Author:
Chen, Fan

Title:
New methods for multivariate, survival and time series data with networks

New Methods For Multivariate, Survival And Time Series Data With Networks

By

Fan Chen

Department of Mathematics
University of Bristol

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Doctor of Philosophy in the Faculty of Science.

November 2021

Word count: twenty-four thousand

ABSTRACT

This thesis presents new methods for multivariate, survival and time series analysis with network.

We begin by describing a new method for computing the projection median, its influence curve and techniques for the production of projected quantile plots. A theoretical result on the form of the influence curve and numerical simulations are displayed. A comparison of the computational performance between the projection median and other existing multivariate medians is conducted as well. We also produce animated multidimensional projection quantile plots, and all results are generated using our R software package `Yamm`.

The second section introduces Bayesian wavelet approaches to estimate the density function and the hazard rate for right-censored data. To estimate the hazard rate, a Bayesian wavelet threshold approach and a Dirichlet process model are used, which shows good performance in our simulation examples. To improve the density estimates, we use the detailed covariance structure of the empirical wavelet coefficients, which enables a non-dyadic grid for the evaluation points.

A method to estimate survival functions using recurrent lifetimes is motivated in the third section, which allows the use of covariate information of individuals to construct a network structure of separate clusters. Our method shows an improvement on estimation performance when the number of data points is not large enough for good performance using standard methods.

The last section provides a model for analysing multivariate time series based on the structure of a network and exogenous regressors. Our model allows the target series to be regressed by previous time lags of itself and its neighbours, as well as another multivariate time series, which is related to the target one on the same network. It is shown numerically that our model has a good prediction performance with few parameters.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Professor Guy Nason, who has been really helpful in both my research progress and daily life since I started my PhD journey four year ago. He is always patient and has given me invaluable guidance on the methodology of carrying out the research and presenting work clearly, which keeps me motivated.

Special thanks to my husband, Chang, for his continuous support, especially during the final few months of writing up when we knew our baby would come into our family after 10 months. I am also grateful to my parents for their love and understanding.

Last but not least, I would like to thank to the Chinese Scholarship Council, as well as the Bristol University Department of Mathematics, for given me the financial support to finish my PhD study.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE **Page**

CHAPTER 1

INTRODUCTION

Methods for analysing high-dimensional data have increased in availability for practical analysis. Meanwhile, the use of networks has been applied in many statistical methods. This thesis focuses on the following four topics: a multivariate median analysis, density and hazard estimation using wavelet methods, survival function estimation and multivariate time series analysis with networks.

Multivariate medians are robust estimates of the center of a multivariate distribution, which are frequently used in practical data analysis. Chapter 2 introduces our new method to compute the projection median, yamm, which is shown theoretically to be equivalent to the one proposed by Durocher and Kirkpatrick (2005) and generalised by Basu et al. (2012). For the first time, we provide a theoretical result on form of the influence curve for the projection median, accompanied by numerical simulations. The theoretical computational complexity for a variety of medians is also explained in this chapter, where we also present some results of running times and accuracy of estimation for real implementations of different medians. Our `Yamm` R package provides users

with functions to compute the projection median according to the different methods, which also introduces functions to produce animated plots of two- and three-dimensional sets' projected quantiles.

There is considerable use of wavelet methods in statistics that have been applied to density and hazard rate function estimation. Based on the Bayesian threshold method proposed by Silverman and Johnstone (2005b), Chapter 3 details our new methods to improve the accuracy of hazard rate estimation for right-censored data. Furthermore, a non-parametric Bayesian method with Dirichlet process prior and a bootstrap aggregating approach are applied, which produces a better hazard estimation performance in simulation examples compared to the presmoothed method introduced by Lopez-de Ullibarri and Jacome (2013). Based upon Herrick et al.'s work (2001), which exploits the non-stationary variance structure of the wavelet coefficients, Chapter 3 also uses the detailed covariance structure of the empirical wavelet coefficients to estimate the density function. We propose a multivariate version of the "mixture of Gaussians" prior distribution in the Bayesian wavelet thresholding approach to attempt to improve performance.

Chapter 4 carries out the survival estimates analysis using recurrent survival lifetimes, which we do forming groups of individuals according to their separate covariate information. Basically, we construct a network with several separate clusters of individuals and assume the lifetimes of individuals in the same cluster are from the same independent distribution to improve the survival estimates for all individuals. Comparing to the AG model (Andersen and Gill, 1982) (i.e. a generalised Cox's regression model), the displayed simulation

examples show that our method usually has better performance when few lifetimes per individual are available.

