], [50]

$$A^2 = -\frac{1}{n}\left\{\sum_{i=1}^{n}(2i-1)[\ln(q_i) + \ln(1 - q_{n+1-i})]\right\}$$

where $q_i = \Phi(\tilde{Y}_{(i)})$ and $\tilde{Y}_{(1)} \leq \tilde{Y}_{(2)} \leq \cdots \leq \tilde{Y}_{(n)}$ are ordered statistics. Cut-off points of A^2 are computed by Stephen [49] and Pettitt [51].

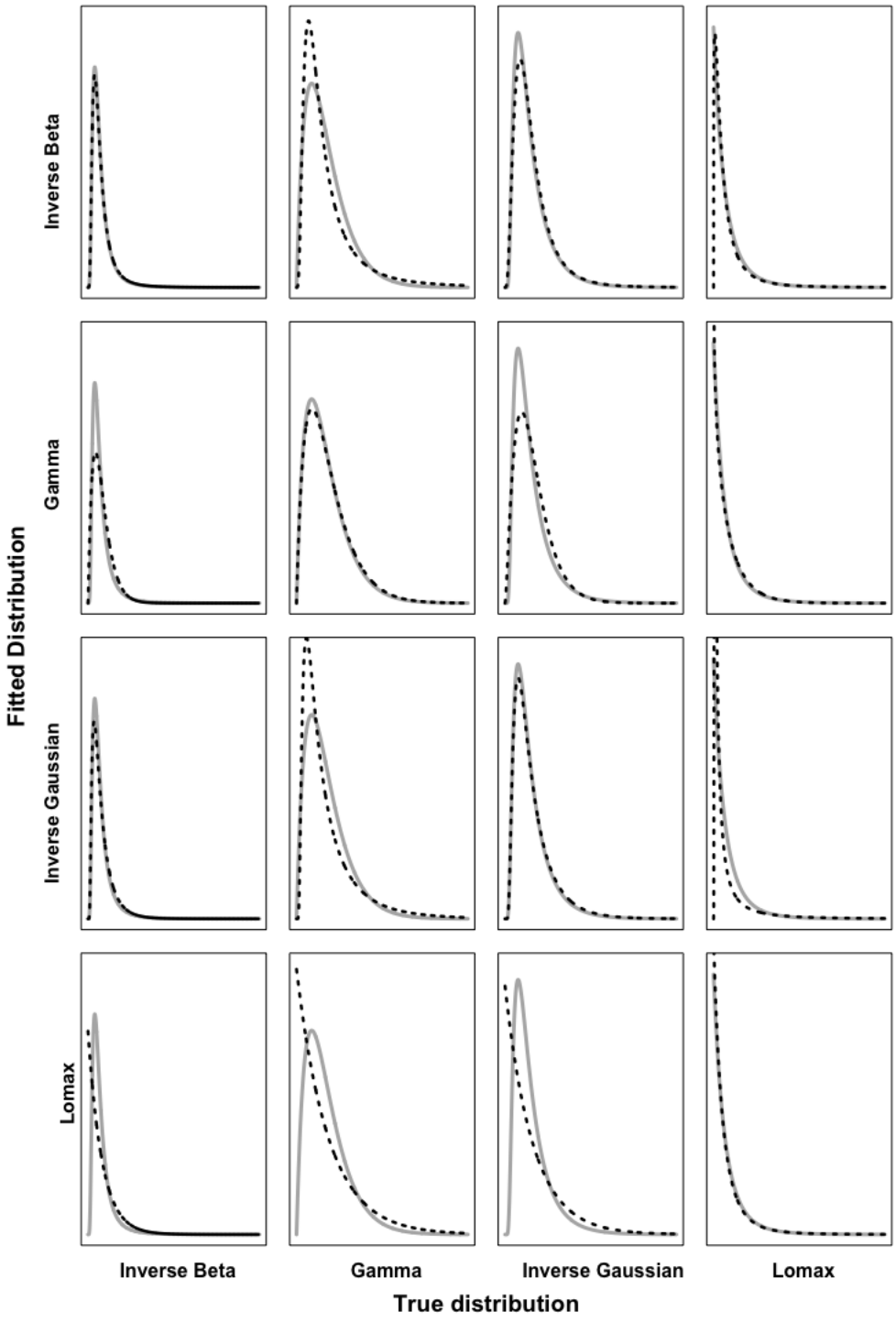


Figure 2.21: Density estimators from the given samples.

(iv) Shapiro - Wilk test statistic [52], [53]

$$W = \frac{\left(\sum_{i=1}^{n} a_i \tilde{Y}_i\right)^2}{\sum_{i=1}^{n} \left(\tilde{Y}_i - \bar{\tilde{Y}}\right)}$$

where $a' = [a_1, a_2, \ldots, a_n]' = \frac{q'V^{-1}}{(q'V^{-1}V^{-1}q)^{1/2}}$, $V = (v_{ij}) = \mathrm{Cov}\left(q_i, q_j\right)$, and q_i used here are as same as those used in the Anderson - Darling test statistic. Cut off points of W has been computed by Shapiro and Wilk [31] Royston [54] and [55].

We will compute approximately Type I error probabilities based on $m = 4000$ simulations each for the sample size $n = 100, 200, \ldots 1000$. We examine which of the tests are able to maintain the prespecified level of significance α (= 0.05 for our calculation). The estimates of Type I error probabilities are graphically presented in Figure 2.22. For each of the four distributions Inverse Beta($\alpha = 10, \beta = 3.5$), Gamma($\alpha = 2, \beta = 2$), Inverse Gaussian($\mu = 1, \lambda = 2$) and Lomax($\beta = 6, \theta = .25$), the test statistics give the estimated probability of Type I error around $\alpha = 0.05$ or smaller. In the case of Lomax distribution, Mardia's skewness test statistic has a Type I error probability near 0.

We now evaluate the powers of these tests using our transformation approach with $\alpha = 0.05$. To do so, we synthetically generate samples from each of the above distributions and fit the other three distributions to each of the data. We obtain the maximum likelihood estimators for the corresponding parameters. Each cell of Table 2.2 contains the power of the tests when data from a given distribution (in rows) and another distribution is fitted (in columns). Power values within each cell are arranged in the increasing order of the sample size ($n = 100, 200, 500$). Results show that Mardia's skewness and Shapiro Wilk's tests generally have superior powers and may be more effective with our transformation method. Admittedly, power could be low if the major part of the density of the actual and the assumed distributions overlap. As seen in Figure 2.21, that was the case for the pair of

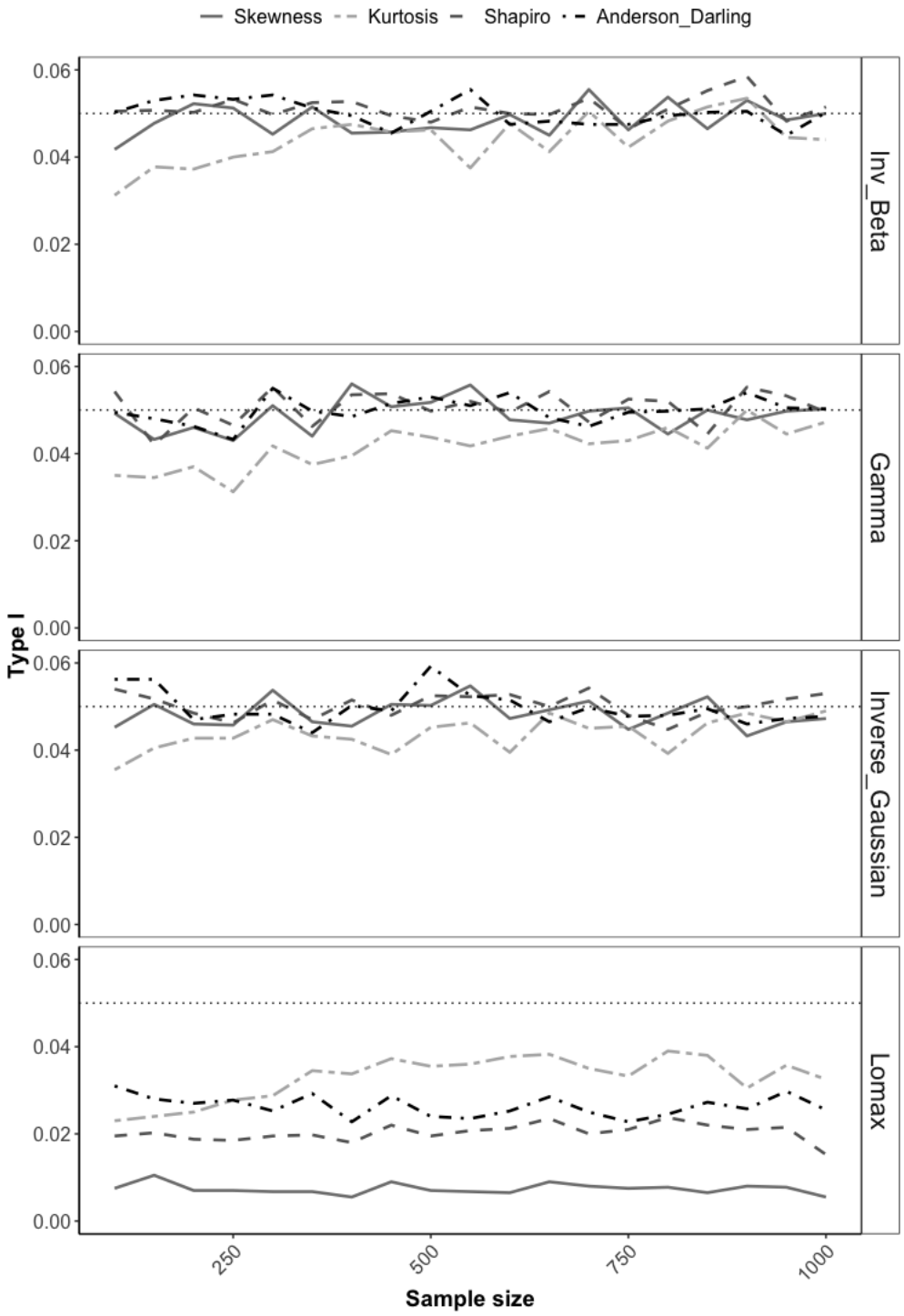


Figure 2.22: Empirical Type I error of transformation method.

Table 2.2 Empirical power of formal hypothesis tests on inverse cdf transformation.

(a) Mardia's skewness test.

True Distributions		Fitted Distribution			
		Inverse Beta	**Gamma**	**Inverse Gaussian**	**Lomax**
	Inverse Beta ($\alpha = 10, \beta = 3.5$)	0.0415 0.0490 0.0455	0.9207 0.9978 1.0000	0.2362 0.3580 0.6643	0.9165 0.9975 1.0000
	Gamma ($\alpha = 2, \beta = 2$)	0.9995 0.9992 0.9992	0.0437 0.0420 0.0480	0.9968 0.9978 0.9978	0.0473 0.0455 0.0493
	Inverse Gaussian ($\mu = 1, \lambda = 2$)	0.0563 0.0520 0.0592	0.9560 0.9553 0.9527	0.0488 0.0450 0.0515	0.9533 0.9507 0.9500
	Lomax ($\beta = 6, \theta = .25$)	0.5188 0.5102 0.5022	0.3965 0.4088 0.4090	1.0000 1.0000 1.0000	0.0080 0.0075 0.0053

Inverse Gaussian and Inverse Beta distributions, and that overlap manifests in the corresponding cells of Table 2.2 (cells (1, 3) and (3, 1), respectively).

In conclusion, we surmise that our transformation method can be used to test the distributional assumption for any continuous-valued data set, and either graphical or formal hypothesis testing methods for normality can then be applied to draw the appropriate conclusions.

2.6 Two Examples: Cork Data and Water Potability Data

Example 2.6.1 (Testing marginal normality, Cork data)**.** Table 2.3 presents the Cork Boring dataset by Rao (1948). The weights of $n = 28$ cork boring were measured in four directions: North, East, South, and West, respectively. Graphical assessments of the univariate normality of each of the four variables for this data set are shown in Figure 2.23 below. The Q - Q plots in Figure 2.23a show some moderate evidence of non-normality. Score plots as in Figure 2.23b yield nearly straight lines in each of the four North, East,

Table 2.2 Empirical power of formal hypothesis tests on inverse cdf transformation.

(b) Mardia's kurtosis test.

True Distributions		Fitted Distribution			
		Inverse Beta	**Gamma**	**Inverse Gaussian**	**Lomax**
	Inverse Beta ($\alpha = 10, \beta = 3.5$)	0.0300 0.0413 0.0478	0.6062 0.8410 0.9905	0.2597 0.4343 0.7342	0.6012 0.8345 0.9892
	Gamma ($\alpha = 2, \beta = 2$)	0.7260 0.7228 0.7290	0.0440 0.0400 0.0408	0.9193 0.9143 0.9120	0.0450 0.0413 0.0428
	Inverse Gaussian ($\mu = 1, \lambda = 2$)	0.0400 0.0370 0.0365	0.3212 0.2920 0.3115	0.0395 0.0362 0.0340	0.3142 0.2853 0.3015
	Lomax ($\beta = 6, \theta = .25$)	0.0638 0.0678 0.0512	0.2855 0.2958 0.2848	0.9988 0.9980 0.9982	0.0283 0.0320 0.0250

Table 2.2 Empirical power of formal hypothesis tests on inverse cdf transformation.

(c) Shapiro-Wilk test

True Distributions		Fitted Distribution			
		Inverse Beta	**Gamma**	**Inverse Gaussian**	**Lomax**
	Inverse Beta ($\alpha = 10, \beta = 3.5$)	0.0503 0.0535 0.0437	0.9110 0.9955 1.0000	0.2457 0.4070 0.7432	0.9070 0.9948 1.0000
	Gamma ($\alpha = 2, \beta = 2$)	0.9995 0.9995 1.0000	0.0512 0.0470 0.0495	0.9955 0.9962 0.9960	0.0520 0.0498 0.0535
	Inverse Gaussian ($\mu = 1, \lambda = 2$)	0.1467 0.1353 0.1457	0.9575 0.9557 0.9507	0.0515 0.0408 0.0500	0.9523 0.9490 0.9480
	Lomax ($\beta = 6, \theta = .25$)	0.5138 0.5002 0.4833	0.3638 0.3735 0.3578	1.0000 1.0000 1.0000	0.0205 0.0180 0.0170

Table 2.2 Empirical power of formal hypothesis tests on inverse cdf transformation.

(d) Anderson-Darling test.

True Distributions		Fitted Distribution			
		Inverse Beta	**Gamma**	**Inverse Gaussian**	**Lomax**
	Inverse Beta ($\alpha = 10, \beta = 3.5$)	0.0500	0.8400	0.1678	0.8345
		0.0537	0.9872	0.2843	0.9852
		0.0483	1.0000	0.5813	1.0000
	Gamma ($\alpha = 2, \beta = 2$)	0.9970	0.0525	0.9878	0.0532
		0.9972	0.0498	0.9888	0.0495
		0.9980	0.0467	0.9885	0.0465
	Inverse Gaussian ($\mu = 1, \lambda = 2$)	0.1375	0.8912	0.0503	0.8832
		0.1325	0.8908	0.0423	0.8800
		0.1375	0.8842	0.0490	0.8738
	Lomax ($\beta = 6, \theta = .25$)	0.4153	0.2470	1.0000	0.0262
		0.4010	0.2597	0.9998	0.0290
		0.4047	0.2482	1.0000	0.0262

Table 2.3 Weights of Cork Boring (in Centigrams) in Four Directions for 28 Trees.

Tree	N	E	S	W	Tree	N	E	S	W
1	72	66	76	77	15	91	79		75
2	60	53	66	63	16	56	68	47	50
3	56	57	64	58	17	79	65	70	61
4	41	29	36	38	18	81	80	68	58
5	32	32	35	36	19	78	55	67	60
6	30	35	34	26	20	46	38	37	38
7	39	39	31	27	21	39	35	34	37
8	42	43	31	25	22	32	30	30	32
9	37	40	31	25	23	60	50	67	54
10	33	29	27	36	24	35	37	48	39
11	32	30	34	28	25	39	36	39	31
12	63	45	74	63	26	50	34	37	40
13	54	46	60	52	27	43	37	39	50
14	47	51	52	43	28	48	54	57	43

Table 2.4 The first and last five rows of dataset Water Portability

Parts	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organiccarbon	Trihalomethanes	Turbidity	Portability
1	NA	204.8	20791.3	7.3	368.5	564.3	10.3	86.9	2.9	0
2	3.7	129.4	18630.0	6.6	NA	592.8	15.1	56.3	4.5	0
3	8.0	224.2	19909.5	9.2	NA	418.6	16.8	66.4	3.0	0
4	8.3	214.3	22018.4	8.0	356.8	363.2	18.4	.3	4.6	0
5	9.0	181.1	17978.9	6.5	310.1	398.4	11.5	31.9	4.0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3272	4.6	193.6	47580.9	7.1	359.9	526.4	13.8	66.6	4.4	1
3273	7.8	193.5	17329.8	8.0	NA	392.4	19.9	NA	2.7	1
3274	9.4	175.7	33155.5	7.3	NA	432.0	11.0	69.8	3.2	1
3275	5.1	230.6	11983.8	6.3	NA	402.8	11.1	77.4	4.7	1
3276	7.8	195.1	17404.1	7.5	NA	327.4	16.1	78.6	2.3	1

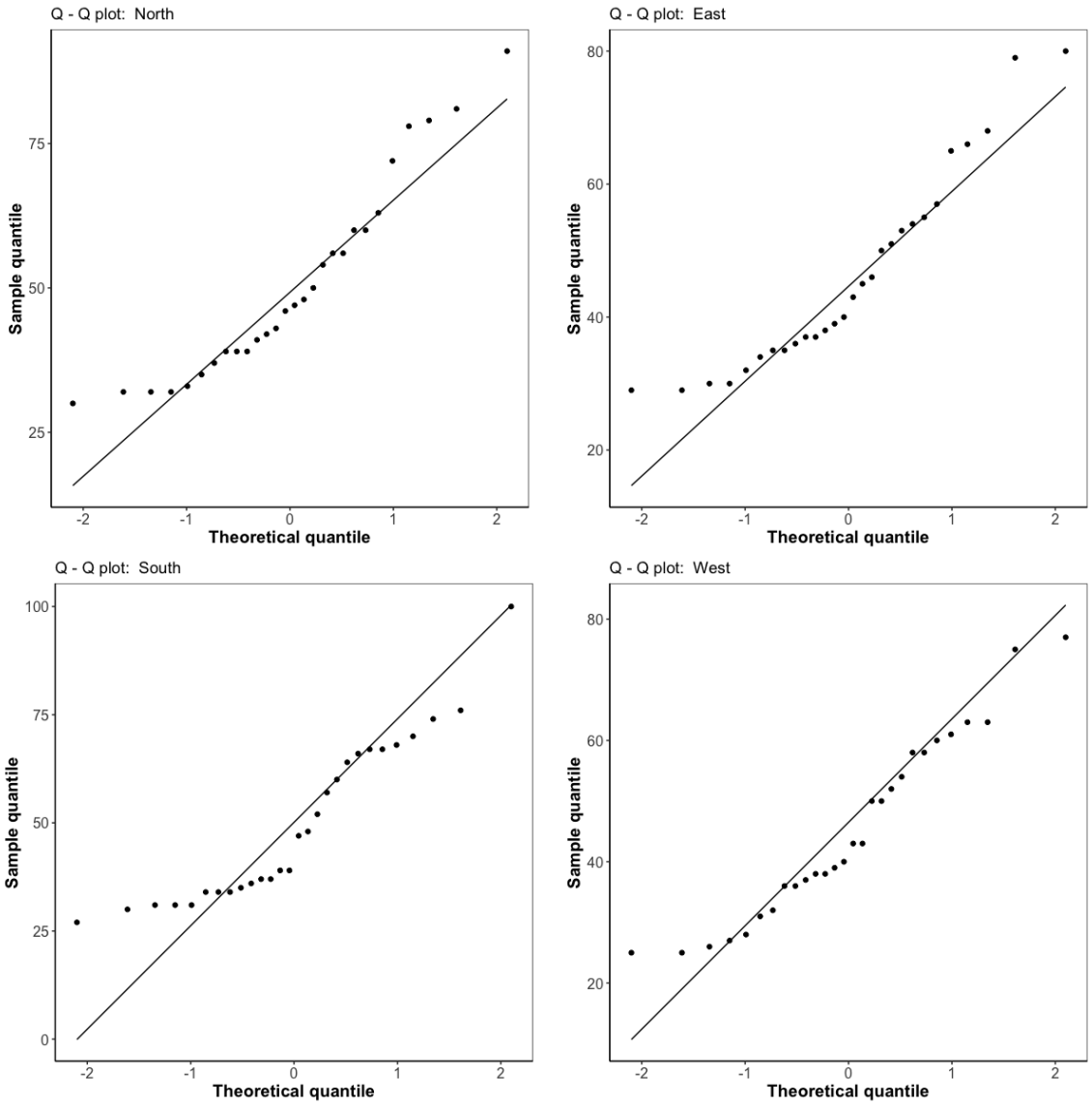
South, and West variables, thereby suggesting near-normality[3]. The T_3 and T_4 plots also imply the same conclusions as evidenced by nearly zero gradients of the corresponding curves. With a small sample size of $n = 28$, the moderate departure from normality shown in the Q - Q plot is understandable. In this case, T_3 and T_4 plots are much more informative.

Example 2.6.2 (Water Potability dataset from Kaggle[4])**.** The dataset consists of 9 metrics for the quality of 3276 different water bodies. Some representative observations are shown in Table 2.4. The last column named *Potability* indicates whether or not the water is safe for human consumption and hence is a categorical variable.

After removing various rows containing one or more missing values, the total number of remaining observations is 2011. Among these, 811 observations correspond to

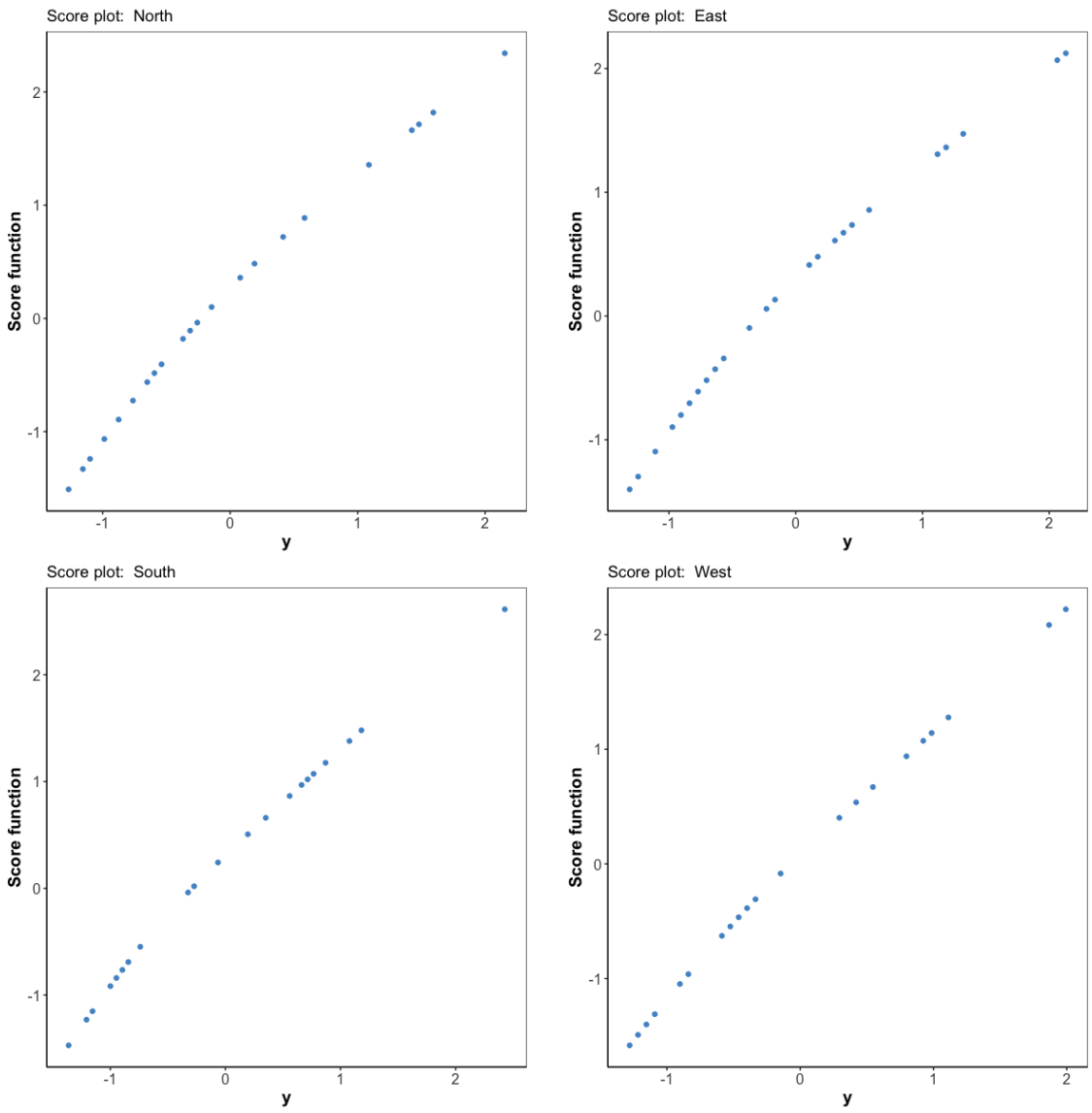
[3]When the points were too close, within a distance of 0.0001, they were merged and their weights were added together, this was done to avoid the computational problem of nearly singular matrix.

[4]https://www.kaggle.com/datasets/adityakadiwal/water-potability/data



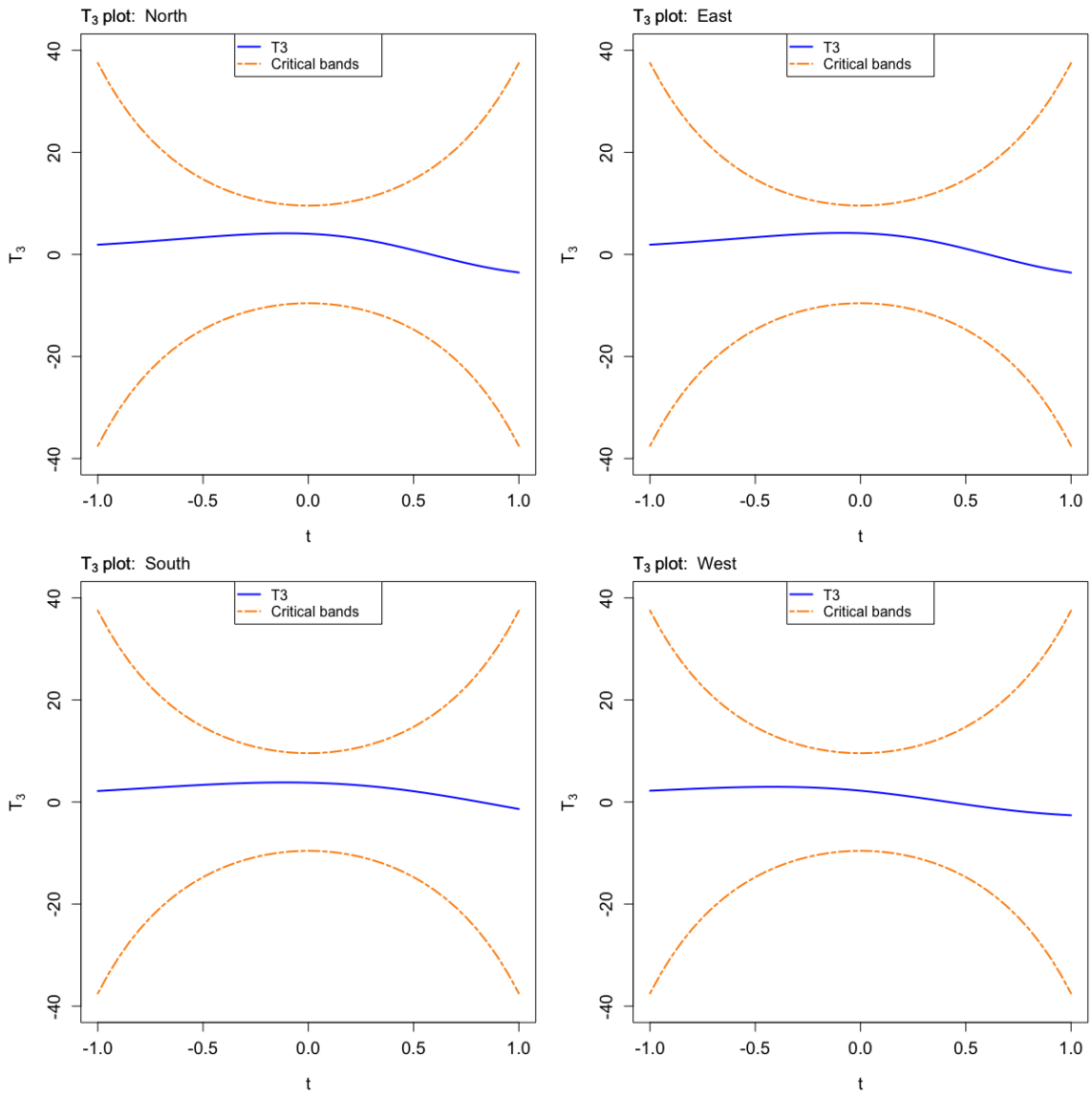
(a) Q - Q plots.

Figure 2.23: Graphical assessment for Cork data set.



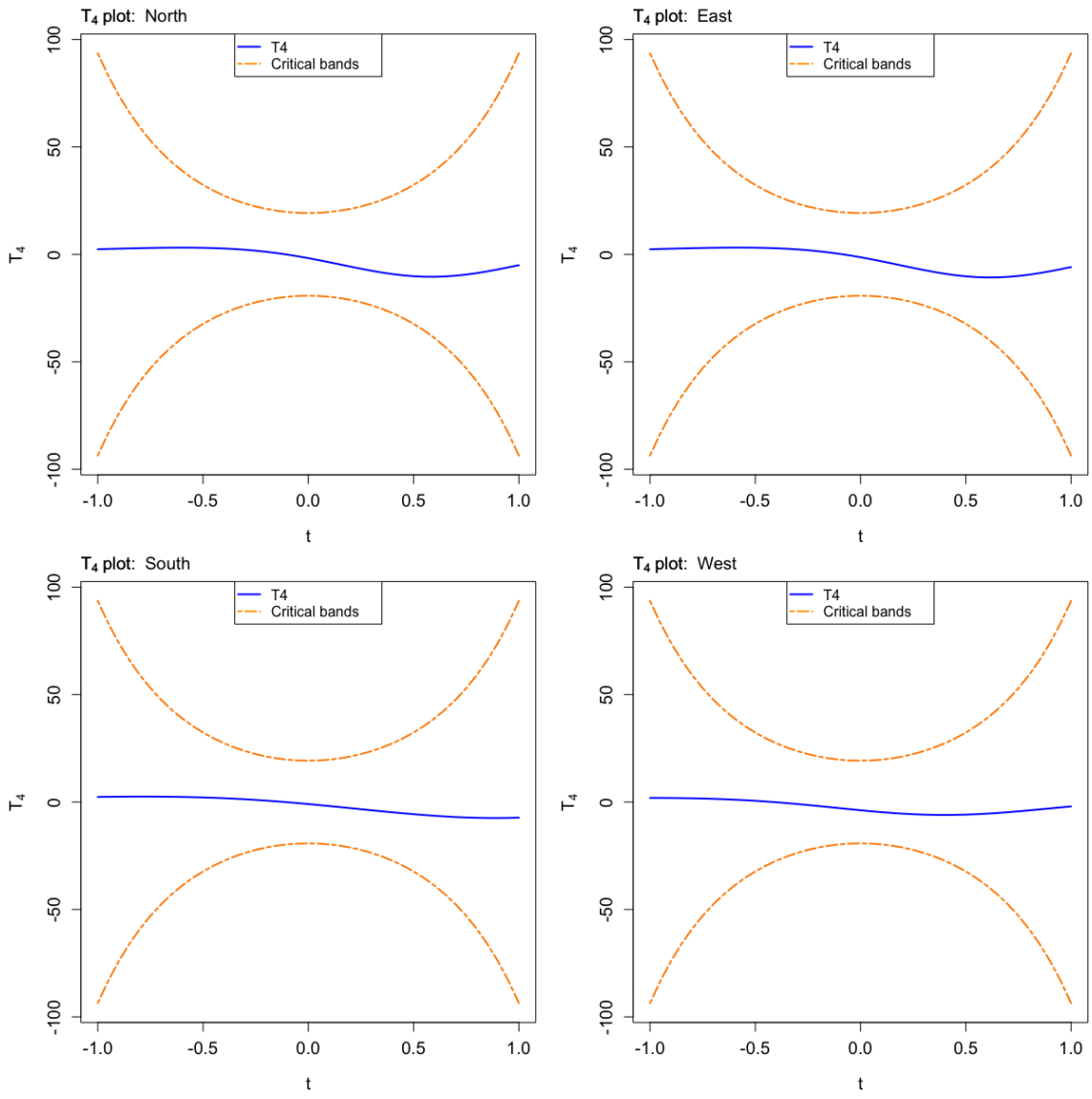
(b) Score plots

Figure 2.23: Graphical assessment for Cork data set.



(c) T_3 plots.

Figure 2.23: Graphical assessment for Cork data set.



(d) T_4 plots.

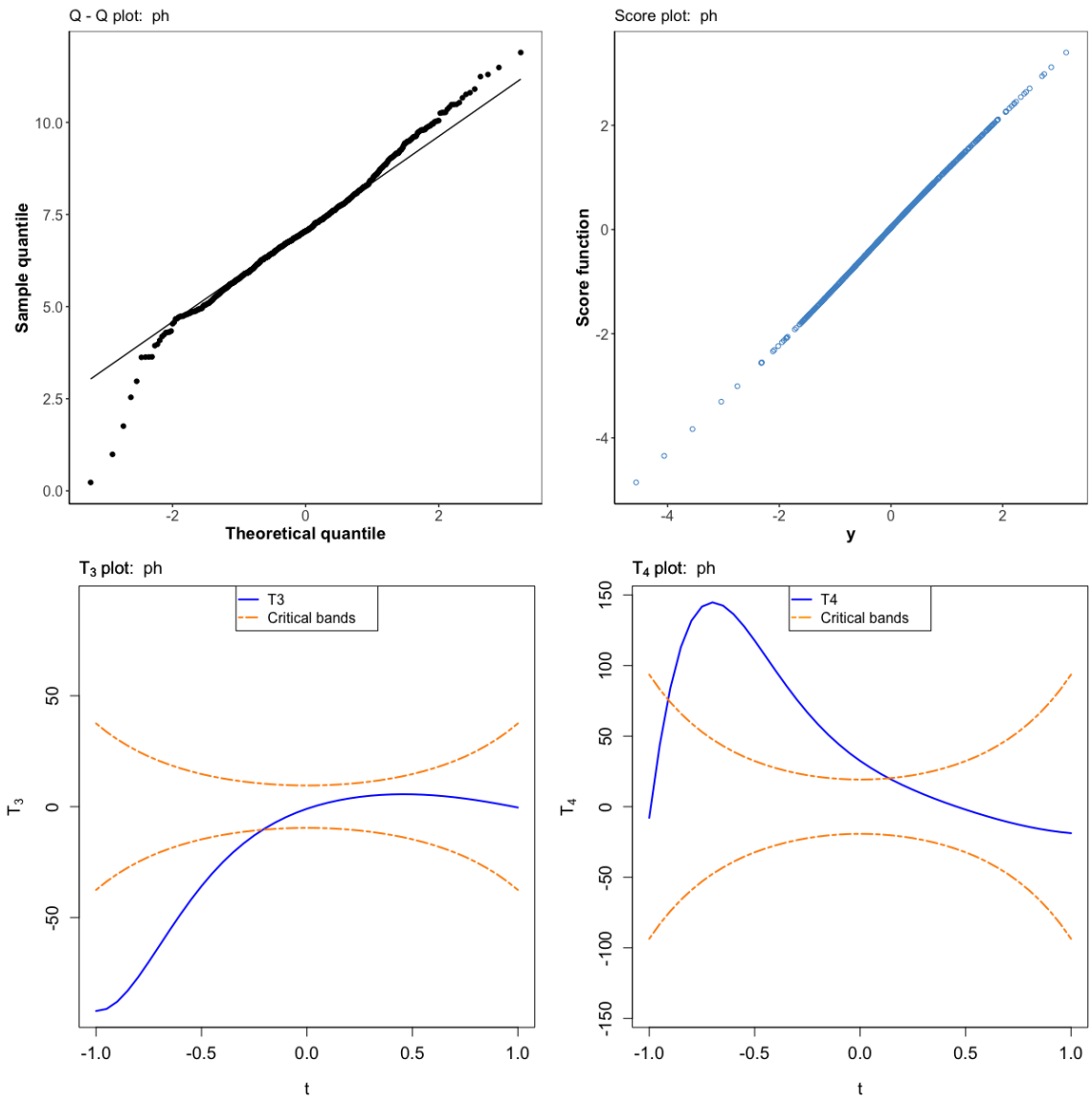
Figure 2.23: Graphical assessment for Cork data set.

the good water population (with Potability = 1), while the rest of the 1200 observations are from the unhealthy water population (Potability = 0). We will consider only the potable water sample (Potability = 1) and perform a graphical assessment for the marginal normality using all four of the above approaches. For score plots, points with a distance less than $.001$ have been merged. The bands for T_3 and T_4 plots correspond to 95% probability. We will contrast the four approaches for one variable at a time.

Figure 2.24 gives the corresponding graphs. For the variable measure *pH*, the Q - Q plot shows a departure from normality when quantiles in the two tails drop off the linear trend. The evidence of non-normality in T_3 and T_4 plots can be vividly seen where the curves even cross the confidence bands. However, the score plot fails to detect such departure from normality. When testing for normality assumption of *Hardness*, all three approaches, namely Q - Q plot, T_3, and T_4 plots, show mild departures. However, the score plot shows many points that are off the linear pattern. The non-normality for variable *Solids* is definitely evident in all four plots[5]. Graphs for *Chloramines* support the assumption of approximate normality, but it is not fully satisfactory. Although the Q - Q plot of variable *Sulfate* exhibits a straight line trend and seems to support normality, the other methods make a case of nonnormality. When assessing the variable *Conductivity*, Q - Q plot, T_4 plots exhibit near-normality. However, the score function appears to have a very large variation from the linear trend. In the T_3 plot, the curve has a somewhat nonzero gradient. Finally, the evidence of normality in variables *Organic Carbon*, *Trihalomethanes* and Turbidity is very strong in all four graphical plots, with linear trends in Q - Q plot and score plot and curves with almost zero gradients in T_3 and T_4 plots.

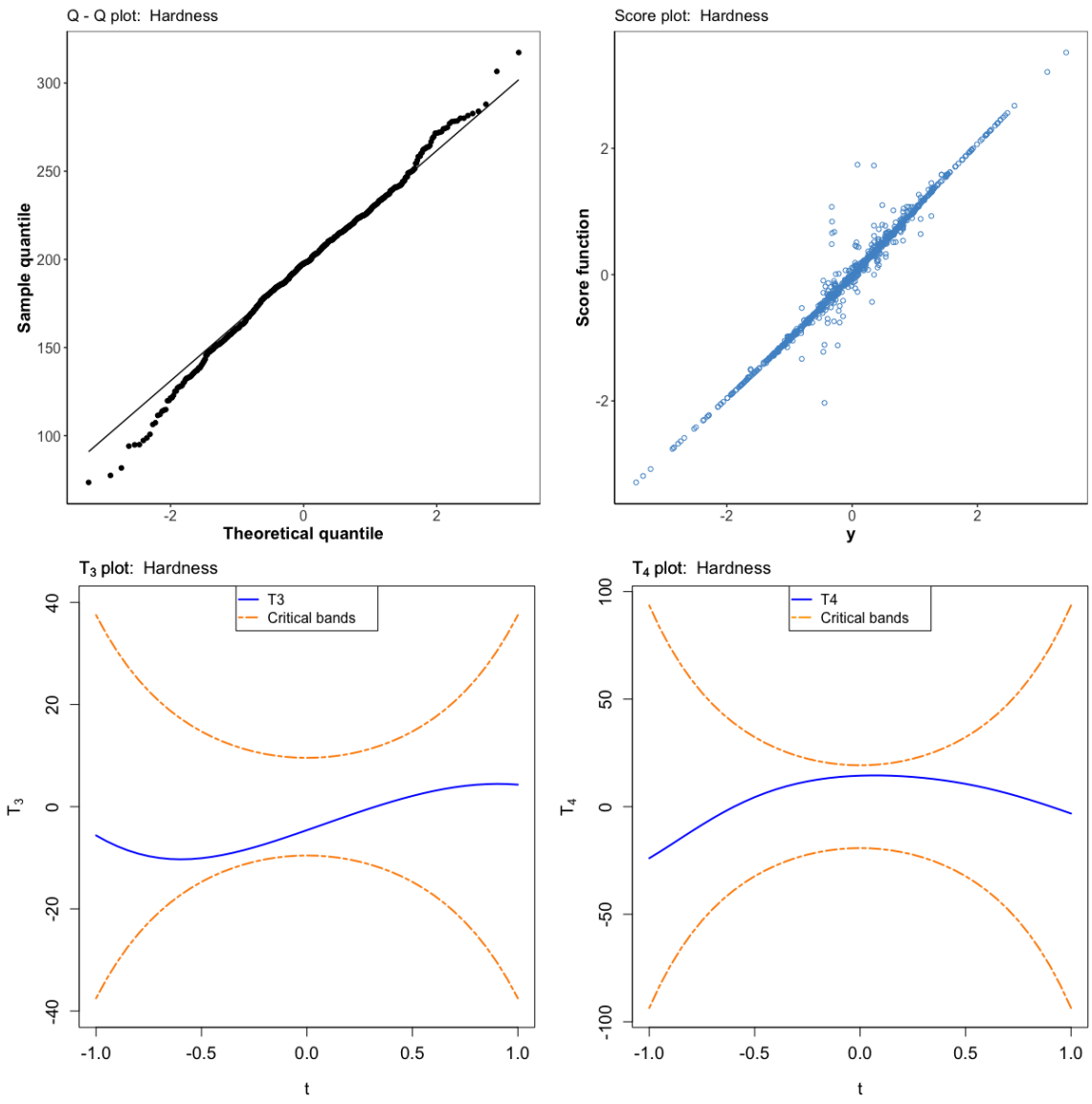
Certain variables have been found to have some non-normal features. What other probability distributions could possibly be fitted into the data for these variables? We can

[5]For score plot, 14 observations with very large score values ($|a_k| \geq 100$) has been deliberately removed, as they mask the trend among remaining scores.



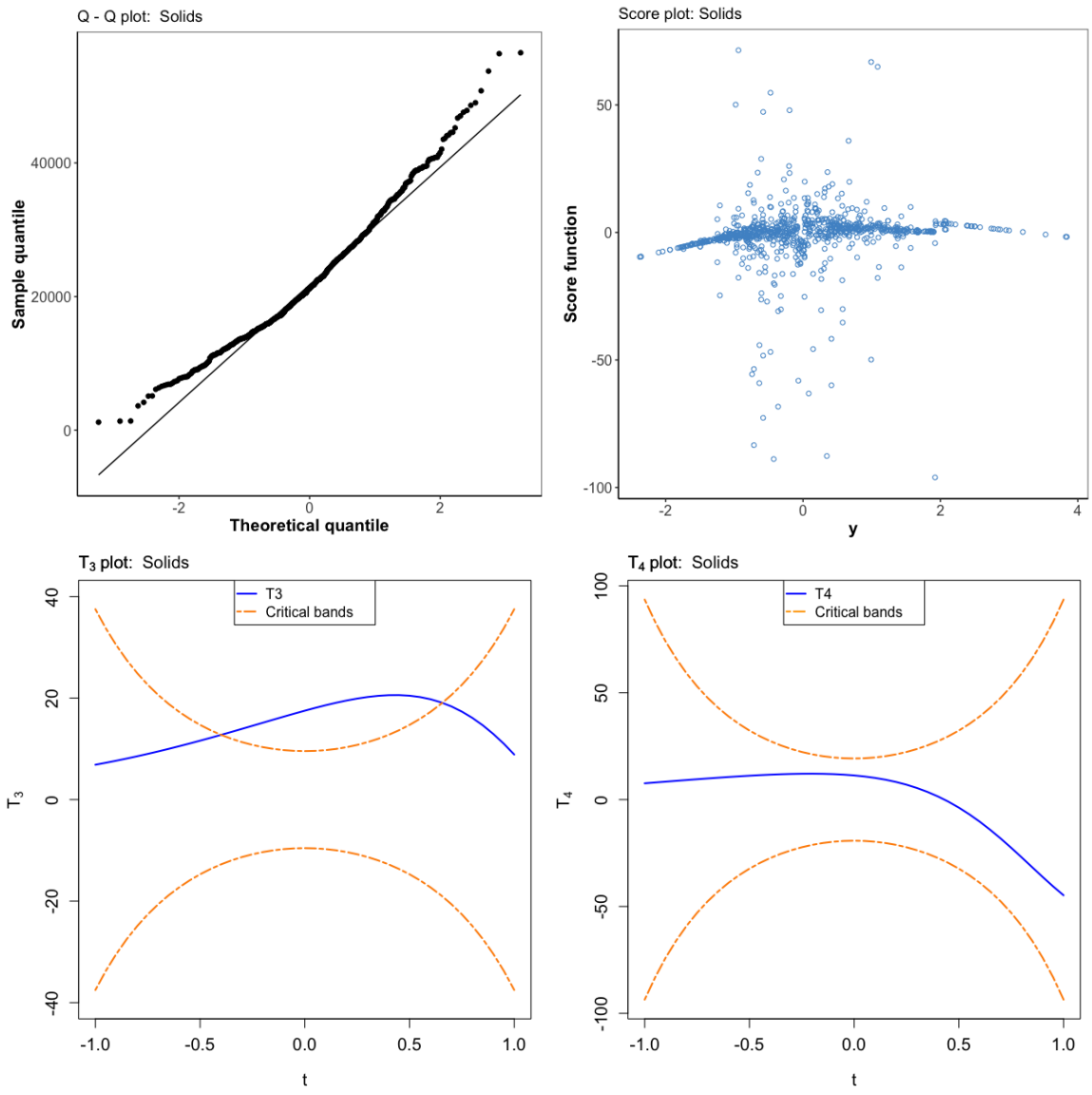
(a) Variable *pH*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



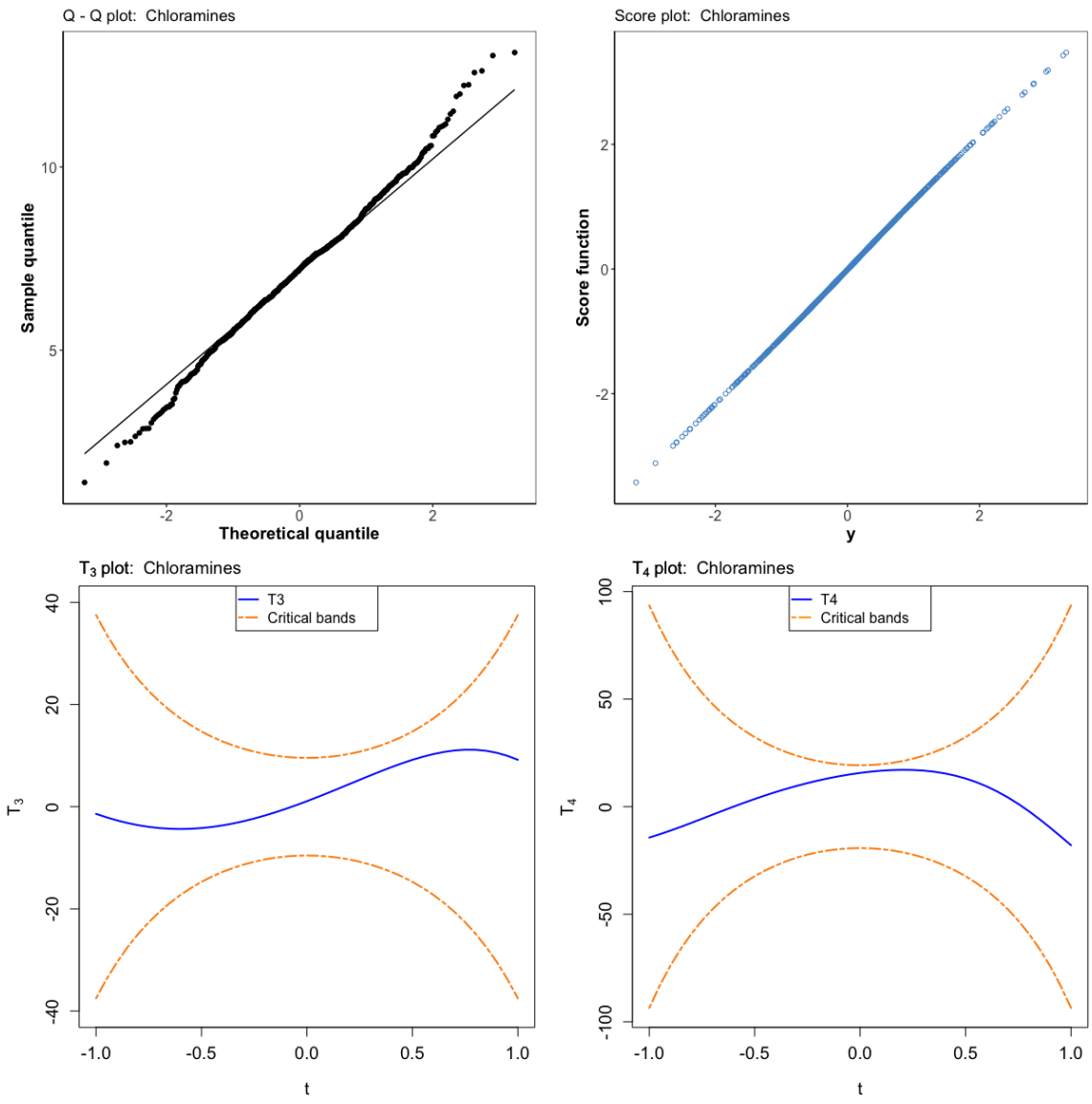
(b) Variable *Hardness*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



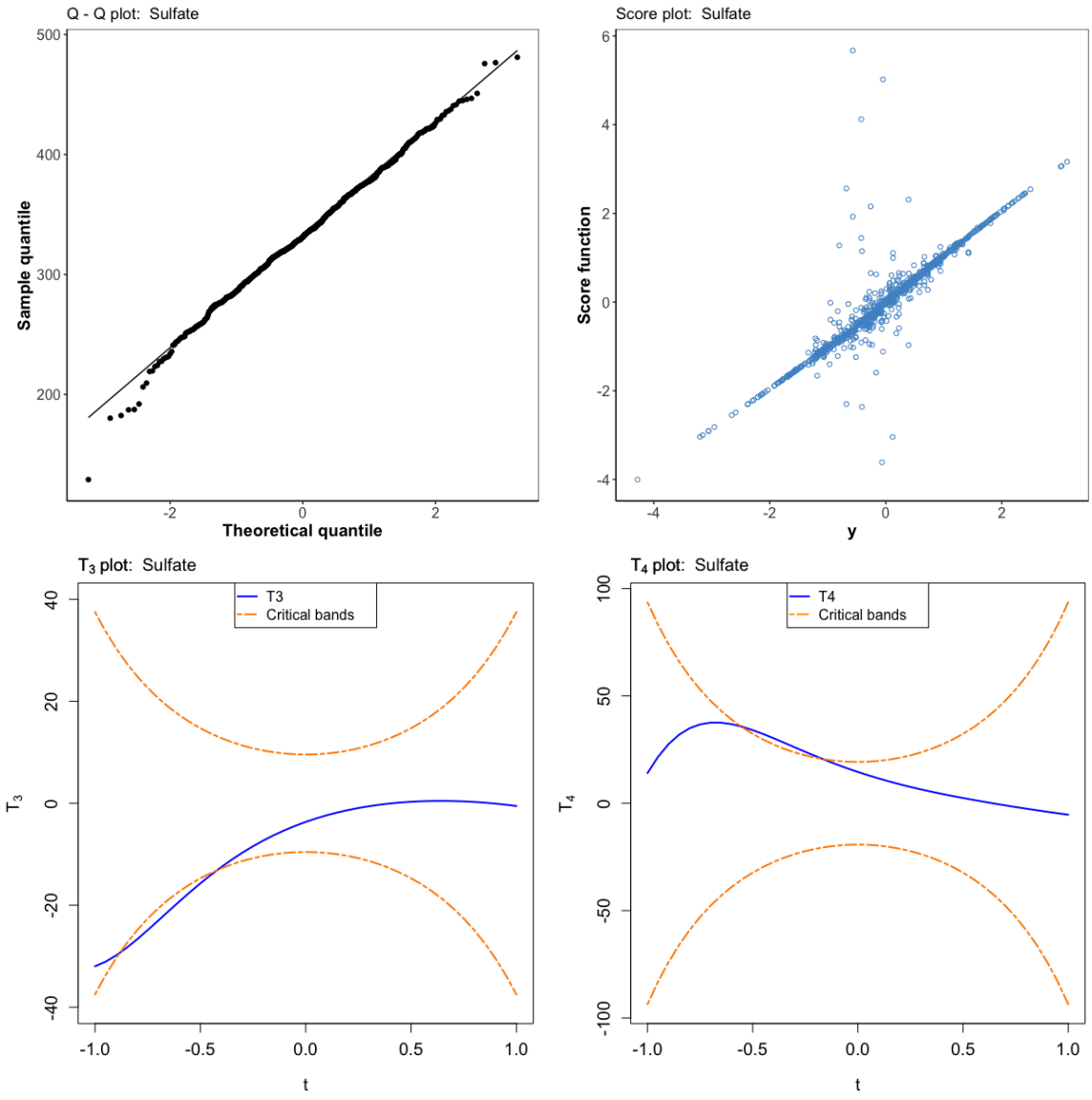
(c) Variable *Solids*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



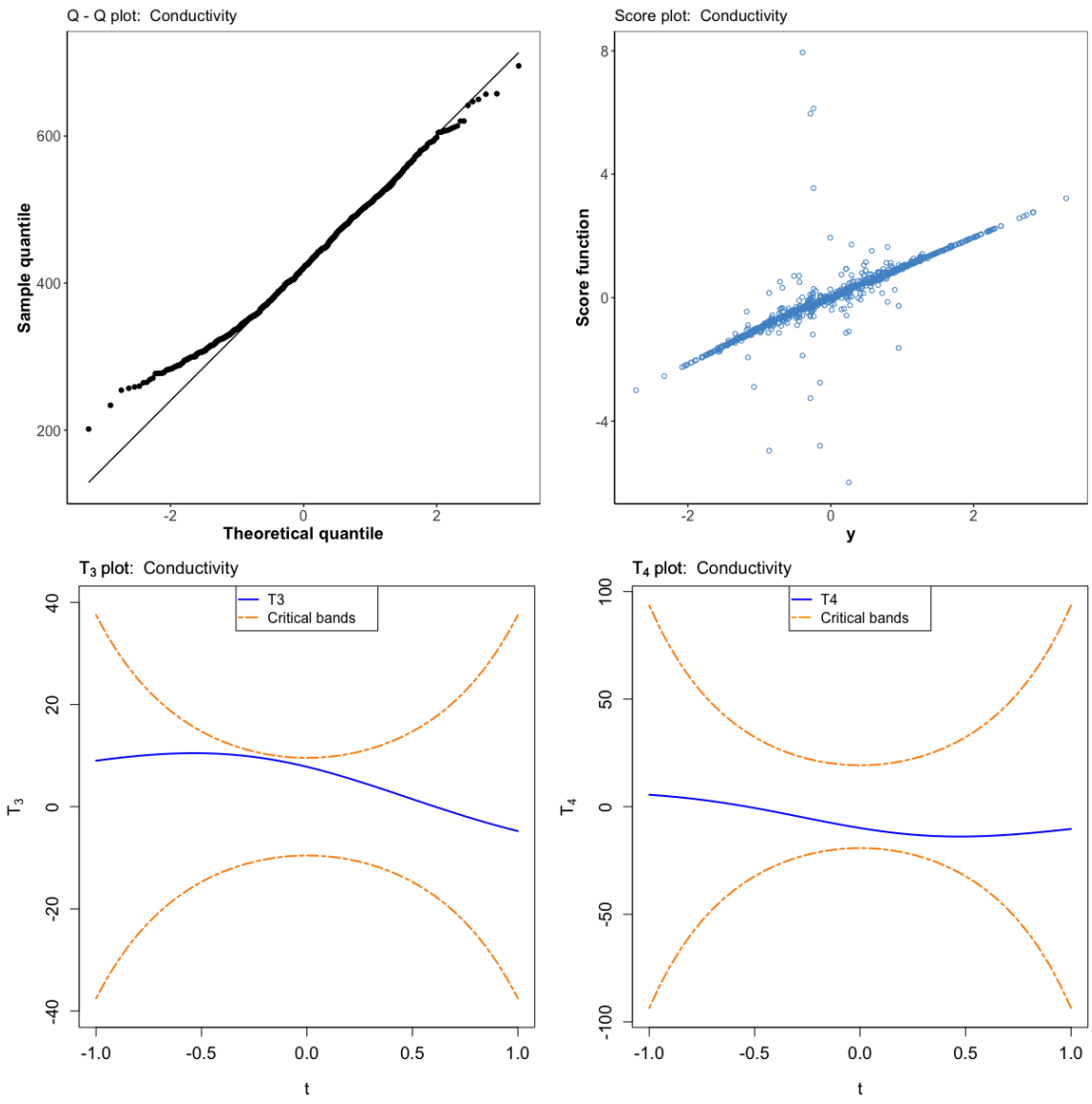
(d) Variable *Chloramines*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



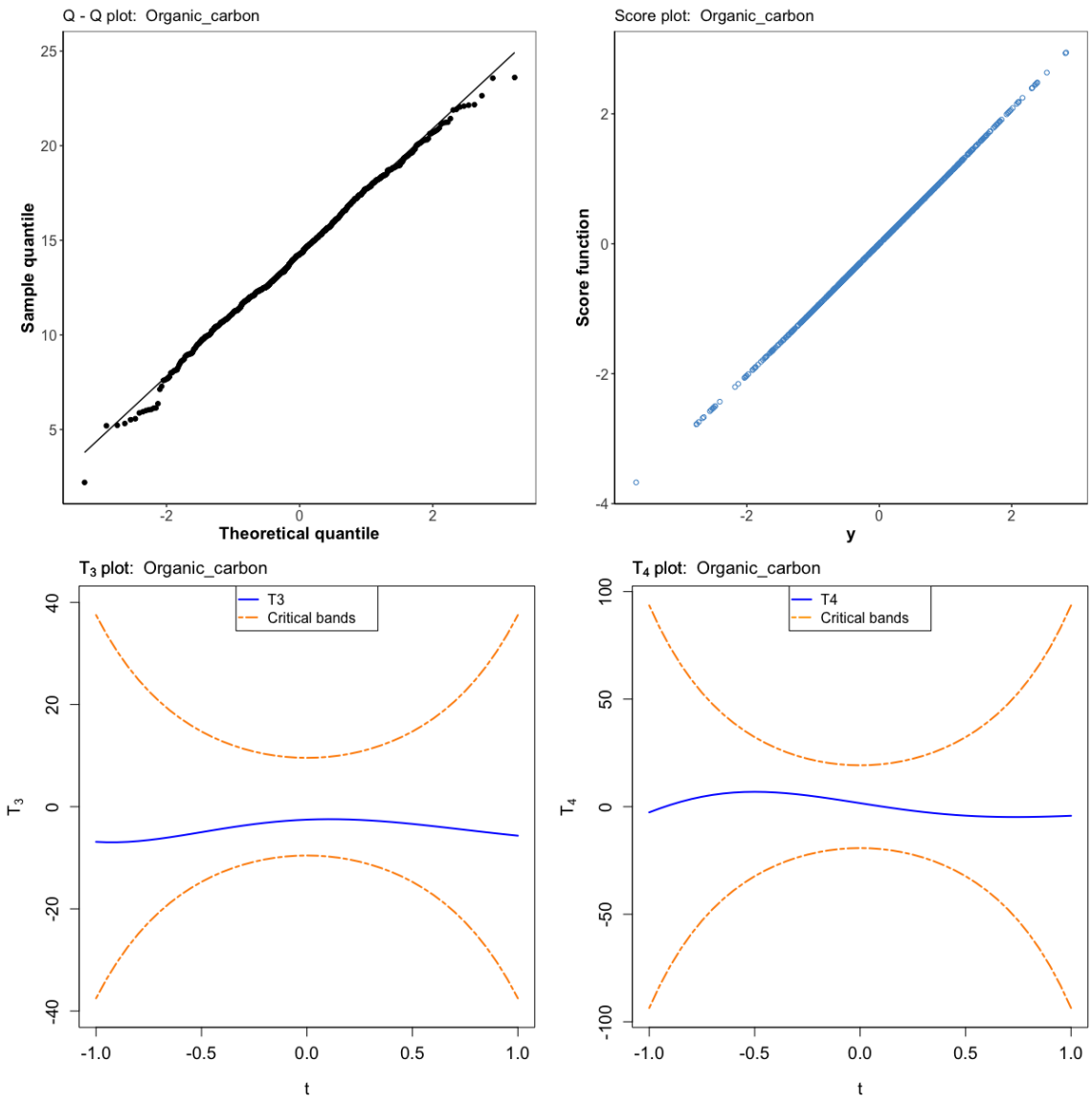
(e) Variable *Sulfate*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



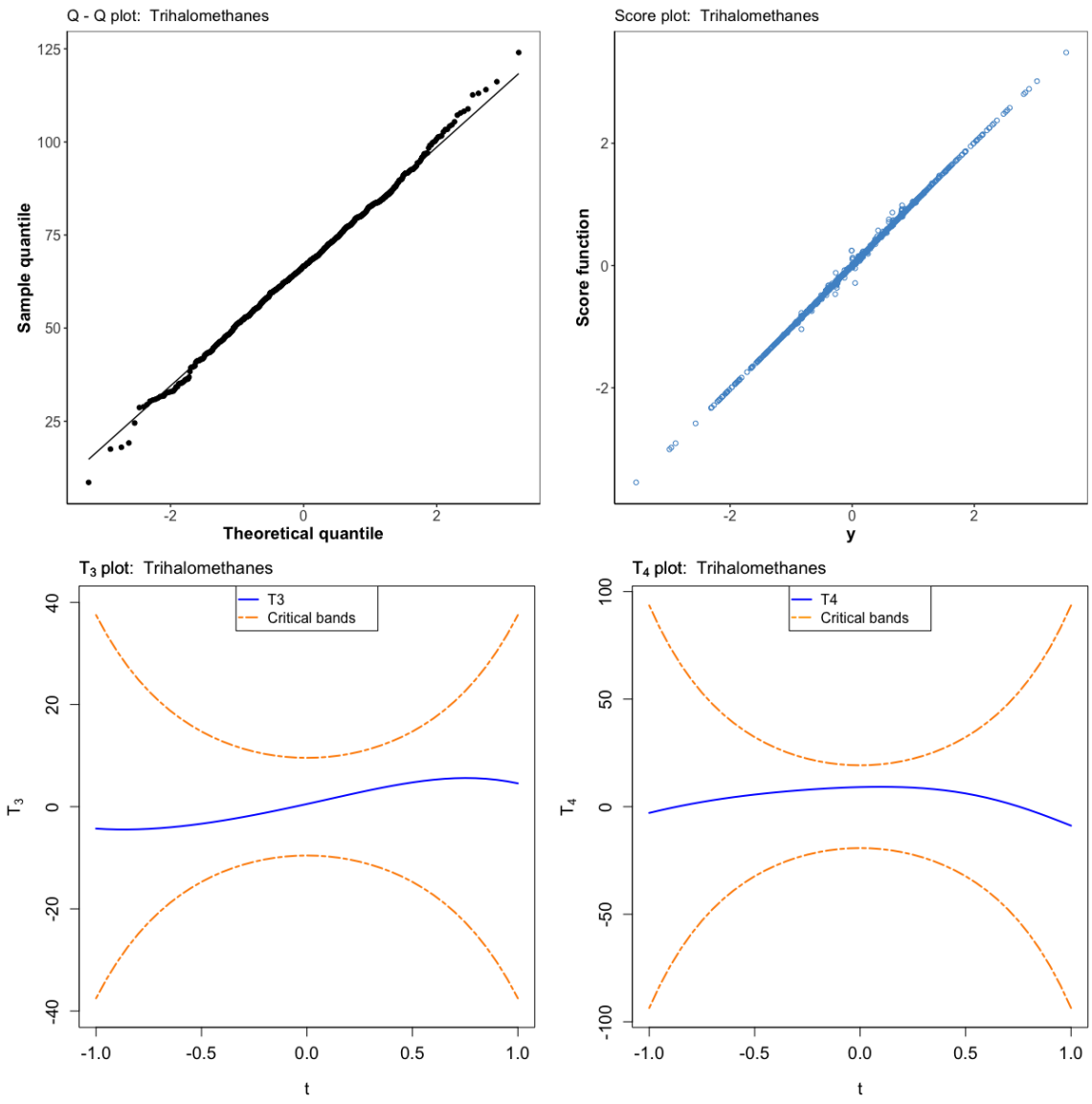
(f) Variable *Conductivity*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



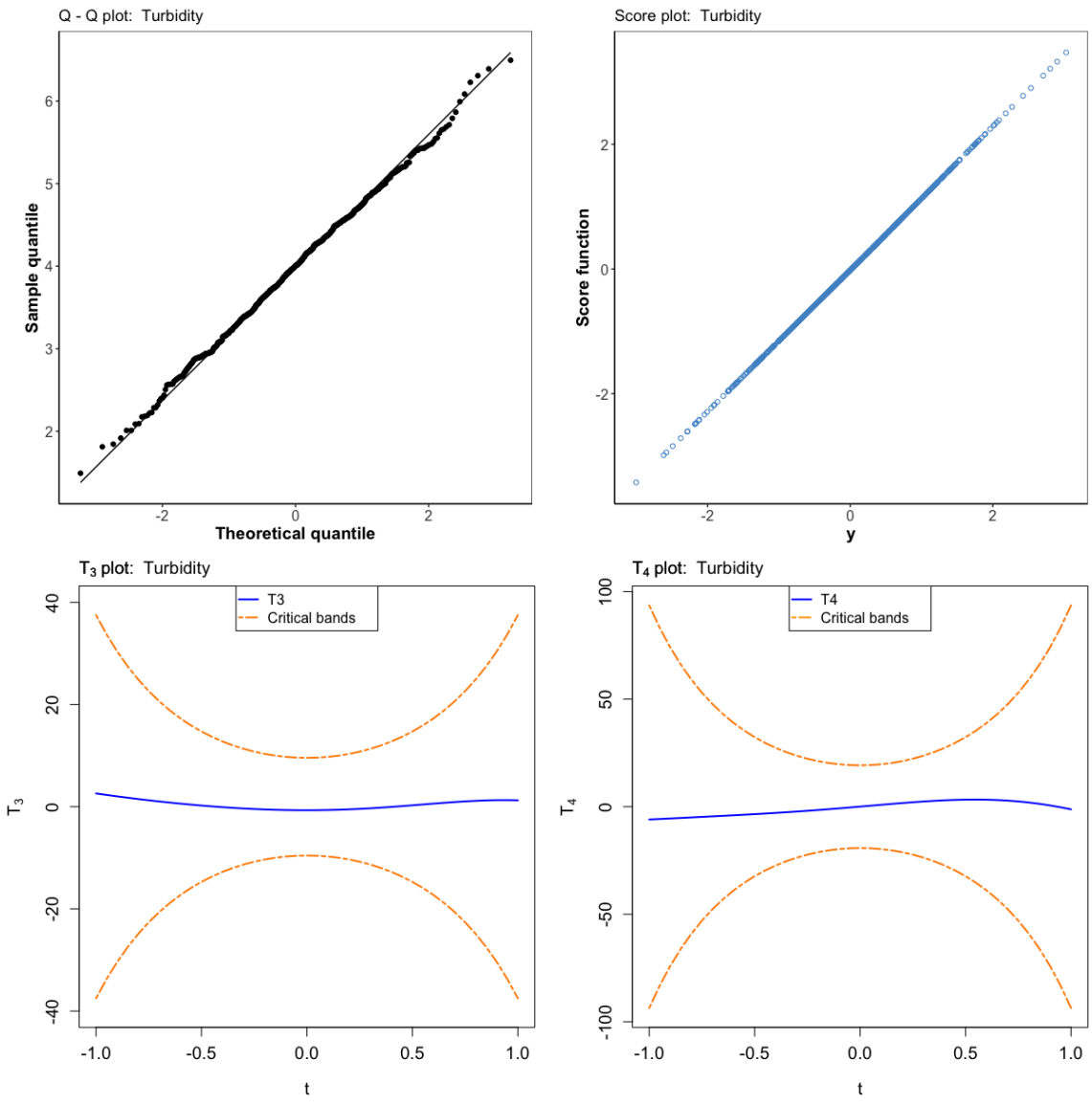
(g) Variable *Organic Carbon*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



(h) Variable *Trihalomethanes*

Figure 2.24: Graphical assessment for Water Potability (= Good) data set.



(i) Variable *Turbidity*

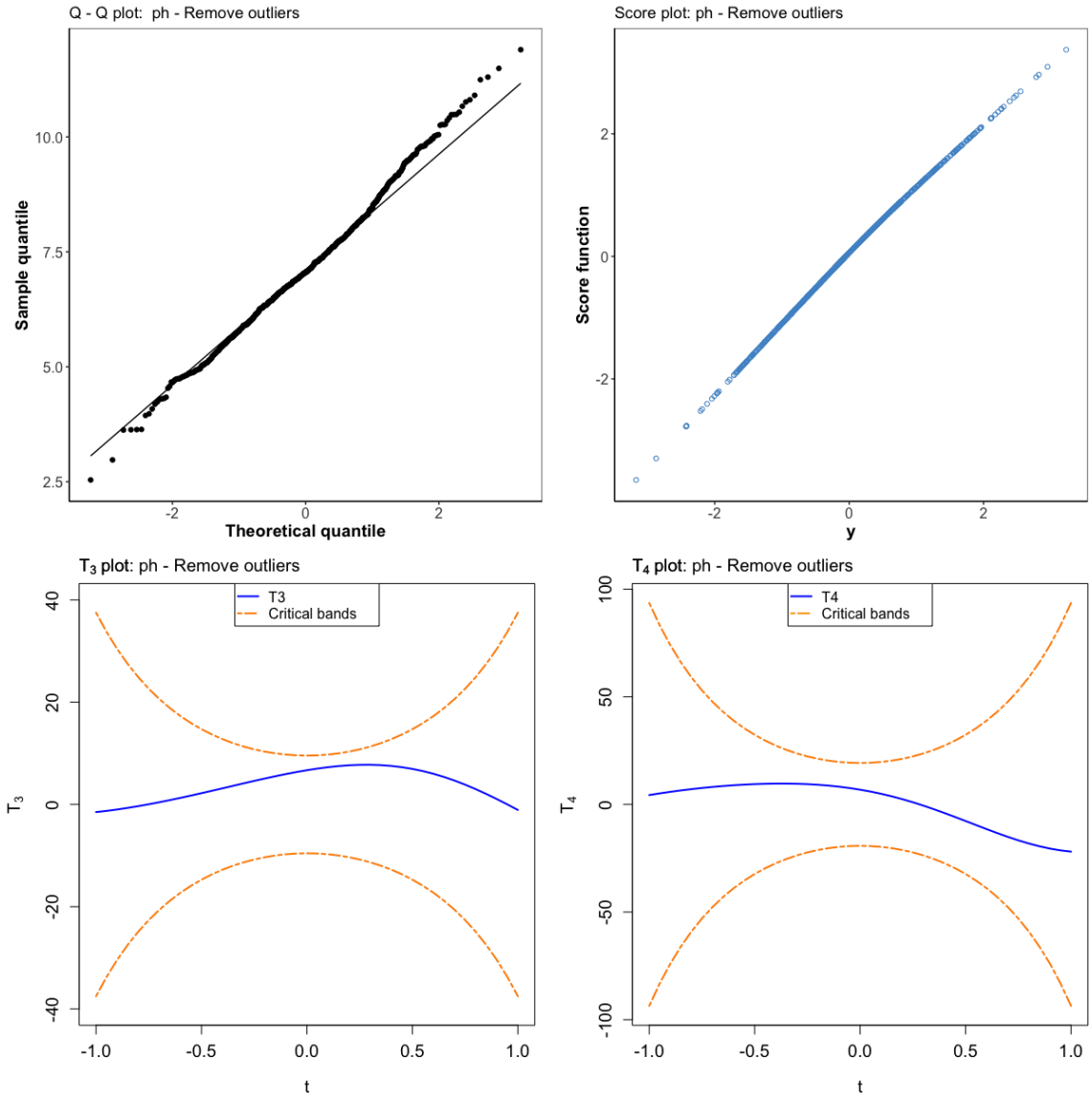
Figure 2.24: Graphical assessment for Water Potability (= Good) data set.

attempt to investigate so by using the transformation method described in Section 2.4. However, before we embark on it, we remove rows 122, 141, 196, 328, 374, 732, 739, and 777, as they may be potential outliers. After we do that, with the rest of the data, the variable *pH* seems to have a normal distribution (Figure 2.25a). It is possible to model *Hardness* as a variable that follows a Logistic distribution with the estimated parameters $\hat{\mu} = 196.6$ and $\hat{\beta} = 19.6$. See Figure 2.25b. A Gamma distribution can be assumed for variable *Solids* with the estimated parameter $\hat{\alpha} = 6.48$ and $\hat{\beta} = 3462.2$ (Figure 2.25c). Similarly, the variable *Sulfate* can be modeled by a Logistic distribution with estimated parameters $\hat{\mu} = 332.2$ and $\hat{\sigma} = 26.4$ (Figure 2.25d). Finally, a Gamma distribution with estimated parameters $\hat{\alpha} = 26.7$ and $\hat{\beta} = 15.9$ can be fitted for the *Conductivity*. See Figure 2.25e.

2.7 Concluding Remarks

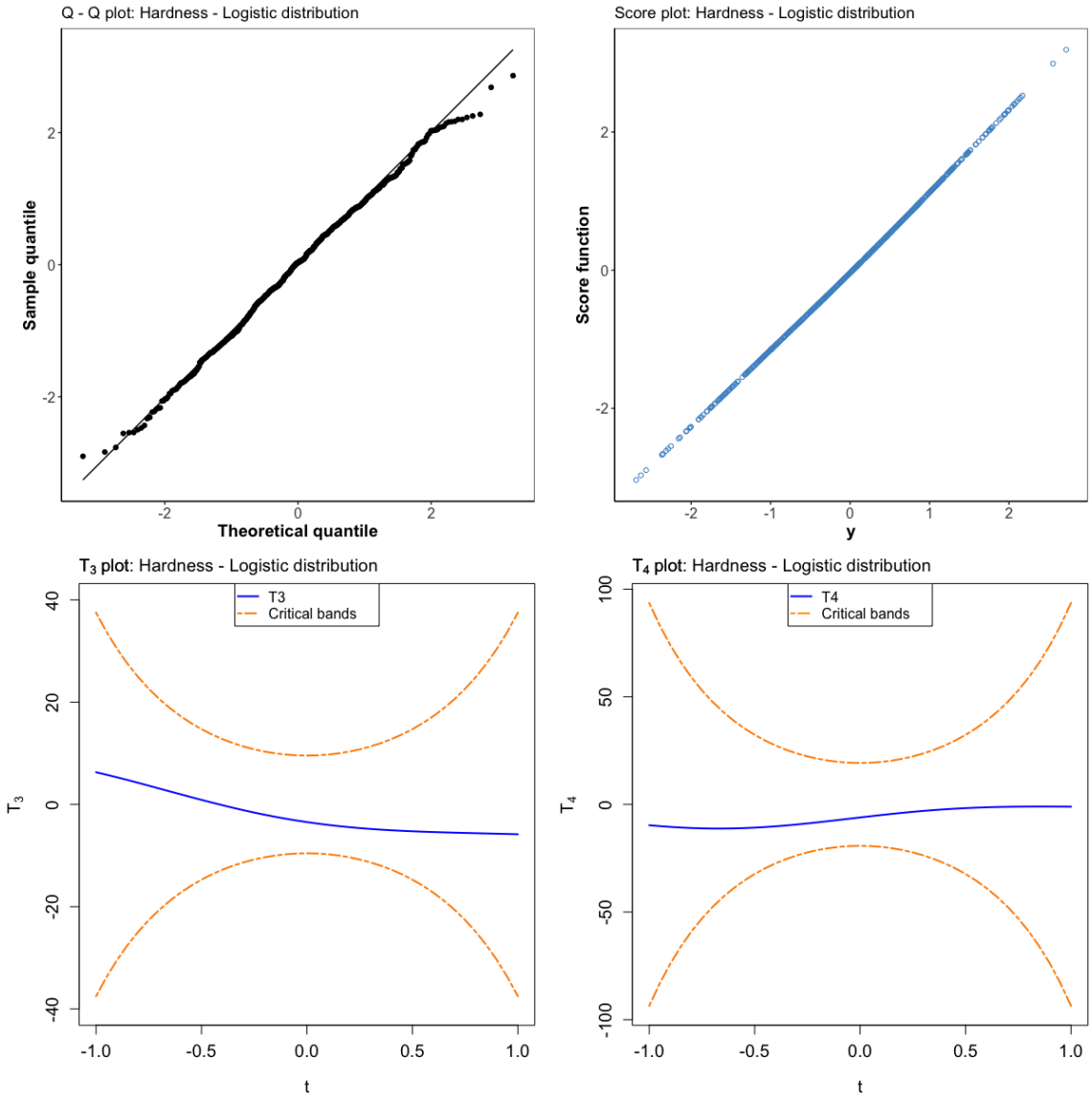
When performing the score plot, we prefixed the smoothing parameter as $\lambda = 0.5$, and also used the criterion of merging two points if their distance is smaller than 0.001 (in many cases). These parameters are very subjective, so, implementing the score plots in real-world problems may be subject to some level of arbitrariness. The minimum distance between two points tends to be smaller as the sample size increases. It can be even smaller than 0.001, which may cause the singularity problem in our algorithm. Therefore, a random sample from a normal distribution with the same sample size should be simulated in advance to obtain good estimates for both the smoothing parameter and the requirement on the distance between two points. These parameters can then be used to test for the normality assumption of data or to detect outliers.

Generally, T_3 and T_4 plots perform better, and each has a somewhat different target. More specifically, the T_3 plot attempts to identify non-normality due to departure from symmetry. On the other hand, the T_4 plot focuses on the departure from normality in the tail area.



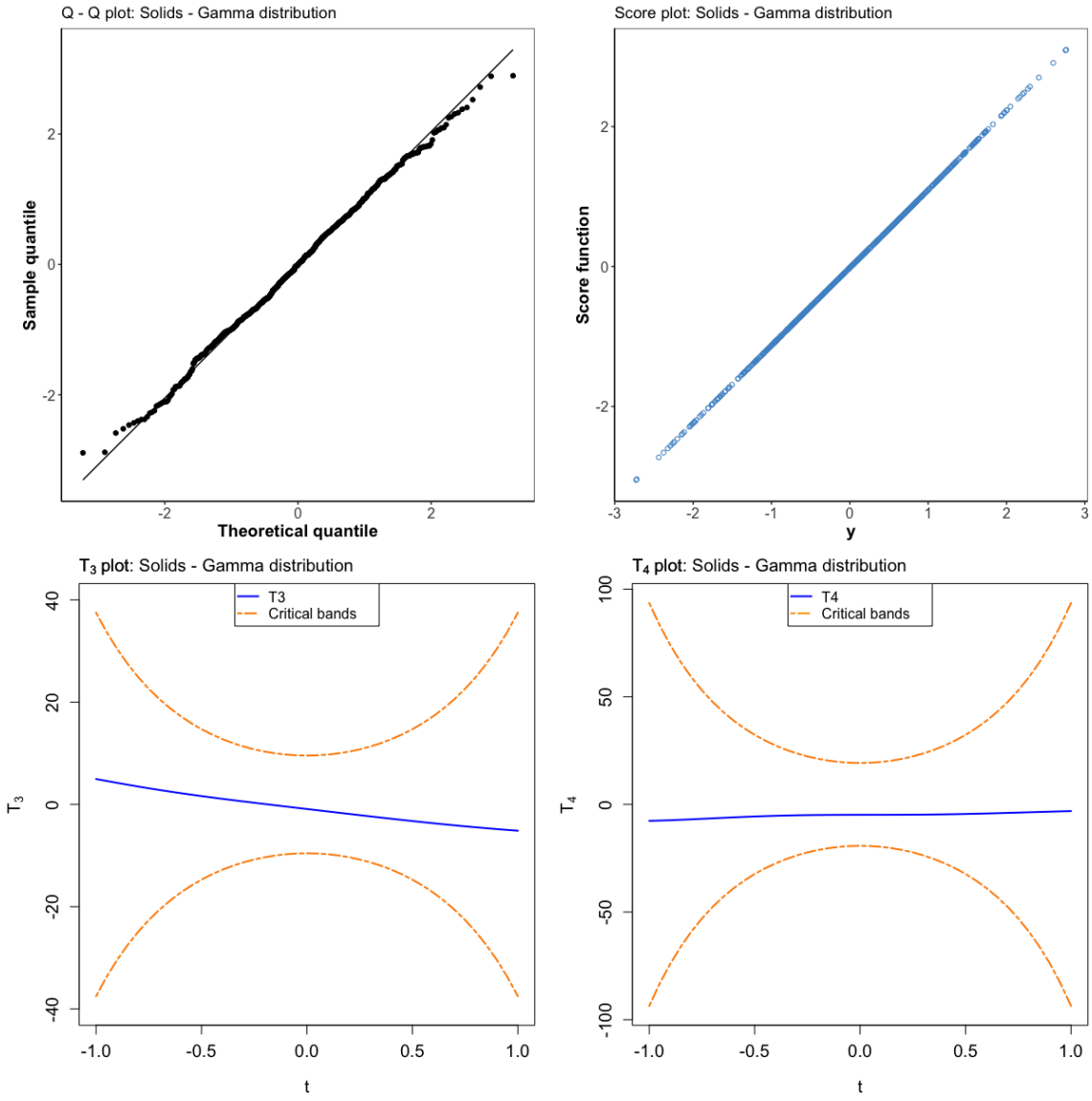
(a) Variable *pH*

Figure 2.25: Fitting appropriate distributional assumptions for non-normal variables of Water Potability (= Good) data set.



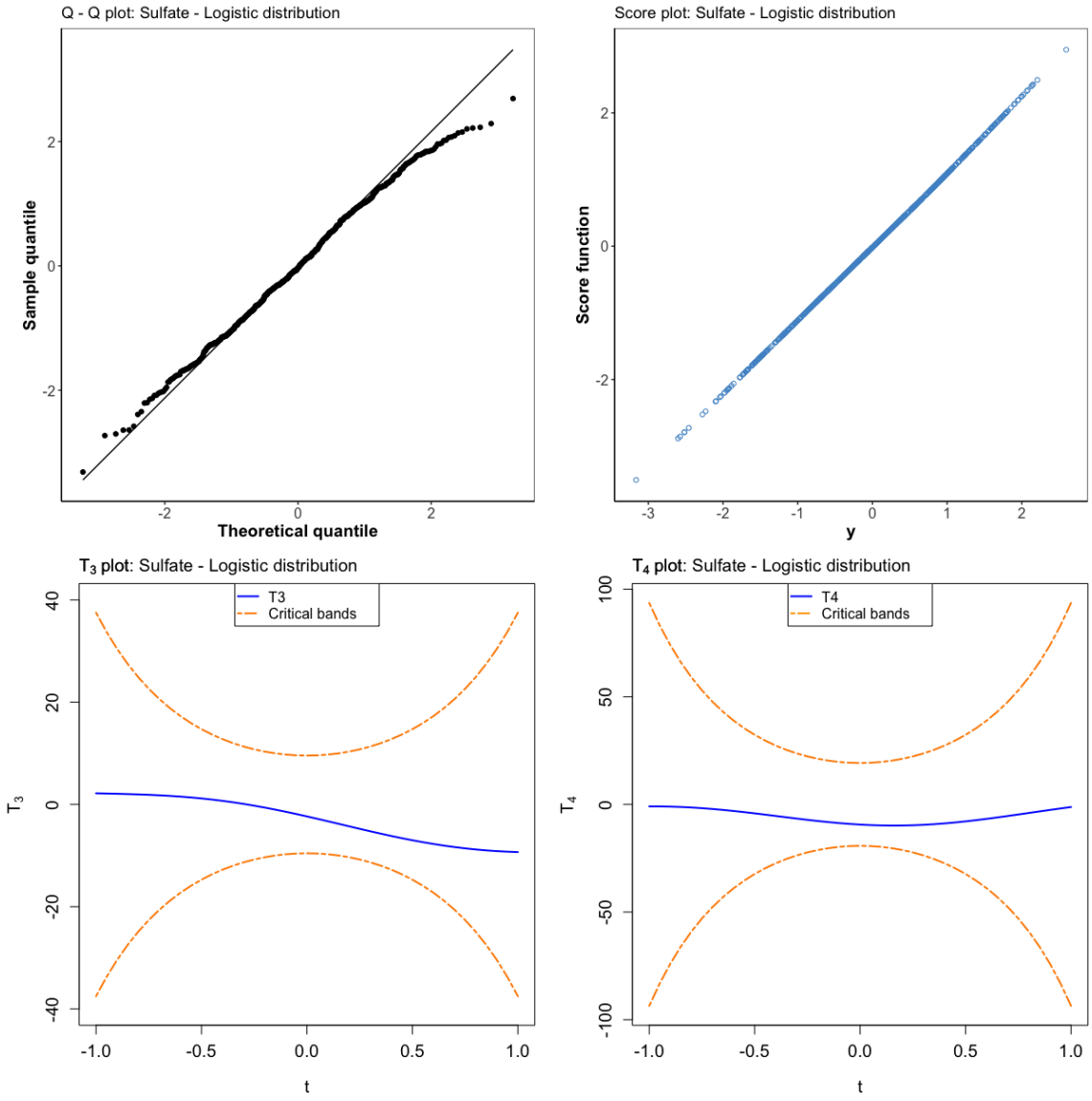
(b) Variable *Hardness*

Figure 2.25: Fitting appropriate distributional assumptions for non-normal variables of Water Potability (= Good) data set.



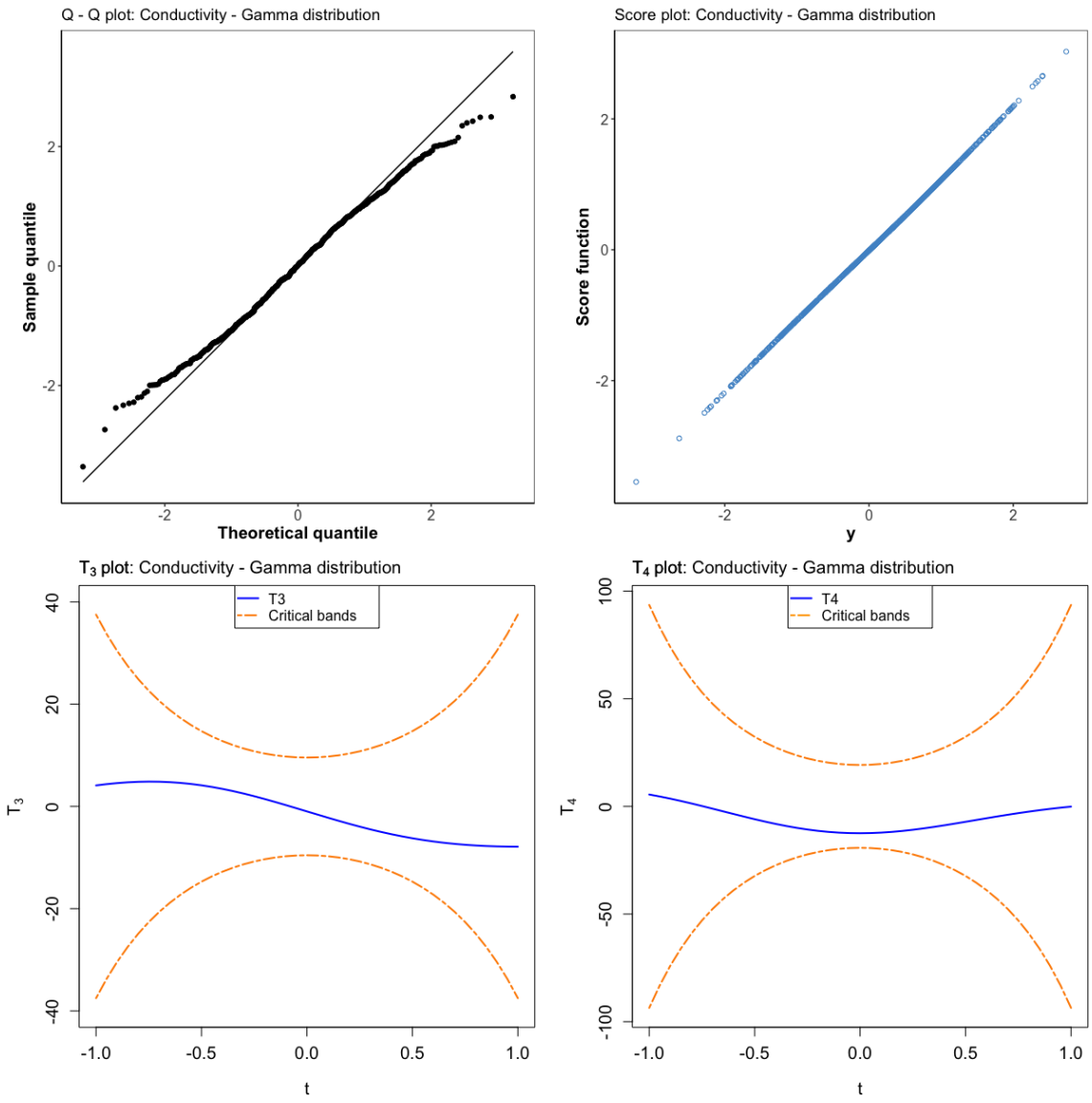
(c) Variable *Solids*

Figure 2.25: Fitting appropriate distributional assumptions for non-normal variables of Water Potability (= Good) data set.



(d) Variable *Sulfate*

Figure 2.25: Fitting appropriate distributional assumptions for non-normal variables of Water Potability (= Good) data set.



(e) Variable *Conductivity*

Figure 2.25: Fitting appropriate distributional assumptions for non-normal variables of Water Potability (= Good) data set.

Finally, we must point out that there are several potential distributions that can be fitted on the same dataset. This is especially the case when the probability densities coincide or overlap in a major region.

CHAPTER THREE

EVALUATION OF MULTIVARIATE NORMALITY USING UNIVARIATE METHODS

3.1 Introduction

Unlike the case of univariate normality, where various hypothesis testing methods and graphical approaches can be developed, the problem of testing for multivariate normality becomes extremely complicated due to the complexity of high dimensional data. Mardia [3] proposed measures of multivariate skewness and multivariate kurtosis, whose respective values, under normality assumption, are 0 and $p(p+2)$, and further developed the well-known Mardia's skewness and Mardia's kurtosis statistics aiming to find evidence of non-normality in term of these measures. Baringhaus and Henze [15] employed the characteristic function of a multivariate normal random vector to derive tests for the hypothesis of normality. Anderson and Darling [32] introduced a test statistic, which quantifies the distance between sample data and the null hypothesis of multivariate normal distribution. However, these formal tests attempt to summarize the multidimensional information into a single number, whereas more insight can be gained via graphical approaches. For the bivariate normality, numerous graphical tests such as scatter plots and contour plots can help assess the data distributional assumption [35]. However, it is difficult to obtain such graphs when data are from a very high dimensional space. To this end, the most popular graphical method for multivariate data is the Q-Q plot [35] based on Mahalanobis distance, whose approximate distribution under multivariate normality is $\chi^2(p)$.

This chapter extends the methods of Chapter 2 to multivariate situations. We will introduce several methods to assess the normality assumption of multivariate data sets. In section 3.2, we will present a fundamental approach by employing the relationship of a standard multivariate normal distribution with a univariate one. This facilitates the use of

methods developed in Chapter 2 to perform the tests and also have the graphical assessments. Illustrative examples for this approach are presented in Sections 3.3 and 3.4. Section 3.5 extends the normality test method to the multivariate outlier detection problem. Section 3.6 will derive three different tests based on the "best" linear transformation with illustrative examples presented in Section 3.7 and 3.8. Specifically, the celebrated Cramer-Wold theorem [56] will play the central role here. We conclude the chapter with remarks in Section 3.9.

3.2 A Method Based on "$n \times p$-sample"

In this section we will test the multivariate normality using various approaches described in the previous chapter. The idea is to transform data, under the assumption of normality to something close to $N_p(\mathbf{0}_p, \mathbf{I}_p)$, where the components can then be treated as a *i.i.d* sample from standard univariate normal. To do this, let $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ be a random sample from a multivariate population with cdf $F(.)$. Let the population have mean $\boldsymbol{\mu}$ and variance-covariance matrix $\boldsymbol{\Sigma}$. Also, let

$$\bar{\mathbf{X}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_i \text{ and } \mathbf{S} = \frac{1}{n-1}\sum_{i=1}^{n} (\mathbf{X}_i - \bar{\mathbf{X}})(X_i - \bar{\mathbf{X}})'$$

be the unbiased estimators of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively. Define

$$\bar{\mathbf{X}}_{-k} = \frac{1}{n-1}\sum_{i=1, i\neq k}^{n} \mathbf{X}_i \text{ and } \mathbf{S}_{-k} = \frac{1}{n-2}\sum_{i=1, i\neq k}^{n} (\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})'$$

which are the unbiased estimators of the corresponding parameters using all but the k^{th} observation. The following lemma enables us to compute $\bar{\mathbf{X}}_{-k}$ and $\mathbf{S}_{-k}$ directly from $\bar{\mathbf{X}}$ and $\mathbf{S}$.

Lemma 3.1. Using the notations above,

(i) $\bar{\mathbf{X}}_{-k} = \frac{1}{n-1}\left(n\bar{\mathbf{X}} - \mathbf{X}_k\right)$.

(ii) $\mathbf{S}_{-k} = \frac{n-1}{n-2}\left[\mathbf{S} - \frac{n}{(n-1)^2}\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)'\right]$.

(iii) $\mathbf{S}_{-k}^{-1} = \dfrac{n-2}{n-1}\left[\mathbf{S}^{-1} + \dfrac{\mathbf{S}^{-1}\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)'\mathbf{S}^{-1}}{\dfrac{(n-1)^2}{n} - (\mathbf{X}_k - \bar{\mathbf{X}})'S^{-1}(\mathbf{X}_k - \bar{\mathbf{X}})}\right]$.

Proof. The first part involves basic algebra. Specifically,
$\bar{\mathbf{X}} = \dfrac{1}{n}\sum_{i=1, i\neq k}^{n} \mathbf{X}_i + \mathbf{X}_k \Rightarrow n\bar{\mathbf{X}} = (n-1)\bar{\mathbf{X}}_{-k} + \mathbf{X}_k$. Hence (i) is derived.

To prove (ii), we observe that

$$
\begin{aligned}
n\bar{\mathbf{X}} - (n-1)\bar{\mathbf{X}} - \mathbf{X}_k &= (n-1)\bar{\mathbf{X}}_{-k} - (n-1)\bar{\mathbf{X}} \\
\bar{\mathbf{X}} - \mathbf{X}_k &= (n-1)\left(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}}\right) \\
\Rightarrow (\bar{\mathbf{X}} - \mathbf{X}_k)(\bar{\mathbf{X}} - \mathbf{X}_k)' &= (n-1)^2(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}})(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}})' \\
\Rightarrow \frac{1}{n-1}(\bar{\mathbf{X}} - \mathbf{X}_k)(\bar{\mathbf{X}} - \mathbf{X}_k)' &= (n-1)(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}})(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}})'.
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\sum_{i=1, i\neq k}^{n} \left(\mathbf{X}_i - \bar{\mathbf{X}}\right)\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)' \\
=&\sum_{i=1, i\neq k}^{n} \left[\left(\mathbf{X}_i - \bar{\mathbf{X}}_{-k}\right) + \left(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}}\right)\right]\left[\left(\mathbf{X}_i - \bar{\mathbf{X}}_{-k}\right) + \left(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}}\right)\right]' \\
=&\sum_{i=1, i\neq k}^{n} \left(\mathbf{X}_i - \bar{\mathbf{X}}_{-k}\right)\left(\mathbf{X}_i - \bar{\mathbf{X}}_{-k}\right)' + (n-1)\left(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}}\right)\left(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}}\right)' \\
&+ 2\left(\bar{\mathbf{X}}_{-k} - \bar{\mathbf{X}}\right)\sum_{i=1, i\neq k}^{n}\left(\mathbf{X}_i - \bar{\mathbf{X}}_{-k}\right)' \\
=&(n-2)\mathbf{S}_{-k} + \frac{1}{n-1}(\bar{\mathbf{X}} - \mathbf{X}_k)(\bar{\mathbf{X}} - \mathbf{X}_k)',
\end{aligned}
$$

since the cross-product term involves the sum of deviations from the mean and is thus zero.

Accordingly,

$$\begin{aligned}(n-1)\mathbf{S} &= \sum_{i=1, i\neq k}^{n} \left(\mathbf{X}_i - \bar{\mathbf{X}}\right)\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)' + \left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)' \\ &= (n-2)\mathbf{S}_{-k} + \frac{1}{n-1}(\bar{\mathbf{X}} - \mathbf{X}_k)(\bar{\mathbf{X}} - \mathbf{X}_k)' + \left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)' \\ &= (n-2)\mathbf{S}_{-k} + \frac{n}{n-1}\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)',\end{aligned}$$

therefore,

$$\begin{aligned}\mathbf{S}_{-k} &= \frac{1}{n-2}\left[(n-1)\mathbf{S} - \frac{n}{n-1}(\mathbf{X}_k - \bar{\mathbf{X}})(\mathbf{X}_k - \bar{\mathbf{X}})'\right] \\ &= \frac{n-1}{n-2}\left[\mathbf{S} - \frac{n}{(n-1)^2}\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)\left(\mathbf{X}_k - \bar{\mathbf{X}}\right)'\right].\end{aligned}$$

For (iii), we apply the Sherman-Morrison formula for the inverse of a matrix with a specific structure, namely:

$$(A + uw')^{-1} = A^{-1} - \frac{A^{-1}uw'A^{-1}}{1 + w'A^{-1}u}$$

with $A = \mathbf{S}, u = -w = \frac{\sqrt{n}}{n-1}(\mathbf{X}_k - \bar{\mathbf{X}})$. Specifically,

$$\begin{aligned}\mathbf{S}_{-k}^{-1} &= \frac{n-2}{n-1}\left[\mathbf{S}^{-1} + \frac{n}{(n-1)^2}\frac{\mathbf{S}^{-1}(\mathbf{X}_k - \bar{\mathbf{X}})(\mathbf{X}_k - \bar{\mathbf{X}})'\mathbf{S}^{-1}}{1 - \frac{n}{(n-1)^2}(\mathbf{X}_k - \bar{\mathbf{X}})'\mathbf{S}^{-1}(\mathbf{X}_k - \bar{\mathbf{X}})}\right] \\ &= \frac{n-2}{n-1}\left[\mathbf{S}^{-1} + \frac{\mathbf{S}^{-1}(\mathbf{X}_k - \bar{\mathbf{X}})(\mathbf{X}_k - \bar{\mathbf{X}}')\mathbf{S}^{-1}}{\frac{(n-1)^2}{n} - (\mathbf{X}_k - \bar{\mathbf{X}})'\mathbf{S}^{-1}(\mathbf{X}_k - \bar{\mathbf{X}})}\right].\end{aligned}$$

□

Result 1. If $\mathbf{X} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ then $\boldsymbol{\Sigma}^{-1/2}(\mathbf{X} - \boldsymbol{\mu}) \sim N_p(\mathbf{0}_p, \mathbf{I}_p)$ where $\boldsymbol{\Sigma}^{-1/2}$ is the matrix square root of $\boldsymbol{\Sigma}^{-1}$.