The analysis of high-dimensional time series based on the network structure of target nodes has become popular recently. Based on the generalised network autoregressive model proposed by Knight et al. (2020), the last chapter provides our new model, which also includes exogenous regressors. Basically, the target series is regressed by the previous time lags of itself and its neighbours, as well as another network time series. Real applications of wind and COVID data show that our new model has a better out-of-sample forecasting performance than some other multivariate time series models.

CHAPTER 2

YET ANOTHER MULTIVARIATE MEDIAN

This chapter is joint work with Professor Guy Nason and has been published (Chen and Nason, 2020b).

2.1 Introduction

This chapter introduces a new formulation of, and method of computation for the projection median. Additionally, we explore its behaviour on a specific bivariate set up, providing the first theoretical result on form of the influence curve for the projection median, accompanied by numerical simulations.

Via new simulations we comprehensively compare our performance with an established method for computing the projection median, as well as other existing multivariate medians displayed in section 2.4. We focus on answering questions about accuracy and computational speed, whilst taking into account the underlying dimensionality. Such considerations are vitally important in situations where the data set is large, or where the operations have to be repeated many times and some well-known techniques are extremely computationally

expensive.

In section 2.5, we briefly describe our R package that includes our new methods and novel functionality to produce animated multidimensional projection quantile plots, and also exhibit its use on some high-dimensional data examples.

2.2 Literature Review

The median is an estimator of location that is robust, i.e. not heavily influenced by outlying values, which are, loosely speaking, points that are far from the main body of the data. Let $\mathbf{x} = (x_1, \ldots, x_k)^T$ be a mutually independent and identically distributed (i.i.d.) sample of length $k \in \mathbb{N}$ from a univariate distribution with distribution function F. The univariate population *median* functional $M(F)$ is

$$M(F) = \inf\left\{x : F(x) \geq 1/2\right\} = \sup\left\{x : F(x) \leq 1/2\right\}. \tag{2.1}$$

There are several equivalent definitions of the univariate median that all yield same unique value of true median μ for a distribution F with a bounded and continuous density $f(\mu)$ at μ.

For multivariate data there is no natural ordering of the data to enable the choice of the middle observation in the same way as for one-dimensional data. However, several different multivariate median concepts have been developed that retain some characteristics of the univariate median. For example, an early extension of the multivariate median was suggested by Hayford (1902),

which is simply the component-wise median, also known as the vector of marginal medians. The spatial median, also known as the L_1 median, (Weber, 1929) and Tukey's median (Tukey, 1975) are two other popular variants. Oja's median (Oja, 1983) provides an alternative to the spatial median, but it is known to be more computationally expensive than other choices. These, and others, are reviewed in Small (1990), Chaudhuri and Sengupta (1993), and Oja (2013). We briefly review some of them here next, not least as we use them later in our simulation study.

2.2.1 Component-wise median

Let $\mathbf{X} = \left(\mathbf{x}_1, \ldots, \mathbf{x}_k\right)^T$ be an n-dimensional i.i.d. sample with distribution function $F : \mathbb{R}^n \to \mathbb{R}$. We assume that the n marginal distributions have bounded densities $f_1(\mu_1), \ldots, f_n(\mu_n)$ at the uniquely defined marginal medians $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$. The component-wise median, also known as the marginal sample median, $M_C(\mathbf{X}) \in \mathbb{R}^n$ minimises

$$k^{-1} \sum_{i=1}^{k} \Big\{ \big(|x_{i1} - m_1| + \cdots + |x_{in} - m_n| \big) - \big(|x_{i1}| + \cdots + |x_{in}| \big) \Big\}, \tag{2.2}$$

the sum of component-wise distances over $\mathbf{m} \in \mathbb{R}^n$, where $\mathbf{m} = (m_1, \ldots, m_n)$. The corresponding population functional, $M_C(F)$, for the vector of population medians minimises

$$E \Big\{ \big(|x_1 - m_1| + \cdots + |x_n - m_n| \big) - \big(|x_1| + \cdots + |x_n| \big) \Big\}. \tag{2.3}$$