Using the above result, if $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then $\boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_1 - \boldsymbol{\mu}), \boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_2 - \boldsymbol{\mu}), \ldots, \boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_n - \boldsymbol{\mu}) \sim N_p(\mathbf{0}_p, \mathbf{I}_p)$. Therefore, components of

each $\mathbf{Z}_i = \mathbf{\Sigma}^{-1/2}(\mathbf{X}_i - \boldsymbol{\mu})$ are all i.i.d $N(0,1)$. Accordingly,

$$Z_{11}, Z_{12}, \ldots Z_{1p}, Z_{21}, Z_{22}, \ldots Z_{2p}, \ldots, Z_{n1}, Z_{n2}, \ldots Z_{np}$$

can be viewed as a random sample of size $n \times p$ from $N(0,1)$.

However, $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ are unknown. We must use their estimators as replacements, and for large n, under multivariate normality of the population, $\mathbf{S}_{-k}^{-1/2}(\mathbf{X}_k - \bar{\mathbf{X}}_{-k})$, $k = 1, 2, \ldots, n$ will have approximate $N_p(\mathbf{0}_p, \mathbf{I}_p)$ and will be approximately independent. Accordingly, all their p components and those for n observations can be collectively seen as a random sample of size np, and hence must be approximately normally distributed in one dimension.

The above premise, under the hypothesis of multivariate normality, allows us to use the tests of univariate normality for the above set of $n \times p$ random observations. A violation of the assumption of normality must show up in the behavior of the corresponding test statistic or the graphical representation. We thus call this method the "$n \times p$-sample" method.

A simulation study of empirical Type I errors of the "$n \times p$-sample" method, using univariate Mardia's skewness, Mardia's kurtosis, Shapiro-Wilk, and Anderson-Darling tests is presented in Figure 3.1. Number of replicates is $m = 4000$. The x-axis represents dimension p, ranging from 2 to 10. The y-axis gives information on sample size n, having values from 24 to 250. The estimated Type I error corresponding to each pair of the dimensions and the sample size is presented by the strength of color in that region. The lighter region gives a lower Type I error. As the dimension increases, the "$n \times p$-sample" method requires larger sample sizes to ensure low Type I error. As shown in these figures, given a dimension p, compared to the Mardia's skewness and Anderson Darling tests, the Mardia's kurtosis and Shapiro-Wilk require a bigger sample to correctly accept the null

hypothesis. Hence, we conclude that Mardia's skewness and Anderson-Darling test statistic have better control of Type I error.

3.3 Illustrative Examples for "$n \times p$-sample" Method: Simulated Data

We illustrate multivariate normality assessment for the following situations:

(i) Multivariate normal $N_5(\mathbf{0}_5, \mathbf{I}_5)$.

(ii) Multivariate normal $N_5\left(\boldsymbol{\mu} = [1, 2, 3, 4, 5]', \boldsymbol{\Sigma} = 2\big(.7\mathbf{I}_5 + .3\mathbf{J}_5{}^{1}\big)\right)$.

(iii) Mixture of two normal $.5N_5(\mathbf{0}_5, \mathbf{I}_5) + .5N_5(3 \times \mathbf{1}_5, 4 \times \mathbf{I}_5)$.

(iv) A five-dimensional probability distribution with each marginal pdf Gamma$(\alpha = 9, \beta = .5)$ and dependence is represented by the Gaussian copula with $\boldsymbol{\Sigma} = 2\big(.7\mathbf{I}_5 + .3\mathbf{J}_5\big)$.

(v) A multivariate Lomax distribution with $\alpha = 1, \beta = \mathbf{1}_5$.

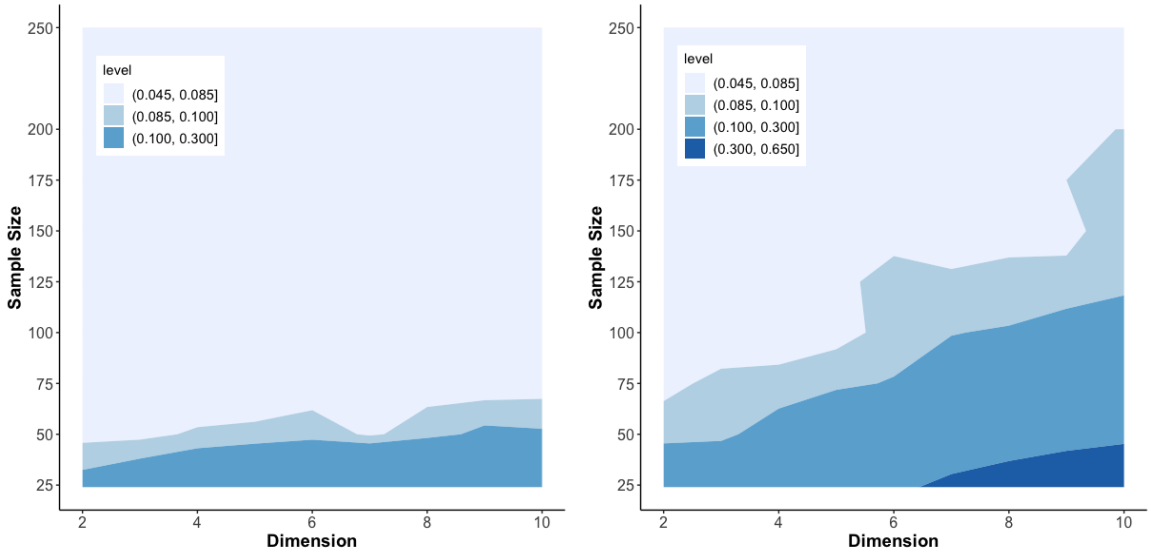
(vi) A multivariate t-distribution in five dimensions with $df = 5$.

(vii) A multivariate t-distribution in five dimensions with $df = 15$.

(viii) A multivariate Kotz distribution [57] with $\boldsymbol{\mu} = \mathbf{0}_5$ and $\boldsymbol{\Sigma} = \mathbf{I}_5$.

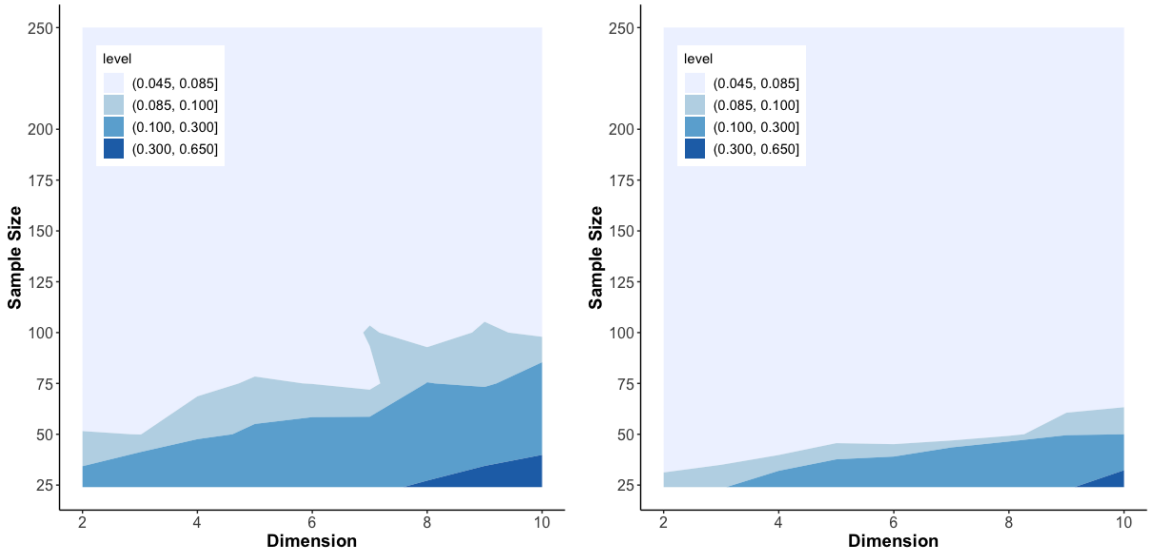
Note that among the non-normal distributions considered above, the last three distributions (vi), (vii), and (viii) are symmetric while the first three (iii), (iv), and (v) are skewed. In each case, a random sample of size $n = 200$ from the corresponding population is generated. For each of simulated data set, we will perform the transformations $\boldsymbol{S}_{-k}^{-1/2}\left(\mathbf{X}_k - \bar{\mathbf{X}}_{-k}\right)$, $k = 1, 2, \ldots, n$, and then test for the normality based on

[1] $\mathbf{J}_p$ is $p \times p$ matrix with all element is 1.



(a) Mardia's skewness test

(b) Mardia's kurtosis test



(c) Shapiro-Wilk test

(d) Anderson-Darling test

Figure 3.1: Empirical Type I error of "$n \times p$-sample" method.

$n \times p = 200 \times 5 = 1000$ univariate data points using the graphical methods[2] introduced in Chapter 2. We will also apply other hypothesis testing approaches with a significant level $\alpha = 0.05$. Specifically, these methods are Mardia's skewness, Mardia's kurtosis, Shapiro-Wilk, and Anderson - Darling tests.

Figures 3.2 and 3.3 correspond to the cases when data are from multivariate normal distributions. As we can see, Q-Q plots show a linear trend, which indicates the proper correspondence of empirical and theoretical $N(0, 1)$ quantiles. In other words, they approximately coincide. Moreover, the recoveries of score functions in the two scenarios are similar to the score function of a $N(0, 1)$ random variable Y with $\psi(y) = y$. The T_3 and T_4 plots show curves around the x-axis with almost zero gradients.

Figures 3.4, 3.5 and 3.6 present graphical tests on samples simulated from the three asymmetric distributions listed above. In the case of Mixture Normal, only the T_3 plot in Figure 3.4c shows strong evidence to reject the hypothesis of normality. Even though T_4 plot appears to be in the 95% acceptance region, the huge decrease in the tails suggests that this data may not be from a univariate normal distribution, and hence, the original multivariate data are not from a multivariate normal distribution. In contrast, although the Q- Q plot and score plot suggest some hint of non-normality, the evidence is not as convincing. In Figures 3.5 and 3.6, the departures from normality for the data from a population with Gaussian copula and marginal distributions as Gamma($\alpha = 9, \beta = .5$), and for the data from multivariate Lomax are self-evident.

In the examples of non-normal yet symmetric distributions, the four graphical methods perform very well when data are from t-distribution with 5 degrees of freedom (Figure 3.7). The Q-Q plots and score function also do not give rise to any linear trend. The T_3 and T_4 plots show very large values of $\hat{T}^3(t)$ and $\hat{T}^4(t)$, thereby causing a huge

[2]Score estimator algorithm merges univariate data if their distance is smaller than .001. T_3 and T_4 plots use 95% confidence bands.

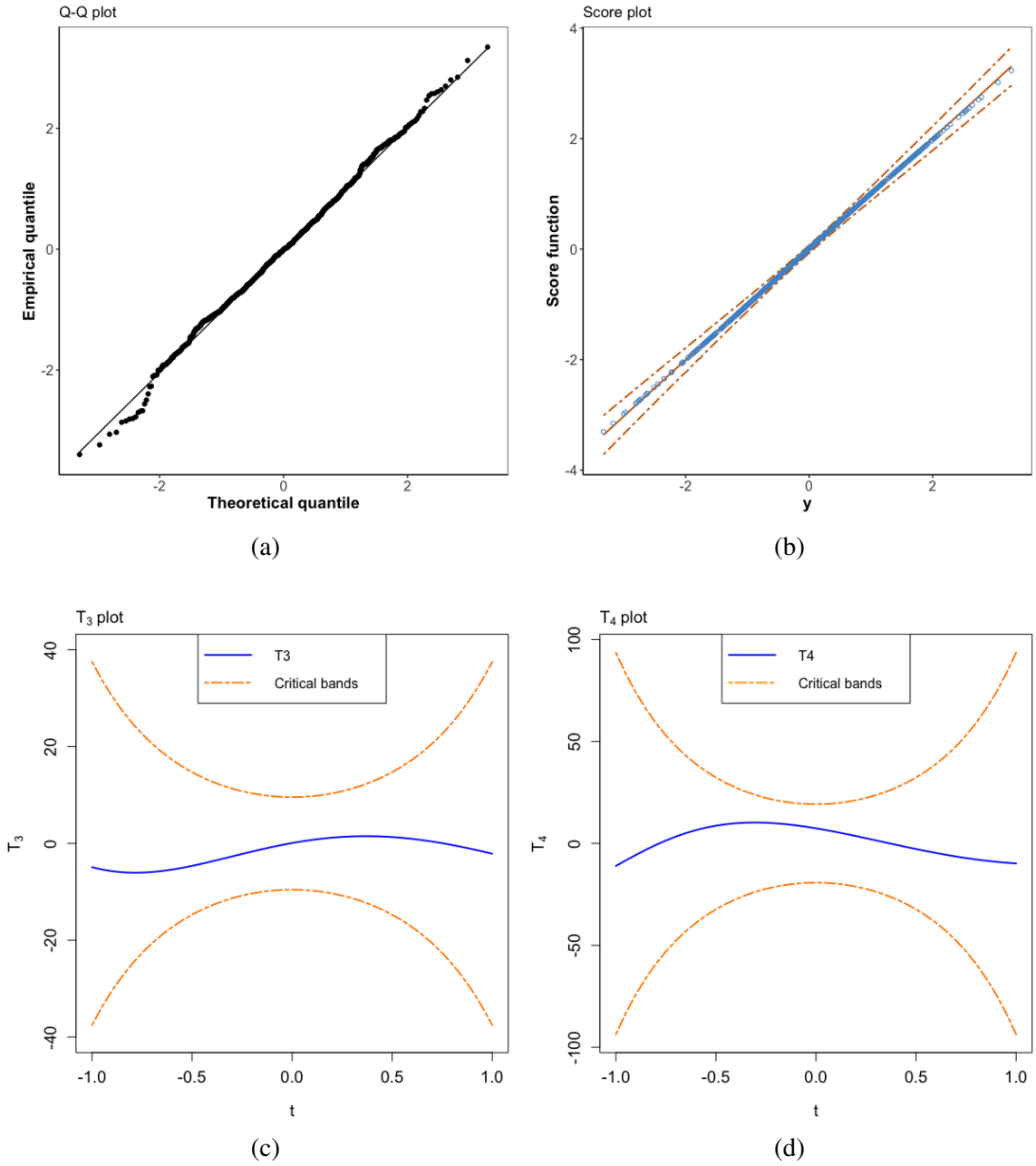
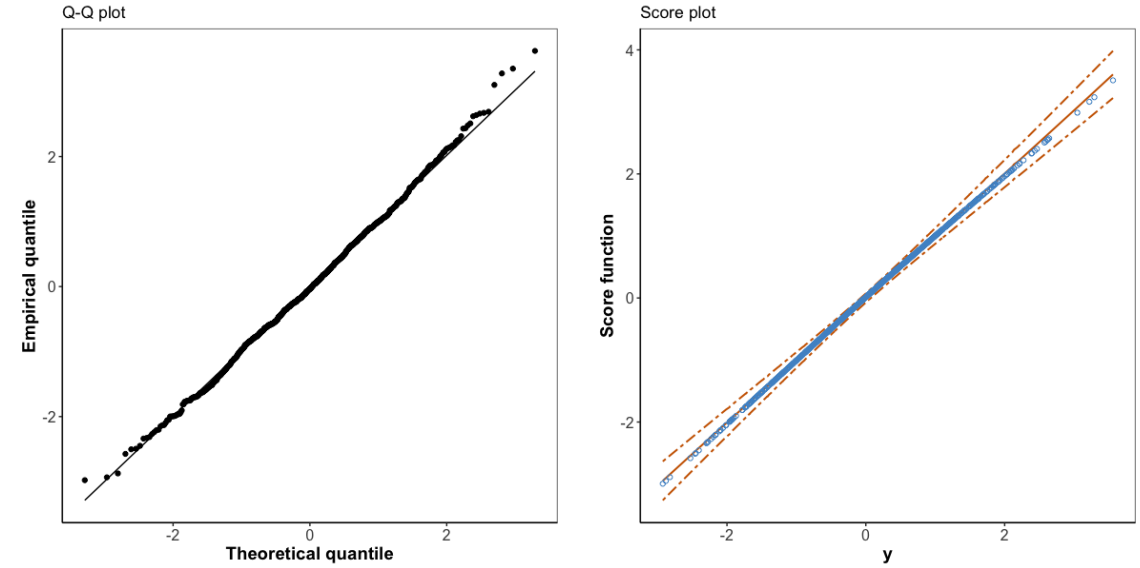
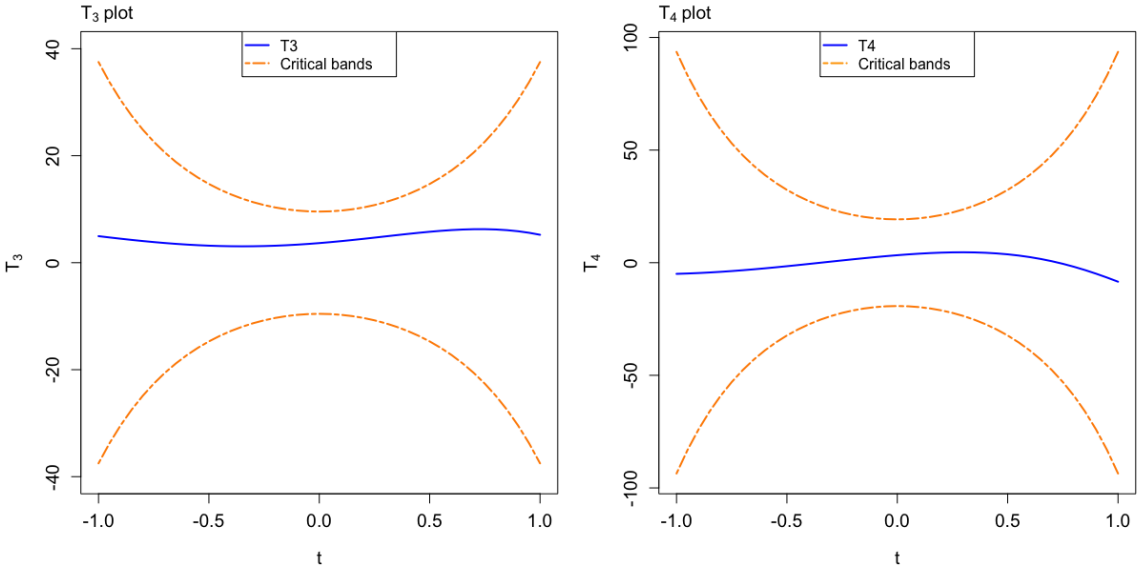


Figure 3.2: Graphical tests on "$n \times p$" univariate observations, data follow $N_5(\mathbf{0}_5, \mathbf{I}_5)$.

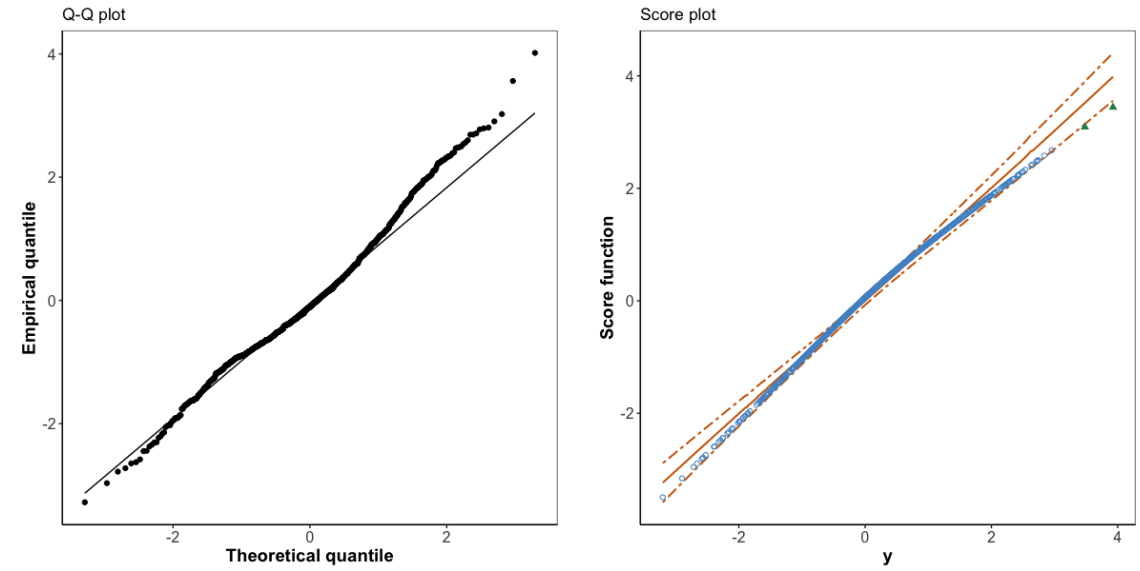


(a) (b)



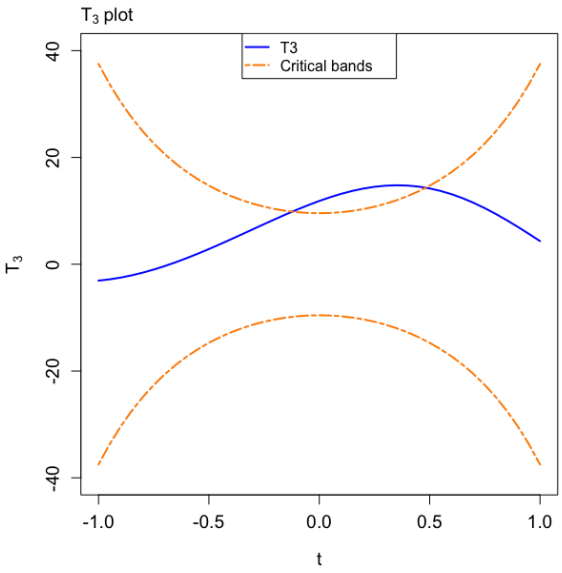
(c) (d)

Figure 3.3: Graphical tests on "$n \times p$" univariate observations, data follow $N_5\left(\boldsymbol{\mu} = [1, 2, 3, 4, 5]', \boldsymbol{\Sigma} = 2\left(.7\mathbf{I}_5 + .3\mathbf{J}_5\right)\right)$.

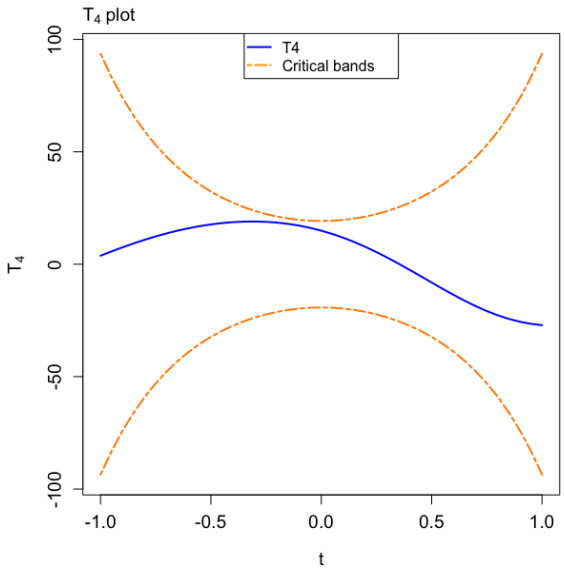


(a)

(b)

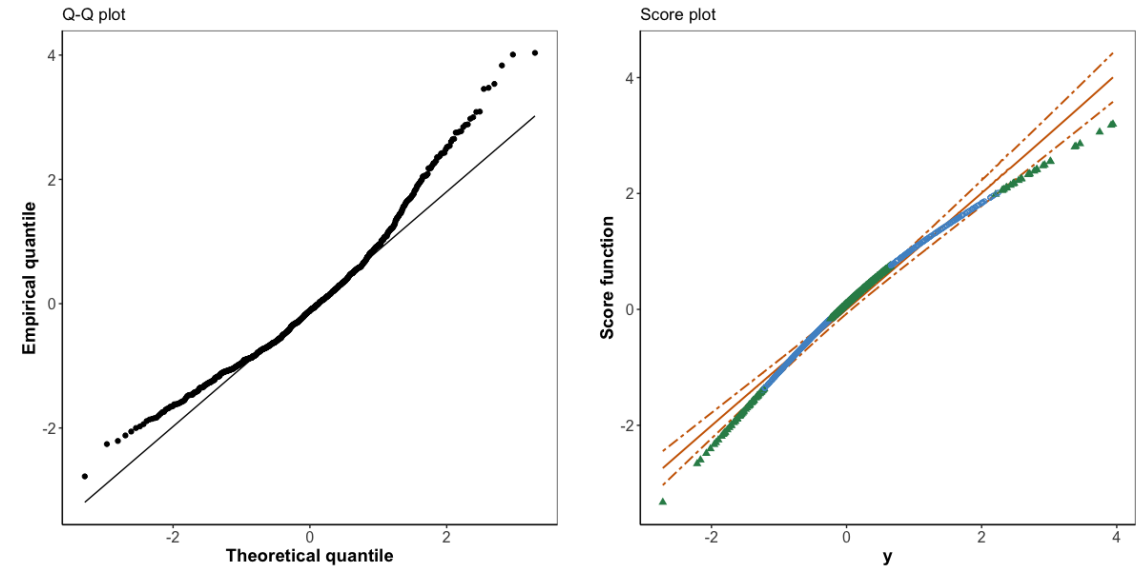


(c)

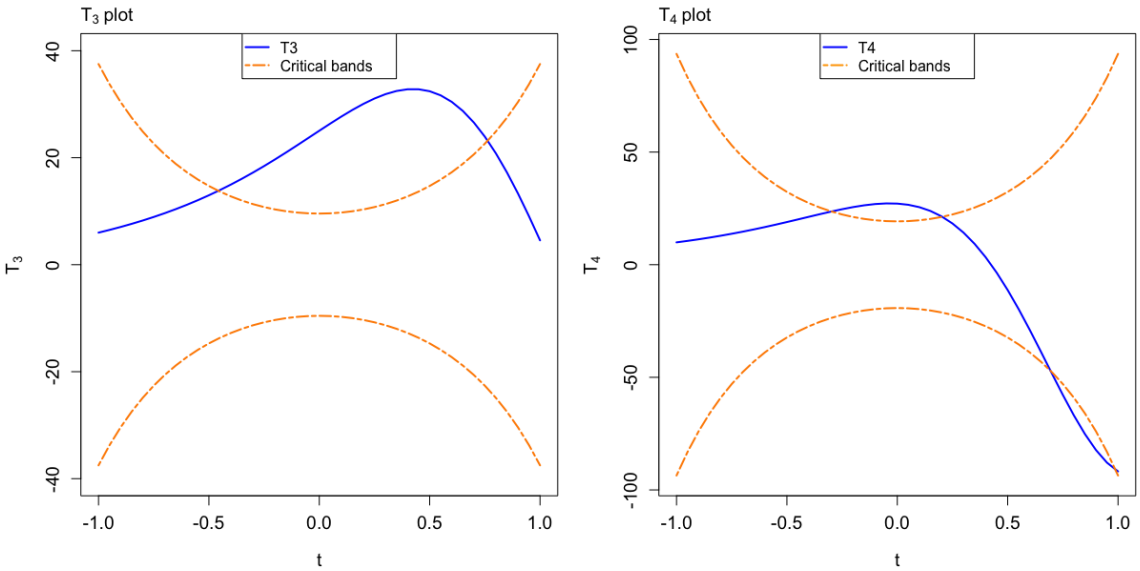


(d)

Figure 3.4: Graphical tests on "$n \times p$" univariate observations, data follow $.5N_5(\mathbf{0}_5, \mathbf{I}_5) + .5N_5(3 \times \mathbf{1}_5, 4 \times \mathbf{I}_5)$.

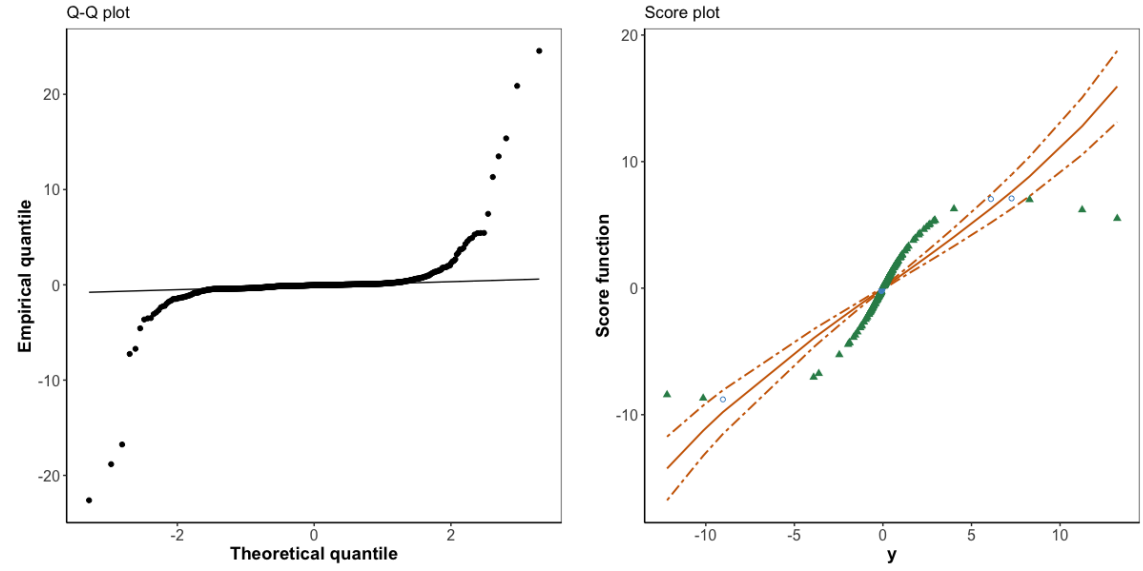


(a) (b)



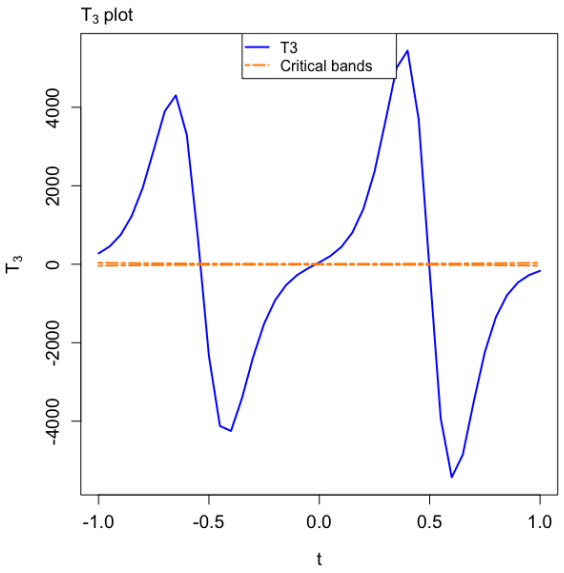
(c) (d)

Figure 3.5: Graphical tests on "$n \times p$" univariate observations, data follow a distribution with Gaussian copula $\mathbf{\Sigma} = 2\big(.7\mathbf{I}_5 + .3\mathbf{J}_5\big)$, and marginal pdf are Gamma$(\alpha = 9, \beta = .5)$.

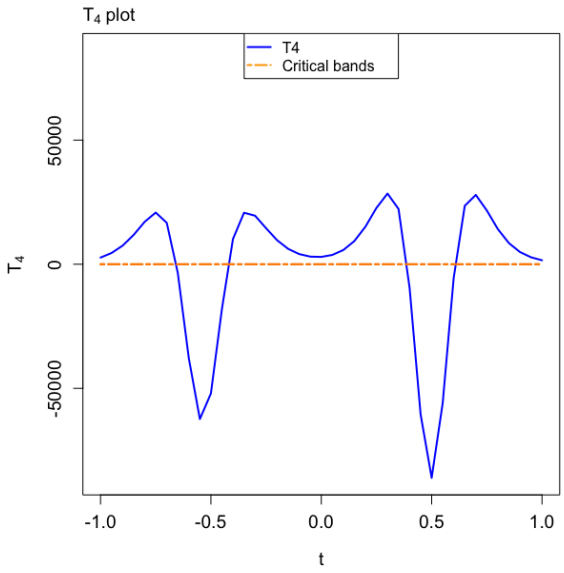


(a)

(b)



(c)

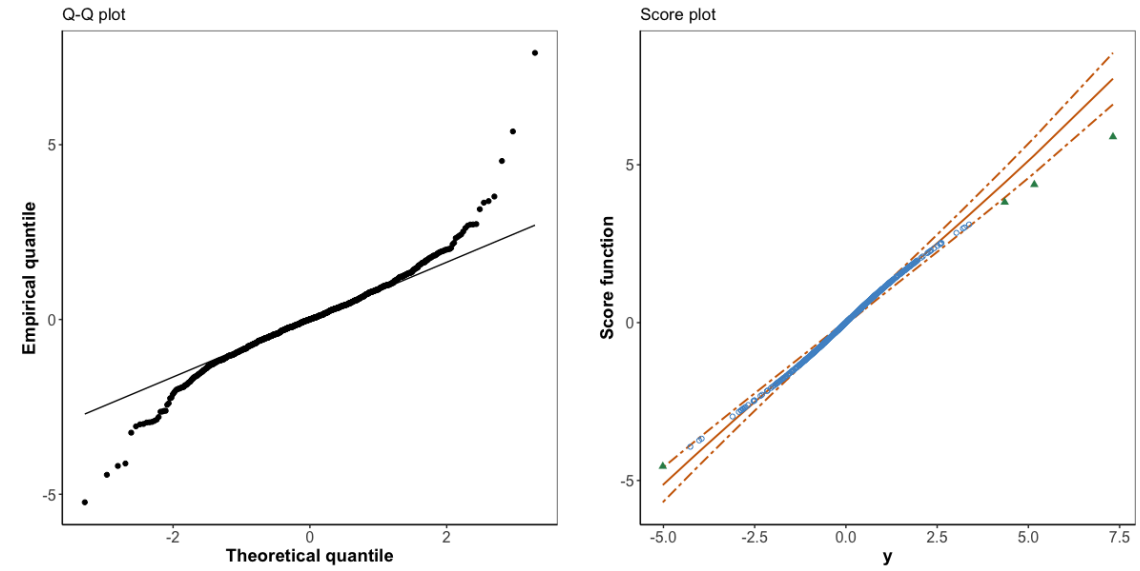


(d)

Figure 3.6: Graphical tests on "$n \times p$" univariate observations, data follow Lomax($\alpha = 1, \beta = \mathbf{1}_5$).

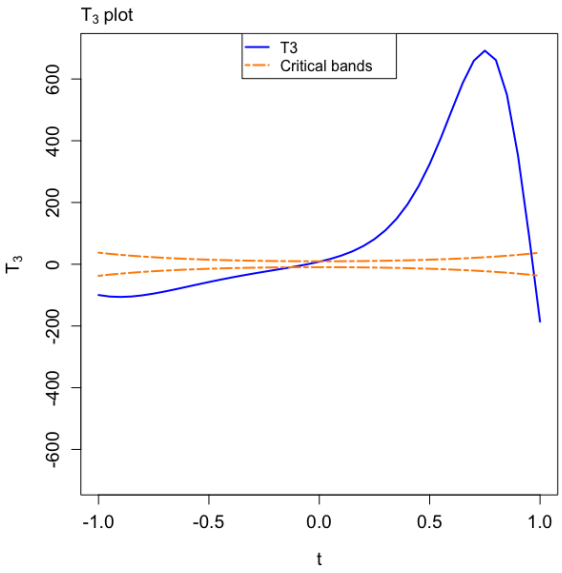
rescaling of the y-axis. However, when the degree of freedom for the t-distribution is increased to 15, as seen in Figure 3.8, the departure from normality isn't as dramatic. In fact, the score function shows more or less a straight line, with no score falling out of the bands. The Q-Q plot only shows systematic deviations (of underestimation and overestimation, respectively) at the two tails. The non-normality is apparent in the T_3 plot where the curve is increasing, and also in the T_4 plot where the graph is not only curved but also crosses the bands. Finally, Figure 3.9 shows evidence of non-normality of the simulated Kotz sample using other three graphical methods, except for the score plot in Figure 3.9b. The score function is almost linear and cannot be distinguished from the score function of a standard normal distribution. The T_3 and T_4 plots vividly convey the highly non-normal nature of the data.

Results from other hypothesis testing approaches based on $n \times p = 200 \times 5 = 1000$ univariate sample points for each of the above simulated examples are presented in Table 3.1. Values reported are p-values of the particular test (columns) for the corresponding simulated distribution (rows). All test statistics have large p-values for the two normally distributed samples, favoring the normality assumption. In all cases of asymmetric distributions, all tests result in small p-values, thereby suggesting the non-normality for the univariate $n \times p$ sample. This also suggests the non-normality of the original multivariate sample. When data are from symmetric yet non-normal distributions, the tests based on Mardia's skewness perform poorly for the cases of $t(15)$ and Kotz distributions. Other univariate tests, including Mardia's kurtosis, Shapiro-Wilk, and Arderson-Darling, all present solid evidence of non-normality.

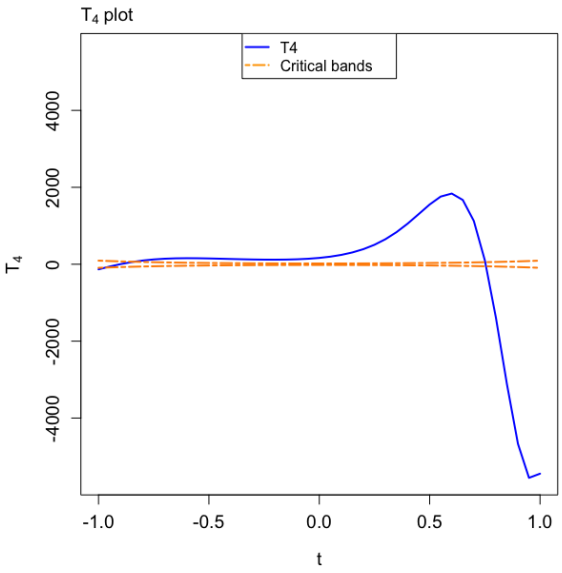


(a)

(b)

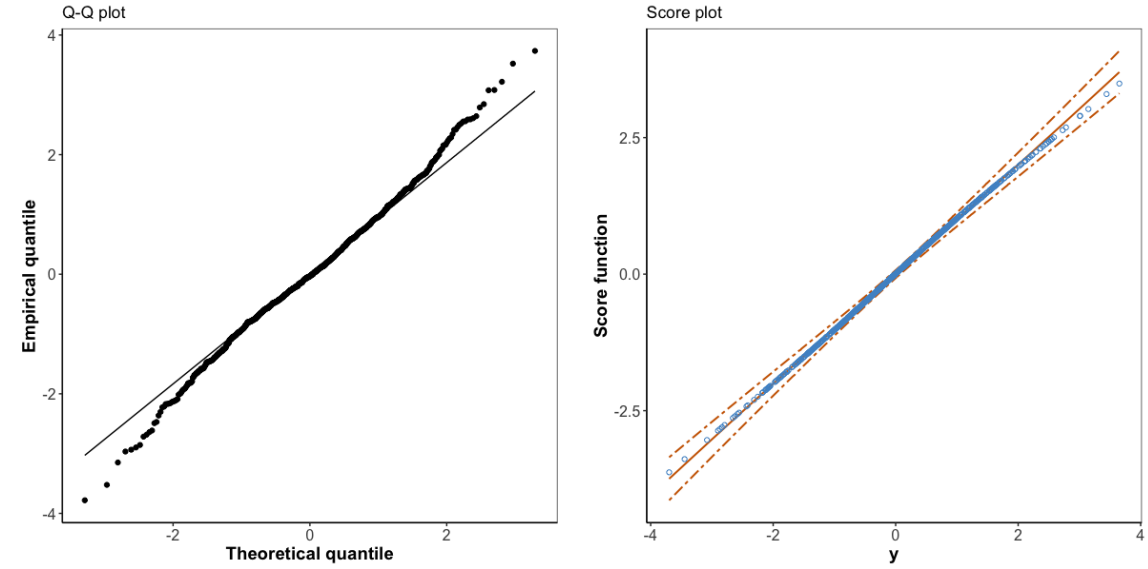


(c)



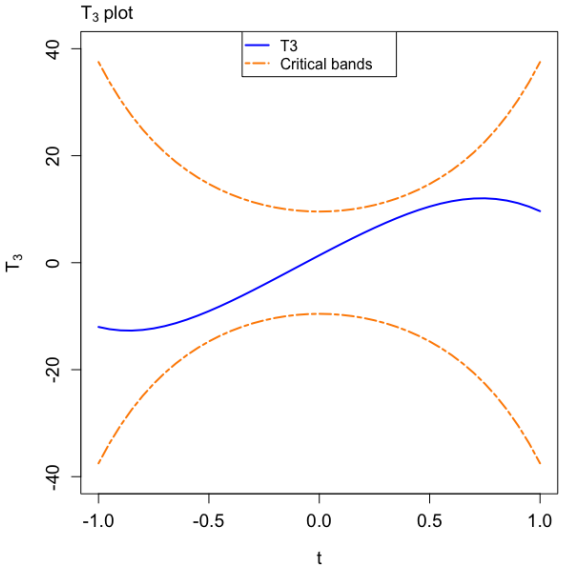
(d)

Figure 3.7: Graphical tests on "$n \times p$" univariate observations, data follow multivariate $t(5)$ distribution.

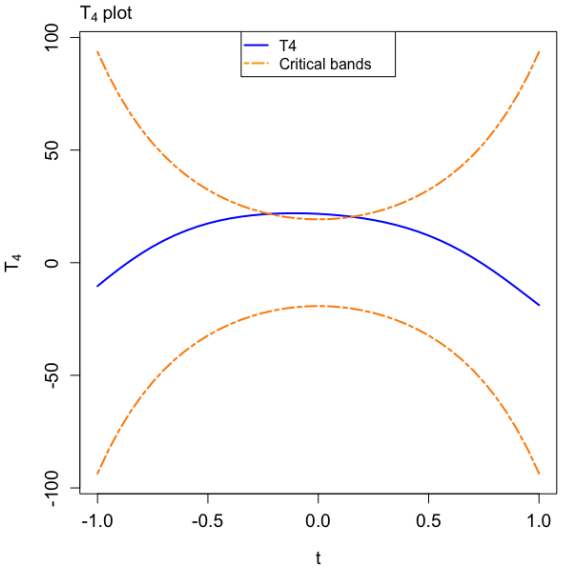


(a)

(b)

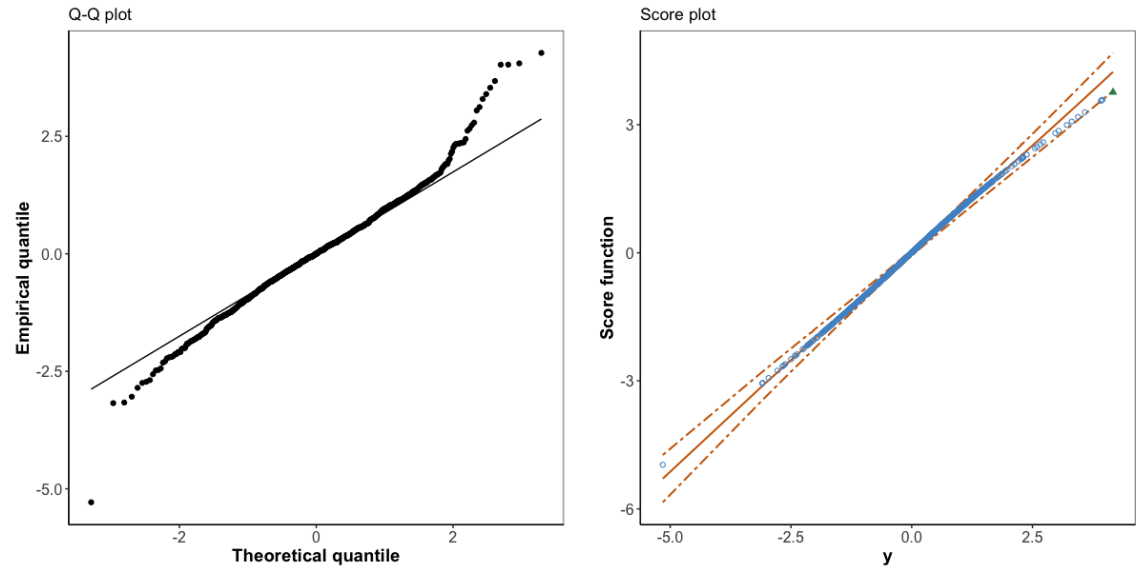


(c)



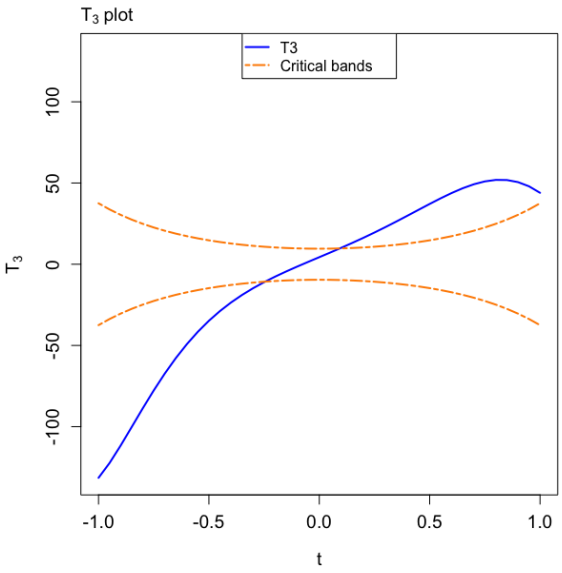
(d)

Figure 3.8: Graphical tests on "$n \times p$" univariate observations, data follow multivariate $t(15)$ distribution.

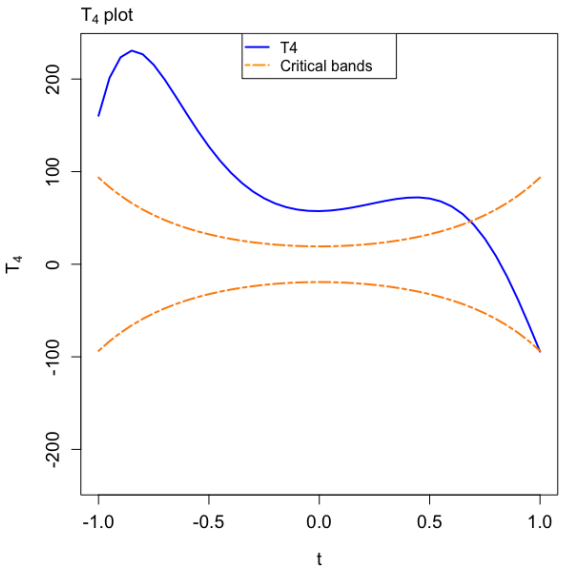


(a)

(b)



(c)



(d)

Figure 3.9: Graphical tests on "$n \times p$" univariate observations, data follow Kotz($\boldsymbol{\mu} = \mathbf{0}_5, \boldsymbol{\Sigma} = \mathbf{I}_5$).

Table 3.1 Formal hypothesis tests on "$n \times p$" univariate observations, simulated data.

Distributions		**Test Statistics**			
		Mardia's skewness	**Mardia's kurtosis**	**Shapiro-Wilk**	**Anderson-Darling**
Normal	Standard (i)	0.9742	0.1202	0.1071	0.1091
	Non-standard (ii)	0.1340	0.4719	0.3728	0.4273
Asymmetric	Mixture Normal (iii)	0	0.0020	0	0
	Gaussian Copulas (iv)	0	0	0	0
	Lomax (v)	0	0	0	0
Symmetric	$t(5)$ (vi)	0.0009	0	0	0
	$t(15)$ (vii)	0.5748	0	0.0014	0.0009
	Kotz (viii)	0.0762	0	0	0

3.4 Two Examples for "$n \times p$-sample" Method: Cork Data and Water Potability Data Revisited

In this section, we will apply the "$n \times p$-sample" approach to assess the multivariate normality assumption of the Cork data and the Water Potability (= Good) data, both of which have been discussed in Chapter 2.

Example 3.4.1 (Cork Data)**.** As discussed in Chapter 2, the Cork data consists of four-variate data where each is marginally distributed as normal. For the "$n \times p$-sample" method, we have $n \times p = 28 \times 4 = 112$ univariate observations. Figure 3.10a gives the Q-Q plot on 112 standardized observations and shows clear evidence of non-normality, as the empirical quantiles toward the right tail do not quite match with the theoretical quantiles of $N(0, 1)$. The score plot in Figure 3.10b has several scores systematically below the reference line. This clearly suggests the systematic underestimation of scores, hence presenting evidence of the departure from normality. Although the T_3 and T_4 are both within their respective bands, as shown in Figures 3.10c and 3.10d, the changes in the right tails of these curves suggest some evidence of non-normality.

Table 3.2 p-values for normality tests based on "$n \times p$-sample" method, Cork dataset.

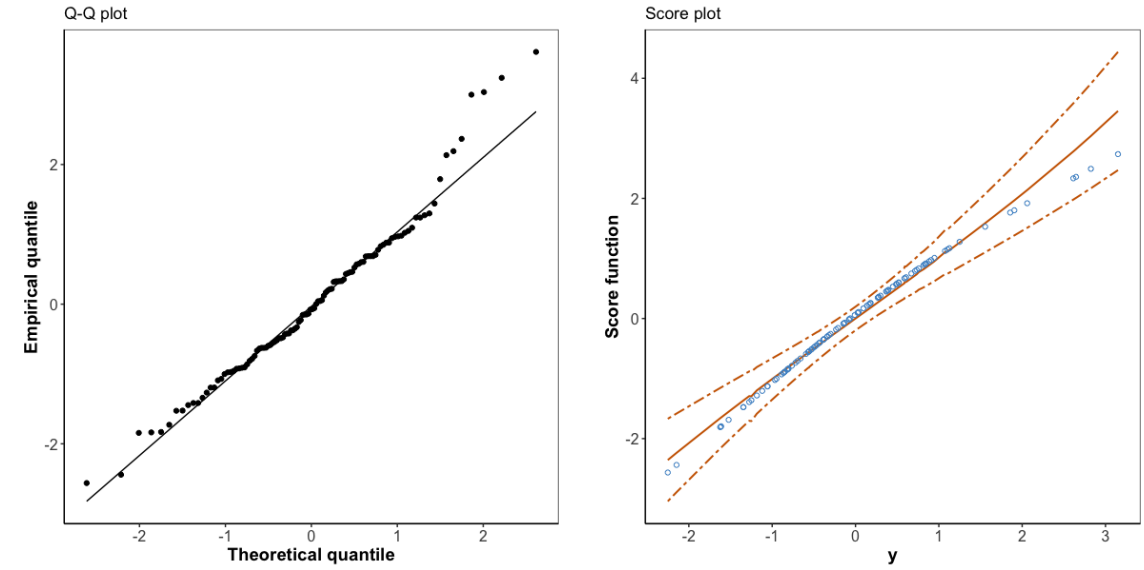
Test Statistics	**Mardia's skewness**	**Mardia's kurtosis**	**Shapiro-Wilk**	**Anderson-Darling**
p-value	0.008	0.0540	0.0144	0.0592

Table 3.3 p-values for normality tests based on "$n \times p$-sample" method, Water Potability (= Good) dataset; variables considered are *Organic carbon, Turbidity, Trihalomethanes*.

Test Statistics	**Mardia's skewness**	**Mardia's kurtosis**	**Shapiro-Wilk**	**Anderson-Darling**
p-value	0.5220	0.1462	0.7074	0.8317

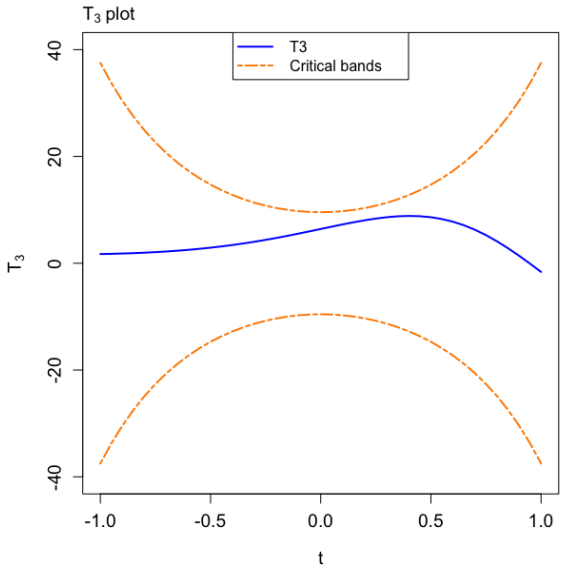
Table 3.2 presents the p-values of four univariate normality tests on these 112 observations. The results are highly debatable when Mardia's kurtosis and Anderson Darling tests seem to support the normality assumption, but Mardia's skewness and Wilk-Shaprio tests provide evidence to reject the hypothesis.

Example 3.4.2 (Water Potability (= Good))**.** After individual exploration of each exploratory variable for the Water Potability data (for water potability = Good, $n = 811$) in Chapter 2, the variables *Organic carbon, Turbidity, Trihalomethanes* were found to be marginally normally distributed. We will use the "$n \times p$-sample" method to test the hypothesis that these three variables are also jointly normal. We have $n \times p = 811 \times 3 = 2433$ univariate observations. Figure 3.11 presents the graphical assessment of this sample. The evidence of trivariate normality is quite solid in all three graphical methods, except the score plot in Figure 3.11b. Points are dropping out of the linear trend in the right tails, suggesting the underestimation of some scores of the extreme values observations. All four hypothesis testing methods give large p-values (Table 3.3).

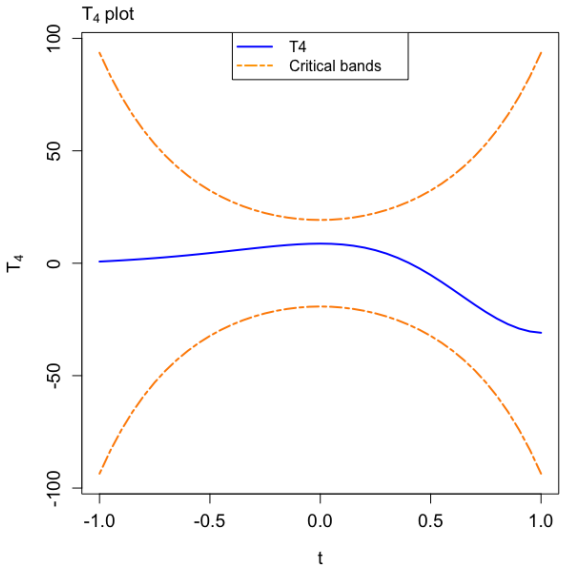


(a)

(b)



(c)



(d)

Figure 3.10: Graphical tests on "$n \times p$" univariate observations, Cork dataset.

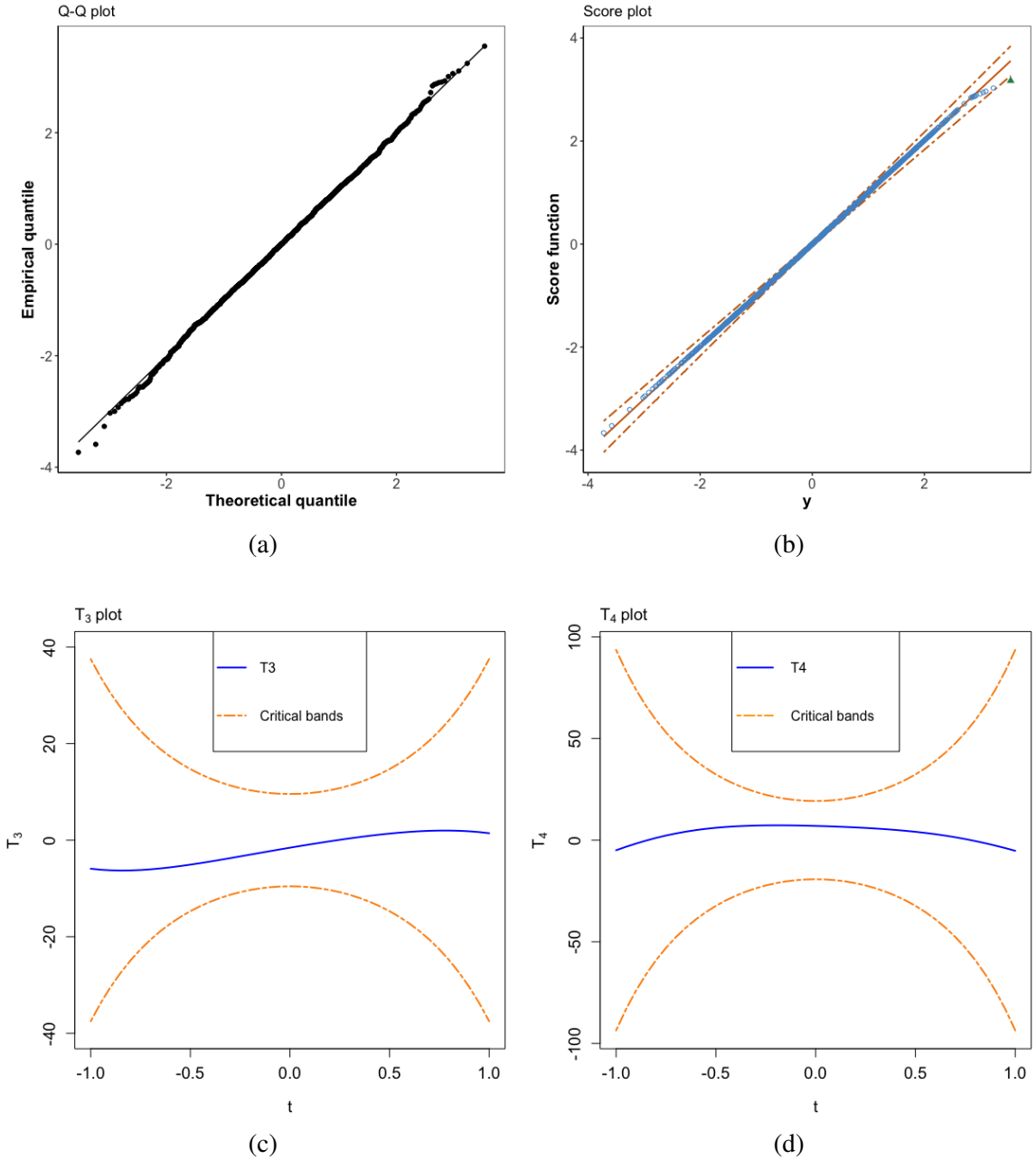


Figure 3.11: Graphical tests on "$n \times p$" univariate observations, Water Potability (= Good) dataset; variables considered are *Organic carbon, Turbidity, Trihalomethanes.*

3.5 Outlier Detection Using Score Function

In Section 2.2, we introduced the 2-sigma bands for the score plot to detect outliers when most of the data are from a univariate normal distribution. Although the data here are multivariate, we have transformed them into a univariate sample of size $n \times p$. Thus, the multivariate outlier detection procedures can be implemented based on this $n \times p$ sample, and accordingly, the corresponding multivariate observation can be identified. We illustrate this by using the simulated data from a five-dimensional standard normal distribution with a sample of size $n = 200$. To evaluate various scenarios, we will replace two observations by the following types of outliers:

(i) Two outliers from $N_5(2 \times \mathbf{1}_5, \mathbf{I}_5 + 3 \times \mathbf{J}_5)$.

(ii) Two outliers, one is generated from $N_5(2 \times \mathbf{1}_5, \mathbf{I}_5 + 3 \times \mathbf{J}_5)$, and another is generated from $N_5(-2 \times \mathbf{1}_5, \mathbf{I}_5 + 3 \times \mathbf{J}_5)$.

(iii) Two outliers randomly generated from $N_5(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ where $\boldsymbol{\mu}_1$ and $\boldsymbol{\Sigma}_1$ are given below

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0 & 0 & 0 & 6 & 6 \end{bmatrix}' \text{ and } \boldsymbol{\Sigma}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 3 \\ 0 & 0 & 0 & 3 & 4 \end{bmatrix}$$

Note that the parameters of the first three components of the above distribution are identical to those of the true distribution $N_5(\mathbf{0}_5, \mathbf{I}_5)$. Hence, any cause of outlyingness lies only in the last two components.

(iv) Two outliers are randomly generated from multivariate Lomax$(\alpha = 1, \beta = \mathbf{1}_5)$ distribution. Note that this distribution does not have a finite mean or finite covariance matrix [58].

Figure 3.12a corresponds to case (i), when the two outliers are randomly generated and the replace the 105^{th} and 178^{th} observations of the original data. To be more specific, the two outliers are[3]

$$\begin{bmatrix} \tilde{\mathbf{X}}'_{105} \\ \tilde{\mathbf{X}}'_{178} \end{bmatrix} = \begin{bmatrix} -3.391 & -2.714 & -2.905 & -3.669 & -4.902 \\ 0.853 & -1.698 & 0.012 & 0.404 & -0.143 \end{bmatrix}.$$

As we can see, in the left tail of the score plot, there is a triangular symbol falling out of the 2-sigma bands. It is considered a (univariate) outlier. In fact, this outlier is a component of $\tilde{\mathbf{X}}_{105}$, which is one of the simulated outliers.

Figure 3.12b illustrates the scenario (ii), and the two randomly generated outliers replace the 36^{th} and 103^{th} observations. These outlying observations are

$$\begin{bmatrix} \tilde{\mathbf{X}}'_{36} \\ \tilde{\mathbf{X}}'_{103} \end{bmatrix} = \begin{bmatrix} 2.563 & 3.473 & 2.905 & 1.931 & 2.946 \\ -5.145 & -3.576 & -5.119 & -5.416 & -5.139 \end{bmatrix}.$$

These observations clearly have very large components compared to what one should anticipate in the standard normal data. These two multivariate outliers result in four univariate observations (over a sample of size 1000), which are $-5.145, -5.119, -5.416$ and -5.139, distributed in the very extreme left tail of $N(0,1)$, hence the score plot in Figure 3.12b give an indication of non-normality.

In the case of (iii), the outliers are

$$\begin{bmatrix} \tilde{\mathbf{X}}_{185} \\ \tilde{\mathbf{X}}_{92} \end{bmatrix} = \begin{bmatrix} -0.687 & 0.461 & 0.129 & 6.054 & 3.849 \\ -0.446 & -1.265 & 1.715 & 5.619 & 5.520 \end{bmatrix}.$$

Under the standard univariate normality assumption, the interval $[-3, 3]$ should contain about 99.7% observations. However, although the first three components of the above observations are around 0, the last two are large and outside of the interval $[-3, 3]$. This

[3]Data is rounded at 3 decimal numbers.

effect is manifested in the score plot, as there are three univariate outliers in the right tail, which can be traced as components of observations $\tilde{\mathbf{X}}_{92}$ and $\tilde{\mathbf{X}}_{185}$.

Finally, Figure 3.12d illustrates the scenario when outliers are

$$\begin{bmatrix} \tilde{\mathbf{X}}_{51} \\ \tilde{\mathbf{X}}_{128} \end{bmatrix} = \begin{bmatrix} 0.637 & 0.183 & 1.105 & 11.232 & 0.315 \\ 0.268 & 0.549 & 0.124 & 0.421 & 0.552 \end{bmatrix},$$

which are generated from Lomax($\alpha = 1, \beta = \mathbf{1}_5$). In this case, all empirical scores except one lie in the lower corner. The one observation on the middle right and off from the linear trend corresponds to $\tilde{X}_{51;3} = 11.232$. Thus the procedure identifies $\tilde{X}_{51}$ as the outlying observation but not the $\tilde{X}_{128}$.

3.6 Tests Based on Best Linear Transformation

In this section, we will introduce tests for multivariate normality, which make use of the well-known "dropping" theorem stated below.

Theorem 3.1 ([60])**.** *Let* $\mathbf{X}$ *be a* p*-dimensional random vector.* $\mathbf{X}$ *is* $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ *iff for every* $l \neq 0$*, not depending on* $\mathbf{X}$*,* $l'\mathbf{X} \sim N(l'\boldsymbol{\mu}, l'\boldsymbol{\Sigma}l)$.

Let $Z \sim N(\mu, \sigma^2)$, then it is well-known that

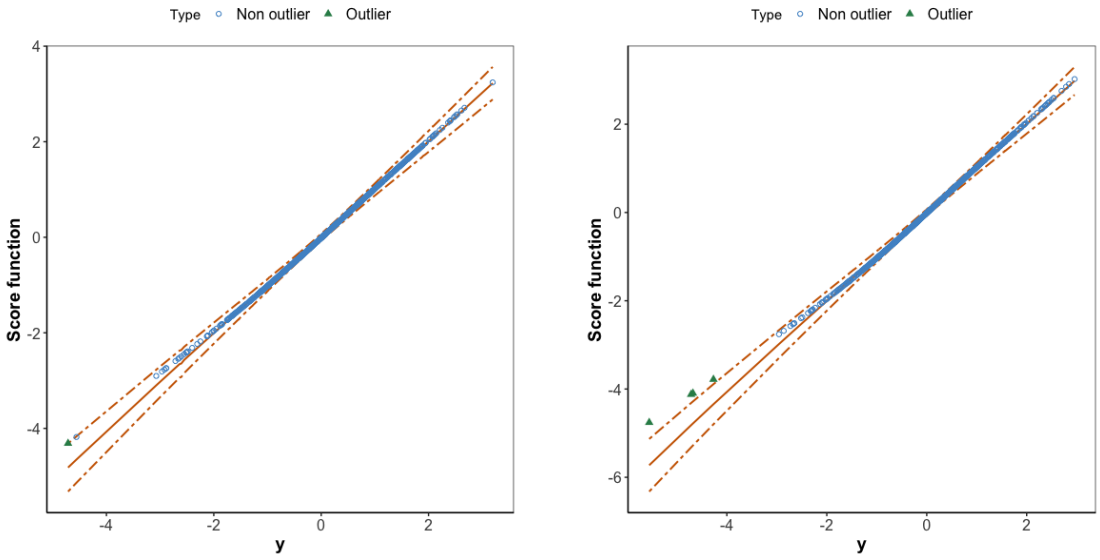
$$\nu_1 = \frac{\mathbb{E}(Z-\mu)^3}{\left[\mathbb{E}(Z-\mu)^2\right]^{\frac{3}{2}}} = 0 \text{ and } \nu_2 = \frac{\mathbb{E}(Z-\mu)^4}{\left[\mathbb{E}(Z-\mu)^2\right]^2} = 3.$$

For a random variable Y, with mean μ, its coefficients of skewness and kurtosis are defined respectively as

$$\kappa_1 = \frac{\mathbb{E}(Y-\mu)^3}{\left[\mathbb{E}(Y-\mu)^2\right]^{\frac{3}{2}}} - \nu_1 \text{ and } \kappa_2 = \frac{\mathbb{E}(Y-\mu)^4}{\left[\mathbb{E}(Y-\mu)^2\right]^2} - \nu_2.$$

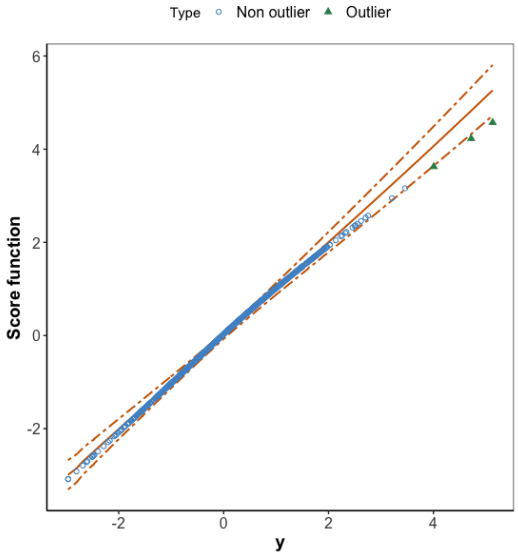
Given a random sample of size n, $Y_1, Y_2, \ldots, Y_n$, the quantities κ_1 and κ_2 can be estimated by

$$\hat{\kappa}_1 = \frac{\frac{1}{n-1}\sum_{i=1}^n (Y_i - \bar{Y})^3}{\left[\frac{1}{n-1}\sum_{i=1}^n (Y_i - \bar{Y})^2\right]^{3/2}} - \nu_1 = \frac{\frac{1}{n-1}\sum_{i=1}^n (Y_i - \bar{Y})^3}{S_Y^3} - \nu_1 = \frac{1}{n-1}\sum_{i=1}^n \left(\frac{Y_i - \bar{Y}}{S_Y}\right)^3 - \nu_1,$$

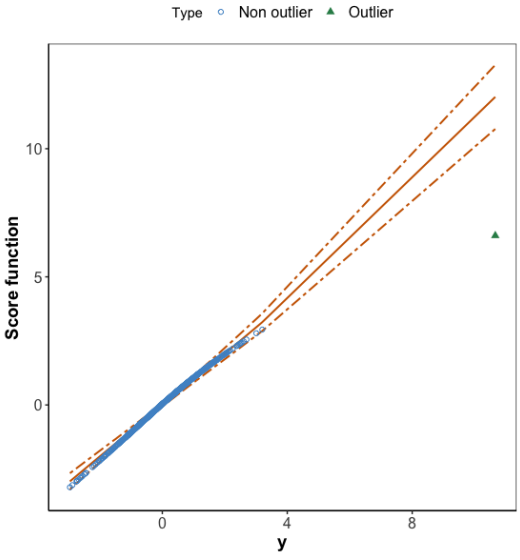


(a)

(b)



(c)



(d)

Figure 3.12: Score plots on "$n \times p$" univariate observations, data follow $N_5(\mathbf{0}_5, \mathbf{I}_5)$ with two outliers; (a) corresponds to scenario (i), (b) corresponds to scenario (ii), (c) corresponds to scenario (iii), corresponds to scenario (iv).

and

$$\hat{\kappa}_2 = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^4}{\left[\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2\right]^2} - \nu_2 = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^4}{S_Y^4} - \nu_2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(\frac{Y_i - \bar{Y}}{S_Y}\right)^4 - \nu_2.$$

where $\bar{Y}$ and S_Y^2 are the univariate sample mean and sample variance, respectively.

Clearly for a univariate normal distribution $\kappa_1 = \kappa_2 = 0$. Thus for the multivariate random vector $\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, this must be true for any linear transformation $l'\mathbf{X}$, $l \neq \mathbf{0}_p$, as well. For any multivariate random vector $\mathbf{X}$, we thus have the skewness and kurtosis of $l'\mathbf{X}$ as,

$$\kappa_{1l} = \frac{\mathbb{E}\left[l'\mathbf{X} - l'\boldsymbol{\mu}\right]^3}{(l'\boldsymbol{\Sigma}l)^{3/2}} - \nu_1 = \frac{\mathbb{E}\left[l'(\mathbf{X} - \boldsymbol{\mu})\right]^3}{(l'\boldsymbol{\Sigma}l)^{3/2}} - \nu_1 = \mathbb{E}\left[\frac{l'(\mathbf{X} - \boldsymbol{\mu})}{\sqrt{(l'\boldsymbol{\Sigma}l)^{1/2}}}\right]^3 - \nu_1$$

and

$$\kappa_{2l} = \frac{\mathbb{E}\left[l'\mathbf{X} - l'\boldsymbol{\mu}\right]^4}{(l'\boldsymbol{\Sigma}l)^2} - \nu_2 = \frac{\mathbb{E}\left[l'(\mathbf{X} - \boldsymbol{\mu})\right]^4}{(l'\boldsymbol{\Sigma}l)^2} - \nu_2 = \mathbb{E}\left[\frac{l'(\mathbf{X} - \boldsymbol{\mu})}{\sqrt{(l'\boldsymbol{\Sigma}l)^{1/2}}}\right]^4 - \nu_2.$$

The subscript l is used to indicate that these expressions depend on the vector $l \in \mathbb{R}^p$. Now, let

$$\varkappa_k^2 = \max_{l \in \mathbb{R}^p} \kappa_{kl}^2, k = 1, 2.$$

Clearly, κ_{kl}^2 is unknown, as the population parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are unknown. However, an estimator for κ_{kl}^2 can be obtained by substituting $\bar{\mathbf{X}}$ and $\mathbf{S}$ in the places of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively. Specifically, from random sample of size n, $\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_n$, and for a fixed vector $l \in \mathbb{R}^p$, an estimator for κ_{kl} is

$$\hat{\kappa}_{kl} = \frac{1}{(n-1)}\sum_{i=1}^{n}\left(\frac{l'(\mathbf{X}_i - \bar{\mathbf{X}})}{(l'\mathbf{S}l)^{1/2}}\right)^{(k+2)} - \nu_k, k = 1, 2.$$

Hence a natural estimator of $\varkappa_k^2$ is

$$\hat{\varkappa}_k^2 = \max_{l \in \mathbb{R}^p} \hat{\kappa}_{kl}^2 = \max_{l \in \mathbb{R}^p} \left[\frac{1}{(n-1)} \sum_{i=1}^{n} \left(\frac{l'(X_i - \bar{X})}{(l'\mathbf{S}l)^{1/2}} \right)^{k+2} - \nu_k \right]^2 \tag{3.1}$$

$$= \max_{l'\mathbf{S}l=1} \left[\frac{1}{(n-1)} \sum_{i=1}^{n} \left[l'(\mathbf{X}_i - \bar{\mathbf{X}}) \right]^{k+2} - \nu_k \right]^2 .$$

Theorem 3.1 ensures the normality of $l'\mathbf{X}$ for any vector $l \in \mathbb{R}^p$, whenever $\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Under the assumption of normality, the κ_{kl} above is 0, for any vector $l \in \mathbb{R}^p$, $l \neq \mathbf{0}_p$. This observation implies that under multivariate normality, the corresponding estimator $\hat{\varkappa}_k^2$ in equation (3.1) may be close 0, for $k = 1, 2$.

To solve the problem in (3.1), we resort to the Cholesky decomposition of sample covariance matrix $\mathbf{S} = \Gamma\Gamma'$. By defining $a = \Gamma l$, we alternatively minimize the function

$$G_k(a) = -\left[\frac{1}{(n-1)} \sum_{i=1}^{n} \left[a'(\Gamma^{-1})'(\mathbf{X}_i - \bar{\mathbf{X}}) \right]^{k+2} - \nu_k \right]^2 ,$$

subject to $1 = l'Sl = a'a$. Thus, with $\tilde{\mathbf{X}}_i = \Gamma^{-1}(\mathbf{X}_i - \bar{\mathbf{X}}), i = 1, 2, \ldots n$ and by replacing a by the generic notation l, the problem (3.1) becomes

$$\hat{\varkappa}_k^2 = \min_{l'l=1} G_k(l) = \min_{l'l=1} \left\{ - \left[\frac{1}{(n-1)} \sum_{i=1}^{n} \left(l'\tilde{\mathbf{X}}_i \right)^{k+2} - \nu_k \right]^2 \right\}, k = 1, 2. \tag{3.2}$$

No closed-form solution to the above is available. However, the problem (3.2) is a minimization of a continuous function on the surface of a unit ball. Hence, the gradient projection method can be used to obtain a numerical solution, see Algorithm 3.1 given in [61] below. The notation $P_c(\gamma)$ in the Algorithm 3.1 is the projection of the vector γ onto the set C. In the current context, the set C is the surface of the unit ball defined as $B = \{l \in \mathbb{R}^p : l'l = 1\}$. Lemma 3.2 gives a formula for this projection.

Algorithm 3.1: Gradient Projection

Data: $m \geq 1, l_0 \in \mathbb{R}^p, t > 0, \epsilon > 0$
$l \leftarrow l_0$;
err $\leftarrow 1$;
$i = 0$;
while $i \leq m$ *or* $err > \epsilon$ **do**
  $\gamma \leftarrow l - t * \nabla G(l)$;
  $l_1 \leftarrow P_c(\gamma)$;
  err $\leftarrow \|l - l_1\|$;
  $i \leftarrow i + 1$;
  $l \leftarrow l_1$;
end

Lemma 3.2 ([61])**.** Projection of $\gamma \in \mathbb{R}^p$ onto the surface of the unit ball $B = \{l \in \mathbb{R}^p : l'l = 1\}$ is

$$P_B(\gamma) = \begin{cases} \gamma, \text{ if } \gamma'\gamma = 1 \\ \dfrac{\gamma}{|\gamma|}, \text{ otherwise.} \end{cases}$$

Moreover, Algorithm 3.1 also requires the gradients of the subjective function $G_k(l), k = 1, 2$, which are as follow

$$\nabla G_k(l) = -2\left[\frac{1}{(n-1)}\sum_{i=1}^{n}\left(l'\tilde{\mathbf{X}}_i\right)^{k+2} - \nu_k\right]\left[\frac{k+2}{(n-1)}\sum_{i=1}^{n}(l'\tilde{\mathbf{X}}_i)^{k+1}\tilde{\mathbf{X}}_i\right].$$

Remark (A faster method to solve for $G_1(l)$). We first notice that $\hat{\kappa}_{1l}$ is an odd function of $l \in \mathbb{R}^p$, that is $\hat{\kappa}_{1(-l)} = -\hat{\kappa}_{1l}$, for every $l \in \mathbb{R}^p$. Thus, for any two vector l_1 and l_2 such that $\hat{\kappa}_{1l_1} \leq \hat{\kappa}_{1l_2}$, we have $-\hat{\kappa}_{1l_1} \geq -\hat{\kappa}_{1l_2}$, i.e, $-\hat{\kappa}_{1(l_1)} \geq \hat{\kappa}_{1(-l_2)}$.

Suppose l^* is a minimizer of $\hat{\kappa}_{1l}$. Then for all vectors $l \in \mathbb{R}^p$, we have

$$\begin{cases} \hat{\kappa}_{1l^*} \leq \hat{\kappa}_{1l} \\ -\hat{\kappa}_{1l^*} \geq \hat{\kappa}_{1(-l)}. \end{cases}$$

Therefore, $\hat{\kappa}_{1l^*} \leq \hat{\kappa}_{1l} \leq -\hat{\kappa}_{1l^*}$, and hence, $\hat{\kappa}_{1l}^2 \leq \hat{\kappa}_{1l^*}^2$ and $G_1(l) = -\hat{\kappa}_{1l}^2 \geq -\hat{\kappa}_{1l^*}^2$.

Therefore, a minimizer of function $\tilde{G}_1(l) = \hat{\kappa}_{1l} = \dfrac{1}{n-1}\sum_{i=1}^{n}\left(l'\tilde{\mathbf{X}}_i\right)^3$ is also a minimizer of the function $G_1(l)$. Therefore, it suffices to solve the following problem:

$$\begin{cases} \min\limits_{l\in\mathbb{R}^p} \tilde{G}_1(l) = \min\limits_{l\in\mathbb{R}^p} \sum\limits_{i=1}^{n}\left(l'\tilde{\mathbf{X}}_i\right)^3 \\ l'l = 1 \end{cases} \tag{3.3}$$

with the constant coefficient $1/(n-1)$ is simplified. Solution l^* of the problem (3.3) is indeed a solution of the problem (3.1) in the case $k = 1$. We derive a more simplified gradient function, namely, $\nabla\tilde{G}_1(l) = 3\sum_{i=1}^{n}\left(l'\tilde{\mathbf{X}}_i\right)^2\tilde{\mathbf{X}}_i$.

In our following computations and simulations, we will obtain solutions for problem (3.1) in the case $k = 1$ by solving the simplified problem (3.3).

The quantities $\hat{\varkappa}_k^2 = \max\limits_{l'Sl=1}\hat{\kappa}_{kl}^2$, $k = 1, 2$ are our estimated square of skewness ($k = 1$) and of kurtosis ($k = 2$) of a multivariate distribution. These can be used to assess the assumption of multivariate normality. A large value of $\hat{\varkappa}_k^2$ provides evidence of non-normality. We thus have a test statistic by using either of the $\hat{\varkappa}_k^2$ (minimizing $G_k(l)$ separately), $k = 1, 2$ or by alternatively minimizing their "average"
$\bar{\varkappa}^2 = \min\limits_{l'l=1}\dfrac{1}{2}\left[G_1(l) + G_2(l)\right]$.

In Tables 3.4, 3.5 and 3.6 we give the 95% and 99% cut off points for $\dfrac{\hat{\varkappa}_1^2}{p}$, $\dfrac{\hat{\varkappa}_2^2}{p}$ and $\dfrac{\bar{\varkappa}^2}{p}$ for sample sizes $n = 24, 30, 50, 80, 100, , 200, 300, 500, 800$. These are obtained via $m = 4000$ simulations of the samples from $N_p(\mathbf{0}_p, \mathbf{I}_p)$. For each simulation, values in line 1 of the Algorithm 3.1 are chosen as $\epsilon = 10^{-6}, t = 0.001, m = 5000$ and $l_0 = \mathbf{1}_p$.

Table 3.4 Critical value for a size α test using the best linear transformation - minimizing skewness $\hat{\kappa}_1^2/p$ (based on 4000 simulations).

Dimension (p)	α	**Sample size** (n)								
		24	**30**	**50**	**80**	**100**	**200**	**300**	**500**	**800**
2	.05	0.5079	0.4530	0.3075	0.2065	0.1655	0.0875	0.0590	0.0354	0.0216
	.01	0.8792	0.7834	0.5580	0.3462	0.2887	0.1400	0.0951	0.0557	0.0345
3	.05	0.5240	0.4987	0.3165	0.2087	0.1684	0.0849	0.0578	0.0342	0.0220
	.01	0.8577	0.8137	0.5070	0.3367	0.2598	0.1273	0.0811	0.0503	0.0312
4	.05	0.5477	0.4999	0.3374	0.2051	0.1673	0.0884	0.0587	0.0335	0.0211
	.01	0.8742	0.7474	0.5050	0.3143	0.2633	0.1249	0.0818	0.0459	0.0284
5	.05	0.5978	0.5253	0.3429	0.2215	0.1751	0.0875	0.0584	0.0347	0.0216
	.01	0.8641	0.7597	0.5122	0.3060	0.2429	0.1167	0.0751	0.0474	0.0272
6	.05	0.6236	0.5523	0.3511	0.2220	0.1780	0.0876	0.0575	0.0345	0.0208
	.01	0.9329	0.7779	0.5308	0.3244	0.2515	0.1171	0.0784	0.0436	0.0268
7	.05	0.6303	0.5878	0.3702	0.2345	0.1842	0.0881	0.0595	0.0345	0.0215
	.01	0.8798	0.8083	0.5101	0.3289	0.2521	0.1203	0.0762	0.0437	0.0275
8	.05	0.6460	0.5757	0.3842	0.2445	0.1896	0.0918	0.0587	0.0336	0.0213
	.01	0.9246	0.7835	0.5344	0.3242	0.2564	0.1148	0.0728	0.0412	0.0259
9	.05	0.6863	0.6144	0.4022	0.2458	0.2005	0.0931	0.0587	0.0347	0.0217
	.01	0.9444	0.8448	0.5695	0.3485	0.2751	0.1155	0.0737	0.0438	0.0265
10	.05	0.7181	0.6575	0.4080	0.2489	0.2023	0.0941	0.0613	0.0349	0.0213
	.01	0.9220	0.9440	0.5662	0.3466	0.2699	0.1214	0.0758	0.0423	0.0258

Table 3.5 Critical value for a size α test using the best linear transformation - minimizing kurtosis $\hat{\kappa}_2^2/p$ (based on 4000 simulations).

Dimension (p)	α	**Sample size** (n)								
		24	**30**	**50**	**80**	**100**	**200**	**300**	**500**	**800**
2	.05	1.2015	1.1144	0.8749	0.7259	0.5682	0.3257	0.2228	0.1452	0.0854
	.01	3.6554	3.1900	2.7926	2.0215	1.7460	0.8054	0.5491	0.2892	0.1492
3	.05	1.3512	1.2647	1.1065	0.8719	0.7164	0.3672	0.2473	0.1434	0.0840
	.01	4.0976	4.5859	3.5198	2.3850	1.7306	0.7711	0.5248	0.2817	0.1653
4	.05	1.6531	1.6264	1.3737	1.0604	0.8848	0.4608	0.2791	0.1629	0.0937
	.01	4.6011	4.8489	3.7990	2.4931	1.9346	0.9065	0.5329	0.2915	0.1768
5	.05	2.0489	2.0813	1.6113	1.3104	0.9937	0.5115	0.3136	0.1749	0.0996
	.01	5.0837	5.4626	3.8055	3.0005	2.3710	1.1445	0.6463	0.3268	0.1716
6	.05	2.6022	2.3686	2.1981	1.5711	1.2048	0.6054	0.3726	0.1971	0.1157
	.01	5.4671	5.8095	6.1946	3.3083	2.7720	1.0966	0.6433	0.3562	0.1856
7	.05	3.0435	2.8076	2.3458	1.7592	1.3763	0.6248	0.4116	0.2238	0.1139
	.01	6.8417	6.9365	5.6492	3.6004	3.0242	1.2534	0.7520	0.3728	0.1927
8	.05	3.5503	3.6888	2.7896	1.9438	1.6179	0.8036	0.4844	0.2384	0.1386
	.01	7.7719	8.2728	5.3308	4.0825	3.3121	1.4377	0.8600	0.3901	0.2344
9	.05	4.5702	4.5661	3.1364	2.3601	1.7608	0.8558	0.5390	0.2640	0.1424
	.01	7.5139	8.6877	6.6582	4.7287	3.2190	1.6798	0.8781	0.4498	0.2378
10	.05	4.5777	4.3835	3.5982	2.5135	2.1758	0.9942	0.5781	0.2924	0.1592
	.01	9.0387	8.4185	6.6059	4.9912	4.1381	1.7120	0.9677	0.4623	0.2337

Table 3.6 Critical value for a size α test using the best linear transformation - minimizing average of skewness and kurtosis $\tilde{x}^2/p$ (based on 4000 simulations).

Dimension (p)	α	**Sample size (n)**								
		24	**30**	**50**	**80**	**100**	**200**	**300**	**500**	**800**
2	.05	0.6801	0.7032	0.5494	0.4270	0.3357	0.1767	0.1296	0.0737	0.0459
	.01	2.6143	2.5765	1.4014	1.0678	0.9039	0.4380	0.2943	0.1491	0.0854
3	.05	0.7386	0.8789	0.6407	0.5523	0.3947	0.2019	0.1324	0.0764	0.0446
	.01	2.0642	2.1825	1.6764	1.4508	0.8892	0.4934	0.2903	0.1461	0.0795
4	.05	0.9046	1.1248	0.8438	0.6058	0.4414	0.2435	0.1439	0.0809	0.0464
	.01	2.5911	2.7216	1.9822	1.5777	0.9878	0.4988	0.2886	0.1492	0.0801
5	.05	1.3897	1.2454	1.0162	0.7678	0.5736	0.2864	0.1791	0.0932	0.0508
	.01	2.9860	3.0159	2.2169	1.5205	1.2229	0.5721	0.3510	0.1705	0.0948
6	.05	1.5658	1.4776	1.1832	0.7969	0.6569	0.3121	0.1951	0.1004	0.0520
	.01	3.2468	3.1819	2.5855	1.8497	1.4796	0.5504	0.3508	0.1793	0.0911
7	.05	1.7400	1.7621	1.3842	0.9371	0.7533	0.3767	0.2185	0.1163	0.0604
	.01	3.8129	3.7834	2.9025	1.8742	1.7378	0.6980	0.3611	0.1999	0.0973
8	.05	2.0691	2.1142	1.7236	1.0930	0.9291	0.3973	0.2330	0.1226	0.0645
	.01	4.1284	4.4907	3.5365	2.3301	1.9675	0.7181	0.4186	0.2083	0.1010
9	.05	2.5134	2.3199	1.8090	1.3107	0.9903	0.4491	0.2791	0.1409	0.0699
	.01	4.8172	4.7693	3.7097	2.7011	2.0807	0.7889	0.5283	0.2468	0.1114
10	.05	2.7237	2.7367	2.0337	1.4133	1.1319	0.4842	0.2998	0.1483	0.0786
	.01	4.7827	4.9765	3.8714	2.4844	2.2383	0.8688	0.5471	0.2464	0.1320

Table 3.7 Test statistics of best linear transformation method, simulated data.

Distributions		$\hat{\varkappa}_1^2/p$	$\hat{\varkappa}_2^2/p$	$\bar{\varkappa}^2/p$
Normal	Standard (i)	0.0840	0.3176	0.5409
	Non-standard (ii)	0.0086	0.0550	0.0276
Asymmetric	Mixture Normal (iii)	0.0058	0.3665	0.1871
	Gaussian Copulas (iv)	0.29	0.73	0.47
	Lomax (v)	19.0618	2789.2117	1404.0693
Symmetric	$t(5)$ (vi)	0.1382	22.9096	2.6978
	$t(15)$ (vii)	0.1582	2.8976	1.5152
	Kotz (viii)	0.1157	3.8505	1.9503
5% critical values, $p = 5$, $n = 200$		.0875	.5115	.2864

3.7 Illustrative Examples for Best Linear Transformation Method: Simulated Data

To demonstrate our method of best linear transformation, we will again consider the same simulated data used in Section 3.3. With a sample size of $n = 200$, $p = 5$, and a pre-chosen Type I error probability $\alpha = .05$, the critical values for tests using skewness $\hat{\varkappa}_1/p$ and kurtosis $\hat{\varkappa}_2/p$ are $.0875$ and $.5115$ respectively (see Tables 3.4 and 3.5). Table 3.6 shows a critical value of the test using the average of skewness and kurtosis $\bar{\varkappa}^2/p$ as $.2864$. Table 3.7 gives values of the three test statistics calculated from the simulated data. Test using the average of skewness and kurtosis on the standard normal sample (i) wrongly rejects the null hypothesis, as the test statistics is $\bar{\varkappa}^2/p = .5409 > .2864$. In the case of asymmetric distributions, except the case of mixture normal (iii), all three tests are able to detect non-normality. All methods give significant results when samples are from symmetric distributions (scenarios (vi), (vii), and (viii)). The individual tests based on skewness and kurtosis appear to be more reliable than the tests based on their average.

3.8 Two Examples for Best Linear Transformation Method: Cork Data and Water Potability Data Revisited

We will apply the above tests based on the best linear transformation method to the Cork and Water Potability (= Good) datasets. Since simulated cut off values are only for

Table 3.8 Test statistics of best linear transformation method, Cork and Water Potability (= Good) datasets (variables considered are *Organic carbon, Turbidity, Trihalomethanes.*).

		Test statistics		
		$\hat{\kappa}_1^2/p$	$\hat{\kappa}_2^2/p$	$\bar{\kappa}^2/p$
Data set	**Cork**	.2553	.5266	.2653
	Water Potability (Good)	.0022	.0674	.0337

certain specific sample size n, we will estimate the corresponding critical values for our problem by using the nearest available yet smaller sample size.

Cork data has four variables, each of which marginally follows a normal distribution. Critical values with significant level $\alpha = .05$, $p = 4$ and sample size $n = 28$[4] are $.5477, 1.6531$ and $.9046$. As shown in Table 3.8, all three test statistic values computed from the Cork data set are all smaller than the corresponding critical values. Hence, we have enough evidence to believe that the Cork data set follows a four-variable normal distribution.

Water Potability (= Good) data set has *Organic carbon, Turbidity, Trihalomethanes*, which can be assumed to be marginally normally distributed. Here $p = 3$ and $n = 811$[5]. The critical values for size $\alpha = .05$ test are $.0220$, $.0840$, and $.0446$ respectively. The values of the test statistics are presented in Table 3.8. All test statistics are comparatively smaller than the corresponding critical values. These support the null hypothesis that variables *Organic carbon, Turbidity, Trihalomethanes* for the good water population can be assumed to have a jointly trivariate normal.

3.9 Concluding Remarks

We have introduced two basic approaches to evaluate the normality or non-normality of a multivariate dataset by transforming that to a univariate dataset.

[4]We will use value of $n = 24$ when using Tables 3.4, 3.5 and 3.6.

[5]We will use values of $n = 800$ in the corresponding tables.

The "$n \times p$-sample" method requires a large sample that upon the standardization using sample statistics instead of parameters, the correlations between components are weak. Hence, for the small and moderate-sized samples, the method may not perform satisfactorily (we have seen in the case of the Cork dataset). Similarly, using the score plot to find multivariate outliers requires a pre-chosen smoothing parameter λ and some condition on the minimum distance of any two univariate observations. The choices of these parameters are determined by taking an independent $n \times p$ sample from $N(0, 1)$.

We have also investigated all possible linear combinations, each of which, under the hypothesis that data follow a multivariate normal distribution, results in univariate normality of the corresponding data. We examined the normality via the least normal projection direction l. We have also discussed the evaluation of non-normality via either skewness, kurtosis, or both.

The above methods are simple and easy to apply, but their premise relies on the transformation of a multivariate dataset into a univariate dataset. Thus, the approach is somewhat indirect and, therefore, may not be effective for a small sample size. In the next chapter, we will introduce two more innovative methods that directly depict the evidence of the non-normality of multivariate data without requiring any such transformations.

CHAPTER FOUR

EVALUATION OF MULTIVARIATE NORMALITY VIA CUMULANT GENERATING FUNCTION

4.1 Introduction

The moment generating function (mgf) is a function that, if it exists, fully characterizes the distribution of a random variable. For a multivariate normal distribution, utilizing the exponential form of mgf, Csörgő [62] derived the tests for normality based on a maximal deviation statistic, which relies on the empirical mgf. Moreover, Fang et al. [63] proposed a graphical method to detect multivariate non-normality by finding a projection direction so that data is appropriately transformed to a univariate normally distributed sample under normality assumption, and then apply Ghosh's T_3 [39] plot.

The cumulant generating function of a multivariate normality distribution is quadratic in vector $\mathbf{t} \in \mathbb{R}^p$. Borrowing Ghosh's [39] idea, we will discuss here the properties of the third and fourth derivatives of the cumulant generating function to obtain a test for multivariate normality assumption. However, the problem here is more complicated since now the third derivatives constitutes a $p \times p \times p$ array, and that for the fourth derivatives will be a $p \times p \times p \times p$ array. Consequently, there are going to be mixed derivatives in the picture. We must now evaluate the multivariate normality or non-normality of a dataset, not just by their marginal distribution as done in Ghosh [39] for the univariate case but also by taking the joint distribution into account. Further, we will do so not just through the third derivatives of the cumulant generating function but also through the fourth ones.

Sections 4.2 and 4.3 discuss the multivariate extensions of univariate T_3 and T_4 plots (see Chapter 2), where we explore the use of the empirical cumulant generating function of a multivariate sample. In each section, we will also illustrate the effectiveness

of these two graphs via various examples, where data are from symmetric and asymmetric distributions. We conclude the chapter with some conclusions and remarks.

4.2 Graphical Method Based on MT_3 Plot

In this section, we generalize the idea of Ghosh's T_3 plot for testing the multivariate normality. As a generalization of Lemma 2.4 to the case of multivariate random vectors, we have the following.

Lemma 4.1 ([64])**.** Let $\mathrm{M}_{\mathbf{X}}(\mathbf{t})$ be the moment generating function of a $p \times 1$ random vector $\mathbf{X}$. Then

$$\frac{\partial^3}{\partial \mathbf{t}^3}\left[\log(M_{\mathbf{X}}(\mathbf{t}))\right] = \mathbf{0} \text{ iff } \mathbf{X} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where $\dfrac{\partial^k}{\partial \mathbf{t}^k}(.)$ is an array containing all mixed derivatives $\dfrac{\partial^k}{\partial t_1^{k_1} \partial t_2^{k_2} \ldots \partial t_p^{k_p}}$, and $\sum_{j=1}^{p} k_j = k$.

Thus, for the cumulant generating function $\mathfrak{C}(\mathbf{t}) = \log\left(M_{\mathbf{X}}(\mathbf{t})\right)$ of $\mathbf{X}$,

$$\frac{\partial^k}{\partial \mathbf{t}^k}\mathfrak{C}(\mathbf{t}) = \mathbf{0}, k = 3, 4, \ldots$$

if and only if $\mathbf{X}$ has a multivariate normal distribution.

Note that for the third derivatives ($k = 3$), $\dfrac{\partial^k}{\partial \mathbf{t}^k}\mathfrak{C}(\mathbf{t})$ represents a $p \times p \times p$ array, and for $k = 4$, it would be a $p \times p \times p \times p$ array. Clearly, the complexity increases considerably for other higher-order derivatives.

We will rely on the above multivariate characterization and follow the idea of Ghosh. However, one expects the multivariate problem to be much more complex as the functions under consideration are the multivariable functions of $\mathbf{t} = \left[t_1, t_2, \ldots, t_p\right]'$.

The following lemma will be needed for the subsequent calculations.

Lemma 4.2. For any vector function $u(\mathbf{t}) = (u_1(\mathbf{t}), \ldots u_p(\mathbf{t}))$, and for $j = 1, 2, \ldots, p$,

$$\begin{aligned}
&\frac{\partial}{\partial t_j} u(\mathbf{t}) u^T(\mathbf{t}) \\
&= \begin{bmatrix} u_1(\mathbf{t}) \\ u_2(\mathbf{t}) \\ \vdots \\ u_p(\mathbf{t}) \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial t_j} u_1(\mathbf{t}) & \frac{\partial}{\partial t_j} u_2(\mathbf{t}) & \ldots & \frac{\partial}{\partial t_j} u_p(\mathbf{t}) \end{bmatrix} + \begin{bmatrix} \frac{\partial}{\partial t_j} u_1(\mathbf{t}) \\ \frac{\partial}{\partial t_j} u_2(\mathbf{t}) \\ \vdots \\ \frac{\partial}{\partial t_j} u_p(\mathbf{t}) \end{bmatrix} \begin{bmatrix} u_1(\mathbf{t}) & u_2(\mathbf{t}) & \ldots & u_p(\mathbf{t}) \end{bmatrix}.
\end{aligned}$$

Given a multivariate random sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$, the sample version of moment generating function is

$$\hat{\mathrm{M}}_{\mathbf{X}}(\mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} \exp(\mathbf{t}'\mathbf{X}_i). \tag{4.1}$$

Similar to the case of the univariate sample, we define the empirical studentized version for the moment generating function as follows:

$$\hat{\mu}(\mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} \exp\left(\mathbf{t}'\mathbf{S}^{-1/2}(\mathbf{X}_i - \bar{\mathbf{X}})\right) = \frac{1}{n} \sum_{i=1}^{n} \exp\left(\mathbf{t}'\tilde{\mathbf{X}}_i\right) \tag{4.2}$$

where $\bar{\mathbf{X}}$ and $\mathbf{S}$ are the usual sample mean and sample covariance matrix and $\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots, \tilde{\mathbf{X}}_n$ are the standardized data with $\tilde{\mathbf{X}}_i = \mathbf{S}^{-1/2}(\mathbf{X}_i - \bar{\mathbf{X}}), i = 1, 2 \ldots n$.

Accordingly, we have

$$\begin{aligned}
\hat{\mu}^{(1)}_{(p\times 1)}(\mathbf{t}) &= \frac{\partial}{\partial \mathbf{t}} \hat{\mu}(\mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} \tilde{\mathbf{X}}_i \exp(\mathbf{t}'\tilde{\mathbf{X}}_i), \\
\hat{\mu}^{(2)}_{2(p\times p)}(\mathbf{t}) &= \frac{\partial^2}{\partial \mathbf{t}^2} \hat{\mu}(\mathbf{t}) = \frac{1}{n} \sum_{i=1}^{n} \tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_i' \exp \mathbf{t}'\tilde{\mathbf{X}}_i), \\
\hat{\mu}^{(3)j}(\mathbf{t}) &= \frac{\partial}{\partial t_j} \left(\frac{\partial^2}{\partial \mathbf{t}^2} \hat{\mu}(\mathbf{t}) \right) = \frac{1}{n} \sum_{i=1}^{n} \tilde{\mathbf{X}}_i \tilde{\mathbf{X}}_i' \tilde{X}_{ij} \exp(\mathbf{t}'\tilde{\mathbf{X}}_i),
\end{aligned}$$

where $\hat{\mu}^{(3)j}(\mathbf{t})$ is a $p \times p$ array and $\tilde{X}_{ij}$ is the j^{th} element of the standardized observation $\tilde{\mathbf{X}}_i$, for $j = 1, 2, \ldots, p$.

A natural estimator of the cumulant generating function $\mathfrak{C}(\mathbf{t})$ is $\hat{\mathfrak{C}}(\mathbf{t}) = \log(\hat{\mathrm{M}}_{\mathbf{X}}(\mathbf{t}))$, from which we can define the empirical cumulant generating function on standardized data as $\hat{K}(\mathbf{t}) = \log[\hat{\mu}(\mathbf{t})]$. It is straightforward to derive the first and second derivatives of $\hat{K}(\mathbf{t})$,

$$\begin{aligned}
\hat{K}^{(1)}(\mathbf{t}) &= \frac{\partial}{\partial \mathbf{t}} \log \hat{\mu}(\mathbf{t}) = \frac{1}{\hat{\mu}(\mathbf{t})}\hat{\mu}^{(1)}(\mathbf{t}), \\
\hat{K}^{(2)}(\mathbf{t}) &= \frac{\partial^2}{\partial \mathbf{t}^2} \log \hat{\mu}(\mathbf{t}) = \frac{1}{\hat{\mu}(\mathbf{t})}\hat{\mu}^{(2)}(\mathbf{t}) - \frac{1}{\hat{\mu}^2(\mathbf{t})}\left[\hat{\mu}^{(1)}(\mathbf{t})\left(\hat{\mu}^{(1)}(\mathbf{t})\right)'\right], \\
&= \frac{\hat{\mu}^{(2)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})} - \left[\frac{\hat{\mu}^{(1)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\right]\left[\frac{\hat{\mu}^{(1)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\right]^T.
\end{aligned}$$

Also, for $j = 1, \ldots p$, we note that

$$\frac{\partial}{\partial t_j}\frac{\hat{\mu}^{(2)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})} = \frac{\hat{\mu}^{(3)j}(\mathbf{t})}{\hat{\mu}(\mathbf{t})} - \frac{\hat{\mu}^{(2)}(\mathbf{t})\hat{\mu}^{(1)j}(\mathbf{t})}{\hat{\mu}^2(\mathbf{t})},$$

where $\hat{\mu}^{(1)j}(\mathbf{t})$ is the j^{th}element of vector $\hat{\mu}^{(1)}(\mathbf{t})$. By Lemma 4.2,

$$\frac{\partial}{\partial t_j}\hat{\mu}^{(1)}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]' = \hat{\mu}^{(1)}(\mathbf{t})[\hat{\mu}^{(2)j}(\mathbf{t})]' + \hat{\mu}^{(2)j}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'$$

where $\hat{\mu}^{(2)j}(\mathbf{t})$ is the j^{th} column of the $p \times p$ matrix $\hat{\mu}^{(2)}(\mathbf{t})$. Therefore,

$$\begin{aligned}
\frac{\partial}{\partial t_j}\left(\frac{\hat{\mu}^{(1)}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'}{\hat{\mu}^2(\mathbf{t})}\right) &= \frac{1}{\hat{\mu}^2(\mathbf{t})}\frac{\partial}{\partial t_j}\left(\hat{\mu}^{(1)}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'\right) - \frac{1}{\hat{\mu}^4(\mathbf{t})}\hat{\mu}^{(1)}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'\frac{\partial}{\partial t_j}\hat{\mu}^2(\mathbf{t}) \\
&= \frac{1}{\hat{\mu}^2(\mathbf{t})}\left(\hat{\mu}^{(1)}(\mathbf{t})[\hat{\mu}^{(2)j}(\mathbf{t})]' + \hat{\mu}^{(2)j}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'\right) \\
&\quad - 2\frac{\hat{\mu}^{(1)j}(\mathbf{t})}{\hat{\mu}^3(\mathbf{t})}\hat{\mu}^{(1)}(\mathbf{t})[\hat{\mu}^{(1)}(\mathbf{t})]'.
\end{aligned}$$

We then obtain the third derivatives of $\hat{K}(\mathbf{t}) = \log(\hat{\mu}(\mathbf{t}))$ starting with $t_j, j = 1 \dots p$ as follow:

$$\begin{aligned}\hat{K}^{(3)j}(\mathbf{t}) =& \frac{\partial}{\partial t_j}\frac{\partial^2}{\partial \mathbf{t}^2}\log\hat{\mu}(\mathbf{t}) \\ =& \frac{\hat{\mu}^{(3)j}(\mathbf{t})}{\hat{\mu}(\mathbf{t})} - \frac{\hat{\mu}^{(2)}(\mathbf{t})\hat{\mu}^{(1)j}(\mathbf{t})}{\hat{\mu}^2(\mathbf{t})} + 2\frac{\hat{\mu}^{(1)j}(\mathbf{t})}{\hat{\mu}^3(\mathbf{t})}\hat{\mu}^{(1)}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]' \\ & - \frac{1}{\hat{\mu}^2(\mathbf{t})}\left[\hat{\mu}^{(1)}(\mathbf{t})[\hat{\mu}^{(2)j}(\mathbf{t})]^T + \hat{\mu}^{(2)j}(\mathbf{t})\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'\right]\end{aligned}$$

or

$$\begin{aligned}\hat{K}^{(3)j}(\mathbf{t}) =& \frac{\hat{\mu}^{(3)j}(\mathbf{t})}{\hat{\mu}(\mathbf{t})} - \frac{\hat{\mu}^{(2)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\frac{\hat{\mu}^{(1)j}(\mathbf{t})}{\hat{\mu}(\mathbf{t})} + 2\frac{\hat{\mu}^{(1)j(\mathbf{t})}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\frac{\hat{\mu}^{(1)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\frac{\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'}{\hat{\mu}(\mathbf{t})} \\ & - \left[\frac{\hat{\mu}^{(1)}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\frac{[\hat{\mu}^{(2)j}(\mathbf{t})]^T}{\hat{\mu}(\mathbf{t})} + \frac{\hat{\mu}^{(2)j}(\mathbf{t})}{\hat{\mu}(\mathbf{t})}\frac{\left[\hat{\mu}^{(1)}(\mathbf{t})\right]'}{\hat{\mu}(\mathbf{t})}\right].\end{aligned} \tag{4.3}$$

Let $\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t})$ be the element indexed by the triplet (j_1, j_2, j_3) in the cubical array $\hat{K}^{(3)}(\mathbf{t})$. Due to the symmetry properties of mixed derivatives, it is obvious that with any permutation of the triplet (j_1, j_2, j_3), the resulting derivatives are the same as $\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t})$ with $j_1 \le j_2 \le j_3$. Therefore, to eliminate redundant cases, in the upcoming discussions, we will consider mixed derivatives $\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t})$ with the restriction that $1 \le j_1 \le j_2 \le j_3 \le p$. That way, we need to work only with distinct values of mixed derivatives.

Remark. The number of unique third derivative terms in the three-dimensional array $\hat{K}^{(3)}(\mathbf{t})$ is:

$$l_{K_3} = p + 2 \times \binom{p}{2} + \binom{p}{3}.$$

In view of Lemma 4.1, third derivatives of the cumulant generating function, $\frac{\partial^3}{\partial^3 \mathbf{t}}\mathfrak{C}(\mathbf{t})$, of any normally distributed vector is an array consisting of all 0. We can thus expect that their empirical estimators $\frac{\partial^3}{\partial \mathbf{t}^3}\hat{\mathfrak{C}}^{(3)}(\mathbf{t})$ will also have values closed 0. Moreover, $\boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_1 - \boldsymbol{\mu}), \boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_2 - \boldsymbol{\mu}), \dots, \boldsymbol{\Sigma}^{-1/2}(\mathbf{X}_n - \boldsymbol{\mu})$ constitute a random sample

from the multivariate standard normal distribution $N_p(\mathbf{0}_p, \mathbf{I}_p)$. When $\mathbf{S}$ and $\bar{\mathbf{X}}$ are used in the placed of $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the standardized sample $\tilde{\mathbf{X}}_1, \tilde{\mathbf{X}}_2, \ldots, \tilde{\mathbf{X}}_n$ will be approximately $N_p(\mathbf{0}_p, \mathbf{I}_p)$ distributed. Accordingly, $\hat{K}^{(3)}(.)$ being an estimator of the function $\dfrac{\partial^3}{\partial^3 \mathbf{t}} \log[\mu(.)]$, is likely to be small as well. In the later section, we will also show that under the normality assumption, $\hat{K}^{(3)}(.)$ asymptotically follows a normal distribution with mean $\dfrac{\partial^3}{\partial^3 \mathbf{t}} \log[\mu(.)] = \mathbf{0}$.

With the above premise, a large magnitude of $\hat{K}^{(3)}(\mathbf{t})$ will be concrete evidence of non-normality. However, the high dimensional structure of $\hat{K}^{(3)}(\mathbf{t})$ limits the ability to obtain a graphical method as a tool to reflect this fact. One possible approach is averaging all mixed third derivatives, which is expected to be around 0 if the data are from a multivariate normal distribution.

One may question, why not use the statistic $\max\left|\hat{K}^{(3)}(\mathbf{t})\right|$ and plot it as a function of $\mathbf{t}$, where the maximum is taken over all third derivatives. Such a function then has to be evaluated for all $\mathbf{t}$ and may not result in a very smooth function $\mathbf{t}$. Further, it would be difficult to mathematically obtain any suitable bands for such a curve. In contrast, as in the above case, a (any) linear combination of derivatives is much more mathematically tractable.

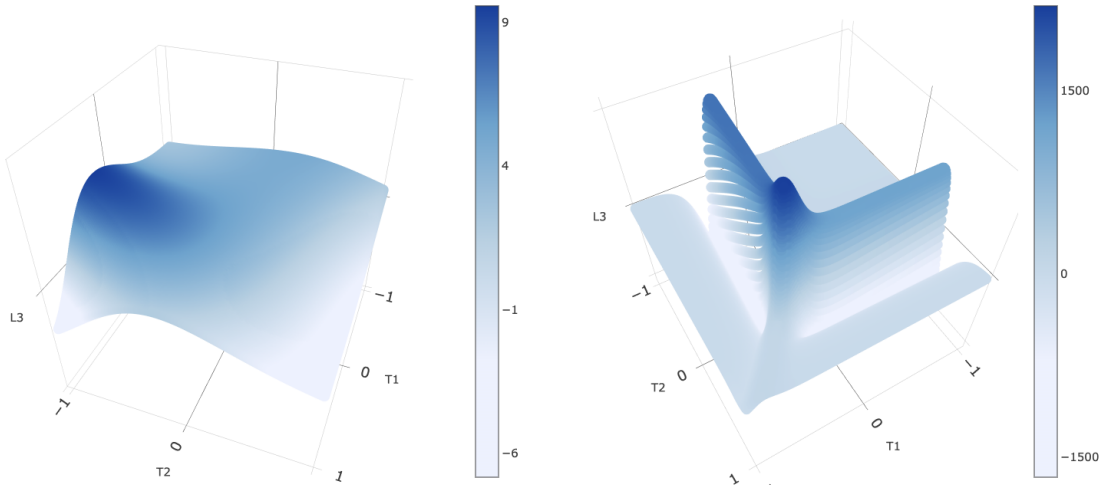
Definition 4.1 (A test statistic for MT_3 plot)**.** Let $\tilde{\mathcal{T}}_3^n(\mathbf{t})$ be a vector of length l_{K_3}, consisting of $\sqrt{n}\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t}), 1 \le j_1 \le j_2 \le j_3 \le p$, i.e

$$\tilde{\mathcal{T}}_3^n(\mathbf{t}) = \sqrt{n}\left[\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t})\right] = \sqrt{n}\left[\hat{K}^{(111)}(\mathbf{t}), \hat{K}^{(112)}(\mathbf{t}), \ldots, \hat{K}^{(ppp)}(\mathbf{t})\right]'.$$

Also let $\mathbf{a}_3 = \dfrac{1}{\sqrt{l_{K_3}}}\mathbf{1}_{l_{K_3}}$. Then, a scaled average of distinct mixed derivatives inside the cube $\hat{K}^{(3)}(\mathbf{t})$ can be rewritten as $L_3(\mathbf{t}) = \mathbf{a}_3'\left[\tilde{\mathcal{T}}_3^n(\mathbf{t})\right]$.

$L_3(\mathbf{t})$ can be used as a test statistic. Under the assumption of normality, $L_3(\mathbf{t})$ should also have a small value[1]. We call a plot of $L_3(\mathbf{t})$ as MT_3 plot.

[1]We will discuss the appropriate magnitude of $L_3(\mathbf{t})$ in a later section.

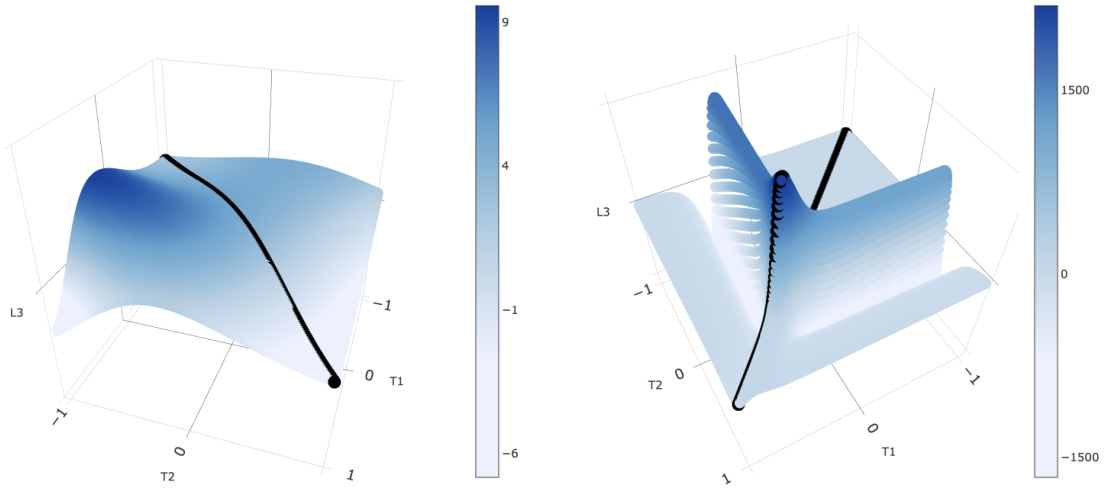


(a) $N_2(\mathbf{0}_5, \mathbf{I}_5)$. (b) $F(a = 3, l = 3 \times \mathbf{1}_5)$.

Figure 4.1: Plot of $L_3(\mathbf{t})$ for bivariate distributions.

Figure 4.1 represents two cases of bivariate data when $L_3(\mathbf{t})$ is a function of $\mathbf{t} = [t_1, t_2]'$, and $\mathbf{t} \in [-1, 1]^2$. With sample size $n = 200$, the two figures reflect different behaviors of normal and non-normal data. More specifically, Figure 4.1a is the scenario when data follow $N_2(\mathbf{0}_2, \mathbf{I}_2)$, the corresponding L_3 function has relatively small values, comparable to 0, ranging from -9 to 9. Here, the surface $L_3(\mathbf{t})$ appears to be relatively flat. On the other hand, in Figure 4.1b for the bivariate $F(a = 3, l = 3 \times \mathbf{1}_2)$ [65], the statistic $L_3(\mathbf{t})$ appears to be excessive large in certain regions with the maximum of $L_3(\mathbf{t})$ around 2000. Also, $L_3(\mathbf{t})$ is a relatively non-smooth surface.

Although the difference between normal and non-normal cases is quite obvious in the above two graphs of $L_3(\mathbf{t})$ with $\mathbf{t} \in [-1, 1]^2$, it is not possible to obtain such a graph for data from higher dimensions. The choice of test points $\mathbf{t} = \frac{1}{\sqrt{p}} t^* \mathbf{1}_p$ implies $L_3(\mathbf{t}) = L_3(t^*)$, a smooth function on $\mathbb{R}$, and hence a 2D graph can be constructed easily. By Lemma 4.1, under normality, $L_3(t^*)$ behaves as a function with nearly zero gradient. As illustrated in Figure 4.2, instead of investigating the whole surface $L_3(\mathbf{t})$, we may

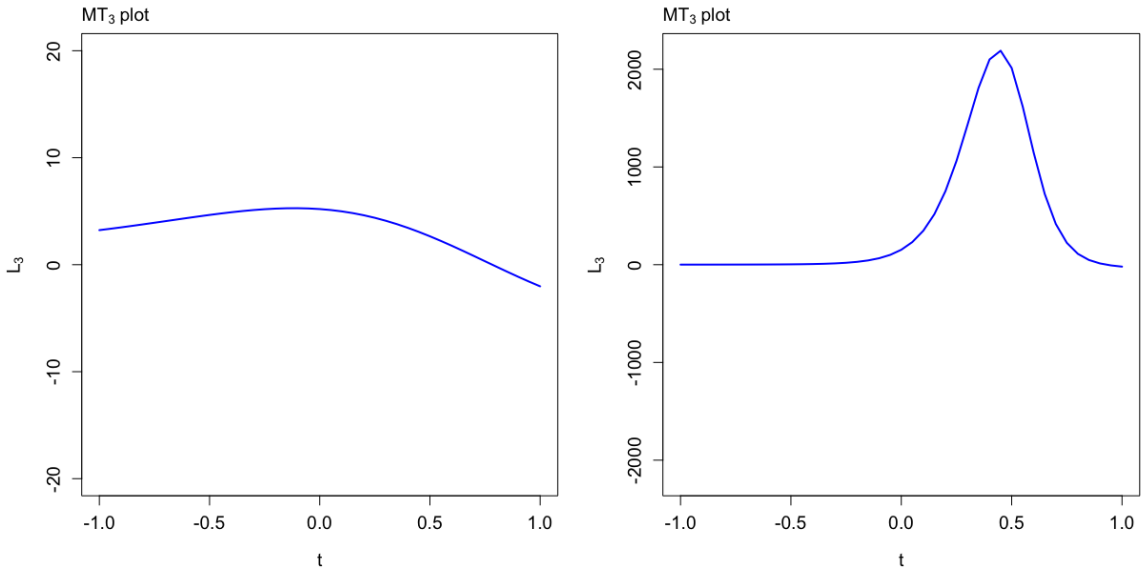


(a) $N_2(\mathbf{0}_5, \mathbf{I}_5)$. (b) $F(a = 3, l = 3 \times \mathbf{1}_5)$.

Figure 4.2: Plot of $L_3(\mathbf{t})$ for bivariate distributions with parameterization.

consider investigating the restricted curves, such as the ones represented by the black line. This results in 2D graphs such as ones shown in Figure 4.3. We name these 2D graphs as a MT_3 plot as a generalization for the T_3 plot introduced in Chapter 2. The $L_3(t^*)$ curve in Figure 4.3b shows strong evidence of non-normality when data is generated from multivariate F-distribution, as indicated by the large value function with non-zero slope.

For any dimension p, this choice of test vector $\mathbf{t} = \dfrac{1}{\sqrt{p}} t^* \mathbf{1}_p$ gives a 2D graph. Figure 4.4 below is an illustration of MT_3 plot in $p = 5$ dimension and sample size $n = 200$ for two different distributions. The two figures therein reflect different behaviors in the normal and non-normal data. Figure 4.4a is the scenario when data follow $N_5(\mathbf{0}_5, \mathbf{I}_5)$, (where the y-axis is limited to $[-20, 20]$), the corresponding L_3 function for different t^* values are relatively small. On the other hand, for the multivariate $F(a = 3, l = 3 \times \mathbf{1}_5)$ distribution [65], the statistic $L_3(t^*)$ appears to be excessive large (so large that the y-axis had to be extended to $[-1000, 1000]$). Comparing the slope as well as



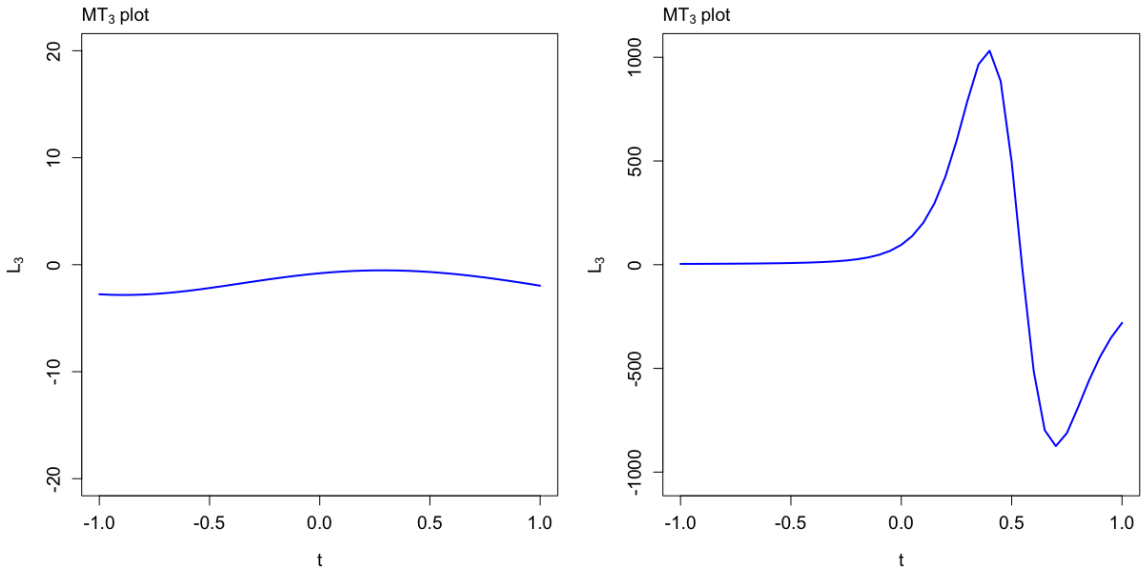
(a) $N_2(\mathbf{0}_2, \mathbf{I}_2)$. (b) $F(a = 3, l = 3 \times \mathbf{1}_5)$.

Figure 4.3: MT_3 plots for bivariate data.

the magnitude in Figure 4.4b with those in Figure 4.4a, the non-normality of data used for Figure 4.4b becomes self-evident. Again, although we are working with $p = 5$ dimension, the 2D Figure 4.4 still gives the same interpretation as in the case of $p = 2$ in Figure 4.3.

Remark (MT_3 plot and Mardia's skewness statistic). The Mardia's skewness [3] test statistic for testing the multivariate normality is given by,

$$b_{1,p} = \sum_{j_1,j_2,j_3=1}^{p} \sum_{j_1',j_2',j_3'=1}^{p} \left[\frac{1}{n} \sum_{i=1}^{n} \tilde{X}_{ij_1} \tilde{X}_{ij_2} \tilde{X}_{ij_3} \right] \left[\frac{1}{n} \sum_{i=1}^{n} \tilde{X}_{ij_1'} \tilde{X}_{ij_2'} \tilde{X}_{ij_3'} \right].$$



(a) $N_5(\mathbf{0}_5, \mathbf{I}_5)$. (b) $F(a = 3, l = 3 \times \mathbf{1}_5)$.

Figure 4.4: MT_3 plots for five-dimensional data.

We can rewrite a typical element $\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t})$ of cube $\hat{K}^{(3)}(\mathbf{t})$ as

$$\begin{aligned}\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t}) &= \frac{\partial^3}{\partial t_{j_1} \partial t_{j_2} \partial t_{j_3}} \log(\hat{\boldsymbol{\mu}}(\mathbf{t})) \\ &= \sum_{i=1}^{n} \tilde{X}_{ij_1} \tilde{X}_{ij_2} \tilde{X}_{ij_3} P_i + 2 \left(\sum_{i=1}^{n} \tilde{X}_{ij_1} P_i\right)\left(\sum_{i=1}^{n} \tilde{X}_{ij_2} P_i\right)\left(\sum_{i=1}^{n} \tilde{X}_{ij_3} P_i\right) \\ &\quad - \left(\sum_{i=1}^{n} \tilde{X}_{ij_1} \tilde{X}_{ij_2} P_i\right)\left(\sum_{i=1}^{n} \tilde{X}_{ij_3} P_i\right) - \left(\sum_{i=1}^{n} \tilde{X}_{ij_1} \tilde{X}_{ij_3} P_i\right)\left(\sum_{i=1}^{n} \tilde{X}_{ij_2} P_i\right) \\ &\quad - \left(\sum_{i=1}^{n} \tilde{X}_{ij_2} \tilde{X}_{ij_3} P_i\right)\left(\sum_{i=1}^{n} \tilde{X}_{ij_1} P_i\right)\end{aligned}$$

with the weights quantities $P_i(\mathbf{t}) = \dfrac{\exp(\mathbf{t}'\tilde{\mathbf{X}}_i)}{\sum_{k=1}^{n} \exp(\mathbf{t}'\tilde{\mathbf{X}}_k)} \geq 0, i = 1, 2, \ldots, n$. Clearly, $\sum_{i=1}^{n} P_i(\mathbf{t}) = 1$. It is easy to see that, as $\|\mathbf{t}\|_p \to 0$, $P_i \to 1/n, i = 1, 2, \ldots n$ and hence

$\hat{K}^{(j_1 j_2 j_3)}(\mathbf{t}) \to \frac{1}{n}\sum_{i=1}^{n} \tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_3}$, for any triplet (j_1, j_2, j_3). Thus,

$$\left[\sum_{j_1,j_2,j_3=1}^{p} \hat{K}^{(j_1 j_2 j_3)}(\mathbf{t})\right]^2 \to \sum_{j_1,j_2,j_3=1}^{p}\sum_{j_1',j_2',j_3'=1}^{p}\left[\frac{1}{n}\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_3}\right]\left[\frac{1}{n}\sum_{i=1}^{n}\tilde{X}_{ij_1'}\tilde{X}_{ij_2'}\tilde{X}_{ij_3'}\right] = b_{1,p}.$$

Specifically, at $\mathbf{t} = \mathbf{0}$,

$$\left|\sum_{j_1,j_2,j_3=1}^{p} \hat{K}^{(j_1 j_2 j_3)}(\mathbf{0}_p)\right| = \sqrt{b_{1,p}}.$$

Hence, our estimator for the third derivative of the cumulant generating function contains the information about multivariate skewness as $\|\mathbf{t}\|_p \to 0$. The use of $L_3(\mathbf{t})$ as the average of distinct mixed derivatives will possibly result in a value different from $\sqrt{b_{1,p}}$. However, $L_3(\mathbf{0}_p)$ gives us an idea about the magnitude of Mardia's skewness.

4.2.1 Asymptotic Distribution of $\hat{\mathfrak{C}}^{(3)}(\mathbf{t})$: Multivariate Standard Normal Case

In the previous section, we obtained $L_3(\mathbf{t})$ as a functional test statistic to evaluate the normality of a given sample. $L_3(\mathbf{t})$ in principle can be plotted as a function of $\mathbf{t}$. For a two dimensional plot, we had replaced $\mathbf{t}$ by smooth scalar function of $t_1, t_2, \ldots t_p$ (we chose $\mathbf{t} = \frac{1}{\sqrt{p}} t^* \mathbf{1}_p$). To further refine such a two-dimensional plot, we may want to also specify some probability bands that may serve as references for the large magnitude of $L_3(\mathbf{t})$. Here, we intend to develop such a region for a given probability $(1-\alpha)$.

We will first derive the needed results applicable for empirical cumulant generating function $\hat{\mathfrak{C}}(\mathbf{t}) = \log\left[\hat{M}(\mathbf{t})\right]$ under the assumption of multivariate standard normal distribution. Let $\mu_p(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right)$ be the moment generating function of $N(\mathbf{0}_p, \mathbf{I}_p)$. The following lemma is needed in the process.

Lemma 4.3. Let $\mathbf{X} = \left[X_1, X_2, \ldots, X_p\right]' \sim N_p(\mathbf{0}_p, \mathbf{I}_p)$, then

$$\mathbb{E}\left(X_1^{k_1} X_2^{k_2} \ldots X_p^{k_p} \exp(\mathbf{t}'\mathbf{X})\right) = \mu^{(k_1)}(t_1)\ldots\mu^{(k_p)}(t_p)$$

where $\mathbf{t} = \left[t_1, t_2, \ldots, t_p\right]' \in \mathbb{R}^p$ and $\mu^{(k)}(t)$ is the k^{th} order derivative of $\mu(t) = \exp\left(\frac{t^2}{2}\right)$.

Proof. Clearly $\mu(t) = \exp\left(\frac{t^2}{2}\right), t \in \mathbb{R}$ is the moment generating function of the univariate standard normal distribution $N(0,1)$. For a p-variable standard normal vector $\mathbf{X} = \left[X_1, X_2, \ldots, X_p\right]'$, the components $X_1, X_2, \ldots, X_p$ are mutually independent and are standard normal. Thus,

$$\begin{aligned}
\mathbb{E}\left[X_1^{k_1} \ldots X_p^{k_p} \exp(\mathbf{t}'\mathbf{X})\right] &= \mathbb{E}\left[X_1^{k_1} \exp\left(t_1 X_1\right) X_2^{k_2} \exp\left(t_2 X_2\right) \ldots X_p^{k_p} \exp\left(t_p X_p\right)\right] \\
&= \mathbb{E}\left[X_1^{k_1} \exp\left(t_1 X_1\right)\right] \mathbb{E}\left[X_2^{k_2} \exp\left(t_2 X_2\right)\right] \ldots \mathbb{E}\left[X_p^{k_p} \exp\left(t_p X_p\right)\right] \\
&= \mu^{(k_1)}(t_1)\mu^{(k_2)}(t_2) \ldots \mu^{(k_p)}(t_p).
\end{aligned}$$

□

Denote $\mu^{(j_1 j_2 j_3)}(\mathbf{t}) = \mathbb{E}X_{j_1}X_{j_2}X_{j_3} \exp(\mathbf{t}'\mathbf{X})$, $0 \le j_1 \le j_2 \le j_3 \le p$, with $X_0 = 1$. Certain expressions of $\mu^{(j_1 j_2 j_3)}(\mathbf{t})$ are given below.

$$\begin{aligned}
\mu^{(000)}(\mathbf{t}) &= \mu_p(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right), \\
\mu^{(00j_1)}(\mathbf{t}) &= \mu^{(1)}(t_{j_1}) \prod_{j \ne j_1} \mu(t_j) = t_{j_1} \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right), \\
\mu^{(0j_1j_1)}(\mathbf{t}) &= \mu^{(2)}(t_{j_1}) \prod_{j \ne j_1} \mu(t_j) = (1 + t_{j_1}^2) \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right), \\
\mu^{(j_1j_1j_1)}(\mathbf{t}) &= \mu^{(3)}(t_{j_1}) \prod_{j \ne j_1} \mu(t_j) = (3t_{j_1} + t_{j_1}^3) \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right), \\
\mu^{(0j_1j_2)}(\mathbf{t}) &= \mu^{(1)}(t_{j_1})\mu^{(1)}(t_{j_2}) \prod_{j \ne j_1, j_2} \mu(t_j) = t_{j_1}t_{j_2} \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right), \\
\mu^{(j_1j_1j_2)}(\mathbf{t}) &= \mu^{(2)}(t_{j_1})\mu^{(1)}(t_{j_2}) \prod_{j \ne j_1, j_2} \mu(t_j) = (1 + t_{j_1}^2)t_{j_2} \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right),
\end{aligned}$$

and

$$\mu^{(j_1j_2j_3)}(\mathbf{t}) = \mu^{(1)}(t_{j_1})\mu^{(1)}(t_{j_2})\mu^{(1)}(t_{j_3}) \prod_{j \ne j_1, j_2, j_3} \mu(t_j) = t_{j_1}t_{j_2}t_{j_3} \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right).$$

For sample of size n, $\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_n$ and for each $0 \leq j_1 \leq j_2 \leq j_3 \leq p$, let

$$\begin{aligned}\hat{M}^{(j_1 j_2 j_3)}(\mathbf{t}) &= \frac{\partial^k}{\partial t_{j_1} \partial t_{j_2} \partial t_{j_3}} \hat{M}(\mathbf{t}) \\ &= \frac{\partial^k}{\partial t_{j_1} \partial t_{j_2} \partial t_{j_3}} \left(\frac{1}{n} \sum_{i=1}^{n} \exp(\mathbf{t}' \mathbf{X}_i) \right) \\ &= \frac{1}{n} \sum_{i=1}^{n} X_{ij_1} X_{ij_2} X_{ij_3} \exp(\mathbf{t}' \mathbf{X}_i),\end{aligned}$$

where $\hat{M}(\mathbf{t})$ has been defined earlier in 4.1, $\mathbf{X}_{i0} = 1$ and k is the number of non-zero indices in the triplet (j_1, j_2, j_3). Then, under the assumption that the sample is from a standard multivariate normal distribution, it is obvious that $\hat{M}^{(j_1 j_2 j_3)}(\mathbf{t})$ are unbiased estimators of $\mu^{(j_1 j_2 j_3)}(\mathbf{t})$. Let

$$Z^{j_1 j_2 j_3}(\mathbf{t}) = \sqrt{n} \left(\hat{M}^{(j_1 j_2 j_3)}(\mathbf{t}) - \mu^{(j_1 j_2 j_3)}(\mathbf{t}) \right),$$

then it is also obvious that

$$\mathbb{E}\left[Z^{j_1 j_2 j_3}(\mathbf{t}) \right] = 0, \text{ for all set of index } (j_1, j_2, j_3).$$

Let $\mathcal{Z}_3^n(\mathbf{t})$ be a vector consisting of all distinct derivatives up to the third order, that is, of all $Z^{j_1 j_2 j_3}(\mathbf{t})$. More explicitly,

$$\mathcal{Z}_3^n(\mathbf{t}) = \begin{bmatrix} Z^{000}(\mathbf{t}) & Z^{001}(\mathbf{t}) & \ldots & Z^{00p}(\mathbf{t}) & Z^{011}(\mathbf{t}) & \ldots & Z^{ppp}(\mathbf{t}) \end{bmatrix}' \tag{4.4}$$

where element $Z^{j_1 j_2 j_3}(\mathbf{t})$ are all ordered within $\mathcal{Z}_3^n(\mathbf{t})$ by j_1, j_2 and j_3 and the respective positions of $Z^{j_1 j_2 j_3}(\mathbf{t})$ inside $\mathcal{Z}_3^n(\mathbf{t})$, say $J(j_1, j_2, j_3)$, are obtained from (j_1, j_2, j_3) as follow:

$$J(j_1, j_2, j_3) = \sum_{t=0}^{j_1 - 1} \sum_{l=0}^{p-1-t} (p + 1 - t - l) + j_1 + \sum_{l=j1}^{j_2 - 1} (p + 1 - l) + \big(j_3 - (j_2 - 1) \big). \tag{4.5}$$

Moreover, $\mathcal{Z}_3^n(\mathbf{t})$ is a vector of length:

$$l_{\mathcal{Z}_3} = J(p, p, p) = 1 + p + \frac{p(p+1)}{2} + l_{K_3}$$

where l_{K_3} has been defined in Remark 4.2[2]. Also let $\mathbf{t}$ and $\mathbf{s}$ be any two vectors in $\mathbb{R}^p$. Then,

$$\begin{aligned}
&\mathrm{Cov}(Z^{j_1j_2j_3}(\mathbf{t}), Z^{j_1'j_2'j_3'}(\mathbf{s})) \qquad (4.6)\\
=&\mathrm{Cov}(X_{j_1}X_{j_2}X_{j_3}\exp(\mathbf{t}'\mathbf{X}), X_{j_1'}X_{j_2'}X_{j_3'}\exp(\mathbf{s}'\mathbf{X}))\\
=&\mathbb{E}\left(X_{j_1}X_{j_2}X_{j_3}X_{j_1'}X_{j_2'}X_{j_3'}\exp\left((\mathbf{s}+\mathbf{t})'\mathbf{X}\right)\right) - \mathbb{E}\left(X_{j_1}X_{j_2}X_{j_3}\exp(\mathbf{t}'\mathbf{X})\right)\mathbb{E}\left(X_{j_1'}X_{j_2'}X_{j_3'}\exp(\mathbf{s}'\mathbf{X})\right)\\
=&\mu^{j_1j_2j_3j_1'j_2'j_3'}(\mathbf{t}+\mathbf{s}) - \mu^{j_1j_2j_3}(\mathbf{t})\mu^{j_1'j_2'j_3'}(\mathbf{s})
\end{aligned}$$

where $\mu^{j_1j_2j_3j_1'j_2'j_3'}(\mathbf{t}+\mathbf{s}) = \mathbb{E}\left(X_{j_1}X_{j_2}X_{j_3}X_{j_1'}X_{j_2'}X_{j_3'}\exp\left((\mathbf{s}+\mathbf{t})'\mathbf{X}\right)\right)$ can be calculated explicitly by using Lemma 4.3. Moreover, with $\mathbf{s}=\mathbf{t}$, we obtain the variance-covariance matrix of the multivariate vector $\mathcal{Z}_3(\mathbf{t})$. Elements of this covariance-variance matrix are defined by

$$\mathrm{Cov}(Z^{j_1j_2j_3}(\mathbf{t}), Z^{j_1'j_2'j_3'}(\mathbf{t})) = \mu^{j_1j_2j_3j_1'j_2'j_3'}(2\mathbf{t}) - \mu^{j_1j_2j_3}(\mathbf{t})\mu^{j_1'j_2'j_3'}(\mathbf{t}) \qquad (4.7)$$

With all these quantities defined, we now introduce the following theorem.

Theorem 4.1 (Asymptotic distribution of $\hat{\mathfrak{C}}^{(3)}$)**.** *Let* $T^{(j_1j_2j_3)(n)}(\mathbf{t}) = \sqrt{n}\hat{\mathfrak{C}}^{(j_1j_2j_3)}(\mathbf{t})$. *Let* $\mathcal{T}_3^n(\mathbf{t})$ *be a vector of length* l_{K_3}*, consisting of* $T^{(j_1j_2j_3)}(\mathbf{t}), 1 \le j_1 \le j_2 \le j_3 \le p$*, i.e*

$$\mathcal{T}_3^n(\mathbf{t}) = \left[T^{(j_1j_2j_3)(n)}(\mathbf{t})\right] = \left[T^{(111)(n)}(\mathbf{t}), T^{(112)(n)}(\mathbf{t}), \ldots, T^{(ppp)(n)}(\mathbf{t})\right]'.$$

Under the hypothesis that sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ *is from the multivariate standard normal distribution* $N_p(\mathbf{0}_p, \mathbf{I}_p)$*,* $\mathcal{T}_3^n(\mathbf{t})$ *approaches a multivariate normal distribution with mean* $\mathbf{0}$ *and some covariance matrix* $\mathbf{\Sigma}_{\mathcal{T}_3}$.

Proof. For fixed $\mathbf{t} \in \mathbb{R}^p$ and for any triplet (j_1, j_2, j_3) such that $1 \le j_1 \le j_2 \le j_3 \le p$, we write the expression of $T^{(j_1j_2j_3)(n)}(\mathbf{t})$ as

$$T^{(j_1j_2j_3)(n)}(\mathbf{t}) = \sqrt{n}\left(g_{j_1j_2j_3}(U_n(\mathbf{t})) - g_{j_1j_2j_3}(\xi(\mathbf{t}))\right)$$

[2]See Appendix B for proof that $J(p,p,p) = l_{\mathcal{Z}_3}$.

where

$$g_{j_1 j_2 j_3}(U_n(\mathbf{t})) = \frac{\partial^3}{\partial t_{j_1} \partial t_{j_2} \partial t_{j_3}} \log\left(\hat{M}(\mathbf{t})\right)$$

and

$$g_{j_1 j_2 j_3}(\xi(\mathbf{t})) = \frac{\partial^3}{\partial t_{j_1} \partial t_{j_2} \partial t_{j_3}} \log \mu_p(\mathbf{t}) = 0.$$

The expressions of vector $U_n(\mathbf{t})$ and $\xi(\mathbf{t})$ will vary according to specific scenarios of the ordered triplet (j_1, j_2, j_3). Let $\mathcal{J}_3$ be an ordered index corresponding to $j_1, j_2, j_3 \in \{1, 2, \ldots, p\}$. We list all possible format of the vectors $U_n(\mathbf{t})$ and $\xi(\mathbf{t})$ according to $\mathcal{J}_3$ below,

$$U_n'(\mathbf{t}) = \begin{cases} \left(\hat{M}_{\mathbf{t}}, \hat{M}_{\mathbf{t}}^{00j_1}, \hat{M}_{\mathbf{t}}^{00j_2}, \hat{M}_{\mathbf{t}}^{00j_3}, \hat{M}_{\mathbf{t}}^{0j_1j_2}, \hat{M}_{\mathbf{t}}^{0j_1j_3}, \hat{M}_{\mathbf{t}}^{0j_2j_3}, \hat{M}_{\mathbf{t}}^{j_1j_2j_3}\right), \text{ when } \mathcal{J}_3 = (j_1, j_2, j_3) \\ \left(\hat{M}_{\mathbf{t}}, \hat{M}_{\mathbf{t}}^{00j_1}, \hat{M}_{\mathbf{t}}^{00j_2}, \hat{M}_{\mathbf{t}}^{0j_1j_2}, \hat{M}_{\mathbf{t}}^{0j_2j_2}, \hat{M}_{\mathbf{t}}^{j_1j_2j_2}\right), \text{ when } \mathcal{J}_3 = (j_1, j_2, j_2) \\ \left(\hat{M}_{\mathbf{t}}, \hat{M}_{\mathbf{t}}^{00j_1}, \hat{M}_{\mathbf{t}}^{00j_2}, \hat{M}_{\mathbf{t}}^{0j_1j_1}, \hat{M}_{\mathbf{t}}^{0j_1j_2}, \hat{M}_{\mathbf{t}}^{j_1j_1j_2}\right), \text{ when } \mathcal{J}_3 = (j_1, j_1, j_2) \\ \left(\hat{M}_{\mathbf{t}}, \hat{M}_{\mathbf{t}}^{00j_1}, \hat{M}_{\mathbf{t}}^{0j_1j_1}, \hat{M}_{\mathbf{t}}^{j_1j_1j_1}\right), \text{ when } \mathcal{J}_3 = (j_1, j_1, j_1) \end{cases}$$

and

$$\xi'(\mathbf{t}) = \begin{cases} \left(\mu_{p(\mathbf{t})}, \mu_{\mathbf{t}}^{00j_1}, \mu_{\mathbf{t}}^{00j_2}, \mu_{\mathbf{t}}^{00j_3}, \mu_{\mathbf{t}}^{0j_1j_2}, \mu_{\mathbf{t}}^{0j_1j_3}, \mu_{\mathbf{t}}^{0j_2j_3}, \mu_{\mathbf{t}}^{j_1j_2j_3}\right), \text{ when } \mathcal{J}_3 = (j_1, j_2, j_3) \\ \left(\mu_{p(\mathbf{t})}, \mu^{00j_1}, \mu_{\mathbf{t}}^{00j_2}, \mu_{\mathbf{t}}^{0j_1j_2}, \mu_{\mathbf{t}}^{0j_2j_2}, \mu_{\mathbf{t}}^{j_1j_2j_2}\right), \text{ when } \mathcal{J}_3 = (j_1, j_2, j_2) \\ \left(\mu_{p(\mathbf{t})}, \mu^{00j_1}, \mu_{\mathbf{t}}^{00j_2}, \mu_{\mathbf{t}}^{0j_1j_1}, \mu_{\mathbf{t}}^{0j_1j_2}, \mu_{\mathbf{t}}^{j_1j_1j_2}\right), \text{ when } \mathcal{J}_3 = (j_1, j_1, j_2) \\ \left(\mu_{p(\mathbf{t})}, \mu_{\mathbf{t}}^{00j_1}, \mu_{\mathbf{t}}^{0j_1j_1}, \mu_{\mathbf{t}}^{j_1j_1j_1}\right), \text{ when } \mathcal{J}_3 = (j_1, j_1, j_1) \end{cases}$$

where the subscript $f_{\mathbf{t}}$ refers to $f(\mathbf{t}), \mathbf{t} \in \mathbb{R}^p$, for any function $f(.)$.

We apply the first order Taylor expansion of the function $g_{j_1 j_2 j_3}(U_n(\mathbf{t}))$ around $\xi(\mathbf{t})$ in each scenarios of $\mathcal{J}_3$ listed above. We will then prove that $\mathcal{T}_3^n(\mathbf{t}) \approx \mathcal{A}_3(\mathbf{t})\mathcal{Z}_3^n(\mathbf{t})$, where $\mathcal{A}_3(\mathbf{t})$ is a sparse matrix, and $\mathcal{Z}_3^n(\mathbf{t})$ is defined in (4.4).

- Case 1: $\mathcal{J}_3 = (j_1, j_2, j_3)$ with $j_1 \neq j_2 \neq j_3$. Taylor expansion of $g_{j_1 j_2 j_3}(U(\mathbf{t}))$ around $\xi(\mathbf{t})$ is

$$
\begin{aligned}
& g_{j_1 j_2 j_3}(U(\mathbf{t})) \\
=& g_{j_1 j_2 j_3}(\xi(\mathbf{t})) + \left(\hat{M}(\mathbf{t}) - \mu(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{00 j_1}(\mathbf{t}) - \mu^{00 j_1}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{00 j_1}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{00 j_2}(\mathbf{t}) - \mu^{00 j_2}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{00 j_2}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{00 j_3}(\mathbf{t}) - \mu^{00 j_3}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{00 j_3}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{0 j_1 j_2}(\mathbf{t}) - \mu^{0 j_1 j_2}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{0 j_1 j_2}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{0 j_1 j_3}(\mathbf{t}) - \mu^{0 j_1 j_3}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{0 j_1 j_3}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{0 j_2 j_3}(\mathbf{t}) - \mu^{0 j_2 j_3}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{0 j_2 j_3}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
& + \left(\hat{M}^{j_1 j_2 j_3}(\mathbf{t}) - \mu^{j_1 j_2 j_3}(\mathbf{t})\right) \frac{\partial}{\partial \hat{M}^{j_1 j_2 j_3}} g_{j_1 j_2 j_3}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \qquad + R_n^{j_1 j_2 j_3}(\mathbf{t})
\end{aligned}
$$

where $\sup_{t \in B} \left| R_n^{j_1 j_2 j_3}(\mathbf{t}) \right|$ converges to 0 in probability on a bounded set $B \subset \mathbb{R}^p$. Specifically,

$$
\begin{aligned}
& g_{j_1 j_2 j_3}(U_n(\mathbf{t})) \\
&= \frac{\hat{M}_{\mathbf{t}}^{j_1 j_2 j_3}}{\hat{M}_{\mathbf{t}}} - \left[\frac{\hat{M}_{\mathbf{t}}^{0 j_1 j_2} \hat{M}_{\mathbf{t}}^{00 j_3} + \hat{M}_{\mathbf{t}}^{0 j_1 j_3} \hat{M}_{\mathbf{t}}^{00 j_2} + \hat{M}_{\mathbf{t}}^{0 j_2 j_3} \hat{M}_{\mathbf{t}}^{00 j_1}}{\hat{M}_{\mathbf{t}}^2} \right] + 2 \frac{\hat{M}_{\mathbf{t}}^{00 j_1} \hat{M}_{\mathbf{t}}^{00 j_2} \hat{M}_{\mathbf{t}}^{00 j_3}}{\hat{M}_{\mathbf{t}}^3}
\end{aligned}
$$

and we obtain its derivatives as follows:

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = \left(-\frac{\hat{M}_{\mathbf{t}}^{j_1 j_2 j_3}}{\hat{M}_{\mathbf{t}}^2} - 6\frac{\hat{M}_{\mathbf{t}}^{00 j_1}\hat{M}_{\mathbf{t}}^{00 j_2}\hat{M}_{\mathbf{t}}^{00 j_3}}{\hat{M}_{\mathbf{t}}^4} + 2\frac{\hat{M}_{\mathbf{t}}^{0 j_1 j_2}\hat{M}_{\mathbf{t}}^{00 j_3} + \hat{M}_{\mathbf{t}}^{0 j_1 j_3}\hat{M}_{\mathbf{t}}^{00 j_2} + \hat{M}_{\mathbf{t}}^{0 j_2 j_3}\hat{M}_{\mathbf{t}}^{00 j_1}}{\hat{M}_{\mathbf{t}}^3} \right)\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{00 j_1}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = 2\frac{\hat{M}_{\mathbf{t}}^{00 j_2}\hat{M}_{\mathbf{t}}^{00 j_3}}{\hat{M}_{\mathbf{t}}^3} - \frac{\hat{M}_{\mathbf{t}}^{0 j_2 j_3}}{\hat{M}_{\mathbf{t}}^2}\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{00 j_2}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = 2\frac{\hat{M}_{\mathbf{t}}^{00 j_1}\hat{M}_{\mathbf{t}}^{00 j_3}}{\hat{M}_{\mathbf{t}}^3} - \frac{\hat{M}_{\mathbf{t}}^{0 j_1 j_3}}{\hat{M}_{\mathbf{t}}^2}\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{00 j_3}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = 2\frac{\hat{M}_{\mathbf{t}}^{00 j_1}\hat{M}_{\mathbf{t}}^{00 j_2}}{\hat{M}_{\mathbf{t}}^3} - \frac{\hat{M}_{\mathbf{t}}^{0 j_1 j_2}}{\hat{M}_{\mathbf{t}}^2}\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0 j_1 j_2}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = -\frac{\hat{M}_{\mathbf{t}}^{00 j_3}}{\hat{M}_{\mathbf{t}}^2}\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0 j_1 j_3}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = -\frac{\hat{M}_{\mathbf{t}}^{00 j_2}}{\hat{M}_{\mathbf{t}}^2}\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0 j_2 j_3}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = -\frac{\hat{M}_{\mathbf{t}}^{00 j_1}}{\hat{M}_{\mathbf{t}}^2}\Bigg|_{\xi(t)},$$

$$\frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{j_1 j_2 j_3}} g_{j_1 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(t)} = \frac{1}{\hat{M}_{\mathbf{t}}}\Bigg|_{\xi(t)}.$$

Together with $\xi(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[1, t_{j_1}, t_{j_2}, t_{j_3}, t_{j_1}t_{j_2}, t_{j_1}t_{j_3}, t_{j_2}t_{j_3}, t_{j_1}t_{j_2}t_{j_3}\right]'$, above expansion yields:

$$T^{(j_1 j_2 j_3)(n)}(\mathbf{t}) \approx \left[\mathcal{A}_3^{j_1 j_2 j_3}(\mathbf{t})\right]' \mathcal{Z}_3^n(\mathbf{t}) + R_n(\mathbf{t}),$$

where $\mathcal{A}_3^{j_1 j_2 j_3}(\mathbf{t})$ is a sparse vector of length $l_{\mathcal{Z}_3}$. Non-zero elements of $\mathcal{A}_3^{j_1 j_2 j_3}(\mathbf{t})$ are accordingly defined by (j_1, j_2, j_3), and has the explicit expression as follow,

$$\begin{aligned}\left[\mathcal{A}_3^{j_1 j_2 j_3}(\mathbf{t})\right]_{\neq 0} &= \left[a_{\mathbf{t}}^{000}, a_{\mathbf{t}}^{00j_1}, a_{\mathbf{t}}^{00j_2}, a_{\mathbf{t}}^{00j_3}, a_{\mathbf{t}}^{0j_1j_2}, a_{\mathbf{t}}^{0j_1j_3}, a_{\mathbf{t}}^{0j_2j_3}, a_{\mathbf{t}}^{j_1j_2j_3}\right]' \\ &= \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[-t_{j_1}t_{j_2}t_{j_3}, t_{j_2}t_{j_3}, t_{j_1}t_{j_3}, t_{j_1}t_{j_2}, -t_{j_3}, -t_{j_2}, -t_{j_1}, 1\right]'.\end{aligned}$$

The position of $a_{\mathbf{t}}^{l_1 l_2 l_3}$ inside vector $\mathcal{A}_3^{j_1 j_2 j_3}(\mathbf{t})$ can be retrieved by the formulas in (4.5).

- Case 2: $\mathcal{J}_3 = (j_1, j_2, j_2)$ with $j_1 < j_2$:

$$g_{j_1 j_2 j_2}(U_n(\mathbf{t})) = \frac{\hat{M}_{\mathbf{t}}^{j_1 j_2 j_2}}{\hat{M}_{\mathbf{t}}} + 2\frac{\hat{M}_{\mathbf{t}}^{00j_1}\left(\hat{M}_{\mathbf{t}}^{00j_2}\right)^2}{\hat{M}_{\mathbf{t}}^3} - \frac{2\hat{M}_{\mathbf{t}}^{0j_1j_2}\hat{M}_{\mathbf{t}}^{00j_2} + \hat{M}_{\mathbf{t}}^{0j_2j_2}\hat{M}_{\mathbf{t}}^{00j_1}}{\hat{M}_{\mathbf{t}}^2}$$

and its derivatives are

$$\begin{aligned}\frac{\partial}{\partial \hat{M}_{\mathbf{t}}} g_{j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= \left(-\frac{\hat{M}_{\mathbf{t}}^{j_1 j_2 j_2}}{\hat{M}_{\mathbf{t}}^2} - 6\frac{\hat{M}_{\mathbf{t}}^{00j_1}\left(\hat{M}_{\mathbf{t}}^{00j_2}\right)^2}{\hat{M}_{\mathbf{t}}^3}\right. \\ &\qquad \left. +2\frac{2\hat{M}_{\mathbf{t}}^{0j_1j_2}\hat{M}_{\mathbf{t}}^{00j_2} + \hat{M}_{\mathbf{t}}^{0j_2j_2}\hat{M}_{\mathbf{t}}^{00j_1}}{\hat{M}_{\mathbf{t}}^3}\right)\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{00j_1}} g_{j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= 2\frac{\left(\hat{M}_{\mathbf{t}}^{00j_2}\right)^2}{\hat{M}_{\mathbf{t}}^3} - \frac{\hat{M}_{\mathbf{t}}^{0j_2j_2}}{\hat{M}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{00j_2}} g_{j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= 4\frac{\hat{M}_{\mathbf{t}}^{00j_1}}{\hat{M}_{\mathbf{t}}^3} - 2\frac{\hat{M}_{\mathbf{t}}^{0j_1j_2}}{\hat{M}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0j_1j_2}} g_{j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= -2\frac{\hat{M}_{\mathbf{t}}^{00j_2}}{\hat{M}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0j_2j_2}} g_{j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= -\frac{\hat{M}_{\mathbf{t}}^{00j_1}}{\hat{M}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})}.\end{aligned}$$

With $\xi(\mathbf{t}) = \exp\left(\frac{t't}{2}\right)\left[1, t_{j_1}, t_{j_2}, t_{j_1}t_{j_2}, (1+t_{j_2}^2), t_{j_1}(1+t_{j_2}^2)\right]'$, we obtain

$$\begin{aligned}\left[\mathcal{A}_3^{j_1j_2j_2}(\mathbf{t})\right]_{\neq 0} &= \left[a_{\mathbf{t}}^{000}, a_{\mathbf{t}}^{00j_1}, a_{\mathbf{t}}^{00j_2}, a_{\mathbf{t}}^{0j_1j_2}, a_{\mathbf{t}}^{0j_2j_2}, a_{\mathbf{t}}^{j_1j_2j_2}\right]' \\ &= \exp\left(\frac{-\mathbf{t}'\mathbf{t}}{2}\right)\left[-t_{j_1}(t_{j_2}^2-1), t_{j_2}^2-1, 2t_{j_1}t_{j_2}, -2t_{j_2}, -t_{j_1}, 1\right]'.\end{aligned}$$

- Case 3: $\mathcal{J}_3 = (j_1, j_1, j_2)$ with $j_1 < j_2$. Applying similar calculations as Case 2, we immediately obtain,

$$\begin{aligned}\left[\mathcal{A}_3^{j_1j_1j_2}(\mathbf{t})\right]'_{\neq 0} &= \left[a_{\mathbf{t}}^{000}, a_{\mathbf{t}}^{00j_1}, a_{\mathbf{t}}^{00j_2}, a_{\mathbf{t}}^{0j_1j_1}, a_{\mathbf{t}}^{0j_1j_2}, a_{\mathbf{t}}^{j_1j_2j_2}\right]' \\ &= \exp\left(\frac{-\mathbf{t}'\mathbf{t}}{2}\right)\left[-(t_{j_1}^2-1)t_{j_3}, 2t_{j_1}t_{j_3}, t_{j_1}^2-1, -t_{j_3}, -2t_{j_1}, 1\right]'.\end{aligned}$$

- Case 4: $\mathcal{J}_3 = (j_1, j_1, j_1)$:

$$g_{j_1j_1j_1}(U_n(\mathbf{t})) = \frac{\hat{M}_{\mathbf{t}}^{j_1j_1j_1}}{\hat{M}_{\mathbf{t}}} + 2\frac{\left(\hat{M}_{\mathbf{t}}^{00j_1}\right)^3}{\hat{M}_{\mathbf{t}}^3} - 3\frac{\hat{M}_{\mathbf{t}}^{00j_1}\hat{M}_{\mathbf{t}}^{0j_1j_1}}{\hat{M}_{\mathbf{t}}^2}$$

and its derivatives are as follows,

$$\begin{aligned}\frac{\partial}{\partial \hat{M}_{\mathbf{t}}} g_{j_1j_1j_1}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= -\frac{\hat{M}_{\mathbf{t}}^{j_1j_1j_1}}{\hat{M}_{\mathbf{t}}^2} - 6\frac{\left(\hat{M}_{\mathbf{t}}^{00j_1}\right)^3}{\hat{M}_{\mathbf{t}}^4} + 6\frac{\hat{M}_{\mathbf{t}}^{00j_1}M^{0j_1j_1}}{\hat{M}_{\mathbf{t}}^3}\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial M_{\mathbf{t}}^{00j_1}} g_{j_1j_1j_1}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= 6\frac{(\hat{M}_{\mathbf{t}}^{00j_1})^2}{\hat{M}_{\mathbf{t}}^3} - 3\frac{\hat{M}_{\mathbf{t}}^{0j_1j_1}}{\hat{M}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0j_1j_1}} g_{j_1j_1j_1}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= -3\frac{\hat{M}_{\mathbf{t}}^{00j_1}}{\hat{M}_{\mathbf{t}}{}^2}\Big|_{\xi(\mathbf{t})}, \\ \frac{\partial}{\partial \hat{M}_{\mathbf{t}}^{0j_1j_1}} g_{j_1j_1j_1}(U(\mathbf{t}))\Big|_{\xi(\mathbf{t})} &= \frac{1}{\hat{M}_{\mathbf{t}}}\Big|_{\xi(\mathbf{t})}.\end{aligned}$$

Together with $\xi(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[1, t_1, 1+t_1^2, t_1^3+3t_1\right]'$, we obtain

$$\begin{aligned}\left[\mathcal{A}_3^{j_1j_1j_1}(\mathbf{t})\right]_{\neq 0} &= \left[a_{\mathbf{t}}^{000}, a_{\mathbf{t}}^{00j_1}, a_{\mathbf{t}}^{0j_1j_1}, a_{\mathbf{t}}^{j_1j_1j_1}\right]' \\ &= \exp\left(\frac{-\mathbf{t}'\mathbf{t}}{2}\right)\left[3t_{j_1} - t_{j_1}^3, 3(t_{j_1}^2-1), -3t_{j_1}, 1\right]'.\end{aligned}$$

We just proved that for each choice of index $1 \le j_1 \le j_2 \le j_3 \le p$,

$$T^{(j_1j_2j_3)(n)} = \left[\mathcal{A}_3^{j_1j_2j_3}(\mathbf{t})\right]' \mathcal{Z}_3^n(\mathbf{t}) + R_n^{j_1j_2j_3}(\mathbf{t})$$

where $\sup_{\mathbf{t}\in\mathbf{B}}\left|R_n^{j_1j_2j_3}(\mathbf{t})\right|$ converges to 0 in probability on a bounded subset $\mathbf{B} \subset \mathbb{R}^p$. Therefore, we immediately obtain

$$\mathcal{T}_3^n(t) = \left(T^{(j_1j_2j_3)(n)}\right) = \mathcal{A}_{3(l_{K_3}\times l_{\mathcal{Z}_3})}(\mathbf{t}) \times \mathcal{Z}_3^n(\mathbf{t}) + R_3^n(\mathbf{t})$$

with $R_3^n(\mathbf{t}) = \left[R_n^{111}(\mathbf{t}), R_n^{112}(\mathbf{t}), \ldots, R_n^{ppp}(\mathbf{t})\right]'$ and

$$\mathcal{A}_3(\mathbf{t}) = \begin{bmatrix} \left[\mathcal{A}_3^{111}(\mathbf{t})\right]' \\ \left[\mathcal{A}_3^{112}(\mathbf{t})\right]' \\ \cdots \\ \left[\mathcal{A}_3^{ppp}(\mathbf{t})\right]' \end{bmatrix}.$$

Here, $\mathcal{A}_3^{j_1j_2j_3}(\mathbf{t})$ is a sparse vector defined separately in each case of $\mathcal{J}_3$ given above.

In view of multivariate central limit theorem the random vector $\mathcal{Z}_3^n(\mathbf{t})$ is asymptotically normally distributed. Applying Slutsky's theorem to the random vector $\mathcal{Z}_3^n(\mathbf{t})$, it follows that $\mathcal{T}_3^n(t)$ converges in probability to a multivariate normal vector of length l_{K_3}, with mean $\mathbf{0}_{l_{K_3}}$ and covariance matrix $\mathbf{\Sigma}_{\mathcal{T}_3}(\mathbf{t}) = [\mathcal{A}_3(\mathbf{t})]\, \mathbf{\Sigma}_{\mathcal{Z}_3^n}(\mathbf{t})\, [\mathcal{A}_3(\mathbf{t})]'$. □

The lemma given below follows from Theorem 4.1.

Lemma 4.4. For a fixed vector $v \in \mathbb{R}^{l_{K_3}}$, the statistic $L_3^v(\mathbf{t}) = v' \left[\mathcal{T}_3^n(\mathbf{t})\right]$ approaches a normally distributed random vector $\mathcal{L}_3^v(\mathbf{t}) \sim N(0, v'\mathbf{\Sigma}_{\mathcal{T}_3} v)$[3]. Moreover,

$$\begin{aligned}
\mathrm{Cov}(\mathcal{L}_3^v(\mathbf{t}), \mathcal{L}_3^v(\mathbf{s})) &= \mathrm{Cov}\left(v' \left[\mathcal{T}_3^n(\mathbf{t})\right], v' \left[\mathcal{T}_3^n(\mathbf{s})\right]\right) \\
&= v' \,\mathrm{Cov}\left(\left[\mathcal{T}_3^n(\mathbf{t})\right], \left[\mathcal{T}_3^n(\mathbf{s})\right]\right) v \\
&= v' \left[\mathcal{A}_3(\mathbf{t})\right] \mathrm{Cov}\left(\mathcal{Z}_3^n(\mathbf{t}), \mathcal{Z}_3^n(\mathbf{s})\right) \left[\mathcal{A}_3(\mathbf{s})\right]' v \\
&= \left[\mathcal{A}_3(\mathbf{t})' v\right]' \mathrm{Cov}\left(\mathcal{Z}_3^n(\mathbf{t}), \mathcal{Z}_3^n(\mathbf{s})\right) \left[\mathcal{A}_3(\mathbf{s})' v\right].
\end{aligned}$$

For the purpose of obtaining a two-dimensional plot, we must define a function that takes the vector $\mathbf{t}$ to one dimension. With the choice of $\mathbf{t} = t^* \mathbf{1}_p'$ and $\mathbf{s} = s^* \mathbf{1}_p'$, $\mathcal{L}_3^v(\mathbf{t}) = \mathcal{L}_3^v(t^*)$ will be a Gaussian process with the kernel function

$$\kappa_3^v(t^*, s^*) = \mathrm{Cov}\left(\mathcal{L}_3^v(t^*), \mathcal{L}_3^v(s^*)\right).$$

Upon the standardization,

$$\tilde{L}_3^v(t^*) = \frac{L_3^v(t^*)}{\sqrt{\mathrm{Var}(\mathcal{L}_3^v(t^*))}},$$

converges in probability to a Gaussian process $\tilde{\mathcal{L}}_3^v(t^*)$ with mean 0 and variance of 1, and the covariance function

$$\mathrm{Cov}\left(\tilde{\mathcal{L}}_3^v(t^*), \tilde{\mathcal{L}}_3^v(s^*)\right) = \frac{\kappa_3^v(t^*, s^*))}{\sqrt{\kappa_3^v(t^*, t^*) \times \kappa_3^v(s^*, s^*)}}. \tag{4.8}$$

Borell's inequality (Lemma 2.7) provides us with the confidence limit for the supremum of any Gaussian process as follows,

$$P\left(\sup_{t^* \in [-1,1]} |\tilde{L}_3^v(t^*)| > \mathbb{E}\left(\sup_{t^* \in -[1,1]} \left|\tilde{L}_3^v(t^*)\right|\right) + u\right) \leq \exp\left(\frac{-u^2}{2}\right), \forall u > 0$$

which then implies

$$P\left(\sup_{t^* \in [-1,1]} |\tilde{L}_3^v(t^*)| \leq \mathbb{E}\left(\sup_{t^* \in -[1,1]} \left|\tilde{L}_3^v(t^*)\right|\right) + u\right) \leq 1 - \exp\left(\frac{-u^2}{2}\right).$$

[3]We have used the upper subscript v to indicate the dependence of the distribution on the choice of test vector $v \in \mathbb{R}^{l_{K_3}}$.

Let $m^v_{p3} = \mathbb{E}\left(\sup_{t^*\in[-1,1]} \left|\tilde{L}^v_3(t^*)\right|\right)$. Due to the complexity of the supremum, it may not be possible to obtain a closed form of m^v_{p3}. As $n \to +\infty$, m^v_{p3} converges to $\mathbb{E}\left(\sup_{t^*\in[-1,1]} \left|\tilde{\mathcal{L}}^v_3(t^*)\right|\right)$, where $\tilde{\mathcal{L}}^v_3(t^*) = \dfrac{\mathcal{L}^v_3(t^*)}{\sqrt{\text{Var}(\mathcal{L}^v_3(t^*))}}$, an asymptotic estimator $\hat{m}^v_{p3}$ can be obtained by simulating the Gaussian process $\tilde{\mathcal{L}}^v_3(t^*)$ with mean 0 and covariance function defined in (4.8).

A suitable candidate for a functional test statistic is then

$$\mathfrak{L} = \sup_{t^*\in[-1,1]} \left|\tilde{L}^v_3(t^*)\right| = \sup_{t^*\in[-1,1]} \left|\frac{L^v_3(t^*)}{\sqrt{\text{Var}(\mathcal{L}^v_3(t^*))}}\right|. \tag{4.9}$$

The critical value for size α test can be taken as $\hat{m}^v_{p3} + u_\alpha$, where u_α is chosen such that $\exp\left(\dfrac{-u^2_\alpha}{2}\right) \leq \alpha$.

Using Lemma 4.1, the derivatives of $\tilde{L}_3(t^*)$ are close to 0, and this piece of information might be missed if we confined ourselves to the supremum test statistic $\mathfrak{L}$ of (4.9) given above. On the other hand, plotting the curve $\tilde{L}_3(t^*)$ as a function of t^* might provide more information. We known that with probability $(1-\alpha)$,

$$\frac{L^v_3(t^*)}{\sqrt{\text{Var}(\mathcal{L}^v_3(t^*))}} \in \pm\mathbb{E}\left(\sup_{t^*\in[-1,1]} \left|\frac{L^v_3(t^*)}{\sqrt{\text{Var}(\mathcal{L}^v_3(t^*))}}\right|\right) + u_\alpha, \forall t^* \in [-1,1].$$

Thus, we can obtain a reasonable 2-D plot to test the normality for any specific choice of the vector $v \in \mathbb{R}^{l_{K_3}}$. For a given vector v, a $(1-\alpha)100\%$ confidence region for the curve $L^v_3(t^*)$ is defined as $\pm\left(\hat{m}^v_{p3} + u_\alpha\right)\sqrt{\text{Var}(\mathcal{L}^v_3(t^*))}$, where $u_\alpha = \sqrt{-2\log(\alpha)}$. In this expression, the quantity $\hat{m}^v_{p3}$ may be obtained via simulation.

Remark. The choice of vector v is rather arbitrarily. We will here consider the choice $v = \mathbf{a}_3 = \dfrac{1}{\sqrt{l_{K_3}}}\mathbf{1}_{l_{K_3}}$ and hence $L^{\mathbf{a}_3}_3(.)$ is the average of all distinct estimators of third derivatives of cumulant generating function,

$$L^{\mathbf{a}_3}_3(\mathbf{t}) = \frac{1}{\sqrt{l_{K3}}}\mathbf{1}'_{l_{K_3}}\left[\mathcal{T}^n_3(\mathbf{t})\right] \tag{4.10}$$

Also, note that,

$$\text{Var}(\mathcal{T}_3(\mathbf{t})) = \mathbf{\Sigma}_{\mathcal{T}_3}(\mathbf{t}) = \mathcal{A}_3(\mathbf{t})\text{Var}\left[\mathcal{Z}_3(\mathbf{t})\right]\mathcal{A}_3'(\mathbf{t})$$

becomes very "large" as $\|\mathbf{t}\|_p$ increases. Thus, it is meaningful to choose test points $\mathbf{t} \in \mathbb{R}^p$ such that $\|\mathbf{t}\|_p \leq 1$. One suggestion is to take $\|\mathbf{t}\|_p = t^* \leq 1$, and consider vector $\mathbf{t}$ as

$$\mathbf{t} = \frac{1}{\sqrt{p}} t^* \mathbf{1}_p, t^* \in [-1, 1].$$

4.2.2 Asymptotic Distribution of $\hat{K}^{(3)}(\mathbf{t})$: Multivariate Non-standard Normal Case

In the previous, we obtained the asymptotic properties of $\hat{\mathfrak{C}}^{(3)}(\mathbf{t}) = \dfrac{\partial^3}{\partial \mathbf{t}^3} \log\left[\hat{M}(\mathbf{t})\right]$, under the assumption of multivariate standard normal distribution. To extend this idea to the general multivariate normal case, we must rely on its standardized version $\hat{\mu}^{(3)}(\mathbf{t}) = \dfrac{\partial^3}{\partial \mathbf{t}^3} \log\left[\hat{\mu}(\mathbf{t})\right]$.

The multivariate central limit theorem ensures the vector $\mathcal{Z}_3^n(\mathbf{t})$ defined in (4.4) converges almost surely to a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Sigma}_{\mathcal{Z}_3(\mathbf{t})}$ as defined in (4.7), for every $\mathbf{t} \in [-1, 1]^p$. Thus, for large n, the random vector $\tilde{\mathcal{Z}}_3^n(\mathbf{t})\Big|_{\tilde{\mathbf{X}}_i = \boldsymbol{S}^{-1/2}(\mathbf{X}_i - \bar{\mathbf{X}})}$ also converges to the same multivariate normal distribution. Accordingly, the cross covariance between respective elements of $\tilde{\mathcal{Z}}_3^n(\mathbf{t})$ and $\tilde{\mathcal{Z}}_3^n(\mathbf{s})$ are given by (4.6). Hence, Theorem 4.1 holds under the standardization of data and the use of studentized empirical mgf $\hat{\mu}(\mathbf{t})$. Therefore, for large samples, it is reasonable to work with the sample-based standardized data $\boldsymbol{S}^{-1/2}(\mathbf{X}_1 - \bar{\mathbf{X}}), \boldsymbol{S}^{-1/2}(\mathbf{X}_2 - \bar{\mathbf{X}}), \ldots, \boldsymbol{S}^{-1/2}(\mathbf{X}_n - \bar{\mathbf{X}})$.

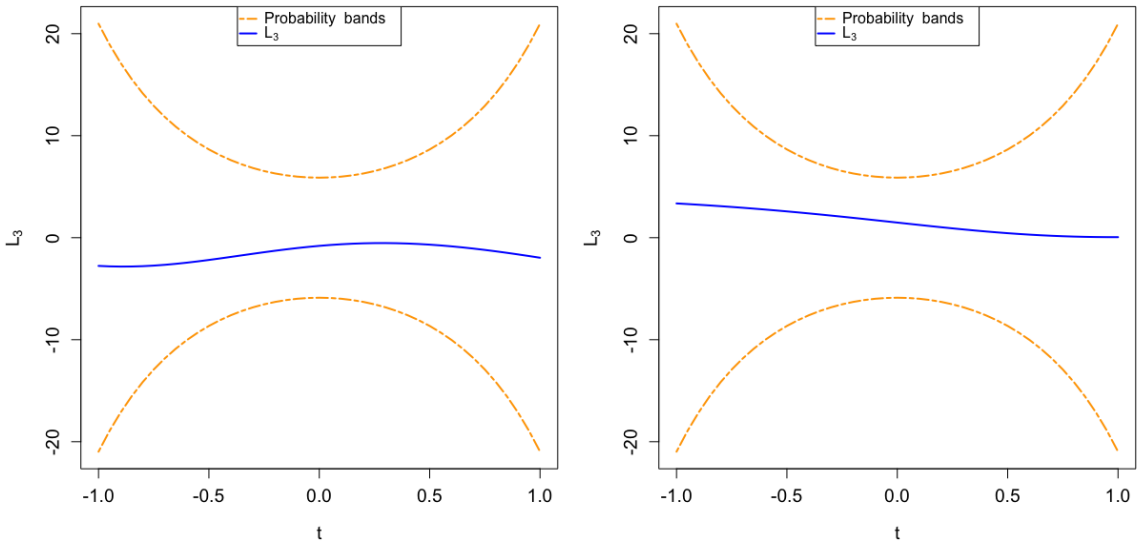
Accordingly the functional statistic $L_3(t^*)$ of Definition 4.1 will have the same asymptotic distribution as the $L_3^{\mathbf{a}_3}(t^*)$ defined in (4.10). The $(1-\alpha)100\%$ confidence band for $L_3(t^*)$ in an MT_3 plot is then given by

$\pm\left(\hat{m}_{p3}^{\mathbf{a}_3} + u_\alpha\right)\sqrt{\text{Var}(\mathcal{L}_3^{\mathbf{a}_3}(t^*))}$, where $u_\alpha = \sqrt{-2 * \log(\alpha)}$.

4.2.3 Illustrative Examples: Simulated Data

We here construct MT_3 plots to assess the multivariate normality of various data sets generated from several five-dimensional distributions. All these datasets have already been considered in Section 3.3. For us, here, with $p = 5$, $\mathbf{a}_3$ is a vector of length 35 with all elements equal to $1/\sqrt{35}$. With 41 equally spaced values of $t^* \in [-1, 1]$, i.e, with $t^* = 0, \pm 0.05, \cdots \pm 1$, we first simulate the zero mean Gaussian process $\tilde{\mathcal{L}}_3^{\mathbf{a}_3}(t^*)$ with covariance matrix given in (4.8). We have taken the number of simulations $m = 4000$. From these 41-variate vectors we obtain $\hat{m}_{p3}^{\mathbf{a}_3} = 1.444$. The choice of $\mathbf{t} = \dfrac{1}{\sqrt{p}} t^* \mathbf{1}_p$ is used, and for probability bands, we have chosen $1 - \alpha = 0.95$. The departure from normality is judged by the violations to $(1 - \alpha)$ probability bands and from the non-flatness of the curve of our functional statistic.

Figure 4.5 represents two MT_3 plots for data generated from two different five-dimensional normal distributions. Both curves fall within the critical bands and have nearly zero gradients, rightly reflecting the typical characteristic of cumulant functions. The non-normal cases are presented thereafter in Figure 4.6. Since MT_3 relies on the third derivatives of the cumulant generating function, one could expect that the skewed distributions will show clearly visible evidence of non-normality. We see that in Figures 4.6a, 4.6b, 4.6c. Especially, for the highly skewed Lomax case, this results in very large values (notice the drastic rescaling of the vertical axis relative to other plots). On the other hand, for the symmetric distributions, as in the case of $t(15)$ in Figure 4.6f and Kotz type distribution in Figure 4.6d, MT_3 gradients in plots are not as vivid, and the plots are also confined within the bands. However, MT_3 gives very strong evidence of non-normality when data is generated from $t(5)$, as in Figure 4.6e. The small degree of freedom apparently still results in fat enough tails so that non-normality is identified despite the symmetry.



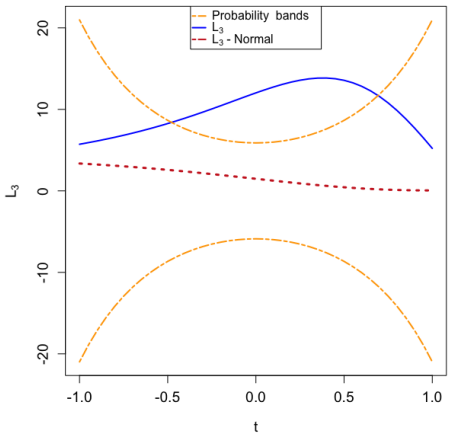
(a) $N_5(\mathbf{0}_5, \mathbf{I}_5)$ (b) $N_5\left(\boldsymbol{\mu} = [1,2,3,4,5]', \boldsymbol{\Sigma} = 2\left(.7\mathbf{I}_5 + .3\mathbf{J}_5\right)\right)$

Figure 4.5: MT_3 plots for normally distributed data.

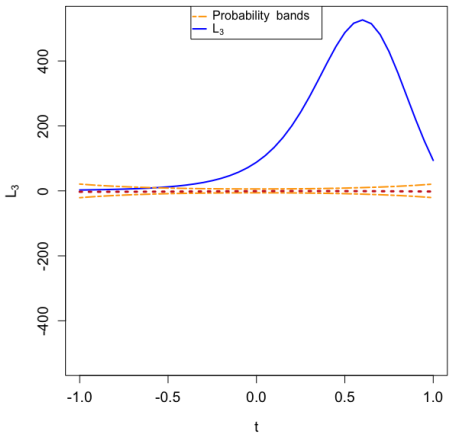
4.2.4 Two Examples: Cork Data and Water Potability Data Revisited

Example 4.2.1 (Cork Data)**.** As discussed earlier, the Cork dataset consists of 28 four-variate observations, with each component being marginally normal. As shown in Figure 4.7, the corresponding MT_3 plot supports the hypothesis of multivariate normality. The curve for the functional statistics $L_3(.)$ is flat and lies well within the two probability bands.

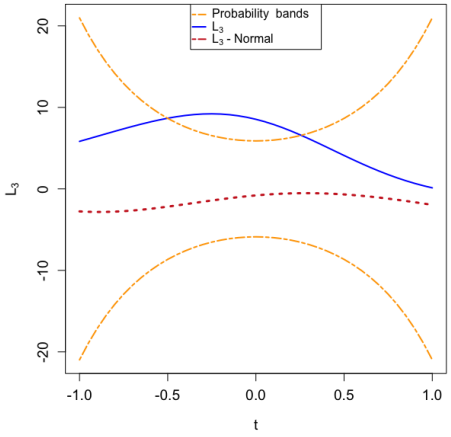
Example 4.2.2 (Water Potability Data)**.** We had earlier found using other approaches that the variables *Organic carbon, Turbidity* and *Trihalomethanes* can be assumed to be marginally and jointly normally distributed. MT_3 plot in Figure 4.8 as a graphical assessment for trivariate normality gives a curve falling within the confidence region, but the downward trend on the left tail gives evidence of some non-normality.



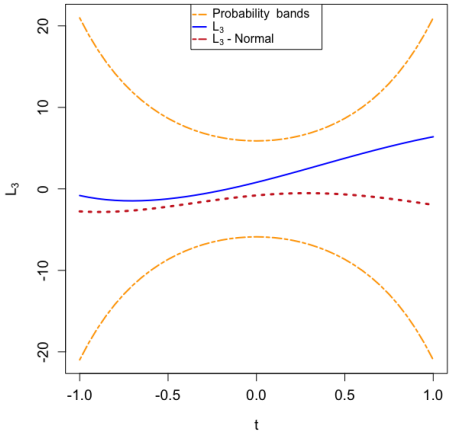
(a) Data generated from copulas



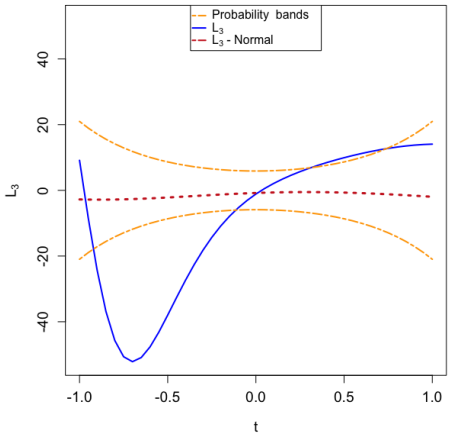
(b) Lomax ($\alpha = 1, \beta = \mathbf{1}_5$)



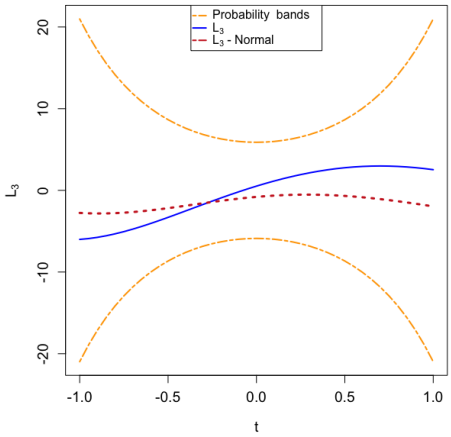
(c) Mixture normal



(d) Kotz($\boldsymbol{\mu} = \mathbf{0}_5, \Sigma = \mathbf{I}_5$)



(e) $t_5(\boldsymbol{\Sigma} = \mathbf{I}_5)$



(f) $t_{15}(\boldsymbol{\Sigma} = \mathbf{I}_5)$

Figure 4.6: MT_3 plots for non-normal data.

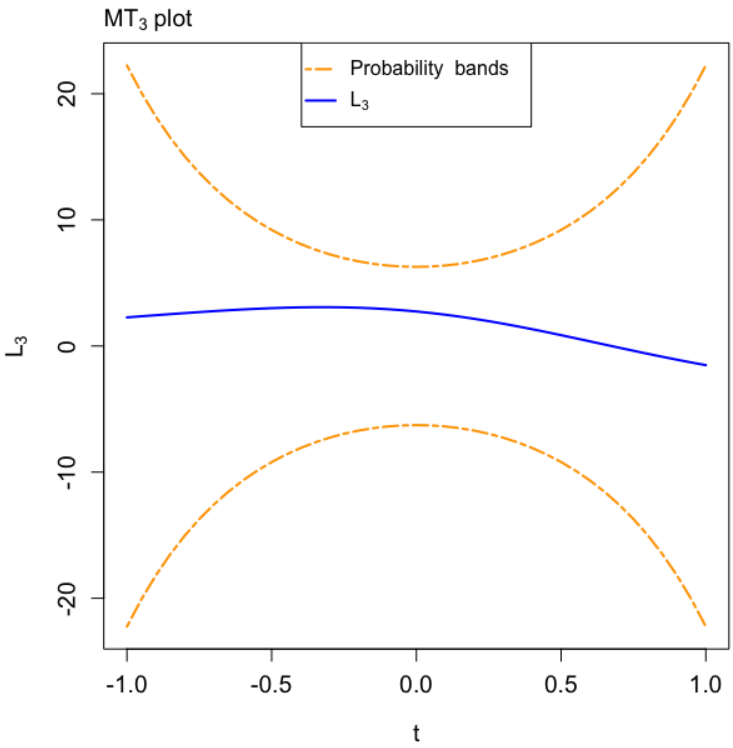


Figure 4.7: MT_3 plot for Cork dataset.

4.3 MT_4 Plot

The MT_3 seems to capture the departure from normality more in terms of skewness and symmetry. Why shouldn't one rely on derivatives of cumulant generating function higher than three? They must certainly also discover the other hidden features of nonnormality. As a development, our focus shifts toward the thickness of the tails. This section will thus introduce a version of the graphical test based on the fourth derivatives of the cumulant generating function, which we will call MT_4 plot. Admittedly, the technical derivation of MT_4 is much more complex, as we are now dealing with a $p \times p \times p \times p$ array. As in Section 4.2, we will now obtain a functional statistic curve $L_4^u(\mathbf{t})$ based on the fourth derivatives of the standardized moment generating function $\hat{\mu}(\mathbf{t})$. The following lemma will be helpful in our subsequent development.

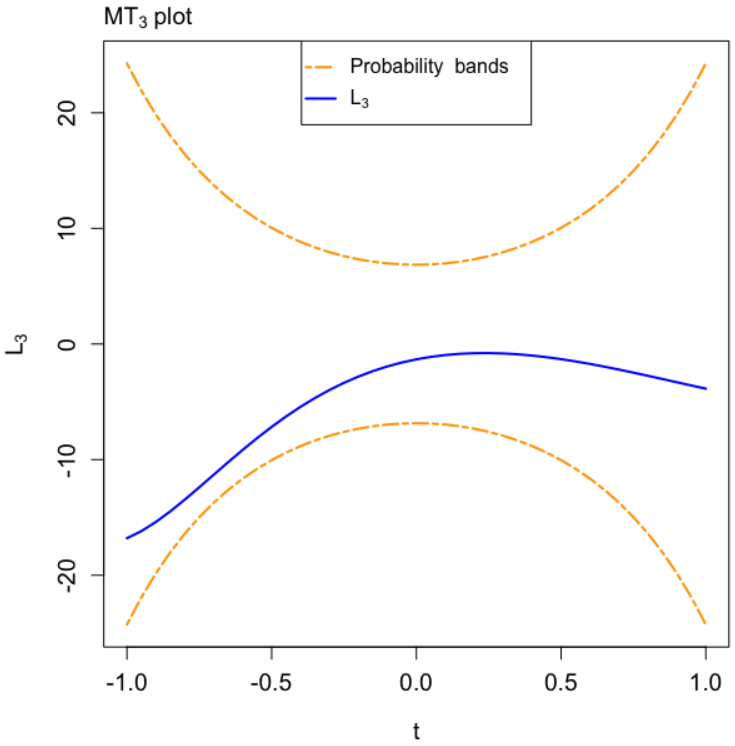


Figure 4.8: MT_3 plot for Water Potability (= Good) dataset; variables considered are *Organic carbon, Turbidity, Trihalomethanes*.

Let $P_i(\mathbf{t}) = \dfrac{\exp(\mathbf{t}'\tilde{\mathbf{X}}_i)}{\sum_{k=1}^{n}\exp(\mathbf{t}'\tilde{\mathbf{X}}_k)} \geq 0, i = 1, 2, \ldots, n$ as defined earlier. Then

$$\frac{\partial}{\partial t_j}P_i = \frac{\partial}{\partial t_j}\frac{\exp(\mathbf{t}'\mathbf{X}_i)}{\sum_{k=1}^{n}\exp(\mathbf{t}'\mathbf{X}_k)} = X_{ij}P_i - P_i\sum_{k=1}^{n}X_{kj}P_k. \tag{4.11}$$

Moreover, if A_i are coefficient that do not depending on $\mathbf{t}$, then

$$\frac{\partial}{\partial t_j}\left(\sum_{i=1}^{n}A_iP_i\right) = \sum_{i=1}^{n}A_iX_{ij}P_i - \left(\sum_{i=1}^{n}A_iP_i\right)\left(\sum_{i=1}^{n}X_{ij}P_i\right). \tag{4.12}$$

As in the case of $\hat{K}^{j_1j_2j_3}(\mathbf{t})$, now let $\hat{K}^{(j_1j_2j_3j_4)}(\mathbf{t})$ be the element indexed by the set (j_1, j_2, j_3, j_4) in the four dimensional tensor. Thus,

$$\hat{K}^{(j_1j_2j_3j_4)}(\mathbf{t}) = \frac{\partial^4}{\partial t_{j_1}t_{j_2}t_{j_3}t_{j_4}}\log\hat{\mu}(\mathbf{t}).$$

Remark. The number of distinct values in the $p \times p \times p \times p$ array $\hat{K}^{(4)}(\mathbf{t})$ is

$$l_{K_4} = p + 3 \times \binom{p}{2} + 3 \times \binom{p}{3} + \binom{p}{4}.$$

Using (4.11) and (4.12), any higher order derivatives of the estimator of the cumulant generating function can be obtained. Thus, the 4^{th} order derivatives of the empirical cumulant generating function are given by,

$$\begin{aligned}
&\hat{K}^{(j_1 j_2 j_3 j_4)}(\mathbf{t}) = \frac{\partial}{\partial t_{j_1} t_{j_2} t_{j_3} t_{j_4}} \log\left(\hat{\mu}(t)\right) \\
&= \sum_{i=1}^{n} \tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_3}\tilde{X}_{ij_4}P_i - 6\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_4}P_i\right) \\
&\quad - \left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_3}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_4}P_i\right) - \left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}P_i\right) \\
&\quad - \left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_3}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}P_i\right) - \left(\sum_{i=1}^{n}\tilde{X}_{ij_2}\tilde{X}_{ij_3}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}P_i\right) \\
&\quad + 2\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_2}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_4}P_i\right) + 2\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_3}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_4}P_i\right) \\
&\quad + 2\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}P_i\right) + 2\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}\tilde{X}_{ij_3}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_4}P_i\right) \\
&\quad + 2\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}P_i\right) + 2\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_1}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}P_i\right) \\
&\quad - \left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_2}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_3}\tilde{X}_{ij_4}P_i\right) - \left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_3}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}\tilde{X}_{ij_4}P_i\right) \\
&\quad - \left(\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_4}P_i\right)\left(\sum_{i=1}^{n}\tilde{X}_{ij_2}\tilde{X}_{ij_3}P_i\right).
\end{aligned}$$

It is obvious that $\hat{K}^{(j_1 j_2 j_3 j_4)}(\mathbf{0}_p) = \frac{1}{n}\sum_{i=1}^{n}\tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_3}\tilde{X}_{ij_4}$. To see the connection between $\hat{K}^{(4)}(\mathbf{t})$ and sample Mardia's kurtosis, we first recall the Mardia's kurtosis test statistic as

$$b_{2,p} = \frac{1}{n}\sum_{i=1}^{n}\left[\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)' \mathbf{S}^{-1}\left(\mathbf{X}_i - \bar{\mathbf{X}}\right)\right]^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\tilde{\mathbf{X}}_i'\tilde{\mathbf{X}}_i\right)^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j_1,j_2,j_3,j_4=1}^{p}\tilde{X}_{ij_1}\tilde{X}_{ij_2}\tilde{X}_{ij_3}\tilde{X}_{ij_4}\right].$$

Clearly,

$$\sum_{j_1,j_2,j_3,j_4=1}^{p} \hat{K}^{(j_1j_2j_3j_4)}(\mathbf{0}_p) = b_{2,p}.$$

When $\|\mathbf{t}\|_p \neq 0$, the expression of $\hat{K}^{j_1j_2j_3j_4}(\mathbf{t})$ also contains information on numerous mixed derivatives up to order four around the origin.

As earlier, we will now also work with $\hat{\mu}(\mathbf{t})$ and derive the results for the studentized estimator $\hat{K}(\mathbf{t})$.

Let $\tilde{Z}^{j_1j_2j_3j_4}(\mathbf{t}) = \sqrt{n}\left(\hat{\mu}^{(j_1j_2j_3j_4)}(\mathbf{t}) - \mu^{(j_1j_2j_3j_4)}(\mathbf{t})\right), 0 \leq j_1 \leq j_2 \leq j_3 \leq j_4$, and let $\mathcal{Z}_4^n(\mathbf{t})$ be a random vector consists of all derivatives of $\hat{\mu}(\mathbf{t})$, up to the 4^{th} order. Thus

$$\tilde{\mathcal{Z}}_4^n(\mathbf{t}) = \left[\tilde{Z}^{0000}(\mathbf{t}) \quad \ldots \tilde{Z}^{000p}(\mathbf{t}) \quad \tilde{Z}^{0011}(\mathbf{t}) \quad \tilde{Z}^{0012}(\mathbf{t}) \quad \ldots \quad \tilde{Z}^{pppp}(\mathbf{t})\right]' \tag{4.13}$$

is a vector of length $l_{\tilde{\mathcal{Z}}_4} = l_{\tilde{\mathcal{Z}}_3} + l_{K_4}$. As earlier, by the central limit theorem, the distribution of $\tilde{\mathcal{Z}}_4^n(\mathbf{t})$ approaches a multivariate normal distribution with mean $\mathbf{0}_{l_{\tilde{\mathcal{Z}}_4}}$ and variance-covariance matrix defined by

$$\mathrm{Cov}\left(\tilde{Z}^{j_1j_2j_3j_4}(\mathbf{t}), \tilde{Z}^{j_1'j_2'j_3'j_4'}(\mathbf{t})\right) = \mu^{j_1j_2j_3j_4j_1'j_2'j_3'j_4'}(2\mathbf{t}) - \mu^{j_1j_2j_3j_4}(\mathbf{t})\mu^{j_1'j_2'j_3'j_4'}(\mathbf{t})$$

The cross-covariance between $\tilde{\mathcal{Z}}_4^n(\mathbf{t})$ and $\tilde{\mathcal{Z}}_4^n(\mathbf{s})$ is defined as

$$\mathrm{Cov}\left(\tilde{Z}^{j_1j_2j_3j_4}(\mathbf{t}), \tilde{Z}^{j_1'j_2'j_3'j_4'}(\mathbf{s})\right) = \mu^{j_1j_2j_3j_4j_1'j_2'j_3'j_4'}(\mathbf{t}+\mathbf{s}) - \mu^{j_1j_2j_3j_4}(\mathbf{t})\mu^{j_1'j_2'j_3'j_4'}(\mathbf{s}).$$

Theorem 4.2 (Asymptotic distribution of $\hat{K}^{(4)}(\mathbf{t})$)**.** *Let* $\tilde{T}^{(j_1j_2j_3j_4)(n)}(\mathbf{t}) = \sqrt{n}\hat{K}^{(j_1j_2j_3j_4)}(\mathbf{t})$*, and* $\tilde{\mathcal{T}}_4^n(\mathbf{t})$ *be a vector of length* l_{T_4}*, consists of* $\tilde{T}^{(j_1j_2j_3j_4)}(\mathbf{t})$*,* $1 \leq j_1 \leq j_2 \leq j_3 \leq p$*, that is*

$$\tilde{\mathcal{T}}_4^n(\mathbf{t}) = \left(\tilde{T}^{(j_1j_2j_3j_4)(n)}(\mathbf{t})\right) = \left[\tilde{T}^{(1111)(n)}(\mathbf{t}) \quad \tilde{T}^{(1112)(n)}(\mathbf{t}) \quad \ldots \quad \tilde{T}^{(pppp)(n)}(\mathbf{t})\right]'$$

Then under the hypothesis that sample $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ *is from a multivariate normal distribution* $N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*, the distribution of random vector* $\tilde{\mathcal{T}}_4^n(\mathbf{t})$ *approaches a multivariate normal distribution with mean* $\mathbf{0}$ *and covariance matrix*

$\mathbf{\Sigma}_{\tilde{\mathcal{T}}_4} = [\mathcal{A}_4(\mathbf{t})] \times \mathrm{Cov}\left(\tilde{\mathcal{Z}}_4^n(\mathbf{t}), \tilde{\mathcal{Z}}_4^n(\mathbf{t})\right) [\mathcal{A}_4(\mathbf{t})]'$, *where* $\mathcal{A}_4(\mathbf{t})$ *is a sparse matrix of size* $l_{\mathcal{T}_4} \times l_{\mathcal{Z}_4}$ *with non-zero elements are in part defined by the indexes* (j_1, j_2, j_3, j_4).

Proof. The main idea of the proof is similar to that of Theorem 4.1. We will thus skip the steps and jump directly to the expression of $\mathcal{A}_4^{j_1 j_2 j_3 j_4}(\mathbf{t})$ for each of the several scenarios for the index set $\mathcal{J}_4 = (j_1, j_2, j_3, j_4)$.