2.2.2 Spatial Median

The spatial median $M_S(\mathbf{X})$, also known as the L_1 median, minimises

$$k^{-1}\sum_{i=1}^{k}\Big\{||\mathbf{x}_i-\mathbf{m}||-||\mathbf{x}_i||\Big\}, \tag{2.4}$$

over $\mathbf{m}\in\mathbb{R}^n$, where $||\mathbf{m}||^2=\sum_{i=1}^{n}m_i^2$ is the (squared) Euclidean norm. The corresponding functional spatial median, $M_S(F)$, minimises

$$E_F\Big\{||\mathbf{x}-\mathbf{m}||-||\mathbf{x}||\Big\}. \tag{2.5}$$

2.2.3 Oja's median

Let $\mathbf{X}=\big(\mathbf{x}_1,\ldots,\mathbf{x}_k\big)^T$ be an i.i.d. sample in $\mathbb{R}^n$ with distribution function $F:\mathbb{R}^n\to\mathbb{R}$. The volume of the n-variate simplex determined by the $n+1$ vertices $(\mathbf{m}_1,\ldots,\mathbf{m}_{n+1})$ is

$$V(\mathbf{m}_1,\ldots,\mathbf{m}_{n+1})=\frac{1}{p!}\left|\det\begin{pmatrix}1 & \cdots & 1\\ \mathbf{m}_1 & \cdots & \mathbf{m}_{n+1}\end{pmatrix}\right|. \tag{2.6}$$

The Oja median, $M_O(\mathbf{X})$, minimises

$$\binom{k}{n}^{-1}\sum_{i_1<\cdots<i_n}V(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_n},\mathbf{m}), \tag{2.7}$$

over $\mathbf{m}\in\mathbb{R}^n$. The corresponding functional $M_O(F)$ minimises

$$E_F\Big\{V(\mathbf{x}_{i_1},\ldots,\mathbf{x}_{i_n},\mathbf{m})\Big\}. \tag{2.8}$$

2.2.4 Tukey's median

Let $\mathbf{X}=\big(\mathbf{x}_1,\ldots,\mathbf{x}_k\big)^T$ be an i.i.d. sample of size k in $\mathbb{R}^n$ with distribution function $F:\mathbb{R}^n\to\mathbb{R}$. Let $\mathscr{H}$ be the class of all closed half spaces in $\mathbb{R}^n$. For each $H\in\mathscr{H}$,

define the empirical distribution

$$\hat{F}(H) = n^{-1} \sum_{i=1}^{k} \mathbb{I}(\mathbf{x}_i \in H), \tag{2.9}$$

where $\mathbb{I}$ is the usual indicator function. Then, define the *depth*, $D(\boldsymbol{\mu})$, of a point $\boldsymbol{\mu} \in \mathbb{R}^n$ within the dataset, to be the infinum of $\hat{F}(H)$, that is taken over all closed half spaces H for which $\boldsymbol{\mu} \in H$. Tukey's median is defined as the set of points $\boldsymbol{\mu}$ of maximal depth.

2.3 The Projection Median

This section introduces our new method for computing the projection median, yamm. We prove that yamm is equivalent to the projection median, as defined by Durocher and Kirkpatrick (2005) in $\mathbb{R}^2$ and then generalised to higher dimensions by Basu et al. (2012). We also explore, theoretically and numerically, the statistical behaviour of yamm using a mixture of two bivariate normal distributions.

2.3.1 Review of the projection median

2.3.1.1 Projection median in $\mathbb{R}^2$

Let $\mathbf{X}$ be a multiset of points in $\mathbb{R}^2$ and $\theta \in [0, 2\pi)$ be an angle. Let $\mathbf{X}_\theta$ denote the multiset defined by the projection of $\mathbf{X}$ onto the unit vector $u_\theta = (\cos\theta, \sin\theta)$, so

$$\mathbf{X}_\theta = \{u_\theta \langle \mathbf{x}, u_\theta \rangle \mid \mathbf{x} \in \mathbf{X}\}, \tag{2.10}$$

where $\langle \cdot \rangle$ denotes the usual inner product.

The projection median of a non-empty finite set $\mathbf{X}$ with points in $\mathbb{R}^2$ (Durocher and Kirkpatrick, 2005) is

$$M_P(\mathbf{X}) = \pi^{-1} \int_0^{2\pi} \text{med}(\mathbf{X}_\theta)\, d\theta, \tag{2.11}$$

where $\text{med}(\mathbf{X}_\theta) \in \mathbb{R}^2$ is the median of the projection of $\mathbf{X}$ onto the line through the origin, parallel to u_θ.