- Case 1: $\mathcal{J}_4 = (j_1, j_2, j_3, j_4)$, with $1 \leq j_1 < j_2 < j_3 < j_4 \leq p$. We have

$$
\begin{aligned}
& g_{j_1 j_2 j_3 j_4}(U_n(\mathbf{t})) \\
=& \frac{\hat{\mu}_{\mathbf{t}}^{j_1 j_2 j_3 j_4}}{\hat{\mu}_{\mathbf{t}}} - \frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_2 j_3} \hat{\mu}_{\mathbf{t}}^{000 j_4} + \hat{\mu}_{\mathbf{t}}^{0 j_1 j_2 j_4} \hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{0 j_1 j_3 j_4} \hat{\mu}_{\mathbf{t}}^{000 j_2} + \hat{\mu}_{\mathbf{t}}^{0 j_2 j_3 j_4} \hat{\mu}_{\mathbf{t}}^{000 j_1}}{\hat{\mu}_{\mathbf{t}}^2} \\
& + 2\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2} \hat{\mu}^{000 j_3} \hat{\mu}^{000 j_4} + \hat{\mu}_{\mathbf{t}}^{00 j_1 j_3} \hat{\mu}^{000 j_2} \hat{\mu}^{000 j_4} \hat{\mu}_{\mathbf{t}}^{00 j_1 j_4} \hat{\mu}^{000 j_2} \hat{\mu}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^3} \\
& + 2\frac{\hat{\mu}_{\mathbf{t}}^{00 j_2 j_3} \hat{\mu}^{000 j_1} \hat{\mu}^{000 j_4} + \hat{\mu}_{\mathbf{t}}^{00 j_2 j_4} \hat{\mu}^{000 j_1} \hat{\mu}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{00 j_3 j_4} \hat{\mu}^{000 j_1} \hat{\mu}^{000 j_2}}{\hat{\mu}_{\mathbf{t}}^3} \\
& - \frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2} \hat{\mu}_{\mathbf{t}}^{00 j_3 j_4} + \hat{\mu}_{\mathbf{t}}^{00 j_1 j_3} \hat{\mu}_{\mathbf{t}}^{00 j_2 j_4} + \hat{\mu}_{\mathbf{t}}^{00 j_1 j_4} \hat{\mu}_{\mathbf{t}}^{00 j_2 j_3}}{\hat{\mu}_{\mathbf{t}}^2} \\
& - 6\frac{\hat{\mu}_{\mathbf{t}}^{000 j_1} \hat{\mu}_{\mathbf{t}}^{000 j_2} \hat{\mu}_{\mathbf{t}}^{000 j_3} \hat{\mu}_{\mathbf{t}}^{000 j_4}}{\hat{\mu}_{\mathbf{t}}^4}.
\end{aligned}
$$

Taylor expansion of $g_{j_1j_2j_3j_4}(U(\mathbf{t}))$ around $\xi(\mathbf{t})$ is:

$$
\begin{aligned}
&\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} \\
=&\Bigg(-\frac{\hat{\mu}_{\mathbf{t}}^{j_1j_2j_3j_4}}{\hat{\mu}_{\mathbf{t}}^2} + 2\frac{\hat{\mu}_{\mathbf{t}}^{0j_1j_2j_3}\hat{\mu}_{\mathbf{t}}^{000j_4} + \hat{\mu}_{\mathbf{t}}^{0j_1j_2j_4}\hat{\mu}_{\mathbf{t}}^{000j_3} + \hat{\mu}_{\mathbf{t}}^{0j_1j_3j_4}\hat{\mu}_{\mathbf{t}}^{000j_2} + \hat{\mu}_{\mathbf{t}}^{0j_2j_3j_4}\hat{\mu}_{\mathbf{t}}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^3} \\
&- 6\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_2}\hat{\mu}^{000j_3}\hat{\mu}^{000j_4} + \hat{\mu}_{\mathbf{t}}^{00j_1j_3}\hat{\mu}^{000j_2}\hat{\mu}^{000j_4}\hat{\mu}_{\mathbf{t}}^{00j_1j_4}\hat{\mu}^{000j_2}\hat{\mu}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^4} \\
&- 6\frac{\hat{\mu}_{\mathbf{t}}^{00j_2j_3}\hat{\mu}^{000j_1}\hat{\mu}^{000j_4} + \hat{\mu}_{\mathbf{t}}^{00j_2j_4}\hat{\mu}^{000j_1}\hat{\mu}^{000j_3} + \hat{\mu}_{\mathbf{t}}^{00j_3j_4}\hat{\mu}^{000j_1}\hat{\mu}^{000j_2}}{\hat{\mu}_{\mathbf{t}}^4} \\
&+ 2\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_2}\hat{\mu}_{\mathbf{t}}^{00j_3j_4} + \hat{\mu}_{\mathbf{t}}^{00j_1j_3}\hat{\mu}_{\mathbf{t}}^{00j_2j_4} + \hat{\mu}_{\mathbf{t}}^{00j_1j_4}\hat{\mu}_{\mathbf{t}}^{00j_2j_3}}{\hat{\mu}_{\mathbf{t}}^3} \\
&+ 24\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}_{\mathbf{t}}^{000j_3}\hat{\mu}_{\mathbf{t}}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^5}\Bigg)\Bigg|_{\xi(\mathbf{t})},
\end{aligned}
$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000j_1}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left(\frac{-6\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}_{\mathbf{t}}^{000j_3}\hat{\mu}_{\mathbf{t}}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{4}} - \frac{\hat{\mu}_{\mathbf{t}}^{0j_2j_3j_4}}{\hat{\mu}_{\mathbf{t}}^{2}} + 2\frac{\hat{\mu}^{00j_2j_3}\hat{\mu}^{000j_4} + \hat{\mu}^{00j_2j_4}\hat{\mu}^{000j_3} + \hat{\mu}^{00j_3j_4}\hat{\mu}^{000j_2}}{\hat{\mu}_{\mathbf{t}}^{3}} \right)\Bigg|_{\xi(\mathbf{t})},$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000j_2}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left(\frac{-6\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_3}\hat{\mu}_{\mathbf{t}}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{4}} - \frac{\hat{\mu}_{\mathbf{t}}^{0j_1j_3j_4}}{\hat{\mu}_{\mathbf{t}}^{2}} + 2\frac{\hat{\mu}^{00j_1j_3}\hat{\mu}^{000j_4} + \hat{\mu}^{00j_1j_4}\hat{\mu}^{000j_3} + \hat{\mu}^{00j_3j_4}\hat{\mu}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^{3}} \right)\Bigg|_{\xi(\mathbf{t})},$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000j_3}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left(\frac{-6\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}_{\mathbf{t}}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{4}} - \frac{\hat{\mu}_{\mathbf{t}}^{0j_1j_2j_4}}{\hat{\mu}_{\mathbf{t}}^{2}} + 2\frac{\hat{\mu}^{00j_1j_2}\hat{\mu}^{000j_4} + \hat{\mu}^{00j_1j_4}\hat{\mu}^{000j_2} + \hat{\mu}^{00j_2j_4}\hat{\mu}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^{3}} \right)\Bigg|_{\xi(\mathbf{t})},$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left(\frac{-6\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}_{\mathbf{t}}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^{4}} - \frac{\hat{\mu}_{\mathbf{t}}^{0j_1j_2j_3}}{\hat{\mu}_{\mathbf{t}}^{2}} + 2\frac{\hat{\mu}^{00j_1j_2}\hat{\mu}^{000j_3} + \hat{\mu}^{00j_1j_3}\hat{\mu}^{000j_2} + \hat{\mu}^{00j_2j_3}\hat{\mu}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^{3}} \right)\Bigg|_{\xi(\mathbf{t})},$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_1j_2}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left. 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_3}\hat{\mu}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{3}} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_3j_4}}{\hat{\mu}_{\mathbf{t}}^{2}} \right|_{\xi(\mathbf{t})},$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_1j_3}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left. 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{3}} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_2j_4}}{\hat{\mu}_{\mathbf{t}}^{2}} \right|_{\xi(\mathbf{t})},$$

$$\left.\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_1j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\right|_{\xi(\mathbf{t})} = \left. 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^{3}} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_2j_3}}{\hat{\mu}_{\mathbf{t}}^{2}} \right|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_2j_3}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{3}} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_4}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_2j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{3}} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_2j_4}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_3j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^{3}} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_2j_3}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_2j_3}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = -\frac{\hat{\mu}_{\mathbf{t}}^{000j_4}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_2j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = -\frac{\hat{\mu}_{\mathbf{t}}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_3j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = -\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_2j_3j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = -\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^{2}}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{j_1j_2j_3j_4}} g_{j_1j_2j_3j_4}(\mathbf{t})\Big|_{\xi(\mathbf{t})} = \frac{1}{\hat{\mu}_{\mathbf{t}}}\Big|_{\xi(\mathbf{t})},$$

with

$$\xi(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[1, t_{j_1}, \ldots, t_{j_4}, t_{j_1}t_{j_2}, \ldots t_{j_3}t_{j_4}, t_{j_1}t_{j_2}t_{j_3}, \ldots, t_{j_2}t_{j_3}t_{j_4}, t_{j_1}t_{j_2}t_{j_3}t_{j_4}\right]'$$

the above expansions yield,

$$\left[\mathcal{A}_4^{j_1j_2j_3j_4}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[t_{j_1}t_{j_2}t_{j_3}t_{j_4}, -t_{j_2}t_{j_3}t_{j_4}, -t_{j_1}t_{j_3}t_{j_4}, -t_{j_1}t_{j_2}t_{j_4}, -t_{j_1}t_{j_2}t_{j_3},\right.$$

$$t_{j_3}t_{j_4}, t_{j_2}t_{j_4}, t_{j_2}t_{j_3}, t_{j_1}t_{j_4}, t_{j_1}t_{j_3}, t_{j_1}t_{j_2},$$

$$\left.-t_{j_4}, -t_{j_3}, -t_{j_2}, -t_{j_1}, 1\right]'.$$

- Case 2: $\mathcal{J}_4 = (j_1, j_1, j_2, j_3)$ with $1 \le j_1 < j_2 < j_3 \le p$. We have

$$
\begin{aligned}
& g_{j_1 j_2 j_2 j_3}(U_n(\mathbf{t})) \\
=& \frac{\hat{\mu}_{\mathbf{t}}^{j_1 j_1 j_2 j_3}}{\hat{\mu}_{\mathbf{t}}} - \frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{0 j_1 j_1 j_3}\hat{\mu}_{\mathbf{t}}^{000 j_2} + 2\hat{\mu}_{\mathbf{t}}^{0 j_1 j_2 j_3}\hat{\mu}_{\mathbf{t}}^{000 j_1}}{\hat{\mu}^2} \\
& - 6\frac{\left(\hat{\mu}^{000 j_1}\right)^2 \hat{\mu}_{\mathbf{t}}^{000 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^4} + 2\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}\hat{\mu}^{000 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{00 j_2 j_3}\left(\hat{\mu}_{\mathbf{t}}^{000 j_1}\right)^2}{\hat{\mu}_{\mathbf{t}}^3} \\
& + 4\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}^{000 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{00 j_1 j_3}\hat{\mu}^{000 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_2}}{\hat{\mu}_{\mathbf{t}}^3} - \frac{\hat{\mu}^{00 j_1 j_1}\hat{\mu}_{\mathbf{t}}^{00 j_2 j_3} + 2\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}^{00 j_1 j_3}}{\hat{\mu}_{\mathbf{t}}^2}.
\end{aligned}
$$

$$
\begin{aligned}
& \frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}} g_{j_1 j_2 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} \\
=& \left(-\frac{\hat{\mu}_{\mathbf{t}}^{j_1 j_1 j_2 j_3}}{\hat{\mu}_{\mathbf{t}}^2} + 2\frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{0 j_1 j_1 j_3}\hat{\mu}_{\mathbf{t}}^{000 j_2} + 2\hat{\mu}_{\mathbf{t}}^{0 j_1 j_2 j_3}\hat{\mu}_{\mathbf{t}}^{000 j_1}}{\hat{\mu}^3} \right. \\
& + 24\frac{\left(\hat{\mu}^{000 j_1}\right)^2 \hat{\mu}_{\mathbf{t}}^{000 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^5} - 6\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}\hat{\mu}^{000 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{00 j_2 j_3}\left(\hat{\mu}_{\mathbf{t}}^{000 j_1}\right)^2}{\hat{\mu}_{\mathbf{t}}^4} \\
& \left. - 12\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}^{000 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_3} + \hat{\mu}_{\mathbf{t}}^{00 j_1 j_3}\hat{\mu}^{000 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_2}}{\hat{\mu}_{\mathbf{t}}^4} + 2\frac{\hat{\mu}^{00 j_1 j_1}\hat{\mu}_{\mathbf{t}}^{00 j_2 j_3} + 2\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}^{00 j_1 j_3}}{\hat{\mu}_{\mathbf{t}}^3} \right) \Bigg|_{\xi(\mathbf{t})},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000 j_1}} g_{j_1 j_2 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} =& \left(2\frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_2 j_3}}{\hat{\mu}^2} - 12\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^4} \right. \\
& \left. + 4\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^3} + 4\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^3} \right) \Bigg|_{\xi(\mathbf{t})},
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000 j_2}} g_{j_1 j_2 j_2 j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} =& \left(\frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_1 j_3}}{\hat{\mu}_{\mathbf{t}}^2} - 6\frac{(\hat{\mu}_{\mathbf{t}}^{000 j_1})^2 \hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^4} \right. \\
& \left. 2\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_3}}{\hat{\mu}_{\mathbf{t}}^3} + 4\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_3}\hat{\mu}_{\mathbf{t}}^{000 j_1}}{\hat{\mu}_{\mathbf{t}}} \right) \Bigg|_{\xi(\mathbf{t})},
\end{aligned}
$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000j_3}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \left(\frac{\hat{\mu}_{\mathbf{t}}^{0j_1j_1j_2}}{\hat{\mu}_{\mathbf{t}}^2} - 6\frac{(\hat{\mu}_{\mathbf{t}}^{000j_1})^2\hat{\mu}_{\mathbf{t}}^{000j_2}}{\hat{\mu}_{\mathbf{t}}^4} \right.$$
$$\left. 2\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_1}\hat{\mu}_{\mathbf{t}}^{000j_2}}{\hat{\mu}_{\mathbf{t}}^3} + 4\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_2}\hat{\mu}_{\mathbf{t}}^{000j_1}}{\hat{\mu}_{\mathbf{t}}} \right)\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_1j_1}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = 2\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}\hat{\mu}_{\mathbf{t}}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^3} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_2j_3}}{\hat{\mu}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_1j_2}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = 4\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_3}}{\hat{\mu}_{\mathbf{t}}^3} - 2\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_3}}{\hat{\mu}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_1j_3}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = 4\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_2}}{\hat{\mu}_{\mathbf{t}}^3} - 2\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_2}}{\hat{\mu}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00j_2j_3}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = 2\frac{(\hat{\mu}_{\mathbf{t}}^{000j_1})^2}{\hat{\mu}_{\mathbf{t}}^3} - \frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_1}}{\hat{\mu}_{\mathbf{t}}^2}\Big|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_1j_2}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = -\frac{\hat{\mu}_{\mathbf{t}}^{000j_3}}{\hat{\mu}^2}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_1j_3}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = -\frac{\hat{\mu}_{\mathbf{t}}^{000j_2}}{\hat{\mu}^2}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_2j_3}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = -2\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}}{\hat{\mu}^2}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{j_1j_1j_2j_3}} g_{j_1j_2j_2j_3}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \frac{1}{\hat{\mu}}\Big|_{\xi(\mathbf{t})}.$$

Together with

$$\xi(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right)\Big[1, t_{j_1}, t_{j_2}, t_{j_3}, 1+t_{j_1}^2, t_{j_1}t_{j_2}, t_{j_1}t_{j_3}, t_{j_2}t_{j_3},$$
$$(1+t_{j_1}^2)t_{j_2}, (1+t_{j_1}^2)t_{j_3}, t_{j_1}t_{j_2}t_{j_3}, (1+t_{j_1}^2)t_{j_2}t_{j_3}\Big]'$$

we then obtain that

$$\left[\mathcal{A}_4^{j_1j_1j_2j_3}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\Big[(t_{j_1}^2-1)t_{j_2}t_{j_3}, -2t_{j_1}t_{j_2}t_{j_3}, (1-t_{j_1}^2)t_{j_2}, (1-t_{j_1}^2)t_{j_3},$$
$$t_{j_2}t_{j_3}, 2t_{j_1}t_{j_3}, 2t_{j_1}t_{j_2}, t_{j_1}^2-1, -t_{j_3}, -t_{j_2}, -2t_{j_1}, 1\Big]'.$$

Similar calculations are done for other cases of fourth derivatives obtained by various triplets as follows,

$$\left[\mathcal{A}_4^{j_1j_2j_2j_3}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\Big[t_{j_1}(t_{j_2}^2-1)t_{j_3}, t_{j_1}(1-t_{j_2}^2), -2t_{j_1}t_{j_2}t_{j_3}, (1-t_{j_2}^2)t_{j_3},$$
$$2t_{j_2}t_{j_3}, t_{j_2}^2-1, t_{j_1}t_{j_3}, 2t_{j_1}t_{j_2}, -t_{j_3}, -2t_{j_2}, -t_{j_1}, 1\Big]',$$
$$\left[\mathcal{A}_4^{j_1j_2j_3j_3}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\Big[t_{j_1}t_{j_2}(t_{j_3}^2-1), t_{j_1}(1-t_{j_3}^2), t_{j_2}(1-t_{j_3}^2), -2t_{j_1}t_{j_2}t_{j_3},$$
$$t_{j_3}^2-1, 2t_{j_2}t_{j_3}, 2t_{j_1}t_{j_3}, t_{j_1}t_{j_2}, -2t_{j_3}, -t_{j_2}, -t_{j_1}, 1\Big]'.$$

- Case 3: $\mathcal{J}_4 = (j_1, j_1, j_2, j_2)$ with $1 \le j_1 < j_2 \le p$. We have

$$\begin{aligned} g_{j_1j_1j_2j_2}(U_n(\mathbf{t})) = {} & \frac{\hat{\mu}_{\mathbf{t}}^{j_1j_1j_2j_2}}{\hat{\mu}_{\mathbf{t}}} - 2\frac{\hat{\mu}^{0j_1j_1j_2}\hat{\mu}_{\mathbf{t}}^{000j_2} + \hat{\mu}^{0j_1j_2j_2}\hat{\mu}_{\mathbf{t}}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^2} \\ & + 2\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_1}(\hat{\mu}_{\mathbf{t}}^{000j_2})^2 + 4\hat{\mu}_{\mathbf{t}}^{00j_1j_3}\hat{\mu}_{\mathbf{t}}^{000j_1}\hat{\mu}_{\mathbf{t}}^{000j_2} + \hat{\mu}^{00j_2j_2}(\hat{\mu}_{\mathbf{t}}^{000j_1})^2}{\hat{\mu}_{\mathbf{t}}^3} \\ & - \frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_1}\hat{\mu}_{\mathbf{t}}^{00j_2j_2} + 2(\hat{\mu}_{\mathbf{t}}^{00j_1j_3})^2}{\hat{\mu}_{\mathbf{t}}^2} \\ & - 6\frac{(\hat{\mu}^{000j_1})^2(\hat{\mu}^{000j_2})^2}{\hat{\mu}_{\mathbf{t}}^4}. \end{aligned}$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}} g_{j_1 j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \Bigg(-\frac{\hat{\mu}_{\mathbf{t}}^{j_1 j_1 j_2 j_2}}{\hat{\mu}_{\mathbf{t}}^2} + 4\frac{\hat{\mu}^{0 j_1 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_2} + \hat{\mu}^{0 j_1 j_2 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_1}}{\hat{\mu}_{\mathbf{t}}^3} - 6\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}(\hat{\mu}_{\mathbf{t}}^{000 j_2})^2 + 4\hat{\mu}_{\mathbf{t}}^{00 j_1 j_3}\hat{\mu}_{\mathbf{t}}^{000 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_2} + \hat{\mu}^{00 j_2 j_2}(\hat{\mu}_{\mathbf{t}}^{000 j_1})^2}{\hat{\mu}_{\mathbf{t}}^4} + 2\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}\hat{\mu}_{\mathbf{t}}^{00 j_2 j_2} + 2(\hat{\mu}_{\mathbf{t}}^{00 j_1 j_3})^2}{\hat{\mu}_{\mathbf{t}}^3} + 24\frac{(\hat{\mu}^{000 j_1})^2(\hat{\mu}^{000 j_2})^2}{\hat{\mu}_{\mathbf{t}}^5} \Bigg)\Bigg|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000 j_1}} g_{j_1 j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \Bigg(-2\frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_2 j_2}}{\hat{\mu}_{\mathbf{t}}^2} + 2\frac{4\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_2} + 2\hat{\mu}_{\mathbf{t}}^{00 j_2 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_1}}{\hat{\mu}_{\mathbf{t}}^3} - 12\frac{\hat{\mu}_{\mathbf{t}}^{000 j_1}(\hat{\mu}_{\mathbf{t}}^{000 j_2})^2}{\hat{\mu}_{\mathbf{t}}^4} \Bigg)\Bigg|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{000 j_2}} g_{j_1 j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \Bigg(-2\frac{\hat{\mu}_{\mathbf{t}}^{0 j_1 j_1 j_2}}{\hat{\mu}_{\mathbf{t}}^2} + 2\frac{4\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}\hat{\mu}_{\mathbf{t}}^{000 j_1} + 2\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_2}}{\hat{\mu}_{\mathbf{t}}^3} - 12\frac{\hat{\mu}_{\mathbf{t}}^{000 j_2}(\hat{\mu}_{\mathbf{t}}^{000 j_1})^2}{\hat{\mu}_{\mathbf{t}}^4} \Bigg)\Bigg|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}} g_{j_1 j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \frac{2(\hat{\mu}_{\mathbf{t}}^{000 j_2})^2}{\hat{\mu}_{\mathbf{t}}^3} - \frac{\hat{\mu}_{\mathbf{t}}^{00 j_2 j_2}}{\hat{\mu}_{\mathbf{t}}^2}\Bigg|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}} g_{j_1 j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = 8\frac{\hat{\mu}_{\mathbf{t}}^{000 j_1}\hat{\mu}_{\mathbf{t}}^{000 j_2}}{\hat{\mu}_{\mathbf{t}}^3} - 4\frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_2}}{\hat{\mu}_{\mathbf{t}}^2}\Bigg|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{00 j_2 j_2}} g_{j_1 j_1 j_2 j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \frac{2(\hat{\mu}_{\mathbf{t}}^{000 j_1})^2}{\hat{\mu}_{\mathbf{t}}^3} - \frac{\hat{\mu}_{\mathbf{t}}^{00 j_1 j_1}}{\hat{\mu}_{\mathbf{t}}^2}\Bigg|_{\xi(\mathbf{t})},$$

$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_1j_2}} g_{j_1j_1j_2j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = -2\frac{\hat{\mu}_{\mathbf{t}}^{000j_3}}{den}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{0j_1j_2j_2}} g_{j_1j_1j_2j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = -2\frac{\hat{\mu}_{\mathbf{t}}^{000j_1}}{den}\Big|_{\xi(\mathbf{t})},$$
$$\frac{\partial}{\partial \hat{\mu}_{\mathbf{t}}^{j_1j_1j_2j_2}} g_{j_1j_1j_2j_2}(U_n(\mathbf{t}))\Big|_{\xi(\mathbf{t})} = \frac{1}{\hat{\mu}_{\mathbf{t}}}\Big|_{\xi(\mathbf{t})}.$$

Together with

$$\xi(\mathbf{t}) = \exp\left(\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[1, t_{j_1}\, t_{j_2}, 1+t_{j_1}^2, t_{j_1}t_{j_2}, 1+t_{j_2}^2, (1+t_{j_1}^2)t_{j_2}, t_{j_2}(1+t_{j_2}^2), (1+t_{j_1}^2)(1+t_{j_2}^2)\right]'$$

we obtain

$$\left[\mathcal{A}_4^{j_1j_1j_2j_2}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\Big[t_{j_1}^2t_{j_2}^2 - t_{j_1}^2 - t_{j_2}^2 + 1, 2t_{j_1}(1-t_{j_2}^2),$$
$$(1-t_{j_1}^2)t_{j_2}, t_{j_1}^2 - 1, 4t_{j_1}t_{j_2}, t_{j_1}^2 - 1, -2t_{j_2}, -2t_{j_1}, 1\Big]'.$$

Similar calculations are applied for other fourth derivatives obtained from any two indices. This yields:

$$\left[\mathcal{A}_4^{j_1j_1j_1j_2}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right) = \Big[(t_{j_1}^3 - 3t_{j_1})t_{j_2}, (-3t_{j_1}^2+6)t_{j_2}, -t_{j_1}^3 + 3t_{j_1},$$
$$3t_{j_1}t_{j_2}, 3t_{j_1}^2 - 3, -t_{j_2}, -3t_{j_1}, 1\Big]',$$
$$\left[\mathcal{A}_4^{j_1j_2j_2j_2}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right) = \Big[t_{j_2}(t_{j_1}^3 - 3t_{j_1}), -t_{j_2}^3 + 3t_{j_2}, t_{j_1}(-3t_{j_2}^2+6),$$
$$3t_{j_2}^2 - 3, 3t_{j_1}t_{j_2}, -3t_{j_2}, -t_{j_1}, 1\Big]'.$$

- Case 4: $\mathcal{J}_4 = (j_1, j_1, j_1, j_1)$ and $1 \le j_1 \le p$. We have,

$$g_{j_1j_1j_1j_1}(U_n(\mathbf{t})) = \frac{\hat{\mu}_{\mathbf{t}}^{j_1j_1j_1j_1}}{\hat{\mu}_{\mathbf{t}}} - 6\frac{(\hat{\mu}_{\mathbf{t}}^{000j_1})^4}{\hat{\mu}_{\mathbf{t}}^4} - 4\frac{\hat{\mu}_{\mathbf{t}}^{0j_1j_1j_1}\hat{\mu}_{\mathbf{t}}^{000j_1}}{\hat{\mu}_{\mathbf{t}}^2}$$
$$+ 12\frac{\hat{\mu}_{\mathbf{t}}^{00j_1j_1}(\hat{\mu}_{\mathbf{t}}^{000j_1})^2}{\hat{\mu}_{\mathbf{t}}^3} - 3\frac{(\hat{\mu}_{\mathbf{t}}^{00j_1j_1})^2}{\hat{\mu}_{\mathbf{t}}^2}$$

we immediately obtain

$$\left[\mathcal{A}_4^{j_1 j_1 j_1 j_1}(\mathbf{t})\right]_{\neq 0} = \exp\left(-\frac{\mathbf{t}'\mathbf{t}}{2}\right)\left[t_{j_1}^4 - 6t_{j_1}^2 + 3, -4t_{j_1}^3 + 12t_{j_1}, 6t_{j_1}^2 - 6, -4t_{j_1}, 1\right]'.$$

Following the same steps as in the proof of Theorem 4.1, we have

$$\tilde{\mathcal{T}}_4^n(\mathbf{t}) = \left(\tilde{T}^{(j_1 j_2 j_3 j_4)(n)}\right) \approx \mathcal{A}_{4\left(l_{T_4} \times l_{\mathcal{Z}_4}\right)}(\mathbf{t}) \times \tilde{\mathcal{Z}}_4^n(\mathbf{t})$$

which then converges in probability to a zero mean multivariate normal distribution with a covariance matrix $\mathbf{\Sigma}_{\tilde{\mathcal{T}}_4}(\mathbf{t}) = [\mathcal{A}_4(\mathbf{t})]\, \mathbf{\Sigma}_{\tilde{\mathcal{Z}}_4^n}(\mathbf{t})\, [\mathcal{A}_4(\mathbf{t})]'$. □

Again, as in the case of using the third derivatives of the cumulant generating function, any linear transformation of vector $\tilde{\mathcal{T}}_4^n(\mathbf{t})$ approaches a univariate Gaussian process. This is formally stated in the following lemma.

Lemma 4.5. For a fixed vector $u \in \mathbb{R}^{l_{K_4}}$, the statistic $L_4^u(\mathbf{t}) = u'\left[\tilde{\mathcal{T}}_4^n(\mathbf{t})\right]$ approaches a normally distributed random vector $\mathcal{L}_4^u(\mathbf{t}) \sim N(0, u'\mathbf{\Sigma}_{\tilde{\mathcal{T}}_4} u)$. Moreover,

$$\begin{aligned}
\mathrm{Cov}(\mathcal{L}_4^u(\mathbf{t}), \mathcal{L}_4^u(\mathbf{s})) &= u'\,\mathrm{Cov}\left(\left[\tilde{\mathcal{T}}_4^n(\mathbf{t})\right], \left[\tilde{\mathcal{T}}_4^n(\mathbf{s})\right]\right)u \\
&= u'\,[\mathcal{A}_4(\mathbf{t})]\,\mathrm{Cov}\left(\tilde{\mathcal{Z}}_4^n(\mathbf{t}), \tilde{\mathcal{Z}}_4^n(\mathbf{s})\right)[\mathcal{A}_4(\mathbf{s})]'\,u \\
&= [\mathcal{A}_4(\mathbf{t})'u]'\,\mathrm{Cov}\left(\tilde{\mathcal{Z}}_4^n(\mathbf{t}), \tilde{\mathcal{Z}}_4^n(\mathbf{s})\right)[\mathcal{A}_4(\mathbf{s})'u].
\end{aligned}$$

As earlier, the choice of vector $\mathbf{t} = \frac{1}{\sqrt{p}} t^* \mathbf{1}_p$ implies that $L_4^u(t^*)$ approaches a univariate Gaussian process $\mathcal{L}_4^u(t^*)$, with mean 0 and covariance function defined as

$$\kappa_4^u(t^*, s^*) = \mathrm{Cov}\left(\mathcal{L}_4^u(t^*), \mathcal{L}_4^u(s^*)\right).$$

With the above machinery in place, we can now, for any suitable choice of vector $u \in \mathbb{R}^{l_{K_4}}$, derive a test based on the fourth derivatives of the cumulant generating function. However, we will focus on the choice $u = \mathbf{a}_4 = \frac{1}{\sqrt{l_{K_4}}}\mathbf{1}_{l_{K_4}}$ and hence $L_4^{\mathbf{a}_4}(.)$ is the average of all distinct estimators of the fourth derivatives. As earlier, we will restrict the choice of the

vector $\mathbf{t}$ to $\mathbf{t} = \frac{1}{\sqrt{p}} t^* \mathbf{1}_p, t^* \in [-1, 1]$. The $1-\alpha$ probability band for $L_4(\mathbf{t}) = L_4(t^*)$ is, thus

$$\pm \left(\hat{m}_{p4}^{\mathbf{a}_4} + u_\alpha \right) \sqrt{\text{Var}(\mathcal{L}_4^{\mathbf{a}_4}(t^*))},$$

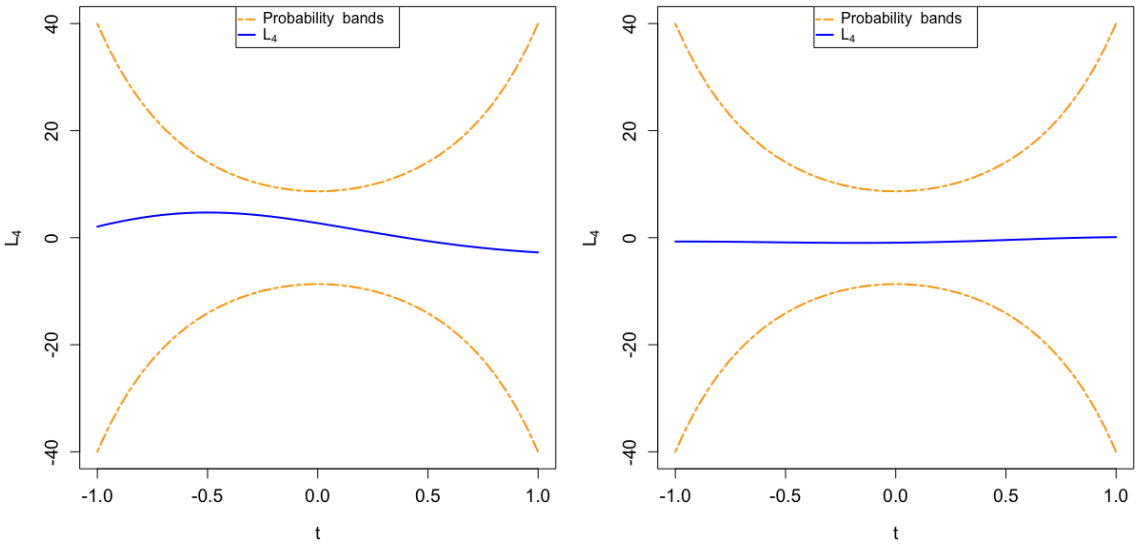
where $u_\alpha = \sqrt{-2\log(\alpha)}$ and $\hat{m}_{p4}^{\mathbf{a}_4}$ is an estimator of $m_{p4}^{\mathbf{a}_4} = \mathbb{E}\left(\sup_{t^* \in [-1,1]} \left| \tilde{\mathcal{L}}_4(t^*) \right| \right)$, which can be easily obtained by simulating the Gaussian process $\tilde{\mathcal{L}}_4^{\mathbf{a}_4}(t^*)$, with 0 mean, and covariance function $\text{Cov}\left(\tilde{\mathcal{L}}_4(t^*), \tilde{\mathcal{L}}_4(s^*) \right) = \frac{\kappa_4^{\mathbf{a}_4}(t^*, s^*))}{\sqrt{\kappa_4^{\mathbf{a}_4}(t^*, t^*) \times \kappa_4^{\mathbf{a}_4}(s^*, s^*)}}$.

4.3.1 Illustrative Examples: Simulated Data

We construct MT_4 plots to assess the multivariate normality assumptions of the data samples generated from the five-dimensional distributions used earlier in Section 3.3. Vector $\mathbf{a}_4$ now has a length of 75, which is almost double the length of vector $\mathbf{a}_3$ in Section 4.2. With $m = 4000$ simulations, and the choice of $t^* = \pm 1, \pm .95, \ldots 0$, we obtain $\hat{m}_{p4}^{\mathbf{a}_4} = 1.4534$. We will, as earlier, take $1 - \alpha = 0.95$.

The two plots in Figure 4.9 correspond to normally distributed data. The standardization does not have any effect, and in both figures, curves are pretty flat and fall in the center of the two bands.

Figure 4.10 gives illustrations of non-normal cases. The first three distributions are asymmetric, and we recall that MT_3 performed very well in detecting the departure from normality. When MT_4 plot is used, and when data are generated using a specific copula (Example (iv)), the graph not only has a nonzero gradient but also crosses the bands (see Figure 4.10a). Figure 4.10b shows a highly nonlinear curve so far away from the center that the vertical axis had to be hugely rescaled. For the case of mixture normal distribution, MT_3 for Figure 4.6c had shown a markedly vivid departure from the multivariate normality due to the lack of symmetry. However, with a bimodal distribution, the interpretation of kurtosis is somewhat ambiguous, yet tails are similar to those of

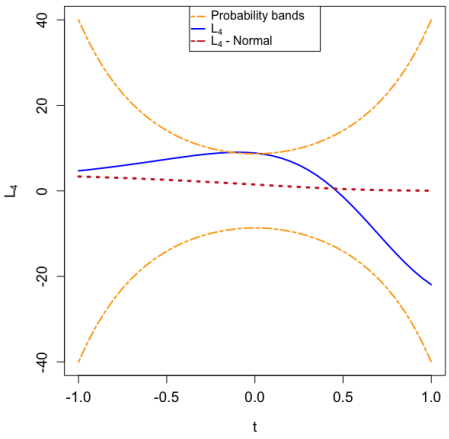


(a) $N_5(\mathbf{0}_5, \mathbf{I}_5)$ (b) $N_5\left(\mu = [1,2,3,4,5]', \mathbf{\Sigma} = 2\left(.7\mathbf{I}_5 + .3\mathbf{J}_5\right)\right)$

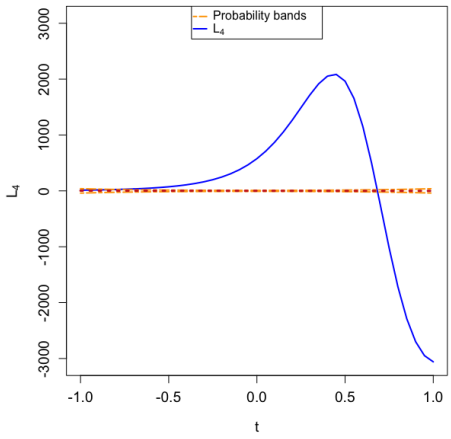
Figure 4.9: MT_4 plots for normally distributed data.

normal distributions. This is why the departure of the curve from the horizontal line is not as dramatic in Figure 4.10c.

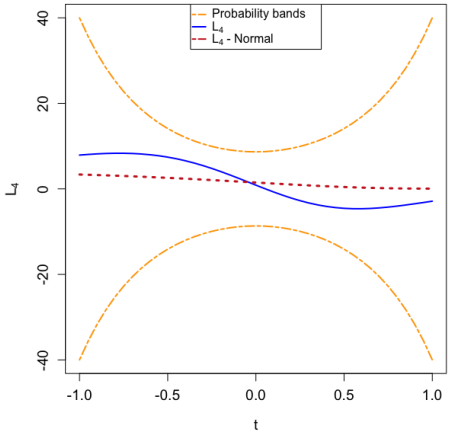
The last three distributions considered were symmetric. However, they all have thicker extremes than the multivariate normal, and these facts vividly show up in Figures 4.10d, 4.10e and 4.10f with highly nonlinear curves each crossing the bands. Recall that for the $t(15)$ distribution, the MT_3 plot in Figure 4.6f was not able to be effectively distinguish the distribution of $L_3(t^*)$ of $t(15)$ and $L_3(t^*)$ of the standard normal distributions. The MT_4 plot, on the other hand, shows large values of $L_4(t^*)$ crossing the bands. For $t(5)$, the evidence is indisputable, where even the vertical axis had to be relatively rescaled.



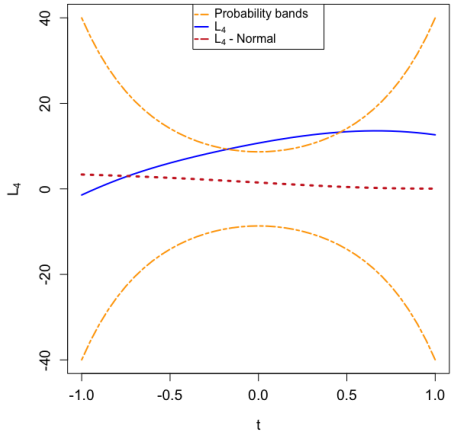
(a) Data generated from copulas



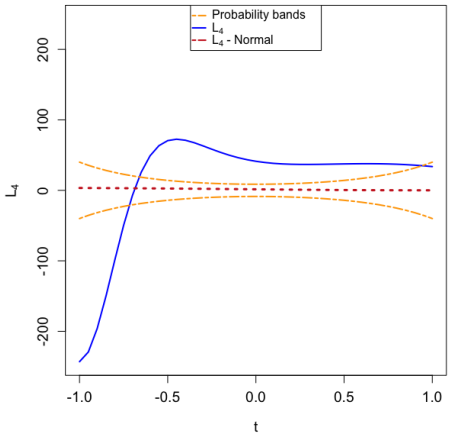
(b) Lomax ($\alpha = 1, \beta = \mathbf{1}_5$)



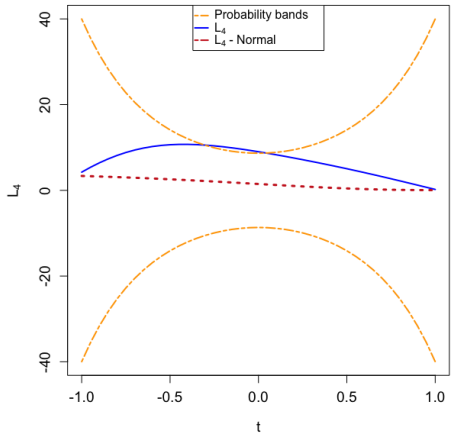
(c) Mixture normal



(d) Kotz($\mu = \mathbf{0}_5, \boldsymbol{\Sigma} = \mathbf{I}_5$)



(e) $t_5(\boldsymbol{\Sigma} = \mathbf{I}_5)$



(f) $t_{15}(\boldsymbol{\Sigma} = \mathbf{I}_5)$

Figure 4.10: MT_4 plots for non-normal data.

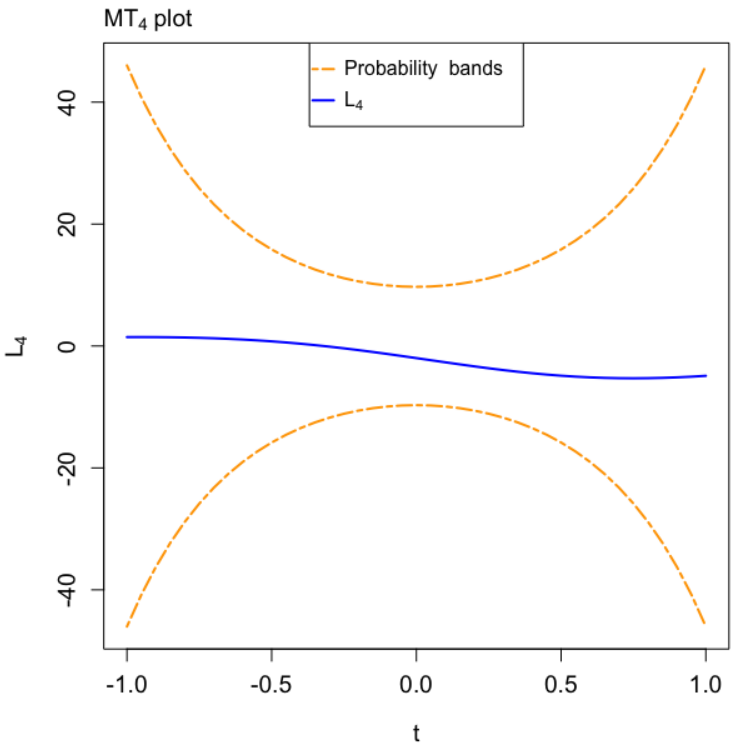


Figure 4.11: MT_4 plot for Cork dataset.

4.3.2 Two Examples: Cork Data and Water Potability Data revisited

Example 4.3.1 (Cork Data)**.** In Figure 4.11, the functional curve $L_4(.)$ is close to the horizontal line $(y = 0)$ and falls within the probability bands. Thus as in MT_3 plot, MT_4 plot also supports the assumption of jointly normally distributed.

Example 4.3.2 (Water Potability Data)**.** The MT_4 plot for the three marginally and normally distributed variables *Organic carbon, Turbidity* and *Trihalomethanes* is given in Figure 4.12. The non-zero slope at the left tail shows some departure from normality for this dataset. Note that the MT_3 plot also had the same conclusion about the non-zero gradient curve as an indication of non-normality.

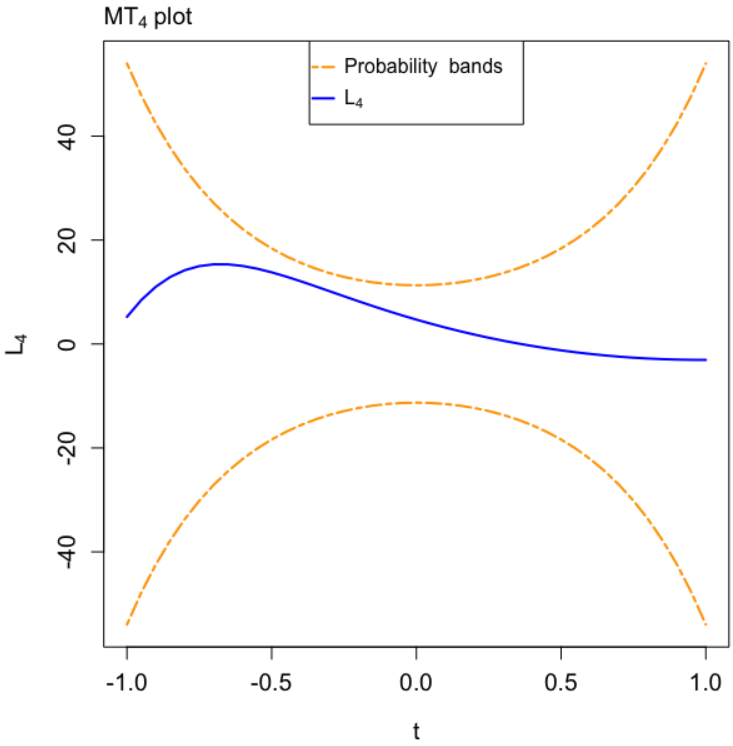


Figure 4.12: MT_4 plot for Water Potability (= Good) dataset; variables considered are *Organic carbon, Turbidity, Trihalomethanes*.

4.4 Concluding Remarks

We can always obtain the exact value of Mardia's skewness and Mardia's kurtosis test statistics by choosing appropriate choices of directional vector $\mathbf{a}_3$ and $\mathbf{a}_4$. More specifically, Mardia's test statistics are indeed weighted sums of third and fourth derivatives of cumulant generating function, calculated as $\mathbf{0}$.

We also emphasize that any choices of vector $\mathbf{t}$ other than $\mathbf{t} = t^*\mathbf{1}_p$, as long as $\mathbf{t}$ is a continuous function of $t^* \in [-1, 1]$, would result in 2D graphs similar to MT_3 and MT_4 introduced above. Moreover,

$$\text{Cov}\left(\mathbf{v}_k'\tilde{\mathcal{T}}_k(\mathbf{t}), \mathbf{u}_k'\tilde{\mathcal{T}}_k(\mathbf{s})\right) = \mathbf{v}_k' \,\text{Cov}\left(\tilde{\mathcal{T}}_k(\mathbf{t}), \tilde{\mathcal{T}}_k(\mathbf{s})\right)\mathbf{u}_k$$

Hence, simultaneous tests on two or more linear combinations can be derived.

Additionally, one wants to obtain the test that can assess both skewness and kurtosis at the same time, this has been done using the method of finding the best linear combination for raw data (see Section 3.6). For our current method, it is quite possible to obtain such tests by noticing that $[L_3(\mathbf{t}), L_4(\mathbf{t})]'$ approaches a bivariate normal distribution with

$$\mathrm{Cov}\left(L_3(\mathbf{t}), L_4(\mathbf{s})\right) = \mathrm{Cov}\left(\mathbf{a}_3'\tilde{\mathcal{T}}_3(\mathbf{t}), \mathbf{a}_4'\tilde{\mathcal{T}}_4(\mathbf{s})\right) = \mathbf{a}_3'\,\mathrm{Cov}\left(\tilde{\mathcal{T}}_3(\mathbf{t}), \tilde{\mathcal{T}}_4(\mathbf{t})\right)\mathbf{a}_4.$$

This underscores the importance of using multivariate normal distributions, where linear combinations also follow a normal distribution. Other probability distribution lacks this property. For other distribution, we may rely on copulas, and the related problem of choosing an appropriate copula is discussed later in Chapter 6.

CHAPTER FIVE

GROUP DISCRIMINATION AND GROUP CLASSIFICATION

5.1 Introduction

In this chapter, we discuss the problem of simultaneously classifying an entire group of data consisting of n p-variate observations into one of the two given populations Π_1 or Π_2. The entire group is classified into either Π_1 or Π_2. These populations may be represented by their respective random samples. This group classification, as opposed to classifying individual observations one at a time, is an essential problem in its own right. At first and intuitively, the most sensible approach, in this case, may rely on the "majority rule", where each individual in this group is first classified into one of the two populations using some suitable approaches, and then the entire group will be classified into one of two populations, in which most of the observations from the group are classified. Here, we will explore various approaches and compare their performances. The generalization to more than two populations is straightforward.

In Section 5.2, we will introduce various "point-wise" classification approaches to which then the "majority rule" criteria will be applied later in Section 5.3. In Section 5.4, we will explore an approach that relies on the features inherent in the entire group rather than features of individual observation one at a time. Specifically, our investigation will pertain to evaluating the similarity of the eigenstructures of the "to be classified" group and the two populations or their random samples. We will then compare the performances of the "majority rule" and the eigenstructure methods by an example of the classification problem of the Water Potability data set in Section 5.5. Finally, we consider the problem of feature selection, finding the best subset of features to improve the performance of classification and discrimination analysis in Section 5.6.

To fix the ideas, suppose $\mathcal{U}_{r\times p}$ is the data matrix where each row is an observation with p features. The task requires the classification of the entire sample $\mathcal{U}$ into either Π_1 or Π_2. Assume further that Π_1 and Π_2 have the respective population means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, and population covariance matrices $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$. Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be respective representative samples of sizes n_1 and n_2 from Π_1 and Π_2.

5.2 Classification of An Observation by Majority Rule

Linear discriminant function was first introduced by R.A Fisher [66], [67], where he considered the discrimination of two populations under the assumption that they share the same covariance matrix $\boldsymbol{\Sigma}$. The main idea of his approach is to choose a direction vector $\mathbf{a}$ so that the distance between $\mathbf{a}'\boldsymbol{\mu}_1$ and $\mathbf{a}'\boldsymbol{\mu}_2$ is largest. Such $\mathbf{a}$ choice is given by [2],

$$\mathbf{a} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2),$$

and linear discriminant function is then,

$$\mathcal{D}(\mathbf{x}) = \mathbf{a}'\mathbf{x} - \frac{\mathbf{a}'(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)}{2}.$$

We classify $\mathbf{x}$ to Π_1 if $\mathcal{D}(\mathbf{x}) \geq 0$ and to Π_2 otherwise. Geometrically, the $\mathcal{D}(.)$ is the optimal hyperplane that maximizes the separation between two populations by using the weighted sum of feature variables, and classification rule is indeed decided by comparison of distances of $\mathbf{x}$ to the two populations on the $\mathbf{a}'\mathbf{x}$ scale. Also note that $\frac{\mathbf{a}'(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)}{2}$ is a constant, and the magnitude of the distance is in fact dependent only on the weights of features determined by the direction vector $\mathbf{a}$. In fact,

$$\begin{aligned} 2\mathcal{D}(\mathbf{x}) &= 2\left[\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right]'\mathbf{x} - (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2) \\ &= -(\mathbf{x} - \boldsymbol{\mu}_1)'\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) + (\mathbf{x} - \boldsymbol{\mu}_2)'\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_2), \end{aligned}$$

which essentially implements a comparison of the corresponding Mahalanobis distances of the point $\mathbf{x}$ to the means of the two populations Π_1 and Π_2. Hence, Classification Rule 1 given below compares the two Mahalanobis distances to the respective populations.

Classification Rule 1 (Population version)**.** When the population means $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ and the common population covariance matrix $\boldsymbol{\Sigma}$ are available, we classify $\mathbf{x}$ into Π_1 if $\mathcal{D}(\mathbf{x}) \geq 0$ and to Π_2 otherwise.

In practice, population parameters might be unknown, but samples from these two populations are available, and hence, a sample version of the linear discriminant function $\hat{\mathcal{D}}(.)$ is used. Let $\mathcal{X}_{1(n_1\times p)}$ and $\mathcal{X}_{2(n_2\times p)}$ be matrices containing random samples of size n_1 and n_2 from Π_1 and Π_2 respectively. We then have an estimate of optimal directional vector $\mathbf{a}$ as $\hat{\mathbf{a}} = \mathbf{S}_{pool}^{-1}\left(\bar{\mathbf{X}}_1 - \bar{\mathbf{X}}_2\right)$, where $\bar{\mathbf{X}}_1$ and $\bar{\mathbf{X}}_2$ are the respectively sample means and $\mathbf{S}_{pool} = \dfrac{(n_1-1)\mathbf{S}_1 + (n_2-1)\mathbf{S}_2}{n_1+n_2-2}$ is the pooled covariance matrix obtained from the two sample covariance matrices $\mathbf{S}_1$ and $\mathbf{S}_2$. An estimator of $\mathcal{D}(.)$ is then

$$\hat{\mathcal{D}}(\mathbf{x}) = \hat{\mathbf{a}}'\mathbf{x} - \frac{\hat{\mathbf{a}}'(\bar{\mathbf{X}}_1 + \bar{\mathbf{X}}_2)}{2}.$$

Accordingly, we derive a classification rule based on the sample estimates as follows.

Classification Rule 1.A (Sample version of Classification Rule 1)**.** Relying on $\hat{\mathcal{D}}(.)$, the membership of $\mathbf{x}$ is Π_1 if $\hat{\mathcal{D}}(\mathbf{x}) \geq 0$.

For a new observation $\mathbf{U} \in \mathbb{R}^p$, $\mathcal{D}(\mathbf{U})$ and $\hat{\mathcal{D}}(\mathbf{U})$ focus on the comparison of the distances from $\mathbf{U}$ to the sample means of $\mathcal{X}_1$ and $\mathcal{X}_2$. To be more specific, $\mathcal{D}(.)$ and $\hat{\mathcal{D}}(.)$ project each sample $\mathcal{X}_{1(n_1\times p)}$ and $\mathcal{X}_{2(n_2\times p)}$, onto a real line and set 0 to be the boundary of the classification. If one or both populations were skewed, then the distance from the means may not be very representative for all pairwise distances between $\mathbf{U}$ and points in these populations. We may thus introduce a new rule of classification as below, which takes into account the cluster effect of data points.

Classification Rule 2 (Population version)**.** Suppose $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ are the probability densities of the two populations. Let

$$d_k = \int_{\mathbb{R}^p} \left[\mathcal{D}(\mathbf{x}) - \mathcal{D}(\mathbf{U})\right]^2 f_k(\mathbf{x}) d\mathbf{x}, k = 1, 2.$$

Then

$$\begin{aligned} d_k &= \int_{\mathbb{R}^p} \left(\mathbf{a}'\mathbf{x} - \mathbf{a}'\mathbf{U}\right)^2 f_k(\mathbf{x}) d\mathbf{x} \\ &= \int_{\mathbb{R}^p} \mathbf{a}'(\mathbf{x} - \mathbf{U})(\mathbf{x} - \mathbf{U})'\mathbf{a} f_k(\mathbf{x}) d\mathbf{x} \\ &= \mathbf{a}' \left[\int_{\mathbb{R}^p} (\mathbf{x} - \mathbf{U})(\mathbf{x} - \mathbf{U})' f_k(\mathbf{x}) d\mathbf{x}\right] \mathbf{a} \\ &= \mathbf{a}' \mathbb{E}_{\Pi_k} \left[(\mathbf{x} - \mathbf{U})(\mathbf{x} - \mathbf{U})'\right] \mathbf{a} \\ &= \mathbf{a}' \left[\boldsymbol{\Sigma}_k + (\mathbf{U} - \boldsymbol{\mu}_k)(\mathbf{U} - \boldsymbol{\mu}_k)'\right] \mathbf{a}, \end{aligned}$$

where in the last equality, we have used the fact,

$$\begin{aligned} \mathbb{E}_{\Pi_k} \left[(\mathbf{x} - \mathbf{U})(\mathbf{x} - \mathbf{U})'\right] &= \mathbb{E}_{\Pi_k} \left[(\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)'\right] \\ &\quad - 2\left[\mathbb{E}_{\Pi_k} \left(\mathbf{x} - \boldsymbol{\mu}_k\right)\right]' (\mathbf{x} - \mathbf{U}) + (\mathbf{U} - \boldsymbol{\mu}_k)(\mathbf{U} - \boldsymbol{\mu}_k)' \\ &= \boldsymbol{\Sigma}_k + (\mathbf{U} - \boldsymbol{\mu}_k)(\mathbf{U} - \boldsymbol{\mu}_k)'. \end{aligned}$$

We classify $\mathbf{U}$ into Π_1 if $d_1 \leq d_2$, or if

$$\mathbf{a}'\left[\boldsymbol{\Sigma}_k + (\mathbf{U} - \boldsymbol{\mu}_1)(\mathbf{U} - \boldsymbol{\mu}_1)'\right]\mathbf{a} \leq \mathbf{a}'\left[\boldsymbol{\Sigma}_k + (\mathbf{U} - \boldsymbol{\mu}_2)(\mathbf{U} - \boldsymbol{\mu}_2)'\right]\mathbf{a} \tag{5.1}$$

If $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$, then the above reduces to: We classify $\mathbf{U}$ into Π_1 if

$$\mathbf{a}'\left[(\mathbf{U} - \boldsymbol{\mu}_1)(\mathbf{U} - \boldsymbol{\mu}_1)' - (\mathbf{U} - \boldsymbol{\mu}_2)(\mathbf{U} - \boldsymbol{\mu}_2)'\right]\mathbf{a} \leq 0. \tag{5.2}$$

Thus, when $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$, then as opposed to comparing the distance of $\mathbf{U}$ from the averages in the former case, here, we compare the average distance of $\mathbf{U}$ from the two populations. If the parameters are being estimated, then we may use the estimators $\hat{d}_k$ of

$d_k, k = 1, 2$. Let $\mathbf{X}_{11}, \mathbf{X}_{12}, \ldots, \mathbf{X}_{1n_1}$ and $\mathbf{X}_{21}, \mathbf{X}_{22}, \ldots, \mathbf{X}_{2n_2}$ be samples of size n_1 and n_2 of Π_1 and Π_2. We introduce a sample version of Classification Rule 2 in (5.1) as below.

Classification Rule 2.A (Sample version of Classification Rule 2)**.** Let

$$\hat{d}_k = \frac{1}{n_k} \sum_{j=1}^{n_k} \left[\hat{\mathcal{D}}(\mathbf{X}_{kj}) - \hat{\mathcal{D}}(\mathbf{U}) \right]^2, k = 1, 2$$

then

$$\begin{aligned} \hat{d}_k &= \hat{\mathbf{a}}' \frac{1}{n_k} \sum_{j=1}^{n_k} (\mathbf{X}_{kj} - \mathbf{U})(\mathbf{X}_{kj} - \mathbf{U})' \hat{\mathbf{a}} \\ &= \frac{1}{n_k} \hat{\mathbf{a}}' \left[(n_k - 1)\mathbf{S}_k + \left(\bar{\mathbf{X}}_k - \mathbf{U} \right) \left(\bar{\mathbf{X}}_k - \mathbf{U} \right)' \right] \hat{\mathbf{a}}, k = 1, 2. \end{aligned}$$

We classify $\mathbf{U}$ into Π_1 if $\hat{d}_1 \leq \hat{d}_2$, that is, if

$$\hat{\mathbf{a}}' \left[\frac{1}{n_1} \left(\bar{\mathbf{X}}_1 - \mathbf{U} \right) \left(\bar{\mathbf{X}}_1 - \mathbf{U} \right)' - \frac{1}{n_2} \left(\bar{\mathbf{X}}_2 - \mathbf{U} \right) \left(\bar{\mathbf{X}}_2 - \mathbf{U} \right)' \right] \hat{\mathbf{a}} \leq \frac{n_2 - 1}{n_2} \hat{\mathbf{a}}' \mathbf{S}_2 \hat{\mathbf{a}} - \frac{n_1 - 1}{n_1} \hat{\mathbf{a}}' \mathbf{S}_1 \hat{\mathbf{a}},$$

where $\hat{\mathbf{a}}$ has been defined earlier. If we assume $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2$, then on the right side, we may want to use $\mathbf{S}_{pool}$ instead of $\mathbf{S}_1$ and $\mathbf{S}_2$, as the common estimators of $\mathbf{\Sigma} (= \mathbf{\Sigma}_1 = \mathbf{\Sigma}_2)$. Thus, the right hand side is replaced by $\left(\frac{1}{n_1} - \frac{1}{n_2} \right) \hat{\mathbf{a}}' \mathbf{S}_{pool}^{-1} \hat{\mathbf{a}}$.

We have discussed the classification rules relying on linear discriminant analysis, where the centers of each population are directly and indirectly involved. Instead of comparing Mahalanobis distance to each of the two populations, we now introduce another method of classification, where the distance of an observation with unknown membership from a population is measured based on the closeness of that observation to each of the members in the samples from the populations. Mathematically, instead of comparing distances on the $\mathbf{a}'\mathbf{X}$ or $\hat{\mathbf{a}}'\mathbf{X}$ scales, we compare the average Mahalanobis distances.

Classification Rule 3 (Classification by average distance)**.** Let the covariance matrices for the two populations be $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$. Let $\mathbf{S}_1$ and $\mathbf{S}_2$ be the sample covariance matrices

obtained from $\mathcal{X}_1$ and $\mathcal{X}_2$. Denote

$$\bar{d_k} = \frac{1}{n_k}\sum_{j=1}^{n_k}(\mathbf{X}_{kj} - \mathbf{U})' S_k^{-1}(\mathbf{X}_{kj} - \mathbf{U}), k = 1, 2.$$

Then we classify $\mathbf{U}$ to Π_1 if $\bar{d_1} \leq \bar{d_2}$.

The "leverage score" have been widely used in the regression context as a measure of how far away one independent observation is from the other observations. Borrowing this idea, we also investigate the leverage score as a measure of distance of an observation to a given group of observations from a population.

Classification Rule 4 (Classification by leverage)**.** Leverage of data point $\mathbf{U}$ to sample $\mathcal{X}_k$ is defined as

$$h_k = \mathbf{U}' \left(\frac{1}{n_k - 1}\mathcal{X}_k'\mathcal{X}_k\right)^{-1} \mathbf{U}, k = 1, 2.$$

We classify $\mathbf{U}$ into Π_1 if $h_1 \leq h_2$.

Remark (A note on the use of leverage). One should be cautious about the use of leverage as the rule may fail in certain situations. We illustrate such a scenario here. For data $\mathcal{X}_{(n\times p)}$ with sample mean $\bar{\mathbf{X}}$ and sample covariance matrix $\mathbf{S}$,

$$(n-1)\mathbf{S} = \mathcal{X}'\mathcal{X} - n\bar{\mathbf{X}}\bar{\mathbf{X}}',$$
$$\text{or } \mathcal{X}'\mathcal{X} = (n-1)\mathbf{S} + n\bar{\mathbf{X}}\bar{\mathbf{X}}'.$$

Thus, using Sherman-Morrison Lemma,

$$(\mathcal{X}'\mathcal{X})^{-1} = \frac{1}{n-1}\mathbf{S}^{-1} - \frac{n}{(n-1)^2}\frac{\mathbf{S}^{-1}\bar{\mathbf{X}}\bar{\mathbf{X}}'\mathbf{S}^{-1}}{1 + \frac{n}{n-1}\bar{\mathbf{X}}'\mathbf{S}^{-1}\bar{\mathbf{X}}}$$
$$\text{or } \left(\frac{1}{n-1}\mathcal{X}\mathcal{X}\right)^{-1} = \mathbf{S}^{-1} - \frac{n}{n-1}\frac{\mathbf{S}^{-1}\bar{\mathbf{X}}\bar{\mathbf{X}}'\mathbf{S}^{-1}}{1 + \frac{n}{n-1}\bar{\mathbf{X}}'\mathbf{S}^{-1}\bar{\mathbf{X}}}.$$

Hence, the leverage of a new observation $\mathbf{U}$ to the sample $\mathcal{X}$ can be rewritten as

$$h = \mathbf{U}'\left(\frac{1}{n-1}\mathcal{X}\mathcal{X}\right)^{-1}\mathbf{U} = \mathbf{U}'\mathbf{S}^{-1}\mathbf{U} - \frac{n}{(n-1)}\frac{\mathbf{U}'\mathbf{S}^{-1}\bar{\mathbf{X}}\bar{\mathbf{X}}'\mathbf{S}^{-1}\mathbf{U}}{1+\frac{n}{n-1}\bar{\mathbf{X}}'\mathbf{S}^{-1}\bar{\mathbf{X}}}.$$

As the sample size n increases, $\mathbf{S}$ converges to the population covariance matrix $\boldsymbol{\Sigma}$, $\bar{\mathbf{X}}$ converges to population mean $\boldsymbol{\mu}$, and the coefficient $\frac{n}{n-1} \to 1$. Hence, the leverage can be expressed asymptotically as follows

$$h \approx \mathbf{U}'\boldsymbol{\Sigma}^{-1}\mathbf{U} - \frac{\mathbf{U}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\mathbf{U}}{1+\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}} \approx \tilde{h} \text{ (say)}\ .$$

Suppose $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = \boldsymbol{\Sigma}$. Further assume that $\boldsymbol{\mu}_1 = a \times 1_p$ and $\boldsymbol{\mu}_2 = b \times 1_p$ and $a^2 < b^2$. Hence, comparison of h_1^* and h_2^* (as defined by h using obvious changes) is indeed a comparison of the magnitude of the fraction in the second term. Specifically,

$$\begin{aligned} h_1^* &= \frac{1}{n-1}\mathbf{U}'\boldsymbol{\Sigma}^{-1}\mathbf{U} - \frac{\mathbf{U}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1\boldsymbol{\mu}_1'\boldsymbol{\Sigma}^{-1}\mathbf{U}}{1+\boldsymbol{\mu}_1'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1} \\ &= \frac{1}{n-1}\mathbf{U}'\boldsymbol{\Sigma}^{-1}\mathbf{U} - \frac{a^2\mathbf{U}'\boldsymbol{\Sigma}^{-1}\mathbf{J}\boldsymbol{\Sigma}^{-1}\mathbf{U}}{1+a^2\mathbf{1}_p'\boldsymbol{\Sigma}^{-1}\mathbf{1}_p} \\ &\geq \frac{1}{n-1}\mathbf{U}'\boldsymbol{\Sigma}^{-1}\mathbf{U} - \frac{b^2\mathbf{U}'\boldsymbol{\Sigma}^{-1}\mathbf{J}_p\boldsymbol{\Sigma}^{-1}\mathbf{U}}{1+b^2\mathbf{1}_p'\boldsymbol{\Sigma}^{-1}\mathbf{1}_p} = h_2^*, \end{aligned}$$

and thus the classifier using leverage will always classify a new observation to the second population Π_2, even if it truly comes from Π_1. Therefore, the use of leverage as a classification rule may not be appropriate if the two covariance matrices are equal.

5.3 Group Classification

We now consider the problem of group classification. Suppose as earlier we have a group of r observations, with p features, represented by data matrix $\mathcal{U}_{r\times p} = \begin{bmatrix}\mathbf{U}_1' & \mathbf{U}_2' & \dots & \mathbf{U}_r'\end{bmatrix}'$. Then, any of the above criteria can be implemented along with the "majority rule" approach. Specifically,

Group Classification Rule I (Majority rule)**.** Let $\mathfrak{D}$ be any of the classification rules defined in the previous section for individual observations. Let $\mathcal{U}_{r\times p}$ be the samples that

need to be classified into either Π_1 or Π_2. Then classify $\mathcal{U}$ into Π_1 if

$$\sum_{i=1}^{r} \mathbb{1}_{\left\{\mathbf{U}_i \xrightarrow{\mathfrak{D}} \Pi_1\right\}} \geq \frac{r}{2},$$

where $\mathbb{1}_{\left\{\mathbf{U} \xrightarrow{\mathfrak{D}} \Pi_k\right\}}$ is the indicator function for the event that observation $\mathbf{U}$ is classified into population Π_k using point-wise classification rule $\mathfrak{D}$.

In the M-population case, $M > 2$, the rule will be: Classify $\mathcal{U}$ into k^{*th} population if

$$\sum_{i=1}^{r} \mathbb{1}_{\left\{\mathbf{U}_i \xrightarrow{\mathfrak{D}} \Pi_{k^*}\right\}} = \max_{k=1,2...,M} \sum_{i=1}^{r} \mathbb{1}_{\left\{\mathbf{U}_i \xrightarrow{\mathfrak{D}} \Pi_k\right\}}.$$

Of course, we assume that $M < r$.

5.3.1 Illustrative Examples: Simulated Data

We examine here the effectiveness of the above group classification rule by considering the following scenarios:

(i) The two independent populations are normal: $\Pi_1 = N_5(\mathbf{0}_5, \mathbf{I}_5)$ and $\Pi_2 = N_5(3 \times \mathbf{1}_5, \mathbf{I}_5)$.

(ii) The two independent populations are normal with zero mean and different covariance matrices [68]: $\mathbf{\Sigma}_1 = .75 \times (2/3 \times \mathbf{I}_5 + 1/3 \times \mathbf{J}_5)$ and $\mathbf{\Sigma}_2 = 2 \times (.1\mathbf{I}_5 + .9\mathbf{J}_5)$.

(iii) Two independent populations have the same covariance matrix $\mathbf{I}_5$, but different means: $\Pi_1 = \text{Kotz}\,(\boldsymbol{\mu} = 3 \times \mathbf{1}_5, \mathbf{\Sigma} = 1/6 \times \mathbf{I}_5)$ and $\Pi_2 =$ Central Multivariate t distribution with $(df = 7, \mathbf{\Sigma} = 5/7 \times \mathbf{I}_5)$.

(iv) Two independent populations have the same means but different covariance matrices: $\Pi_1 = \text{Kotz}\left(\boldsymbol{\mu} = \mathbf{0}_5, \mathbf{\Sigma} = 1/6 \times [.75 \times (2/3 \times \mathbf{I}_5 + 1/3 \times \mathbf{J}_5)]\right)$, and $\Pi_2 =$ Central Multivariate t distribution with $(df = 7, \Sigma = 5/7 \times [2 \times (.1\mathbf{I}_5 + .9\mathbf{J}_5)])$.

When population parameters $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and covariance matrix $\mathbf{\Sigma}$ (or $\mathbf{\Sigma}_1$, $\mathbf{\Sigma}_2$) are known, Classification Rules 1 and 2 can be applied directly. Otherwise, when samples $\mathcal{X}_1$ and $\mathcal{X}_2$

Table 5.1 Misclassification rate using "Majority rule" and Optimal classification rule.

Examples	from Group	Group Classification Rule							
		"Majority rule" approach						Optimal Classification	
								Known parameters	
		1	1.A	2	2.A	3	4	Yes	No
(i)	Π_1	0	0	0	0	0	1	0	NA
	Π_2	0	0	0	0	0	0	0	NA
(ii)	Π_1	NA	0.5565	NA	0.5585	0	0	0	NA
	Π_2	NA	0.4125	NA	.4425	.7855	.7245	0	NA
(iii)	Π_1	0	0	0	0	0	0	0	1
	Π_2	0	0	0	0	0	1	0	0
(iv)	Π_1	NA	0.3655	NA	0.365	0	0	.0595	1
	Π_2	NA	.7185	NA	.721	.0075	.022	0	0

are already available, Classification Rules 1.A, 2.A, 3 and 4 can be used. We randomly generate $\mathcal{X}_1$ and $\mathcal{X}_2$ as available samples data from Π_1 and Π_2 respectively. Let $r = 200$ and $n_1 = n_2 = 200$. For the group classification problem, misclassification rates are obtained by generating $m = 2000$ replications of test samples $\mathcal{U}_{1(200\times p)}$ from Π_1 and $\mathcal{U}_{2(200\times p)}$ from Π_2 respectively. It is counted as a correct classification if the entire sample $\mathcal{U}_k$ is correctly classified into Π_k, $k = 1, 2$.

Table 5.1 gives the misclassification rates when Group Classification Rule I is used in each scenario. In case the two populations have the same variance-covariance matrix but different means, as in Examples (i) and (iii), empirical misclassification rates are all 0, except in the case of Classification Rule 4. When using Classification Rule 4, all groups are classified into Π_2 in Example (i), and to Π_1 in Example (iii). Classification Rule 4 uses leverage and performs poorly if the populations have equal covariance matrices. This was somewhat anticipated.

Examples (ii) and (iv) are the scenarios when the two populations have zero means and different variance-covariance matrices. Results in Table 5.1 show that the methods based on the linear discriminant functions (Rules 1.A and 2.A) have serious misclassification issues, as the misclassification rates are around 50% in both cases of Example (ii), and it tends to classify 72% samples from Π_2 into Π_1 in Example (iv). Although the leverage-based and average distance-based rules, namely Classification Rules 3 and 4, do not have good performances in the case of two normal populations (Example (ii)), it performs well in Example (iv) where misclassification rate is at most $.022$.

We also recall that if the probability distributions of the two populations were known, then the optimal classification rules classifying an observation $\mathbf{U}$ based on respective density functions is given by: Classify $\mathbf{U}$ into Π_1 if $pf_1(\mathbf{U}) > (1-p)f_2(\mathbf{U})$, where f_1 and f_2 are the probability density functions and p, $(1-p)$ are the prior probabilities of the respective populations, (see Rao [69], Rencher [60]). The same idea can be generalized in the group classification as given in the theorem below.

Theorem 5.1. *Let p and $1-p$ be the respective prior probabilities that data $\mathcal{U}_{r\times p}$ comes from Π_1 or Π_2. Then $\mathcal{U}$ is classified into Π_1 if*

$$pf_1^*(\mathcal{U}) > (1-p)f_2^*(\mathcal{U}),$$

where f_1^* and f_2^* are the joint pdfs of observations in $\mathcal{U}$, for the populations Π_1 and Π_2 respectively.

If $\mathcal{U}_{r\times p}$ consists of r *i.i.d* observations $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_r$, then the above can be written as

Group Classification Rule II (Optimal classification based on densities)**.** Classify $\mathcal{U}$ to Π_1 if

$$p\prod_{i=1}^{r} f_1(\mathbf{U}_i) > (1-p)\prod_{i=1}^{r} f_2(\mathbf{U})$$
$$\Leftrightarrow \log(p) + \left[\sum_{i=1}^{r} \log f(\mathbf{U}_i)\right] > \log(1-p) + \left[\sum_{i=1}^{r} \log f(\mathbf{U}_i)\right].$$

If f_1 and f_2 are known only up to their functional forms and the corresponding parameters are unknown, then these parameters may have to be estimated from independent random samples, and classification may have to be performed based on estimates $\hat{f_1}$ and $\hat{f_2}$.

Assuming equal prior probabilities $p = 1/2$, Table 5.1 shows that optimal classification performs poorly when parameters are estimated, as it tends to classify all groups into Π_2 even if data is from Π_1 (Examples (iii) and (iv)).

The above discussion and examples show that none of the rules seems to perform uniformly better for all scenarios. This is understandable because, in the case of multivariate data, different probability distributions will give rise to different data structures. Distance-based approaches will not be able to capture various subtle differences in data structures.

5.4 Group Classification by Eigenstructures

In the previous section, we discussed various group classification approaches, but the performances of their classifiers were mixed. This issue occurs because point-wise distance approaches tend to ignore the group structure of data. Borrowing from Khattree's idea [70] of measuring the effect of deleting observations from a given dataset, we will now introduce a new group classification rule that measures the group similarity by observing the effect of appending the to-be-classified data to the available representative samples from each population, and evaluate how it affects the eigenstructures. To do so, let

the data matrices $\mathcal{X}_{n\times p}$ and $\mathcal{U}_{r\times p}$ be of rank p. Let $\mathbf{G}$ be the upper triangular square root matrix of $\mathcal{X}'\mathcal{X}$ such that $\mathbf{G}'\mathbf{G} = \mathcal{X}'\mathcal{X}$. Define

$$\mathbf{H} = \frac{n+r}{n}\mathbf{G}\left(\mathcal{X}'\mathcal{X} + \mathcal{U}'\mathcal{U}\right)^{-1}\mathbf{G}',$$

where we assume that $\mathcal{X}'\mathcal{X} + \mathcal{U}'\mathcal{U}$ is invertible. Clearly, $\mathbf{H}$ is symmetric and positive definite. If $\mathcal{X}'\mathcal{X}$ and $\mathcal{U}'\mathcal{U}$ are close to each other, then $\mathbf{H}$ must resemble $\mathbf{I}_p$, and hence we anticipate its eigenvalues to be close to 1. Thus, a measure of similarity between the two matrices $\mathcal{X}$ and $\mathcal{U}$ can be based on the eigenvalues of matrix $\mathbf{H}$. Therefore, one may wish to suggest an index based on this spectrum. Such an index must express the similarity in a meaningful way by suitably comparing the dispersion of the eigenvalues. One such index can be based on antieigenvalues, or on their functions called the generalized antieigenvalues. We define these below.

Definition 5.1 ([71], [72])**.** Let $\mathbf{A}$ be a $p \times p$ real symmetric positive definite matrix, with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p > 0$. The j^{th} antieigenvalue of matrix $\mathbf{A}$ is defined as $\eta_j = \dfrac{2\sqrt{\lambda_j\lambda_{p-j+1}}}{\lambda_j + \lambda_{p-j+1}}, j = 1 \ldots \left\lceil \dfrac{p}{2} \right\rceil$.

We observe that η_j is indeed the ratio of geometric mean and arithmetic mean of λ_j and λ_{p-j+1}. By arithmetic-geometric means inequality, it follows that $0 < \eta_j \leq 1, j = 1, 2, \ldots, \left\lceil \dfrac{p}{2} \right\rceil$, and the equality holds only when $\lambda_j = \lambda_{p-j+1}$. Hence, for the case $j = 1$, $\eta_1 = 1$ if and only if $\lambda_1 = \lambda_p$, thereby implying that all eigenvalues are equal. We also define the generalized antieigenvalues below.

Definition 5.2 (Generalized antieigenvalues [71])**.** Under the assumptions of Definition 5.1, the generalized antieigenvalue of order $q = 1, 2, \ldots \left\lceil \dfrac{p}{2} \right\rceil$ of a matrix $\mathbf{A}$ is defined as

$$\eta_{[q]} = \prod_{j=1}^{q} \eta_j.$$

Clearly, $\eta_1 = \eta_{[1]}$. The highest order of generalized antieigenvalues,

$$\Delta = \eta_{\lceil p/2 \rceil} = \prod_{j=1}^{\lceil p/2 \rceil} \eta_j = \prod_{j=1}^{\lceil p/2 \rceil} \frac{2\sqrt{\lambda_j \lambda_{p-j+1}}}{\lambda_j + \lambda_{p-j+1}}$$

is a way of combining all information provided by all antieigenvalues. With the use of antieigenvalues and generalized antieigenvalues, we will now introduce some new rules for the group classification problem. To do so, we first state the necessary assumptions and introduce notations below. These will be used later for group classification rules.

Let $\mathcal{U}_{r\times p}$ be the sample matrix with unknown membership, which is to be classified into either Π_1 or Π_2. The populations are assumed to be independent. Suppose $\mathcal{X}_{1(n_1\times p)}$ and $\mathcal{X}_{2(n_2\times p)}$ are the two available random samples from population Π_1 and Π_2 respectively. Let $\mathbf{G}_k$ be the upper triangular square root matrix of $\mathcal{X}_k'\mathcal{X}_k$ such that $\mathbf{G}_k'\mathbf{G}_k = \mathcal{X}_k'\mathcal{X}_k$, $k = 1, 2$. Define matrix $\mathbf{H}_k$ as

$$\mathbf{H}_k = \frac{r + n_k}{n_k}\mathbf{G}_k \left(\mathcal{X}_k'\mathcal{X}_k + \mathcal{U}'\mathcal{U}\right)^{-1} \mathbf{G}_k'. \tag{5.3}$$

We can then classify the group data $\mathcal{U}$ into the population, for which a suitably defined measure based on antieigenvalues is close to 1. Thus, with the first antieigenvalue, the rule reduces to

Group Classification Rule III. Let $\eta_1^{(1)}$ and $\eta_1^{(2)}$ be the first antieigenvalues of $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively. Then, classify $\mathcal{U}$ into Π_1 if $\eta_1^{(1)} \geq \eta_1^{(2)}$.

The first antieigenvalue captures the separation between the largest and the smallest eigenvalues of matrix $\mathbf{A}$. We may similarly measure the separation between other pairs of eigenvalues by other antieigenvalues. In fact, antieigenvalues may also be viewed as the the eccentricity of an ellipsoid defined by the positive definite matrix under consideration and in certain specific directions. See Khattree [71] for more details.

Another measure for classification can be based on the generalized antieigenvalues. We can propose another group classification rule by evaluating $\Delta = \eta_{\lceil p/2 \rceil}$.

Group Classification Rule III.A. Let $\Delta^{(1)}$ and $\Delta^{(2)}$ be the $\left\lceil \frac{p}{2} \right\rceil^{th}$ order generalized antieigenvalues of $\mathbf{H}_1$ and $\mathbf{H}_2$ respectively. Then, classify $\mathcal{U}$ to Π_1 if $\Delta^{(1)} \geq \Delta^{(2)}$.

The following lemma is needed for further discussion in subsequent work.

Lemma 5.1. Let $\mathcal{X}_{n\times p}$ and $\mathcal{U}_{r\times p}$ be two matrices of rank p and $\mathbf{G}$ be the upper triangular square root matrix of $\mathcal{X}'\mathcal{X}$ so that $\mathbf{G}'\mathbf{G} = \mathcal{X}'\mathcal{X}$. Also let $\mathbf{D} = \frac{r}{n}\mathbf{G}\left(\mathcal{U}'\mathcal{U}\right)^{-1}\mathbf{G}'$. Then,

(i) The eigenvalues of $\mathbf{D}$ are same as the eigenvalues of $\frac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'\mathbf{G} = \frac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathcal{X}'\mathcal{X}$.

(ii) Let λ be an eigenvalue of matrix $\mathbf{H} = \frac{n+r}{n}\mathbf{G}(\mathcal{X}'\mathcal{X} + \mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'$. Then $\delta = \frac{r\lambda}{r + n(1-\lambda)}$ is an eigenvalue of matrix $\mathbf{D}$. Accordingly, $\lambda = \frac{(n+r)\delta}{r+n\delta}$.

(iii) Let $\tilde{\mathbf{G}}$ be the upper triangular square root matrix of $\mathcal{U}'\mathcal{U}$, such that $\tilde{\mathbf{G}}'\tilde{\mathbf{G}} = \mathcal{U}'\mathcal{U}$. Denote $\tilde{\mathbf{D}} = \frac{n}{r}\tilde{\mathbf{G}}\left(\mathcal{X}'\mathcal{X}\right)^{-1}\tilde{\mathbf{G}}'$. Then, the antieigenvalues of $\mathbf{D}$ and $\tilde{\mathbf{D}}$ are identical.

Proof. Let δ be an eigenvalue of matrix $\mathbf{D}$, then δ satisfies:

$$
\begin{aligned}
|\mathbf{D} - \delta\mathbf{I}| &= 0 \\
\Leftrightarrow \left|\frac{r}{n}\mathbf{G}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}' - \delta\mathbf{I}\right| &= 0 \\
\Leftrightarrow \left|\frac{r}{n}\mathbf{G}'\mathbf{G}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'\mathbf{G} - \delta\mathbf{G}'\mathbf{G}\right| &= 0 \\
\Leftrightarrow \left|\frac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'\mathbf{G} - \delta\mathbf{I}\right| &= 0 \\
\Leftrightarrow \left|\frac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathcal{X}'\mathcal{X} - \delta\mathbf{I}\right| &= 0,
\end{aligned}
$$

where the second equality follows from the fact that $\mathbf{G}$ is a square upper triangular matrix. This completes the proof for (i).

To prove (ii), we assume that λ is an eigenvalue of matrix $\mathbf{H}$, and apply (i). Thus,

$$\begin{aligned}
0 &= \left|\frac{n+r}{n}(\mathcal{U}'\mathcal{U}+\mathcal{X}'\mathcal{X})^{-1}\mathbf{G}'\mathbf{G}-\lambda\mathbf{I}\right| \\
&= \left|\frac{n+r}{n}\mathbf{G}'\mathbf{G}-\lambda(\mathcal{U}'\mathcal{U}+\mathbf{G}'\mathbf{G})\right| \\
&= \left|\left(\frac{n+r}{n}-\lambda\right)\mathbf{G}'\mathbf{G}-\lambda\mathcal{U}'\mathcal{U}\right| \\
&= \left|\mathbf{G}'\mathbf{G}-\frac{\lambda}{\frac{n+r}{n}-\lambda}\mathcal{U}'\mathcal{U}\right| \\
&= \left|\frac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'\mathbf{G}-\frac{r\lambda}{r+n(1-\lambda)}\mathbf{I}\right|.
\end{aligned}$$

Hence $\dfrac{r\lambda}{r+(1-\lambda)n}$ is an eigenvalue of matrix $\dfrac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'\mathbf{G}$. Thus it is also an eigenvalue matrix of matrix $\mathbf{D}$.

To prove (iii), we note that

$$\mathbf{D}=\frac{r}{n}(\mathcal{U}'\mathcal{U})^{-1}\mathcal{X}'\mathcal{X}=\left(\frac{n}{r}(\mathcal{X}'\mathcal{X})^{-1}\mathcal{U}'\mathcal{U}\right)^{-1}=\tilde{\mathbf{D}}^{-1}$$

and thus, their respective eigenvalues are reciprocals of each other. Hence with $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_p$ as eigenvalues of matrix $\mathbf{D}$, we immediate observe that $\dfrac{1}{\delta_p} \geq \dfrac{1}{\delta_{p-1}} \geq \cdots \geq \dfrac{1}{\delta_1}$ are the eigenvalues of $\tilde{\mathbf{D}}$. Hence for $j = 1, 2, \ldots, \left\lceil \dfrac{p}{2} \right\rceil$,

$$\eta_j^{\tilde{\mathbf{D}}}=\frac{2\sqrt{\dfrac{1}{\delta_j\delta_{p-j+1}}}}{\dfrac{1}{\delta_j}+\dfrac{1}{\delta_{p-j+1}}}=\frac{2\sqrt{\delta_j\delta_{p-j+1}}}{\delta_j+\delta_{p-j+1}}=\eta_j^{\mathbf{D}}.$$

□

Lemma 5.1 gives a monotone relationship between the eigenvalues of matrix $\mathbf{H}$ and $\mathbf{D}$ but does not provide any useful connections between their antieigenvalues. Thus, it may also be interesting to look at the antieigenvalues of matrix $\mathbf{D}$ and accordingly state another rule. For ease in computing, using matrix $\mathbf{D}$ is preferred since in this case, only

the inverse of matrix $\mathcal{U}'\mathcal{U}$ needs to be computed for the calculations of antieigenvalues of $\frac{r}{n_1}(\mathcal{U}'\mathcal{U})^{-1}\mathcal{X}_1'\mathcal{X}_1$ and $\frac{r}{n_2}(\mathcal{U}'\mathcal{U})^{-1}\mathcal{X}_2'\mathcal{X}_2$ and it need to be done only once. Thus, we can state a group classification rule as

Group Classification Rule III.B. Let $\mathbf{D}_1 = \frac{r}{n_1}\mathbf{G}_1(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}_1'$ and $\mathbf{D}_2 = \frac{r}{n_2}\mathbf{G}_2(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}_2'$. Then classify $\mathcal{U}$ into Π_1 if $\eta_1^{(\mathbf{D}_1)} \geq \eta_1^{(\mathbf{D}_2)}$, where $\eta_1^{(\mathbf{D}_i)}$ is the first antieigenvalue of $\mathbf{D}_i$.

We can similarly use the $\left\lceil \frac{p}{2} \right\rceil^{th}$ generalized antieigenvalue as a criterion for the group classification problem.

Group Classification Rule III.C. Under the same assumption of Classification Rule III.B, let $\Delta^{(\mathbf{D}_1)}$ and $\Delta^{(\mathbf{D}_2)}$ be the $\left\lceil \frac{p}{2} \right\rceil^{th}$ order generalized antieigenvalues of $\mathbf{D}_1$ and $\mathbf{D}_2$ respectively. Then classify $\mathcal{U}$ into Π_1 if $\Delta^{(\mathbf{D}_1)} \geq \Delta^{(\mathbf{D}_2)}$.

5.4.1 Illustrative Examples: Simulated Data

We will use the same set of examples as in Section 5.3.1, and hence the same matrices generated in that section as our available samples. Misclassification rates are obtained by $m = 2000$ replicates of the test samples $\mathcal{U}_{1(200\times p)}$ from Π_1 and then also $\mathcal{U}_{2(200\times p)}$ from Π_2. Each of these is then classified into either Π_1 or Π_2.

Table 5.2 shows the empirical misclassification rate when using antieigenvalues and generalized antieigenvalues as measures of similarity in classification problems. In general, the methods based on eigenstructures perform well, as misclassification rates are all 0 or close to 0. Moreover, the use of generalized antieigenvalue of order $\left\lceil \frac{p}{2} \right\rceil$, that is, the product of all antieigenvalues Δ as a measure of similarity does not result in significantly different outcomes compared to the use of first antieigenvalue η_1.

5.5 An Example: Water Potability Dataset

The Water Potability dataset introduced in Chapter 2 consists of 3276 different water bodies from either the healthy water population (Potability = 1) or the unhealthy

Table 5.2 Misclassification rate using Eigenstructures.

Examples	Group from	Group Classification Rule			
		III (η_1)	III.A (Δ)	III.B ($\eta_1^{\mathbf{D}}$)	III.C ($\Delta^{\mathbf{D}}$)
(i)	Π_1	0	0	0	0
	Π_2	0	0	0	0
(ii)	Π_1	0	0	0	0
	Π_2	0	0	0	0
(iii)	Π_1	0	0	0	0
	Π_2	0.0005	0.0005	0	0
(iv)	Π_1	0	0	0	0
	Π_2	0	0	0	0

water population (Potability = 0). These two populations contain nine features: *ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic Carbon, Trihalomethanes, Turbidity*.

If a new sample of size r, say $\mathcal{U}_{r\times 9}$ is obtained, then it is natural to evaluate the potability of this sample. We will classify this group of data into one of the above two populations by group classification rules discussed in Sections 5.3 and 5.4.

No distributional assumptions are made. Hence only Group Classification Rules I (with point-wise Classification Rules 1.A, 2.A, 3, and 4), and eigenstructure related Group Classification Rules III- III.C will be used.

For illustration, we first split samples from each population (Π_1 for Potability = 1 and Π_2 for Potability = 0) into two parts: 70% of data will be used as the representative sample from each population, this is, as $\mathcal{X}_1$ (for Potability = 1) and $\mathcal{X}_2$ (for Potability = 0), respectively. The other 30% will be classified as a whole group by pretending to have an unknown population membership. Let them be $\mathcal{U}_1$ and $\mathcal{U}_2$. We will simulate this scenario for $m = 2000$ replicates and obtain the misclassification rates.

Table 5.3 illustrates results for the group classification problem. Relying on eigenstructure gives excellent performances as the misclassification rates are all 0. The "Majority rule", however, has serious classification issues, as all methods have misclassification rates of at least 61%. Classification Rules 1.A, 2.A, and 4 classify good water samples into the bad water population more than 78% of times. The performance of Classification Rule 3 is just the opposite, as samples from the bad water population are classified as the good water 61% of times. The high misclassification rates among "Majority rule" may be due to the reason that the Classification Rules 1.A and 2.A assume equal covariance matrices but different means. This is not the scenario in this case, since the ratios $\frac{\bar{\mathbf{X}}_{2i}}{\bar{\mathbf{X}}_{1i}}, i = 1, 2 \ldots, 9$ are all close to 1 (see Figure 5.1a). However, similar ratios of corresponding covariance matrices are quite different from 1 (see Figure 5.1b[1]). We also notice that the ratios of eigenvalues of sample covariance matrices from the two populations are close to 1, see Figure 5.1c[2].

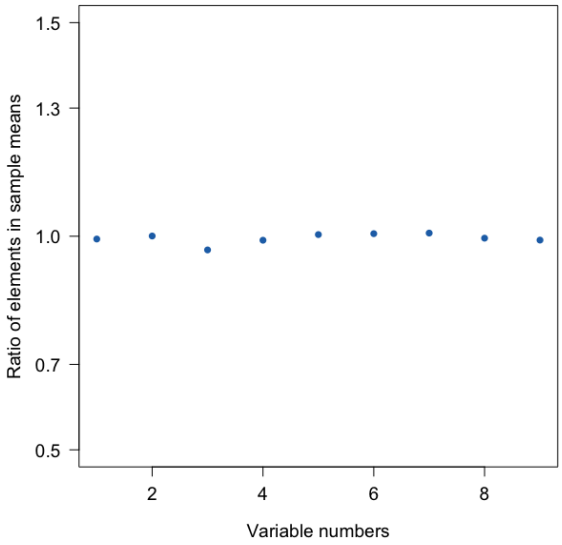
5.6 Feature Selection Using Eigenstructures

To improve the correct classification rates, it is important that the two populations Π_1 and Π_2 are as different as possible. While some features may significantly contribute to the discrimination between the two populations, others might provide only limited or redundant information. In fact, including this redundant information may even mask the differences indicated by other features. The goal of feature selection is to identify a subset of relevant features that results in the maximum discrimination between the populations or samples thereof.

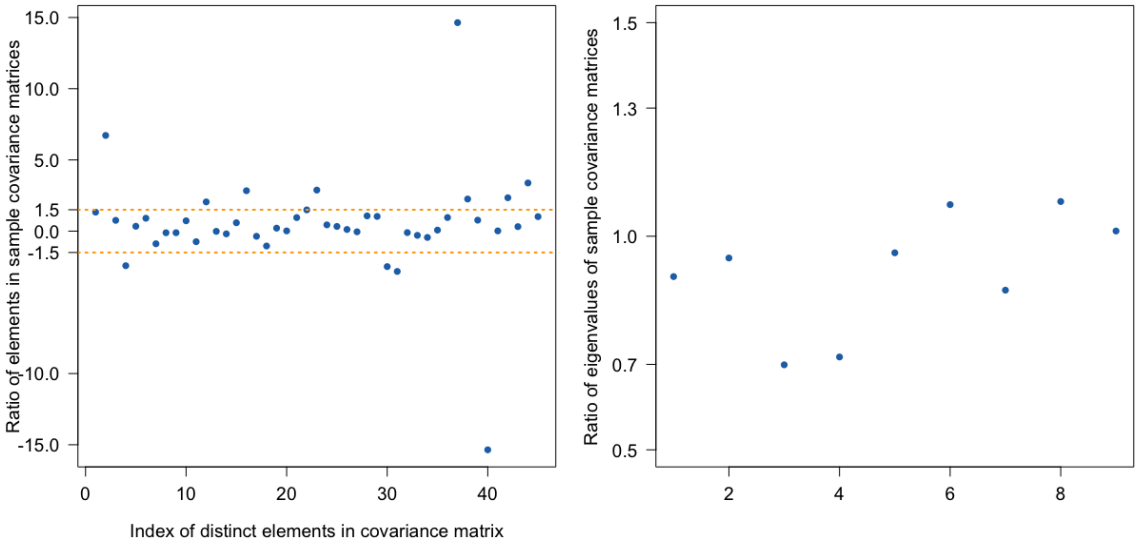
As in Section 5.4, we considered matrix $\mathbf{H} = \frac{n+r}{n}\mathbf{G}(\mathcal{X}'\mathcal{X} + \mathcal{U}'\mathcal{U})\mathbf{G}'$. The similarity or dissimilarity between two populations or their sample counterparts may

[1]Ratio calculated by $\frac{\mathbf{S}_{2(ij)}}{\mathbf{S}_{1(ij)}}, j \leq i = 1, 2, \ldots, 9$

[2]Ratio calculated by $\frac{\lambda_{2i}}{\lambda_{1i}}, i = 1, 2, \ldots, 9$



(a)



(b) (c)

Figure 5.1: Comparison of summary statistics of the good water (Potability = 1) and the bad water (Potability = 0) populations.

Table 5.3 Misclassification rates using Eigenstructures and "Majority rule" - Water Potability dataset.

Group from	Group Classification Rule							
	"Majority - Rule"				Eigenstructures			
	1.A	2.A	3	4	III	III.A	III.B	III.C
Π_1	0.784	0.7825	0	1	0	0	0	0
Π_2	0.3445	0.3425	0.61	0	0	0	0	0

(i) "Majority rule" with point-wise Classification Rule 1.A uses the estimator of linear discriminant function $\hat{\mathcal{D}}$ as a classifier; Classification Rule 2.A uses the average of distances in the discriminant space; Classification Rule 3 uses the average of Mahalanobis distance; Classification Rule 4 use the leverage as a measure of distance;

(ii) Group Rules III and III.A rely on the first antieigenvalue and the $\left\lceil \frac{p}{2} \right\rceil^{th}$ generalized antieigenvalue of the matrix $\frac{n+r}{n}\mathbf{G}(\mathcal{X}'\mathcal{X} + \mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'$;

(iii) Group Rules III.B and III.C rely on the first antieigenvalue and the $\left\lceil \frac{p}{2} \right\rceil^{th}$ generalized antieigenvalue of matrix $\frac{r}{n}\mathbf{G}(\mathcal{U}'\mathcal{U})^{-1}\mathbf{G}'$;

manifest in the eigenstructure of the matrix $\mathbf{H}$. We, therefore, accordingly use $\mathbf{H}$ or any similarly defined matrix to evaluate the discrimination capabilities of classifiers by including or excluding various features in the classifiers.

Given two multivariate data sets $\mathcal{X}_{1(n_1\times p)}$ and $\mathcal{X}_{2(n_2\times p)}$ as representative samples from Π_1 and Π_2, let $\mathcal{Y}_1^J$ and $\mathcal{Y}_2^J$ be the matrices that include only a subset of $q \leq p$ specific columns $J = \{j_1, \ldots j_q\}$ of $\mathcal{X}_1$ and $\mathcal{X}_2$. Also let

$$\mathbf{Q}_1^J = \frac{n_1}{n_2}\left(\mathcal{Y}_1^{J'}\mathcal{Y}_1^J\right)^{-1}\left(\mathcal{Y}_2^{J'}\mathcal{Y}_2^J\right) \text{ and } \mathbf{Q}_2^J = \frac{n_2}{n_1}\left(\mathcal{Y}_2^{J'}\mathcal{Y}_2^J\right)^{-1}\left(\mathcal{Y}_1^{J'}\mathcal{Y}_1^J\right). \tag{5.4}$$

Clearly, there is a one to one correspondence between $\mathbf{Q}_k^J$ and $\mathbf{H}_k^J = \frac{n_1+n_2}{n_k}\mathbf{G}_k^*(\mathcal{Y}_1^{J'}\mathcal{Y}_1^J + \mathcal{Y}_2^{J'}\mathcal{Y}_2)^{-1}\mathbf{G}_k^{*'}$ where $\mathbf{H}_k^J$ is defined in the same way as $\mathbf{H}_k$ but with $\mathcal{X}$ replaced by $\mathcal{Y}_1^J$ and $\mathcal{U}$ replaced by $\mathcal{Y}_2^J$. Here, $\mathbf{G}_k^*$ is an upper triangular matrix with $\mathbf{G}_k^{*'}\mathbf{G}_k = \mathcal{Y}_k^{J'}\mathcal{Y}_k^J$.

Let us define,

$$\zeta_k^J = \frac{q|\mathbf{Q}_k^J|^{1/q}}{tr(\mathbf{Q}_k^J)}, k = 1, 2.$$

The quantities $\zeta_k^J, k = 1, 2$ measure how different matrices $\boldsymbol{\mathcal{Y}}_k'\boldsymbol{\mathcal{Y}}_k$ are from each other. The smaller the values of each ζ_1^J and ζ_2^J, the more different are the two populations. Note that ζ_k^J are the ratios of the geometric means and arithmetic means of eigenvalues of $\mathbf{Q}_k^J$, $k = 1, 2$. If there is only an insignificant difference between $\boldsymbol{\mathcal{Y}}_1^J$ and $\boldsymbol{\mathcal{Y}}_2^J$, then each of matrix $\mathbf{Q}_k^J$ are resemble $\mathbf{I}_q$, and thus $\zeta_k^J, k = 1, 2$ must be close to 1. Therefore, in terms of discrimination, we must look for a subset of column J^* such that either one or both $\zeta_1^{J^*}$ and $\zeta_2^{J^*}$ are small.

We also observe that

$$\begin{aligned}
0 \leq \zeta_1^J \zeta_2^J &= \frac{q^2 \left(|\mathbf{Q}_1^J||\mathbf{Q}_2^J|\right)^{1/q}}{\mathrm{tr}\mathbf{Q}_1^J \mathrm{tr}\mathbf{Q}_2^J} \\
&= \frac{q^2}{\left(\lambda_1^1 + \lambda_2^1 + \ldots \lambda_q^1\right)\left(1/\lambda_1^1 + 1/\lambda_2^1 \ldots 1/\lambda_q^1\right)} \\
&= \frac{q}{\frac{1/\lambda_1^1 + 1/\lambda_2^1 \ldots 1/\lambda_q^1}{\lambda_1^1 + \lambda_2^1 + \ldots \lambda_q^1}} \\
&= \frac{\text{Harmonic mean}(\lambda_1^1, \lambda_2^1, \ldots, \lambda_q^1)}{\text{Arithmetic mean}(\lambda_1^1, \lambda_2^1, \ldots, \lambda_q^1)} \leq 1,
\end{aligned}$$

where λ_i^1 are the eigenvalues of $\mathbf{Q}_1^J$, which are all greater than zero. The equality is achieved when $\lambda_i = \frac{1}{\lambda_i}, i = 1, 2, \ldots q$, that is when all $\lambda_i = 1$. That would be the case when the two population can not be discriminated. The above observation suggests that the geometric means of ζ_J^1 and ζ_J^2 can be used as a measure of discriminatory power for the discrimination between $\boldsymbol{\mathcal{Y}}_1^J$ and $\boldsymbol{\mathcal{Y}}_2^J$. Let

$$\overline{\zeta}^J = \sqrt{\zeta_1^J \zeta_2^J}. \tag{5.5}$$

The smaller the value of $\overline{\zeta}^J$, the better the discrimination.