2.3.1.2 Generalisation of the projection median

Given a fixed positive integer, $n \geq 2$, and a finite set of points $\mathbf{X}$ in $\mathbb{R}^n$, the n-dimensional projection median of $\mathbf{X}$ (Basu et al., 2012) is

$$M_P(\mathbf{X}) = n \frac{\int_{\mathbf{X}^{n-1}} \text{med}(\mathbf{X_a}) d\mathbf{a}}{\int_{\mathbf{X}^{n-1}} d\mathbf{a}} = n \int_{\mathbf{X}^{n-1}} \text{med}(\mathbf{X_a}) df(\mathbf{a}), \tag{2.12}$$

where $\mathbf{X}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x}|| = 1\}$ is the unit n-dimensional hypersphere, $\text{med}(\mathbf{X_a})$ is the median of the projection of $\mathbf{X}$ onto the line through the origin parallel to $\mathbf{a}$, and f is the normalised uniform measure over $\mathbf{X}^{n-1}$. Hence, for a point $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbf{X}^{n-1}$, the n-dimensional spherical coordinates are given by

$$\begin{aligned}
x_1 &= \cos\theta_1 \\
x_2 &= \sin\theta_1 \cos\theta_2 \\
x_3 &= \sin\theta_1 \sin\theta_2 \cos\theta_3 \\
&\cdots \\
x_{n-1} &= \sin\theta_1 \cdots \sin\theta_{n-2} \cos\theta_{n-1} \\
x_n &= \sin\theta_1 \cdots \sin\theta_{n-2} \sin\theta_{n-1},
\end{aligned} \tag{2.13}$$

where each angle $\theta_1, \theta_2, \ldots, \theta_{n-2}$ has a range of π and θ_{n-1} has range of 2π.

Also, the normalised uniform measure f over $\mathbf{X}^{n-1}$ is given by

$$df = \frac{\mathrm{d}_{\mathbf{X}^{n-1}} V}{\int_0^{\pi} \int_0^{\pi} \cdots \int_0^{2\pi} d_{\mathbf{X}^{n-1}} V}, \tag{2.14}$$

where $\mathrm{d}_{\mathbf{X}^{n-1}} V = \sin^{n-2}\theta_1 \sin^{n-3}\theta_2 \cdots \sin\theta_{n-2}\,\mathrm{d}\theta_1 \mathrm{d}\theta_2 \cdots \mathrm{d}\theta_{n-1}$ is the volume element of the $(n-1)$-sphere.

Basu et al. (2012) proved that the projection median has a breakdown point of 1/2 for all $n \geq 2$.

2.3.2 Yet Another Multivariate Median (Yamm)

Let $\mathbf{X} = \left(\mathbf{x}_1, \ldots, \mathbf{x}_k\right)^T \in \mathbb{R}^{k \times n}$ be a random sample of size $k \in \mathbb{N}$, $\mathbf{x}_i \in \mathbb{R}^n$. Let $\mathbf{a}$ be a $n \times 1$ projection vector of unit length, 1_k be the $k \times 1$ vector of ones and $\boldsymbol{\mu}$ a shift vector of length n. Let $\mathbf{y}$ be the projection of $\mathbf{X}$ onto $\mathbf{a}$ after $\mathbf{X}$ has been shifted by $\boldsymbol{\mu}$:

$$\mathbf{y} = (\mathbf{X} - 1_k \boldsymbol{\mu}^T)\mathbf{a}, \tag{2.15}$$

where $\mathbf{y} \in \mathbb{R}^k$. The univariate median m of the projected points $\mathbf{y}$ is

$$m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a}) = m(\mathbf{y}). \tag{2.16}$$

Now define the integral

$$M_{\mathbf{X},m}(\boldsymbol{\mu}) = \int_{\{\mathbf{a}:\mathbf{a}^T\mathbf{a}=1\}} m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})^2 d\mathbf{a}. \tag{2.17}$$

The yamm estimator of location for $\mathbf{X}$ is

$$\hat{\boldsymbol{\mu}} = \text{yamm}(\mathbf{X}) = \text{argmin}_{\boldsymbol{\mu}} M_{\mathbf{X},m}(\boldsymbol{\mu}). \tag{2.18}$$

Equations (2.17) and (2.18) illustrate the rationale behind yamm. Intuitively, if the shift vector $\boldsymbol{\mu}$ is far away from the true "middle" of the dataset, then the magnitude of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$, as well as the integral $M_{\mathbf{X},m}(\boldsymbol{\mu})$, will be large. By contrast, a smaller $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ can be obtained when the $\boldsymbol{\mu}$ is moving closer to the true "middle" of the data set.

Instead of computing the squared value of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ for the integral, we also considered the absolute value as an alternative. However, this leads to similar numerical results. This is because we want to find the shifted vector μ as our multivariate median. Basically, our yamm is produced by projecting a set of multidimensional points onto a unit direction vector and minimising the univariate median of the projected points among all directions. The purpose of using the absolute or squared univariate median is to avoid the cancellation of the negative and positive projected medians. Both methods are able to do this well. Hence, in most situations, both methods will give similar results. However, the squaring method may not be suitable when there are some large outliers in the projected points, which will overemphasise the effect of the outliers comparing to using the absolute loss.

We now generate two polar plots of the absolute value of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$, when $\boldsymbol{\mu}$ is both close to, and far away, from the true median, respectively. A random two-dimensional dataset with $k = 100$ points was generated, whose Tukey's

median computed as $(2.78, 8.16)$. Here, the Tukey median is to be interpreted as a "sensible" middle of the data set. The shift vector $\boldsymbol{\mu}$ is set to be $(2.2, 8)$ and $(2, 7.5)$ respectively, and for each plot, two thousand random projections were used to calculate the univariate median $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$, using methods to be explained in Section 2.3.4. Figure 2.1 shows that when $\boldsymbol{\mu}$ is near Tukey's median, the magnitude of each $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ is less than 0.65, while a larger value, ranging from 0 to 1.2, is shown in the figure when $\boldsymbol{\mu}$ is far away from the median. Overall, when integrated the quantity involving the $\boldsymbol{\mu}$ is closer to the Tukey median it gives a smaller result.