Thus, we view the choices $\mathcal{Y}_1^{J^*}$ and $\mathcal{Y}_2^{J^*}$ as the best reductions of $\mathcal{X}_1$ and $\mathcal{X}_2$ respectively if $\overline{\zeta}^{J^*} = \sqrt{\zeta_1^{J^*}\zeta_2^{J^*}} = \min_J \left(\sqrt{\zeta_1^J \zeta_2^J}\right)$, where the minimum is taken over all possible subsets of all possible size $q = 1, 2, \ldots, p$.

The number of possible subsets $J \subset \{1, \ldots, p\}$ is 2^p. Finding the best subset of discrimination by considering all possible subsets may become computationally expensive, for large p. We thus introduce the forward and backward selection approaches.

5.6.1 Forward and Backward Selection Algorithms

For a given set of index $J = \left\{j_1, j_2, \ldots, j_q\right\}$ including q columns (or q features), let $\mathcal{Y}_k^J$, $k = 1, 2$ be the matrices containing q columns of $\mathcal{X}_1$ and $\mathcal{X}_2$. By appending a new column $\mathbf{y}_k^j$ from $\mathcal{X}_k$ into $\mathcal{Y}_k^J$, we obtain a set of $q + 1$ columns $\{J, j\}$. We say that new column j, i.e, having feature j, brings more information for the discrimination if $\overline{\zeta}^{\{J,j\}} < \overline{\zeta}^J$. Further, if column $j^* \notin J$ is such that $\overline{\zeta}^{\{J,j^*\}} = \min_{j \notin J} \left\{\overline{\zeta}^{\{J,j\}} | \overline{\zeta}^{\{J,j\}} < \overline{\zeta}^J\right\}$, then the column j^* is the best choice to give furthest discrimination among all possible columns that has not yet been included.

The following lemma is needed to ease the computation burden and help us compute $\overline{\zeta}^{\{J,j\}}$ from $\overline{\zeta}^J$.

Lemma 5.2. Let $\mathbf{Q} = (\mathbf{Y}_1'\mathbf{Y}_1)^{-1}(\mathbf{Y}_2'\mathbf{Y}_2)$ and $\mathfrak{Q} = (\mathcal{Y}_1'\mathcal{Y}_1)^{-1}(\mathcal{Y}_2'\mathcal{Y}_2)$, where $\mathcal{Y}_1 = \begin{bmatrix}\mathbf{Y}_1 & \mathbf{y}_1\end{bmatrix}$ and $\mathcal{Y}_2 = \begin{bmatrix}\mathbf{Y}_2 & \mathbf{y}_2\end{bmatrix}$. Then

(i) Let $B = \mathbf{y}_1' \left[\mathbf{I} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right] \mathbf{y}_1$. Then

$$\begin{aligned}\mathrm{tr}\mathfrak{Q} =& \mathrm{tr}\mathbf{Q} + \frac{1}{B}\left\{\mathbf{y}_1'\left[\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right]'\left[\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right]\mathbf{y}_1\right.\\ &\left.-2\mathbf{y}_2'\left(\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right)\mathbf{y}_1 + \mathbf{y}_2'\mathbf{y}_2\right\},\end{aligned}$$

(ii)

$$|\mathfrak{Q}| = |\mathbf{Q}|\frac{\mathbf{y}_2'\left(\mathbf{I} - \mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_2'\right)\mathbf{y}_2}{\mathbf{y}_1'\left(\mathbf{I} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right)\mathbf{y}_1}.$$

Proof. We have

$$\mathcal{Y}_k'\mathcal{Y}_k = \begin{bmatrix} \mathbf{Y}_k' \\ \mathbf{y}_k' \end{bmatrix} \begin{bmatrix} \mathbf{Y}_k & \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_k'\mathbf{Y}_k & \mathbf{Y}_k'\mathbf{y}_k \\ \mathbf{y}_k'\mathbf{Y}_k & \mathbf{y}_k'\mathbf{y}_k \end{bmatrix}, k = 1, 2.$$

Hence

$$(\mathcal{Y}_1'\mathcal{Y}_1)^{-1} = \begin{bmatrix} (\mathbf{Y}_1'\mathbf{Y}_1)^{-1} + (\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\mathbf{y}_1 B^{-1}\mathbf{y}_1'\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1} & -(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\mathbf{y}_1 B^{-1} \\ -B^{-1}\mathbf{y}_1'\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1} & B^{-1} \end{bmatrix}$$

with $B = \mathbf{y}_1'\mathbf{y}_1 - \mathbf{y}_1'\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\mathbf{y}_1 = \mathbf{y}_1'\left[\mathbf{I} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right]\mathbf{y}_1$. Therefore,

$$\begin{aligned} \mathrm{tr}\mathfrak{Q} &= \mathrm{tr}\left[(\mathbf{Y}_1'\mathbf{Y}_1)^{-1} + (\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\mathbf{y}_1 B^{-1}\mathbf{y}_1'\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\right](\mathbf{Y}_2'\mathbf{Y}_2) \\ &\quad - \mathrm{tr}\left[(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\mathbf{y}_1 B^{-1}\mathbf{Y}_2\mathbf{y}_2\right] - \mathrm{tr}\left[B^{-1}\mathbf{y}_1'\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}(\mathbf{Y}_2'\mathbf{y}_2)\right] + \mathrm{tr}\left[B^{-1}\mathbf{y}_2'\mathbf{y}_2\right] \\ &= \mathrm{tr}\mathbf{Q} + \frac{1}{B}\left[\mathbf{y}_1'\left[\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}(\mathbf{Y}_2'\mathbf{Y}_2)(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right]\mathbf{y}_1 - 2\mathbf{y}_2'\left(\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right)\mathbf{y}_1 + \mathbf{y}_2'\mathbf{y}_2\right] \\ &= \mathrm{tr}\mathbf{Q} + \frac{1}{B}\left[\mathbf{y}_1'\left[\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right]'\left[\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right]\mathbf{y}_1 - 2\mathbf{y}_2'\left(\mathbf{Y}_2(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right)\mathbf{y}_1 + \mathbf{y}_2'\mathbf{y}_2\right]. \end{aligned}$$

To prove (ii), we have

$$|\mathfrak{Q}| = \frac{|\mathcal{Y}_2'\mathcal{Y}_2|}{|\mathcal{Y}_1\mathcal{Y}_1|} = \frac{|\mathbf{Y}_2'\mathbf{Y}_2|\mathbf{y}_2'\left(\mathbf{I} - \mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_2'\right)\mathbf{y}_2}{|\mathbf{Y}_1'\mathbf{Y}_1|\mathbf{y}_1'\left(\mathbf{I} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right)\mathbf{y}_1} = |\mathbf{Q}|\frac{\mathbf{y}_2'\left(\mathbf{I} - \mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_2'\right)\mathbf{y}_2}{\mathbf{y}_1'\left(\mathbf{I} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'\right)\mathbf{y}_1}.$$

□

Let $\overline{\zeta}^{\{J,j\}}$ be calculated by appending the corresponding j^{th} columns into $\mathcal{Y}_1^J$ and $\mathcal{Y}_2^J$ respectively. Lemma 5.2 provides us a method to compute $\overline{\zeta}^{\{J,j\}}$ efficiently, as the terms $(\mathbf{I}_{n_1} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1')$, $(\mathbf{I}_{n_2} - \mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_2')$, $(\mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_2')$ and $(\mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_1')$ do not depend on the appended column and hence are computed only once in the process.

Remark. Since the matrices $\mathcal{Y}_1$ and $\mathcal{Y}_2$ may contain different number of observations $n_1 \neq n_2$, before applying Lemma 5.2, we should rescale each data matrix $\mathcal{X}_k$ by

$\frac{1}{\sqrt{n_k}}, k = 1, 2$. Thus

$$\mathbf{Q}_1 = \frac{n_1}{n_2}(\mathcal{Y}_1'\mathcal{Y}_1)^{-1}(\mathcal{Y}_2'\mathcal{Y}_2) = \left[\left(\frac{1}{\sqrt{n_1}}\mathcal{Y}_1\right)'\left(\frac{1}{\sqrt{n_1}}\mathcal{Y}_1\right)\right]^{-1}\left[\left(\frac{1}{\sqrt{n_2}}\mathcal{Y}_2\right)'\left(\frac{1}{\sqrt{n_2}}\mathcal{Y}_2\right)\right], \text{ and } \mathbf{Q}_2 = \mathbf{Q}_1^{-1}.$$

We now introduce a forward selection algorithm for selecting the most discriminating features. It is stated below.

Algorithm - Forward Selection

---

1. Input: Data matrix $\mathcal{X}_{1(n_1\times p)}$ and $\mathcal{X}_{2(n_2\times p)}$.

2. Initialization: Let $\mathbf{y}_k^j, j = 1, 2, \ldots p$ are columns of matrices $\frac{1}{\sqrt{n_k}}\mathcal{X}_k$. Starting with $q = 2$, find the best two columns $J^* = \{j_1^*, j_2^*\}$ such that $\overline{\zeta}^{J^*} = \min_{\{j_1\neq j_2\}} \overline{\zeta}_{\{j_1,j_2\}}$, see (5.4) and (5.5). Let $\mathbf{Y}_1 = \left[\mathbf{y}_1^{j_1^*} \quad \mathbf{y}_1^{j_2^*}\right]$ and $Y_2 = \left[\mathbf{y}_2^{j_1^*} \quad \mathbf{y}_2^{j_2^*}\right]$, and denote $\mathbf{Q}_k = (\mathbf{Y}_k'\mathbf{Y}_k)^{-1}(\mathbf{Y}_k'\mathbf{Y}_k), k = 1, 2$, and let

   - $\mu_k = \text{tr}\mathbf{Q}_k$ and $\nu_k = |\mathbf{Q}_k|$.
   - $\mathbf{Z}_1 = \mathbf{I}_{n_1} - \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_1'$, $\mathbf{Z}_2 = \mathbf{I}_{n_2} - \mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_2'$,
   - $\mathbf{Z}_{12} = \mathbf{Y}_1(\mathbf{Y}_1'\mathbf{Y}_1)^{-1}\mathbf{Y}_2'$, $\mathbf{Z}_{21} = \mathbf{Y}_2(\mathbf{Y}_2'\mathbf{Y}_2)^{-1}\mathbf{Y}_1'$.

3. Searching for a new feature (or a new column): For each $j = 1, 2 \ldots p$ and $j \notin J^*$, compute $\zeta^{\{J^*,j\}}$ using Lemma 5.2 by the following steps:

   (i) $\mathbf{y}_1 \leftarrow \mathbf{y}_1^j$ and $\mathbf{y}_2 \leftarrow \mathbf{y}_2^j$.

   (ii) Compute $B_1 = \mathbf{y}_1'\mathbf{Z}_1\mathbf{y}_1$ and $B_2 = \mathbf{y}_2'\mathbf{Z}_2\mathbf{y}_2$ and obtain

$$\text{tr}\mathfrak{Q}_1 = \mu_1 + \frac{1}{B_1}\left[\mathbf{y}_1'\mathbf{Z}_{12}\mathbf{Z}_{12}'\mathbf{y}_1 - 2\mathbf{y}_2'\mathbf{Z}_{12}'\mathbf{y}_1 + \mathbf{y}_2'\mathbf{y}_2\right]$$
$$\text{tr}\mathfrak{Q}_2 = \mu_2 + \frac{1}{B_2}\left[\mathbf{y}_2'\mathbf{Z}_{21}\mathbf{Z}_{21}'\mathbf{y}_2 - 2\mathbf{y}_1'\mathbf{Z}_{21}'\mathbf{y}_2 + \mathbf{y}_1'\mathbf{y}_1\right].$$

   (iii) Compute $|\mathfrak{Q}_1| = \nu_1\frac{B_1}{B_2}$ and let $|\mathfrak{Q}_2| = |\mathfrak{Q}_1|^{-1}$.

(iv) Compute $\zeta_k^{\{J^*,j\}} = \dfrac{(q+1)|\mathfrak{Q}_k|^{1/q+1}}{\text{tr}\mathfrak{Q}_k}$, $k = 1, 2$ and $\overline{\zeta}^{\{J^*,j\}} = \sqrt{\zeta_1^{\{J^*,j\}}\zeta_2^{\{J^*,j\}}}$

(v) Choose j^* such that $\overline{\zeta}^{\{J^*,j^*\}} = \min_{j \notin J*} \overline{\zeta}^{\{J^*,j\}}$.

4. Let $q \leftarrow q+1$, $J^* \leftarrow \{J^*, j^*\}$, $\mu_k \leftarrow \text{tr}\mathfrak{Q}_k$ and $\nu_k \leftarrow |\mathfrak{Q}_k|$. Repeat step 3 until $\overline{\zeta}^{J^*}$ can not be made smaller.

With a similar idea, we also introduce the backward feature selection method, in which we start by considering all features of Π_1 and Π_2, and we keep removing redundant features. To do so, we introduce the following criterion for removing a feature.

With a given set of indices J consisting of q columns, let $\mathcal{Y}_k^J$ be the matrices containing J columns of $\mathcal{X}_k$, $k = 1, 2$. Let $j \in J$. By removing column j from each of the matrix $\mathcal{Y}_k^J, k = 1, 2$ we obtain matrices with $q-1$ columns $J\backslash\{j\}$. We say that removing column j (or the feature j) improves the discrimination between the two populations Π_1 and Π_2 if $\overline{\zeta}^{J\backslash\{j\}} < \overline{\zeta}^J$. Thus, if we choose column j^* such that $\overline{\zeta}^{J\backslash\{j^*\}} = \min_{j\in J}\left\{\overline{\zeta}^{J\backslash\{j\}} | \overline{\zeta}^{J\backslash\{j\}} < \overline{\zeta}^J\right\}$, then removing column j^* is the best choice for the removal of a variable from set J.

The backward selection algorithm is as follows.

Algorithm - Backward selection

---

1. Input: Data matrices $\mathcal{X}_1$ and $\mathcal{X}_2$.

2. Initialization: $\mathcal{Y}_k \leftarrow \mathcal{X}_k$. Starting with $q = p$ columns, let $J = \{1, 2, \ldots p\}$ and obtain $\overline{\zeta}$ as in (5.4) and (5.5).

3. Search and remove redundant feater (redundant column):

   (i) For each column in J, let $\mathbf{Y}_k$ be the matrix obtained by removing column j from from $\mathcal{Y}_k$ and denote $\mathbf{Q}_1 = \dfrac{n_1}{n_2}\left(\mathbf{Y}_1'\mathbf{Y}_1\right)^{-1}\left(\mathbf{Y}_2'\mathbf{Y}_2\right)$ and $\mathbf{Q}_2 = \mathbf{Q}_1^{-1}$.

   (ii) For any columns $j \in J$, compute $\overline{\zeta}^{J\backslash\{j\}}$.

(iii) Find column j^* such that $\overline{\zeta}^{J\backslash\{j^*\}} = \min_{j\in J} \overline{\zeta}^{J\backslash\{j\}}$.

(iv) Update $q \leftarrow q-1$, $\overline{\zeta} \leftarrow \overline{\zeta}^{J\backslash\{j^*\}}$, $\mathcal{Y}_1 \leftarrow \mathbf{Y}_1$, $\mathcal{Y}_2 \leftarrow \mathbf{Y}_2$ and $J^* \leftarrow J\backslash\{j^*\}$.

(v) Repeat step 3 until $\overline{\zeta}$ can not be made further smaller.

It may be further remarked that another alternative criterion based on ζ_1^J and ζ_2^J can be taken up for the forward selection, the $\min - \max$ criterion. Specifically, given J, a variable $j^* \notin J$ is selected such that $\min_{j\notin J} \max\left\{\zeta_1^{\{J,j\}}, \zeta_2^{\{J,j\}}\right\}$ is smallest. For backward selection, the criterion for discarding a variable could be based on $\max_{j\in J} \min\left\{\zeta_1^{J\backslash\{j\}}, \zeta_2^{J\backslash\{j\}}\right\}$.

5.6.2 Illustrative Examples: Simulated Data

Example 5.6.1. In this example, we consider the case when two data sets are generated from two independent twenty-dimensional normal populations Π_1 and Π_2 with different means and the same covariance matrices. More specifically, we consider

$$\boldsymbol{\mu}_1 = [0, 15, 0, -10, 0, -10, 0, 0, -5, -5, 7, 0, 15, 7, 0, 0, 7, -10, 0, 0]'$$

and

$$\boldsymbol{\mu}_2 = [0, -10, 0, 0, 0, 15, 0, 15, -5, -10, 7, -5, 0, 7, -10, 0, 0, 0, 7, 0]'$$

and the common covariance matrix $\boldsymbol{\Sigma} = \mathbf{I}_{20}$. The differences in the distributions between Π_1 and Π_2 occur only at a few components of features. Let $\mathcal{X}_1$ be sample matrix of size 500×20 of Π_1 and $\mathcal{X}_2$ be the sample matrix of size 700×20 of Π_2.

The forward selection stops after two steps, as in Table 5.4. It starts by finding the best two variables in terms of discrimination, which results in the smallest $\overline{\zeta}^{\{8,13\}} = 0.00889$. From $J = \{8, 13\}$, it searches for a new variable that contributes most to the discrimination by computing $\overline{\zeta}^{\{8,13,j\}}$, and thus obtains $\overline{\zeta}^{\{8,13,2\}} = 0.008737$ as the smallest. From this, appending new variables does not bring more information to the

Table 5.4 Forward selection - different means and same covariance matrix, simulated example.

Step	Added Variables	$\overline{\zeta}^{J}$
Started columns	8, 13	0.008891
1	2	0.008737
12	7	0.008744

discrimination, as shown in the Table 5.4, where $\overline{\zeta}^{\{8,18,2,7\}} = 0.008744$ is the smallest among all possible sets that can be expanded from the set $J = \{8, 13, 2\}$, and it is even larger than $\overline{\zeta}^{\{8,13,2\}} = 0.008737$. Hence, the algorithm stops and concludes that the set $J = \{8, 13, 2\}$ is the best discrimination subset.

As shown in Table 5.5, $\overline{\zeta} = 0.02083$ when considering all variables. Backward selection tries to remove one variable at each step. The following is the order of variables removed after each successive iteration:
$\{1, 16, 5, 3, 7, 20, 12, 17, 14, 19, 9, 4, 15, 18, 11, 6, 10\}$. The algorithm stops after 17 iterations. In the 18^{th} step, removing the second variable (or variables numbers 8 or 13) results in a larger value of $\overline{\zeta}$. Hence, the algorithm terminates and concludes that the best discrimination variables are $\{8, 13, 2\}$. We also notice that for the variables $\{1, 3, 5, 7, 9, 11, 14, 16, 20\}$, their population means are equal. Hence, most of them have been removed in the early steps of backward selection.

In fact, one observe that $\boldsymbol{\mu}_{1;2} = 15$, $\boldsymbol{\mu}_{1;8} = 0$, $\boldsymbol{\mu}_{1;13} = 15$ and $\boldsymbol{\mu}_{2;2} = -10$, $\boldsymbol{\mu}_{2;8} = 15$, $\boldsymbol{\mu}_{2;13} = 0$. Thus, the differences in population mean manifest their effects via such a small value of $\overline{\zeta}$. Both backward and forward selection algorithms are able to identify such differences in location parameters.

Example 5.6.2. Let $\Pi = N_q(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\Pi_1 = N_p(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\Pi_2 = N_p(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. Let them be independent. Suppose $\boldsymbol{\mathcal{Y}}_{1(n_1\times q)}$ and $\boldsymbol{\mathcal{Y}}_{2(n_2\times q)}$ are data matrices generated from Π, $\boldsymbol{\mathcal{X}}_{1(n_1\times p)}$

Table 5.5 Backward selection - different means and same covariance matrix, simulated example.

Step	Discarded Variables	$\overline{\zeta}^J$
All columns		0.02083
1	1	0.01981
2	16	0.01880
3	5	0.01781
4	3	0.01683
5	7	0.01584
6	20	0.01485
7	12	0.01406
8	17	0.01322
9	14	0.01253
10	19	0.01183
11	9	0.01131
12	4	0.01082
13	15	0.01025
14	18	0.00960
15	11	0.00937
16	6	0.008744
17	10	0.008737
18	2	0.00889

and $\mathcal{X}_{2(n_2\times p)}$ are data matrices generated from Π_1 and Π_2 respectively. Let $\tilde{\mathcal{X}}_1 = \begin{bmatrix} \mathcal{Y}_1 & \mathcal{X}_1 \end{bmatrix}$ and $\tilde{\mathcal{X}}_2 = \begin{bmatrix} \mathcal{Y}_2 & \mathcal{X}_2 \end{bmatrix}$. Then, $\tilde{\mathcal{X}}_1$ and $\tilde{\mathcal{X}}_2$ can be treated as random samples of size $n_1 \times (q+p)$ and $n_2 \times (q+p)$ generated from two independent populations $\tilde{\Pi}_1$ and $\tilde{\Pi}_2$, in which the first q components have the same probability distribution and thus have no discriminating power. We will perform the feature selection algorithms to investigate which subset of columns gives the best discrimination between $\tilde{\mathcal{X}}_1$ and $\tilde{\mathcal{X}}_2$, and hence between $\tilde{\Pi}_1$ and $\tilde{\Pi}_2$.

For our illustrative example, we consider $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \boldsymbol{\mu} = \mathbf{0}_{10}$ and $\boldsymbol{\Sigma} = \mathbf{I}_{10}$. Also, let

$$\boldsymbol{\Sigma}_1 = \begin{bmatrix} 27 & 15 & -1 & -11 & 11 & -14 & -9 & 7 & 7 & -2 \\ 15 & 24 & 2 & -9 & 8 & -11 & -11 & 3 & 1 & -7 \\ -1 & 2 & 6 & 3 & -3 & -3 & -4 & 1 & -2 & 6 \\ -11 & -9 & 3 & 19 & -6 & -1 & 0 & -5 & -8 & 12 \\ 11 & 8 & -3 & -6 & 17 & 2 & -5 & 4 & 0 & -6 \\ -14 & -11 & -3 & -1 & 2 & 33 & 11 & -6 & -7 & 1 \\ -9 & -11 & -4 & 0 & -5 & 11 & 19 & -2 & -1 & -3 \\ 7 & 3 & 1 & -5 & 4 & -6 & -2 & 15 & 3 & -2 \\ 7 & 1 & -2 & -8 & 0 & -7 & -1 & 3 & 15 & -5 \\ -2 & -7 & 6 & 12 & -6 & 1 & -3 & -2 & -5 & 17 \end{bmatrix}$$

and

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} 9 & 13 & 10 & 4 & 11 & 5 & 8 & 6 & 8 & 5 \\ 13 & 29 & 20 & 11 & 22 & 14 & 14 & 14 & 16 & 12 \\ 10 & 20 & 28 & 6 & 22 & 10 & 8 & 16 & 12 & 10 \\ 4 & 11 & 6 & 9 & 11 & 9 & 6 & 8 & 10 & 5 \\ 11 & 22 & 22 & 11 & 25 & 12 & 13 & 18 & 14 & 10 \\ 5 & 14 & 10 & 9 & 12 & 13 & 5 & 9 & 12 & 8 \\ 8 & 14 & 8 & 6 & 13 & 5 & 12 & 8 & 8 & 7 \\ 6 & 14 & 16 & 8 & 18 & 9 & 8 & 17 & 6 & 5 \\ 8 & 16 & 12 & 10 & 14 & 12 & 8 & 6 & 20 & 12 \\ 5 & 12 & 10 & 5 & 10 & 8 & 7 & 5 & 12 & 12 \end{bmatrix}.$$

Clearly, any differences between the two populations are exclusively due to the different covariance structures. Distance-based methods will thus be very poor choices. Our approach, which is based on eigenstructures, is more relevant. Here, $q = p = 10$, we take $n_1 = 500$ and $n_2 = 700$, and generate data from the corresponding populations.

Table 5.6 shows that the forward selection starts with variables number 13 and 15 because $\overline{\zeta}^{\{13,15\}} = .36910$ is the smallest value that can be obtained among all possible subsets with two variables. The forward selection algorithm keeps appending a new column after each iteration until $\overline{\zeta}$ reaches the smallest value $\overline{\zeta}^{J} = 0.00715$ when the set of variables is $J = \{11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$. As in the Table 5.6, the forward selection algorithm tries to find one more column in the complementary set of J, say $\bar{J} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$. Appending the 7^{th} variable gives the smallest $\overline{\zeta}^{\{J,7\}} = .0078004$, and it is still larger than $\overline{\zeta}^{J}$. Hence, the algorithm reports the subset

Table 5.6 Forward selection - same mean and different covariance matrices, simulated example.

Step	Added Variables	$\overline{\zeta}^{J}$
Started columns	13, 15	0.36910
1	17	0.22606
2	14	0.16530
3	19	0.11604
4	12	0.10166
5	18	0.08173
6	16	0.07743
7	20	0.07734
8	11	0.00715
9	7	0.00780

$J = \{11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$ giving the best discrimination between the two populations, with $\overline{\zeta}^{J} = 0.00715$.

Backward selection in Table 5.7, on the other hand, starts by computing $\overline{\zeta}$ for the full matrices $\tilde{X}_1$ and $\tilde{X}_2$, resulting in $\overline{\zeta} = 0.01336$. In each iteration, backward selection tries to remove one variable to increase the discriminating power, i.e., to decrease $\overline{\zeta}$. It stops when the remaining variables are the set $J = \{11, 12, 13, 14, 15, 16, 17, 18, 19, 20\}$ (similar result of the forward selection). Backward selection tries to remove one more variable, and $\overline{\zeta}^{J\setminus\{17\}} = .01582$ is the smallest value that can be obtained from removing a column from J, but it is still larger than $\overline{\zeta}^{J}$. Hence, the algorithm stops and returns the set J as the best discriminating variables.

5.6.3 Variable Selection for Water Potability Data

As we discussed in Section 5.5, the Water Potability dataset can be viewed as samples from two independent populations, namely Potability = 1 and Potability = 0. For both samples, nine features are measured: *ph, Hardness, Solids, Chloramines, Sulfate,*

Table 5.7 Backward selection - same mean and different covariance matrices, simulated example.

Step	Discarded Variables	$\overline{\zeta}^J$
All columns		0.01336
1	3	0.01276
2	1	0.01216
3	6	0.01155
4	8	0.01094
5	4	0.01032
6	10	0.00970
7	9	0.00907
8	5	0.00844
9	2	0.00780
10	7	0.00715
11	17	0.01582

Conductivity, Organic Carbon, Trihalomethanes, Turbidity. Sample means are nearly equal (see Figure 5.1a), and any difference between the two populations manifests only in the covariance structures. We will perform the feature selection for these two samples $\mathcal{X}_{1(811\times9)}$ and $\mathcal{X}_{2(1200\times9)}$ from the respective two populations.

As in Table 5.8, the forward selection identifies variables 1 and 4 (corresponding to *ph* and *Chloramines*) as the best discrimination that can be obtained from any subset of two variables, with $\overline{\zeta}^{\{1,4\}} = 0.9804$. It continues two more steps until reaching the subsets including variables *ph, Chloramines, Sulfate* and *Solids* with $\overline{\zeta}^{\{1,4,3,5\}} = 0.8581$. Appending one more variable gives redundant information and also decreases the discrimination power, as shown in Table 5.8, where the last row has a value of $0.8714 > \overline{\zeta}^{\{1,4,3,5\}}$.

Table 5.9 gives the decrease of $\overline{\zeta}$ when the variables *Conductivity, Trihalomethanes, Organic Carbon, Turbidity, and Hardness* is removed after each step.

Table 5.8 Forward selection - Water Potability dataset.

Step	Added Variables	$\overline{\zeta}^J$
Started columns	1, 4	0.9804
1	5	0.8975
2	3	0.8581
3	2	0.8714

Table 5.9 Backward selection - Water Potability dataset.

Step	Discarded Variables	$\overline{\zeta}^J$
All columns		0.9081
1	6	0.9011
2	8	0.8927
3	7	0.8828
4	9	0.8714
5	2	0.8581
6	4	0.8917

However, continue removing the variable *Chloramines* will decrease the discrimination power (as $\overline{\zeta}$ is increased). Hence, the algorithm terminates and concludes that the four variables *ph, Chloramines, Sulfate, and Solids* bring the most discrimination. This conclusion is identical to the result from forward selection.

In fact, the sample covariance matrices of these four variables are

$$\mathbf{S}_{1J} = \begin{bmatrix} 2.7526 & -2785.8517 & -0.4402 & 13.0618 \\ -2785.8517 & 71590360.3079 & 237.7502 & 2246.6775 \\ -0.4402 & 237.7502 & 2.1803 & -0.2854 \\ 13.0618 & 2246.6775 & -0.2854 & 1324.8438 \end{bmatrix}$$

and

$$\mathbf{S}_{0J} = \begin{bmatrix} 2.0668 & 1147.583 & 0.4965 & -17.6058 \\ 1147.5826 & 79059625.237 & -2140.8333 & -146609.5270 \\ 0.4965 & -2140.833 & 3.0026 & 1.4874 \\ -17.6058 & -146609.527 & 1.4874 & 2251.1410 \end{bmatrix},$$

where $\mathbf{S}_{1J}$ and $\mathbf{S}_{0J}$ are the sample covariance matrices of the corresponding variables set J, calculated as Potability = 1 (Good) water and Potability = 0 (Bad) water samples, respectively. While the sample variances of these features are not very different, all their covariances have opposite signs. Our backward and forward selection procedures and our criterion successfully identify the differences in these covariance patterns, which could not been observed otherwise if the distance-based methods were used.

5.7 A Few Comments on Limitations

We must point out that our approach is not universal and cannot always outperform the distance-based approaches. Our approach's utility lies in its application for non-routine, specialized applications where distance is simply not an appropriate metric. In addition to the examples given above, discrimination for copulas will be another such situation, which will be discussed in detail in the next chapter.

To reinforce our point further, we must point out that our approach relies on comparing the eigenstructures of $\frac{1}{n_k}\mathcal{X}_k'\mathcal{X}_k, k = 1, 2$. Clearly $\mathbb{E}\left(\frac{1}{n_k}\mathcal{X}_k'\mathcal{X}_k\right) = \mathbf{\Sigma}_k + \mu_k\mu_k'$. The two populations may have the same means and equal variance-covariance matrices, yet their probability distributions may be drastically different. One such scenario can be provided by taking the two independent populations, Π_1 = multivariate Lomax $(a = 3, \theta = \mathbf{1}_5)$ (see [58]) and $\Pi_2 = N_5(\boldsymbol{\mu}^*, \mathbf{\Sigma}^*)$, where $\boldsymbol{\mu}^* = .5\mathbf{1}_5$ and $\mathbf{\Sigma}^* = .75\,(2/3\mathbf{I}_5 + 1/3\mathbf{J}_5) = 1/2\mathbf{I}_5 + 1/4\mathbf{1}_5\mathbf{1}_5'$. One can check that Π_1 and Π_2 both have the

same means and equal covariance matrices. Thus the eigenstructures of the $\frac{1}{n_k}\mathcal{X}_k\mathcal{X}_k', k = 1, 2$ may not be very distinguishable.

Another situation when the above approach would be ineffective is the scenario in which the two populations are orthogonal to each other, that is $\mathcal{X}_2 \overset{\mathfrak{D}}{\equiv} P\mathcal{X}_1$, where P is a $p \times p$ orthogonal matrix. Clearly in this case, $\mathcal{X}_2'\mathcal{X}_2 \overset{\mathfrak{D}}{\equiv} \mathcal{X}_1'P'P\mathcal{X}_1 \overset{\mathfrak{D}}{\equiv} \mathcal{X}_1'\mathcal{X}_1$. Thus, no discrimination may be possible even though both populations may have very different means and covariance matrices. The same comment applies if $\mathcal{X}_2 \overset{\mathfrak{D}}{\equiv} c\mathcal{X}_1$ where c is a nonzero constant.

5.8 Conclusions and Remarks

Individual classifications are not always an effective tool for the group classification problem, as they tend to focus on differences in location parameters. A single observation may heavily skew the estimated location parameter but may not substantially change the eigenstructures. Hence, comparing eigenstructures as a criterion to perform the task of discrimination or classification shows effective results. However, the method may have a high misclassification rate if both mean and covariance variance parameters of the two populations overlap or if two samples have a one-to-one correspondence between them through an orthogonal transformation.

The approach of analyzing eigenstructure is also extended to the context of variable selection or data reduction to find "the most discriminating" variables between two groups of data. We have introduced both forward and backward selection to either appending or removing one column at a time. The forward selection method looks for variables that contribute most to the discrimination at each iteration, and the backward selection approach identifies and removes, at any iteration, features that possess little information over and above the remaining variables.

We have considered the geometric means of ζ_1 and ζ_2 of the ratio matrices $\mathfrak{Q}_1 = \frac{n_1}{n_2}\left(\mathcal{Y}_1'\mathcal{Y}_1\right)^{-1}\left(\mathcal{Y}_2'\mathcal{Y}_2\right)$ and $\mathfrak{Q}_1 = \mathfrak{Q}_1^{-1}$ as a criterion of finding the best separation.

Other criteria to quantify the difference in eigenstructures between two sets of data samples can also be used. One such criterion may make use of the eigenstructures of matrices $H_1 = \frac{n_1 + n_2}{n_1} \left(\mathcal{X}_1'\mathcal{X}_1 + \mathcal{X}_2'\mathcal{X}_2\right)^{-1} \left(\mathcal{X}_1'\mathcal{X}_1\right)$ and $H_2 = \frac{n_1 + n_2}{n_2} \left(\mathcal{X}_1'\mathcal{X}_1 + \mathcal{X}_2'\mathcal{X}_2\right)^{-1} \left(\mathcal{X}_2'\mathcal{X}_2\right)$. If $n_1 = n_2$, then $H_1 + H_2 = 2\mathbf{I}_p$, and hence the two criteria are equivalent.

CHAPTER SIX

ON CHOOSING COPULAS

6.1 Introduction

In this chapter, we consider the applications of the group classification rule to evaluate copulas assumptions. Choosing a suitable copula is an important step when modeling multivariate data and assessing the nature of dependence. Copulas provide a way to evaluate dependence among variables while keeping it free from the consideration of the marginal distributions, thereby offering flexibility and allowing one a more accurate representation of the dependence structure. Hence, choosing a copulas is indeed a problem of finding an appropriate description of the dependencies between variables.

Specifically, we consider the problem of choosing the "best" copula that optimally fits our data in some meaningful senses. Suppose there is a finite sample from one of the potential copulas $C_1 \ldots C_M$. We want to choose the copula that fits best among these. *Valdo Durrleman et al.* [73] proposed a method to choose an appropriate copula for data, which makes use of the empirical copula probabilities as described by Deheuvels [74]. His idea was first to compute empirical copula probabilities, which are compared to the theoretical probabilities. Their criterion of the "best" copula was the smallest average distance between these empirical and theoretical probabilities. The method becomes computationally intensive as the dimension p or the sample size n increases.

Graphically, a copula is often chosen based on the resemblance of the scatter plot of the to-be-classified data with that for the random samples from various copulas. Such graphical approaches can be applicable only in two or three dimensions. However, the dependence structure can be considerably more complex than what can be visually depicted in two or three dimensions. Further comprehending and combining the subjective

evaluations from numerous two or three-dimensional plots to arrive at a choice of p-dimensional copula may not be an easy task.

Our approach is based on evaluating the similarity of eigenstructures, and the problem here is viewed as a group classification problem. Thus, Section 6.2 reviews the group classification rules. Illustrative examples for simulated data are presented in Section 6.3. Finally, we apply our methodology to the stock data set in Section 6.4. We conclude our chapter with some critical remarks.

6.2 Some Approaches for Copula Selection

Suppose a dataset and M copulas candidates $C_1, C_2, \ldots, C_M$ are available. In case the functional form but not the parameters are known, we can obtain the estimators, say $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_M$ of $C_1, C_2, \ldots C_M$ by estimating the corresponding parameters. To do so, one may employ the method of maximum likelihood and obtain the estimates of the corresponding parameters. Let $\mathcal{U}_{r\times p}$ be the data for which we must choose a copula. Also suppose $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$ are the respective representative random samples from copulas $C_1, C_2, \ldots, C_M$ (or $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_M$) consisting of $n_1, n_2, \ldots, n_M$ observations. We view the problem as a group classification problem by evaluating the similarity of data $\mathcal{U}$ with each of the $\mathcal{X}_k$, $k = 1, 2, \ldots M$. Note that for any copula, marginals are all uniform with means $1/2$, and variances $1/12$. Thus, any distance-based approach to measure similarity is unlikely to be very effective. The similarity has to be evaluated via the closeness of dependence structure, which may be, to a great extent, reflected in the eigenstructures of $\mathcal{U}$ and $\mathcal{X}_k$. Thus, we may rely on the methods developed in the previous chapter, and here, we therefore summarize various approaches of group classification based on the eigenstructure of the data. We assume $p \leq r + \min_{k=1,\ldots,M} n_k$.

Choosing Copulas: Rule 1. Let $H_k, k = 1, 2, \ldots M$ be the symmetric positive definite matrix defined as

$$H_k = \frac{n_k + r}{n_k} G_k \left(\mathcal{X}_k' \mathcal{X}_k + \mathcal{U}' \mathcal{U} \right)^{-1} G_k', k = 1, 2, \ldots, M$$

where G_k is a $p \times p$ upper triangular square root matrix such that $G_k' G_k = \mathcal{X}_k' \mathcal{X}_k$. Denote by $\eta_1^{(k)}$ as the first antieigenvalue of H_k. Then we classify $\mathcal{U}$ to C_{k^*} if $\eta_1^{(k^*)} = \max_{k=1 \ldots M} \eta_1^{(k)}$.

Dependence may especially affect how the variables within an observation are associated towards the extremes. Thus, when choosing a copula, it is important to emphasize the tail behaviors (that is, near 0 or 1). It is, therefore, reasonable to evaluate the differences in various regions of the interval $[0, 1]$. Thus, we will divide this interval into three segments, namely, $[0, t_1], (t_1, t_2], (t_2, 1]$ for one of the variables within $\mathcal{U}$. Let this component be the first one, U_1. Thus data with $\mathcal{U}$ are divided into three subsets, namely, $U_1 \leq t_1, t_1 < U_1 \leq t_2, U_1 > t_2$. Let us denote these subsamples as $\mathcal{U}^1, \mathcal{U}^2$ and $\mathcal{U}^3$. The same subsetting process is performed on each of the data $\mathcal{X}_k$ to obtain $\mathcal{X}_k^1, \mathcal{X}_k^2$ and $\mathcal{X}_k^3$, $k = 1, 2, \ldots, M$. Based on this, we suggest the following as an alternative to Rule 1.

Choosing Copulas: Rule 2. Let $\eta_1^{(kj)}$ be the first antieigenvalues of matrix

$$H_k^j = \frac{n_{kj} + r_j}{n_{kj}} G_k^j \left(\mathcal{X}_k^{j'} \mathcal{X}_k^j + \mathcal{U}^{j'} \mathcal{U}^j \right)^{-1} G_k^{j'}, k = 1, 2, \ldots, M \text{ and } j = 1, 2, 3. \tag{6.1}$$

where as earlier $G_k^{j'} G_k^j = \mathcal{X}_k^{j'} \mathcal{X}_k^j$ and G_k^j is the $p \times p$ upper triangular matrix. The quantities n_{kj} and r_j are the number of observations in matrices $\mathcal{X}_k^j$ and $\mathcal{U}^j$, respectively. Let w^1, w^2, and w^3 be the (known) weights assigned to each of the three regions[1] Define $\bar{\eta}_1^{(k)} = w^1 \eta_1^{(k1)} + w^2 \eta_1^{(k2)} + w^3 \eta_1^{(k3)}$. Then classify $\mathcal{U}$ to C_{k^*} if $\bar{\eta}_1^{(k^*)} = \max_{k=1,\ldots M} \bar{\eta}_1^{(k)}$.

With the given copulas $C_1, C_2, \ldots, C_M$ or their estimates versions $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_M$, the optimal classification based on density functions discussed in Section 5.3 is applicable.

[1]We may perhaps assign more weights to tails.

We restate the optimal rule in this situation, as given below. To avoid being repetitive, the rule is stated for the estimated version only.

Choosing Copulas: Rule 3. Let p_k be the prior probability that $\mathcal{U}$ is from C_k (or $\hat{C}_k$), $k = 1, 2, \ldots, M$. Then, classify $\mathcal{U}$ into C_{k^*} if

$$p_{k^*} \prod_{i=1}^{r} \hat{c}_{k^*}(\mathbf{U}_i) = \max_{k=1,\ldots,M} p_k \prod_{i=1}^{r} \hat{c}_k(\mathbf{U}_i),$$

where $\hat{c}_k(.)$ is the copula density corresponding to $\hat{C}_k$.

6.3 A Simulated Study

Example 6.3.1 (Bivariate data)**.** In this bivariate example, we will illustrate as well as evaluate the performance of our approach to the problem of finding an optimal copula. We generate samples from the bivariate Frank, Gumble, Clayton, Joe, t and Normal copulas, each with Kendall Tau is $\tau = .7$. Note that the first four copulas belong to the Archimedean family, while the last two are from the elliptical family. These data sets are taken as the "to be classified" sample, denoted by $\mathcal{U}$ in each of the six cases. Table 6.1 summarizes the corresponding generators and parameters of these copulas. In Figure 6.1, we give the scatter plots of $n = 500$ data points generated from each of the six copulas. In Figure 6.2, we illustrate the joint distribution when each bivariate data is marginally normally distributed, but the dependence structure is expressed by one of the six copulas mentioned above.

Clayton and Joe copulas appear to have opposite trends. The data from the Clayton copula span wider in the right tail, but the Joe copula data spread more in the lower left tail. The distributions of data around the centers of Frank copula and Normal copula look similar, but their tail behaviors differ - Frank copula has thicker tails. Data from $t(df = 2)$ copula contain many more extreme values compared to other copulas. This illustrates that copulas may still have substantially different tail behavior despite having the same Kendall's tau correlation.

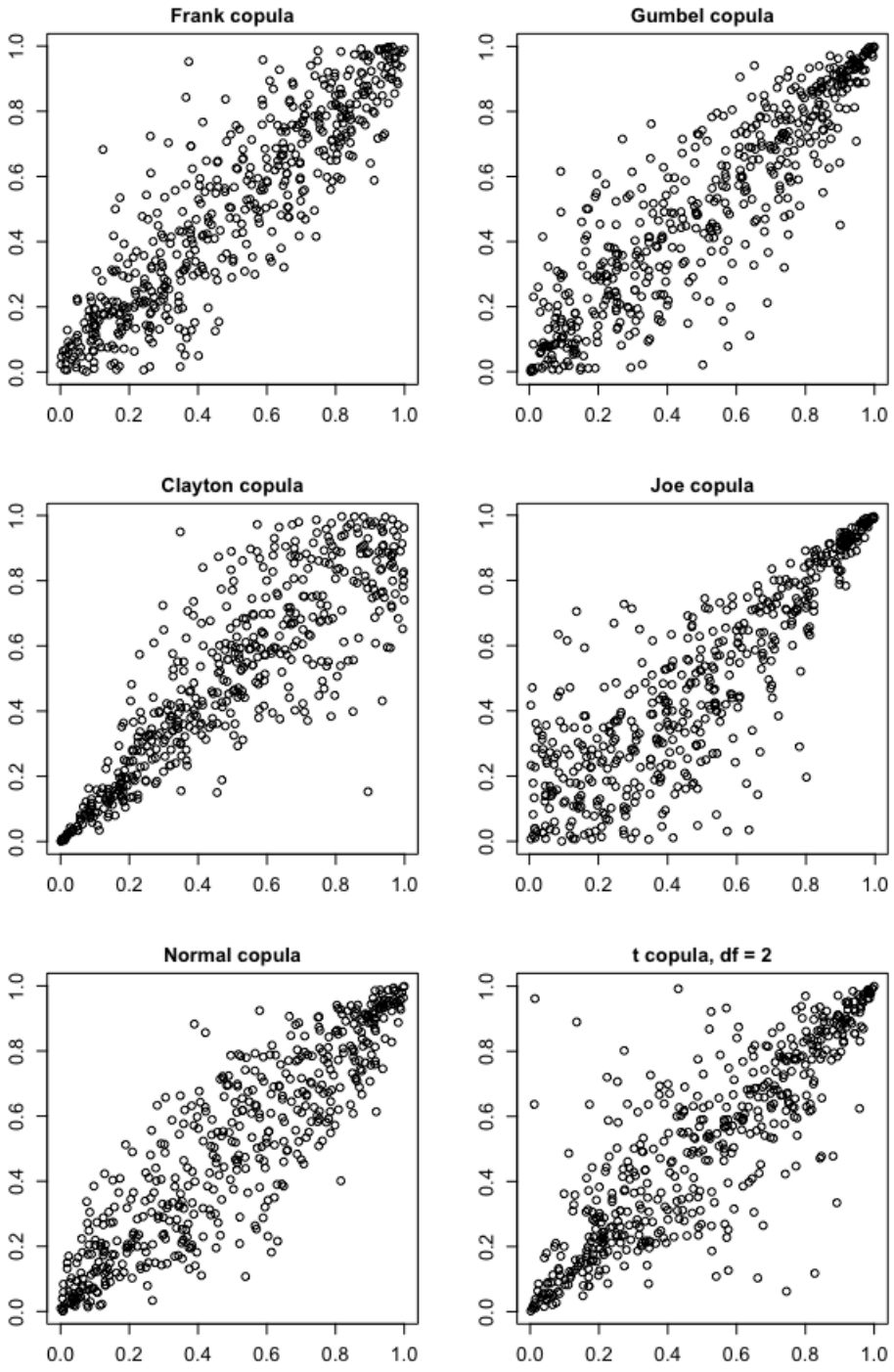


Figure 6.1: Scatter plots of certain bivariate copulas ($n = 500$, Kendall's $\tau = 0.7$).

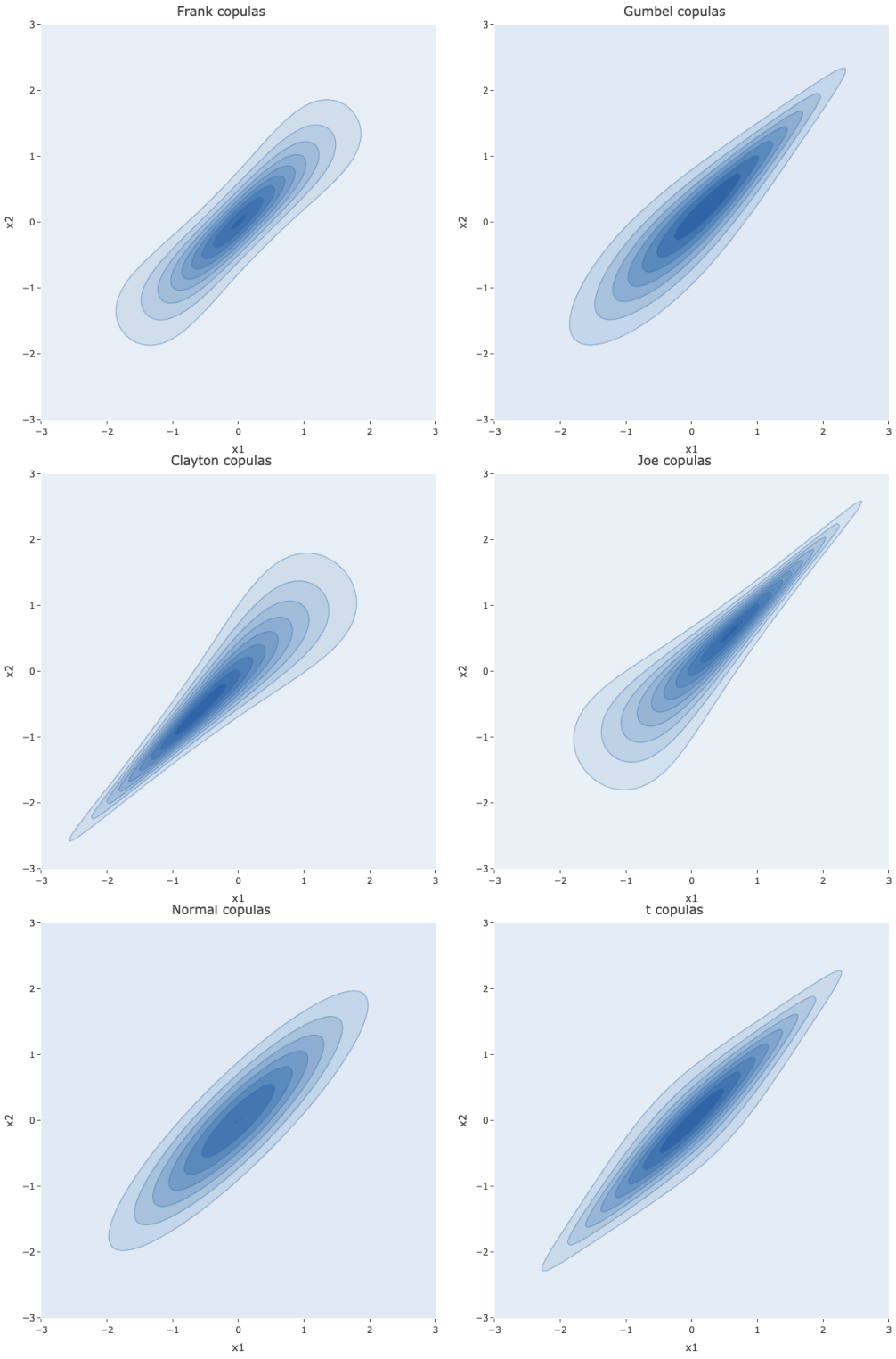


Figure 6.2: Contour plots of various copulas with Kendall's $\tau = 0.7$ and standard marginal normal.

Table 6.1 Some popular copulas and their Kendall's tau correlation.

Copulas	Generator $\phi(t)$	Parameters	Kendall's tau correlation in bivariate case
Clayton	$\frac{1}{\theta}(t^{-\theta}-1)$	$\theta \geq -1, \theta \neq 0$	$\frac{\theta}{\theta+2}$
Gumbel	$(-\log(t))^{\theta}$	$\theta \geq 1$	$1-\frac{1}{\theta}$
Frank	$-\log\left(\frac{e^{-\theta t}-1}{e^{-\theta}-1}\right)$	$\theta \neq 0$	$1+\frac{4}{\theta}(D_1(\theta)-1)$, $D_1(\theta)=\frac{1}{\theta}\int_0^{\theta}\frac{t}{e^t-1}dt$
Joe	$-\log(1+t^{\theta})$	$\theta \in \mathbb{R}$	$1+\frac{4}{\theta^2}\int_0^1 x\log(x)(1-x)^{2(1-\theta)/\theta}dt$
Normal	NA	$\Sigma_{p\times p}$	
		$-1 \leq \rho \leq 1$ $(p=2)$	$\tau = \arcsin(\rho)$
t	NA	$\Sigma_{p\times p}$, $df > 0$	
		$-1 \leq \rho \leq 1$ and $df > 0$ $(p=2)$	$\tau = \arcsin(\rho)$

Assuming that data $\mathcal{U}_{r\times 2}$ are from $C_k, k = 1, \ldots, 6$ (for Frank, Gumble, Clayton, Joe, Normal and t copulas, respectively). We obtain the corresponding maximum pseudo-likelihood estimators $\hat{C}_k$ using the R-package *copula* [75] [2]. To imitate the realistic scenario, we generate the samples $\mathcal{X}_{k(n_k\times p)}$, $k = 1, \ldots, 6$ from $\hat{C}_k$ (as opposed to C_k). We compare the eigenstructure of $\mathcal{U}$ with each of the $\hat{C}_k$ using the criteria described in Rules 1 and 2. We take the choice of $t_1 = .35$ and $t_2 = .65$. Accordingly, the subsamples are obtained for the region $[0, .35]$, $(.35, .65]$ and $[0.65, 1]$ of the first component of the bivariate vector when using Rule 2. Antieigenvalues of these subsamples may be weighted and two choices of weight vectors will be used, including assigning more weight into the two tails $w_1 = [.4, .2, .4]'$ or weighing each region equally, $w_2 = [1/3, 1/3, 1/3]'$. We perform simulations with $r = 500$, and take $n_1 = n_2 = \cdots = n_k = n$. We also vary the ratio

[2] If the maximum likelihood method does not converge, we use the moment estimators instead.

n/r as $1, 2, 4, 8, 16, 32$ in the case of weighing each region equally. Number of simulation is taken to be $m = 2000$.

Table 6.2 gives the empirical probabilities of correct classification when data actually come from the given copula. In general, Rule 2 is more effective than Rule 1. Further assigning higher weights to the intervals toward the extreme appears to have no effect compared to the equal weight scenario. The probabilities of correct classification increase as the ratio $\frac{n}{r}$ increases. Surprisingly, the method does not seem to be very effective for correctly classifying data from Normal copula.

Table 6.3 gives the highest misclassification rate for classifying $\mathcal{U}$ in each scenario of the six generated $\mathcal{U}$. Most of the time, data generated from other copulas are wrongly classified to t-copula. In the case $\mathcal{U}$ comes from Joe copula, these are mostly misclassified into Gumble copula (but the misclassification rate is small). Data from elliptical families is misclassified only within elliptical families. As Normal and t are both symmetric, the only difference in some extreme values may be at the tail. Performance is often not good because all six families have the same values of Kendall's tau, and thus, they have some degree of similarity in some ways.

Example 6.3.2 (Multivariate Data)**.** We will now investigate the performance of the eigenstructure method in the problem of choosing an appropriate copula for a multivariate data set. To do so, we will generate the "to be classified" sample $\mathcal{U}_{1000\times 5}$ of size $r = 1000$ for the $p = 5$ dimensional distribution for the following copulas: Frank($\alpha = 2$),

Table 6.2 Estimated probabilities of correct classification for various copulas ($p = 2$, sample size $r = 500$, Kendall's $\tau = 0.7$).

Method	**Data From**					
	Frank	Gumbel	Clayton	Joe	Normal	$t(df = 2)$
Rule 1 ($n/r = 1$)	0.6125	0.1975	0.4005	0.2115	0.2530	0.2380
Rule 2^{w_1} ($n/r = 1$)	0.8630	0.4905	0.9075	0.8205	0.3925	0.3950
Rule 2^{w_2} ($n/r = 1$)	0.8835	0.5045	0.9145	0.8215	0.3990	0.4095
Rule 2^{w_2} ($n/r = 2$)	0.9520	0.5995	0.9480	0.9020	0.4130	0.4975
Rule 2^{w_2} ($n/r = 4$)	0.9775	0.6850	0.9615	0.9150	0.4355	0.5635
Rule 2^{w_2} ($n/r = 8$)	0.9860	0.7395	0.9670	0.9335	0.4315	0.6240
Rule 2^{w_2} ($n/r = 16$)	0.9870	0.7695	0.9635	0.9370	0.4555	0.6475
Rule 2^{w_2} ($n/r = 32$)	0.9875	0.7775	0.9660	0.9385	0.4305	0.6655
Rule 3	1	0.9990	1	1	0.5905	1

For Rule 2 and its variants, the weight vectors are $w_1 = [.4, .2, .4]'$ and $w_2 = [1/3, 1/3, 1/3]'$.

Gumbel($\alpha = 2$), Clayton($\alpha = 2$), Joe($\alpha = 2$), Normal(Σ) and $t(\Sigma, df = 2)$, where

$$\Sigma = \begin{bmatrix} 1.0 & -0.1 & 0.1 & -0.1 & 0.5 \\ -0.1 & 1.0 & 0.2 & -0.1 & 0.5 \\ 0.1 & 0.2 & 1.0 & 0.1 & 0.5 \\ -0.1 & -0.1 & 0.1 & 1.0 & 0.2 \\ 0.5 & 0.5 & 0.5 & 0.2 & 1.0 \end{bmatrix}.$$

Notice that the parameters for different copulas have very different interpretations. Thus, despite having the same parameter value (= 2) for the first four copulas, they may or may not be very similar. Thus, the multivariate example contrasts itself with the previous bivariate example when all copulas had the same value of Kendall's tau.

Table 6.3 The highest of estimated probabilities of misclassification for various copulas ($p = 2$, sample size $r = 500$, Kendall's $\tau = 0.7$).

Method		**Data From**					
		Frank	Gumbel	Clayton	Joe	Normal	$t(df = 2)$
Rule 1	Classified to	Clayton	Normal	Frank	Clayton	Frank	Normal
($n/r = 1$)	Rate	0.3595	0.227	0.35	0.2735	0.2615	0.305
Rule 2^{w_1}	Classified to	*t*	Normal	*t*	Gumbel	*t*	Normal
($n/r = 1$)	Rate	0.076	0.157	0.0375	0.153	0.39	0.3645
Rule 2^{w_2}	Classified to	*t*	Normal	*t*	Gumbel	*t*	Normal
($n/r = 1$)	Rate	0.068	0.1655	0.034	0.158	0.3805	0.3505
Rule 2^{w_2}	Classified to	*t*	*t*	*t*	Gumbel	*t*	Normal
($n/r = 2$)	Rate	0.025	0.1315	0.0235	0.089	0.417	0.3325
Rule 2^{w_2}	Classified to	*t*	*t*	*t*	Gumbel	*t*	Normal
($n/r = 4$)	Rate	0.013	0.1075	0.019	0.0825	0.4185	0.32
Rule 2^{w_2}	Classified to	*t*	*t*	*t*	Gumbel	*t*	Normal
($n/r = 8$)	Rate	0.0105	0.089	0.016	0.065	0.437	0.2725
Rule 2^{w_2}	Classified to	*t*	*t*	*t*	Gumbel	*t*	Normal
($n/r = 16$)	Rate	0.008	0.0875	0.018	0.0625	0.419	0.252
Rule 2^{w_2}	Classified to	*t*	*t*	*t*	Gumbel	*t*	Normal
($n/r = 32$)	Rate	0.0095	0.094	0.02	0.0615	0.4435	0.239
Rule 3	Classified to	Gumbel	*t*	Frank	Frank	*t*	Frank
	Rate	0	0.001	0	0	0.4095	0

For Rule 2 and its variants, the weight vectors are $w_1 = [.4, .2, .4]'$ and $w_2 = [1/3, 1/3, 1/3]'$.

Table 6.4 Estimated probabilities of correct classification for various copulas ($p = 5$, sample size $r = 1000$).

Method	**From**					
	Frank	Gumbel	Clayton	Joe	Normal	$t(df = 2)$
Rule 1 ($n/r = 1$)	0.1215	0.1150	0.1075	0.0975	0.5050	0.4065
Rule 2^{w_1} ($n/r = 1$)	0.4730	0.4865	0.8535	0.5955	0.4990	0.5845
Rule 2^{w_2} ($n/r = 1$)	0.4945	0.5340	0.8950	0.6470	0.4950	0.5890
Rule 2^{w_2} ($n/r = 2$)	0.6280	0.6555	0.9690	0.7545	0.4765	0.7140
Rule 2^{w_2} ($n/r = 4$)	0.7435	0.7400	0.9885	0.8125	0.5030	0.8010
Rule 2^{w_2} ($n/r = 8$)	0.7605	0.8280	0.9945	0.8855	0.4895	0.8675
Rule 2^{w_2} ($n/r = 16$)	0.8110	0.8605	0.9950	0.9050	0.4980	0.9070
Rule 2^{w_2} ($n/r = 32$)	0.8490	0.8900	0.9960	0.9220	0.4995	0.9240
Rule 3	1	1	1	1	0.8185	1

For Rule 2 and its variants, the weight vectors are $w_1 = [.4, .2, .4]'$ and $w_2 = [1/3, 1/3, 1/3]'$.

Tables 6.4 and 6.3 summarize the performances of various group classification methods. As shown in Table 6.4, one requires a very large representative sample size to attain high correct classification rates. The highest rate in each case of the to-be-classified $\mathcal{U}_{1000\times5}$ was obtained when using eigenstructures by region with sample size $n = 32 \times 1000$. The highest rate is $.9960$ when $\mathcal{U}$ is from the Clayton copula. In Table 6.5, we observe a very different pattern of misclassification rates compared to bivariate data in Table 6.3. Frank copula correlation structure is wrongly assumed to be in Gumbel copula with a high misclassification rate at $.137\%$. Data generated from the Gumbel copula and Joe copula are wrongly classified to each other. Finally, the misclassification rate in the case $\mathcal{U}$ is generated from a normal copula is particularly high and is mostly wrongly classified as a t copula.

In both examples of bivariate and multivariate ($p = 5$) copula classification, we observe that the optimal classification rule (Rule 3) has a very impressive performance. In

Table 6.5 The highest estimated probabilities of misclassification for various copulas ($p = 5$, sample size $r = 1000$).

Method		**From**					
		Frank	Gumbel	Clayton	Joe	Normal	$t(df = 2)$
Rule 1	Classified to	Normal	Normal	Normal	Normal	t	Normal
($n/r = 1$)	Rate	0.3525	0.427	0.3935	0.3895	0.495	0.5935
Rule 2^{w_1}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 1$)	Rate	0.3435	0.272	0.0835	0.355	0.501	0.4155
Rule 2^{w_2}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 1$)	Rate	0.31	0.2385	0.061	0.324	0.505	0.411
Rule 2^{w_2}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 2$)	Rate	0.273	0.1885	0.018	0.2345	0.5235	0.286
Rule 2^{w_2}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 4$)	Rate	0.2105	0.1435	0.0075	0.185	0.497	0.199
Rule 2^{w_2}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 8$)	Rate 2	0.207	0.0985	0.0035	0.1145	0.5105	0.1325
Rule 2^{w_2}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 16$)	Rate	0.1655	0.083	0.004	0.095	0.502	0.093
Rule 2^{w_2}	Classified to	Gumbel	Joe	Normal	Gumbel	t	Normal
($n/r = 32$)	Rate	0.137	0.0705	0.003	0.078	0.5005	0.076
Rule 3	Classified to	Gumbel	Frank	Frank	Frank	t	Frank
	Rate	0	0	0	0	0.1815	0

For Rule 2 and its variants, the weight vectors are $w_1 = [.4, .2, .4]'$ and $w_2 = [1/3, 1/3, 1/3]'$.

both cases, the misclassification rate is 0 or around 0. The only scenario with a high (maximum) misclassification rate is when data are generated from a Normal copula and misclassified into a t copula. However, part of the reason for this high misclassification is the similarity in shape of these two copulas.

6.4 Choosing Copulas for ETF Data

Example 6.4.1 (Bivariate copula assumption: ETFs dataset.)**.** Historical daily closing prices on two Fidelity exchange-traded funds, namely, Fidelity Growth Company Fund (FDGRX)[3] and Fidelity Value (FDVLX)[4] are available on the Yahoo website. We consider the data from April 30, 2022, to April 30, 2024 (501 days). Daily returns are calculated from closing price X_t as $W_t = \dfrac{X_t - X_{t-1}}{X_{t-1}}$. We wish to choose an appropriate copula that describes the dependence between their daily returns. Which copula is the right one? Scatterplot of the daily return and the empirical copula observations[5] are given in Figures 6.3a and 6.3b, respectively.

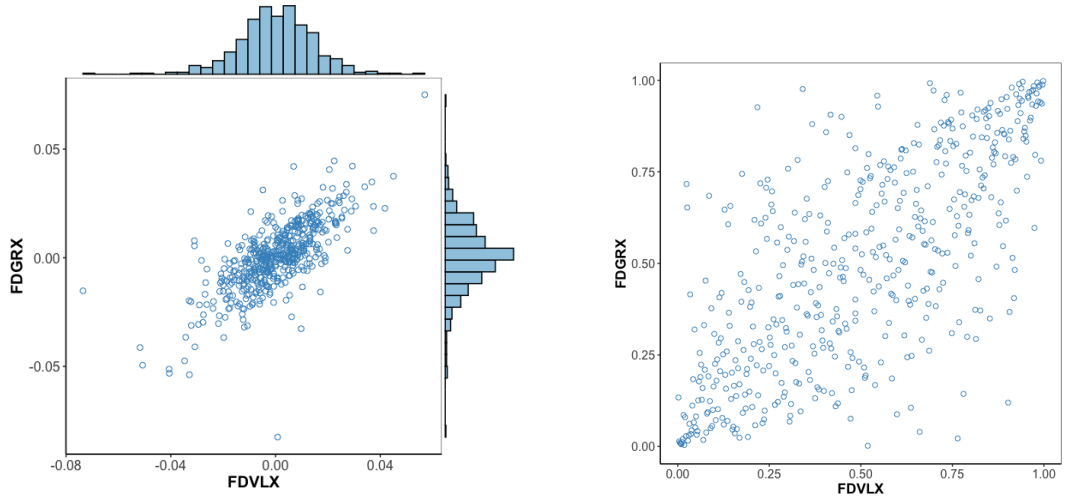
To choose a suitable copula, we will use our eigenstructure-based approach along with its variants and the optimal classification rule described earlier. Given various choices of copulas, we estimate the corresponding parameters to obtain $\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_M$ and then simulate samples $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$ from the corresponding copulas.

As earlier, we consider two families of copulas: the Archimedean family, including Frank, Gumble, Clayton, and Joe copulas, and the Elliptical family, including Normal and t copulas. The maximum likelihood estimators for the parameters corresponding to each of the copulas are Frank($\hat{\alpha} = 6.402$), Gumbel($\hat{\alpha} = 2.08$), Clayton($\hat{\alpha} = 2.287$), Joe($\hat{\alpha} = 2.423$), Normal($\hat{\rho} = .741$) and t($\hat{\rho} = .745, \hat{df} = 4$). With the number of trading days equal to 501, we have the number of daily returns $r = 500$. Simulated sample of size $n = r = 500$ from

[3]https://finance.yahoo.com/quote/FDGRX/

[4]https://finance.yahoo.com/quote/FDGRX/

[5]In the R package *copula*, this is referred to as *pseudo-observation* and is computed via `copula::pobs()`



(a) Stock returns price.

(b) Empirical copula observations.

Figure 6.3: Scatterplots of stock return prices of FDVLX and FDGRX (04/30/2022 - 04/30/2024).

these estimated copulas are plotted in Figure 6.4. Which of these resembles the plot in Figure 6.3b? The answer is not very apparent when visually comparing the plots.

We observed in Section 6.3 that the group classification method using the average of antieigenvalues in each of the three regions, Copulas Rule 2, with a reference sample size n equal to $16r$ gives superior performance for attaining the high correct classification rate. We will adopt this scenario here. Thus, we take $n_1 = n_2 = \cdots = n_6 = 16 \times r = 8000$ and generate a random sample of size 8000 from each of the estimated copulas.

Table 6.6 gives the average of the first antieigenvalues of matrices H_k^j (see (6.1)), $j = 1, 2, 3$ and $k = 1, 2, \ldots, 6$ for various copula candidates. The largest average of antieigenvalues in each region is 0.9999969 corresponding to the $t(\hat{\rho} = .745, \hat{df} = 4)$ copula. The smallest average of antieigenvalues is 0.9998818, corresponding to the Clayton($\hat{\alpha} = 2.287$). The difference between these values is only $0.9999969 - 0.9998818 = 0.0001151$. We observe that

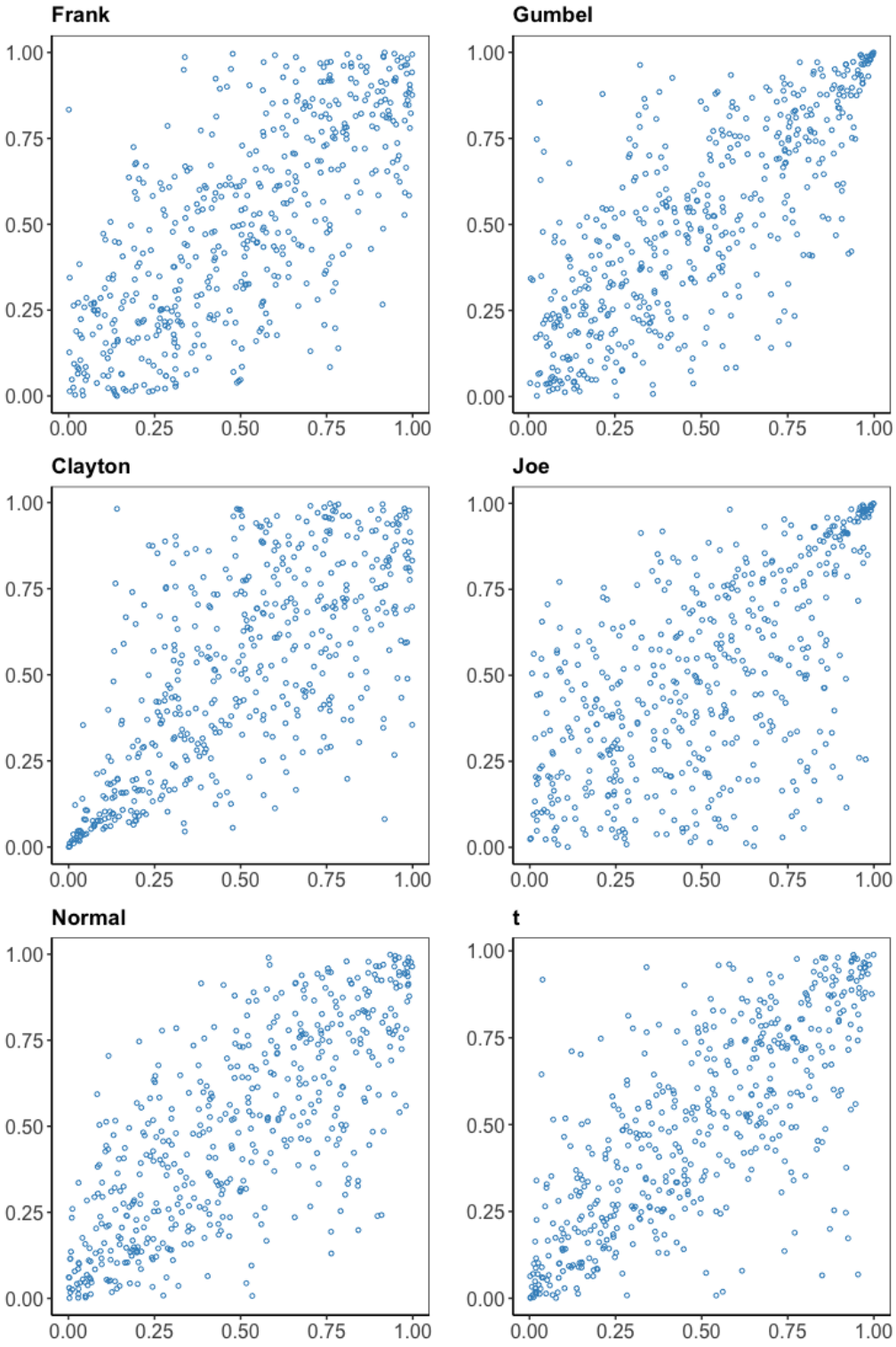


Figure 6.4: Simulated samples from copulas estimated from stock returns of FDVLX and FDGRX, sample size $n = 500$.

Table 6.6 Copulas classification for stock returns of FDVLX and FDGRX (04/30/2022 - 04/30/2024).

Copulas		Rule $2^{w=[1/3,1/3,1/3]'}$ ($n/r = 16$)	Rule $\hat{2}^{(i)}$ ($n/r = 16$)	Rule 3 (ii)
Archimedean	Frank	0.9999775	0.9891702	182.2603
	Gumble	0.9999906	0.9930208	195.5218
	Clayton	0.9998818	0.9713446	138.8645
	Joe	0.9999503	0.9813105	160.0535
Elliptical	Normal	0.9999913	0.9919457	195.0245
	t	0.9999969	0.9952425	208.5971

(i) Values are computed as in (6.2) and are $\hat{\eta}_1^{(k)}$.
(ii) Values are the logarithm of the joint density.

$$\eta_1 = \frac{2\sqrt{\lambda_1\lambda_p}}{\lambda_1 + \lambda_p} = \frac{2}{\sqrt{\lambda_p/\lambda_1} + \frac{1}{\sqrt{\lambda_p/\lambda_1}}},$$

and thus η_1 is a monotonically increasing function of $\sqrt{\lambda_p/\lambda_1}$. Thus it is equally meaningful to look at the average of $\frac{\lambda_p^{(kj)}}{\lambda_1^{(kj)}}$, where $k = 1, 2, \ldots M$ indicates the k^{th} copula, and $j = 1, 2, 3$ indicates the region number (for $[0, .35]$, $(.35, .65]$, $(.65, 1]$, respectively).

Thus, we have

$$\hat{\eta}_1^{(k)} = \frac{1}{3}\left(\frac{\lambda_p^{(k1)}}{\lambda_1^{(k1)}} + \frac{\lambda_p^{(k2)}}{\lambda_1^{(k2)}} + \frac{\lambda_p^{(k3)}}{\lambda_1^{(k3)}}\right), \tag{6.2}$$

where $\lambda_p^{(kj)}$ and $\lambda_1^{(kj)}$ are the smallest and largest eigenvalues of the matrix H_k^j in (6.1), $j = 1, 2, 3$ and $k = 1, 2, \ldots, 6$. Table 6.7 presents all of the relevant information.

The ratio of the smallest and largest eigenvalues is consistently towards the highest side for all three regions in t-copula. The quantity $\hat{\eta}_1^{(k)}$ defined above is highest as well. For the corresponding t copula, we have $\hat{\rho} = .745$ and $\hat{df} = 4$. This is the copula we chose for our bivariate return data.

Again, as in Table 6.6, the first antieigenvalue, $\overline{\eta}_1$, and the ratio of the two largest and smallest eigenvalues, $\hat{\eta}_1$, bring similar results that $t(\hat{\rho} = .745, \hat{df} = 4)$ is the best

Table 6.7 Ratios of eigenvalues of sub-datasets obtained from appending copula observations of the FDVLX- FDGRX dataset to each of the corresponding samples from fitted copulas.

Copulas (k)	Regions (j)			Average ($\hat{\eta}_1^{(k)}$)
	1	2	3	
Frank	0.9784155	0.9959012	0.9931940	0.9891702
Gumbel	0.9872548	0.9922515	0.9995561	0.9930208
Clayton	0.9670661	0.9622887	0.9846788	0.9713446
Joe	0.9761062	0.9776264	0.9901988	0.9813105
Normal	0.9895341	0.9918580	0.9944450	0.9919457
t	0.9935333	0.9951778	0.9970164	0.9952425

model for the correlation structure of FDVLX and FDGRX stocks return. We also use the optimal classification Rule 3 to find the best copula for this bivariate dataset. The estimated density of $\mathcal{U}$ under the t copula is the highest as (= $\exp(208.5971)$). The lowest density is $\exp(138.8645)$ from Clayton ($\hat{\alpha} = 2.287$). See the last column of Table 6.6.

Example 6.4.2 (Multivariate copula assumption: ETFs dataset.)**.** We will now simultaneously model the dependence structure of the daily return from the five ETFs: iShares Russell 1000 Growth ETF (IWF)[6], iShares US Equity Factor (LRGF)[7], SPDR Russell 1000 Momentum Focus (ONEO)[8], ALPS O'Shares US Quality Dividend (OUSA)[9], and ProShares Short QQQ (PSQ)[10]. The dates considered are in the four-year period from April 30, 2020, to April 30, 2024.

As in Example 6.4.1, daily returns in this period are calculated for these historical data. There are 1006 trading days during this period, hence $r = 1006 - 1 = 1005$. Bivariate scatter plots, Spearman's rank correlations, and histograms of the data are given in Figure

[6]https://finance.yahoo.com/quote/IWF
[7]https://finance.yahoo.com/quote/LRGF
[8]https://finance.yahoo.com/quote/ONEO
[9]https://finance.yahoo.com/quote/OUSA
[10]https://finance.yahoo.com/quote/PSQ

6.5. Figure 6.6 shows the plots of empirical copulas for all pairs of two variables at a time. The rank correlations between any two variables are quite different. The return values of the four stocks, IWF, LRGF, ONEO, and OUSA, are all pairwise positively correlated but have opposite trends with PSQ. This is anticipated because PSQ is a "short" Nasdaq index while all other funds are "long". However, to choose a copula is rather impossible.

We assume that the dependence structure of this IWF-LRGF-ONEO-OUSA-PSQ dataset can be modeled by one of the four Archimedean families (Frank, Gumble, Clayton, Joe) or two from elliptical family (Normal or t with both un-structured correlation matrices). The maximum likelihood estimators for the corresponding parameters are: Frank ($\hat{\alpha} = .440$), Gumbel ($\hat{\alpha} = 1.062$), Clayton ($\hat{\alpha} = 0.254$), Joe ($\hat{\alpha} = 1$), Normal ($\hat{\mathbf{\Sigma}}_1$) and t ($\hat{\mathbf{\Sigma}}_2, \hat{df} = 7$), where

$$\hat{\mathbf{\Sigma}}_1 = \begin{bmatrix} 1 & 0.900 & 0.796 & 0.818 & -0.984 \\ 0.900 & 1 & 0.948 & 0.910 & -0.870 \\ 0.796 & 0.948 & 1 & 0.861 & -0.756 \\ 0.818 & 0.910 & 0.861 & 1 & -0.779 \\ -0.984 & -0.870 & -0.756 & -0.779 & 1 \end{bmatrix}$$

and

$$\hat{\mathbf{\Sigma}}_2 = \begin{bmatrix} 1 & 0.904 & 0.801 & 0.814 & -0.986 \\ 0.904 & 1 & 0.947 & 0.907 & -0.876 \\ 0.801 & 0.947 & 1 & 0.855 & -0.763 \\ 0.814 & 0.907 & 0.855 & 1 & -0.776 \\ -0.986 & -0.876 & -0.763 & -0.776 & 1 \end{bmatrix}$$

Copula classification Rule 2 with $w = [1/3, 1/3, 1/3]'$,
$n = 16 \times r = 16 \times 1005 = 16080$, together with its modification Rule $\hat{2}$ (using criteria in

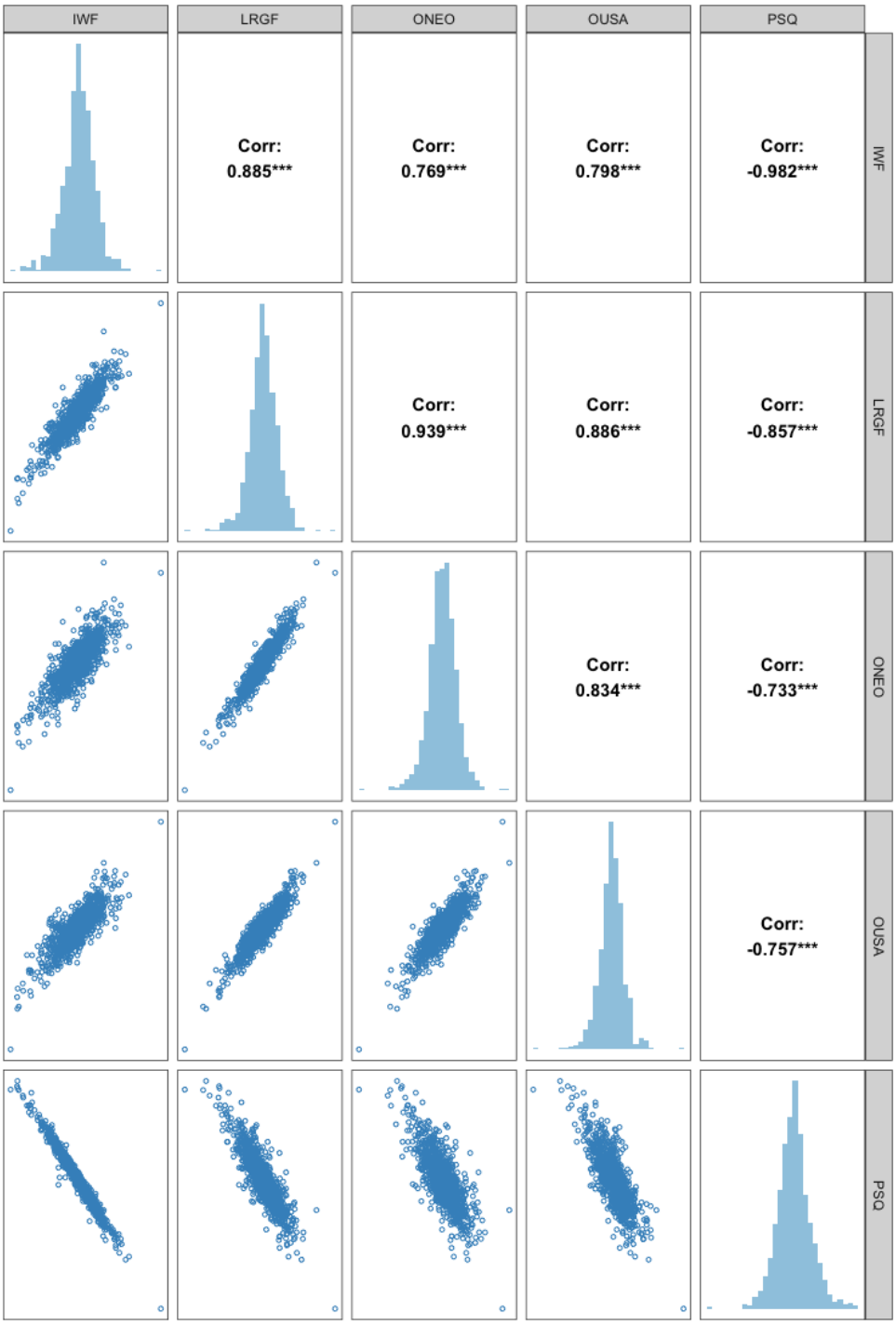


Figure 6.5: Scatterplots of daily return of IWF, LRGF, ONEO, OUSA and PSQ (04/30/2020 - 04/30/2024).

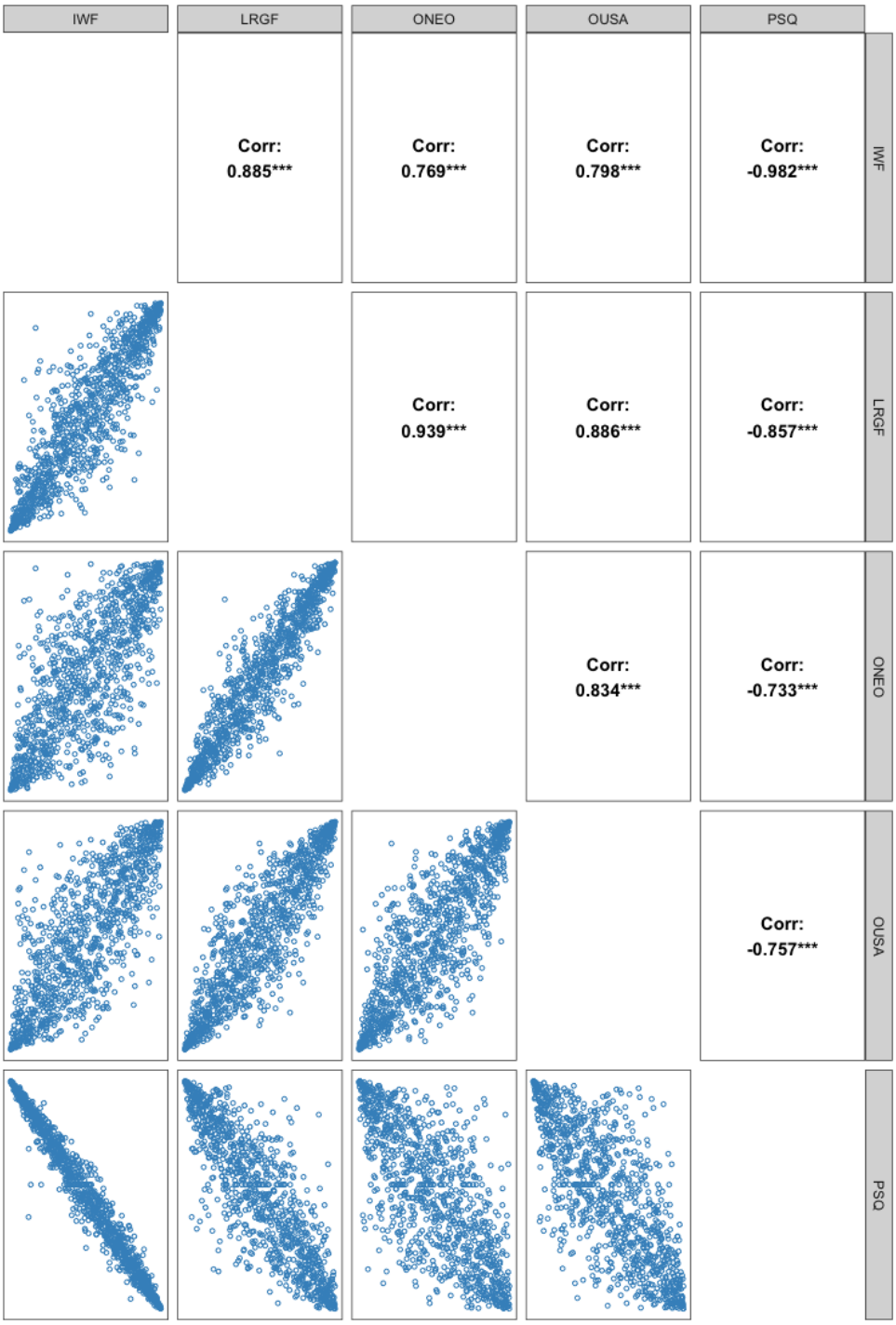


Figure 6.6: Scatterplots of copula observations from the daily return of IWF, LRGF, ONEO, OUSA and PSQ (04/30/2020 - 04/30/2024).

Table 6.8 Copulas choice for stock returns of IWF, LRGF, ONEO, OUSA and PSQ (04/30/2020 - 04/30/2024).

Copulas (k)		Rule $2^{w=[1/3,1/3,1/3]'}$ ($n/r = 16$)	Rule $\hat{2}$ (i) ($n/r = 16$)	Rule 3 (ii)
Archimedean	Frank	0.9942996	0.8298777	12.79060
	Gumble	0.9939547	0.8257266	13.85558
	Clayton	0.9934471	0.8195648	189.5931
	Joe	0.9944277	0.8315588	-3.805890 $\times e^{-6}$
Elliptical	Normal	0.9998749	0.9694647	4735.523
	t	0.9998868	0.9708112	4848.495

(i) Values are computed as in (6.2) and are $\hat{\eta}_1^{(k)}$.
(ii) Values are the logarithm of the joint density.

(6.2)), and Rule 3 will be used to choose the best copula assumption for this five-dimensional dataset.

Antieigenvalues and the logarithm of the joint density are reported in Table 6.8. The conclusions drawn from copula classification rules 2 and $\hat{2}$ (using $\overline{\eta}_1^{(k)}$ and $\hat{\eta}_1^{(k)}$, respectively) are quite similar, as values calculated under the $t(\hat{\mathbf{\Sigma}}, \hat{df} = 7)$ are the highest (0.9998868 and 0.9708112). This indicates a good fit. Moreover, compared to the use of antieigenvalues, using ratios between eigenvalues as a criterion will bring more substantial difference between the best fit, as computed under $t(\hat{\mathbf{\Sigma}}, \hat{df} = 7)$ copula, and the worst fit, as computed under Clayton ($\hat{\alpha} = 0.254$), with the difference is $0.9708112 - 0.8195648 = 0.1512464$.

Table 6.9 shows ratios $\lambda_5^{(kj)}/\lambda_1^{(kj)}$ (as $p = 5$) computed in each region. None of the Archimedean copulas ($k = 1, 2, 3, 4$) fit well. The ratios of eigenvalues range from $\lambda_5^{(33)}/\lambda_1^{(33)} \approx .737$ (Clayton copula) to $\lambda_5^{(11)}/\lambda_1^{(11)} \approx .794$ (Frank copula) in the region $[0, .35)$ and $(.65, 1]$ ($j = 1, 3$). There are higher values of ratios in the region $j = 2$ across all Archimedean copula, but these are still inferior to those corresponding to elliptical copulas. Among the latter two, the t copula appears to have a slight edge over the Normal copula.

Table 6.9 Ratios of eigenvalues of sub-datasets obtained from appending copula observations of the IWF-LRGF -ONEO-OUSA-PSQ dataset to each of the corresponding samples from fitted copulas.

Copulas (k)	Regions (j)			Average ($\hat{\eta}_1^{(k)}$)
	1	2	3	
Frank	0.7942048	0.9435368	0.7518916	0.8298777
Gumbel	0.7902708	0.9427395	0.7441696	0.8257266
Clayton	0.7791311	0.9420565	0.7375070	0.8195648
Joe	0.7986612	0.9433476	0.7526675	0.8315588
Normal	0.9757293	0.9712002	0.9614645	0.9694647
t	0.9772691	0.9703686	0.9647960	0.9708112

The optimal classification Rule 3 (Table 6.8) results in the highest density for the t-copula, $\exp(4848.495)$. This conclusion also agrees with the previous rules based on the eigenstructure. We also notice that density values calculated under Archimedean copulas are extremely low, indicating a very poor fit in all these cases.

6.5 Concluding Remarks

In this chapter, our emphasis has been on copula selection. We pose the problem as group classification. Distance-based methods do not fare well in this context since the means and standard deviations for all marginals for all copulas are the same. However, as shown here, the eigenstructure and the density based approach both give very comparative results, and both can be used as effective tools for choosing a suitable copula for a given dataset.

In the scenarios where, along with the group to be classified, the reference samples $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$ are available, the copula selection by the similarity of eigenstructure is a practical approach. The method is akin to the support vector machine approach. However, since the support vector machine is explicitly distance-based, separation for such data is not

expected to be very good. Consequently, it is not going to be very useful. Our approach, as shown here, is, on the other hand, relatively simple, easy to program, and quite effective.

APPENDIX A

SOURCE CODE FOR R-PACKAGE *PlotNormTest*

A.1 Source code for score plot

Source Code A.1: cox.R

```
## -- Filename: cox.R --#
#' @title Graphical plots to assess the univarite noramality assumption of data.
#' @name Univariate_Score_function
#' @description Score function of a univariate normal distribution is
#' a straight line. A non-linear graph of score function estimator shows
#' evidence of non-normality.
#' @param x univariate data.
#' @param P vector of weight.
#' @param lambda smoothing parameter, default is \eqn{0.5}.
#' @param x.dist the minimum distance between two data points in vector x.
#' @return \code{cox} returns the estimate of score function.
#' \itemize{
#' \item{\code{x}: The updated univariate data if merging happens.}
#' \item{\code{a}: Score value estimated at \code{x}.}
#' \item{\code{P}: Updated weight (if merging happens).}
#' \item{\code{slt}: Index of merged data point
#' (is \code{NULL} if \code{x.dist = NULL}).}
#' }
#' @details
#' To avoid the singularity of coefficient matrices in spline method, points
#' with distance less than \code{x.dist} are merged and weight of the
#' representative points is updated by the summation of weight of
#'discarded points.
#' @export
#' @import MatrixExtra
#' @import Matrix
#' @importFrom Rdpack reprompt
#' @references
#' \insertRef{ref:sp_estimate}{PlotNormTest}
#' @examples
#' set.seed(1)
#' x ←rnorm(100, 2, 4)
#' re ←cox(sort(x))
#' plot(re$x, re$a, xlab = "x", ylab = "Estimated Score",
#' main = "Estimator of score function")
#' abline(0, 1)
#'
cox ←function(x, P = NULL, lambda = 0.5, x.dist = NULL) {
  nx ←length(x)
  if (is.unsorted(x) == T){
    stop("x should be sort")
  }
```

```
if (lambda < 0) {
  stop("penalty parameters is non negative")
}

if (is.null(P)) {
  P ←rep(1/nx, nx)
} else {
  if (length(P) < nx) {
    stop("Incorrect dimensions")
  }
}
x ←(x - mean(x))/stats::sd(x) #standardize data
#merge all close data points and updata x
if (!is.null(x.dist)){
  y ←x[-1] - x[-nx]
  dif ←which(y < x.dist)
  #slt: data point that we skipped
  slt ←split(dif, cumsum(c(1, diff(dif) != 1)))
  if(sum(slt[[1]]) != 0){
    for (i in length(slt):1){
      temp1 ←slt[[i]]
      P[max(temp1) + 1] ←P[max(temp1) + 1] + sum(P[temp1])
      P ←P[-temp1]
      x ←x[-temp1]
    }
  }
  slt ←unlist(slt)
} else {
  slt = NULL
}
nx ←length(x)
onethd ←1/3
twothd ←2/3
ift ←0
nxm1 ←nx - 1; nxm2 ←nx - 2; nxm3 ←nx - 3
nxp1 ←nx + 1
lambda ←lambda
h ←x[-1] - x[-nx]
# Matrix C:
c ←Matrix::bandSparse(nx, nxm2, -1:0, list(twothd *h[-1],
                                           onethd *h[-nxm1]))
c[nx, nxm2] ←-onethd * h[nxm1]
# Store matrix T as w7:
w7 ←Matrix::bandSparse(nxm2, nx, 1:2, list(2*h[-length(h)]/ 3, h[-1]/3))
w7[1, 1] ←-h[1]/3
```

```
  # Matrix Q:
  Q ←Matrix::bandSparse(nx, nxm2, -2: 0, list(1 / h[-1],
                                              - 1/h[-nxm1] - 1/h[-1],
                                              1 / h[-nxm1]))
  # Matrix P^-1Q is pq:
  pq ←Matrix::bandSparse(
    nx, nxm2, -2:0, list(1/(P[-c(1,2)] * h[-1]),
                         1/P[-c(1, nx)] * ( -1/h[-nxm1] - 1/h[-1]),
                         1/(P[-c(nxm1, nx)] * h[-nxm1])))
  # Matrix A is a:
  a ←Matrix::bandSparse(nx, nx, 0: 1, list(-1/c(h, -h[nxm1]) , 1 / h ))
  a[nx, nx -1] ←-1/h[nxm1]
  #Matrix P^-1A' is pa:
  pa ←Matrix::bandSparse(nx, nx, -1:0, list(1/(h * P[-1]),
                                            -1/(P * c(h, -h[nxm1]))))
  pa[nxm1, nx] ←-1/(P[nxm1] * h[nxm1])
  #Matrix R is r:
  r ←Matrix::bandSparse(nxm2, nxm2, (-1): 1,
                        list(h[-c(1, length(h))] / 3,
                             2 / 3 *(h[-1] + h[-length(h)]),
                             h[-c(1, length(h))] / 3))
  #Pseudo y:
  #Create C'P1 store in cp1
  cp ←onethd * (h[-nxm1] * P[-c(nxm1, nx)]) + twothd * (h[-1] * P[-c(1, nx)])
  cp[nxm2] ←cp[nxm2] - onethd * h[nxm1]*P[nx]
  #invert of R:
  invr ←solve(r)
  # Peuso y is in suy
  suy ←pa %*% P - pq%*%t(invr)%*%cp
  #compute R + 2lambda * Q'*P^-1 *Q
  temp ←r + 2*lambda* t(Q) %*% pq
  invtemp ←solve(temp)
  #compute Q'y:
  qy ←suy[-c(nxm1, nx)] * (1/h[-nxm1]) -
    suy[-c(1,nx)] *(1/h[-nxm1] + 1/h[-1]) + suy[-c(1,2)] * (1/h[-1])
  #compute a:
  a ←suy - 2 *lambda *pq %*% invtemp %*% qy
  return(list(x = x, a = a, P= P, slt = slt))
}
```

Source Code A.2: score_plot.R

```
# -- Filename: score_plot.R --#
#' @rdname Univariate_Score_function
#' @description Outliers are detected using the 2-sigma bands method.
```

```
#' @param ori.index original index of vector x, default is \code{NULL}
#' when index is just the order.
#' @return \code{score_plot1D} returns score functions together with
#' 2-sigma bands for outlier detection.
#' \itemize{
#' \item{\code{plot}: plot of estimate score function and its band.}
#' \item{\code{outlier}: index of outliers.}
#' }
#' @details Under null hypothesis, a unbiased estimator score function of a
#' given data point \eqn{x_k} is
#' \deqn{
#' \hat{\psi}(x_k) = \dfrac{n - 4}{n - 2} \dfrac{x_k - \bar{X}_{-k}}{S_{-k}^2}
#' }
#' and if \eqn{a_{k}} is the estimate score from function \code{cox} at
#' the point \eqn{x_k}, then
#' \deqn{a_k\in \hat{\psi}(x_k) \pm 2 \sqrt{\hat{\text{Var}}(\hat{\psi}(x_k))}.}
#' Hence points outside the 2-sigma bands are outliers.
#' @import ggplot2
#' @importFrom stats weighted.mean
#' @export
#' @examples
#' set.seed(1)
#' x <-rnorm(100, 2, 4)
#' score_plot1D(sort(x))
#'
score_plot1D <-function(x, P = NULL,
                        lambda = .5, x.dist = NULL,ori.index = NULL){
  nx <-length(x)
  if (is.unsorted(x) == T){
    stop("x should be sort")
  }
  if(is.null(ori.index)){
    ori.index <-1:nx
  } else{
    if (length(ori.index) != nx){
      stop("Uncomformable length of index and vector")
    }
  }
  if (is.null(P)) {
    P <-rep(1/nx, nx)
  } else {
    if (length(P) < nx) {
      stop("Incorrect dimensions")
    }
  }
```

```
index ←ori.index

recox ←cox(x, P = P, lambda = lambda, x.dist = x.dist)
a ←as.numeric(unlist(recox$a))
x ←as.numeric(unlist(recox$x)) #already standardized
weight ←as.numeric(unlist(recox$P))
slt ←as.numeric(unlist(recox$slt))

if(length(slt) != 0){
  index ←index[-slt]
}
nx ←length(x)
f ←c()
upper ←c()
lower ←c()
for (i in 1: nx){
  x.temp ←x[-i]
  weight.temp ←weight[-i]
  xbar ←stats::weighted.mean(x.temp, weight.temp)
  varx ←sum(weight.temp * (x.temp - xbar)^2)
  fi ←((nx - 4)/ (nx - 2))* ((x[i] - xbar)/varx)
  bi ←2 *(nx - 4)/((nx - 2)^2) * fi^2 + (nx - 4)/((nx - 1)*(nx-2)*varx)
  upperi ←fi + 2*sqrt(bi)
  loweri ←fi - 2*sqrt(bi)
  f ←c(f, fi)
  upper ←c(upper, upperi)
  lower ←c(lower, loweri)
}
df ←data.frame(x = x, a = a, f= f, upper = upper, lower = lower)
ind ←which(df$a <= df$lower | df$a >= df$upper)
out ←index[ind]
df$Type ←as.factor(
  ifelse(df$a < lower|df$a > upper, "Outlier", "Non outlier")
)
g ←ggplot2::ggplot(data = df)+
  geom_line(mapping = aes(x, f), colour = "darkorange3" , linewidth = 0.7) +
  ggplot2::geom_line(
    mapping = aes(x, upper), linetype = "twodash",
    linewidth = 0.7, colour = "darkorange3" ) +
  ggplot2::geom_line(
    mapping = aes(x, lower), linetype = "twodash",
    linewidth = 0.7, colour = "darkorange3" ) +
  ggplot2::geom_point(aes(x, a, color = Type, shape = Type, size = Type)) +
  ggplot2::scale_shape_manual(values=c(1, 17)) +
```

```
    ggplot2::scale_color_manual(values = c("steelblue3","seagreen")) +
    ggplot2::scale_size_manual(values = c(1.5, 2)) +
    ggplot2::labs(y = expression(hat(psi))) + ggplot2::coord_fixed() +
    ggplot2::theme_bw() +
    ggplot2::theme(
      aspect.ratio = 1/1, panel.grid = element_blank(),
      axis.line = element_line(colour = "black"),
      axis.text=element_text(size=12),
      axis.title=element_text(size=14,face="bold"),
      legend.background = element_rect(
        size=0.5, linetype="solid"),
      legend.text = element_text(size=12)) + ylab("Score function") +
    ggplot2::theme(legend.position="top")
  return(list(plot= g, outlier = unique(out)))
}
```

A.2 Source code for method of "$n \times p$" samples

Source Code A.3: Multi_Uni.R

```
# -- Filename: Multi_Uni.R -----#
#' @title Transformation to Independent Univariate Sample
#' @name Independent_transformation
#' @description Leave-one-out method gives approximately independent sample of
#' standard multivariate normal distribution,
#' which then produces sample of standard univariate normal distribution.
#' @param x multivariate data matrix
#' @return Data frame contains univariate data and
#' the index from multivariate data.
#' @details
#' Let \eqn{\bar{X}_{-k} } and \eqn{S_{-k}} are the sample mean sample variance
#' covariance matrix obtained by using all but \eqn{k^{th}} data point. Then
#' \eqn{S_{-k}^{-1/2} (X_k - \bar{X}_{-k}) , k = 1,... n} are approximately
#' independently distributed as \eqn{N_p(0, I)}. Thus all \eqn{n \times p}
#' entries in the data matrix so constructed can be treated as
#' univariate samples of size \eqn{n \time p} from \eqn{N(0, 1)}.
# @note The transformation gives number of data points as a multiple of
# dimension \eqn{p}, density of data points is high and
# spline method may produce ill-posed matrices. Merging with \code{x.dist}
# parameter is recommended.
#' @export
#' @examples
#' set.seed(1)
#' x <-MASS::mvrnorm(100, mu = rep(0, 5), diag(5))
#' df <-Multi.to.Uni(x)
```

```
#' qqnorm(df$x.new); abline(0, 1)
Multi.to.Uni <-function(x){
  nx <-nrow(x)
  p <-ncol(x)
  xbar <-colMeans(x)
  S <-stats::cov(x)
  inv.S <-solve(S)
  temp <-((nx - 1)**2)/nx
  ###############
  inv.Si <-function(i){
    #function calculate inverse S(-k)
    top <-inv.S %*% (x[i,] - xbar) %*% t((x[i, ] - xbar)) %*% inv.S
    bottom <-temp - t((x[i, ] - xbar)) %*% inv.S %*% (x[i, ] - xbar)
    inv.temp <-inv.S + top/sum(bottom)
    inv.temp <-(nx - 2) / (nx - 1) *inv.temp
    return(inv.temp)
  }
  x.new <-c()
  ###############
  for (i in 1: nx){
    xbari <-colMeans(x[-i, ])
    inv.S.not.i <-inv.Si(i)
    a <-chol(inv.S.not.i)
    xi <-a %*% (x[i,] - xbari)
    x.new <-c(x.new, xi)
  }
  ind <-kronecker(c(1:nx), rep(1, p))
  df <-data.frame(x.new, ind)
  df <-df[order(df$x.new),]
  return(newdata = df)
}
```

A.3 Source code for "best" linear transformations method

Source Code A.4: linear_transform.R

```
# -- Filename: maximum_skewness.R --
#' @name skewness_kurtosis
#' @details Sample kurtosis is
#' \deqn{
#' \hat{\kappa}_4 =
#' \dfrac{1}{n-1} \sum_{i = 1}^n \left(\dfrac{X_i - \bar{X}}{S}\right)^4.
#' }
#' @return \code{kurtosis} returns sample kurtosis.
#' @export
```

```
#' @examples
#' set.seed(123)
#' y <-rnorm(100)
#' kurtosis(y)
kurtosis <-function(x){
  nx <-length(x)
  sdx <-stats::sd(x)
  y <-(x - mean(x))/sdx
  k <-sum(y^4)
  k <-1/(nx-1) * k
  return(k - 3)
}
#' @rdname skewness_kurtosis
#' @title Sample skewness and Sample Kurtosis.
#' @param x univariate data sample
#' @details
#' Sample skewness is
#' \deqn{
#' \hat{\kappa}_3 =
#' \dfrac{1}{n-1} \sum_{i = 1}^n \left(\dfrac{X_i - \bar{X}}{S}\right)^3.
#' }
#' @return \code{skewness} returns sample skewness.
#' @export
#' @examples
#' set.seed(123)
#' x <-rnorm(100)
#' skewness(x)
skewness <-function(x) {
  nx <-length(x)
  sdx <-stats::sd(x)
  y <-(x - mean(x))/sdx
  k <-sum(y^3)
  k <-1/(nx - 1) * k
  return(k)
}

################################
######## max k^2 ###############
# -- Filename: maximum_skewness.R --
#' @title Best Linear Transformations
#' @name linear_transform
#' @description
#' The algorithm uses gradient descent algorithm to obtain the maximum of the
#' square of sample skewness, of the kurtosis or of their average under any
#' univariate linear transformation of the multivariate data.
```

```
# Departure from 0 of these values is an indication of non-normality.
#'
#' @param x multivariate data matrix.
#' @param l0 starting point for projection algorithm,
#' default is \code{rep(1, ncol(x))}.
#' @param method character strings,
#' one of \code{c("skewness", "kurtosis", "both")}.
#' @param epsilon bounds on error of optimal solution, default is \code{1e-10}.
#' @param iter number of iteration of projection algorithm,
#' default is \code{5000}.
#' @param stepsize gradient descent stepsize, default is \code{.001}.
# @details
# The algorithm looks for vector \eqn{l \in \mathbb{R}^p} that maximizes the
# univariate skewness, kurtosis, average of these both under any possible linear
# transformation.
#' @return
#' \itemize{
#' \item{\code{max_result}: The maximum value after linear transformation.}
#' \item{\code{x_uni}: Univariate data after transformation.}
#' \item{\code{vector_k}: Vector of the "best" linear transformation.}
#' \item{\code{error}: Error of projection algorithm.}
#' \item{\code{iteration}: Number of iteration. }
#' }
#' @seealso \code{\link[=skewness]{skewness()}},
#' \code{\link[=kurtosis]{kurtosis()}}
#' @export
#' @examples
#' set.seed(1)
#' x <-MASS::mvrnorm(100, mu = rep(0, 2), diag(2))
#' linear_transform(x, method = "skewness")$max_result
#' linear_transform(x, method = "kurtosis")$max_result
#' linear_transform(x, method = "both")$max_result

######## All method ##########
linear_transform <-function(x, l0 = rep(1, ncol(x)), method = "both",
                            epsilon = 1e-10, iter = 5e3, stepsize = 1e-3){
  nx <-nrow(x)
  p <-ncol(x)

  if (epsilon <= 0){
    stop("error should be positive number")
  }
  # if (sum(t(l0) %*% l0) != 1){
  # stop(sQuote("l0"), "must has norm of 1")
  # }
```

```
if (length(l0) != p){
  stop(sQuote("l0"), "wrong dimension")
}
if (iter < 0 || iter != round(iter) ){
  stop(sQuote("iter"), "must be positive integer")
}
if (stepsize < 0){
  stop(sQuote("stepsize"), "must be positive")
}
#### Data standardization ##################
l0 <-l0 / sqrt(sum(t(l0) %*% l0))
S <-stats::cov(x)
m <-matrix(rep(colMeans(x), nx), byrow = T, ncol = p)
chol.S <-chol(S) # 50% faster
A <-backsolve(r = chol.S, x = diag(p))
x <-(x - m) %*% A # check var(x) is an identity matrix
#### Choosing method #################
if (method == "skewness"){
  fun <-function(y) skewness(y)^2
  gradG <-function(l, x){
    xl <-(x %*% l)
    v <-3 * rowSums(sapply(1:nx, function(i) xl[i]^2 * x[i, ]))
    return(v)
  }
} else{
  if (method == "kurtosis"){
    fun <-function(y) kurtosis(y)^2
    gradG <-function(l, x){
      xl <-(x %*% l)
      v <--2 * (mean(xl^4) - 3) *4*rowMeans(
        sapply(1:nx, function(i) xl[i]^3 * x[i, ])
      )
      return(v)
    }
  } else {
    fun <-function(y) .5*(skewness(y)^2 + kurtosis(y)^2)
    gradG <-function(l, x){
      xl <-x %*% l
      v1 <-(mean(xl^4) - 3) * 4 * rowMeans(
        sapply(1:nx, function(i) xl[i]^3 * x[i, ])
      )
      v2 <-(mean(xl^3)) * 3 * rowMeans(
        sapply(1:nx, function(i) xl[i]^2 * x[i, ])
      )
      v <--(v1+v2)
```

```
      return(v)
    }
  }
}

lk ←l0; err ←1; i ←1;
while (i <= iter && err >= epsilon ) {
  gradG.lk ←gradG(l = lk, x = x)
  gamma ←lk - stepsize * gradG.lk
  lkp1 ←gamma / sqrt(sum(gamma^2))
  # err ←sqrt(t(lk - lkp1) %*% (lk - lkp1))
  err ←sqrt(sum((lk - lkp1)^2))
  i ←i + 1
  lk ←lkp1
}

x.new ←x %*% lk
re ←fun(x.new) / p #scale by p
return(list(max_result = re, x_new = x.new, l = lk,
           error = err, iteration = i))


}
```

A.4 Source code for method of using cumulant generating function

Source Code A.5: d_hCGF.R

```
# --- Filename: d_hCGF.R ---#
#' @title Calculation of derivatives of empirical
#' cumulant generating function (CGF).
#' @name dCGF
#' @description
#' Get the third/fortth derivatives of sample CGF at a given point.
#' @details
#' Estimator of standardized cumulant function is
#' \deqn{ \log\hat{M}_X(t) = \log \left(\dfrac{1}{n}
#' \sum_{i = 1}^n \exp(t'S^{\frac{-1}{2}}(X_i - \bar{X})) \right)
#' }
#' and its \deqn{k^{th}} order derivatives is defined as
#' \deqn{
#' T_k(t) = \dfrac{\partial^k}{
#' \partial t_{j_1}t_{j_2} \dots t_{j_k}} \log(\hat{M}_X(t)), t \in \mathbb{R}^p
#' }
#' where \eqn{t_{j_1}t_{j_2} \dots t_{j_k}} are the corresponding components
```

```
#' of vector \eqn{t \in \mathbb{R}^p}.
# The number of distinct third derivatives is:
# \deqn{
# l_T = p + 2 \times \begin{pmatrix}
# p\\2
# \end{pmatrix} + \begin{pmatrix}
# p \\ 3
# \end{pmatrix}
# }
#' @param myt,t numeric vector of length \code{p}.
#' @param x data matrix.
#' @return \code{d3hCGF} returns the sequence of third derivatives of
#' empirical CGF, ordered by index of \eqn{j_1 \leq j_2 \leq j_3 \leq p}.
#' @export
#' @examples
#' p <-3
#' # Number of distinct derivatives
#' l_dhCGF(p)
#' set.seed(1)
#' x <-MASS::mvrnorm(100, rep(0, p), diag(p))
#' myt <-rep(.2, p)
#' d3hCGF(myt = myt, x = x)
#' d4hCGF(myt = myt, x = x)
d3hCGF <-function(myt, x){
  nx <-nrow(x)
  p <-ncol(x)
  #standardize data: not using Cholesky decomposition
  S <-stats::cov(x)
  xbar <-colMeans(x)
  m <-matrix(rep(xbar, nx), byrow = T, ncol = p)
  x <-x - m
  chol.S <-chol(S) # 50% faster
  A <-backsolve(r = chol.S, x = diag(p))
  x <-x %*% A # check var(x) is an identity matrix
  ##############################################
  pi <-c(exp(x %*% myt))
  pbar <-1/sum(exp(x %*% myt))
  pi <-pbar*pi
  m1 <-colSums(x*pi)
  m2 <-t(x*pi)%*%x
  t3 <-lapply( 1:p, function(k){
    temp2 <-m1 %*% t(m2[k, ]);
    t(x*pi*x[,k])%*%x - m1[k] *m2 -temp2 - t(temp2) +2* (m1[k]) * (m1) %*% t(m1)
    # t(x*pi*x[,k])%*%x - m1[k] *m2 -temp2 - t(temp2) + (m1[k]) * (m1) %*% t(m1)
  })
```

```
  v3 ←c()
  for (i in 1:p){
    t3i ←t3[[i]][i:p, i:p]
    t3i ←t3i[upper.tri(t3i, diag= T)]
    v3 ←c(v3, c(t3i))
  }
  return(v3)
}
#######################################################
#######################################################
#' @rdname dCGF
#' @return \code{d4hCGF} returns the sequence of fourth derivatives of empirical
#' CGF ordered by index of \eqn{j_1 \leq j_2 \leq j_3 \leq j_4 \leq p}.
#' @export
d4hCGF ←function(myt, x){
  nx ←nrow(x)
  p ←ncol(x)
  #standardize data: not using Cholesky decomposition
  S ←stats::cov(x)
  xbar ←colMeans(x)
  m ←matrix(rep(xbar, nx), byrow = T, ncol = p)
  x ←x - m
  if (kappa(S) > 1e13){
    warning("Non-invertable covariance matrix")
  } else {
    chol.S ←chol(S) # 50% faster
    A ←backsolve(r = chol.S, x = diag(p))
    x ←x %*% A # check var(x) is an identity matrix
  }
  ###############################################
  pi ←c(exp(x %*% myt))
  pbar ←1/sum(exp(x %*% myt))
  pi ←pbar*pi
  v4 ←c()
  for (j1 in (1:p)){
    for (j2 in (j1:p)){
      for (j3 in (j2:p)){
        for (j4 in (j3:p)){
          s0 ←sum(pi *x[, j1]*x[, j2]*x[,j3] *x[,j4])

          sj1 ←pi * x[,j1]; s.sj1 ←sum(sj1)
          sj2 ←pi * x[,j2]; s.sj2 ←sum(sj2)
          sj3 ←pi * x[,j3]; s.sj3 ←sum(sj3)
          sj4 ←pi * x[,j4]; s.sj4 ←sum(sj4)
```

```
          sj1j2 <-sj1*x[, j2]; s.sj1j2 <-sum(sj1j2)
          sj1j3 <-sj1*x[, j3]; s.sj1j3 <-sum(sj1j3)
          sj1j4 <-sj1*x[, j4]; s.sj1j4 <-sum(sj1j4)
          sj2j3 <-sj2*x[, j3]; s.sj2j3 <-sum(sj2j3)
          sj2j4 <-sj2*x[, j4]; s.sj2j4 <-sum(sj2j4)
          sj3j4 <-sj3*x[, j4]; s.sj3j4 <-sum(sj3j4)

          s.sj1j2j3 <-sum(pi *x[, j1]*x[, j2]*x[,j3])

          s_j4 <-s.sj1j2j3*s.sj4
          s_j3 <-sum(pi *x[, j1]*x[, j2]*x[,j4])*s.sj3
          s_j2 <-sum(pi *x[, j1]*x[, j3]*x[,j4])*s.sj2
          s_j1 <-sum(pi *x[, j2]*x[, j3]*x[,j4])*s.sj1


          tmp1 <-s.sj1j2 *s.sj3 * s.sj4 +
            s.sj1j3 * s.sj2 * s.sj4 +
            s.sj1j4 * s.sj3 * s.sj2 +
            s.sj2j3 * s.sj1 * s.sj4 +
            s.sj2j4 * s.sj3 * s.sj1 +
            s.sj3j4 * s.sj1 * s.sj2

          tmp2 <-s.sj1j2 *s.sj3j4 +
            s.sj1j3 * s.sj2j4 +
            s.sj2j3 * s.sj1j4

          tmp3 <-s.sj1*s.sj2*s.sj3*s.sj4

          s4 <-s0 - s_j4 - s_j3 - s_j2 - s_j1 + 2*tmp1 - tmp2 - 6*tmp3
          v4 <-c(v4, s4)
        }
      }
    }
  }
  return(v4)
}
###################################
#' @rdname dCGF
#' @param p Dimension.
#' @return \code{l_dhCGF} returns number of distinct third and
#' fourth derivatives.
#' @export
l_dhCGF <-function(p){
   l3 <-p + choose(p, 2)*2 + choose(p, 3)
   l4 <-p + 3 *choose(p, 2) + 3*choose(p, 3) + choose(p, 4)
```

```
   return(list("Third derivatives" = l3, "fourth derivatives" = l4))
}
############################
############################
#' @rdname dCGF
#' @return \code{dhCGF1D} returns third/fourth derivatives of univariate
#' empirical CGF, which are \code{d3hCGF} and \code{d4hCGF} when \eqn{p = 1}.
#' @export
#' @examples
#' #Univariate data
#' set.seed(1)
#' x <-rnorm(100)
#' t <-.3
#' dhCGF1D(t, x)
dhCGF1D <-function(t, x){
  z <-(x - mean(x))/stats::sd(x)
  nz <-length(z)
  pi <-c(exp(z*t))
  pbar <-1/sum(exp(z * t))
  pi <-pbar*pi

  m1 <-sum(z*pi)
  m2 <-sum(z^2 * pi)
  m3 <-sum(z^3 *pi)
  m4 <-sum(z^4 * pi)

  t3 <-m3 - 3*m2*m1 + 2*m1^3
  t4 <-m4 - 4*m3 *m1 - 3*m2^2 + 12 *m2*m1^2 - 6 *m1^4

  return(list(t3 = t3, t4 = t4))
}
```

Source Code A.6: dhCGF_plot1D.R

```
## -- Filename: dhCGF_plot1D.R
#' @title Graphical plots to assess multivariate univarite assumption of data.
#' @description
#' Plots the empirical third/fourth derivatives of cumulant generating function
#' together with confidence probability region.
#' Indication of non-normality is either violation of probability bands or
#' curves with high slope.
#' @name Univariate_CGF_plot
#' @param x Univariate data
#' @param alpha Significant level (default is \eqn{.05})
#' @param method string, \code{"T3"} used the third derivatives,
```

```
#' and \code{"T4"} uses the fourth derivatives.
#' @return Plots
#' @export
#' @references
#' \insertRef{ref:ghoshuni}{PlotNormTest}
#' @examples
#' set.seed(123)
#' x <-rnorm(100)
#' dhCGF_plot1D(x, method = "T3")
#' dhCGF_plot1D(x, method = "T4")
#'
dhCGF_plot1D <-function(x, alpha = 0.05, method){
  if (!(method %in% c("T3", "T4"))){
    stop("Method is either T3 or T4")
  }
  bigt = seq(-1, 1, by = 0.05)

  # load("data/mt3_lst_param.RData")
  # load("data/mt4_lst_param.RData")

  sT3 <-c(
    92.42158217, 74.73877805, 60.7302487, 49.59500711, 40.7146832,
    33.61041073, 27.91023887, 23.3244267, 19.62665994, 16.63973104,
    14.22459407, 12.2719798, 10.69596014, 9.429002869, 8.418171,
    7.622206539, 7.00930269, 6.555417571, 6.243020088, 6.060187813,
    6
  )
  sT3 <-c(sT3, rev(sT3[-21]))
  # sT3 <-diag(mt3_lst_param$'1'$l.sTtTs)
  m.supLt3 <-1.453986705

  sT4 <-c(568.1209021, 448.0515896, 355.0182299, 282.6795124, 226.2374173,
          182.0500981, 147.3446669, 120.0034577, 98.40450305, 81.30213209,
          67.73735645, 56.97044429, 48.4300794, 41.67496479, 36.3648053,
          32.23839726, 29.09714181, 26.79273777, 25.2181407, 24.30112719,
          24)

  sT4 <-c(sT4, rev(sT4[-21]))

  m.supLt4 <-1.478985651

  nx <-length(x)
  re <-lapply(bigt, dhCGF1D, x = x)
  T3 <-sqrt(nx)* unlist(lapply(re, function(i) i$t3))
  T4 <-sqrt(nx)* unlist(lapply(re, function(i) i$t4))
```

```
  u ←sqrt(-2 *log(alpha))
  bandt4 ←(u + m.supLt4)*sqrt(unlist(sT4))
  bandt3 ←(u + m.supLt3)*sqrt(unlist(sT3))

  ylimt3 ←max(40, abs(T3))
  ylimt4 ←max(95, abs(T4))

  if (method == "T3"){
    plot(bigt, T3, type = "l", lty = 1, lwd = 2, col = "blue",
         xlab = "t",
         ylab = bquote(T[3]),
         ylim = c(-ylimt3, ylimt3)
    )
    graphics::lines(bigt, -bandt3, lty = 6, lwd = 2, col = "darkorange")
    graphics::lines(bigt, bandt3, lty = 6, lwd = 2, col = "darkorange")
    graphics::legend("top", c("T3","Probability bands"),
           lwd = c(2, 2),
           lty = c(1, 6), col = c("blue", "darkorange"),
           merge = TRUE, y.intersp = 2,text.width = .6)
    graphics::title(main = bquote(T[3] ~"plot"),
          adj = 0)
  } else {
    plot(bigt, T4, type = "l", lty = 1, lwd = 2, col = "blue",
         xlab = "t", ylab = bquote(T[4]),
         ylim = c(-ylimt4, ylimt4)
         )
    graphics::lines(bigt, bandt4, lty = 6, lwd = 2, col = "darkorange")
    graphics::lines(bigt, -bandt4, lty = 6, lwd = 2, col = "darkorange")
    graphics::legend("top", c("T4","Probability bands"),
           lwd = c(2, 2),
           lty = c(1, 6), col = c("blue","orange"),
           merge = TRUE, y.intersp = 2, text.width = .6)
    graphics::title(main = bquote(T[4] ~"plot"),
          adj = 0)
  }

}
```

Source Code A.7: dMGF.R

```
## -- Filename: hCGF.R ---#
#' @title Moment generating functions (MGF) of standard normal distribution.
#' @name dMGF
#' @description
```

```
#' Get the polynomial term in the expression of derivatives of moment
#' generating function of \eqn{N_p(0, I_p)}, with
#' respect to a given component and its exponent. Up to eighth order.
#'
#' @param tab a dataframe with the first column contain indices of components
#' of a multivariate random vector \eqn{\bold{X}}, and the second column is the
#' order derivatives with respect to that components.
#' @param t vector in \eqn{\mathbb{R}^p}.
#' @param coef take \code{TRUE} or \code{FALSE} value to
#' obtain only polynomial or whole expression by multiplying the
#' polynomial term with the exponent term \eqn{\exp(.5 t't)}.
#' @return Value of derivatives.
#' @details
#' For a standard multivariate normal random variables \eqn{Y \sim N_p(0, I_p)}
#' \deqn{
#' \mathbb{E}\left(Y_1^{k_1} ... Y_p^{k_p} \exp(t'X)\right) =
#' \dfrac{\partial^{k_1}\dots
#' \partial^{k_p}}{t_1^{k_1} \dots t_p^{k_p}} \exp(t't/2) =
#' \mu^{(k_1)} (t_1) ... \mu^{(k_p)}(t_p) \exp(t't/2)
#' }
#' For example,
#' \eqn{
#' \mathbb{E}Y_2^4 \exp(t'Y) = \dfrac{\partial^4}{\partial t_2^4} \exp(t't/2)
#' = \mu^{(4)}(t_2) \exp(t't/2).
#' }
#' @examples
#' #Calculation of above example
#' t <-rep(.2, 7)
#' tab <-data.frame(j = 2, exponent = 4)
#' dMGF(tab, t = t)
#' dMGF(tab, t = t, coef = FALSE)
#'
#' @export
dMGF <-function(tab, t, coef = TRUE){ #upto j.num == 8
  j <-as.numeric(tab[1])
  j.num <-tab[2]
  if (j == 0){
    mu <-1
  } else {
    s <-t[j]
    if (j.num == 1){
      mu <-s
    } else if (j.num == 2) {
      mu <-(1 + s^2)
    } else if (j.num == 3){
```

```
    mu ←(3*s + s^3)
   } else if (j.num == 4){
    mu ←(3 + 6*s^2 + s^4)
   } else if (j.num == 5){
    mu ←(15*s + 10*s^3 + s^5)
   } else if (j.num == 6) {
    mu ←(15 + 45*s^2 + 15*s^4 + s^6)
   } else if (j.num == 7){
    mu ←105*s + 105*s^3 + 21*s^5 + s^7
   } else {
    mu ←105 + 420*s^2 + 210 * s^4 + 28*s^6 + s^8
   }
  }
  mu ←ifelse(coef, mu, mu * exp(sum(t*t)/2))
  return(mu)
}
```

Source Code A.8: pos.R

```
# -- Filname: pos.R --#
#' @rdname CGF_transformations
#' @title Derivatives of empirical moment generating function (MGF).
# @description
# Given the position in the chain containg all derivatives of estimator
# \eqn{\hat{M}_X(t)}, get the information
# of which derivatives were taken.
#' @param j1 Index of the first variables
#' @param j2 Index of the first variables, should be at least \code{j1}
#' @param j3 Index of the first variables, should be at least \code{j2}
#' @param p Dimension
#' @return \code{mt3_rev_pos} returns the position of this particular derivative
#' in the chain of all derivatives, up to third order.
#' @details
#' The estimator of multivariate moment generating function is
#' \eqn{\hat{M}_X(t) = \dfrac{1}{n} \sum_{i = 1}^n \exp(t'X_i)}
#' The chain containing all derivatives up to the third order is
#' \deqn{
#' Z = \bigg(\hat{M}, \hat{M}^{001}, \dots \hat{M}^{00p},
#' \hat{M}^{011}, \hat{M}^{012}, \dots \hat{M}^{0pp}, \hat{M}^{111},
#' \hat{M}^{112}, \dots \hat{M}^{ppp}\bigg)'
#' }
#' and
#' \deqn{
#' \hat{M} = \hat{M}^{000}(t)= \hat{M}_X(t)
#' }
```

```
#' \deqn{
#' \hat{M}^{j_1j_2j_3}(t) =
#' \dfrac{\partial^k}{\partial t_{j_1} t_{j_2} t_{j_3}} \hat{M}(t)
#' }
#' where \eqn{k} is the number of \eqn{j_1, j_2, j_3} different from 0.
#' Similar notation is applied when fourth derivatives is used.
#' @export
#' @examples
#' mt3_rev_pos(1, 2, 2, p = 3)
#' p <-3
#' mt3_pos(p)
#' mt4_pos(p)
mt3_rev_pos <-function(j1, j2, j3, p){
  if (is.unsorted(c(j1, j2, j3))){
    stop("Index should be is descending order")
  }
  pos <-0
  if (j1 >= 1){
    for (t in 0:(j1-1)){
      for (l in 0:(p-1-t)){
        pos <-pos + (p+1 - t-l)
      }
    }
  }
  pos <-pos + j1
  if (j1 <= (j2 - 1)){
    for (l in ((j1):(j2 - 1))){

      pos <-pos + p + 1 - l
    }
  }
  pos <-pos + j3 - (j2 -1)
  return(pos)
}
###################################
###################################
#' @rdname CGF_transformations
#' @description
#' Given dimension \eqn{p}, returns a dataframe containing the position of
#' all derivatives of
#' estimator of moment generating function \eqn{\hat{M}_X(t)},
#' upto third/fourth order.
# Reverse of \code{\link[=rev_pos]{rev_pos()}}.
#' @param p Dimension
#' @return \code{mt3_pos} an array contaning all position with respect
```

```
#' to index of \eqn{j_1, j_2, j_3}.
# @seealso \code{\link[=rev_pos]{rev_pos()}}
#' @export
mt3_pos ←function(p){
  j1j2j3 ←c()
  for (j1 in 0:p){
    for (j2 in j1:p){
      for (j3 in j2:p){
        j1j2j3 ←rbind(j1j2j3, c(j1, j2, j3))
      }
    }
  }
  return(j1j2j3)
}
##################################
##################################
#' @rdname CGF_transformations
#' @return \code{mt4_pos} an array contaning all position with respect to
#' the index of \eqn{j_1, j_2, j_3, j_4}.
#' @export
mt4_pos ←function(p){
  j1j2j3j4 ←c()
  for (j1 in 0:p){
    for (j2 in j1:p){
      for (j3 in j2:p){
        for (j4 in j3:p){
          j1j2j3j4 ←rbind(j1j2j3j4, c(j1, j2, j3, j4))
        }

      }
    }
  }
  return(j1j2j3j4)
}
```

Source Code A.9: covZtZs.R

```
## --Filename: covZtZs.R
#' @name covZtZs
#' @title Covariance matrix of derivatives of sample moment
#' generating function (MGF).
#' @description
#' Stacking derivatives upto the third/fourth orders of sample MGF
#' together to obtain a vector, which (under normality assumption) approaches
#' a multivariate normally distributed vector
```

```
#' with zero mean and a covariance matrix.
#' \code{covZtZs} calculates covariance between any two points
#' \eqn{t} and \eqn{s} in \eqn{\mathbb{R}^p}.
#' @param t,s a vector of length \eqn{p}.
# @param s a vector of length \eqn{p}.
#' @param pos.matrix matrix contains information of positions of derivatives.
#' Default is \code{NULL}, where the function will call
#' \code{\link[=mt3_pos]{mt3_pos()}} or \code{\link[=mt4_pos]{mt4_pos()}}.
#' @return \code{mt3_covZtZs} Covariance matrix relating to the use
#' of third derivatives.
# @seealso \code{\link[=pos]{pos()}}.
#' @export
#' @examples
#' set.seed(1)
#' p <-3
#' x <-MASS::mvrnorm(100, rep(0, p), diag(p))
#' t <-rep(0.2, p)
#' s <-rep(-.3, p)
#' # Using third derivatives
#' pos.matrix3 <-mt3_pos(p)
#' sZtZs3 <-mt3_covZtZs(t, s, pos.matrix = pos.matrix3)
#' dim(sZtZs3)
#' sZtZs3[1:5, 1:5]
mt3_covZtZs <-function(t, s, pos.matrix = NULL){
  p <-length(t)
  if (is.null(pos.matrix)){
    pos.matrix <-mt3_pos(p)
  }
  lT <-sum(unlist(lapply(2:(p +1), FUN = function(x) choose(x,k = 2))))
  lZ <-1 + p + p*(p+1)/2 + lT
  ts <-t + s
  expt <-exp(sum(t*t)/2)
  exps <-exp(sum(s*s)/2)
  expts <-exp(sum(ts * ts)/2)
  sZtZs <-array(0, dim = c(lZ, lZ))
  #################
  for (i in 1:lZ){
    for (k in 1:lZ){
      j1 <-pos.matrix[i, 1]
      j2 <-pos.matrix[i, 2]
      j3 <-pos.matrix[i, 3]

      k1 <-pos.matrix[k, 1]
      k2 <-pos.matrix[k, 2]
      k3 <-pos.matrix[k, 3]
```

```
      tab1 <-as.data.frame(table(c(j1, j2, j3)))
      # muj <-prod(apply(tab1, MARGIN = 1, dMGF, t = t))
      muj <-prod(apply(tab1, MARGIN = 1, function(tab) dMGF(tab, t = t)))
      tab2 <-as.data.frame(table(c(k1, k2, k3)))
      muk <-prod(apply(tab2, MARGIN = 1, dMGF, t = s))
      mytab <-as.data.frame(table(c(j1, j2, j3, k1, k2, k3)))
      sZtZs[i, k] <-prod(apply(mytab, MARGIN = 1, dMGF, t = ts))*expts -
        muj*expt * muk*exps
    }
  }
  return(sZtZs)
}
#####################################
#####################################
#' @rdname covZtZs
#' @return \code{mt4_covZtZs} Covariance matrix relating to the use of
#' fourth derivatives. This also contains information on the third
#' third derivatives \code{mt3_covZtZs}.
#' @export
#' @examples
#' # Using fourth derivatives
#' sZtZs4 <-mt4_covZtZs(t, s)
#' dim(sZtZs4)
#' sZtZs4[1:5, 1:5]
mt4_covZtZs <-function(t, s, pos.matrix = NULL){
  p <-length(t)
  if (is.null(pos.matrix)){
    pos.matrix <-mt4_pos(p)
  }
  lT3 <-p + choose(p, 2)*2 + choose(p, 3)
  lZ3 <-1 + p + p*(p+1)/2 + lT3
  lT4 <-p + 3 *choose(p, 2) + 3*choose(p, 3) + choose(p, 4)
  lZ4 <-(lT4 + lZ3)

  ts <-t + s
  expt <-exp(sum(t*t)/2)
  exps <-exp(sum(s*s)/2)

  expts <-exp(sum(ts * ts)/2)

  sZtZs <-array(0, dim = c(lZ4, lZ4))
  #################
  for (i in 1:lZ4){
    for (k in 1:lZ4){
```

```
      j1 ←pos.matrix[i, 1]
      j2 ←pos.matrix[i, 2]
      j3 ←pos.matrix[i, 3]
      j4 ←pos.matrix[i, 4]

      k1 ←pos.matrix[k, 1]
      k2 ←pos.matrix[k, 2]
      k3 ←pos.matrix[k, 3]
      k4 ←pos.matrix[k, 4]

      tab1 ←as.data.frame(table(c(j1, j2, j3, j4)))
      muj ←prod(apply(tab1, MARGIN = 1, function(tab) dMGF(tab, t = t)))

      tab2 ←as.data.frame(table(c(k1, k2, k3, k4)))
      muk ←prod(apply(tab2, MARGIN = 1, dMGF, t = s))

      mytab ←as.data.frame(table(c(j1, j2, j3, j4, k1, k2, k3, k4)))
      sZtZs[i, k] ←prod(apply(mytab, MARGIN = 1, dMGF, t = ts))*expts -
        muj*expt * muk*exps
    }
  }
  return(sZtZs)
}
```

Source Code A.10: matrix_A.R

```
## -- Filename: matrix_A.R
#' @name matrix_A
#' @title From derivatives of MGF to derivatives of CGF.
#' @description
#' Taylor expansion implies that vectors of derivatives of
#' \eqn{\log(\hat{M}_X(t))} can be approximated
#' by a linear combination of vectors of derivatives of \eqn{\hat{M}_X(t)}.
#' \code{matrix_A} results the corresponding
#' linear combinations.
#' @param t vector of \eqn{\mathbb{R}^p}
#' @return \code{mt3_matrix_A} returns coefficient matrix relating to the use
#' of third derivatives.
#' @examples
#' p ←3
#' t ←rep(.2, p)
#' A3 ←mt3_matrix_A(t)
#' dim(A3)
#' A3[1:5, 1:5]
#' A4 ←mt4_matrix_A(t)
```

```
#' dim(A4)
#' A4[1:5, 1:5]
#' @export
mt3_matrix_A ←function(t){
  p ←length(t)
  lT ←p + choose(p, 2)*2 + choose(p, 3)
  lZ ←1 + p + p*(p+1)/2 + lT
  capA ←array(0, c(lT, lZ))
  i ←1
  for (j1 in 1:p){
    vA001 ←mt3_rev_pos(0, 0, j1, p)
    vA011 ←mt3_rev_pos(0, j1, j1, p)
    vA111 ←mt3_rev_pos(j1, j1, j1, p)
    for (j2 in j1:p){
      vA002 ←mt3_rev_pos(0, 0, j2, p)
      vA022 ←mt3_rev_pos(0, j2, j2, p)
      vA222 ←mt3_rev_pos(j2, j2, j2, p)
      vA012 ←mt3_rev_pos(0, j1, j2, p)

      vA112 ←mt3_rev_pos(j1, j1, j2, p)
      vA122 ←mt3_rev_pos(j1, j2, j2, p)
      for (j3 in j2:p){
        vA003 ←mt3_rev_pos(0, 0, j3, p)
        vA033 ←mt3_rev_pos(0, j3, j3, p)
        vA333 ←mt3_rev_pos(j3, j3, j3, p)

        vA013 ←mt3_rev_pos(0, j1, j3, p)
        vA023 ←mt3_rev_pos(0, j2, j3, p)
        vA113 ←mt3_rev_pos(j1, j1, j3, p)
        vA123 ←mt3_rev_pos(j1, j2, j3, p)
        vA133 ←mt3_rev_pos(j1, j3, j3, p)
        vA223 ←mt3_rev_pos(j2, j2, j3, p)
        vA233 ←mt3_rev_pos(j2, j3, j3, p)

        if (j1 == j2){
          if (j2 == j3){
            capA[i, 1] ←3*t[j1] - t[j1]^3
            capA[i, vA001] ←3*(t[j1]^2 - 1)
            capA[i, vA011] ←-3*t[j1]
            capA[i, vA111] ←1
          } else {
            capA[i, 1] ←-t[j3]*(t[j1]^2 - 1)
            capA[i, vA001] ←2 * t[j1]*t[j3]
            capA[i, vA003] ←t[j1]^2 - 1
            capA[i, vA011] ←-t[j3]
```

```
          capA[i, vA013] ←-2*t[j1]
          capA[i, vA113] ←1
        }
      } else{
        if (j2 == j3){
          capA[i, 1] ←-t[j1]*(t[j2]^2 - 1)
          capA[i, vA001] ←t[j2]^2 - 1
          capA[i, vA002] ←2*t[j1]*t[j2]
          capA[i, vA012] ←-2*t[j2]
          capA[i, vA022] ←-t[j1]
          capA[i, vA122] ←1
        } else {
          capA[i, 1] ←-t[j1]*t[j2]*t[j3]
          capA[i, vA001] ←t[j2]*t[j3]
          capA[i, vA002] ←t[j1]*t[j3]
          capA[i, vA003] ←t[j2]*t[j1]
          capA[i, vA012] ←-t[j3]
          capA[i, vA013] ←-t[j2]
          capA[i, vA023] ←-t[j1]
          capA[i, vA123] ←1
        }
      }
      i ←i+1
    }
  }
}
capA ←exp(-sum(t*t)/2) * capA
return(capA)
}
##############################################
##############################################
#' @rdname matrix_A
#' @return \code{mt4_matrix_A} returns coefficient matrix relating to the
#' use of fourth derivatives.
#' @export
mt4_matrix_A ←function(t){
p ←length(t)
lT3 ←p + choose(p, 2)*2 + choose(p, 3)
lZ3 ←1 + p + p*(p+1)/2 + lT3
lT4 ←p + 3 *choose(p, 2) + 3*choose(p, 3) + choose(p, 4)
lZ4 ←(lT4 + lZ3)
capA ←array(0, c(lT4, lZ4))
pos.matrix ←mt4_pos(p)
pos.matrixT4 ←matrix(pos.matrix[(lZ4 - lT4 +1):lZ4, ], ncol = 4)
for (j1 in 1: p){
```

```
indT4 ←which(apply(pos.matrixT4, 1,
                   function(r) all(r == rep(j1, 4)))
             )
vA0001 ←which(
  apply(pos.matrix, 1, function(r) all(r == c(0, 0, 0, j1)))
  )
vA0011 ←which(
  apply(pos.matrix, 1, function(r) all(r == c(0, 0, j1, j1)))
              )
vA0111 ←which(
  apply(pos.matrix, 1, function(r) all(r == c(0, j1, j1, j1)))
  )
vA1111 ←which(
  apply(pos.matrix, 1, function(r) all(r == c(j1, j1, j1, j1)))
  )
t1 ←t[j1]
capA[indT4, 1] ←t1^4 - 6*t1^2 + 3
capA[indT4, vA0001] ←-4*t1^3 + 12*t1
capA[indT4, vA0011] ←6*t1^2 - 6
capA[indT4, vA0111] ←-4*t1
capA[indT4, vA1111] ←1
for (j2 in j1:p){
  ##########################
  vA0002 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, 0, j2)))
    )
  vA0012 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j1, j2)))
    )
  vA0022 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j2, j2)))
    )
  vA0112 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j1, j2)))
    )
  vA0122 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j2, j2)))
    )
  vA0222 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j2, j2, j2)))
    )
  vA1112 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(j1, j1, j1, j2)))
    )
  vA1222 ←which(
```

```
  apply(pos.matrix, 1, function(r) all(r == c(j1, j2, j2, j2)))
  )
vA1122 ←which(
  apply(pos.matrix, 1, function(r) all(r == c(j1, j1, j2, j2)))
  )
if (j2 > j1){
  ##########################
  # when c(j1, j1, j1, j2)
  indT4 ←which(
    apply(pos.matrixT4, 1, function(r) all(r == c(j1, j1, j1, j2)))
    )
  capA[indT4, 1] ←(t[j1]^3 - 3*t[j1])*t[j2]
  capA[indT4, vA0001] ←(-3*t[j1]^2 + 6)*t[j2]
  capA[indT4, vA0002] ←-t[j1]^3 + 3*t[j1]
  capA[indT4, vA0011] ←3*t[j1]*t[j2]
  capA[indT4, vA0012] ←3*t[j1]^2 - 3
  capA[indT4, vA0111] ←-t[j2]
  capA[indT4, vA0112] ←-3*t[j1]
  capA[indT4, vA1112] ←1
  ##########################
  # when c(j1, j2, j2, j2)
  indT4 ←which(
    apply(pos.matrixT4, 1, function(r) all(r == c(j1, j2, j2, j2)))
    )
  capA[indT4, 1] ←t[j1] *(t[j2]^3 - 3*t[j2])
  capA[indT4, vA0001] ←-t[j2]^3 + 3*t[j2]
  capA[indT4, vA0002] ←t[j1]*(-3*t[j2]^2 + 6)
  capA[indT4, vA0012] ←3*t[j2]^2 -3
  capA[indT4, vA0022] ←3*t[j1]*t[j2]
  capA[indT4, vA0122] ←-3*t[j2]
  capA[indT4, vA0222] ←-t[j1]
  capA[indT4, vA1222] ←1
  ##########################
  # when c(j1, j1, j2, j2)
  indT4 ←which(
    apply(pos.matrixT4, 1, function(r) all(r == c(j1, j1, j2, j2)))
    )
  capA[indT4, 1] ←t[j1]^2*t[j2]^2 - t[j1]^2 - t[j2]^2 + 1 + 1
  capA[indT4, vA0001] ←2*t[j1]*(1 - t[j2]^2)
  capA[indT4, vA0002] ←2*t[j2]*(1 - t[j1]^2)
  capA[indT4, vA0011] ←t[j2]^2 -1
  capA[indT4, vA0012] ←4*t[j1]*t[j2]
  capA[indT4, vA0022] ←t[j1]^2 -1
  capA[indT4, vA0112] ←-2 *t[j2]
  capA[indT4, vA0122] ←-2*t[j1]
```

```
  capA[indT4, vA1122] ←1
}

for (j3 in j2:p){
  vA0003 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, 0, j3)))
    )
  vA0013 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j1, j3)))
    )
  vA0023 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j2, j3)))
    )
  vA0033 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j3, j3)))
    )
  vA0113 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j1, j3)))
    )
  vA0123 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j2, j3)))
    )
  vA0133 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j3, j3)))
    )
  vA0223 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j2, j2, j3)))
    )
  vA0233 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j2, j3, j3)))
    )
  vA1123 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(j1, j1, j2, j3)))
    )
  vA1223 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(j1, j2, j2, j3)))
    )
  vA1233 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(j1, j2, j3, j3)))
    )
  if ((j3 > j2)&(j2 > j1)){
    ##########################
    # when c(j1, j1, j2, j3)
    indT4 ←which(
      apply(pos.matrixT4, 1, function(r) all(r == c(j1, j1, j2, j3)))
```

```
  )
capA[indT4, 1] ←(t[j1]^2 - 1)*t[j2]*t[j3]
capA[indT4, vA0001] ←-2 *t[j1] * t[j2]* t[j3]
capA[indT4, vA0002] ←(1 - t[j2]^2)* t[j3]
capA[indT4, vA0003] ←(1 - t[j1]^2)* t[j2]
capA[indT4, vA0011] ←t[j2] * t[j3]
capA[indT4, vA0012] ←2 * t[j1] * t[j3]
capA[indT4, vA0013] ←2 * t[j1] * t[j2]
capA[indT4, vA0023] ←t[j1]^2 - 1
capA[indT4, vA0112] ←- t[j3]
capA[indT4, vA0113] ←- t[j2]
capA[indT4, vA0123] ←-2*t[j1]
capA[indT4, vA1123] ←1
#########################
# when c(j1, j2, j2, j3)
indT4 ←which(
  apply(pos.matrixT4, 1, function(r) all(r == c(j1, j2, j2, j3)))
  )
capA[indT4, 1] ←t[j1]*(t[j2]^2 - 1)* t[j3]
capA[indT4, vA0001] ←(1 - t[j2]^2)*t[j3]
capA[indT4, vA0002] ←-2*t[j1]* t[j2]*t[j3]
capA[indT4, vA0003] ←t[j1]*(1 - t[j2]^2)
capA[indT4, vA0012] ←2*t[j2]*t[j3]
capA[indT4, vA0013] ←t[j2]^2 - 1
capA[indT4, vA0022] ←t[j1]*t[j3]
capA[indT4, vA0023] ←2*t[j1]* t[j2]
capA[indT4, vA0122] ←-t[j3]
capA[indT4, vA0123] ←- 2*t[j2]
capA[indT4, vA0223] ←-t[j1]
capA[indT4, vA1223] ←1
#########################
# when c(j1, j2, j3, j3)
indT4 ←which(
  apply(pos.matrixT4, 1, function(r) all(r == c(j1, j2, j3, j3)))
  )
capA[indT4, 1] ←t[j1]*t[j2]*(t[j3]^2 -1)
capA[indT4, vA0001] ←t[j2]*(t[j3]^2 - 1)
capA[indT4, vA0002] ←t[j1]*(t[j3]^2 - 1)
capA[indT4, vA0003] ←-2 *t[j1]*t[j2]*t[j3]
capA[indT4, vA0012] ←t[j3]^2 - 1
capA[indT4, vA0013] ←2 * t[j2]*t[j3]
capA[indT4, vA0023] ←2 * t[j1]* t[j3]
capA[indT4, vA0033] ←t[j1]*t[j2]
capA[indT4, vA0123] ←- 2*t[j3]
capA[indT4, vA0133] ←-t[j2]
```

```
  capA[indT4, vA0233] ←- t[j1]
  capA[indT4, vA1233] ←1
}
for (j4 in j3:p){
  vA0004 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, 0, j4)))
    )
  vA0014 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j1, j4)))
    )
  vA0024 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j2, j4)))
    )
  vA0024 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j3, j4)))
    )
  vA0034 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, 0, j3, j4)))
    )
  vA0124 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j2, j4)))
    )
  vA0134 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j1, j3, j4)))
    )
  vA0234 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(0, j2, j3, j4)))
    )
  vA1234 ←which(
    apply(pos.matrix, 1, function(r) all(r == c(j1, j2, j3, j4)))
    )
  # When j1 < j2 < j3 < j4
  if (all(!duplicated(c(j1, j2, j3, j4)))){
    indT4 ←which(
      apply(pos.matrixT4, 1, function(r) all(r == c(j1, j2, j3, j4)))
      )
    capA[indT4, 1] ←t[j1]*t[j2]*t[j3]*t[j4]
    capA[indT4, vA0001] ←-t[j2]*t[j3]*t[j4]
    capA[indT4, vA0002] ←-t[j1]*t[j3]*t[j4]
    capA[indT4, vA0003] ←-t[j1]*t[j2]*t[j4]
    capA[indT4, vA0004] ←-t[j1]*t[j2]*t[j3]

    capA[indT4, vA0012] ←t[j3]*t[j4]
    capA[indT4, vA0013] ←t[j2]*t[j4]
    capA[indT4, vA0014] ←t[j2]*t[j3]
```

```
            capA[indT4, vA0023] ←t[j1]*t[j4]
            capA[indT4, vA0024] ←t[j1]*t[j3]
            capA[indT4, vA0034] ←t[j1]*t[j2]

            capA[indT4, vA0123] ←-t[j4]
            capA[indT4, vA0124] ←-t[j3]
            capA[indT4, vA0134] ←-t[j2]
            capA[indT4, vA0234] ←-t[j1]

            capA[indT4, vA1234] ←1

          }
        }
      }
    }
  }
  capA ←capA * exp(-sum(t*t)/2)
  return(capA)
}
```

Source Code A.11: covTtTs.R

```
## --Filename: covTtTs.R
#' @name covTtTs
#' @title Covariance matrix of derivatives of sample cumulant
#' generating function (CGF).
#' @description Stacking third/fourth derivatives of sample CGF together
#' to obtain a vector, which (under normality assumption on data) approaches
#' a normally distributed vector with zero mean and a covariance matrix.
#' More specifically, \code{covTsTs} computes covariance between any two
#' points as the form \eqn{t = t^*1_p} and \eqn{s = s^*1_p}.
#' @details Number of distinct third derivatives is
#' \eqn{
#' l_{T_3}= p + 2 \times \begin{pmatrix}
#' p\\2
#' \end{pmatrix} + \begin{pmatrix}
#' p \\ 3
#' \end{pmatrix}
#' }
#' Number of distinct fourth derivatives is
#' \eqn{
#' l_{T_4} = p + 3 \times \begin{pmatrix}
#' p\\2
#' \end{pmatrix} + 3 \times \begin{pmatrix}
#' p \\ 3
```

```
#' \end{pmatrix} + \begin{pmatrix}
#' p \\ 4
#' \end{pmatrix}
#' }
#' For each pairs of \eqn{(t^*, s^*)}, \code{covTsTt} results a covariance
#' matrix of size \eqn{l_{T_3} \times l_{T_3}} or \eqn{l_{T_4} \times l_{T_4}}.
#' @param bigt array contains value of \eqn{t^*}.
#' @param p dimension of multivariate random vector which data are collected.
#' @param pos.matrix matrix containing information of position of any
#' derivatives. Default is \code{NULL}, the function will call
#'\code{\link[=mt3_pos]{mt3_pos()}} or \code{\link[=mt4_pos]{mt4_pos()}}.
#' @return A 2 dimensional upper triangular array, with size equals to
#' length of \code{bigt}. Each element contains a covariance matrix of
#' derivatives sequences between any two points \eqn{t = t^* 1_p} and
#' \eqn{s = s^*1_p}.
#' \code{mt3_covTsTt} returns the resulting third derivatives.
#' @export
#' @examples
#' bigt <-seq(-1, 1, .4)
#' p <-2
#' # Third derivatives
#' mt3_pos.matrix <-mt3_pos(p)
#' sTsTt3 <-mt3_covTtTs(bigt = bigt, p = p, pos.matrix = mt3_pos.matrix)
#' dim(sTsTt3)
#' sTsTt3[1:5, 1:5]
#' # Fourth derivatives
#' mt4_pos.matrix <-mt4_pos(p)
#' sTsTt4 <-mt4_covTtTs(bigt = bigt, p = p, pos.matrix = mt4_pos.matrix)
#' dim(sTsTt4)
#' sTsTt4[1:5, 1:5]
mt3_covTtTs <-function(bigt, p = 1, pos.matrix= NULL){
  if (is.null(pos.matrix)){
    pos.matrix<- mt3_pos(p)
  }
  numt <-length(bigt)
  re <-vector(mode = "list", length = numt * numt)
  dim(re) <-c(numt, numt)
  lst_matrixA <-lapply(bigt, function(t) mt3_matrix_A(rep(t, p)))
  for (i in 1:numt){
    myt <-rep(bigt[i], p)
    capAt <-lst_matrixA[[i]]
    for (j in i:numt){
      mys <-rep(bigt[j], p)
      # capAs <-mt3_matrix_A(t = mys)
      capAs <-lst_matrixA[[j]]
```

```
      sZtZs ←mt3_covZtZs(t = myt, s = mys, pos.matrix= pos.matrix)
      sTtTs ←capAt %*% sZtZs %*% t(capAs)
      re[[i, j]] ←sTtTs
    }
  }
  return(re)
}
################################################
#' @rdname covTtTs
#' @return \code{mt4_covTsTt} returns the resulting forth derivatives.
#' @export
mt4_covTtTs ←function(bigt, p = 1, pos.matrix= NULL){
  if (is.null(pos.matrix)){
    pos.matrix ←mt4_pos(p)
  }
  numt ←length(bigt)
  re ←vector(mode = "list", length = numt * numt)
  dim(re) ←c(numt, numt)

  lst_matrixA ←lapply(bigt, function(t) mt4_matrix_A(rep(t, p)))
  for (i in 1:numt){
    myt ←rep(bigt[i], p)
    capAt ←lst_matrixA[[i]]
    for (j in i:numt){
      mys ←rep(bigt[j], p)
      capAs ←lst_matrixA[[j]]
      sZtZs ←mt4_covZtZs(t = myt, s = mys, pos.matrix= pos.matrix)
      sTtTs ←capAt %*% sZtZs %*% t(capAs)
      re[[i, j]] ←sTtTs
    }
  }
  return(re)
}
```

Source Code A.12: covLtLs.R

```
## -- Filename: covLtLs.R
#' @name covLtLs
#' @title Linear combinations of distinct derivatives of empirical
#' cumulant generating function (CGF).
#' @description
#' Linear combination of third/fourth derivatives of CGF gives an asymptotically
#' univariate Gaussian process with mean 0 and covariance between two points
#' \eqn{t \in \mathbb{R}^p} and \eqn{s \in \mathbb{R}^p} is defined.
#' We consider vector \eqn{t} and \eqn{s} as the form \eqn{t = t^*1_p}
```

```
#' and \eqn{s = s^*1_p}.
#' @param l vector of linear combination of size equal to the number of distinct
#' derivatives, see \code{\link[=l_dhCGF]{l_dhCGF()}}.
#' @param p dimension of multivariate random vector which data are collected.
#' @param bigt array of value \eqn{t^*} and \eqn{s^*}.
#' @param sTtTs Covariance matrix of derivatives vector,
#' see \code{\link[=covTtTs]{covTtTs()}}. Default is \code{NULL},
#' when the algorithm
#' will call \code{\link[=mt3_covTtTs]{mt3_covTtTs()}} or
#' \code{\link[=mt4_covTtTs]{mt4_covTtTs()}}.
#' @param seed Random seed to get the estimate of the supremum of the
#' univariate Gaussian process obtained from the linear combination.
#' @return
#' \itemize{
#' \item{\code{sLtLs} covariance matrix of the linear combination of distinct
#' derivatives, which is a zero-mean Gaussian process. }
#' \item{\code{m.supLt} Monte-Carlo estimates of supremum of this
#' Gaussian process}
#' }
#' \code{mt3_covLtLs} returns values related to the use of third derivatives.
#' \code{mt4_covLtLs} returns values related to the use of fourth derivatives.
#' @export
#' @examples
#' bigt ←seq(-1, 1, .4)
#' p ←3
#' # Third derivatives
#' lT3 ←l_dhCGF(p)[[1]]
#' l3 ←rep(1/sqrt(lT3), lT3)
#' mt3_covLtLs(l = l3, p = p, bigt = bigt/sqrt(p), seed = 1)
#' #fourth derivatives
#' lT4 ←l_dhCGF(p)[[2]]
#' l4 ←rep(1/sqrt(lT4), lT4)
#' mt4_covLtLs(l = l4, p = p, bigt = bigt/sqrt(p), seed = 1)
mt3_covLtLs ←function(l, p, bigt = seq(-1, 1, 0.05)/sqrt(p),
                      sTtTs = NULL, seed = 1){
  # bigt = seq(-1, 1, by = 0.05)/sqrt(p)
  # bigt ←bigt/sqrt()
  if (is.null(l)){
    lT3 ←p + choose(p, 2)*2 + choose(p, 3)
    l ←rep(1/sqrt(lT3), lT3)
  }
  if (is.null(sTtTs)){
    sTtTs ←mt3_covTtTs(bigt, p = p)
  }
  numt ←length(bigt)
```

```
  numt ←nrow(sTtTs)
  sLtLs ←array(0, dim = c(numt, numt))

  for (i in 1:numt){
    for (j in i:numt){
      sLtLs[i, j] ←t(l) %*% sTtTs[[i, j]] %*% l
      sLtLs[j, i] ←sLtLs[i, j]
    }
  }
  til.Lt ←diag(1/sqrt(diag(sLtLs))) %*% sLtLs %*% diag(1/sqrt(diag(sLtLs)))
  set.seed(seed)
  sup.Lt ←replicate(8e3, expr= {
    Ltrep ←MASS::mvrnorm(1, mu = rep(0, numt), Sigma = til.Lt)
    max(abs(Ltrep))
  })
  set.seed(NULL)
  m.supLt ←mean(sup.Lt)
  return(list(sLtLs = sLtLs, m.supLt = m.supLt))
}
##############################################
##############################################
#' Title
#' @rdname covLtLs
#' @export
mt4_covLtLs ←function(l, p, bigt = seq(-1, 1, 0.05)/sqrt(p),
                      sTtTs = NULL, seed = 1){
  if (is.null(l)){
    lT4 ←p + 3 *choose(p, 2) + 3*choose(p, 3) + choose(p, 4)
    l ←rep(1/sqrt(lT4), lT4)
  }
  if (is.null(sTtTs)){
    sTtTs ←mt4_covTtTs(bigt, p = p)
  }
  numt ←nrow(sTtTs)
  sLtLs ←array(0, dim = c(numt, numt))
  for (i in 1:numt){
    for (j in i:numt){
      sLtLs[i, j] ←t(l) %*% sTtTs[[i, j]] %*% l
      sLtLs[j, i] ←sLtLs[i, j]
    }
  }
  til.Lt ←diag(1/sqrt(diag(sLtLs))) %*% sLtLs %*% diag(1/sqrt(diag(sLtLs)))
  set.seed(seed)
  sup.Lt ←replicate(8e3, expr= {
    Ltrep ←MASS::mvrnorm(1, mu = rep(0, numt), Sigma = til.Lt)
```

```
    max(abs(Ltrep))
  })
  set.seed(NULL)
  m.supLt ←mean(sup.Lt)
  return(list(sLtLs = sLtLs, m.supLt = m.supLt))
}
```

Source Code A.13: get_params.R

```
##-- Filename: get_params.R --#
#' @name get_params
#' @title Get parameters for plots derivatives of multivariate CGF to assess
#' normality assumption.
#' @description Obtain necessary parameters to build a graphical test using
#' the third/fourth derivatives of cumulant generating function.
# Here, \eqn{1 \leq p \leq 10, t = t^*1_p } with the value of \eqn{t^*} in
# \code{bigt = seq(-1, 1, by = 0.05)/sqrt(p)} and \code{l} chosen as the
# average of distinct derivatives is available.
#' @param p Dimension.
#' @param l Linear transformation of vector of third/fourth distinct
#' derivatives, default is their average.
#' @param bigt Array containing value of \eqn{t^*}.
#' @return
#' \itemize{
#' \item{\code{p}}{ Dimension.}
#' \item{\code{lT}}{ Number of distinct third/fourth order derivatives.}
#' \item{\code{sTtTs}}{ Two dimensional array, each element contains covariance
#' matrix of vector of derivatives, the function called
#' \code{\link[=mt3_covTtTs]{mt3_covTtTs()}}, or
#' \code{\link[=mt4_covTtTs]{mt4_covTtTs()}}}.
#' \item{\code{l.sTtTs}}{ Covariance matrix of linear combination of distinct
#' derivatives, the function called \code{\link[=mt3_covLtLs]{mt3_covLtLs()}},
#' or \code{\link[=mt4_covLtLs]{mt4_covLtLs()}}.}
#' \item{\code{m.supLT}}{ The Monte Carlo estimate of expected value supremum of
#' the Gaussian process, see \code{\link[=covLtLs]{covLtLs()}}.
#' }
#' }
#' \code{mt3_get_param} returns necessary parameters for the 2D plot
#' relying on third derivatives.
#' \code{mt4_get_param} returns necessary parameters for the 2D plot
#' relying on fourth derivatives.
#' @seealso \code{\link[=covZtZs]{covZtZs()}},
#' \code{\link[=covLtLs]{covLtLs()}}, \code{\link[=covTtTs]{covTtTs()}}
#' @export
#' @examples
```

```
#' p <-2
#' mt3 <-mt3_get_param(p, bigt = seq(-1, 1, .4)/sqrt(p))
#' names(mt3)
#' mt4 <-mt4_get_param(p, bigt = seq(-1, 1, .4)/sqrt(p))
#' names(mt4)
mt3_get_param <-function(p,
                         bigt = seq(-1, 1, by = .05)/sqrt(p),
                         l = NULL){
  print(paste("MT3 params for dimension: ", p))
  lT <-p + choose(p, 2)*2 + choose(p, 3)
  numt <-length(bigt)
  pos.matrix <-mt3_pos(p)
  l.sTtTs <-mt3_covTtTs(bigt = bigt, p)
  ############################
  if (is.null(l)){
    l <-rep(1/sqrt(lT), lT)
    # l <-rep(1/, lT)
  }
  tmp <-mt3_covLtLs(l= l, p = p, sTtTs = l.sTtTs, bigt)
  sLtLs <-tmp$sLtLs
  m.supLt <-tmp$m.supLt
  ############################
  mt3 <-list(p = p, lT = lT, bigt = bigt, l = l,
             pos.matrix = pos.matrix,
             l.sTtTs = l.sTtTs,
             sLtLs = sLtLs,
             m.supLt = m.supLt
  )
  return(mt3)
}
################################
################################
#' @rdname get_params
#' @export
mt4_get_param <-function(p, bigt = seq(-1, 1, by = .05)/sqrt(p),
                         l = NULL){
  print(paste("MT4 params for dimension: ", p))
  lT4 <-p + 3 *choose(p, 2) + 3*choose(p, 3) + choose(p, 4)
  numt <-length(bigt)
  pos.matrix <-mt4_pos(p)
  l.sTtTs <-mt4_covTtTs(bigt = bigt, p)
  ############################
  if (is.null(l)){
    l <-rep(1/sqrt(lT4), lT4)
    # l <-rep(1/, lT)
```

```
  }
  tmp ←mt4_covLtLs(l= l, p = p, sTtTs = l.sTtTs)
  sLtLs ←tmp$sLtLs
  m.supLt ←tmp$m.supLt
  ############################
  mt4 ←list(p = p, lT = lT4, bigt = bigt, l = l,
            pos.matrix = pos.matrix,
            l.sTtTs = l.sTtTs,
            sLtLs = sLtLs,
            m.supLt = m.supLt
  )
  return(mt4)
}
```

Source Code A.14: get_params.R

```
## -- Filename: hCGF_plot.R
#' @title Graphical plots to assess multivariate normality assumption.
#' @name Multivariate_CGF_PLot
#' @description Cumulant generating functions of normally distributed
#' random variables has derivatives of order higher than 3 are all 0.
#' Hence, plots of empirical third/fourth order derivatives with large value
#' or high slope gives indication of non-normality.
#' \code{Multivariate_CGF_PLot} estimates and provides confidence region for
#' average (or any linear combination) of third/fourth derivatives of empirical
#' cumulant function at the points \eqn{t = t^*1_p}. Plots for
#' \eqn{p = 2, 3, \dots, 10} will be faster to obtain, as confidence regions
#' and other necessary parameters are available in \code{mt3_lst_param.rda} and
#' \code{mt4_lst_param.rda}.
#' Higher dimension requires expensive computational cost.
#' @param x Data matrix of size \eqn{n \times p}
# @param l Vector of linear combination, having length is the number of
# unique third/fourth derivatives. Default is \code{NULL} where the algorithm
# will use the average of distinct derivatives.
#' @param alpha Significant level (default is \eqn{.05})
#' @return \code{d3hCGF_plot} returns plot relying in third derivatives.
#' @seealso \code{\link[=dhCGF_plot1D]{dhCGF_plot1D()}}
#' @export
#' @examples
#' set.seed(1234)
#' p ←3
#' x ←MASS::mvrnorm(500, rep(0, p), diag(p))
#' d3hCGF_plot(x)
#' d4hCGF_plot(x)
d3hCGF_plot ←function(x,
```

```
                    # l = NULL,
                    alpha = 0.05){
bigt ←seq(-1, 1, by = 0.05)
p ←ncol(x); n ←nrow(x)
lT ←p + choose(p, 2)*2 + choose(p, 3)
numt ←length(bigt)
l ←rep(1/sqrt(lT), lT)
# This part allows other values of l
# if (is.null(l)){
# l ←rep(1/sqrt(lT), lT)
# }
if (!(p %in% 1:10)){
  lst ←mt3_get_param(p, l = l)
  warning("Heavy computation")
} else {
  Hvar ←
    matrix(
      nrow = 10, ncol = 21, byrow = T,
      data = c(
        92.42158217, 74.73877805, 60.7302487, 49.59500711, 40.7146832,
        33.61041073, 27.91023887, 23.3244267, 19.62665994, 16.63973104,
        14.22459407, 12.2719798, 10.69596014, 9.429002869, 8.418171,
        7.622206539, 7.00930269, 6.555417571, 6.243020088, 6.060187813,
        6,
        51.64735474, 42.06049707, 34.43383781, 28.34512273, 23.46757024,
        19.54747827, 16.3872706, 13.83262291, 11.76265797, 10.08245759,
        8.717328805, 7.608403151, 6.70925268, 5.983284647, 5.401735469,
        4.942128632, 4.58709455, 4.323475808, 4.141660713, 4.035103292,
        4,
        38.88149786, 31.69987493, 25.98318627, 21.41640716, 17.75565754,
        14.81153012, 12.43646553, 10.51516324, 8.957276844, 7.691833972,
        6.662962446, 5.826609195, 5.148016599, 4.599779034, 4.160346054,
        3.812871459, 3.544332294, 3.344860732, 3.207246338, 3.126577521,
        3.1,
        32.68733899, 26.64565406, 21.8369797, 17.99606145, 14.91756688,
        12.44204029, 10.4452675, 8.830198359, 7.520794105, 6.457327483,
        5.592783363, 4.890095636, 4.320022099, 3.859508021, 3.490425881,
        3.198606386, 2.973096793, 2.805598483, 2.690047987, 2.622315183,
        2.6,
        29.04678297, 23.66659881, 19.38571068, 15.96742076, 13.22853561,
        11.0268189, 9.251491708, 7.816018372, 6.652612096, 5.708038546,
        4.940402436, 4.316681083, 3.810827712, 3.402311133, 3.074991259,
        2.816254638, 2.616352869, 2.467900968, 2.365503721, 2.305486558,
        2.285714286,
        26.6578591, 21.70842882, 17.77156459, 14.62903647, 12.11197704,
```

```
        10.08929945, 8.458929531, 7.141148429, 6.0735229, 5.20703381,
        4.503111954, 3.931363246, 3.467819647, 3.093592715, 2.793836949,
        2.556952971, 2.373977783, 2.238122517, 2.144428152, 2.089517571,
        2.071428571,
        24.97305857, 20.3259158, 16.63063154, 13.68187755, 11.32079857,
        9.424102982, 7.895805196, 6.66095645, 5.6608713, 4.849482818,
        4.190552338, 3.655528423, 3.221900887, 2.871933871, 2.591690565,
        2.370283663, 2.199301874, 2.0723752, 1.984851205, 1.933561885,
        1.916666667,
        23.72267204, 19.29911891, 15.78260114, 12.97730357, 10.73175543,
        8.928416104, 7.475793238, 6.302459321, 5.352496366, 4.582018088,
        3.956507809, 3.448776016, 3.037390364, 2.705467401, 2.439742546,
        2.22985541, 2.067803051, 1.947525575, 1.864597561, 1.816005865,
        1.8,
        22.75872054, 18.50712799, 15.12814181, 12.43324677, 10.27663975,
        8.54519503, 7.150870282, 6.024942309, 5.113618989, 4.374689512,
        3.774960568, 3.288289533, 2.89407277, 2.576082348, 2.321570752,
        2.12058297, 1.965430277, 1.850291432, 1.770915745, 1.724409283,
        1.709090909,
        21.99337116, 17.87807412, 14.60812103, 12.00077286, 9.914711778,
        8.24030648, 6.892245795, 5.803947699, 4.923303532, 4.209429498,
        3.630180608, 3.160243679, 2.779671764, 2.472756467, 2.227160141,
        2.033249152, 1.883583916, 1.772532444, 1.69598262, 1.651135061,
        1.636363636
      )
    )

  Msup ←c(1.453986705, 1.445928528, 1.444276978, 1.44268609, 1.444282713,
           1.444054044, 1.446514826, 1.445421625, 1.44661582, 1.448280449
  )

  lst ←list(m.supLt = Msup[p],
            varLtLs = c(Hvar[p, ], Hvar[p, 20:1])
  )

  # lst ←mt3_lst_param[[p]]
  # This part costs memories, improve later.
  # if (any(l != rep(1/sqrt(lT), lT))){
  # tmp ←mt3_covLtLs(l= l, p = p, sTtTs = lst$l.sTtTs)
  # lst$sLtLs ←tmp$sLtLs
  # lst$m.supLt ←tmp$m.supLt
  # lst$varLtLs ←diag(lst$sLtLs)
  # }
}
v3 ←lapply(bigt/sqrt(p),function(u) d3hCGF(myt = rep(u, p), x = x))
```

```
  L3 ←unlist(lapply(v3, function(v) sqrt(n) * sum(l*v)))
  u ←sqrt(-2 *log(alpha))
  band1 ←(u + lst$m.supLt)*sqrt(lst$varLtLs)
  band2 ←-(u + lst$m.supLt)*sqrt(lst$varLtLs)
  ylim ←max(max(abs(band1)), abs(L3))
  plot(bigt, L3, ylim = c(-ylim, ylim), col = "blue",
       lty = 1, type = "l", lwd = 2, ylab = bquote(L[3]), xlab = "t")
  graphics::lines(bigt, band1, col = "orange", lty = 6, lwd = 2)
  graphics::lines(bigt, band2, col = "orange", lty = 6, lwd = 2)
  graphics::legend("top", c("Probability bands", expression(L[3])),
                   lty = c(6, 1), col = c("orange", "blue"), lwd = c(2, 2),
                   merge = TRUE, y.intersp = 1.5, text.width = .6)
  graphics::title(main = bquote(MT[3] ~"plot"), adj = 0)
  til.L ←L3/sqrt(lst$varLtLs)
  deci ←ifelse(max(abs(til.L)) >= u + lst$m.supLt, "reject", "accept")
  return(deci)
}
##########################################################
#' Title
#' @rdname Multivariate_CGF_PLot
#'
#' @return \code{d4hCGF_plot} returns plot relying in forth derivatives.
#' @export
d4hCGF_plot ←function(x,
                      # l = NULL,
                      alpha = 0.05){
  bigt ←seq(-1, 1, by = 0.05)
  p ←ncol(x); n ←nrow(x)
  lT4 ←p + 3 *choose(p, 2) + 3*choose(p, 3) + choose(p, 4)
  numt ←length(bigt)
  l ←rep(1/sqrt(lT4), lT4)
  # if (is.null(l)){
  # l ←rep(1/sqrt(lT4), lT4)
  # }
  if (!(p %in% 1:10)){
    lst ←mt4_get_param(p, l = l)
    warning("Heavy computation")
  } else {
    Hvar ←
      matrix(nrow = 10, ncol = 21, byrow = T,
             data = c(
               568.1209021, 448.0515896, 355.0182299, 282.6795124, 226.2374173,
               182.0500981, 147.3446669, 120.0034577, 98.40450305, 81.30213209,
               67.73735645, 56.97044429, 48.4300794, 41.67496479, 36.3648053,
               32.23839726, 29.09714181, 26.79273777, 25.2181407, 24.30112719,
```

```
24,
280.078945, 220.4020183, 174.2856492, 138.5357285, 110.735366,
89.05053643, 72.08563482, 58.77581847, 48.30654298, 40.05326137,
33.53611834, 28.38583129, 24.31794483, 21.11337599, 18.60370546,
16.66006845, 15.18479424, 14.10516504, 13.36883132, 12.94054861,
12.8,
192.017125, 150.4754251, 118.5211313, 93.85864813, 74.76032964,
59.92208683, 48.35667067, 39.31457154, 32.22522209, 26.65317137,
22.26533269, 18.80644783, 16.08066803, 13.93770524, 12.2624126,
10.96695142, 9.984921882, 9.266999309, 8.777740015, 8.493314772,
8.4,
139.4637519, 109.4943184, 86.42252828, 68.59431806, 54.76680757,
44.00306309, 35.5945326, 29.00370386, 23.82158724, 19.73609938,
16.50848791, 13.95570699, 11.93721173, 10.34504666, 9.096400137,
8.128014685, 7.392003839, 6.85274504, 6.484607316, 6.270339886,
6.2,
104.9909838, 82.87394461, 65.77466418, 52.50044615, 42.15382226,
34.05709724, 27.69674951, 22.68233197, 18.71599557, 15.56982032,
13.0689038, 11.0787099, 9.495581318, 8.239611218, 7.249281821,
6.477434023, 5.888246794, 5.454990303, 5.15838038, 4.98541008,
4.928571429,
82.011931, 65.17873306, 52.08615944, 41.85866521, 33.83509445,
27.51431271, 22.51507366, 18.54629239, 15.38495284, 12.85963481,
10.83819206, 9.218508179, 7.921543093, 6.886091921, 6.064829952,
5.421329472, 4.927816535, 4.563496973, 4.313326754, 4.167136587,
4.119047619,
66.62374719, 53.30475074, 42.87967482, 34.68359842, 28.21195052,
23.08037733, 18.99500689, 15.73033808, 13.11274092, 11.00810133,
9.31253868, 7.945410092, 6.8440239, 5.959636079, 5.254414896,
4.699140766, 4.27146887, 3.954627172, 3.73645638, 3.608724244,
3.566666667,
56.33114824, 45.31277767, 36.64217968, 29.78935694, 24.35000213,
20.01455816, 16.54517401, 13.75847828, 11.51265495, 9.69771301,
8.228135973, 7.037312106, 6.073302562, 5.295619503, 4.67277072,
4.180389742, 3.799816819, 3.51703094, 3.321859343, 3.207411075,
3.16969697,
49.50809201, 39.95575422, 32.4136072, 26.4335349, 21.67217263,
17.86555133, 14.81010191, 12.34837317, 10.35825587, 8.744824767,
7.434142751, 6.36854146, 5.503017571, 4.802476072, 4.239619901,
3.793336062, 3.447466197, 3.189878105, 3.011776472, 2.907207735,
2.872727273,
45.07928983, 36.41542436, 29.56901672, 24.13680211, 19.80897198,
16.34684669, 13.5660621, 11.32386571, 9.50948683, 8.036814083,
6.838811252, 5.863247519, 5.069425651, 4.425671118, 3.907404411,
3.495663057, 3.175973205, 2.937495809, 2.772391759, 2.675365267,
```

```
            2.643356643
          )
    )

  Msup ←c(1.478985651,
          1.454432793,
          1.448680764,
          1.449313815,
          1.453737199,
          1.4561804,
          1.463732497,
          1.470303197,
          1.476776159,
          1.484990068)

  lst ←list(m.supLt = Msup[p],
            varLtLs = c(Hvar[p, ], Hvar[p, 20:1])
  )

 }
 v4 ←lapply(bigt/sqrt(p),function(u) d4hCGF(myt = rep(u, p), x = x))
 L4 ←unlist(lapply(v4, function(v) sqrt(n) * sum(l*v)))
 u ←sqrt(-2 *log(alpha))
 band1 ←(u + lst$m.supLt)*sqrt(lst$varLtLs)
 band2 ←-(u + lst$m.supLt)*sqrt(lst$varLtLs)
 ylim ←max(max(abs(band1)), abs(L4))
 plot(bigt, L4, ylim = c(-ylim, ylim), col = "blue",
      lty = 1, type = "l", lwd = 2, ylab = bquote(L[4]), xlab = "t")
 graphics::lines(bigt, band1, col = "orange", lty = 6, lwd = 2)
 graphics::lines(bigt, band2, col = "orange", lty = 6, lwd = 2)
 graphics::legend("top", c("Probability bands", expression(L[4])),
                  lty = c(6, 1), col = c("orange", "blue"), lwd = c(2, 2),
                  merge = TRUE, y.intersp = 1.5, text.width = .6)
 til.L ←L4/sqrt(lst$varLtLs)
 deci ←ifelse(max(abs(til.L)) >= u + lst$m.supLt, "reject", "accept")
 return(deci)
}
```

APPENDIX B

DETAILS OF PROOFS FOR CHAPTER FOUR

Lemma 2.1. The position of $Z^{ppp}(\mathbf{t})$ equals the length of vector $\mathcal{Z}_3^n(\mathbf{t})$.

Proof. We will prove that when $(j_1, j_2, j_3) = (p, p, p)$, position of Z^{ppp} is indeed the length of vector $\mathcal{Z}_3^n(\mathbf{t})$. In other word, we will prove that

$$\begin{aligned}&\sum_{t=0}^{p-1}\sum_{l=0}^{p-1-t}(p+1-t-l)+p+\sum_{l=p}^{p-1}(p+1-l)+\big(p-(p-1)\big)\\ =&1+p+\frac{p(p+1)}{2}+p+2\binom{p}{2}+\binom{p}{3}\end{aligned}$$

which is equivalent to prove the following equality

$$\begin{aligned}&\sum_{t=0}^{p-1}\sum_{l=0}^{p-1-t}(p+1-t-l)=\frac{p(p+1)}{2}+p+p(p-1)+\frac{p(p-1)(p-2)}{6}\\ \Leftrightarrow&\sum_{t=0}^{p-1}\sum_{l=0}^{p-1-t}(p+1-t-l)=\frac{p^3+6p^2+5p}{6}\end{aligned}$$

We will break the summation on the left hand side by each summation of value of $t = 0, 1, \ldots p-1$ as follow

$$\begin{aligned}t=0:&\sum_{l=0}^{p-1}(p+1)-l &&=(p+1)+p+(p-1)+\ldots 3+2\\ t=1:&\sum_{l=0}^{p-2}p-l &&=p+(p-1)+\ldots 3+2\\ &\ldots\ldots &&\ldots\\ t=p-1:&\sum_{l=0}^{0}(2-l) &&=2\end{aligned}$$

Adding each row vertically, we then obtain that

$$\begin{aligned}
\sum_{t=0}^{p-1}\sum_{l=0}^{p-1-t}(p+1-t-l) =& (p+1)+2p+3(p-1)+\cdots+(p-1)\times 3+p\times 2\\
=& (p+1)+\frac{1}{2}\left[\left((p+2)^2-\left(p^2+2^2\right)\right)\right.\\
&\left.+\left((p+2)^2-\left((p-1)^2+3^2\right)\right)+\cdots+\left((p+2)^2-\left(p^2+2^2\right)\right)\right]\\
=& (p+1)+\frac{1}{2}(p-1)(p+2)^2-\left(2^2+3^2+\cdots+p^2\right)\\
=& (p+1)+\frac{1}{2}(p-1)(p+2)^2-\frac{p(p+1(2p+1))}{6}+1\\
=& \frac{p^3+6p^2+5p}{6}=L.H.S
\end{aligned}$$

□

APPENDIX C

SOURCE CODE OF CHAPTER TWO

Meo meo

Source Code C.1: distributions.R (Generating random sample from some distributions)

```
## --Filename: distributions.R
## Generating random sample of some distributions
### Univariate Distribtuion #########
#####################################
rbetapr ←function(n, shape1 = 1, shape2 = 1){
  # Inverse Beta generator
  u ←rbeta(n, shape1 = shape1, shape2 = shape2)
  x ←u/(1 - u)
  return(x)
}
dbetapr ←function(y, shape1 = 1, shape2 = 1){
  logf ←(shape1 - 1)*log(y) - (shape1 + shape2)*log(1 + y) -
    lbeta(shape1, shape2)
  exp(logf)
}
################
# Power exponential distribution
fbeta ←function(x, beta, mu = 0, s = 1){
  #Density function
  const ←s *gamma(1 + 1/(2 *beta)) * 2^(1 + 1/(2*beta))
  1/const * exp(-.5 * (((x - mu)/s)^2)^beta)
}
rpe ←function(n, mu = 0, s = 1, beta = 1){
  # generate univariate power exponential distribution
  alpha ←1/(2*beta)
  qg ←rgamma(n, shape = alpha, scale = 2)
  u ←qg^(alpha)
  tmp ←runif(100)
  u[tmp < 0.5] ←-u[tmp < 0.5]
  u ←mu + u*s
  return(u)
}
# kurtosis
Upe.kur ←function(beta){
  # Kurtosis of of power exponential
  (gamma(1/(2*beta)) * gamma(5/(2*beta)))/gamma(3/(2*beta))^2
}
#####################################
#####################################
# Multivariate Gamma distribution
rmvgamma ←function(n, shape=rep(1, p), rate= rep(1, p), corr=diag(p)){
  require(MASS)
```

```
  ## extract parameters, do sanity checks, deal with univariate case
  if(!is.matrix(corr) || !isSymmetric(corr))
    stop("'corr' must be a symmetric matrix")
  p = ncol(corr)
  if(p == 1) rgamma(n, shape, rate)
  ## generate standard multivariate normal matrix, convert to CDF
  Z = MASS::mvrnorm(n, mu = rep(0, p), Sigma=corr)
  cdf = pnorm(Z)
  ## convert to gamma, return
  sapply(1:p, function(d) qgamma(cdf[,d], shape[d], rate[d]))
}
####################################
####################################
# Multivariate exponential distribution
rmvexp ←function(n, rate= rep(1, p), corr=diag(p)){
  require(MASS)
  ## extract parameters, do sanity checks, deal with univariate case
  if(!is.matrix(corr) || !isSymmetric(corr))
    stop("'corr' must be a symmetric matrix")
  p = ncol(corr)
  if(p == 1) rexp(n, rate)
  ## generate standard multivariate normal matrix, convert to CDF
  Z = MASS::mvrnorm(n, mu = rep(0, p), Sigma=corr)
  cdf = pnorm(Z)
  ## convert to exp, return
  sapply(1:p, function(d) qexp(cdf[,d], rate[d]))
}
#####################################
#####################################
# Multivariate chisquare distribution
rmvchisq ←function(n, df = rep(1, p), ncp = rep(0, p), corr= diag(p)){
  require(MASS)
  ## extract parameters, deal with univariate case
  if(!is.matrix(corr) || !isSymmetric(corr))
    stop("'corr' must be a symmetric matrix")
  p = ncol(corr)
  if(p == 1) rchisq(n, df = df, ncp = ncp)
  ## generate standard multivariate normal matrix, convert to CDF
  Z = MASS::mvrnorm(n, mu = rep(0, p), Sigma=corr)
  cdf = pnorm(Z)
  ## convert to exp, return
  sapply(1:p, function(d) qchisq(cdf[,d], df = df[d], ncp = ncp[d]))
}
#####################################
#####################################
```

```
# Comtaminated distributions:
rmvmix ←function(e, n, corr = diag(p), dist2, ...){
  require("MASS")
  p ←ncol(corr)
  S ←sample(c(1, 2), replace = T, size = n, prob = c(e, 1- e))
  n1 ←length(which(S == 1))
  if (n1 == n){
    df ←MASS::mvrnorm(n, mu = rep(0, p), Sigma = corr)
  } else if (n1 == 0){
    df ←dist2(n, corr = corr)
  } else {
    df ←rbind(MASS::mvrnorm(n1, mu = rep(0, p), Sigma = corr),
              df ←dist2(n - n1, corr = corr))
    df ←df[sample(n, size = n), ]
  }
  return(df)
}
#####################################
#####################################
mix.t ←function(e, n, ndf, corr){
  require("MASS")
  p ←ncol(corr)
  S ←sample(c(1, 2), replace = T, size = n, prob = c(e, 1- e))
  n1 ←length(which(S == 1))
  if (n1 == n){
    df ←MASS::mvrnorm(n, mu = rep(0, p), Sigma = corr)
  } else if (n1 == 0){
    df ←mvtnorm::rmvt(n, sigma = (df - 2)/df * corr, df = ndf)
  } else {
    df ←rbind(MASS::mvrnorm(n1, mu = rep(0, p), Sigma = corr),
              mvtnorm::rmvt(n, sigma = (ndf - 2)/ndf * corr, df = ndf))
    df ←df[sample(n, size = n), ]
  }
  return(df)
}
#################################
#################################
mixture_normal ←function(e, n, S1, S2, mu1, mu2){
  # e: mixture ratio
  require("MASS")
  S ←sample(c(1, 2), replace = T, size = n, prob = c(e, 1- e))
  n1 ←length(which(S == 1))
  if (n1 == n){
    df ←MASS::mvrnorm(n, mu = mu1, Sigma = S1)
  } else if (n1 == 0){
```

```
    df ←MASS::mvrnorm(n, mu = mu2, Sigma = S2)
  } else {
    df ←rbind(MASS::mvrnorm(n1, mu = mu1, Sigma = S1),
              MASS::mvrnorm(n - n1, mu = mu2, Sigma = S2))
    df ←df[sample(n, size = n), ]
  }
  return(data = df)
}
```

Source Code C.2: To_normal.R (Transformation from other univariate distributions to approximately $N(0, 1)$.)

```
## Filename: To_normal.R
## Transformation from other unvariate distribution to approximately normal dist.
########### Exp to normal #################
exp_to_normal ←function(x){
  xbar ←mean(x)
  lambda ←1/xbar
  u ←sapply(x, pexp, rate = lambda)

  return(list(z = qnorm(u), param = c(lambda = lambda)))
}
#################################################
########### Gamma to Normal ####################
gamma_to_normal ←function(x){
  n ←length(x)
  mlogf ←function(param, x){
    alpha ←param[1]
    beta ←param[2]
    n*alpha*log(beta) + n * lgamma(alpha) -
      (alpha - 1)*sum(log(x)) + n/beta*mean(x)
  }
  re ←optim(c(1, 1), mlogf, x = x)$par
  halpha ←re[1]
  hbeta ←re[2]
  u ←sapply(x, pgamma, shape = halpha, scale = hbeta)
  return(list(z = qnorm(u), param = c(alpha = halpha, beta = hbeta)))
}

#################################################
#############Logis to Normal ###################
logis_to_normal ←function(x){
  n ←length(x)
  mlogf ←function(param, x){
    # This function gives -log(likelihood)
```

```
    mu ←param[1]
    sigma ←param[2]
    sum((x - mu)/sigma) + n*log(sigma) +
      2 * sum(log(1 + exp(-(x - mu)/sigma)))
  }
  re ←optim(c(1, 1), mlogf, x = x)$par
  hmu ←re[1]
  hsigma ←re[2]
  u ←sapply(x, function(y) 1/(1 + exp(- (y - hmu)/hsigma)))
  return(list(z = qnorm(u), param = c(mu = hmu, sigma = hsigma)))
}
################################################
############## Lomax to Normal #################
lomax_to_normal ←function(x){
  n ←length(x)
  mlogf ←function(param, x){
    theta ←param[1]
    beta ←param[2]
    - sum(VGAM::dlomax(x, scale = 1/theta, shape3.q = beta, log = T))
  }
  re ←optim(c(1, 1), mlogf, x = x)$par
  htheta ←re[1]
  hbeta ←re[2]
  u ←sapply(x, function(y) 1 - (1 + y*htheta)^(-hbeta))
  return(list(z = qnorm(u), param = c(theta = htheta, beta = hbeta)))
}
####################################
######### Chisquare to Normal #########
invGaussian_to_normal ←function(x){
  n ←length(x)
  hmu ←mean(x)
  hlambda ←1/mean(1/x - 1/(mean(x)))
  u ←sapply(x, statmod::pinvgauss, mean = hmu, shape = hlambda)
  return(list(z = qnorm(u), param = c(mu = hmu, lambda = hlambda)))
}

################################################
############### Power to Normal ################
power_to_normal ←function(x){
  n ←length(x)
  pdf ←function(x, beta, mu = 0, s = 1){
    const ←s *gamma(1 + 1/(2 *beta)) * 2^(1 + 1/(2*beta))
    1/const * exp(-.5 * (((x - mu)/s)^2)^beta)
  }
  mlogf ←function(param, x){
```

```
    mu ←param[1]
    sigma ←param[2]
    beta ←param[3]
    n *log(sigma) + n*lgamma(1 + .5/beta) + n*(1 + .5/beta)*log(2) +
      .5 * sum(abs((x - mu)/sigma)^(2*beta))
  }
  re ←optim(c(0, 1, 0.005), mlogf, x = x)$par
  hmu ←re[1]
  hsigma ←re[2]
  hbeta ←re[3]
  u ←sapply(x, function(y) integrate(pdf, lower = -Inf, upper = y,
                                     beta = hbeta, mu = hmu,
                                     s = hsigma)$value)
  return(list(z = qnorm(u), param = c(mu = hmu, sigma = hsigma, beta = hbeta)))
}
####################################
######### Cauchy to Normal #########
cauchy_to_normal ←function(x){
  n ←length(x)
  mlogf ←function(param, x){
    mu ←param[1]
    sigma ←param[2]
    -n*log(sigma) + sum(log((x - mu)^2 + sigma^2))
  }
  re ←optim(c(0,1), mlogf, x = x)$par
  hmu ←re[1]
  hsigma ←re[2]
  u ←sapply(x, pcauchy, location = hmu, scale = hsigma)
  return(list(z = qnorm(u), param = c(mu = hmu, sigma = hsigma)))
}
##################################
########### Lomax to Normal #######
lomax_to_normal ←function(x){
  n ←length(x)
  mlogf ←function(param, x){
    theta ←param[1]
    beta ←param[2]
    - sum(VGAM::dlomax(x, scale = 1/theta, shape3.q = beta, log = T))
  }
  re ←optim(c(1, 1), mlogf, x = x)$par
  htheta ←re[1]
  hbeta ←re[2]
  u ←sapply(x, function(y) 1 - (1 + y*htheta)^(-hbeta))
  return(list(z = qnorm(u), param = c(theta = htheta, beta = hbeta)))
}
```

```
###################################
###################################
mixture_to_normal ←function(x){
  # EM_algorithm
  ###################################
  em_algorithm ←function(data1, initial_params, tol = 1e-6, max_iter = 1000) {
    n ←length(data1)
    k ←length(initial_params$mix_prob)
    ll_old ←-Inf
    iter ←0
    while (iter < max_iter) {
      # Expectation step
      z ←matrix(0, n, k)
      for (i in 1:k) {
        z[, i] ←dnorm(data1, mean = initial_params$mu[i],
                      sd = initial_params$sd[i]) * initial_params$mix_prob[i]
      }
      z ←z / rowSums(z)
      # Maximization step
      n_k ←colSums(z)
      new_mix_prob ←n_k / n
      new_mu ←colSums(z * data1) / n_k
      new_sd ←sqrt(colSums(z * (outer(data1, new_mu,"-"))^2) / n_k)
      # Check convergence
      ll_new ←sum(log(rowSums(z * initial_params$mix_prob)))
      if (abs(ll_new - ll_old) < tol) {
        break } else {
          initial_params$mix_prob ←new_mix_prob
          initial_params$mu ←new_mu
          initial_params$sd ←new_sd
          ll_old ←ll_new
          iter ←iter + 1
        }
    }
    result ←list(mu = initial_params$mu, sd = initial_params$sd,
                 mix_prob = initial_params$mix_prob,
                 loglik = ll_old, converged = iter < max_iter)
    return(result)
  }
  ########################################
  re ←em_algorithm(x, initial_params = list(mu = c(0, 1), sd = c(1, 2),
                                            mix_prob = c(0.3, .7)))
  param ←list(mu = re$mu,
              sd = re$sd,
              mix_prob = re$mix_prob)
```

```
  u ←re$mix_prob[1] * pnorm(x, mean = re$mu[1], sd = re$sd[1]) +
    re$mix_prob[2] * pnorm(x, mean = re$mu[2], sd = re$sd[2])
  return(list(z = qnorm(u), param = param))
}
################################
###### Beta Prime to normal #####
betapr_to_normal ←function(x){
  n ←length(x)
  mlogf ←function(param, x){
    # This function gives -log(likelihood)
    shape1 ←param[1]
    shape2 ←param[2]
   - (shape1 - 1) *sum(log(x)) + (shape1 + shape2) * sum(log(1+x)) +
      n*lbeta(shape1, shape2)
  }
  re ←optim(c(1, 1), mlogf, x = x)$par
  hshape1 ←re[1]
  hshape2 ←re[2]
  myfunc ←function(y){
    (y^(hshape1 - 1) * (1 + y)^(-hshape1 - hshape2))/beta(hshape1, hshape2)
    }
  u ←sapply(x, function(y) integrate(myfunc, lower = 0, upper = y)$value)
  return(list(z = qnorm(u), param = c(shape1 = hshape1, shape2 = hshape2)))
}
```