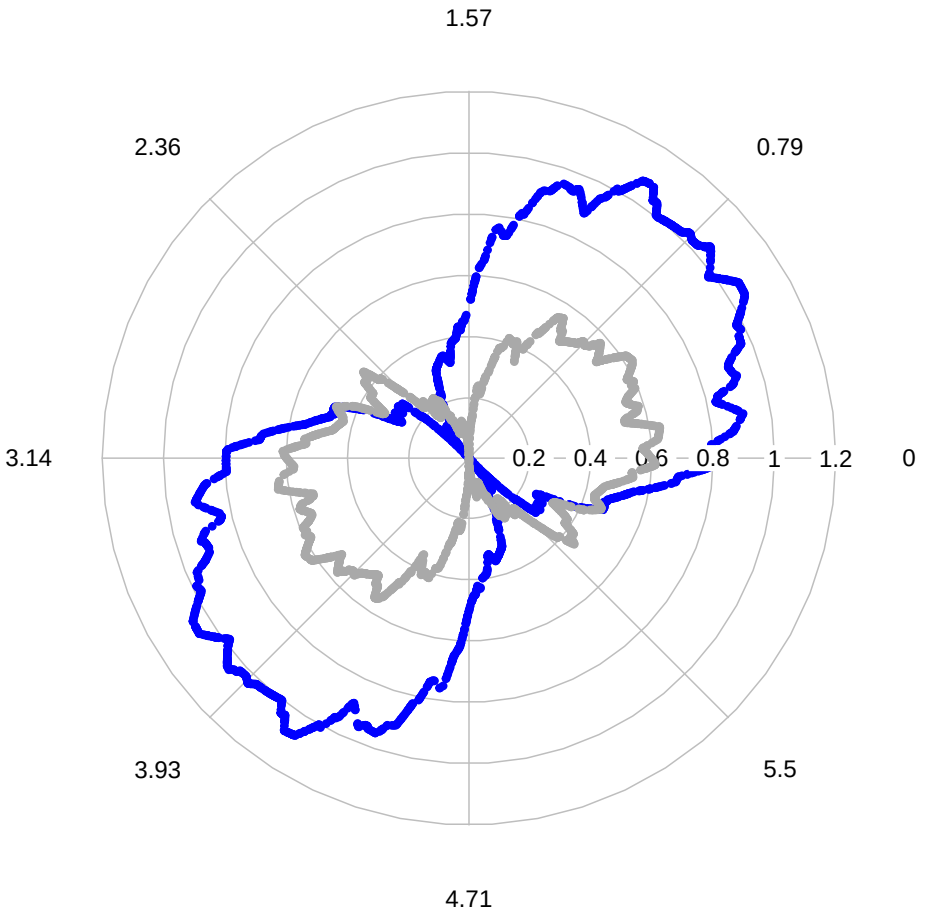


Figure 2.1: Polar plot (in radians) of the magnitude of $m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})$ with Grey line: $\boldsymbol{\mu} = (2.2, 8)$ and Blue line: $\boldsymbol{\mu} = (2, 7.5)$.

The projection median and yamm definitions seem similar, as both project the multiset onto the line passing through the origin, and then take the median. However, the projection median integrates med($\mathbf{X_a}$) directly over the unit hypersphere in $\mathbb{R}^n$, whereas yamm minimises the objective function $M_{\mathbf{X},m}(\boldsymbol{\mu}) \in \mathbb{R}$ over the shift vector $\boldsymbol{\mu}$. Despite these differences, the following theorem shows that the projection median and yamm are identical.

Theorem. *For any finite multiset $\boldsymbol{X} \subseteq \mathbb{R}^n$ with $n \geq 2$, yamm is equivalent to the projection median.*

Proof. Let $\mathbf{X} = \left(\mathbf{x}_1, \ldots, \mathbf{x}_k\right)^T \in \mathbb{R}^{k \times n}$ be a random sample of size $k \in \mathbb{N}$, $\mathbf{x}_i \in \mathbb{R}^n$. Let $\mathbf{a}$ be a $n \times 1$ projection vector of unit length, 1_k be the $k \times 1$ vector of ones and $\boldsymbol{\mu}$ a shift vector of length n. $\mathbf{y}$ is the projection of $\mathbf{X}$ onto $\mathbf{a}$ after $\mathbf{X}$ has been shifted by $\boldsymbol{\mu}$.

We now show the proof of the equivalence in 2-dimensional case, and then generalise it to the higher dimensions. Let

$$\mathbf{a} = \mathbf{a}_\theta = \left(\cos\theta, \sin\theta\right)^T, \tag{2.19}$$

$$\boldsymbol{\mu} = \left(\mu_1, \mu_2\right)^T, \tag{2.20}$$

then we have

$$\mathbf{y} = (\mathbf{X} - 1_k \boldsymbol{\mu}^T)\,\mathbf{a}_\theta \quad \text{and} \quad m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a}_\theta) = m(\mathbf{y}). \tag{2.21}$$

Hence, our objective function becomes

$$M_{\mathbf{X},m}(\boldsymbol{\mu}) = \int_{\theta=0}^{2\pi} m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a}_\theta)^2 d\theta \tag{2.22}$$

$$= \int_{\theta=0}^{2\pi} \left[m\{(\mathbf{X} - 1_k \boldsymbol{\mu}^T)\mathbf{a}_\theta\} \right]^2 d\theta \tag{2.23}$$

$$= \int_{\theta=0}^{2\pi} \left\{ m(\mathbf{X}\mathbf{a}_\theta) - \boldsymbol{\mu}^T \mathbf{a}_\theta \right\}^2 d\theta \tag{2.24}$$

$$= \int_{\theta=0}^{2\pi} m(\mathbf{X}\mathbf{a}_\theta)^2 - 2\, m(\mathbf{X}\mathbf{a}_\theta)(\boldsymbol{\mu}^T \mathbf{a}_\theta) + (\boldsymbol{\mu}^T \mathbf{a}_\theta)^2 d\theta. \tag{2.25}$$