Source Code C.3: CF_univariate (Chapter 2)

```
## --Filename: CF_univariate.R
source("To_normal.R")
source("distributions.R")
library(PlotNormTest)
library(ggplot2)
library(dplyr)
library(GGally)
mytheme ←theme_bw() +
  theme(aspect.ratio = 1/1, panel.grid = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"),
        legend.background = element_rect(size=0.5, linetype="solid"),
        legend.text = element_text(size=12))
#######################################
# Cox: Normal
set.seed(123)
```

```
x ←rnorm(200, mean = 0, sd = 1)
re ←cox(sort(x), lambda = 0.5)
a ←re$a[, 1]
ggplot(data.frame(x = re$x, a = a), aes(x = x, y = a)) +
  geom_point(color = "steelblue3", shape = 1, size = 1.5) +
  coord_fixed() +
  xlab("y") +
  ylab("Score function") + mytheme
########################################
# Cox: non-normal
set.seed(123)
x ←rnorm(200, mean = 0, sd = 1)
score_plot1D(sort(x))$plot + xlab("y") +theme(legend.position = "none") +
  ylab("Score function")
set.seed(124)
mybeta ←.5
x ←rpe(200, beta = mybeta)
score_plot1D(sort(x))$plot + xlab("y") +theme(legend.position = "none") +
  ylab("Score function")
set.seed(12)
x ←rt(200, df = 5)
score_plot1D(sort(x))$plot + xlab("y") +theme(legend.position = "none") +
  ylab("Score function")
set.seed(123)
x ←rexp(200)
score_plot1D(sort(x))$plot + xlab("y") +theme(legend.position = "none") +
  ylab("Score function")
########################################
# Cox: outliers:
mu ←0; sd = 1
set.seed(123)
x ←rnorm(200, mean = 0, sd = 1)
#outlier a: 1
set.seed(12)
out ←c(rnorm(1, mean = mu + 1, sd = 2*sd),
        rnorm(1, mean = mu + 1, sd = 2.5*sd))
ind.ori← sample.int(200, 2)
x.out ←x
x.out[ind.ori] ←out
x.out ←sort(x.out)
score_plot1D(x.out)$plot + theme(legend.position = "top")+ xlab("y") +
  ylab("Score function")
#outlier d: 2
set.seed(1234)
out ←rgamma(2, shape = 7.5, scale = 1)
```

```
ind.ori← sample.int(200, 2)
x.out ←x
x.out[ind.ori] ←out
x.out ←sort(x.out)
score_plot1D(x.out)$plot + theme(legend.position = "top")+ xlab("y")+
  ylab("Score function")
# outlier c: 3
set.seed(12)
out ←c(rnorm(1, mean = mu - 1, sd = 2.5*sd),
        rnorm(1, mean = mu + 1, sd = 2.5*sd))
ind.ori← sample.int(200, 2)
x.out ←x
x.out[ind.ori] ←out
x.out ←sort(x.out)
score_plot1D(x.out)$plot + theme(legend.position = "top") + xlab("y")+
  ylab("Score function")
#outlier b:2
set.seed(1234)
out ←c(rnorm(1, mean = mu, sd = 5*sd), rnorm(1, mean = mu, sd = .2*sd))
ind.ori← sample.int(200, 2)
x.out ←x
x.out[ind.ori] ←out
x.out ←sort(x.out)
score_plot1D(x.out)$plot + theme(legend.position = "top")+ xlab("y")+
  ylab("Score function")
################################################
################################################
#Ghosh:
dev.off()
set.seed(123)
x ←rnorm(200); nx ←200

par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x, method = "T3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands"),
       lwd = c(2, 2),
       lty = c(1, 6), col = c("blue", "darkorange"),
       merge = T, y.intersp = 2, text.width = .6)

par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x, method = "T4")
legend("top", c(as.expression(bquote(T[4])),"Probability bands"),
       lwd = c(2, 2),
       lty = c(1, 6), col = c("blue", "darkorange"),
       merge = T, y.intersp = 2, text.width = .6)
```

```
bigt ←seq(-1, 1, by = 0.05)
re ←lapply(bigt, dhCGF1D, x = x)
T3.normal ←sqrt(nx)* unlist(lapply(re, function(i) i$t3))
T4.normal ←sqrt(nx)* unlist(lapply(re, function(i) i$t4))
##############################
# Ghosh - Non normal
# t15
set.seed(245)
y ←rt(200, df = 15)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
lines(bigt, T3.normal, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands",
               as.expression(bquote(T[3] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .6)

dhCGF_plot1D(y, method = "T4")
lines(bigt, T4.normal, lty = "dotted", lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[4])),"Probability bands",
               as.expression(bquote(T[4] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .6)

# rpe.5
set.seed(123)
y ←rpe(200, beta = .5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
lines(bigt, T3.normal, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands",
               as.expression(bquote(T[3] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .6)

dhCGF_plot1D(y, method = "T4")
lines(bigt, T4.normal, lty = "dotted", lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[4])),"Probability bands",
               as.expression(bquote(T[4] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .6)
```

```
# rpe5
set.seed(123)
y ←rpe(200, beta = 5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
lines(bigt, T3.normal, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands",
               as.expression(bquote(T[3] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .6)

dhCGF_plot1D(y, method = "T4")
lines(bigt, T4.normal, lty = "dotted", lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[4])),"Probability bands",
               as.expression(bquote(T[4] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .6)

#rexp
set.seed(123)
y ←rexp(200, rate = 1)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
lines(bigt, T3.normal, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands",
               as.expression(bquote(T[3] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 1.8, text.width = .5)

dhCGF_plot1D(y, method = "T4")
lines(bigt, T4.normal, lty = "dotted", lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[4])),"Probability bands",
               as.expression(bquote(T[4] ~"- Normal"))),
       lwd = c(2, 2, 3),
       lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
       merge = TRUE, y.intersp = 2, text.width = .5)

#mixutre
set.seed(123)
y ←mixture_normal(e = .3, n = 200, S1 = diag(1),
                  S2 = 3*diag(1), mu1 = 0, mu = 5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
```

```
dhCGF_plot1D(y, method = "T3")
lines(bigt, T3.normal, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands",
              as.expression(bquote(T[3] ~"- Normal"))),
      lwd = c(2, 2, 3),
      lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
      merge = TRUE, y.intersp = 2, text.width = .6)

dhCGF_plot1D(y, method = "T4")
lines(bigt, T4.normal, lty = "dotted", lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[4])),"Probability bands",
              as.expression(bquote(T[4] ~"- Normal"))),
      lwd = c(2, 2, 3),
      lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
      merge = TRUE, y.intersp = 2, text.width = .6)

#cauchy
set.seed(123)
y ←rcauchy(200, location = 0, scale = 1)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
lines(bigt, T3.normal, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[3])),"Probability bands",
              as.expression(bquote(T[3] ~"- Normal"))),
      lwd = c(2, 2, 3),
      lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
      merge = TRUE, y.intersp = 1.8, text.width = .5)

dhCGF_plot1D(y, method = "T4")
lines(bigt, T4.normal, lty = "dotted", lwd = 3, col = "firebrick3")
legend("top", c(as.expression(bquote(T[4])),"Probability bands",
              as.expression(bquote(T[4] ~"- Normal"))),
      lwd = c(2, 2, 3),
      lty = c(1, 6, 3), col = c("blue","orange", "firebrick3"),
      merge = TRUE, y.intersp = 1.8, text.width = .6)

##############################
# Transformation methods:
# Density plot
mypar ←par(cex.axis = 1.2, cex.lab = 1.2,
           mar = c(4, 4.2, 2,1), cex.main = 1.2)
y ←seq(0, 10, by = 0.01)
plot(y, dbetapr(y, shape1 = 10, shape2 = 3.5), type = "l",
    col = "grey70", lty = 1, ylim = c(0, 1), lwd = 2,
    ylab = "Density functions")
```

```
lines(y, dgamma(y, shape = 2, scale = 2), lty=2, lwd = 3, col = "grey40")
lines(y, statmod::dinvgauss(y, mean = 1, shape = 2),
     col = "grey0", lty = 3, lwd = 2)
lines(y, VGAM::dlomax(y, scale = 4, shape3.q = 6), lty = 4,
     col = "grey50", lwd = 3)
legend("top", c("Inverse Beta", "Gamma", "Inverse Gaussian", "Lomax"),
      merge = T, y.intersp = 2, text.width = 3,
      lwd = c(2, 3, 2, 3),
      lty = 1:4, col = c("grey70", "grey40", "grey0", "grey50"))
############### Inverse Beta to Normal ###############
set.seed(1234)
x ←rbetapr(200, 10, 3.5)
y ←betapr_to_normal(x)$z
score_plot1D(sort(y))$plot + theme(legend.position = "none") +
  ggtitle("Score function plot") +
  xlab("y") + ylab("Score function")
ggplot(data = data.frame(y), aes(sample = y)) + stat_qq() +
  stat_qq_line() + mytheme + ggtitle("Q-Q plot")+
  xlab("Theoretical quantile") + ylab("Sample quantile")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
dhCGF_plot1D(y, method = "T4")
################ Gamma to Normal ################
set.seed(1234)
x ←rgamma(200, shape = 2, scale = 2)
y ←gamma_to_normal(x)$z
score_plot1D(sort(y))$plot + theme(legend.position = "none") +
  ggtitle("Score function plot") +
  xlab("y") + ylab("Score function")
ggplot(data = data.frame(y), aes(sample = y)) + stat_qq() +
  stat_qq_line() + mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Sample quantile")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
dhCGF_plot1D(y, method = "T4")
################ Inverse Gaussian to Normal ################
set.seed(1234)
x ←statmod::rinvgauss(200, mean = 1, shape = 2)
y ←invGaussian_to_normal(x)$z
score_plot1D(sort(y))$plot + theme(legend.position = "none") +
  ggtitle("Score function plot") +
  xlab("y") + ylab("Score function")
ggplot(data = data.frame(y), aes(sample = y)) + stat_qq() +
  stat_qq_line() + mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Sample quantile")
```

```
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
dhCGF_plot1D(y, method = "T4")
################ Lomax to Normal ################
set.seed(1234)
x ←VGAM::rlomax(200, scale = 4, shape3.q = 6)
y ←lomax_to_normal(x)$z
score_plot1D(sort(y))$plot + theme(legend.position = "none") +
  ggtitle("Score function plot") +
  xlab("y") + ylab("Score function")
ggplot(data = data.frame(y), aes(sample = y)) + stat_qq() +
  stat_qq_line() + mytheme + ggtitle("Q-Q plot")+
  xlab("Theoretical quantile") + ylab("Empirical quantile")

par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3")
dhCGF_plot1D(y, method = "T4")
#####################################################################
############### Testing one to the others ######################
############# 4 by 4 plots ###########
# Having support (0, Inf)
# Exponential
myx ←list()
set.seed(1234)
# myx[[1]] ←rexp(200, rate = 3)
myx[[1]] ←x ←rbetapr(200, 10, 3.5)
set.seed(1234)
myx[[2]] ←rgamma(200, shape = 2, scale = 2)
set.seed(1234)
myx[[3]] ←statmod::rinvgauss(200, mean = 1, shape = 2)
set.seed(1234)
myx[[4]] ←VGAM::rlomax(200, scale = 4, shape3.q = 6)
# Cox
plot_list ←list()
layout.matrix ←matrix(1:16, nrow = 4, byrow = F)
j ←1
######## Using Score plot ####################
for (i in 1:4){
  x ←myx[[i]]

  # y1 ←exp_to_normal(x)$z
  y1 ←betapr_to_normal(x)$z
  y2 ←gamma_to_normal(x)$z
  y3 ←invGaussian_to_normal(x)$z
  y4 ←lomax_to_normal(x)$z
```

```
  # ############# Cox ########################
  plot_list[[j]] ←score_plot1D(sort(y1))$plot +
    theme(legend.position = "none",
          axis.title=element_blank(),
          axis.text=element_blank(),
          axis.ticks=element_blank(),
          aspect.ratio = 3/2)
  plot_list[[j + 1]] ←score_plot1D(sort(y2))$plot +
    theme(legend.position = "none",
          axis.title=element_blank(),
          axis.text=element_blank(),
          axis.ticks=element_blank(),
          aspect.ratio = 3/2)
  plot_list[[j + 2]] ←score_plot1D(sort(y3))$plot +
    theme(legend.position = "none",
          axis.title=element_blank(),
          axis.text=element_blank(),
          axis.ticks=element_blank(),
          aspect.ratio = 3/2)
  plot_list[[j+ 3]] ←score_plot1D(sort(y4))$plot +
    theme(legend.position = "none",
          axis.title=element_blank(),
          axis.text=element_blank(),
          axis.ticks=element_blank(),
          aspect.ratio = 3/2)
  j ←j + 4
}

ggmatrix(plot_list,
         nrow = 4, ncol = 4, byrow = F,
         xAxisLabels = c("Inverse Beta", "Gamma",
                         "Inverse Gaussian", "Lomax"),
         yAxisLabels = c("Inverse Beta", "Gamma",
                          "Inverse Gaussian", "Lomax"),
         xlab = "True distribution",
         ylab = "Fitted distribution", switch = "both") +
  mytheme +
  theme(legend.position = "none",
        # axis.title=element_text(hjust = 1),
        # axis.title.y=element_text(angle=0,vjust=1),
        axis.line = element_line(color = "white"),
        axis.text=element_blank(),
        axis.ticks=element_blank(), aspect.ratio = 3/2,
        strip.background = element_rect(fill = "white", colour = NA),
        strip.text = element_text(colour = "black",
```

```
                                size = rel(1), face = "bold"))
#################################################
################
# Using Q- Q plot
plot_list ←-list()
j ←-1
for (i in 1:4){
  x ←-myx[[i]]

  # y1 ←-exp_to_normal(x)$z
  y1 ←-betapr_to_normal(x)$z
  y2 ←-gamma_to_normal(x)$z
  y3 ←-invGaussian_to_normal(x)$z
  y4 ←-lomax_to_normal(x)$z
  plot_list[[j]] ←-
    ggplot(data = data.frame(y1), aes(sample = y1)) + stat_qq(size= 1) +
    stat_qq_line(distribution = qnorm) +
    mytheme + theme(legend.position = "none",
                  axis.title=element_blank(),
                  axis.text=element_blank(),
                  axis.ticks=element_blank(), aspect.ratio = 3/2)
  plot_list[[j + 1]] ←-
    ggplot(data = data.frame(y2), aes(sample = y2)) + stat_qq(size= 1) +
    stat_qq_line(distribution = qnorm) +
    mytheme + theme(legend.position = "none",
                  axis.title=element_blank(),
                  axis.text=element_blank(),
                  axis.ticks=element_blank(), aspect.ratio = 3/2)
  plot_list[[j + 2]] ←-
    ggplot(data = data.frame(y3), aes(sample = y3)) +
    stat_qq(size= 1) + stat_qq_line() +
    mytheme + theme(legend.position = "none",
                  axis.title=element_blank(),
                  axis.text=element_blank(),
                  axis.ticks=element_blank(), aspect.ratio = 3/2)# +
  plot_list[[j+ 3]] ←-
    ggplot(data = data.frame(y4), aes(sample = y4)) +
    stat_qq(size= 1) + stat_qq_line() +
    mytheme + theme(legend.position = "none",
                  axis.title=element_blank(),
                  axis.text=element_blank(),
                  axis.ticks=element_blank(), aspect.ratio = 3/2)# +
  j ←-j + 4
}
```

```
ggmatrix(plot_list,
        nrow = 4, ncol = 4, byrow = F,
        xAxisLabels = c("Inverse Beta", "Gamma",
                        "Inverse Gaussian", "Lomax"),
        yAxisLabels = c("Inverse Beta", "Gamma",
                         "Inverse Gaussian", "Lomax"),
        xlab = "True distribution",
        ylab = "Fitted distribution", switch = "both") +
  mytheme +
  theme(legend.position = "none",
       # axis.title=element_text(hjust = 1),
       # axis.title.y=element_text(angle=0,vjust=1),
       axis.line = element_line(color = "white"),
       axis.text=element_blank(),
       axis.ticks=element_blank(), aspect.ratio = 3/2,
       strip.background = element_rect(fill = "white", colour = NA),
       strip.text = element_text(colour = "black",
                                 size = rel(1), face = "bold"))
################
################################################
################
# Using T3 plot
# Need to refine graphics parameters inside dhCGF_plot1D function
par(mfrow = c(4, 4), oma = c(3.5, 3.5, 0.1, 0.1),
    pty = "m", omi = c(.6, .6, .1, .1))
layout(mat = layout.matrix)
for (i in 1:4){
  x ←myx[[i]]

  # y1 ←exp_to_normal(x)$z
  y1 ←betapr_to_normal(x)$z
  y2 ←gamma_to_normal(x)$z
  y3 ←invGaussian_to_normal(x)$z
  y4 ←lomax_to_normal(x)$z
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y1, method = "T3")
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y2, method = "T3")
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y3, method = "T3")
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y4, method = "T3")
}

mtext("Inverse Beta",side=1,line= .5,
```

```
      outer=TRUE,cex=1,las = 1, at = c(.13), font = 2)
mtext("Gamma",side=1,line=.5,
      outer=TRUE,cex=1, las = 1, at = c(.38), font = 2)
mtext("Inverse Gaussian",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.64), font = 2)
mtext("Lomax",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.88), font = 2)
mtext("Lomax",side=2,line=-.5,
      outer=TRUE,cex=1, las = 0, at = c(.13), font = 2)
mtext("Inverse Gaussian",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.36), font = 2)
mtext("Gamma",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.62), font = 2)
mtext("Inverse Beta",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.86), font = 2)

mtext("True distribution", side = 1, line = 2.7,
      outer = T, las = 1, cex = 1.2, font = 2)
mtext("Fitted Distribution", side = 2, line = 2.7,
      outer = T, las = 0, cex = 1.2, font = 2)
################

################################################
################
# Using T4 plot:
# Need to refine graphics parameters inside dhCGF_plot1D function
par(mfrow = c(4, 4), oma = c(3.5, 3.5, 0.1, 0.1),
    pty = "m", omi = c(.6, .6, .1, .1))
layout(mat = layout.matrix)
for (i in 1:4){
  x ←myx[[i]]
  y1 ←betapr_to_normal(x)$z
  y2 ←gamma_to_normal(x)$z
  y3 ←invGaussian_to_normal(x)$z
  y4 ←lomax_to_normal(x)$z
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y1, method = "T4")
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y2, method = "T4")
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y3, method = "T4")
  par(mai =c(.1, .1, 0.1, 0.1))
  dhCGF_plot1D(y4, method = "T4")
}
```

```
mtext("Inverse Beta",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.13), font = 2)
mtext("Gamma",side=1,line=.5,
      outer=TRUE,cex=1, las = 1, at = c(.38), font = 2)
mtext("Inverse Gaussian",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.64), font = 2)
mtext("Lomax",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.88), font = 2)
mtext("Lomax",side=2,line=-.5,
      outer=TRUE,cex=1, las = 0, at = c(.13), font = 2)
mtext("Inverse Gaussian",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.36), font = 2)
mtext("Gamma",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.62), font = 2)
mtext("Inverse Beta",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.86), font = 2)

mtext("True distribution", side = 1, line = 2.7,
      outer = T, las = 1, cex = 1.2, font = 2)
mtext("Fitted Distribution", side = 2, line = 2.7,
      outer = T, las = 0, cex = 1.2, font = 2)
##################################
# Density estimators plots
true_dens ←c(function(y) dbetapr(y, shape1 = 10, shape2 = 3.5),
             function(y) dgamma(y, shape = 2, scale = 2),
             function(y) statmod::dinvgauss(y, mean = 1, shape = 2),
             function(y) VGAM::dlomax(y, scale = 4, shape3.q = 6))
######## Plot true density and estimated density ######################
par(mfrow = c(4, 4), oma = c(3.5, 3.5, 0.1, 0.1),
    pty = "m", omi = c(.6, .6, .1, .1))
layout(mat = layout.matrix)
for (i in 1:4){
  x ←myx[[i]]
  dens ←true_dens[[i]]
  # param1 ←exp_to_normal(x)$param
  param1 ←betapr_to_normal(x)$param
  param2 ←gamma_to_normal(x)$param
  param3 ←invGaussian_to_normal(x)$param
  param4 ←lomax_to_normal(x)$param
  y ←seq(0, 10/9*max(x), by = 0.01)
  par(mai =c(.1, .1, 0.1, 0.1))
  plot(y, dens(y),type = "l", lty = 1,
       col = "grey70", ylab = "",xlab = "",yaxt='n',xaxt ='n', lwd = 3,
       ylim = c(0, max(dens(y)) + .06))
  lines(y, dbetapr(y, shape1 = param1[1], shape2 = param1[2]),
```

```
       col = "black", lty = 3, lwd = 2.5)

  plot(y, dens(y),type = "l", lty = 1,
       col = "grey70", ylab = "",xlab = "",yaxt='n',xaxt ='n', lwd = 3,
       ylim = c(0, max(dens(y)) + .06))
  lines(y, dgamma(y, shape = param2[1], scale = param2[2]),
        lty = 3, lwd = 2.5, col = "black")
  plot(y, dens(y),type = "l", lty = 1,
       col = "grey70", ylab = "",xlab = "",yaxt='n',xaxt ='n', lwd = 3,
       ylim = c(0, max(dens(y)) + .06))
  lines(y, statmod::dinvgauss(y, mean = param3[1], shape = param3[2]),
        col = "black", lty = 3, lwd = 2.5)
  plot(y, dens(y),type = "l", lty = 1,
       col = "grey70", ylab = "",xlab = "",yaxt='n',xaxt ='n', lwd = 3,
       ylim = c(0, max(dens(y)) + .06))
  lines(y, VGAM::dlomax(y, scale = 1/param4[1], shape3.q = param4[2]),
        lty = 3, col = "black", lwd = 2.5)
}
mtext("Inverse Beta",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.13), font = 2)
mtext("Gamma",side=1,line=.5,
      outer=TRUE,cex=1, las = 1, at = c(.38), font = 2)
mtext("Inverse Gaussian",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.64), font = 2)
mtext("Lomax",side=1,line= .5,
      outer=TRUE,cex=1, las = 1, at = c(.88), font = 2)
mtext("Lomax",side=2,line=-.5,
      outer=TRUE,cex=1, las = 0, at = c(.13), font = 2)
mtext("Inverse Gaussian",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.36), font = 2)
mtext("Gamma",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.62), font = 2)
mtext("Inverse Beta",side=2,line=.5,
      outer=TRUE,cex=1, las = 0, at = c(.86), font = 2)
mtext("True distribution", side = 1, line = 2.7,
      outer = T, las = 1, cex = 1.2, font = 2)
mtext("Fitted Distribution", side = 2, line = 2.7,
      outer = T, las = 0, cex = 1.2, font = 2)
################################
############### Empirical power of the test ##################
power_skewness <-power_kur <-power_sp <-power_ad <-array(dim = c(12, 4))
colnames(power_skewness) <-colnames(power_kur) <-colnames(power_sp) <-
  colnames(power_ad) <-c("Inverse Beta", "Gamma",
                         "Inverse Gaussian", "Lomax")
rownames(power_skewness) <-rownames(power_kur) <-rownames(power_sp) <-
```

```
  rownames(power_ad) ←
  c(rep("Inverse Beta", 3), rep("Gamma", 3),
    rep("Inverse Gaussian", 3), rep("Lomax", 3))

m ←4e3
sample_size ←c(100, 200, 500); seed ←2345
#### Inverse Beta ######
set.seed(seed)
print("Inverse Beta")
for (i in 1:length(sample_size)){
  nx ←sample_size[i]
  print(nx)
  print(m)
  re ←replicate(m, expr = {
    x ←rbetapr(nx, shape1 = 10, shape2 = 3.5)

    y1 ←betapr_to_normal(x)$z
    y1[y1 == Inf] ←1e10
    y2 ←gamma_to_normal(x)$z
    y2[y2 == Inf] ←1e10
    y3 ←invGaussian_to_normal(x)$z
    y3[y3 == Inf] ←1e10
    y4 ←lomax_to_normal(x)$z
    y4[y4 == Inf] ←1e10

    pv1 ←mvnormalTest::mardia(y1)$mv.test["p-value"]
    pv2 ←mvnormalTest::mardia(y2)$mv.test["p-value"]
    pv3 ←mvnormalTest::mardia(y3)$mv.test["p-value"]
    pv4 ←mvnormalTest::mardia(y4)$mv.test["p-value"]

    sw1 ←stats::shapiro.test(y1)
    sw2 ←stats::shapiro.test(y2)
    sw3 ←stats::shapiro.test(y3)
    sw4 ←stats::shapiro.test(y4)

    ad1 ←nortest::ad.test(y1)
    ad2 ←nortest::ad.test(y2)
    ad3 ←nortest::ad.test(y3)
    ad4 ←nortest::ad.test(y4)

    lst1 ←c(as.numeric(as.character(pv1[1, ])) <= alpha,
            as.numeric(as.character(pv2[1, ])) <= alpha,
            as.numeric(as.character(pv3[1, ])) <= alpha,
            as.numeric(as.character(pv4[1, ])) <= alpha)
```

```
    lst2 ←c(as.numeric(as.character(pv1[2, ])) <= alpha,
             as.numeric(as.character(pv2[2, ])) <= alpha,
             as.numeric(as.character(pv3[2, ])) <= alpha,
             as.numeric(as.character(pv4[2, ])) <= alpha)

    lst3 ←c(sw1$p.value <= alpha,
             sw2$p.value <= alpha,
             sw3$p.value <= alpha,
             sw4$p.value <= alpha)

    lst4 ←c(ad1$p.value <= alpha,
             ad2$p.value <= alpha,
             ad3$p.value <= alpha,
             ad4$p.value <= alpha)
    list(lst1, lst2, lst3, lst4)
  })

  re1 ←array(unlist(re[1, ]), dim = c(4, m))
  re2 ←array(unlist(re[2, ]), dim = c(4, m))
  re3 ←array(unlist(re[3, ]), dim = c(4, m))
  re4 ←array(unlist(re[4, ]), dim = c(4, m))

  power_skewness[i, ] ←apply(re1, 1, mean)
  power_kur[i, ] ←apply(re2, 1, mean)
  power_sp[i, ] ←apply(re3, 1, mean)
  power_ad[i, ] ←apply(re4, 1, mean)
}

##### Gamma ##############
print("Gamma")
set.seed(seed)
for (i in 1:length(sample_size)){
  nx ←sample_size[i]
  print(m)
  re ←replicate(m, expr = {
    x ←rgamma(200, shape = 2, scale = 2)

    y1 ←betapr_to_normal(x)$z
    y1[y1 == Inf] ←1e10
    y2 ←gamma_to_normal(x)$z
    y2[y2 == Inf] ←1e10
    y3 ←invGaussian_to_normal(x)$z
    y3[y3 == Inf] ←1e10
    y4 ←lomax_to_normal(x)$z
    y4[y4 == Inf] ←1e10
```

```
  pv1 <-mvnormalTest::mardia(y1)$mv.test["p-value"]
  pv2 <-mvnormalTest::mardia(y2)$mv.test["p-value"]
  pv3 <-mvnormalTest::mardia(y3)$mv.test["p-value"]
  pv4 <-mvnormalTest::mardia(y4)$mv.test["p-value"]

  sw1 <-stats::shapiro.test(y1)
  sw2 <-stats::shapiro.test(y2)
  sw3 <-stats::shapiro.test(y3)
  sw4 <-stats::shapiro.test(y4)

  ad1 <-nortest::ad.test(y1)
  ad2 <-nortest::ad.test(y2)
  ad3 <-nortest::ad.test(y3)
  ad4 <-nortest::ad.test(y4)

  lst1 <-c(as.numeric(as.character(pv1[1, ])) <= alpha,
           as.numeric(as.character(pv2[1, ])) <= alpha,
           as.numeric(as.character(pv3[1, ])) <= alpha,
           as.numeric(as.character(pv4[1, ])) <= alpha)

  lst2 <-c(as.numeric(as.character(pv1[2, ])) <= alpha,
           as.numeric(as.character(pv2[2, ])) <= alpha,
           as.numeric(as.character(pv3[2, ])) <= alpha,
           as.numeric(as.character(pv4[2, ])) <= alpha)

  lst3 <-c(sw1$p.value <= alpha,
           sw2$p.value <= alpha,
           sw3$p.value <= alpha,
           sw4$p.value <= alpha)

  lst4 <-c(ad1$p.value <= alpha,
           ad2$p.value <= alpha,
           ad3$p.value <= alpha,
           ad4$p.value <= alpha)
  list(lst1, lst2, lst3, lst4)
})

re1 <-array(unlist(re[1, ]), dim = c(4, m))
re2 <-array(unlist(re[2, ]), dim = c(4, m))
re3 <-array(unlist(re[3, ]), dim = c(4, m))
re4 <-array(unlist(re[4, ]), dim = c(4, m))

power_skewness[i + 3, ] <-apply(re1, 1, mean)
power_kur[i + 3, ] <-apply(re2, 1, mean)
```

```
  power_sp[i + 3, ] ←apply(re3, 1, mean)
  power_ad[i + 3, ] ←apply(re4, 1, mean)
}
##### Inverse Gaussian ##############
print("Inverse Gaussian")
set.seed(seed)
for (i in 1:length(sample_size)){
  nx ←sample_size[i]
  print(m)
  re ←replicate(m, expr = {
    x ←statmod::rinvgauss(200, mean = 1, shape = 2)

    y1 ←betapr_to_normal(x)$z
    y1[y1 == Inf] ←1e10
    y2 ←gamma_to_normal(x)$z
    y2[y2 == Inf] ←1e10
    y3 ←invGaussian_to_normal(x)$z
    y3[y3 == Inf] ←1e10
    y4 ←lomax_to_normal(x)$z
    y4[y4 == Inf] ←1e10

    pv1 ←mvnormalTest::mardia(y1)$mv.test["p-value"]
    pv2 ←mvnormalTest::mardia(y2)$mv.test["p-value"]
    pv3 ←mvnormalTest::mardia(y3)$mv.test["p-value"]
    pv4 ←mvnormalTest::mardia(y4)$mv.test["p-value"]

    sw1 ←stats::shapiro.test(y1)
    sw2 ←stats::shapiro.test(y2)
    sw3 ←stats::shapiro.test(y3)
    sw4 ←stats::shapiro.test(y4)

    ad1 ←nortest::ad.test(y1)
    ad2 ←nortest::ad.test(y2)
    ad3 ←nortest::ad.test(y3)
    ad4 ←nortest::ad.test(y4)

    lst1 ←c(as.numeric(as.character(pv1[1, ])) <= alpha,
            as.numeric(as.character(pv2[1, ])) <= alpha,
            as.numeric(as.character(pv3[1, ])) <= alpha,
            as.numeric(as.character(pv4[1, ])) <= alpha)

    lst2 ←c(as.numeric(as.character(pv1[2, ])) <= alpha,
            as.numeric(as.character(pv2[2, ])) <= alpha,
            as.numeric(as.character(pv3[2, ])) <= alpha,
            as.numeric(as.character(pv4[2, ])) <= alpha)
```

```
    lst3 ←c(sw1$p.value <= alpha,
             sw2$p.value <= alpha,
             sw3$p.value <= alpha,
             sw4$p.value <= alpha)

    lst4 ←c(ad1$p.value <= alpha,
             ad2$p.value <= alpha,
             ad3$p.value <= alpha,
             ad4$p.value <= alpha)
    list(lst1, lst2, lst3, lst4)
  })

  re1 ←array(unlist(re[1, ]), dim = c(4, m))
  re2 ←array(unlist(re[2, ]), dim = c(4, m))
  re3 ←array(unlist(re[3, ]), dim = c(4, m))
  re4 ←array(unlist(re[4, ]), dim = c(4, m))

  power_skewness[i + 6, ] ←apply(re1, 1, mean)
  power_kur[i + 6, ] ←apply(re2, 1, mean)
  power_sp[i + 6, ] ←apply(re3, 1, mean)
  power_ad[i + 6, ] ←apply(re4, 1, mean)
}
###################### Lomax ######################
print("Lomax")
set.seed(seed)
for (i in 1:length(sample_size)){
  nx ←sample_size[i]
  print(m)
  re ←replicate(m, expr = {
    x ←VGAM::rlomax(200, scale = 4, shape3.q = 6)

    y1 ←betapr_to_normal(x)$z
    y1[y1 == Inf] ←1e10
    y2 ←gamma_to_normal(x)$z
    y2[y2 == Inf] ←1e10
    y3 ←invGaussian_to_normal(x)$z
    y3[y3 == Inf] ←1e10
    y4 ←lomax_to_normal(x)$z
    y4[y4 == Inf] ←1e10

    pv1 ←mvnormalTest::mardia(y1)$mv.test["p-value"]
    pv2 ←mvnormalTest::mardia(y2)$mv.test["p-value"]
    pv3 ←mvnormalTest::mardia(y3)$mv.test["p-value"]
    pv4 ←mvnormalTest::mardia(y4)$mv.test["p-value"]
```

```
    sw1 ←stats::shapiro.test(y1)
    sw2 ←stats::shapiro.test(y2)
    sw3 ←stats::shapiro.test(y3)
    sw4 ←stats::shapiro.test(y4)

    ad1 ←nortest::ad.test(y1)
    ad2 ←nortest::ad.test(y2)
    ad3 ←nortest::ad.test(y3)
    ad4 ←nortest::ad.test(y4)

    lst1 ←c(as.numeric(as.character(pv1[1, ])) <= alpha,
             as.numeric(as.character(pv2[1, ])) <= alpha,
             as.numeric(as.character(pv3[1, ])) <= alpha,
             as.numeric(as.character(pv4[1, ])) <= alpha)

    lst2 ←c(as.numeric(as.character(pv1[2, ])) <= alpha,
             as.numeric(as.character(pv2[2, ])) <= alpha,
             as.numeric(as.character(pv3[2, ])) <= alpha,
             as.numeric(as.character(pv4[2, ])) <= alpha)

    lst3 ←c(sw1$p.value <= alpha,
             sw2$p.value <= alpha,
             sw3$p.value <= alpha,
             sw4$p.value <= alpha)

    lst4 ←c(ad1$p.value <= alpha,
             ad2$p.value <= alpha,
             ad3$p.value <= alpha,
             ad4$p.value <= alpha)
    list(lst1, lst2, lst3, lst4)
  })

  re1 ←array(unlist(re[1, ]), dim = c(4, m))
  re2 ←array(unlist(re[2, ]), dim = c(4, m))
  re3 ←array(unlist(re[3, ]), dim = c(4, m))
  re4 ←array(unlist(re[4, ]), dim = c(4, m))

  power_skewness[i + 9, ] ←apply(re1, 1, mean)
  power_kur[i + 9, ] ←apply(re2, 1, mean)
  power_sp[i + 9, ] ←apply(re3, 1, mean)
  power_ad[i + 9, ] ←apply(re4, 1, mean)
}
############### Empricial type I ############
#############################################
```

```
alpha ←.05; m ←4e3; sample_size ←seq(100, 1000, by = 50)
type1 ←list()
set.seed(seed)
for (i in 1:length(sample_size)){
  nx ←sample_size[i]
  print(i)
  # Inverse Beta
  type1_betapr ←replicate(m, expr = {
    x ←rbetapr(nx, shape1 = 10, shape2 = 3.5)
    y ←betapr_to_normal(x)$z
    y[y == Inf] ←1e10

    pv ←mvnormalTest::mardia(y)$mv.test["p-value"]
    # ks ←stats::ks.test(y, "pnorm")
    sw ←stats::shapiro.test(y)
    ad ←nortest::ad.test(y)
    c(Skewness = (as.numeric(as.character(pv[1, ])) >= alpha), # skewness
      Kurtosis = (as.numeric(as.character(pv[2, ])) >= alpha), # kurtosis
      # ks$p.value >= alpha, # Kolmogorov-Smirnov test
      Shapiro = (sw$p.value >= alpha), # Shapiro - Wilk test
      Anderson_Darling = (ad$p.value >= alpha) # Anderson-Darling test
    )
  })
  ####################################
  # gamma
  type1_gamma ←replicate(m, expr = {
    x ←rgamma(nx, shape = 2, scale = 2)
    y ←gamma_to_normal(x)$z
    y[y == Inf] ←1e10

    pv ←mvnormalTest::mardia(y)$mv.test["p-value"]
    # ks ←stats::ks.test(y, "pnorm")
    sw ←stats::shapiro.test(y)
    ad ←nortest::ad.test(y)
    c(Skewness = (as.numeric(as.character(pv[1, ])) >= alpha), # skewness
      Kurtosis = (as.numeric(as.character(pv[2, ])) >= alpha), # kurtosis
      # ks$p.value >= alpha, # Kolmogorov-Smirnov test
      Shapiro = (sw$p.value >= alpha), # Shapiro - Wilk test
      Anderson_Darling = (ad$p.value >= alpha) # Anderson-Darling test
    )
  })
  ####################################
  # invGaussian
  type1_invG ←replicate(m, expr = {
    x ←statmod::rinvgauss(nx, mean =1, shape =2)
```

```
    y ←invGaussian_to_normal(x)$z

    pv ←mvnormalTest::mardia(y)$mv.test["p-value"]
    # ks ←stats::ks.test(y, "pnorm")
    sw ←stats::shapiro.test(y)
    ad ←nortest::ad.test(y)
    c(Skewness = (as.numeric(as.character(pv[1, ])) >= alpha), # skewness
      Kurtosis = (as.numeric(as.character(pv[2, ])) >= alpha), # kurtosis
      # ks$p.value >= alpha, # Kolmogorov-Smirnov test
      Shapiro = (sw$p.value >= alpha), # Shapiro - Wilk test
      Anderson_Darling = (ad$p.value >= alpha) # Anderson-Darling test
    )
  })

  ####################################
  type1_lomax ←replicate(m, expr = {
    x ←VGAM::rlomax(nx, scale = 4, shape3.q = 6)
    y ←lomax_to_normal(x)$z
    y[y == Inf] ←1e10
    pv ←mvnormalTest::mardia(y)$mv.test["p-value"]
    # ks ←stats::ks.test(y, "pnorm")
    sw ←stats::shapiro.test(y)
    ad ←nortest::ad.test(y)
    c(Skewness = (as.numeric(as.character(pv[1, ])) >= alpha), # skewness
      Kurtosis = (as.numeric(as.character(pv[2, ])) >= alpha), # kurtosis
      # ks$p.value >= alpha, # Kolmogorov-Smirnov test
      Shapiro = (sw$p.value >= alpha), # Shapiro - Wilk test
      Anderson_Darling = (ad$p.value >= alpha) # Anderson-Darling test
    )
  })
  re ←cbind(
    # apply(type1_exp, 1, sum),
    apply(type1_betapr, 1, sum),
    apply(type1_gamma, 1, sum),
    apply(type1_invG, 1, sum),
    apply(type1_lomax, 1, sum)
  )
  re ←re/m
  colnames(re) ←c("Inverse Beta", "Gamma", "Inverse-Gaussian", "Lomax")
  type1[[i]] ←re
}
type1_betapr ←1 - simplify2array(
  lapply(type1, function(u) u[, "Inverse Beta"])
)
type1_gamma ←1 - simplify2array(
```

```
  lapply(type1, function(u) u[, "Gamma"])
)
type1_invG<- 1- simplify2array(
  lapply(type1, function(u) u[, "Inverse-Gaussian"])
)
type1_lomax <-1 - simplify2array(
  lapply(type1, function(u) u[, "Lomax"])
)
type1_non_normal <-list(betapr = type1_betapr,
                        gamma = type1_gamma,
                        invG = type1_invG,
                        lomax = type1_lomax)
mydata <-cbind(reshape::melt(
  data.frame(t(type1_betapr)) %>% dplyr::mutate(Size = sample_size),
  id.vars = "Size"
) %>%
  dplyr::rename(
    Method = variable, Inverse_Beta = value),
reshape::melt(data.frame(t(type1_gamma)) %>%
                dplyr::mutate(Size = sample_size), id.vars = "Size"
) %>%
  dplyr::rename(Method = variable,
                Gamma = value) %>% dplyr::select(Gamma),
reshape::melt(data.frame(t(type1_invG)) %>%
                dplyr::mutate(Size = sample_size), id.vars = "Size"
) %>%
  dplyr::rename(Method = variable,
                Inverse_Gaussian = value) %>% dplyr::select(Inverse_Gaussian),
reshape::melt(data.frame(t(type1_lomax)) %>%
                dplyr::mutate(Size = sample_size), id.vars = "Size"
)%>%
  dplyr::rename(Method = variable,
                Lomax = value) %>% dplyr::select(Lomax))
mydata <-reshape::melt(mydata, id.vars = c("Method", "Size")) %>%
  dplyr::rename(Distribution = variable,
                Type_I= value)
mydata$Method <-as.factor(mydata$Method)
mydata$Distribution <-as.factor(mydata$Distribution)
mydata$Size <-as.numeric(mydata$Size)
mydata$Type_I <-as.numeric(mydata$Type_I)
# load("~Normality Test/Simulation_results/CF_Univariate/Uni_type1.Rdata")
# mydata <-Uni_type1
ggplot(data = mydata, aes(Size , Type_I)) +
  geom_line(aes(color = Method, linetype = Method), linewidth = .9) +
  facet_grid(rows = vars(Distribution)) +
```

```
  geom_hline(aes(yintercept = .05), linetype="dotted") +
  # scale_size_manual(values = c("Kolmogorov-Smirnov" = 2) )+
  scale_linetype_manual(values=c("solid","twodash", "dashed", "dotdash"))+
  scale_color_manual(values = c("grey50", "grey70", "grey40", "grey0")) +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1),
        panel.grid = element_blank(), axis.line = element_line(colour = "black"),
        axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"),
        legend.background = element_rect(fill= "white",
                                         size=1, linetype="solid"),
        legend.text = element_text(size = 12),
        legend.position = "top",
        legend.direction = "horizontal",
        strip.text.y = element_text(size = 16),
        legend.title=element_blank(),
        strip.background = element_rect(fill = "white", colour = "black"),
  ) + ylim(0, .06) +
  labs(y = "Type I", x = "Sample size")
```

Source Code C.4: Data_cork.R (Marginal assessment of Cork dataset)

```
##-- Filename: Data_cork.R
library(PlotNormTest)
library(ggplot2)
mytheme ←theme_bw() +
  theme(aspect.ratio = 1/1, panel.grid = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"),
        legend.background = element_rect(
          size=0.5, linetype="solid"),
        legend.text = element_text(size=12))
########################
cork ←matrix(c(
  72, 66, 76, 77,
  60, 53, 66, 63,
  56, 57, 64, 58,
  41, 29, 36, 38,
  32, 32, 35, 36,
  30, 35, 34, 26,
  39, 39, 31, 27,
  42, 43, 31, 25,
  37, 40, 31, 25,
  33, 29, 27, 36,
```

```
  32, 30, 34, 28,
  63, 45, 74, 63,
  54, 46, 60, 52,
  47, 51, 52, 43,
  91, 79, 100, 75,
  56, 68, 47, 50,
  79, 65, 70, 61,
  81, 80, 68, 58,
  78, 55, 67, 60,
  46, 38, 37, 38,
  39, 35, 34, 37,
  32, 30, 30, 32,
  60, 50, 67, 54,
  35, 37, 48, 39,
  39, 36, 39, 31,
  50, 34, 37, 40,
  43, 37, 39, 50,
  48, 54, 57, 43
), nrow = 28, ncol = 4, byrow = T)
colnames(cork) ←c("North", "East", "South", "West")
# Score function
lapply(1:4, FUN = function(mycol) {
  re ←cox(matrix(sort(cork[, mycol])), x.dist = 0.0001)
  a ←re$a[, 1]
  ggplot(data.frame(x = re$x, a = a), aes(x = x, y = a)) +
    geom_point(color = "steelblue3", shape = 19, size = 1.5) +
    ggtitle(paste("Score plot: ", colnames(cork)[mycol])) +
    coord_fixed() + xlab("y")+
    ylab("Score function") + mytheme }
)
# Q - Q
lapply(1:4, FUN = function(mycol) {
  x ←cork[, mycol]
  ggplot(data.frame(x), aes(sample = x)) + stat_qq() +
    stat_qq_line(distribution = qnorm) +
    ggtitle("Q-Q plot")+
    ggtitle(paste("Q - Q plot: ", colnames(cork)[mycol])) +
    xlab("Theoretical quantile") + ylab("Sample quantile") + mytheme
}
)
# T3
lapply(1:4, FUN = function(mycol) {
  x ←cork[, mycol]
  mypar ←par(cex.axis = 1.2, cex.lab = 1.2,
             mar = c(4, 4.2, 2,1), cex.main = 1.2)
```

```
  dhCGF_plot1D(x, method = "T3")
  namex ←colnames(cork)[mycol]
  title(main = bquote(T[3]~"plot: "~.(namex)), adj = 0)
}
)
# T4
mypar ←par(cex.axis = 1.2, cex.lab = 1.2,
           mar = c(4, 4.2, 2,1), cex.main = 1.2)
lapply(1:4, FUN = function(mycol) {
  x ←cork[, mycol]
  dhCGF_plot1D(x, method = "T4")
  namex ←colnames(cork)[mycol]
  title(main = bquote(T[4]~"plot: "~.(namex)), adj = 0)
}
)
```

Source Code C.5: Data_water.R (Marginal assessmeng of Water Potability dataset)

```
##--Filename: Data_water.R
library(dplyr)
library(ggplot2)
library(PlotNormTest)
source("To_normal.R")
water←read.csv("Data/water_potability.csv")
water ←na.omit(water)
X1 ←water %>% dplyr::filter(Potability == 1)
X0 ←water %>% dplyr::filter(Potability == 0)
mytheme ←theme_bw() +
  theme(aspect.ratio = 1/1, panel.grid = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"),
        legend.background = element_rect(
          size=0.5, linetype="solid"),
        legend.text = element_text(size=12))
##############################################################################
lapply(1:9, FUN = function(mycol) {
  x ←X1[, mycol]
  namex ←colnames(X1)[mycol]
  re ←cox(matrix(sort(x)), x.dist = 0.001)
  a ←re$a[, 1]
  ggplot(data.frame(x = re$x, a = a), aes(x = x, y = a)) +
    geom_point(color = "steelblue3", shape = 1, size = 1.5) +
    ggtitle(paste("Score plot: ", namex)) +
    # geom_abline(intercept = 0, slope = 1) +
```

```
    coord_fixed() + xlab("y") +
    ylab("Score function") + mytheme
  }
)
#############################################################################
# Q - Q
lapply(1:9, FUN = function(mycol) {
  x ←X1[, mycol]
  namex ←colnames(X1)[mycol]
  ggplot(data.frame(x), aes(sample = x)) + stat_qq() +
    stat_qq_line(distribution = qnorm) +
    ggtitle("Q-Q plot")+ ggtitle(paste("Q - Q plot: ", namex)) +
    xlab("Theoretical quantile") + ylab("Sample quantile") +mytheme}
)
#############################################################################
lapply(1:9, FUN = function(mycol) {
  namex ←colnames(X1)[mycol]
  par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
  dhCGF_plot1D(X1[, mycol], method = "T4")
  title(main = bquote(T[4]~"plot: "~.(namex)), adj = 0)
}
)
#############################################################################
lapply(1:9, FUN = function(mycol) {
  namex ←colnames(X1)[mycol]
  par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
  dhCGF_plot1D(X1[, mycol], method = "T3")
  title(main = bquote(T[3]~"plot: "~.(namex)), adj = 0)
}
)
#############################################################################
rm.ind ←c(which(X1$ph < 2), 328, 374, 732, 739, 777)
# ph
x ←X1$ph[-rm.ind]
re ←cox(matrix(sort(x)), x.dist = 0.001)
ggplot(data.frame(x = as.matrix(re$x),
                  a = as.matrix(re$a)), aes(x = x, y = a)) +
  geom_point(color = "steelblue3", shape = 1, size = 1.5) +
  ggtitle("Score plot: ph - Remove outliers") +
  # geom_abline(intercept = 0, slope = 1) +
  coord_fixed() +
  xlab("y") +
  ylab("Score function") + mytheme
ggplot(data.frame(x), aes(sample = x)) +
  stat_qq() + stat_qq_line(distribution = qnorm) +
```

```
  ggtitle("Q-Q plot")+ ggtitle("Q - Q plot: ph - Remove outliers") +
  xlab("Theoretical quantile") + ylab("Sample quantile") + mytheme
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x, method = "T3");
title(main = bquote(T[3]~"plot: ph - Remove outliers"), adj = 0)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x, method = "T4");
title(main = bquote(T[4]~"plot: ph - Remove outliers"), adj = 0)
########################
# Solids
x ←X1$Solids[-rm.ind]
tmp ←gamma_to_normal(x)
y ←tmp$z
tmp$param
# y ←gamma_to_normal(x)$z
re ←cox(matrix(sort(y)), x.dist = 0.001)
ggplot(data.frame(x = as.matrix(re$x),
                  a = as.matrix(re$a)), aes(x = x, y = a)) +
  geom_point(color = "steelblue3", shape = 1, size = 1.5) +
  ggtitle("Score plot: Solids - Gamma distribution") +
  # geom_abline(intercept = 0, slope = 1) +
  coord_fixed() + xlab("y") +
  ylab("Score function") + mytheme
ggplot(data.frame(y), aes(sample = y)) +
  stat_qq() + stat_qq_line(distribution = qnorm) +
  ggtitle("Q - Q plot: Solids - Gamma distribution") +
  xlab("Theoretical quantile") + ylab("Sample quantile") + mytheme
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3");
title(main = bquote(T[3]~"plot: Solids - Gamma distribution"), adj = 0)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T4");
title(main = bquote(T[4]~"plot: Solids - Gamma distribution"), adj = 0)
#####################
# Sulfate:
x ←X1$Sulfate[-rm.ind]
hist(x)
tmp ←logis_to_normal(x)
y ←tmp$z
tmp$param
re ←cox(matrix(sort(y)), x.dist = 0.001)
ggplot(data.frame(x = as.matrix(re$x),
                  a = as.matrix(re$a)), aes(x = x, y = a)) +
  geom_point(color = "steelblue3", shape = 1, size = 1.5) +
  ggtitle("Score plot: Sulfate - Logistic distribution") + xlab("y") +
```

```
  # geom_abline(intercept = 0, slope = 1) +
  coord_fixed() +
  ylab("Score function") + mytheme
ggplot(data.frame(y), aes(sample = y)) +
  stat_qq() + stat_qq_line(distribution = qnorm) +
  ggtitle("Q-Q plot")+
  ggtitle("Q - Q plot: Sulfate - Logistic distribution") +
  xlab("Theoretical quantile") + ylab("Sample quantile") + mytheme
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3");
title(main = bquote(T[3]~"plot: Sulfate - Logistic distribution"), adj = 0)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T4");
title(main = bquote(T[4]~"plot: Sulfate - Logistic distribution"), adj = 0)
##############################################################################
# Hardness:
x ←X1$Hardness[-rm.ind]
tmp ←logis_to_normal(x)
y ←tmp$z
tmp$param
re ←cox(matrix(sort(y)), x.dist = 0.001)
ggplot(data.frame(x = as.matrix(re$x),
                  a = as.matrix(re$a)), aes(x = x, y = a)) +
  geom_point(color = "steelblue3", shape = 1, size = 1.5) +
  ggtitle("Score plot: Hardness - Logistic distribution ") +
  # geom_abline(intercept = 0, slope = 1) +
  coord_fixed() + xlab("y") +
  ylab("Score function") + mytheme
ggplot(data.frame(y), aes(sample = y)) +
  stat_qq() + stat_qq_line(distribution = qnorm) +
  ggtitle("Q-Q plot") +
  ggtitle("Q - Q plot: Hardness - Logistic distribution") +
  xlab("Theoretical quantile") + ylab("Sample quantile") + mytheme
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3");
title(main = bquote(T[3]~"plot: Hardness - Logistic distribution"), adj = 0)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T4");
title(main = bquote(T[4]~"plot: Hardness - Logistic distribution"), adj = 0)
##############################################################################
# Conductivity
x ←X1$Conductivity[-rm.ind]
tmp ←gamma_to_normal(x)
y ←tmp$z
tmp$param
```

```
re ←cox(matrix(sort(y)), x.dist = 0.001)
ggplot(data.frame(x = as.matrix(re$x),
                  a = as.matrix(re$a)), aes(x = x, y = a)) +
  geom_point(color = "steelblue3", shape = 1, size = 1.5) +
  ggtitle("Score plot: Conductivity - Gamma distribution") +
  # geom_abline(intercept = 0, slope = 1) +
  coord_fixed() + xlab("y") +
  ylab("Score function") + mytheme
ggplot(data.frame(y), aes(sample = y)) +
  stat_qq() + stat_qq_line(distribution = qnorm) +
  ggtitle("Q-Q plot")+
  ggtitle("Q - Q plot: Conductivity - Gamma distribution") +
  xlab("Theoretical quantile") + ylab("Sample quantile") + mytheme
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T3");
title(main = bquote(T[3]~"plot: Conductivity - Gamma distribution"), adj = 0)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(y, method = "T4");
title(main = bquote(T[4]~"plot: Conductivity - Gamma distribution"), adj = 0)
```

Source Code C.6: T3_plot_counter_examples.R (Two Pathological Examples: T_3 and T_4)

```
# Two pathological examples about T3 and T4 plots.
# Example 2:
## logistic distrution
x ←seq(-5, 5, by = 0.01)
mu = 0; s ←sqrt(3)/pi
F1 ←function(x) integrate(dnorm, lower = -Inf, upper = x)$value
F2 ←function(x) 1/(1 + exp(-x/s))
# compare cdf
y ←seq(-4, 4, by = 0.01)
v1 ←sapply(y, F1)
v2 ←F2(y)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
plot(y, v1 - v2, type = "l", col = "grey0", ylim = c(-0.025, 0.025), lwd = 3,
     ylab = "Difference between Cumulative Distribution Functions",
     yaxt = 'n')
axis(side = 2, at = seq(-.025, .025, by = 0.005))
#compare pdf
y ←seq(-5, 5, by = 0.01)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
plot(y, dnorm(y), type = "l", col = "grey70",
     lty = 1, lwd = 3, ylim = c(0, .5),
     ylab = "Density functions")
lines(y, dlogis(y, location = 0, scale = s), col = "grey0", lty = 2, lwd = 3)
```

```
legend("topleft", c("N(0, 1)", expression(Logistic (0, sqrt(3)/pi )
)
),
merge = T, lwd = c(3, 3), lty = c(1, 2), y.intersp = 2,
col = c("grey70", "grey0"),
text.width = 3, seg.len = 2.5
)
#compare cgf
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
t ←seq(-1.5, 1.5, by = 0.01)
plot(t, t^2, type = "l", col = "grey70", lty = 1, lwd = 3,
     ylab = "Cumulant Generating Functions")
lines(t, lbeta(1 - s*t, 1 + s*t), col = "grey0", lty = 2, lwd = 3)
legend("top", c("N(0, 1)", expression(Logistic (0, sqrt(3)/pi )
)
),
merge = T, lwd = c(3, 3), lty = c(1, 2), y.intersp = 2,
text.width = 1, seg.len = 2.5, col = c("grey70", "grey0")
)
nx ←500
set.seed(1234)
x2 ←rlogis(nx, location = 0, scale = s)
library(PlotNormTest)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x2, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x2, method = "T4")

score_plot1D(sort(x1), x.dist = .0001 )
score_plot1D(sort(x2), x.dist = .0001 )

# Example 1
####
fdist ←function(x) dnorm(x) * (1 + .5*sin(2*base::pi*x))
Fdist ←function(x) sapply(
  x, function(x) integrate(
    fdist, lower = -Inf, upper = x
  )$value
)
rdist ←function(n){
  # Random generators
  u ←runif(n)
  sapply(u,
         function(u) uniroot(
```

```
          function(x) Fdist(x) - u,
          lower = -10, upper = 10)$root
  )
}
cgfdist ←function(t){
  .5*t^2 + log(1 + 0.5*exp(-2*base::pi^2) *sin(2*base::pi*t))
}
# Check the pdf are different
par(cex.axis = 1.2, cex.lab = 1.2,
    mar = c(4, 4.2, 2,1), cex.main = 1.2)
y ←seq(-3, 3, by = 0.01)
# par(mfrow = c(1, 2))
plot(y, dnorm(y), type = "l", col = "grey40",
     lty = 2, lwd = 3, ylim = c(0, .6),
     ylab = "Density Functions")
lines(y, unlist(lapply(y, fdist)), col = "grey0",lty = 1, lwd = 3)
legend("topleft",
       c(
         expression(phi(x)),
         expression(phi(x)*(1 + .5*sin(2*pi*x)))
       ),
       merge = T, lwd = c(3, 3), lty = c(2, 1),
       y.intersp = 1.5, text.width = 2, seg.len = 3
)

# check cgf are approximately equal
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
t ←seq(-1.5, 1.5, by = 0.01)
plot(t, .5*t^2, type = "l", col = "grey90", lty = 1, lwd = 3,
     ylim = c(-0.2, 1), ylab = "")
lines(t, cgfdist(t), col = "grey0",lty = 3, lwd = 3)
lines(t, (.5*t^2 - cgfdist(t))*1e8, col = "grey40", lty = 1, lwd = 3)
abline(h = 0, col = "grey70", lty = 3, lwd = 3)
text(.45,
     .9,
     expression(C[1](t) == .5*t^2 + log(1 + .5*e^{-2*pi^2}* sin(2*pi*t) )),
     cex = 1
)
text(1.3, .4, expression(C[2](t) == .5*t^2), cex = 1)
text(-1, -.18, expression((C[1](t) - C[2](t))%*%10^8), cex = 1)
# Check if correct generator
y ←rdist(1e4)
hist(y, probability = T)
EnvStats::epdfPlot(y, add = T, epdf.col = "red")
lines(seq(-5, 5, by = 0.01), fdist(seq(-5, 5, by = 0.01)), col = "blue")
```

```
set.seed(1234)
x2 ←rdist(nx)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x2, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(x2, method = "T4")


score_plot1D(sort(x1), x.dist = .0001)
score_plot1D(sort(x2), x.dist = .0001)
```

APPENDIX D

SOURCE CODE OF CHAPTER THREE

Source Code D.1: Kotz.R (Multvirate Kotz distribution)

```
## -- Filename: Kotz.R ###
## Multivariate Kotz distribution.
ftheta ←function(p, theta){
  if (p%%2 == 0){
    const ←(2^p)/base::pi * (factorial(p/2)^2)/factorial(p)
  } else {
    if (p == 1){
      const ←1/2
    } else {
      const ←(p/2) * prod((1:(p-1))[as.logical(1:(p-1) %% 2) == 1]) /
        prod((1:(p-1))[as.logical(1:(p-1) %% 2) == 0])
    }
  }
  return(const*(sin(theta)^p))
}
# ### Check above is the correct sample ####
# unlist(
# lapply(1:20,
# function(p) integrate(ftheta,
# lower = 0, upper = base::pi, p = p)$value
# )
# )
# p ←4
# x ←inv.cdf.theta(p, u = runif(1000))
# y ←seq(0, base::pi, length.out = 1e3)
# plot(y, ftheta(p, y))
# EnvStats::epdfPlot(x,add=TRUE, epdf.col = "red" ,epdf.lwd = 1 )
##############################
rmvKotz ←function(n, mu = rep(0, p), Sigma = diag(p)){
  ftheta ←function(p, theta){
    if (p%%2 == 0){
      const ←(2^p)/base::pi * (factorial(p/2)^2)/factorial(p)
    } else {
      if (p == 1){
        const ←1/2
      } else {
        const ←(p/2) * prod((1:(p-1))[as.logical(1:(p-1) %% 2) == 1]) /
          prod((1:(p-1))[as.logical(1:(p-1) %% 2) == 0])
      }
    }
    return(const*(sin(theta)^p))
  }
  cdf.theta ←function(p, theta){
    sapply(theta,
```

```
          function(x) integrate(ftheta, lower = 0, upper = x, p = p)$value
          )
  }
  inv.cdf.theta ←function(p, u){
    sapply(u,
          function(u) uniroot(function(theta) cdf.theta(p, theta) - u,
                              lower = 0, upper = base::pi)$root
          )
  }
  G ←chol(Sigma)
  p ←ncol(Sigma)
  r ←matrix(rgamma(n, shape = p, scale = 1), nrow = n, ncol = 1)
  #Generate \theta_{p-1}
  thetapm1 ←matrix(runif(n, 0, 2*base::pi), nrow = n, ncol = 1)
  if ( p > 2){
    #Generation of \theta_1 \dots \theta_{p - 2}
    u ←matrix(runif(n*(p-2)), ncol = p-2, nrow = n)
    theta ←matrix(sapply((p-2):1,
                         function(myp) inv.cdf.theta(myp, u = u[, myp]),
                         simplify = "array"), nrow = n, ncol = p-2)
  } else{
    theta ←c()
  }
  sin.tmp ←sin(cbind(theta, thetapm1))
  cos.tmp ←cbind(cos(cbind(theta, thetapm1)), rep(1, n))
  y ←array(NA, c(n, p))
  y[, 1] ←r * cos.tmp[, 1]
  for (i in 2:p){
    y[, i] ←r * cos.tmp[, i] * apply(matrix(sin.tmp[,1:(i - 1)], nrow = n),
                                     1, prod
                                     )
  }

  x ←y %*% t(G) + mu
  return(x)
}
#########################
dmvKotz ←function(x, mu = rep(0, p), Sigma = diag(p)){
  # Get the density function
  lcp ←lgamma(p/2) - p/2*log(2*base::pi) - lgamma(p)
  ltop ←-.5*log(det(Sigma)) - sqrt((t(x - mu) %*% solve(Sigma) %*% (x - mu)))
  lf ←ltop + lcp
  return(exp(lf))
}
########################
```

```
mleKotz ←function(mu0, Sigma0, X, maxit = 1e4){
  # Get the estimate of mu and Sigma for Kotz distribution
  # we have p + p(p+1)/2 params in the case of general \mu and Sigma
  n ←nrow(X); p ←ncol(X)
  myparams ←c(mu0, unlist(sapply(1:p, FUN= function(i) Sigma0[i, i:p])))
  mlogf ←function(params){
    # -logf
    mu ←params[1:p]
    Sigma ←array(NA, c(p, p))
    Sigma[lower.tri(Sigma, diag = T)] ←params[(p+1):length(params)]
    Sigma[upper.tri(Sigma, diag = F)] ←t(Sigma)[upper.tri(t(Sigma), diag = F)]
    eSigma ←eigen(Sigma)
    if(any(eSigma$values <= 0)){
      # covariance matrix is pd
      d ←eSigma$values; Q ←eSigma$vectors
      d[which(d <= 0)] ←0.0001
      D ←diag(d)
      Sigma ←Q %*% D%*% solve(Q)
    }
    mysum ←sum(
      apply(X, MARGIN = 1,
            FUN = function(y) sqrt((t(y - mu) %*% solve(Sigma) %*% (y - mu)))
            )
      )
    mf ←(n/2)*log(det(Sigma)) + mysum
    return(mf)
  }
  re ←optim(par = myparams, mlogf,
             control = list(maxit = maxit, abstol = 1e-10), method = "BFGS")
  is.converge ←ifelse(re$convergence == 0, T, F)
  hparams ←re$par
  hmu ←hparams[1:p]
  hSigma ←array(NA, c(p, p))
  hSigma[lower.tri(hSigma, diag = T)] ←hparams[(p+1):length(hparams)]
  hSigma[upper.tri(hSigma, diag = F)] ←
    t(hSigma)[upper.tri(t(hSigma), diag = F)]
  return(list(mu = hmu, Sigma = hSigma, Is.converge = is.converge))
}
```

Source Code D.2: CF_Multi_to_Uni (Multivairate normality test based on $n \times p$ sample).R

```
## --Filename: CF_Multi_to_Uni.R
library(PlotNormTest)
library(ggplot2)
source("distributions.R")
```

```
source("Kotz.R")
mytheme <-theme_bw() +
  theme(aspect.ratio = 1/1, panel.grid = element_blank(),
        axis.line = element_line(colour = "black"),
        axis.text=element_text(size=12),
        axis.title=element_text(size=14,face="bold"),
        legend.background = element_rect(
          size=0.5, linetype="solid"),
        legend.text = element_text(size=12))
##############################################
##### Normal ####
set.seed(1234)
nx <-200
p <-5
x <-MASS::mvrnorm(nx, mu = rep(0, p), diag(p))
df <-Multi.to.Uni(x)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot+
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot") +
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line()+
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv <-mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
sw <-stats::shapiro.test(df$x.new)
ad <-nortest::ad.test(df$x.new)
uniNormal1 <-c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
            Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
            Shapiro = (sw$p.value), # Shapiro - Wilk test
            "Anderson Darling" = ad$p.value
)
##############################################
##############################################
# Normal 2
set.seed(12345)
p <-5
sigma2 <-2; rho <-0.3;
Sigma <-sigma2 * ((1 -rho)*diag(p) +
```

```
                    rho *matrix(rep(1, p^2), nrow = p, byrow = T))
x ←MASS::mvrnorm(nx, mu = 1:5, Sigma)
df ←Multi.to.Uni(x)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot") +
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv ←mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks ←stats::ks.test(df$x.new, "pnorm")
sw ←stats::shapiro.test(df$x.new)
ad ←nortest::ad.test(df$x.new)
uniNormal2 ←c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
            Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
            # ks$p.value >= alpha, # Kolmogorov-Smirnov test
            Shapiro = (sw$p.value), # Shapiro - Wilk test
            "Anderson Darling" = ad$p.value
)
##############################
### Copulas ######
p ←5
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * ((1 -rho)*diag(p) +
                    rho *matrix(rep(1, p^2), nrow = p, byrow = T))
set.seed(120)
x ←MASS::mvrnorm(n =nx, mu = rep(0, p), S = Sigma)
U ←pnorm(x)
mydata ←qgamma(U, shape = 9, scale = 0.5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
df ←Multi.to.Uni(mydata)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot") +
  ylab("Score function")
# Q-Q plot
```

```
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv <-mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks <-stats::ks.test(df$x.new, "pnorm")
sw <-stats::shapiro.test(df$x.new)
ad <-nortest::ad.test(df$x.new)
uniCopula <-c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
                Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
                # ks$p.value >= alpha, # Kolmogorov-Smirnov test
                Shapiro = (sw$p.value), # Shapiro - Wilk test
                "Anderson Darling" = ad$p.value
)
##########################
## Mixture normal
set.seed(1234)
mydata <-mixture_normal(e = .5, n = nx, S1 = diag(p), S2 = 4*diag(p),
                        mu1 = rep(0, p), mu2 = rep(3, p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
df <-Multi.to.Uni(mydata)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot")+
  ylab("Score function")

# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv <-mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks <-stats::ks.test(df$x.new, "pnorm")
sw <-stats::shapiro.test(df$x.new)
ad <-nortest::ad.test(df$x.new)
uniMix <-c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
```

```
                Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
                # ks$p.value >= alpha, # Kolmogorov-Smirnov test
                Shapiro = (sw$p.value), # Shapiro - Wilk test
                "Anderson Darling" = ad$p.value)
######################
# Lomax
set.seed(1234)
mydata ←-NonNorMvtDist::rmvlomax(nx, parm =1, parm2 = rep(1, 5))
df ←-Multi.to.Uni(mydata)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot") +
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv ←-mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks ←-stats::ks.test(df$x.new, "pnorm")
sw ←-stats::shapiro.test(df$x.new)
ad ←-nortest::ad.test(df$x.new)
uniLomax ←-c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
                Kurtosis = as.numeric(as.character(pv[1, ])), # kurtosis
                # ks$p.value >= alpha, # Kolmogorov-Smirnov test
                Shapiro = (sw$p.value), # Shapiro - Wilk test
                "Anderson Darling" = ad$p.value)

####################
# t5
set.seed(1234)
mydata ←-mvtnorm::rmvt(nx, 3/5 *diag(p), 5)
df ←-Multi.to.Uni(mydata)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot")+
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
```

```
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv <-mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks <-stats::ks.test(df$x.new, "pnorm")
sw <-stats::shapiro.test(df$x.new)
ad <-nortest::ad.test(df$x.new)
unit5 <-c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
            Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
            # ks$p.value >= alpha, # Kolmogorov-Smirnov test
            Shapiro = (sw$p.value), # Shapiro - Wilk test
            "Anderson Darling" = ad$p.value)
########################
#t15
set.seed(333)
mydata <-mvtnorm::rmvt(nx, 13/15*diag(p), 15)

df <-Multi.to.Uni(mydata)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot")+
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv <-mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks <-stats::ks.test(df$x.new, "pnorm")
sw <-stats::shapiro.test(df$x.new)
ad <-nortest::ad.test(df$x.new)
unit15 <-c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
          Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
          # ks$p.value >= alpha, # Kolmogorov-Smirnov test
          Shapiro = (sw$p.value), # Shapiro - Wilk test
          "Anderson Darling" = ad$p.value)
```

```
##########################
p ←5
set.seed(1234)
mydata ←rmvKotz(n = nx, mu = rep(0, p), Sigma = 1/(p+1)*diag(p))
df ←Multi.to.Uni(mydata)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot") +
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv ←mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
# ks ←stats::ks.test(df$x.new, "pnorm")
sw ←stats::shapiro.test(df$x.new)
ad ←nortest::ad.test(df$x.new)
uniKotz ←c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
          Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
          # ks$p.value >= alpha, # Kolmogorov-Smirnov test
          Shapiro = (sw$p.value), # Shapiro - Wilk test
          "Anderson Darling" = ad$p.value)
####### Hypothesis Testing ################
mydf ←round(data.frame("Standard Normal" = uniNormal1,
                       "Normal" = uniNormal2,
                       "Mixuture Normal" = uniMix,
                       "Gaussian copulas" = uniCopula,
                       "Lomax" = uniLomax,
                       "t(5)" = unit5,
                       "t(15)" = unit15,
                       "kotz" = uniKotz), 4)
library(xtable)
mydf←xtable::xtable(t(mydf), caption= "Tranformation to univariate data",
           align=c("|c","|c","|c","|c","|c|"),digits = 4)
#####################################
#####################################
#####################################
#### Outlier detection using score #####
#### Outlier 1#####
```

```
library(MASS)
set.seed(1234)
nx ←200
p ←5
mu ←rep(0, 5); Sigma = diag(5)
x ←MASS::mvrnorm(nx, mu = mu, Sigma)
#outlier 1
set.seed(9090)
out ←MASS::mvrnorm(2, mu = mu + 2, Sigma = Sigma + 3)
(ind.ori← sample.int(nx, 2))
round(out, 3)
x.out ←x
x.out[ind.ori,] ←out
df ←Multi.to.Uni(x.out)
re ←score_plot1D(x = df$x.new, ori.index = df$ind, x.dist = .001, lambda = .5)
re$out
re$plot + mytheme+ theme(legend.position = "top")+ xlab("y") +
  ylab("Score function")
# #outlier 2
set.seed(4321)
out ←matrix(c(mvrnorm(1, mu = mu + 2, Sigma = Sigma+3),
              mvrnorm(1, mu = mu - 2, Sigma = Sigma+3)), byrow = T, nrow = 2)
(ind.ori← sample.int(nx, 2))
round(out, 3)
x.out ←x
x.out[ind.ori,] ←out
df ←Multi.to.Uni(x.out)
re ←score_plot1D(x = df$x.new, ori.index = df$ind, x.dist = .001)
re$out
re$plot + mytheme+ theme(legend.position = "top")+ xlab("y")+
  ylab("Score function")
#outlier 3:
set.seed(123)
Sigma1 ←diag(p)
Sigma1[5, 5] ←4
Sigma1[4, 4] ←4
Sigma1[4, 5] ←Sigma1[5, 4] ←3
mu1 ←mu
mu1[c(4, 5)] ←rep(6, 2)
out ←MASS::mvrnorm(2, mu = mu1, Sigma = Sigma1)
(ind.ori← sample.int(nx, 2))
round(out, 3)
x.out ←x
x.out[ind.ori,] ←out
df ←Multi.to.Uni(x.out)
```

```
re ←score_plot1D(x = df$x.new, ori.index = df$ind, x.dist = .001)
re$out
re$plot + mytheme+ theme(legend.position = "top")+ xlab("y")+
  ylab("Score function")
# Outlier 4:
set.seed(4321)
out ←NonNorMvtDist::rmvlomax(2, parm =1, parm2 = rep(1, 5))
(ind.ori← sample.int(nx, 2))
round(out, 3)
x.out ←x
x.out[ind.ori,] ←out
df ←Multi.to.Uni(x.out)
re ←score_plot1D(x = df$x.new, ori.index = df$ind, x.dist = .001)
re$out
re$plot + mytheme+ theme(legend.position = "top")+ xlab("y")+
  ylab("Score function")
###### Empirical size alpha test ######
# Do not run
m ←4e3
sample_size ←c(24, seq(50, 200, by = 25), seq(200, 500, by = 50))
# sample_size ←c(50, 60, 80)
myp ←2:10
to_Uni_type1_1 ←to_Uni_type1_2 ←to_Uni_type1_3 ←to_Uni_type1_4 ←
  array(dim = c(length(myp), length(sample_size)))
rownames(to_Uni_type1_1) ←rownames(to_Uni_type1_2) ←
  rownames(to_Uni_type1_3) ←rownames(to_Uni_type1_4) ←as.character(myp)
colnames(to_Uni_type1_1) ←colnames(to_Uni_type1_2) ←
  colnames(to_Uni_type1_3) ←colnames(to_Uni_type1_4) ←
  as.character(sample_size)
alpha ←.05
set.seed(1234)
for (i in 1:length(myp)){
  for (j in 1:length(sample_size)){
    print(c(i, j))
    re ←replicate(m, expr = {
      p ←myp[i]
      nx ←sample_size[j]
      x ←MASS::mvrnorm(nx, rep(0, p), diag(p))
      df ←Multi.to.Uni(x)
      pv ←mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
      # ks ←stats::ks.test(df$x.new, "pnorm")
      sw ←stats::shapiro.test(df$x.new)
      ad ←nortest::ad.test(df$x.new)
      as.numeric(c(
        Skewness = (as.numeric(as.character(pv[1, ])) >= alpha), # skewness
```

```
      Kurtosis = (as.numeric(as.character(pv[2, ])) >= alpha), # kurtosis
      Shapiro = ((sw$p.value) >= alpha), # Shapiro - Wilk test
      "Anderson Darling" = (ad$p.value >= alpha)
    )
    )
  }
  )
  tmp ←apply(re, 1, mean)
  to_Uni_type1_1[i, j] ←tmp[1]
  to_Uni_type1_2[i, j] ←tmp[2]
  to_Uni_type1_3[i, j] ←tmp[3]
  to_Uni_type1_4[i, j] ←tmp[4]
 }
}
######## Plot Type 1 ############
# Change number of "to_Uni_type1_... " to plot
library(palettes)
library(ggplot2)
library(scales)
# Contour plot of Empirical Type I as a function of dimension and sample size.
meo ←data.frame(1 - to_Uni_type1_4[, 1:10]) %>%
 mutate(Dimension = as.numeric(row.names(.)))
meo1 ←reshape2::melt(meo, id.vars = "Dimension") %>%
 mutate(Size = as.numeric(sub("X", "", variable)))
ggplot(meo1, aes(x = Dimension, y = Size)) +
 geom_contour_filled(aes(z = value), breaks = c(0, 0.045, 0.085, .1, .3,.65)) +
 scale_y_continuous(breaks= c(seq(25, 200, by = 25), 250)) + theme_bw() +
 scale_fill_palette_d(RColorBrewer::brewer.pal(4, "Blues")[1:6]) +
 ylab("Sample Size") +
 theme(aspect.ratio = 1/1, panel.grid = element_blank(),
       axis.line = element_line(),
       panel.border = element_blank(),
       axis.text=element_text(size=12),
       axis.title=element_text(size=14,face="bold"),
       legend.background = element_rect(
         size=0.5, linetype="solid"),
       legend.text = element_text(size=11)) +
 theme(legend.position = c(.2, .8))
######################### Experience on data set ##########################
###### cork #########
source("Data_cork.R")
df ←Multi.to.Uni(cork)
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
 theme(legend.position = "none")+ xlab("y") +
```

```
  ggtitle("Score plot")+
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv ←mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
sw ←stats::shapiro.test(df$x.new)
ad ←nortest::ad.test(df$x.new)
c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
  Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
  Shapiro = (sw$p.value), # Shapiro - Wilk test
  "Anderson Darling" = ad$p.value)
###### Water #######
source("Data_water.R")
mycol ←c("Organic_carbon", "Turbidity", "Trihalomethanes")
X1.3 ←X1[, mycol]
df ←Multi.to.Uni(as.matrix(X1.3))
# Cox
score_plot1D(df$x.new, ori.index = df$ind, x.dist = .001)$plot +
  theme(legend.position = "none")+ xlab("y") +
  ggtitle("Score plot") +
  ylab("Score function")
# Q-Q plot
ggplot2::ggplot(df, aes(sample = x.new)) + stat_qq() + stat_qq_line() +
  mytheme + ggtitle("Q-Q plot") +
  xlab("Theoretical quantile") + ylab("Empirical quantile")
#T3 and T4
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T3")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
dhCGF_plot1D(df$x.new, method = "T4")
pv ←mvnormalTest::mardia(df$x.new)$mv.test["p-value"]
sw ←stats::shapiro.test(df$x.new)
ad ←nortest::ad.test(df$x.new)
c(Skewness = as.numeric(as.character(pv[1, ])), # skewness
  Kurtosis = as.numeric(as.character(pv[2, ])), # kurtosis
  Shapiro = (sw$p.value), # Shapiro - Wilk test
  "Anderson Darling" = ad$p.value)
```

Source Code D.3: CF_linear_transform.R (A test based on best linear transformations)

```
## --Filename: CF_linear_transform.R
library(MvNormTest)
source("Kotz.R")
source("distributions.R")
m ←8e3;
sample_size ←c(24, 30, 60, 100, 200, 300, 500, 800)
myp ←2:10
lst_1 ←array(dim = c(length(myp)*2, length(sample_size)))
lst_2 ←array(dim = c(length(myp)*2, length(sample_size)))
lst_3 ←array(dim = c(length(myp)*2, length(sample_size)))
# lst_199 ←array(dim = c(length(myp), length(sample_size)))
rownames(lst_1)← rownames(lst_2) ←
  rownames(lst_3) ←as.character(rep(myp, each = 2))
colnames(lst_1) ←colnames(lst_2) ←
  colnames(lst_3) ←as.character(sample_size)
for (i in 1:length(myp)){
  for (j in 1:length(sample_size)) {
    print(c("skewness", i, j))
    p ←myp[i]
    nx ←sample_size[j]
    re ←replicate(m, expr = {
      x ←MASS::mvrnorm(nx, rep(0, p), diag(p))
      linear_transform(x, method = "skewness")$max_result
    }
    )
    lst_1[2*i - 1, j] ←quantile(re, .95)
    lst_1[2*i, j] ←quantile(re, .99)
  }
}
#################
for (i in 1:length(myp)){
  for (j in 1:length(sample_size)) {
    print(c("kurtosis", i, j))
    p ←myp[i]
    nx ←sample_size[j]
    re ←replicate(m, expr = {
      x ←MASS::mvrnorm(nx, rep(0, p), diag(p))
      linear_transform(x, method = "kurtosis")$max_result
    }
    )
    lst_2[2*i - 1, j] ←quantile(re, .95)
    lst_2[2*i, j] ←quantile(re, .99)
  }
}
```

```
################
for (i in 1:length(myp)){
  for (j in 1:length(sample_size)) {
    print(c("both", i, j))
    p ←myp[i]
    nx ←sample_size[j]
    re ←replicate(m, expr = {
      x ←MASS::mvrnorm(nx, rep(0, p), diag(p))
      linear_transform(x, method = "both")$max_result
    }
    )
    lst_3[2*i - 1, j] ←quantile(re, .95)
    lst_3[2*i, j] ←quantile(re, .99)
  }
}
################

# Data Experiment
###############################
###############################
myre ←array(dim = c(8, 3))
colnames(myre) ←c(" Skewness", "Kurtosis", "Both")
# Data experiment
##### Normal ####
set.seed(1234)
nx ←200
p ←5
x ←MASS::mvrnorm(nx, mu = rep(0, p), diag(p))
(c(linear_transform(x, method = "skewness")$max_result,
   linear_transform(x, method = "kurtosis")$max_result,
   linear_transform(x, method = "both")$max_result))
myre[1, ] ←c(linear_transform(x, method = "skewness")$max_result,
             linear_transform(x, method = "kurtosis")$max_result,
             linear_transform(x, method = "both")$max_result)
######################
set.seed(12345)
p ←5
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * (
  (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
  )
x ←MASS::mvrnorm(nx, mu = 1:5, Sigma)
myre[2, ] ←c(linear_transform(x, method = "skewness")$max_result,
             linear_transform(x, method = "kurtosis")$max_result,
             linear_transform(x, method = "both")$max_result)
```

```
##########################################################################
### Copulas ######
p ←5
nx ←nx
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * (
  (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
  )
set.seed(120)
x ←MASS::mvrnorm(n =nx, mu = rep(0, p), S = Sigma)
U ←pnorm(x)
mydata ←qgamma(U, shape = 9, scale = 0.5)
myre[3, ] ←c(linear_transform(mydata, method = "skewness")$max_result,
             linear_transform(mydata, method = "kurtosis")$max_result,
             linear_transform(mydata, method = "both")$max_result)
###### #######
p ←5
set.seed(1234)
mydata ←mvtnorm::rmvt(nx, 3/5 *diag(p), 5)
(c(linear_transform(mydata, method = "skewness")$max_result,
   linear_transform(mydata, method = "kurtosis")$max_result,
   linear_transform(mydata, method = "both")$max_result))
myre[4, ] ←c(linear_transform(mydata, method = "skewness")$max_result,
             linear_transform(mydata, method = "kurtosis")$max_result,
             linear_transform(mydata, method = "both")$max_result)
########################
p ←5
set.seed(333)
mydata ←mvtnorm::rmvt(nx, 13/15*diag(p), 15)
(c(linear_transform(mydata, method = "skewness")$max_result,
   linear_transform(mydata, method = "kurtosis")$max_result,
   linear_transform(mydata, method = "both")$max_result))
myre[5, ] ←c(linear_transform(mydata, method = "skewness")$max_result,
             linear_transform(mydata, method = "kurtosis")$max_result,
             linear_transform(mydata, method = "both")$max_result)
########################
p ←5
set.seed(1234)
mydata ←NonNorMvtDist::rmvlomax(nx, parm =1, parm2 = rep(1, 5))
myre[6, ] ←c(linear_transform(mydata, method = "skewness")$max_result,
             linear_transform(mydata, method = "kurtosis")$max_result,
             linear_transform(mydata, method = "both")$max_result)
##########################
p ←5
set.seed(1234)
```

```
mydata ←rmvKotz(n = nx, mu = rep(0, p), Sigma = 1/(p+1)*diag(p))
myre[7, ] ←c(linear_transform(mydata, method = "skewness")$max_result,
             linear_transform(mydata, method = "kurtosis")$max_result,
             linear_transform(mydata, method = "both")$max_result)
############################
p ←5;
# sigma2 ←2; rho ←0.4;
# Sigma ←sigma2 * (
# (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
# )
set.seed(1234)
mydata ←mixture_normal(e = .5, n = nx, S1 = diag(p),
                       S2 = 4*diag(p), mu1 = rep(0, p), mu2 = rep(3, p))
myre[8, ] ←c(linear_transform(mydata, method = "skewness")$max_result,
             linear_transform(mydata, method = "kurtosis")$max_result,
             linear_transform(mydata, method = "both")$max_result)
########################################
############ Experiment on real dataset #################
# cork
source("Datas_cork.R")
round(c(linear_transform(cork, method = "skewness")$max_result,
       linear_transform(cork, method = "kurtosis")$max_result,
       linear_transform(cork, method = "both")$max_result), 4)
# Water
source("Data_water.R")
mycol ←c("Organic_carbon", "Turbidity", "Trihalomethanes")
X1.3 ←X1[, mycol]
round(c(linear_transform(as.matrix(X1.3), method = "skewness")$max_result,
       linear_transform(as.matrix(X1.3), method = "kurtosis")$max_result,
       linear_transform(as.matrix(X1.3), method = "both")$max_result), 4)
```

APPENDIX E

SOURCE CODE OF CHAPTER FOUR

Source Code E.1: CF_mt3 (MT_3 plot)

```
## --Filename: CF_mt3.R
#### Ex of test statsitic #####
library(PlotNormTest)
source("Kotz.R")
source("distributions.R")
###############
#3D plot
library(plotly); library(viridis)
myx ←seq(-1, 1, 0.01)
myy ←seq(-1, 1, 0.01)
bigt ←expand.grid(x = myx, y = myy)
set.seed(1234); nx ←200; p ←2; lT ←p + choose(p, 2)*2 + choose(p, 3)
x ←MASS::mvrnorm(200, mu = rep(0, 2), diag(2));
# set.seed(1458)
# x ←NonNorMvtDist::rmvf(nx, df = rep(3, p+1))
s ←sqrt(nx) * unlist(
  apply(bigt, MARGIN = 1,
        FUN = function(myt) (sqrt(lT))*mean(d3hCGF(myt = as.numeric(myt), x = x)
                                            )
        )
  )
mydf ←data.frame(bigt)
mydf$L3 ←s
colnames(mydf) ←c("T1", "T2", "L3")
mydf ←mydf %>% mutate(group = ifelse(mydf$T1 == mydf$T2, T, F))
mycolor ←matrix(c(0, "#eff3ff",
                  0.25, "#eff3ff",
                  0.5, "#bdd7e7",
                  0.75, "#6baed6",
                  1, "#0047AB"), nrow = 5, byrow = T)

plot_ly() %>%
  add_trace(data = mydf, x = ~T1, y = ~T2, z = ~L3,
            type = "scatter3d",mode = "markers",
            marker =
              list(color =~L3,
                   colorscale = mycolor,
                   showscale = T,
                   reversescale = F,
                   colorbar = list(tickfont = list(size = 15),
                                   dtick = 5 , tick0 = -6)
              )
  ) %>%
  # Uncomment below to add trace
```

```
  add_trace(data = mydf %>% mutate(L3 = ifelse(group, L3, NA)),
          x = ~T1,y = ~T2, z = ~L3,type = "scatter3d",mode = "line+markers",
          marker = list(color = "black", showscale = F, size = 10,
                        symbol = "diamond-wide")) %>%
  layout(
    showlegend = F,
    scene = list(
      zaxis = list(
        range = c(-20, 20),
        tickfont = list(size = 15),
        title = list(text = "L3", font = list(size = 15)),
        tick0 = -20, dtick = 20, showticklabels = FALSE
      ),
      xaxis = list(
        tickfont = list(size = 15),
        title = list(text = "T1", font = list(size = 15)),
        tick0 = -1, dtick = 1
      ),
      yaxis = list(
        tickfont = list(size = 15),
        title = list(text = "T2", font = list(size = 15)),
        tick0 = -1, dtick = 1
      )
    )
  )
## Bivariate F
set.seed(1458)
x ←NonNorMvtDist::rmvf(nx, df = rep(3, p+1))
s ←sqrt(nx) * unlist(apply(bigt, MARGIN = 1,
                           FUN = function(myt) {
                             (sqrt(lT))*mean(d3hCGF(myt = as.numeric(myt),
                                                    x = x))} ) )
mydf ←data.frame(bigt)
mydf$L3 ←s
colnames(mydf) ←c("T1", "T2", "L3")

mydf ←mydf %>% mutate(group = ifelse(mydf$T1 == mydf$T2, T, F))

mycolor ←matrix(c(0, "#eff3ff",
                  0.25, "#eff3ff",
                  0.5, "#bdd7e7",
                  0.75, "#6baed6",
                  1, "#0047AB"), nrow = 5, byrow = T)

plot_ly() %>%
```

```
  add_trace(data = mydf, x = ~T1, y = ~T2, z = ~L3,
            type = "scatter3d",mode = "markers",
            marker =
              list(color =~L3,
                   colorscale = mycolor, showscale = T, reversescale = F,
                   colorbar = list(tickfont = list(size = 15),
                                   dtick = 1500 , tick0 = -1500 )
                   )
  ) %>%
  # Uncomment to add trace
  add_trace(data = mydf %>% mutate(L3 = ifelse(group, L3, NA)),
            x = ~T1,y = ~T2, z = ~L3,type = "scatter3d",mode = "line+markers",
            marker = list(color = "black", showscale = F, size = 10,
                          symbol = "diamond-wide")) %>%
  layout(
    showlegend = F,
    scene = list(
      zaxis = list(
        range = c(-2200, 2200),
        tickfont = list(size = 15),
        title = list(text = "L3", font = list(size = 15)),
        tick0 = -2000, dtick = 2000, showticklabels = FALSE
      ),
      xaxis = list(
        tickfont = list(size = 15),
        title = list(text = "T1", font = list(size = 15)),
        tick0 = -1, dtick = 1
      ),
      yaxis = list(
        tickfont = list(size = 15),
        title = list(text = "T2", font = list(size = 15)),
        tick0 = -1, dtick = 1
      )
    )
  )
#####################################################
# Normal
set.seed(1234)
nx ←200
p ←5;
bigt ←seq(-1, 1, by = 0.05)

lT ←p + choose(p, 2)*2 + choose(p, 3);l ←rep(1/sqrt(lT), lT)
x ←MASS::mvrnorm(nx, mu = rep(0, p), diag(p));
```

```
v3 ←lapply(seq(-1, 1, by = 0.05)/sqrt(p),
           function(u) d3hCGF(myt = rep(u, p), x = x))
L3.normal1 ←unlist(lapply(v3, function(v) sqrt(nx) * sum(l*v)))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
plot(bigt, L3.normal1, ylim = c(-20, 20),
     col = "blue", lty = 1, type = "l", lwd = 2,
     ylab = bquote(L[3]),
     xlab = "t")
title(main = bquote(MT[3] ~"plot"),
      adj = 0)

# mt3_normal_nobands
set.seed(1234)
mydata ←NonNorMvtDist::rmvf(nx, df = rep(3, p+1))
mydata ←x
v3 ←lapply(bigt/sqrt(p),function(u) d3hCGF(myt = rep(u, p), x = mydata))
L3.mydata ←unlist(lapply(v3, function(v) sqrt(nx) * sum(l*v)))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)

plot(bigt, L3.normal1, col = "blue", lty = 1, type = "l", lwd = 2,
     ylim = c(-max(abs(L3.mydata)), max(abs(L3.mydata))),
     ylab = bquote(L[3]),
     xlab = "t")
title(main = bquote(MT[3] ~"plot"),
      adj = 0)

##### Normal ####
set.seed(1234)
nx ←200
p ←5
x ←MASS::mvrnorm(nx, mu = rep(0, p), diag(p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(x)

v3 ←lapply(bigt/sqrt(p),function(u) d3hCGF(myt = rep(u, p), x = x))
L3.normal1 ←unlist(lapply(v3, function(v) sqrt(nx) * sum(l*v)))

######################
set.seed(12345)
p ←5
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * (
  (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
  )
x ←MASS::mvrnorm(nx, mu = 1:5, Sigma)
```

```
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(x)

v3 ←lapply(bigt/sqrt(p),function(u) d3hCGF(myt = rep(u, p), x = x))
L3.normal2 ←unlist(lapply(v3, function(v) sqrt(nx) * sum(l*v)))
############################
### Copulas ######
p ←5
nx ←nx
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * (
  (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
  )
set.seed(120)
x ←MASS::mvrnorm(n =nx, mu = rep(0, p), S = Sigma)
U ←pnorm(x)
mydata ←qgamma(U, shape = 9, scale = 0.5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L3.normal2, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands",
               expression(L[3]), as.expression(bquote(L[3] ~"- Normal")) ),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)

###### #######
set.seed(1234)
mydata ←mvtnorm::rmvt(nx, 3/5 *diag(p), 5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L3.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands",
               expression(L[3]), as.expression(bquote(L[3] ~"- Normal"))),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
########################
p ←5
set.seed(333)
mydata ←mvtnorm::rmvt(nx, 13/15*diag(p), 15)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L3.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands",
```

```
              expression(L[3]), as.expression(bquote(L[3] ~"- Normal")) ),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)

######################
p ←5
set.seed(1234)
mydata ←NonNorMvtDist::rmvlomax(nx, parm =1, parm2 = rep(1, 5))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L3.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands", expression(L[3]) ),
       lty = c(6, 1), col = c("orange", "blue"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
#########################
p ←5
set.seed(1234)
mydata ←rmvKotz(n = nx, mu = rep(0, p), Sigma = 1/(p+1)*diag(p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L3.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands",
              expression(L[3]), as.expression(bquote(L[3] ~"- Normal")) ),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
#########################
p ←5;
nx ←nx
set.seed(1234)
mydata ←mixture_normal(e = .5, n = nx, S1 = diag(p), S2 = 4*diag(p),
                       mu1 = rep(0, p), mu2 = rep(3, p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L3.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands",
              expression(L[3]), as.expression(bquote(L[3] ~"- Normal")) ),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
########################################
############ Experiment on real dataset #
# cork
```

```
source("Data_cork.R")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d3hCGF_plot(cork)
# normal
source("Data_water.R")
mycol ←c("Organic_carbon", "Turbidity", "Trihalomethanes")
X1.3 ←X1[, mycol]
d3hCGF_plot(as.matrix(X1.3))
```

Source Code E.2: CF_mt4 (MT_4 plot)

```
## -- Filename: CF_mt4.R
library(MvNormTest)
source("Kotz.R")
source("distributions.R")
p ←5; n ←200
bigt ←seq(-1, 1, by = 0.05)
lT4 ←l_dhCGF(p)[[2]]; l ←rep(1/sqrt(lT4), lT4)
##### Normal ####
set.seed(1234)
nx ←200
p ←5
x ←MASS::mvrnorm(nx, mu = rep(0, p), diag(p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(x)
v4 ←lapply(bigt/sqrt(p),function(u) d4hCGF(myt = rep(u, p), x = x))
L4.normal1 ←unlist(lapply(v4, function(v) sqrt(nx) * sum(l*v)))
#######################
set.seed(12345)
p ←5
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * (
  (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
  )
x ←MASS::mvrnorm(nx, mu = 1:5, Sigma)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(x)
v4 ←lapply(bigt/sqrt(p),function(u) d4hCGF(myt = rep(u, p), x = x))
L4.normal2 ←unlist(lapply(v4, function(v) sqrt(nx) * sum(l*v)))
############################
### Copulas ######
p ←5
nx ←nx
sigma2 ←2; rho ←0.3;
Sigma ←sigma2 * (
```

```
  (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T)
  )
set.seed(120)
x ←MASS::mvrnorm(n =nx, mu = rep(0, p), S = Sigma)
U ←pnorm(x)
mydata ←qgamma(U, shape = 9, scale = 0.5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L4.normal2, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands", expression(L[4]),
               as.expression(bquote(L[4] ~"- Normal"))),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
###### #######
p ←5
set.seed(1234)
mydata ←mvtnorm::rmvt(nx, 3/5 *diag(p), 5)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L4.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands", expression(L[4]),
               as.expression(bquote(L[4] ~"- Normal")) ),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
########################
p ←5
set.seed(333)
mydata ←mvtnorm::rmvt(nx, 13/15*diag(p), 15)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L4.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands", expression(L[4]),
               as.expression(bquote(L[4] ~"- Normal")) ),
       lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
       lwd = c(2, 2),
       merge = TRUE, y.intersp = 1.5, text.width = .6)
########################
p ←5
set.seed(1234)
mydata ←NonNorMvtDist::rmvlomax(nx, parm =1, parm2 = rep(1, 5))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L4.normal1, lty =3 , lwd = 3, col = "firebrick3")
```

```
legend("top", c("Probability bands", expression(L[4]) ),
      lty = c(6, 1), col = c("orange", "blue"),
      lwd = c(2, 2),
      merge = TRUE, y.intersp = 1.5, text.width = .6)
#########################
p ←5
set.seed(1234)
mydata ←rmvKotz(n = nx, mu = rep(0, p), Sigma = 1/(p+1)*diag(p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L4.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands", expression(L[4]),
               as.expression(bquote(L[4] ~"- Normal")) ),
      lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
      lwd = c(2, 2),
      merge = TRUE, y.intersp = 1.5, text.width = .6)
#########################
p ←5;
# sigma2 ←2; rho ←0.4;
# Sigma ←sigma2 * (
# (1 -rho)*diag(p) + rho *matrix(rep(1, p^2), nrow = p, byrow = T))
nx ←nx
set.seed(1234)
mydata ←mixture_normal(e = .5, n = nx, S1 = diag(p), S2 = 4*diag(p),
                       mu1 = rep(0, p), mu2 = rep(3, p))
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(mydata)
lines(seq(-1, 1, by = 0.05), L4.normal1, lty =3 , lwd = 3, col = "firebrick3")
legend("top", c("Probability bands", expression(L[4]),
               as.expression(bquote(L[4] ~"- Normal")) ),
      lty = c(6, 1), col = c("orange", "blue", "firebrick3"),
      lwd = c(2, 2),
      merge = TRUE, y.intersp = 1.5, text.width = .6)
#########################################
############ Experiment on real dataset #################
# cork
source("Data_cork.R")
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.2, 2,1), cex.main = 1.2)
d4hCGF_plot(cork)
title(bquote(MT[4] ~ "plot") , adj = 0)
# water
source("Data_water.R")
mycol ←c("Organic_carbon", "Turbidity", "Trihalomethanes")
X1.3 ←X1[, mycol]
d4hCGF_plot(as.matrix(X1.3))
```

```
title(bquote(MT[4] ~ "plot") , adj = 0)
```

APPENDIX F

SOURCE CODE OF CHAPTER FIVE

Source Code F.1: covLomax.R (Computation of multivariate Lomax distribution)

```
## --Filename: covLomax.R
covLomax ←function(a, theta){
  # Get covariance matrix of multivariate lomax distribution
  # a >= 2 to ensure finite covariance
  p ←length(theta)
  S ←array(dim = c(p, p))
  mu ←c()
  for (i in 1:p){
    tmpi ←(gamma(a - 1)*gamma(2))/(gamma(a) * theta[i])
    mu ←c(mu, tmpi)
    tmpi2 ←(gamma(a - 2)*gamma(3))/(gamma(a)*theta[i]^2)
    S[i, i] ←tmpi2 - tmpi*tmpi

    for (j in (i+1):p){
      if (j > p){break}
      tmpj ←(gamma(a - 1)*gamma(2))/(gamma(a) * theta[j])
      tmp1 ←(gamma(a - 2)*gamma(2)^2)/(gamma(a) * theta[i] *theta[j])
      S[i, j] ←S[j, i] ←tmp1 - tmpi*tmpj
    }
  }

  return(list(mu = mu, Sigma = S))
}

(S ←covLomax(3, rep(1, 5))$Sigma)
(mu ←covLomax(3, rep(1, 5))$mu)

x ←NonNorMvtDist::rmvlomax(10000, parm1 = 3, parm2 = rep(1, 5))
cov(x)
colMeans(x)
```

Source Code F.2: discr_Distance.R (Group classification using "majority rule")

```
## --Filename: discr_Distance.R
## Group discriminations my majority classification rules
########################################
discr.linear ←function(X1 = NULL, X2 = NULL, U, true.params = T,
                       params = ifelse(
                         true.params, list(mu1, mu2, Sigma), NULL)
                       ){
  nU ←nrow(U)
```

```
  if (!true.params){
    warning("Unknown true parameter, estimators will be uses")
    n1 ←nrow(X1)
    n2 ←nrow(X2)
    mu1 ←colMeans(X1)
    mu2 ←colMeans(X2)

    S1 ←cov(X1)
    S2 ←cov(X2)
    Sigma ←((n1 - 1) * S1 + (n2 - 1) *S2)/(n1 + n2 - 2)
  } else {
    warning("Using true parameters")
    mu1 ←params$mu1
    mu2 ←params$mu2
    Sigma ←params$Sigma
  }
  # assume that the two training set have equal variance
  a ←solve(Sigma) %*% (mu1 - mu2)
  myu ←U %*% a - as.numeric(.5 * t(a) %*% (mu1 + mu2))
  re ←ifelse(myu >= 0, "Population 1", "Population 2")
  num1 ←sum(re == "Population 1")
  deci ←ifelse(num1 == nU/2, "Can not decide",
               ifelse(num1 > nU/2, "Population 1", "Population 2" ))
  return(list(pop_1 = num1, pop_2 = nU - num1, pop = deci))
}
########################################
########################################
discr.MaxSep ←function(X1 = NULL, X2 = NULL, U, true.params = T,
                       params = ifelse(
                         true.params, list(mu1, mu2, Sigma), NULL)
                       ){
  # apply linear discriminant
  # average them by each available sample before classification
  n1 ←nrow(X1)
  n2 ←nrow(X2)
  nU ←nrow(U)
  if (true.params){
    warning("Using true parameters")
    mu1 ←params$mu1
    mu2 ←params$mu2
    Sigma ←params$Sigma
    a ←solve(Sigma) %*% (mu1 - mu2)
    bigU ←apply(U,1 ,
                function(y) t(a) %*% ((y - mu1) %*% t((y - mu1)) -
                                        (y - mu2)%*% t((y - mu2)))%*% a
```

```
                )
    re ←ifelse(bigU <= 0, "Population 1", "Population 2")
  } else {
    warning("Unknown true parameter, estimators will be uses")
    mu1 ←colMeans(X1)
    mu2 ←colMeans(X2)
    S1 ←cov(X1)
    S2 ←cov(X2)
    Sigma ←((n1 - 1) * S1 + (n2 - 1) *S2)/(n1 + n2 - 2)
    a ←solve(Sigma) %*% (mu1 - mu2)
    w1 ←X1 %*% a
    w2 ←X2 %*% a
    dU ←U %*% a
    bigU ←apply(dU, 1, function(x) mean((w1 - x))^2 - mean((w2 - x))^2)
    re ←ifelse(bigU <= 0, "Population 1", "Population 2")
  }
  num1 ←sum(re == "Population 1")
  deci ←ifelse(num1 == nU/2, "Can not decide",
               ifelse(num1 > nU/2, "Population 1", "Population 2" )
               )
  return(list(pop_1 = num1, pop_2 = nU - num1, pop = deci))
}
########################################
########################################
discr.averageD ←function(X1, X2, U){
  invX1 ←solve(cov(X1))
  invX2 ←solve(cov(X2))
  n1 ←nrow(X1); n2 ←nrow(X2); nU ←nrow(U)
  d1 ←apply(U, MARGIN = 1, FUN = function(x){
    mean(sqrt(unlist(apply(X1, MARGIN = 1,
                           FUN = function(y) sqrt((x - y) %*% invX1 %*% (x - y))
                           )
                     )
              )
         )
  }
  )
  d2 ←apply(U, MARGIN = 1, FUN = function(x){
    mean(sqrt(unlist(apply(X2, MARGIN = 1,
                           FUN = function(y) sqrt((x - y) %*% invX2 %*% (x - y))
                           )
                     )
              )
         )
  }
```

```
  )
  re ←ifelse(d1 <= d2, "Population 1", "Population 2")
  num1 ←sum(re == "Population 1")
  deci ←ifelse(num1 == nU/2, "Can not decide",
               ifelse(num1 > nU/2, "Population 1", "Population 2" )
               )
  return(list(pop_1 = num1, pop_2 = nU - num1, pop = deci))
}
########################################
########################################
discr.leverage ←function(X1, X2, U){
  nU ←nrow(U)
  n1 ←nrow(X1)
  n2 ←nrow(X1)
  invX1 ←(n1 - 1)*solve(t(X1) %*% X1)
  invX2 ←(n2 - 1)*solve(t(X2) %*% X2)
  H1 ←apply(U, MARGIN = 1, FUN = function(x) t(x) %*% invX1 %*% x )
  H2 ←apply(U, MARGIN = 1, FUN = function(x) t(x) %*% invX2 %*% x)
  re ←ifelse(H1 <= H2, "Population 1", "Population 2")
  num1 ←sum(re == "Population 1")
  deci ←ifelse(num1 == nU/2, "Can not decide",
               ifelse(num1 > nU/2, "Population 1", "Population 2" )
               )
  return(list(pop_1 = num1, pop_2 = nU - num1, pop = deci))
}
```

Source Code F.3: discr_EigenValues.R (Group classificiation using method of eigenstructures)

```
## -- Filename: discr_Eigenvalues.R
## Group discrimination using eigenvalue
antieigen ←function(X, U, type = 1){
  # Methods calculating eigenvalues
  # eta1 - H matrix
  p ←ncol(U)
  n ←nrow(U)
  r ←nrow(X)
  if (type == 1){
    H ←(n + r)/r * solve( (t(X)%*% X + t(U) %*% U ) ) %*% (t(X) %*% X)
    mye ←Re(eigen(H)$values)
    mu ←(2 * sqrt(mye[1] * mye[p])) / (mye[1] + mye[p])
    # cat("This is e:", mye) # Remove later
  }

  if (type == 2){
```

```
  H ←(n + r)/r * solve((t(X)%*% X + t(U) %*% U )) %*% (t(X) %*% X)
  mye ←Re(eigen(H)$values)
  mu ←prod(
    sapply(
      1:ceiling(p/2),
      function(i) (2 * sqrt(mye[i] * mye[p-i+1]))/(mye[i] + mye[p - i +1])
      )
    )
}

if (type == 3){
  D ←n/r * solve(t(U) %*% U) %*% (t(X) %*% X)
  mye ←Re(eigen(D)$values)
  mu ←(2 * sqrt(mye[1] * mye[p])) / (mye[1] + mye[p])
}

if (type == 4){
  D ←n/r * solve(t(U) %*% U) %*% (t(X) %*% X)
  mye ←Re(eigen(D)$values)
  mu ←prod(
    sapply(
      1:ceiling(p/2),
      function(i) (2 * sqrt(mye[i] * mye[p-i+1]))/(mye[i] + mye[p - i +1])
      )
    )
}

return(mu)
}
#####################
discr.antieigen ←function(X1, X2, U, type = 1){
mu1 ←antieigen(X1, U, type = type)
mu2 ←antieigen(X2, U, type = type)
deci ←ifelse(mu1 == mu2, "Can not decide",
             ifelse(mu1 > mu2, "Population 1", "Population 2")
             )
return(list(pop_1 = mu1, pop_2 = mu2, pop = deci))
}
###########################################
###########################################
antieigen.reg ←function(X, U, cut.point = c(.3, .7)){
#calculate antieigen values bX each region
re ←c(
  antieigen(X[which(X[, 1] <=cut.point[1]), ],
            U[which(U[,1] <= cut.point[1]), ]
```

```
    ),
    antieigen(X[which((X[, 1] > cut.point[1])&(X[, 1] <=cut.point[2])), ],
              U[which((U[, 1] > cut.point[1])&(U[, 1] <=cut.point[2])), ]
              ),
    antieigen(X[which(X[, 1] > cut.point[2]), ],
              U[which(U[,1] > cut.point[2]), ]
              )
    )
  return(re) #the higher the more similar
}
```

Source Code F.4: discr_NP.R (Group classification using optimal classifciation rule)

```
## -- Filename: discr_NP.R
# Perform optimal classification rule, Neyman-Pearson
NP ←function(U, lst_f, use.log = T){
  re ←lapply(lst_f, function(f) {
    ifelse(
      use.log == T,
      sum(log(apply(U, MARGIN = 1, f))),
      sum(apply(U, MARGIN = 1, f))
    )
  })
  return(which.max(re))
}
```

Source Code F.5: multiple_group_rules.R (Multiple group classificaiton rules)

```
## --Filename: multiple_group_rules
# lstX: list of references samples
# U: "to be classified" sample
##############################
mul.discr.linear ←function(lstX, U){
  require(MASS)
  k ←length(lstX)
  DF ←lapply(1:k, FUN = function(j) {
    df ←data.frame(lstX[[j]])
    df$pop ←j
    df}
    )
  DF ←Reduce(rbind, DF)
  DF$pop ←as.factor(DF$pop)
  lda_model ←MASS::lda(pop ~., data = DF)

  lda_predictions ←predict(lda_model, data.frame(U))
```

```
  re ←table(lda_predictions$class)
  deci ←paste("Population", as.character(which.max(re)))
  # deci ←ifelse(num1 >= nU/2, "Population 1", "Population 2" )
  return(list(pop = deci, value = re))
}
################################
################################
mul.discr.MaxSep ←function(lstX, U){
  require(MASS)
  p ←ncol(U); nU ←nrow(U)
  k ←length(lstX)
  DF ←lapply(1:k, FUN = function(j) {
    df ←data.frame(lstX[[j]])
    df$pop ←j
    df}
  )
  DF ←Reduce(rbind, DF)
  DF$pop ←as.factor(DF$pop)
  lda_model ←MASS::lda(pop ~., data = DF)

  bigD ←array(NA, dim = c(nrow(U), length(lstX))) # For U to each pop
  lda_U ←U %*% lda_model$scaling
  for (i in 1:length(lstX)){
    lda_functions ←as.matrix(
      subset(DF, pop == as.character(i), select = -(p+1))
    ) %*% lda_model$scaling
    d ←apply(
      lda_functions, MARGIN = 1,
      FUN = function(u) mean(norm(lda_U - rep(u, nU), "F"))
    )
    bigD[, i] ←d
  }

  re ←table(apply(bigD, MARGIN = 1, which.min))
  deci ←paste("Population", as.character(which.max(re)))
  return(list(pop = deci, value = re))
}
################################
################################
mul.discr.averageD ←function(lstX, U){
  nU ←nrow(U)
  bigD ←array(NA, dim = c(nrow(U), length(lstX)))
  for (i in 1:length(lstX)){
    X ←lstX[[i]]
    invX ←solve(cov(X))
```

```
    nX ←nrow(X)
    d ←apply(U, MARGIN = 1, FUN = function(x){
      mean(
        sqrt(
          unlist(
            apply(
              X, MARGIN = 1, FUN = function(y) sqrt((x - y) %*% invX %*% (x - y))
              )
            )
          )
        )
    })
    bigD[, i] ←d
  }
  re ←table(apply(bigD, MARGIN = 1, which.min))
  deci ←paste("Population", as.character(which.max(re)))
  return(list(pop = deci, value = re))
}
################################
################################
mul.discr.leverage ←function(lstX, U){
  nU ←nrow(U)
  bigH ←array(NA, dim = c(nrow(U), length(lstX)))
  for (i in 1:length(lstX)){
    X ←lstX[[i]]
    invX ←solve(t(X) %*% X)
    h ←apply(U, MARGIN = 1, FUN = function(x) t(x) %*% invX %*% x )
    bigH[, i] ←h
  }
  re ←table(apply(bigH, MARGIN = 1, which.min))
  deci ←paste("Population", as.character(which.max(re)))
  return(list(pop = deci, value = re))
}
################################
################################
mul.discr.eigen ←function(lstX, U, type = 1){
  re ←lapply(lstX,antieigen, U = U, type = type)
  deci ←paste("Population", as.character(which.max(re)))
  return(list(pop = deci, value = unlist(re)))
}
################################
################################
mul.discr.eigen.reg ←function(lstX, U,
                              cut.point = c(.35, .65),
                              w = rep(1, length(cut.point) + 1)
```

```
                              ){
 # w is the weight
  re ←lapply(
    lstX, FUN = function(x) weighted.mean(
      antieigen.reg(x, U, cut.point = cut.point), w = w
    )
    # lstX, FUN = function(x) min(
    # antieigen.reg(x, U, cut.point = cut.point)
    # )
  )
  deci ←paste("Population", as.character(which.max(re)))
  return(list(pop = deci, value = unlist(re)))
}
################################
################################
mul.Emp.copulas ←function(lstCop, U){
  require(copula)
  ############################
  eC ←emp.copulas(U)
  nU ←nrow(U)
  ############################
  myfunc ←function(Ck){
    eCk ←matrix(nrow = nx, ncol = nx)
    for (i in 1:nU){
      for (j in 1:nU){
        eCk[i,j] ←copula::pCopula(u = c(i/nx, j/nx), copula = Ck)
      }
    }
    re ←norm(eC- eCk, "F")
    return(re)
  }
  ############################
  re ←c()
  for (k in 1:length(lstCop)){
    Ck ←lstCop[[k]]
    re ←c(re, myfunc(Ck))
  }

  deci ←paste("Population", as.character(which.min(re)))
  return(list(pop = deci, value = re))
}
################################
################################
mul.NP ←function(lstCop, U){
  require(copula)
```

```
  re ←c()
  ##############################
  for (k in 1:length(lstCop)){
    Ck ←lstCop[[k]]
    re_tmp ←sum(
      apply(
        U, MARGIN = 1, function(u) copula::dCopula(u, copula = Ck, log = T)
        )
      )
    re ←c(re, re_tmp)
  }
  deci ←paste("Population", as.character(which.max(re)))
  return(list(pop = deci, value = re))
}
```

F.2 Variable selection methods

Source Code F.6: forward_selection.R

```
## --filename: forward_selection.R
forward_selection ←function(X1, X2, mincol = 2){
  # mincol: Number of dersied columns
  # choose the best columns at every step to add on
  p ←ncol(X1)
  nX1 ←nrow(X1)
  nX2 ←nrow(X2)
  q ←2
  col ←combn(1:p, q)

  sX1 ←1/sqrt(nX1) * X1
  sX2 ←1/sqrt(nX2) * X2

  # Initilization
  mycol ←col[, 1]
  H1 ←solve(
    t(sX1[, mycol]) %*% sX1[, mycol]
  ) %*% (
    t(sX2[, mycol]) %*% sX2[, mycol]
  )
  detH1 ←det(H1); trH1 ←sum(diag(H1))
  zeta1 ←detH1^(1/q)/(trH1/q)
  H2 ←solve(
    t(sX2[, mycol]) %*% sX2[, mycol]
  ) %*% (
    t(sX1[, mycol]) %*% sX1[, mycol]
```

```
)
# detH2 ←det(H2);
detH2 ←1/detH1;
trH2 ←sum(diag(H2))
zeta2 ←detH2^(1/q)/(trH2/q)
# zeta ←min(zeta1, zeta2)
zeta ←sqrt(prod(zeta1, zeta2))

for (subcol in 2:ncol(col)){
  mycol_tmp ←col[, subcol]
  H1_tmp ←solve(
    t(sX1[, mycol_tmp]) %*% sX1[, mycol_tmp]
  ) %*% (
    t(sX2[, mycol_tmp]) %*% sX2[, mycol_tmp]
  )
  detH1_tmp ←det(H1_tmp); trH1_tmp ←sum(diag(H1_tmp))
  zeta1_tmp ←detH1_tmp^(1/q)/(trH1_tmp/q)

  H2_tmp ←solve(
    t(sX2[, mycol_tmp]) %*% sX2[, mycol_tmp]
  ) %*% (
    t(sX1[, mycol_tmp]) %*% sX1[, mycol_tmp]
  )
  detH2_tmp ←1/detH1_tmp
  trH2_tmp ←sum(diag(H2_tmp))
  zeta2_tmp ←detH2_tmp^(1/q)/(trH2_tmp/q)
  zeta_tmp ←sqrt(prod(zeta1_tmp, zeta2_tmp))

  if (zeta_tmp < zeta){
    mycol ←mycol_tmp
    H1 ←H1_tmp
    H2 ←H2_tmp
    detH1 ←detH1_tmp; trH1 ←trH1_tmp
    detH2 ←detH2_tmp; trH2 ←trH2_tmp
    zeta ←zeta_tmp
  }
}

Y1 ←sX1[, mycol]
Y2 ←sX2[, mycol]
othercol ←c(1:p)[-mycol]
# print(cat("Stared colum: ", mycol))
Vzeta ←c(zeta) # Keep track of zeta after each adding iteration

# q ←q;
```

```
zetamin ←0
# Searching new columns to append
while (zetamin < zeta | q < mincol){
  #count ←count + 1 # Avoid the case q is best,
  # then algorithm will stop after searching in p - q
  Z1 ←Y1 %*% solve(t(Y1) %*% Y1) %*% t(Y1)
  Z2 ←Y2 %*% solve(t(Y2) %*% Y2) %*% t(Y2)
  iZ1 ←diag(nX1) - Z1
  iZ2 ←diag(nX2) - Z2
  Z12 ←Y1 %*% solve(t(Y1) %*% Y1) %*% t(Y2)
  Z21 ←Y2 %*% solve(t(Y2) %*% Y2) %*% t(Y1)
  lstzeta ←c()
  lstdet1 ←lstdet2 ←c()
  lsttr1 ←lsttr2 ←c()
  for (j in othercol){
    ####### (sX1sX2)-1(sX2sX2)
    B ←t(sX1[,j]) %*% iZ1 %*% sX1[,j]
    ZyyZ ←t(Z12) %*% sX1[,j] %*% t(sX1[,j]) %*% Z12
    trcapH1 ←trH1 + 1/B * (
      sum(diag((ZyyZ))) -
        2 * t(sX2[,j]) %*% t(Z12) %*% sX1[,j] +
        t(sX2[,j]) %*% sX2[,j]
    )
    detcapH1 ←detH1 * (
      t(sX2[,j]) %*% iZ2 %*% sX2[, j]
    )/(
      t(sX1[,j]) %*% iZ1 %*% sX1[, j]
    )
    zetacapH1 ←(q + 1) * detcapH1^(1/(q + 1))/trcapH1
    lstdet1 ←c(lstdet1, detcapH1)
    lsttr1 ←c(lsttr1, trcapH1)
    ####### (sX2X2)-1(x1X1)
    B ←t(sX2[,j]) %*% iZ2 %*% sX2[,j]
    ZyyZ ←t(Z21) %*% sX2[,j] %*% t(sX2[,j]) %*% Z21
    trcapH2 ←trH2 + 1/B * (
      sum(diag((ZyyZ))) -
        2 * t(sX1[,j]) %*% t(Z21) %*% sX2[,j] + t(sX1[,j]) %*% sX1[,j]
    )
    detcapH2 ←1/detcapH1
    zetacapH2 ←(q + 1) * detcapH2^(1/(q + 1))/trcapH2
    lstdet2 ←c(lstdet2, detcapH2)
    lsttr2 ←c(lsttr2, trcapH2)

    myzeta ←sqrt(prod(zetacapH1, zetacapH2))
    lstzeta ←c(lstzeta, myzeta)
```

```
    }
    zetamin ←min(lstzeta)
    # print(cat("List of zeta: ", round(lstzeta, 7)))
    # print(cat("Other colums: ", othercol[-mycol]))

    if (zetamin < zeta | q < mincol){
      index ←which.min(lstzeta)
      mycol ←c(mycol, othercol[index])
      othercol ←c(1:p)[-mycol]
      # print(othercol)
      zeta ←zetamin
      Vzeta ←c(Vzeta, zeta)
      zetamin ←0
      Y1 ←sX1[, mycol]
      Y2 ←sX2[, mycol]
      detH1 ←lstdet1[index]
      trH1 ←lsttr1[index]

      detH2 ←lstdet2[index]
      trH2 ←lsttr2[index]
      q ←q + 1;
    } else {
      break
    }
  }
  return(list(namecol = mycol, zeta = zeta, Vzeta = Vzeta))
}
```

Source Code F.7: backward_selection.R

```
## --Filename: backward_selection.R
backward_selection ←function(X1, X2){
  p ←ncol(X1)
  nX1 ←nrow(X1)
  nX2 ←nrow(X2)

  capH1 ←(nX1/nX2) *solve(t(X1) %*% X1) %*% (t(X2) %*% X2)
  detcapH1 ←det(capH1); trcapH1 ←sum(diag(capH1))
  zeta1 ←detcapH1^(1/p)/(trcapH1/p)
  capH2 ←(nX2/nX1) *solve(t(X2) %*% X2) %*% (t(X1) %*% X1)

  detcapH2 ←1/detcapH1;
  trcapH2 ←sum(diag(capH2))
  zeta2 ←detcapH2^(1/p)/(trcapH2/p)
```

```
zeta ←sqrt(prod(zeta1, zeta2))
cat(round(zeta, 7))
zetamin ←0
allcol ←1:p
capY1 ←X1
capY2 ←X2
q ←p
rm ←c() # keep track of removed columns
Vzeta ←c(zeta) # keep track of each zeta calcuated after removing each colum

while (zetamin < zeta){
  lstzeta ←c()
  for (j in 1:q){
    Y1 ←capY1[, -j]
    Y2 ←capY2[, -j]
    y1 ←capY1[, j]
    y2 ←capY2[, j]

    H1 ←(nX1/nX2) * solve(t(Y1) %*% Y1) %*% (t(Y2) %*% Y2)
    H2 ←(nX2/nX1) * solve(t(Y2) %*% Y2) %*% (t(Y1) %*% Y1)
    detH1 ←det(H1)
    detH2 ←det(H2)
    trH1 ←sum(diag(H1))
    trH2 ←sum(diag(H2))

    zetaH1 ←(q - 1) * detH1^(1/(q - 1))/trH1
    zetaH2 ←(q - 1) * detH2^(1/(q - 1))/trH2

    myzeta ←sqrt(prod(zetaH1, zetaH2))
    lstzeta ←c(lstzeta, myzeta)
  }
  zetamin ←min(lstzeta)
  cat("List of zeta: ", round(lstzeta, 5))

  if (zetamin < zeta){
    index ←which.min(lstzeta)
    rm ←c(rm, allcol[index])
    Vzeta ←c(Vzeta, lstzeta[index])
    allcol ←allcol[-index]
    cat(allcol)
    q ←q - 1
    zeta ←zetamin
    zetamin ←0
    capY1 ←capY1[, -index]
    capY2 ←capY2[, -index]
```

```
    } else {
      break
    }
  }
  return(list(
    namecol = allcol,
    zeta = zeta,
    remove.col = rm,
    Vzeta = round(Vzeta, 6)
  )
  )
}
```

F.3 Synthetic data experiments

Source Code F.8: CF_MaxSep.R (Synthetic data experiments for "majority class" rule)

```
## --Filename: CF_MaxSep.R
## Data experiment for majority class rule
########################################
# Fixed covariance matrix
p ←5
fSigma ←diag(p);
fSigma1 ←.75*(2/3 * diag(p) + 1/3 * rep(1, p) %*% t(rep(1, p)))
fSigma2 ←2 * (.1 * diag(p) + .9 * rep(1, p) %*% t(rep(1, p)))
########################################
m ←2e3
# 2 normal populations: Same Covariance matrix
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p); Sigma ←fSigma
mu2 ←rep(0, p) + 3;
set.seed(1234)
X1 ←MASS::mvrnorm(n1, mu1, Sigma)
X2 ←MASS::mvrnorm(n2, mu2, Sigma)
params ←list(mu1 = mu1, mu2 = mu2, Sigma = Sigma)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←MASS::mvrnorm(n1, mu1, Sigma)
  U2 ←MASS::mvrnorm(n2, mu2, Sigma)

  re.linear_known1 ←discr.linear(U = U1, params = params)$pop
  re.linear_known2 ←discr.linear(U = U2, params = params)$pop

  re.linear1 ←discr.linear(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.linear2 ←discr.linear(X1 = X1, X2 = X2, U = U2, true.params = F)$pop
```

```
  re.MaxSep_known1 ←discr.MaxSep(U = U1,params = params)$pop
  re.MaxSep_known2 ←discr.MaxSep(U = U2, params = params)$pop

  re.MaxSep1 ←discr.MaxSep(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.MaxSep2 ←discr.MaxSep(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.avgD1 ←discr.averageD(X1 = X1, X2 = X2, U = U1)$pop
  re.avgD2 ←discr.averageD(X1 = X1, X2 = X2, U = U2)$pop

  re.leverage1 ←discr.leverage(X1 = X1, X2 = X2, U = U1)$pop
  re.leverage2 ←discr.leverage(X1 = X1, X2 = X2, U = U2)$pop

  tmp ←c(linear_known11 = re.linear_known1,
           linear_known22 = re.linear_known2,
           linear11 = re.linear1,
           linear22 = re.linear2,
           MaxSep_known11 = re.MaxSep_known1,
           MaxSep_known22 = re.MaxSep_known2,
           MaxSep11 = re.MaxSep1,
           MaxSep22 = re.MaxSep2,
           avgD11 = re.avgD1,
           avgD22 = re.avgD2,
           leverage11 = re.leverage1,
           leverage22 = re.leverage2)
  # Randomly assign to each population if Can not decide.
  tmp[tmp == "Can not decide"] ←sample(c("Population 1", "Population 2"),
                                       size = sum(tmp == "Can not decide"),
                                       replace = T)
  tmp
})

re1 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
print(re1)
########################################
########################################
# 2 normal populations: Same mean/Different Covariance matrix
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p); Sigma1 ←fSigma1
mu2 ←mu1; Sigma2 ←fSigma2
set.seed(1234)
X1 ←MASS::mvrnorm(n1, mu1, Sigma1)
X2 ←MASS::mvrnorm(n2, mu2, Sigma2)
# params ←list(mu1 = mu1, mu2 = mu2, Sigma = Sigma)
```

```
set.seed(789)
re ←replicate(m, expr = {
  U1 ←MASS::mvrnorm(n1, mu1, Sigma1)
  U2 ←MASS::mvrnorm(n2, mu2, Sigma2)

  re.linear1 ←discr.linear(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.linear2 ←discr.linear(X1 = X1, X2 = X2, U = U2, true.params = F)$pop
  re.MaxSep1 ←discr.MaxSep(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.MaxSep2 ←discr.MaxSep(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.avgD1 ←discr.averageD(X1 = X1, X2 = X2, U = U1)$pop
  re.avgD2 ←discr.averageD(X1 = X1, X2 = X2, U = U2)$pop

  re.leverage1 ←discr.leverage(X1 = X1, X2 = X2, U = U1)$pop
  re.leverage2 ←discr.leverage(X1 = X1, X2 = X2, U = U2)$pop

  tmp ←c(
    linear11 = re.linear1,
    linear22 = re.linear2,
    MaxSep11 = re.MaxSep1,
    MaxSep22 = re.MaxSep2,
    avgD11 = re.avgD1,
    avgD22 = re.avgD2,
    leverage11 = re.leverage1,
    leverage22 = re.leverage2)
  # Randomly assign to each population if Can not decide.
  tmp[tmp == "Can not decide"] ←sample(c("Population 1", "Population 2"),
                                       size = sum(tmp == "Can not decide"),
                                       replace = T)
  tmp
})

re2 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
print(re2)
########################################
########################################
# Kotz/t: Same Covariance matrix, different mean
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p) + 3; Sigma ←fSigma
mu2 ←rep(0, p);
params ←list(mu1 = mu1, mu2 = mu2, Sigma = Sigma)
set.seed(1234)
source("~/OU /Dissertation/Normality Test/Kotz.R")
X1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma)
```

```
X2 ←mvtnorm::rmvt(n2, 5/7*Sigma, df = 7, delta = mu2)
set.seed(789)
re ←replicate(m, expr = {

  U1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma)
  U2 ←mvtnorm::rmvt(n2, 5/7*Sigma, df = 7, delta = mu2)

  re.linear_known1 ←discr.linear(U = U1, params = params)$pop
  re.linear_known2 ←discr.linear(U = U2, params = params)$pop

  re.linear1 ←discr.linear(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.linear2 ←discr.linear(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.MaxSep_known1 ←discr.MaxSep(U = U1,params = params)$pop
  re.MaxSep_known2 ←discr.MaxSep(U = U2, params = params)$pop

  re.MaxSep1 ←discr.MaxSep(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.MaxSep2 ←discr.MaxSep(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.avgD1 ←discr.averageD(X1 = X1, X2 = X2, U = U1)$pop
  re.avgD2 ←discr.averageD(X1 = X1, X2 = X2, U = U2)$pop

  re.leverage1 ←discr.leverage(X1 = X1, X2 = X2, U = U1)$pop
  re.leverage2 ←discr.leverage(X1 = X1, X2 = X2, U = U2)$pop

  tmp ←c(linear_known11 = re.linear_known1,
          linear_known22 = re.linear_known2,
          linear11 = re.linear1,
          linear22 = re.linear2,
          MaxSep_known11 = re.MaxSep_known1,
          MaxSep_known22 = re.MaxSep_known2,
          MaxSep11 = re.MaxSep1,
          MaxSep22 = re.MaxSep2,
          avgD11 = re.avgD1,
          avgD22 = re.avgD2,
          leverage11 = re.leverage1,
          leverage22 = re.leverage2)
  # Randomly assign to each population if Can not decide.
  tmp[tmp == "Can not decide"] ←sample(c("Population 1", "Population 2"),
                                       size = sum(tmp == "Can not decide"),
                                       replace = T)
  tmp

})
```

```
re3 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
print(re3)
########################################
########################################
# Kotz/t: Same mean, different covariance matrix
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p); Sigma1 ←fSigma1
mu2 ←rep(0, p); Sigma2 ←fSigma2
set.seed(1234)
source("~/OU /Dissertation/Normality Test/Kotz.R")
# X1 ←MASS::mvrnorm(n1, mu1, Sigma1)
X1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma1)
X2 ←mvtnorm::rmvt(n2, 5/7*Sigma2, df = 7, delta = mu2)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma1)
  U2 ←mvtnorm::rmvt(n2, 5/7*Sigma2, df = 7, delta = mu2)


  re.linear1 ←discr.linear(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.linear2 ←discr.linear(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.MaxSep1 ←discr.MaxSep(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.MaxSep2 ←discr.MaxSep(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.avgD1 ←discr.averageD(X1 = X1, X2 = X2, U = U1)$pop
  re.avgD2 ←discr.averageD(X1 = X1, X2 = X2, U = U2)$pop

  re.leverage1 ←discr.leverage(X1 = X1, X2 = X2, U = U1)$pop
  re.leverage2 ←discr.leverage(X1 = X1, X2 = X2, U = U2)$pop

  tmp ←c(
    # linear_known11 = re.linear_known1,
    # linear_known22 = re.linear_known2,
    linear11 = re.linear1,
    linear22 = re.linear2,
    # MaxSep_known11 = re.MaxSep_known1,
    # MaxSep_known22 = re.MaxSep_known2,
    MaxSep11 = re.MaxSep1,
    MaxSep22 = re.MaxSep2,
    avgD11 = re.avgD1,
    avgD22 = re.avgD2,
    leverage11 = re.leverage1,
    leverage22 = re.leverage2)
```

```
  # Randomly assign to each population if Can not decide.
  tmp[tmp == "Can not decide"] <-sample(c("Population 1", "Population 2"),
                                        size = sum(tmp == "Can not decide"),
                                        replace = T)
  tmp

})

re4 <-rbind(apply(re == "Population 1", 1, mean),
            apply(re == "Population 2", 1, mean))
print(re4)

########################################
########################################
# Lomax/t: Same mean, diff covariance matrix
n1 <-200; n2 <-200; p <-5
a <-3; theta = rep(1, p)
S1 <-covLomax(a, theta)$Sigma
mu <-covLomax(a, theta)$mu
S2 <-fSigma2
set.seed(111)
source("~/OU /Dissertation/Normality Test/Kotz.R")
# X1 <-MASS::mvrnorm(n1, mu1, Sigma1)
X1 <-NonNorMvtDist::rmvlomax(n1, parm1 = a, parm2 = theta)
X2 <-mvtnorm::rmvt(n2, 5/7*S2, df = 7, delta = mu)
set.seed(789)
re <-replicate(m, expr = {
  U1 <-NonNorMvtDist::rmvlomax(n1, parm1 = a, parm2 = theta)
  U2 <-mvtnorm::rmvt(n2, 5/7*S2, df = 7, delta = mu)

  re.linear1 <-discr.linear(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.linear2 <-discr.linear(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.MaxSep1 <-discr.MaxSep(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.MaxSep2 <-discr.MaxSep(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.avgD1 <-discr.averageD(X1 = X1, X2 = X2, U = U1)$pop
  re.avgD2 <-discr.averageD(X1 = X1, X2 = X2, U = U2)$pop

  re.leverage1 <-discr.leverage(X1 = X1, X2 = X2, U = U1)$pop
  re.leverage2 <-discr.leverage(X1 = X1, X2 = X2, U = U2)$pop

  tmp <-c(
    linear11 = re.linear1,
    linear22 = re.linear2,
```

```
    MaxSep11 = re.MaxSep1,
    MaxSep22 = re.MaxSep2,
    avgD11 = re.avgD1,
    avgD22 = re.avgD2,
    leverage11 = re.leverage1,
    leverage22 = re.leverage2)
  # Randomly assign to each population if Can not decide.
  tmp[tmp == "Can not decide"] <-sample(c("Population 1", "Population 2"),
                                        size = sum(tmp == "Can not decide"),
                                        replace = T)
  tmp

})

re5 <-rbind(apply(re == "Population 1", 1, mean),
            apply(re == "Population 2", 1, mean))
print(re5)
#####################################################
# Lomax/t: Same Covariance matrix, different mean
n1 <-200; n2 <-200; p <-5
a <-3; theta = rep(1, p)
S <-covLomax(a, theta)$Sigma
mu1 <-covLomax(a, theta)$mu
mu2 <-rep(3, p)

set.seed(111)
source("~/OU /Dissertation/Normality Test/Kotz.R")
# X1 <-MASS::mvrnorm(n1, mu1, Sigma1)
X1 <-NonNorMvtDist::rmvlomax(n1, parm1 = a, parm2 = theta)
X2 <-mvtnorm::rmvt(n2, 5/7*S, df = 7, delta = mu2)
params <-list(mu1 = mu1, mu2 = mu2, Sigma = S)
set.seed(1234)
source("~/OU /Dissertation/Normality Test/Kotz.R")
set.seed(789)
re <-replicate(m, expr = {

  U1 <-NonNorMvtDist::rmvlomax(n1, parm1 = a, parm2 = theta)
  U2 <-mvtnorm::rmvt(n2, 5/7*S, df = 7, delta = mu2)

  re.linear_known1 <-discr.linear(U = U1, params = params)$pop
  re.linear_known2 <-discr.linear(U = U2, params = params)$pop

  re.linear1 <-discr.linear(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.linear2 <-discr.linear(X1 = X1, X2 = X2, U = U2, true.params = F)$pop
```

```
  re.MaxSep_known1 ←discr.MaxSep(U = U1,params = params)$pop
  re.MaxSep_known2 ←discr.MaxSep(U = U2, params = params)$pop

  re.MaxSep1 ←discr.MaxSep(X1 = X1, X2 = X2, U = U1, true.params = F)$pop
  re.MaxSep2 ←discr.MaxSep(X1 = X1, X2 = X2, U = U2, true.params = F)$pop

  re.avgD1 ←discr.averageD(X1 = X1, X2 = X2, U = U1)$pop
  re.avgD2 ←discr.averageD(X1 = X1, X2 = X2, U = U2)$pop

  re.leverage1 ←discr.leverage(X1 = X1, X2 = X2, U = U1)$pop
  re.leverage2 ←discr.leverage(X1 = X1, X2 = X2, U = U2)$pop

  tmp ←c(linear_known11 = re.linear_known1,
          linear_known22 = re.linear_known2,
          linear11 = re.linear1,
          linear22 = re.linear2,
          MaxSep_known11 = re.MaxSep_known1,
          MaxSep_known22 = re.MaxSep_known2,
          MaxSep11 = re.MaxSep1,
          MaxSep22 = re.MaxSep2,
          avgD11 = re.avgD1,
          avgD22 = re.avgD2,
          leverage11 = re.leverage1,
          leverage22 = re.leverage2)
  # Randomly assign to each population if Can not decide.
  tmp[tmp == "Can not decide"] ←sample(c("Population 1", "Population 2"),
                                       size = sum(tmp == "Can not decide"),
                                       replace = T)
  tmp

})
re6 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
print(re6)
# ##############################################
# miss_points_normal_normal ←re1
# save(miss_points_normal_normal, file = "miss_points_normal_normal.RDATA")
# miss_points_kotz_t ←re2
# save(miss_points_kotz_t, file = "miss_points_kotz_t.RDATA")
# miss_points_kotz_t2 ←re3
# save(miss_points_kotz_t2, file = "miss_points_kotz_t2.RDATA")
# miss_points_lomax_normal ←re4
# save(miss_points_lomax_normal, file = "miss_points_lomax_normal.RDATA")
```

Source Code F.9: CF_NP.R (Synthetic data experiments for optimal classification rule)

```
## -- Filename: CF_NP.R
m ←2e3;
n1 ←200; n2 ←200; p ←5
source("~/OU /Dissertation/Normality Test/Kotz.R")
#####################################
#####################################
# 2 normal populations: Same Covariance matrix
mu1 ←rep(0, p); Sigma ←fSigma
mu2 ←rep(0, p) + 3;
f1 ←function(u) mvtnorm::dmvnorm(u, mean = mu1, sigma = Sigma)
f2 ←function(u) mvtnorm::dmvnorm(u, mean = mu2, sigma = Sigma)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←MASS::mvrnorm(n1, mu1, Sigma)
  U2 ←MASS::mvrnorm(n2, mu2, Sigma)
  re1.known ←ifelse(
    NP(U1, lst_f = c(f1, f2)) == 1,
    "Population 1", "Population 2"
  )
  re2.known ←ifelse(
    NP(U2, lst_f = c(f1, f2)) == 1,
    "Population 1", "Population 2"
  )
  c(NP_known11 = re1.known,
    NP_known22 = re2.known
  )
})
re1_NP ←rbind(apply(re == "Population 1", 1, mean),
              apply(re == "Population 2", 1, mean))
print(re1_NP)
#####################################
#####################################
# 2 normal populations: Same mean/Different Covariance matrix
mu1 ←rep(0, p); Sigma1 ←fSigma1
mu2 ←mu1; Sigma2 ←fSigma2
set.seed(789)
f1 ←function(u) mvtnorm::dmvnorm(u, mean = mu1, sigma = Sigma1)
f2 ←function(u) mvtnorm::dmvnorm(u, mean = mu2, sigma = Sigma2)

set.seed(789)
re ←replicate(m, expr = {
  U1 ←MASS::mvrnorm(n1, mu1, Sigma1)
  U2 ←MASS::mvrnorm(n2, mu2, Sigma2)
  re1.known ←ifelse(
```

```
    NP(U1, lst_f = c(f1, f2)) == 1,
    "Population 1", "Population 2"
  )
  re2.known ←ifelse(
    NP(U2, lst_f = c(f1, f2)) == 1,
    "Population 1", "Population 2"
  )
  c(NP_known11 = re1.known,
    NP_known22 = re2.known
  )
})
re2_NP ←rbind(apply(re == "Population 1", 1, mean),
              apply(re == "Population 2", 1, mean))
print(re2_NP)
########################################
########################################
# Kotz/t: Same Covariance matrix, different mean
mu1 ←rep(0, p) + 3; Sigma ←fSigma
mu2 ←rep(0, p);
f1 ←function(u) log(dmvKotz(u, mu = mu1, Sigma = 1/(p+1)*Sigma))
f2 ←function(u) mvtnorm::dmvt(u, 5/7*Sigma, df = 7, delta = mu2, log = T)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma)
  U2 ←mvtnorm::rmvt(n2, 5/7*Sigma, df = 7, delta = mu2)
  re1.known ←ifelse(
    NP(U1, lst_f = c(f1, f2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
  re2.known ←ifelse(
    NP(U2, lst_f = c(f1, f2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
  # For U1:
  re1 ←mleKotz(colMeans(U1), cov(U1), U1)
  hf1 ←function(u) log(dmvKotz(u, mu = re1$mu, Sigma = re1$Sigma))
  re2 ←Compositional::multivt(U1)
  hf2 ←function(u) mvtnorm::dmvt(u,
                                 ((re2$df-2)/re2$df * re2$covariance),
                                 df = re2$df, delta = re2$center,log = T
  )
  re1_unknown ←ifelse(
    NP(U1, lst_f = c(hf1, hf2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
```

```
  # For U2:
  re1 ←mleKotz(colMeans(U2), cov(U2), U2)
  hf1 ←function(u) log(dmvKotz(u, mu = re1$mu, Sigma = re1$Sigma))
  re2 ←Compositional::multivt(U2)
  hf2 ←function(u) mvtnorm::dmvt(u,
                                 ((re2$df-2)/re2$df * re2$covariance),
                                 df = re2$df, delta = re2$center, log = T
  )
  re2_unknown ←ifelse(NP(U2, lst_f = c(hf1, hf2), use.log = F) == 1,
                      "Population 1", "Population 2"
  )
  c(NP_known11 = re1.known,
    NP_known22 = re2.known,
    NP11 = re1_unknown,
    NP22 = re2_unknown
  )
})
re3_nP ←rbind(apply(re == "Population 1", 1, mean),
              apply(re == "Population 2", 1, mean))
print(re3_nP)
########################################
########################################
# Kotz/t: Same mean, difference covariance matrix
mu1 ←rep(0, p); Sigma1 ←fSigma1
mu2 ←rep(0, p); Sigma2 ←fSigma2
# source("~/OU /Dissertation/Normality Test/Kotz.R")
f1 ←function(u) log(dmvKotz(u, mu = mu1, Sigma = 1/(p+1)*Sigma1))
f2 ←function(u) mvtnorm::dmvt(u, 5/7*Sigma2, df = 7, delta = mu2, log = T)
set.seed(789)
re ←replicate(m, expr = {

  U1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma1)
  U2 ←mvtnorm::rmvt(n2, 5/7*Sigma2, df = 7, delta = mu2)

  re1.known ←ifelse(
    NP(U1, lst_f = c(f1, f2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
  re2.known ←ifelse(
    NP(U2, lst_f = c(f1, f2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
  # For U1:
  re1 ←mleKotz(colMeans(U1), cov(U1), U1)
  hf1 ←function(u) log(dmvKotz(u, mu = re1$mu, Sigma = re1$Sigma))
```

```
  re2 ←Compositional::multivt(U1)
  hf2 ←function(u) mvtnorm::dmvt(u,
                                 ((re2$df-2)/re2$df * re2$covariance),
                                 df = re2$df, delta = re2$center, log = T
  )
  re1_unknown ←ifelse(
    NP(U1, lst_f = c(hf1, hf2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
  # For U2:
  re1 ←mleKotz(colMeans(U2), cov(U2), U2)
  hf1 ←function(u) log(dmvKotz(u, mu = re1$mu, Sigma = re1$Sigma))
  re2 ←Compositional::multivt(U2)
  hf2 ←function(u) mvtnorm::dmvt(u,
                                 ((re2$df-2)/re2$df * re2$covariance),
                                 df = re2$df, delta = re2$center, log = T
  )
  re2_unknown ←ifelse(
    NP(U2, lst_f = c(hf1, hf2), use.log = F) == 1,
    "Population 1", "Population 2"
  )
  c(NP_known11 = re1.known,
    NP_known22 = re2.known,
    NP11 = re1_unknown,
    NP22 = re2_unknown
  )
})

re4_NP ←rbind(apply(re == "Population 1", 1, mean),
              apply(re == "Population 2", 1, mean))
print(re4_NP)
```

Source Code F.10: CF_MaxSep.R (Synthetic data experiments for eigenstructure method)

```
## --Filename: CF_antieigen.R
########################################
########################################
# Fixed covariance matrix
p ←5
fSigma ←diag(p);
fSigma1 ←.75*(2/3 * diag(p) + 1/3 * rep(1, p) %*% t(rep(1, p)))
fSigma2 ←2 * (.1 * diag(p) + .9 * rep(1, p) %*% t(rep(1, p)))
########################################
m ←2e3
########################################
```

```
# 2 normal populations: Same Covariance matrix
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p); Sigma ←fSigma
mu2 ←rep(0, p) + 3;
set.seed(1234)
X1 ←MASS::mvrnorm(n1, mu1, Sigma)
X2 ←MASS::mvrnorm(n2, mu2, Sigma)
# params ←list(mu1 = mu1, mu2 = mu2, Sigma = Sigma)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←MASS::mvrnorm(n1, mu1, Sigma)
  U2 ←MASS::mvrnorm(n2, mu2, Sigma)

  discr_type_1_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =1)$pop
  discr_type_1_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =1)$pop

  discr_type_2_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =2)$pop
  discr_type_2_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =2)$pop

  discr_type_3_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =3)$pop
  discr_type_3_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =3)$pop

  discr_type_4_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =4)$pop
  discr_type_4_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =4)$pop

  c(type1_11 = discr_type_1_1,
    type1_22 = discr_type_1_2,
    type2_11 = discr_type_2_1,
    type2_22 = discr_type_2_2,
    type3_11 = discr_type_3_1,
    type3_22 = discr_type_3_2,
    type4_11 = discr_type_4_1,
    type4_22 = discr_type_4_2
    )
})
re1 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
re1
########################################
########################################
# 2 normal populations: Same mean/Different Covariance matrix
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p); Sigma1 ←fSigma1
mu2 ←mu1; Sigma2 ←fSigma2
set.seed(1234)
```

```
X1 ←MASS::mvrnorm(n1, mu1, Sigma1)
X2 ←MASS::mvrnorm(n2, mu2, Sigma2)
# params ←list(mu1 = mu1, mu2 = mu2, Sigma = Sigma)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←MASS::mvrnorm(n1, mu1, Sigma1)
  U2 ←MASS::mvrnorm(n2, mu2, Sigma2)

  discr_type_1_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =1)$pop
  discr_type_1_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =1)$pop

  discr_type_2_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =2)$pop
  discr_type_2_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =2)$pop

  discr_type_3_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =3)$pop
  discr_type_3_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =3)$pop

  discr_type_4_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =4)$pop
  discr_type_4_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =4)$pop

  c(type1_11 = discr_type_1_1,
    type1_22 = discr_type_1_2,
    type2_11 = discr_type_2_1,
    type2_22 = discr_type_2_2,
    type3_11 = discr_type_3_1,
    type3_22 = discr_type_3_2,
    type4_11 = discr_type_4_1,
    type4_22 = discr_type_4_2
  )
})
re2 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
re2
########################################
########################################
# Kotz/t: Same Covariance matrix, different mean
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p) + 3; Sigma ←fSigma
mu2 ←rep(0, p);
params ←list(mu1 = mu1, mu2 = mu2, Sigma = Sigma)
set.seed(1234)
source("~/OU /Dissertation/Normality Test/Kotz.R")
X1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma)
X2 ←mvtnorm::rmvt(n2, 5/7*Sigma, df = 7, delta = mu2)
set.seed(789)
```

```
re ←replicate(m, expr = {
  U1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma)
  U2 ←mvtnorm::rmvt(n2, 5/7*Sigma, df = 7, delta = mu2)

  discr_type_1_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =1)$pop
  discr_type_1_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =1)$pop

  discr_type_2_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =2)$pop
  discr_type_2_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =2)$pop

  discr_type_3_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =3)$pop
  discr_type_3_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =3)$pop

  discr_type_4_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =4)$pop
  discr_type_4_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =4)$pop

  c(type1_11 = discr_type_1_1,
    type1_22 = discr_type_1_2,
    type2_11 = discr_type_2_1,
    type2_22 = discr_type_2_2,
    type3_11 = discr_type_3_1,
    type3_22 = discr_type_3_2,
    type4_11 = discr_type_4_1,
    type4_22 = discr_type_4_2
  )
})
re3 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean))
re3
########################################
########################################
# Kotz/t: Same mean, difference covariance matrix
n1 ←200; n2 ←200; p ←5
mu1 ←rep(0, p); Sigma1 ←fSigma1
mu2 ←rep(0, p); Sigma2 ←fSigma2
set.seed(1234)
source("~/OU /Dissertation/Normality Test/Kotz.R")
# X1 ←MASS::mvrnorm(n1, mu1, Sigma1)
X1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma1)
X2 ←mvtnorm::rmvt(n2, 5/7*Sigma2, df = 7, delta = mu2)
set.seed(789)
re ←replicate(m, expr = {
  U1 ←rmvKotz(n = n1, mu = mu1, Sigma = 1/(p+1)*Sigma1)
  U2 ←mvtnorm::rmvt(n2, 5/7*Sigma2, df = 7, delta = mu2)
```

```
  discr_type_1_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =1)$pop
  discr_type_1_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =1)$pop

  discr_type_2_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =2)$pop
  discr_type_2_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =2)$pop

  discr_type_3_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =3)$pop
  discr_type_3_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =3)$pop

  discr_type_4_1 ←discr.antieigen(X1 = X1, X2 = X2, U = U1, type =4)$pop
  discr_type_4_2 ←discr.antieigen(X1 = X1, X2 = X2, U = U2, type =4)$pop

  c(type1_11 = discr_type_1_1,
    type1_22 = discr_type_1_2,
    type2_11 = discr_type_2_1,
    type2_22 = discr_type_2_2,
    type3_11 = discr_type_3_1,
    type3_22 = discr_type_3_2,
    type4_11 = discr_type_4_1,
    type4_22 = discr_type_4_2
  )
})
re4 ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Population 2", 1, mean)
           )
re4
##########################################
##########################################
# A drawback of eigenstructure method
# #Normal/Lomax: same mean/same covariance
# n1 ←200; n2 ←200; p ←2
# a ←3; theta = rep(1, p)
# S ←covLomax(a, theta)$Sigma
# mu ←covLomax(a, theta)$mu
#
# set.seed(111)
# X1 ←NonNorMvtDist::rmvlomax(n1, parm1 = a, parm2 = theta)
# X2 ←MASS::mvrnorm(n2, mu, S)
#
# set.seed(999)
# set.seed(789)
#
# re ←replicate(m, expr = {
#
# U1 ←NonNorMvtDist::rmvlomax(n1, parm1 = a, parm2 = theta)
```

```
# U2 <-MASS::mvrnorm(n2, mu, S)
#
# discr_type_1_1 <-discr.antieigen(X1 = X1, X2 = X2, U = U1, type =1)$pop
# discr_type_1_2 <-discr.antieigen(X1 = X1, X2 = X2, U = U2, type =1)$pop
#
# discr_type_2_1 <-discr.antieigen(X1 = X1, X2 = X2, U = U1, type =2)$pop
# discr_type_2_2 <-discr.antieigen(X1 = X1, X2 = X2, U = U2, type =2)$pop
#
# discr_type_3_1 <-discr.antieigen(X1 = X1, X2 = X2, U = U1, type =3)$pop
# discr_type_3_2 <-discr.antieigen(X1 = X1, X2 = X2, U = U2, type =3)$pop
#
# discr_type_4_1 <-discr.antieigen(X1 = X1, X2 = X2, U = U1, type =4)$pop
# discr_type_4_2 <-discr.antieigen(X1 = X1, X2 = X2, U = U2, type =4)$pop
#
# c(type1_11 = discr_type_1_1,
# type1_22 = discr_type_1_2,
# type2_11 = discr_type_2_1,
# type2_22 = discr_type_2_2,
# type3_11 = discr_type_3_1,
# type3_22 = discr_type_3_2,
# type4_11 = discr_type_4_1,
# type4_22 = discr_type_4_2
# )
# })
#
# re4 <-rbind(apply(re == "Population 1", 1, mean),
# apply(re == "Can not decide", 1, mean),
# apply(re == "Population 2", 1, mean))
```

Source Code F.11: CF_varsel.R (Synthetic data experiments methods of variable selections)

```
##--Filename: CF_varsel.R
### Sampe covariance, diff mean
mu <-c(rep(-10, 3), rep(-5, 2), rep(0, 10), rep(7, 3), rep(15, 2))
p <-length(mu)
set.seed(123)
mu1 <-sample(mu)
mu2 <-sample(mu)
plot(abs(mu1 - mu2))
text(1:p, abs(mu1 - mu2) , as.character(1:p), pos=1)

set.seed(1111)
X1 <-MASS::mvrnorm(500, mu1, diag(p))
X2 <-MASS::mvrnorm(700, mu2, diag(p))
```

```
forward_selection(X1, X2)
backward_selection(X1, X2)

### Same mean, diff covariance matrix
n1 ←500; n2 ←700;
p ←20;p ←30
q1 ←10
S ←diag(q1)
eigen(S, only.values = T)
set.seed(1234)
Y1 ←MASS::mvrnorm(n1, rep(0, q1), S)
Y2 ←MASS::mvrnorm(n2, rep(0, q1), S)
###
q2 ←p - q1
###
set.seed(123)
J ←c(0, 1, 2)
A ←matrix(sample(J, size = q2*q2, replace = T), nrow = q2, byrow = T)
diag(A)[diag(A) == 0] ←rep(1, sum(diag(A)==0))
S1 ←t(A) %*% A
set.seed(123)
X1 ←MASS::mvrnorm(n1, rep(0, q2), S1)
####
J ←c(-2, -1, 0, 1, 2)
set.seed(123)
A ←matrix(sample(J, size = q2*q2, replace = T), nrow = q2, byrow = T)
diag(A)[diag(A) <= 0] ←rep(1, sum(diag(A)==0))
S2 ←t(A) %*% A
set.seed(123)
X2 ←MASS::mvrnorm(n2, rep(0, q2), S2)

X1 ←cbind(Y1, X1)
X2 ←cbind(Y2, X2)
st ←Sys.time()
forward_selection(X1, X2)
et ←Sys.time()
et - st
st ←Sys.time()
backward_selection(X1, X2)
et ←Sys.time()
et - st
```

F.4 Water Potability revisited

Source Code F.12: Water Potability -Classification and Variable selection

```
## -- Filename: Data_water.R
water ←
  read.csv("Data/water_potability.csv")
library(dplyr)
water ←na.omit(water)
X1 ←water %>% dplyr::filter(Potability == 1) %>% select(-c("Potability"))
X0 ←water %>% dplyr::filter(Potability == 0) %>% select(-c("Potability"))
S1 ←cov(X1)
S0 ←cov(X0)
# S_compare ←(S1 - S0)/S1
S_compare ←S0/S1
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.1, 2,1), cex.main = 1.2)
plot(c(S_compare[upper.tri(S_compare, diag = T)]),
     ylab = "Ratio of elements in sample covariance matrices",
     lwd = 2, pch = 16, col = "#2171b5", yaxt="n",
     xlab = "Index of distinct elements in covariance matrix"
     )
axis(2, at = c(-15, -10, -1.5, 0, 1.5, 5, 10, 15), las=2)
abline(h = 1.5, lty = "dotted", col = "orange", lwd = 2)
abline(h = -1.5, lty = "dotted", col = "orange", lwd = 2)

e1 ←eigen(cov(X1), only.values = T)$values
e0 ←eigen(cov(X0), only.values = T)$values
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.1, 2,1), cex.main = 1.2)
plot(e0/e1, ylim = c(0.5, 1.5),
     ylab = "Ratio of eigenvalues of sample covariance matrices",
     lwd = 2, pch = 16, col = "#2171b5", yaxt="n",
     xlab = "")
axis(2, at = c( .5, .7, 1, 1.3, 1.5), las=2)

mu1 ←colMeans(X1); mu0 ←colMeans(X0)
par(cex.axis = 1.2, cex.lab = 1.2, mar = c(4, 4.1, 2,1), cex.main = 1.2)
plot(mu0/mu1, ylim = c(0.5, 1.5),
     ylab = "Ratio of elements in sample means",
     lwd = 2, pch = 16, col = "#2171b5", yaxt="n",
     xlab = "Variable numbers")
axis(2, at = c( .5, .7, 1, 1.3, 1.5), las=2)
##############################
m ←2e3
# m ←1e2
set.seed(1234)
re ←replicate(m, expr ={
  split1 ←rbinom(nrow(X1), 1, 0.3)
  sX1 ←as.matrix(X1[split1 == 0, ]) #Labeled sample
```

```
U1 <-as.matrix(X1[split1 == 1,]) # Unlabeled sample

split0 <-rbinom(nrow(X0), 1, 0.3)
sX0 <-as.matrix(X0[split0 == 0,])
U0 <-as.matrix(X0[split0 == 1, ])

re.linear1 <-discr.linear(X1 = sX1, X2 = sX0, U = U1, true.params = F)$pop
re.linear2 <-discr.linear(X1 = sX1, X2 = sX0, U = U0, true.params = F)$pop

re.MaxSep1 <-discr.MaxSep(X1 = sX1, X2 = sX0, U = U1, true.params = F)$pop
re.MaxSep2 <-discr.MaxSep(X1 = sX1, X2 = sX0, U = U0, true.params = F)$pop

re.avgD1 <-discr.averageD(X1 = sX1, X2 = sX0, U = U1)$pop
re.avgD2 <-discr.averageD(X1 = sX1, X2 = sX0, U = U0)$pop

re.leverage1 <-discr.leverage(X1 = sX1, X2 = sX0, U = U1)$pop
re.leverage2 <-discr.leverage(X1 = sX1, X2 = sX0, U = U0)$pop

discr_type_1_1 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U1, type =1)$pop
discr_type_1_2 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U0, type =1)$pop

discr_type_2_1 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U1, type =2)$pop
discr_type_2_2 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U0, type =2)$pop

discr_type_3_1 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U1, type =3)$pop
discr_type_3_2 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U0, type =3)$pop

discr_type_4_1 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U1, type =4)$pop
discr_type_4_2 <-discr.antieigen(X1 = sX1, X2 = sX0, U = U0, type =4)$pop

tmp <-c(
  linear11 = re.linear1,
  linear22 = re.linear2,
  MaxSep11 = re.MaxSep1,
  MaxSep22 = re.MaxSep2,
  avgD11 = re.avgD1,
  avgD22 = re.avgD2,
  leverage11 = re.leverage1,
  leverage22 = re.leverage2,
  ############################
  type1_11 = discr_type_1_1,
  type1_22 = discr_type_1_2,
  type2_11 = discr_type_2_1,
  type2_22 = discr_type_2_2,
  type3_11 = discr_type_3_1,
```

```
    type3_22 = discr_type_3_2,
    type4_11 = discr_type_4_1,
    type4_22 = discr_type_4_2
  )
  tmp[tmp == "Can not decide"] ←sample(c("Population 1", "Population 2"),
                                       size = sum(tmp == "Can not decide"),
                                       replace = T)
  tmp
})
re_Water ←rbind(apply(re == "Population 1", 1, mean),
           apply(re == "Can not decide", 1, mean),
           apply(re == "Population 2", 1, mean))
############################
### Variable Selection ###########
X1 ←as.matrix(X1)
X0 ←as.matrix(X0)
forward_selection(X1, X0)
backward_selection(X1, X0)
```

APPENDIX G

SOURCE CODE OF CHAPTER SIX

Source Code G.1: Arc_copula.R (Archimedean and Elliptical copulas generators)

```
## --Filename: Arc_copulas.R
## Generate some copulas with given tau correlation and dimension
Arc_copula <-function(p, tau, df = 4){
  require(copula)
  normal.cor <-sin(pi*tau/2)
  clayton.alpha <-(2 *tau)/(1- tau)
  gumbel.alpha <-1 /(1 - tau)
  ####
  fun <-function(theta){
    1/theta * integrate(
      function(t) t/(exp(t) - 1), lower = 0, upper = theta
      )$value
  }
  frank.alpha <-uniroot(function(theta) 1 + 4/theta *(fun(theta) - 1) - tau,
                        lower = -100, upper = 100)$root

  fun.joe <-function(theta){
    1 + 4/theta^2 * integrate(
      function(t) t*log(t)*(1 - t)^{2*(1 - theta)/theta}, lower = 0, upper = 1
      )$value
  }
  joe.alpha <-uniroot(
    function(theta) tau - fun.joe(theta), lower = 1, upper = 100
    )$root

  frank.cop <-copula::frankCopula(param = frank.alpha, dim = p)
  gumbel.cop <-copula::gumbelCopula(gumbel.alpha, dim = p)
  clayton.cop <-copula::claytonCopula(param = clayton.alpha ,dim = p)
  joe.cop <-copula::joeCopula(param = joe.alpha, dim = p)
  normal.cop <-copula::normalCopula(param = normal.cor, dim = p)
  t.cop <-copula::tCopula(param = normal.cor, dim = p, df = df)

  return(list(frank.cop = frank.cop,
              gumbel.cop = gumbel.cop,
              clayton.cop = clayton.cop,
              joe.cop = joe.cop,
              normal.cop = normal.cop,
              t.cop = t.cop))
}
```

Source Code G.2: CF_bivariate_2.R (Copulas choice by group classification rules ($p = 2$))

```
## -- filename: CF_bivariate_2.R
library(plotly)
library(copula)
source("Arc_copulas.R")
########################## Miss rate ##############################################
nx ←500
tau ←0.7; p ←2
mycops ←Arc_copula(p = p, tau = tau, df = 2)
bi_Clayton ←mycops$clayton.cop
bi_Frank ←mycops$frank.cop
bi_Gumbel ←mycops$gumbel.cop
bi_joe ←mycops$joe.cop
bi_t ←mycops$t.cop
bi_Normal ←mycops$normal.cop

set.seed(12)
test1 ←copula::rCopula(nx, copula = bi_Frank)
test2 ←copula::rCopula(nx, copula = bi_Gumbel)
test3 ←copula::rCopula(nx, copula = bi_Clayton)
test4 ←copula::rCopula(nx, copula = bi_joe)
test5 ←copula::rCopula(nx, copula = bi_Normal)
test6 ←copula::rCopula(nx, copula = bi_t)

par(
  mfrow = c(3, 2), cex.axis = 1.2,
  cex.lab = 1.2, mar = c(4, 4, 2,1), cex.main = 1.2
)
plot(test1[, 1], test1[, 2], main = "Frank copula", xlab = "", ylab = "")
plot(test2[, 1], test2[, 2], main = "Gumbel copula", xlab = "", ylab = "")
plot(test3[, 1], test3[, 2], main = "Clayton copula", xlab = "", ylab = "")
plot(test4[, 1], test4[, 2], main = "Joe copula", xlab = "", ylab = "")
plot(test5[, 1], test5[, 2], main = "Normal copula", xlab = "", ylab = "")
plot(test6[, 1], test6[, 2], main = "t copula, df = 2", xlab = "", ylab = "")
myBiCopulas ←list(
  "Frank" = bi_Frank, "Gumbel" = bi_Gumbel,
  "Clayton" = bi_Clayton, "Joe" = bi_joe,
  "Normal" = bi_Normal, "t" = bi_t
)
# Plots copulas with different margins
########################################
library(plotly)
mycolor ←matrix(c(0, "#FFFFFF",
```

```
                  1, "#2171B5"), nrow = 2, byrow = T)
for (k in 1:length(myBiCopulas)){
  cops ←myBiCopulas[[k]]
  print(k)
  dcops ←copula::mvdc(
    cops, margins = c("norm", "norm"),
    paramMargins = list(list(mean = 0, sd = 1), list(mean = 0, sd = 1))
  )
  x1 ←x2 ←seq(-3, 3, length = 100)
  v←c()
  for (i in x1){
    for (j in x2){
      v←c(v,copula::dMvdc(c(i,j), dcops) )
    }
  }
  f←t(matrix(v,nrow=100,byrow=TRUE))
  g ←plotly::plot_ly(x=x1,y=x2,z=f,type = "contour",colorscale = mycolor,
                     reversescale = F, showscale = F) %>%
    plotly::layout(xaxis=list(title="x1"),
                   yaxis=list(title="x2"),
                   title=paste(names(myBiCopulas)[k], "copulas"))
  print(g)
}

########################################
#### Get misrate of Copulas classification using eigenstructure
myBiRate ←list()
length(myBiRate) ←length(myBiCopulas)
names(myBiRate) ←names(myBiCopulas)
######################### Miss rate ############################################
m ←2e3; nx ←500; myweight ←c(2/5, 1/5, 2/5) #change the number
seed ←1234;
set.seed(seed)
for (i in 1:length(myBiCopulas)){
  cops ←myBiCopulas[[i]]
  print(i)
  re ←replicate(m, expr = {
    U ←copula::rCopula(nx, copula = cops)
    #### Get the estimate and generate reference samples
    frank.param ←coef(tryCatch(
      copula::fitCopula(
        copula = copula::frankCopula(dim = p), data = U, method = "mpl"
      ),
      error = function(ex){
        warning("MLE has error - ", ex)
```

```
    copula::fitCopula(
      copula = copula::frankCopula(dim = p), data = U, method = "itau"
    )
  }
)
)
frank.cop ←copula::frankCopula(param = frank.param, dim = p)
frank.Y ←copula::rCopula(nx, copula = frank.cop)
frank.Y2 ←copula::rCopula(2*nx, copula = frank.cop)
frank.Y4 ←copula::rCopula(4*nx, copula = frank.cop)
frank.Y8 ←copula::rCopula(8*nx, copula = frank.cop)
frank.Y16 ←copula::rCopula(16*nx, copula = frank.cop)
frank.Y32 ←copula::rCopula(32*nx, copula = frank.cop)

gumbel.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::gumbelCopula(dim = p), data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::gumbelCopula(dim = p), data = U, method = "itau"
    )
  }
)
)
gumbel.cop ←copula::gumbelCopula(param = gumbel.param, dim = p)
gumbel.Y ←copula::rCopula(nx, copula = gumbel.cop)
gumbel.Y2 ←copula::rCopula(2*nx, copula = gumbel.cop)
gumbel.Y4 ←copula::rCopula(4*nx, copula = gumbel.cop)
gumbel.Y8 ←copula::rCopula(8*nx, copula = gumbel.cop)
gumbel.Y16 ←copula::rCopula(16*nx, copula = gumbel.cop)
gumbel.Y32 ←copula::rCopula(32*nx, copula = gumbel.cop)

clayton.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::claytonCopula(dim = p), data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::claytonCopula(dim = p), data = U, method = "itau"
    )
  }
)
```

```
)
clayton.cop ←copula::claytonCopula(param = clayton.param, dim = p)
clayton.Y ←copula::rCopula(nx, copula = clayton.cop)
clayton.Y2 ←copula::rCopula(2*nx, copula = clayton.cop)
clayton.Y4 ←copula::rCopula(4*nx, copula = clayton.cop)
clayton.Y8 ←copula::rCopula(8*nx, copula = clayton.cop)
clayton.Y16 ←copula::rCopula(16*nx, copula = clayton.cop)
clayton.Y32 ←copula::rCopula(32*nx, copula = clayton.cop)

joe.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::joeCopula(dim = p), data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::joeCopula(dim = p), data = U, method = "itau"
    )
  }
)
)
joe.cop ←copula::joeCopula(param = joe.param, dim = p)
joe.Y ←copula::rCopula(nx, copula = joe.cop)
joe.Y2 ←copula::rCopula(2*nx, copula = joe.cop)
joe.Y4 ←copula::rCopula(4*nx, copula = joe.cop)
joe.Y8 ←copula::rCopula(8*nx, copula = joe.cop)
joe.Y16 ←copula::rCopula(16*nx, copula = joe.cop)
joe.Y32 ←copula::rCopula(32*nx, copula = joe.cop)

normal.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::normalCopula(dim = p), data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::normalCopula(dim = p), data = U, method = "itau"
    )
  }
)
)
normal.cop ←copula::normalCopula(param = normal.param, dim = p)
normal.Y ←copula::rCopula(nx, copula = normal.cop)
normal.Y2 ←copula::rCopula(2*nx, copula = normal.cop)
normal.Y4 ←copula::rCopula(4*nx, copula = normal.cop)
```

```
normal.Y8 ←copula::rCopula(8*nx, copula = normal.cop)
normal.Y16 ←copula::rCopula(16*nx, copula = normal.cop)
normal.Y32 ←copula::rCopula(32*nx, copula = normal.cop)

t.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::tCopula(
      dim = p, df.fixed = FALSE
    ),
    data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::tCopula(dim = p, dispstr = "un"),
      data = U, method = "itau.mpl"
    )
  })
)

t.cop ←copula::tCopula(
  param = t.param[1], dim = p, df = round(t.param[2])
)
t.Y ←copula::rCopula(nx, copula = t.cop)
t.Y2 ←copula::rCopula(2*nx, copula = t.cop)
t.Y4 ←copula::rCopula(4*nx, copula = t.cop)
t.Y8 ←copula::rCopula(8*nx, copula = t.cop)
t.Y16 ←copula::rCopula(16*nx, copula = t.cop)
t.Y32 ←copula::rCopula(32*nx, copula = t.cop)

lstX ←list(frank.Y, gumbel.Y, clayton.Y, joe.Y, normal.Y, t.Y)
lstX2 ←list(frank.Y2, gumbel.Y2, clayton.Y2, joe.Y2, normal.Y2, t.Y2)
lstX4 ←list(frank.Y4, gumbel.Y4, clayton.Y4, joe.Y4, normal.Y4, t.Y4)
lstX8 ←list(frank.Y8, gumbel.Y8, clayton.Y8, joe.Y8, normal.Y8, t.Y8)
lstX16 ←list(frank.Y16, gumbel.Y16, clayton.Y16,
             joe.Y16, normal.Y16, t.Y16
)
lstX32 ←list(frank.Y32, gumbel.Y32, clayton.Y32,
             joe.Y32, normal.Y32, t.Y32
)
lstCop ←list(frank.cop, gumbel.cop, clayton.cop,
             joe.cop, normal.cop, t.cop
)
c(
  Eigen = mul.discr.eigen(lstX, U)$pop,
```

```
    Eigen_reg = mul.discr.eigen.reg(lstX, U, w = rep(1, 3))$pop,
    Eigen_reg2 = mul.discr.eigen.reg(lstX2, U, w = rep(1, 3))$pop,
    Eigen_reg4 = mul.discr.eigen.reg(lstX4, U, w = rep(1, 3))$pop,
    Eigen_reg8 = mul.discr.eigen.reg(lstX8, U, w = rep(1, 3))$pop,
    Eigen_reg16 = mul.discr.eigen.reg(lstX16, U, w = rep(1, 3))$pop,
    Eigen_reg32 = mul.discr.eigen.reg(lstX32, U, w = rep(1, 3))$pop,
    Eigen_reg_w = mul.discr.eigen.reg(lstX, U, w = myweight)$pop,
    NP = mul.NP(lstCop, U)$pop)
  }
  )
  bi_rate_tmp ←t(
    apply(
      re, MARGIN = 1,
      FUN = function(arr) {
        c(mean(arr== "Population 1"),
          mean(arr=="Population 2"),
          mean(arr == "Population 3"),
          mean(arr== "Population 4"),
          mean(arr== "Population 5"),
          mean(arr== "Population 6")
        )
      }
    )
  )
  colnames(bi_rate_tmp) ←
    c("Frank", "Gumbel", "Clayton", "Joe", "Normal", "t")
  myBiRate[[i]] ←bi_rate_tmp
}
tmp ←myBiRate[[1]][, 1]
for (k in 2:6){
  tmp ←cbind(tmp, myBiRate[[k]][, k])
}
tmp
xtable::xtable(tmp, digits = 4)
tmp ←myBiRate[[1]]
# print the results.
for (k in 1:9){
  # k ←9
  tmp ←sapply(1:6, function(x) myBiRate[[x]][k, ])
  diag(tmp) ←NA
  tmp1 ←rbind(
    apply(tmp, MARGIN = 2, function(x) which.max(x)),
    apply(tmp, MARGIN = 2, function(x) max(x, na.rm = T))
  )
  colnames(tmp1) ←c("Frank", "Gumbel", "Clayton", "Joe", "Normal", "t")
```

```
  tmp1[1, ] ←case_when(tmp1[1, ] == 1 ~ "Frank",
                       tmp1[1, ] == 2 ~ "Gumbel",
                       tmp1[1, ] == 3 ~ "Clayton",
                       tmp1[1, ] == 4 ~ "Joe",
                       tmp1[1, ] == 5 ~ "Normal",
                       tmp1[1, ] == 6 ~ "t",
  )
  tmp1 ←xtable::xtable(tmp1, digits = 4)
  print(tmp1)
}
```

Source Code G.3: CF_fiveCopulas.R (Copulas choice by group classification rules ($p = 5$))

```
## --Filename: CF_fiveCopulas.R
########################## Miss rate ##########################################
library(copula)
source("Arc_copulas.R")
nx ←500
p ←5
five_Clayton ←claytonCopula(param = 2, dim = p)
X1 ←rCopula(500, five_Clayton)
five_Frank ←frankCopula(param = 2, dim = p)
five_Gumbel ←gumbelCopula(param = 2, dim = p)
five_joe ←joeCopula(param = 2, dim = p)
J ←c(-1, 0, 1, 2)
set.seed(11)
A ←matrix(sample(J, size = p*p, replace = T), nrow = p, byrow = T)
diag(A)[diag(A) <= 0] ←rep(10, sum(diag(A)==0))
S ←t(A) %*% A
S ←diag(1/sqrt(diag(S))) %*% S %*% diag(1/sqrt(diag(S)))
S ←round(S, digits = 1) # check with equal covariance.
eigen(S,only.values = T) # Check S is pd
five_t ←tCopula(S[upper.tri(S, diag = F)], dim = p, dispstr = "un",
                df = 2, df.fixed = TRUE)

five_Normal ←normalCopula(S[upper.tri(S, diag = F)], dim = p, dispstr = "un")
myfiveCopulas ←list(
  "Frank" = five_Frank, "Gumbel" = five_Gumbel,
  "Clayton" = five_Clayton, "Joe" = five_joe,
  "Normal" = five_Normal , "t" = five_t
)
#### Get misrate of Copulas classification using eigenstructure
myfiveRate ←list()
length(myfiveRate) ←length(myfiveCopulas)
names(myfiveRate) ←names(myfiveCopulas)
```

```
######################### Miss rate ##########################################
m ←2e3; nx ←1e3; myweight ←c(2/5, 1/5, 2/5) #change the number
print(m)
print(p)
seed ←1234;
set.seed(seed)
for (i in 1:length(myfiveCopulas)){
  cops ←myfiveCopulas[[i]]
  print(i)
  re ←replicate(m, expr = {
    U ←copula::rCopula(nx, copula = cops)
    #### Get the estimate and generate reference samples
    frank.param ←coef(tryCatch(
      copula::fitCopula(
        copula = copula::frankCopula(dim = p), data = U, method = "mpl"
      ),
      error = function(ex){
        warning("MLE has error - ", ex)
        copula::fitCopula(
          copula = copula::frankCopula(dim = p), data = U, method = "itau"
        )
      }
    )
    )
    frank.cop ←copula::frankCopula(param = frank.param, dim = p)
    frank.Y ←copula::rCopula(nx, copula = frank.cop)
    frank.Y2 ←copula::rCopula(2*nx, copula = frank.cop)
    frank.Y4 ←copula::rCopula(4*nx, copula = frank.cop)
    frank.Y8 ←copula::rCopula(8*nx, copula = frank.cop)
    frank.Y16 ←copula::rCopula(16*nx, copula = frank.cop)
    frank.Y32 ←copula::rCopula(32*nx, copula = frank.cop)

    gumbel.param ←coef(tryCatch(
      copula::fitCopula(
        copula = copula::gumbelCopula(dim = p), data = U, method = "mpl"
      ),
      error = function(ex){
        warning("MLE has error - ", ex)
        copula::fitCopula(
          copula = copula::gumbelCopula(dim = p), data = U, method = "itau"
        )
      }
    )
    )
    gumbel.cop ←copula::gumbelCopula(param = gumbel.param, dim = p)
```

```
gumbel.Y ←copula::rCopula(nx, copula = gumbel.cop)
gumbel.Y2 ←copula::rCopula(2*nx, copula = gumbel.cop)
gumbel.Y4 ←copula::rCopula(4*nx, copula = gumbel.cop)
gumbel.Y8 ←copula::rCopula(8*nx, copula = gumbel.cop)
gumbel.Y16 ←copula::rCopula(16*nx, copula = gumbel.cop)
gumbel.Y32 ←copula::rCopula(32*nx, copula = gumbel.cop)

clayton.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::claytonCopula(dim = p), data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::claytonCopula(dim = p), data = U, method = "itau"
    )
  })
)
if (clayton.param < 0) {clayton.param ←0}
clayton.cop ←copula::claytonCopula(param = clayton.param, dim = p)
clayton.Y ←copula::rCopula(nx, copula = clayton.cop)
clayton.Y2 ←copula::rCopula(2*nx, copula = clayton.cop)
clayton.Y4 ←copula::rCopula(4*nx, copula = clayton.cop)
clayton.Y8 ←copula::rCopula(8*nx, copula = clayton.cop)
clayton.Y16 ←copula::rCopula(16*nx, copula = clayton.cop)
clayton.Y32 ←copula::rCopula(32*nx, copula = clayton.cop)

joe.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::joeCopula(dim = p), data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::joeCopula(dim = p), data = U, method = "itau"
    )
  })
)
joe.cop ←copula::joeCopula(param = joe.param, dim = p)
joe.Y ←copula::rCopula(nx, copula = joe.cop)
joe.Y2 ←copula::rCopula(2*nx, copula = joe.cop)
joe.Y4 ←copula::rCopula(4*nx, copula = joe.cop)
joe.Y8 ←copula::rCopula(8*nx, copula = joe.cop)
joe.Y16 ←copula::rCopula(16*nx, copula = joe.cop)
joe.Y32 ←copula::rCopula(32*nx, copula = joe.cop)
```

```
normal.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::normalCopula(dim = p, dispstr = "un"),
    data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::normalCopula(dim = p, dispstr = "un"),
      data = U, method = "itau"
    )
  }
)
)
normal.cop ←copula::normalCopula(param = normal.param,
                                 dim = p, dispstr = "un"
)
normal.Y ←copula::rCopula(nx, copula = normal.cop)
normal.Y2 ←copula::rCopula(2*nx, copula = normal.cop)
normal.Y4 ←copula::rCopula(4*nx, copula = normal.cop)
normal.Y8 ←copula::rCopula(8*nx, copula = normal.cop)
normal.Y16 ←copula::rCopula(16*nx, copula = normal.cop)
normal.Y32 ←copula::rCopula(32*nx, copula = normal.cop)

t.param ←coef(tryCatch(
  copula::fitCopula(
    copula = copula::tCopula(dim = p, df.fixed = FALSE, dispstr = "un"),
    data = U, method = "mpl"
  ),
  error = function(ex){
    warning("MLE has error - ", ex)
    copula::fitCopula(
      copula = copula::tCopula(dim = p, dispstr = "un"),
      data = U, method = "itau.mpl"
    )
  }
)
)
t.cop ←copula::tCopula(param = t.param[-length(t.param)], dim = p,
                       df = round(t.param["df"]), dispstr = "un"
)
t.Y ←copula::rCopula(nx, copula = t.cop)
t.Y2 ←copula::rCopula(2*nx, copula = t.cop)
t.Y4 ←copula::rCopula(4*nx, copula = t.cop)
```

```
  t.Y8 ←copula::rCopula(8*nx, copula = t.cop)
  t.Y16 ←copula::rCopula(16*nx, copula = t.cop)
  t.Y32 ←copula::rCopula(32*nx, copula = t.cop)

  lstX ←list(frank.Y, gumbel.Y, clayton.Y, joe.Y, normal.Y, t.Y)
  lstX2 ←list(frank.Y2, gumbel.Y2, clayton.Y2, joe.Y2, normal.Y2, t.Y2)
  lstX4 ←list(frank.Y4, gumbel.Y4, clayton.Y4, joe.Y4, normal.Y4, t.Y4)
  lstX8 ←list(frank.Y8, gumbel.Y8, clayton.Y8, joe.Y8, normal.Y8, t.Y8)
  lstX16 ←list(frank.Y16, gumbel.Y16, clayton.Y16,
               joe.Y16, normal.Y16, t.Y16
  )
  lstX32 ←list(frank.Y32, gumbel.Y32, clayton.Y32,
               joe.Y32, normal.Y32, t.Y32
  )
  lstCop ←list(frank.cop, gumbel.cop, clayton.cop,
               joe.cop, normal.cop, t.cop
  )

  c(
    Eigen = mul.discr.eigen(lstX, U)$pop,
    Eigen_reg = mul.discr.eigen.reg(lstX, U, w = rep(1, 3))$pop,
    Eigen_reg2 = mul.discr.eigen.reg(lstX2, U, w = rep(1, 3))$pop,
    Eigen_reg4 = mul.discr.eigen.reg(lstX4, U, w = rep(1, 3))$pop,
    Eigen_reg8 = mul.discr.eigen.reg(lstX8, U, w = rep(1, 3))$pop,
    Eigen_reg16 = mul.discr.eigen.reg(lstX16, U, w = rep(1, 3))$pop,
    Eigen_reg32 = mul.discr.eigen.reg(lstX32, U, w = rep(1, 3))$pop,
    Eigen_reg_w = mul.discr.eigen.reg(lstX, U, w = myweight)$pop,
    NP = mul.NP(lstCop, U)$pop)
}
)
five_rate_tmp ←t(
  apply(re, MARGIN = 1,
        FUN = function(arr) {
          c(mean(arr== "Population 1"),
            mean(arr=="Population 2"),
            mean(arr == "Population 3"),
            mean(arr== "Population 4"),
            mean(arr== "Population 5"),
            mean(arr== "Population 6")
          )
        }
  )
)
colnames(five_rate_tmp) ←
  c("Frank", "Gumbel", "Clayton", "Joe", "Normal", "t")
```

```
  myfiveRate[[i]] ←five_rate_tmp
}
myfiveRate
tmp ←myfiveRate[[1]][, 1]
for (k in 2:6){
  tmp ←cbind(tmp, myfiveRate[[k]][, k])
}
tmp
xtable::xtable(tmp, digits = 4)
tmp ←myfiveRate[[1]]

for (k in 1:9){
  tmp ←sapply(1:6, function(x) myfiveRate[[x]][k, ])
  diag(tmp) ←NA
  tmp1 ←rbind(apply(tmp, MARGIN = 2, function(x) which.max(x)),
              apply(tmp, MARGIN = 2, function(x) max(x, na.rm = T))
  )
  colnames(tmp1) ←c("Frank", "Gumbel", "Clayton", "Joe", "Normal", "t")
  tmp1[1, ] ←dplyr::case_when(tmp1[1, ] == 1 ~ "Frank",
                              tmp1[1, ] == 2 ~ "Gumbel",
                              tmp1[1, ] == 3 ~ "Clayton",
                              tmp1[1, ] == 4 ~ "Joe",
                              tmp1[1, ] == 5 ~ "Normal",
                              tmp1[1, ] == 6 ~ "t",
  )

  tmp1 ←xtable::xtable(tmp1, digits = 4)
  print(tmp1)
}
```

G.2 Choosing copulas for ETF data

Source Code G.4: FDVLX_FDGRX.R (Analysis of FDVLX and FDGRX stock return)

```
library(dplyr)
library(ggplot2)
FDVLX ←read.csv(file = "Data/FDVLX.csv")
FDVLX ←FDVLX %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
  arrange(Date)
fund1 ←(diff(FDVLX$Close)/FDVLX$Close[-nrow(FDVLX)])

FDGRX ←read.csv(file = "Data/FDGRX.csv")
FDGRX ←FDGRX %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
```

```
  arrange(Date)
fund2 ←(diff(FDGRX$Close)/FDGRX$Close[-nrow(FDGRX)])
mydf← data.frame(fund1 = fund1, fund2 = fund2)

# plot
p ←ggplot2::ggplot(mydf, aes(fund1, fund2)) +
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 2) +
  mytheme + xlab("FDVLX") + ylab("FDGRX")
ggExtra::ggMarginal(p, type = "histogram", fill = "#9ECAE1")
###################################
U ←copula::pobs(mydf);
p ←ncol(U); nu ←nrow(U);
nx ←nu # sample size of simulated sample
nx ←16*nu # Change and rerun for classification
ggplot2::ggplot(data.frame(U), aes(fund1, fund2)) +
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 2) +
  mytheme + xlab("FDVLX") + ylab("FDGRX")

seed ←1234
par(mfrow = c(2, 3))
#
set.seed(seed)
frank.param ←coef(
  copula::fitCopula(
    copula = copula::frankCopula(dim = p), data = U, method = "mpl"
  )
)
frank.cop ←copula::frankCopula(param = frank.param, dim = p)
frank.Y ←copula::rCopula(nx, copula = frank.cop)
frank.g ←ggplot2::ggplot(data.frame(frank.Y), aes(X1, X2)) +
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 1) +
  mytheme + ggtitle("Frank") +
  theme(plot.title = element_text(face = "bold"),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
set.seed(seed)
#
gumbel.param ←coef(
  copula::fitCopula(
    copula = copula::gumbelCopula(dim = p), data = U, method = "mpl"
  )
)
gumbel.cop ←copula::gumbelCopula(param = gumbel.param, dim = p)
gumbel.Y ←copula::rCopula(nx, copula = gumbel.cop)
gumbel.g ←ggplot2::ggplot(data.frame(gumbel.Y), aes(X1, X2)) +
```

```
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 1) +
  mytheme + xlab(bquote(U[1])) + ylab(bquote(U[2])) + ggtitle("Gumbel") +
  theme(plot.title = element_text(face = "bold"),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
#
set.seed(seed)
clayton.param ←coef(
  copula::fitCopula(
    copula = copula::claytonCopula(dim = p), data = U, method = "mpl"
  )
)
clayton.cop ←copula::claytonCopula(param = clayton.param, dim = p)
clayton.Y ←copula::rCopula(nx, copula = clayton.cop)
clayton.g ←ggplot2::ggplot(data.frame(clayton.Y), aes(X1, X2)) +
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 1) +
  mytheme + xlab(bquote(U[1])) + ylab(bquote(U[2])) + ggtitle("Clayton") +
  theme(plot.title = element_text(face = "bold"),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
#
set.seed(seed)
joe.param ←coef(
  copula::fitCopula(
    copula = copula::joeCopula(dim= p), data = U, method = "mpl"
  )
)
joe.cop ←copula::joeCopula(param = joe.param, dim = p)
joe.Y ←copula::rCopula(nx, copula = joe.cop)
joe.g ←ggplot2::ggplot(data.frame(joe.Y), aes(X1, X2)) +
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 1) +
  mytheme + xlab(bquote(U[1])) + ylab(bquote(U[2])) + ggtitle("Joe") +
  theme(plot.title = element_text(face = "bold"),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
#
set.seed(seed)
normal.param ←coef(
  copula::fitCopula(
    copula = copula::normalCopula(dim= p), data = U, method = "mpl"
  )
)
normal.cop ←copula::normalCopula(param = normal.param, dim = p)
normal.Y ←copula::rCopula(nx, copula = normal.cop)
normal.g ←ggplot2::ggplot(data.frame(normal.Y), aes(X1, X2)) +
```

```
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 1) +
  mytheme + xlab(bquote(U[1])) + ylab(bquote(U[2])) + ggtitle("Normal") +
  theme(plot.title = element_text(face = "bold"),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
#
set.seed(seed)
t.param ←coef(
  copula::fitCopula(
    copula = copula::tCopula(dim= p), data = U, method = "mpl"
  )
)
t.cop ←copula::tCopula(
  param = c(t.param["rho.1"]), df = t.param["df"], dim = p
)
t.Y ←copula::rCopula(nx, copula = t.cop)
t.g ←ggplot2::ggplot(data.frame(t.Y), aes(X1, X2)) +
  ggplot2::geom_point(shape = 1, color = "#4292c6", size = 1) +
  mytheme + xlab(bquote(U[1])) + ylab(bquote(U[2])) + ggtitle("t") +
  theme(plot.title = element_text(face = "bold"),
        axis.title.x=element_blank(),
        axis.title.y=element_blank())
#
gridExtra::grid.arrange(grobs = list(frank.g, gumbel.g, clayton.g,
                                     joe.g, normal.g, t.g),
                        padding = unit(0, "line"),
                        layout_matrix = matrix(1:6, nrow = 3, byrow = T))

lstX ←list(frank.Y, gumbel.Y, clayton.Y, joe.Y, normal.Y, t.Y)
lstCop ←list(frank.cop, gumbel.cop, clayton.cop,
             joe.cop, normal.cop, t.cop)

re.eigen ←mul.discr.eigen.reg(lstX, U, w = rep(1/3, 3))

re.NP ←mul.NP(lstCop, U)

re_df ←data.frame(Eigen_reg = re.eigen$value,
                  NP = re.NP$value #,
                  # Eigen_reg_rm = unlist(lapply(lstX, FUN = function(x) {
                  # antieigen.reg.rm(x, U = U)
                  # })
                  # )
)

rownames(re_df) ←c("Frank", "Gumble", "Clayton", "Joe","Normal", "t")
```

```
re_df
c(
  Eigen_reg = re.eigen$pop,
  NP = re.NP$pop.
)


### Ratio criteria ####
mul.discr.eigen.ratio(lstX, U, region = T)
```

Source Code G.5: IWF_LRGF_ONEO_OUSA_PSQ.R (Analysis of IWF, LRGF, ONEO, OUSA and PSQ stock return)

```
library(ggplot2)
library(GGally)
library(copula)
library(dplyr)
IWF ←read.csv(file = "Data/IWF.csv")
IWF ←IWF %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
  arrange(Date)
fund1 ←(diff(IWF$Close)/IWF$Close[-nrow(IWF)])
#
LRGF ←read.csv(file = "Data/LRGF.csv")
LRGF ←LRGF %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
  arrange(Date)
fund2 ←(diff(LRGF$Close)/LRGF$Close[-nrow(LRGF)])
mydf← data.frame(fund1 = fund1, fund2 = fund2)
#
ONEO ←read.csv(file = "Data/ONEO.csv")
ONEO ←ONEO %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
  arrange(Date)
fund3 ←(diff(ONEO$Close)/ONEO$Close[-nrow(ONEO)])
#
OUSA ←read.csv(file = "Data/OUSA.csv")
OUSA ←OUSA %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
  arrange(Date)
fund4 ←(diff(OUSA$Close)/OUSA$Close[-nrow(OUSA)])
#
PSQ ←read.csv(file = "Data/PSQ.csv")
PSQ ←PSQ %>%
  mutate_at(c("Date"), function(x) as.Date(x)) %>%
```

```
  arrange(Date)
fund5 ←(diff(PSQ$Close)/PSQ$Close[-nrow(PSQ)])
mydf← data.frame(IWF = fund1, LRGF = fund2,
                 ONEO = fund3, OUSA = fund4, PSQ = fund5
)
mysymbol ←c("IWF", "LRGF", "ONEO", "OUSA", "PSQ")
GGally::ggpairs(
  mydf,
  diag = list(
    continuous = GGally::wrap(GGally::ggally_barDiag, fill = "#9ECAE1")
  ),
  lower = list(
    continuous = GGally::wrap(GGally::ggally_points,
                              shape = 1,
                              color = "#4292c6",
                              size = 1)
  ),
  upper = list(
    continuous = GGally::wrap(GGally::ggally_cor, method = "spearman",
                              fontface ="bold", colour = "black")
  )
) +
  ggplot2::theme_bw() + ggplot2::theme(panel.grid = element_blank(),
                                       axis.ticks = element_blank(),
                                       axis.line = element_blank(),
                                       axis.text = element_blank()
  )

###################################
U ←copula::pobs(mydf)
p ←ncol(U);
nu ←nrow(U)
nx ←16*nu
GGally::ggpairs(
  data.frame(U),
  upper = list(
    continuous = wrap(
      ggally_cor, method = "spearman", fontface ="bold", colour = "black")
  ),
  diag = list(continuous = "blankDiag"),
  lower = list(
    continuous = wrap(
      ggally_points, shape = 1, color = "#4292c6", size = 1)
  )
) +
```

```
  theme_bw() + ggplot2::theme(panel.grid = element_blank(),
                              axis.ticks = element_blank(),
                              axis.line = element_blank(),
                              axis.text = element_blank()
  )
###################################
seed ←1234
par(mfrow = c(2, 3))
set.seed(seed)
frank.param ←coef(
  copula::fitCopula(
    copula = copula::frankCopula(dim = p), data = U, method = "mpl"
  )
)
frank.cop ←copula::frankCopula(param = frank.param, dim = p)
frank.Y ←copula::rCopula(
  nx, copula = frank.cop
)
#
set.seed(seed)
gumbel.param ←coef(
  copula::fitCopula(
    copula = copula::gumbelCopula(dim = p), data = U, method = "mpl"
  )
)
gumbel.cop ←copula::gumbelCopula(param = gumbel.param, dim = p)
gumbel.Y ←copula::rCopula(
  nx, copula = gumbel.cop
)
#
set.seed(seed)
clayton.param ←coef(
  copula::fitCopula(
    copula = copula::claytonCopula(dim = p), data = U, method = "mpl"
  )
)
clayton.cop ←copula::claytonCopula(param = clayton.param, dim = p)
clayton.Y ←copula::rCopula(
  nx, copula = clayton.cop
)
#
set.seed(seed)
joe.param ←coef(
  copula::fitCopula(
    copula = copula::joeCopula(dim= p), data = U, method = "mpl"
```

```
  )
)
joe.cop ←copula::joeCopula(param = joe.param, dim = p)
joe.Y ←copula::rCopula(
  nx, copula = joe.cop
)
#
set.seed(seed)
normal.param ←coef(
  copula::fitCopula(
    copula = copula::normalCopula(
      dim= p, dispstr = "un"
    ), data = U, method = "mpl"
  )
)
normal.cop ←copula::normalCopula(
  param = normal.param, dim = p, dispstr = "un"
)
getSigma(normal.cop)
normal.Y ←copula::rCopula(
  nx, copula = normal.cop
)
#
set.seed(seed)
t.param ←coef(
  copula::fitCopula(
    copula = copula::tCopula(
      dim= p, df.fixed = FALSE, dispstr = "un"
    ), data = U, method = "mpl"
  )
)
t.cop ←copula::tCopula(
  param = t.param[1:(.5*p*(p-1))], df = t.param["df"], dim = p, dispstr = "un"
)
round(getSigma(t.cop), 3)
t.Y ←copula::rCopula(
  nx, copula = t.cop
)
lstX ←list(frank.Y, gumbel.Y, clayton.Y, joe.Y, normal.Y, t.Y)
lstCop ←list(frank.cop, gumbel.cop, clayton.cop,
             joe.cop, normal.cop, t.cop)

re.eigen ←mul.discr.eigen.reg(lstX, U, w = rep(1, 3))
re.NP ←mul.NP(lstCop, U)
re_df ←data.frame(Eigen_reg = re.eigen$value,
```

```
                  NP = re.NP$value)
rownames(re_df) ←c("Frank","Gumble", "Clayton", "Joe","Normal", "t")
re_df
c(
  Eigen_reg = re.eigen$pop,
  NP = re.NP$pop
)

mul.discr.eigen.ratio(lstX, U)
```

REFERENCES

[1] N. Henze, "Invariant tests for multivariate normality: a critical review," *Statistical Papers*, vol. 43, pp. 467–506, Oct 2002.

[2] R. Khattree and D. N. Naik, "Multivariate data reduction and discrimination with sas software," 01 2000.

[3] K. V. Mardia, "Measures of multivariate skewness and kurtosis with applications," *Biometrika*, vol. 57, no. 3, pp. 519–530, 1970.

[4] T. Móri, V. Rohatgi, and G. Szekely, "On multivariate skewness and kurtosis," *Theory of Probability and Its Applications*, vol. 38, Jan 1993.

[5] J. F. Malkovich and A. A. Afifi, "On tests for multivariate normality," *Journal of the American Statistical Association*, vol. 68, no. 341, pp. 176–179, 1973.

[6] J. A. Koziol, "A note on measures of multivariate kurtosis," *Biometrical Journal*, vol. 31, no. 5, pp. 619–624, 1989.

[7] L. Rayleigh, "On the problem of random vibrations, and of random flights in one, two, or three dimensions.," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science.*, 1919.

[8] J. A. Koziol, "A class of invariant procedures for assessing multivariate normality," *Biometrika*, vol. 69, no. 2, pp. 423–427, 1982.

[9] J. A. Koziol, "On assessing multivariate normality," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 45, no. 3, pp. 358–361, 1983.

[10] A. S. Paulson, P. Roohan, and P. Sullo, "Some empirical distribution function tests for multivariate normality," *Journal of Statistical Computation and Simulation*, vol. 28, pp. 15–30, July 1987.

[11] D. S. Moore and J. B. Stubblebine, "Chi-square tests for multivariate normality with application to common stock prices," *Communications in Statistics - Theory and Methods*, vol. 10, pp. 713–738, Jan. 1981.

[12] K.-T. Tsai and J. A. Koziol, "A correlation type procedure for assessing multivariate normality," *Communications in Statistics - Simulation and Computation*, vol. 17, pp. 637–651, Jan. 1988.

[13] G. S. Mudholkar, M. McDermott, and D. K. Srivastava, "A test of p-variate normality," *Biometrika*, vol. 79, no. 4, pp. 850–854, 1992.

[14] S. Csörgő, "Consistency of some tests for multivariate normality," *Metrika*, vol. 36, pp. 107–116, Dec. 1989.

[15] L. Baringhaus and N. Henze, "A consistent test for multivariate normality based on the empirical characteristic function," *Metrika*, vol. 35, pp. 339–348, Dec 1988.

[16] N. Henze and B. Zirkler, "A class of invariant consistent tests for multivariate normality," *Communications in Statistics - Theory and Methods*, vol. 19, pp. 3595–3617, Jan. 1990.

[17] A. W. Bowman and P. J. Foster, "Adaptive smoothing and density-based tests of multivariate normality," *Journal of the American Statistical Association*, vol. 88, no. 422, pp. 529–537, 1993.

[18] S. N. Roy, "On a heuristic method of test construction and its use in multivariate analysis," *The Annals of Mathematical Statistics*, vol. 24, no. 2, pp. 220–238, 1953.

[19] L.-X. Zhu, K.-T. Fang, and M. I. Bhatti, "On estimated projection pursuit-type crámer–von mises statistics," *Journal of Multivariate Analysis*, vol. 63, no. 1, pp. 1–14, 1997.

[20] L.-X. Zhu, H. L. Wong, and K.-T. Fang, "A test for multivariate normality based on sample entropy and projection pursuit," *Journal of Statistical Planning and Inference*, vol. 45, no. 3, pp. 373–385, 1995.

[21] Y. Kuwana and T. Kariya, "Lbi tests for multivariate normality in exponential power distributions," *Journal of Multivariate Analysis*, vol. 39, no. 1, pp. 117–134, 1991.

[22] T. Kariya and E. I. George, "Lbi tests for multivariate normality in curved families and mardia's test," *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, vol. 57, no. 3, pp. 440–451, 1995.

[23] J. Beirlant, D. Mason, and C. Vynckier, "Goodness-of-fit analysis for multivariate normality based on generalized quantiles," *Computational Statistics & Data Analysis*, vol. 30, no. 2, pp. 119–142, 1999.

[24] D. R. Cox and N. J. H. Small, "Testing multivariate normality," *Biometrika*, vol. 65, no. 2, pp. 263–272, 1978.

[25] C. Campbell and Y. Ying, *Learning with Support Vector Machines*. Synthesis lectures on artificial intelligence and machine learning, Morgan & Claypool, 2011.

[26] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[27] D. Coomans and D. Massart, "Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules," *Analytica Chimica Acta*, vol. 136, pp. 15–27, 1982.

[28] M. Sklar, "Fonctions de repartition an dimensions et leurs marges," *Publ. inst. statist. univ. Paris*, vol. 8, pp. 229–231, 1959.

[29] R. B. Nelsen, *An Introduction to Copulas*. No. 2 in Springer Series in Statistics, Springer, June 2007.

[30] K. Pearson, "Tables for statisticians and biometricians," *Biometrika*, 1974.

[31] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[32] T. W. Anderson and D. A. Darling, "A test of goodness of fit," *Journal of the American Statistical Association*, vol. 49, pp. 765–769, dec 1954.

[33] C.-C. Lin and G. S. Mudholkar, "A simple test for normality against asymmetric alternatives," *Biometrika*, vol. 67, no. 2, pp. 455–461, 1980.

[34] K. Murota and K. Takeuchi, "The studentized empirical characteristic function and its application to test for the shape of distribution," *Biometrika*, vol. 68, no. 1, pp. 55–65, 1981.

[35] R. Khattree and D. N. Naik, *Applied Multivariate Statistics with SAS Software, Second Edition*. USA: SAS Institute Inc., 2nd ed., 2018.

[36] D. N. Naik, "Ch. 26. diagnostic methods for univariate and multivariate normal data," in *Statistics in Industry*, vol. 22 of *Handbook of Statistics*, pp. 957–993, Elsevier, 2003.

[37] P. T. Ng, "Smoothing spline score estimation," *SIAM Journal on Scientific Computing*, vol. 15, pp. 1003–1025, sep 1994.

[38] A. K. Bera and P. T. Ng, "Tests for normality using estimated score function," *Journal of Statistical Computation and Simulation*, vol. 52, pp. 273–287, may 1995.

[39] S. Ghosh, "A new graphical tool to detect non-normality," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 4, pp. 691–702, 1996.

[40] D. D. Cox, "A penalty method for nonparametric estimation of the logarithmic derivative of a density function," *Annals of the Institute of Statistical Mathematics*, vol. 37, pp. 271–288, dec 1985.

[41] A. M. Kagan, Y. V. Linnik, and C. R. Rao, *Characterization Problems in Mathematical Statistics.* Wiley, 1973.

[42] "Gaussian inequalities," in *Springer Monographs in Mathematics*, pp. 49–64, Springer New York.

[43] P. McCullagh, "Does the moment-generating function characterize a distribution?," *The American Statistician*, vol. 48, no. 3, pp. 208–208, 1994.

[44] N. Balakrishnan, S. Kotz, and N. L. Johnson, *Continuous Univariate Distributions*, vol. 2. Wiley, 1995.

[45] G. Casella and R. L. Berger., *Statistical inference*. Thomson Learning, 2nd ed. ed., 2002.

[46] Z. Lun and R. Khattree, "An overlooked lomax-exponential connection," *Communications in Statistics - Theory and Methods*, vol. 51, pp. 26–28, Mar 2020.

[47] R. D'Agostino and E. S. Pearson, "Tests for departure from normality. empirical results for the distributions of b2 and $\sqrt{b1}$," *Biometrika*, vol. 60, no. 3, pp. 613–622, 1973.

[48] Y. Zhang, M. Zhou, and Y. Shao, "mvnormaltest: Powerful tests for multivariate normality."

[49] M. A. Stephens, "EDF statistics for goodness of fit and some comparisons," *Journal of the American Statistical Association*, vol. 69, pp. 730–737, sep 1974.

[50] J. Gross and U. Ligges, "nortest: Tests for normality."

[51] A. N. Pettitt, "Testing the normality of several independent samples using the anderson-darling statistic," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 26, no. 2, pp. 156–161, 1977.

[52] J. P. Royston, "An extension of shapiro and wilk's w test for normality to large samples," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 2, pp. 115–124, 1982.

[53] R. C. Team, "R: A language and environment for statistical computing," 2023.

[54] J. P. Royston, "The w test for normality," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 31, no. 2, pp. 176–180, 1982.

[55] P. Royston, "A remark on algorithm as 181: The w-test for normality," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 44, no. 4, pp. 547–551, 1995.

[56] H. Cramér and H. Wold, "Some theorems on distribution functions," *Journal of the London Mathematical Society*, vol. s1-11, no. 4, pp. 290–294, 1936.

[57] D. N. Naik and K. Plungpongpun, "A kotz-type distribution for multivariate statistical inference," pp. 111–124.

[58] T. K. Nayak, "Multivariate lomax distribution: Properties and usefulness in reliability theory," *Journal of Applied Probability*, vol. 24, no. 1, pp. 170–177, 1987.

[59] S. M., T. Hayakawa, and X. Fujikoshi, "Modern multivariate statistical analysis. a graduate course and handbook," *Biometrical Journal*, vol. 30, no. 7, pp. 873–873, 1988.

[60] A. C. Rencher, *Multivariate statistical inference and applications.* Wiley series in probability and statistics: Texts and references section, Wiley, 1998.

[61] A. Beck, *Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with Python and MATLAB*. Society for Industrial and Applied Mathematics, 2nd ed., Jan 2023.

[62] S. Csorgo, "Testing for normality in arbitrary dimension," *The Annals of Statistics*, vol. 14, jun 1986.

[63] K.-T. Fang, R.-Z. Li, and J.-J. Liang, "A multivariate version of ghosh's t3-plot to detect non-multinormality," *Computational Statistics & Data Analysis*, vol. 28, no. 4, pp. 371–386, 1998.

[64] C. R. Rao, "Some problems in the characterization of the multivariate normal distribution," in *A Modern Course on Statistical Distributions in Scientific Work*, pp. 1–13, Springer Netherlands, 1975.

[65] Z. Lun and R. Khattree, "Multivariate lomax (pareto type ii) and its related distributions."

[66] R. A. Fisher, "The precision of discriminant functions," *Annals of Eugenics*, vol. 10, no. 1, pp. 422–429, 1940.

[67] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[68] M. S. Bartlett and N. W. Please, "Discrimination in the case of zero mean differences," *Biometrika*, vol. 50, no. 1/2, pp. 17–21, 1963.

[69] C. Rao, *Advanced statistical methods in biometric research.* Wiley, 1952.

[70] R. Khattree, "A note on effects on the eigenstructure of a data matrix when deleting a subset of observations," *Journal of the indian society of agricultural statistics*, vol. 73, pp. 11–17, 01 2019.

[71] R. Khattree, "Antieigenvalues and antieigenvectors in statistics," *Journal of Statistical Planning and Inference*, vol. 114, pp. 131–144, 06 2003.

[72] R. Khattree, “On generalized antieigenvalue and antieigenmatrix of order r,” *American Journal of Mathematical and Management Sciences*, vol. 22, pp. 89–98, jan 2002.

[73] V. Durrleman, A. Nikeghbali, and T. Roncalli, “Which copula is the right one?,” *SSRN Electronic Journal*, 08 2000.

[74] P. Deheuvels, “Caractérisation complète des lois extrêmes multivariées et de la convergence des types extrêmes,” *Annales de l’ISUP*, vol. XXIII, no. 3-4, pp. 1–36, 1978.

[75] M. Hofert, I. Kojadinovic, M. Maechler, and J. Yan, “copula: Multivariate dependence with copulas.”