To minimise $M_{\mathbf{X},m}(\boldsymbol{\mu})$, we want

$$0 = \frac{\partial}{\partial \mu} \int_{\theta=0}^{2\pi} m(\mathbf{X}\mathbf{a}_\theta)^2 - 2\, m(\mathbf{X}\mathbf{a}_\theta)(\boldsymbol{\mu}^T \mathbf{a}_\theta) + (\boldsymbol{\mu}^T \mathbf{a}_\theta)^2 d\theta \tag{2.26}$$

$$\iff 0 = \int_{\theta=0}^{2\pi} \frac{\partial}{\partial \mu} \left\{ m(\mathbf{X}\mathbf{a}_\theta)^2 - 2\, m(\mathbf{X}\mathbf{a}_\theta)(\boldsymbol{\mu}^T \mathbf{a}_\theta) + (\boldsymbol{\mu}^T \mathbf{a}_\theta)^2 \right\} d\theta \tag{2.27}$$

$$\iff 0 = \int_{\theta=0}^{2\pi} \frac{\partial}{\partial \mu} \left\{ -2\, m(\mathbf{X}\mathbf{a}_\theta)(\boldsymbol{\mu}^T \mathbf{a}_\theta) + (\boldsymbol{\mu}^T \mathbf{a}_\theta)^2 \right\} d\theta \tag{2.28}$$

$$\iff \int_{\theta=0}^{2\pi} \frac{\partial}{\partial \mu} 2\, m(\mathbf{X}\mathbf{a}_\theta)(\boldsymbol{\mu}^T \mathbf{a}_\theta) \mathrm{d}\theta = \int_{\theta=0}^{2\pi} \frac{\partial}{\partial \mu} (\boldsymbol{\mu}^T \mathbf{a}_\theta)^2 d\theta \tag{2.29}$$

$$\iff \int_{\theta=0}^{2\pi} 2\mathbf{a}_\theta\, m(\mathbf{X}\mathbf{a}_\theta) d\theta = \int_{\theta=0}^{2\pi} 2\mathbf{a}_\theta (\boldsymbol{\mu}^T \mathbf{a}_\theta) d\theta \tag{2.30}$$

$$\iff \int_{\theta=0}^{2\pi} \mathbf{a}_\theta\, m(\mathbf{X}\mathbf{a}_\theta) d\theta = \int_{\theta=0}^{2\pi} \mathbf{a}_\theta (\boldsymbol{\mu}^T \mathbf{a}_\theta) d\theta. \tag{2.31}$$

Here, $\mathbf{a}_\theta\, m(\mathbf{X}\mathbf{a}_\theta)$ is the projection median of the multiset $\mathbf{X}$ on $\mathbf{a}_\theta$ in $\mathbb{R}^2$, which is defined in Equation (2.11) and denoted by $\mathrm{med}(\mathbf{X}_\theta)$. Also, when $\mathbf{a}_\theta = \left(\cos\theta, \sin\theta\right)^T$, and $\boldsymbol{\mu} = \left(\mu_1, \mu_2\right)^T$, we have

$$\int_{\theta=0}^{2\pi} \mathbf{a}_\theta (\boldsymbol{\mu}^T \mathbf{a}_\theta) d\theta = \pi \boldsymbol{\mu}. \tag{2.32}$$

Hence,

$$\int_{\theta=0}^{2\pi} \text{med}(\mathbf{X}_\theta) d\theta = \pi \boldsymbol{\mu} \tag{2.33}$$

$$\frac{1}{\pi} \int_{\theta=0}^{2\pi} \text{med}(X_\theta) d\theta = \boldsymbol{\mu}, \tag{2.34}$$

which shows that the shift vector is the projection median in $\mathbb{R}^2$ minimising our objective function $M_{\mathbf{X},m}(\boldsymbol{\mu})$.

Our proof for higher dimensions has a similar structure. For $n > 2$, let $\boldsymbol{\mu} = \left(\mu_1, \mu_2, \ldots, \mu_n\right)^T$, and $\mathbf{a} = \mathbf{a}_{\theta_1,\theta_2,\ldots,\theta_{n-1}} = \left(a_1, a_2, \ldots, a_n\right)^T$, such that

$$\begin{aligned}
a_1 &= \cos\theta_1 \\
a_2 &= \sin\theta_1 \cos\theta_2 \\
a_3 &= \sin\theta_1 \sin\theta_2 \cos\theta_3 \\
&\cdots \\
a_{n-1} &= \sin\theta_1 \cdots \sin\theta_{n-2} \cos\theta_{n-1} \\
a_n &= \sin\theta_1 \cdots \sin\theta_{n-2} \sin\theta_{n-1}.
\end{aligned} \tag{2.35}$$

Taking the volume element of the hypersphere into account, we obtain the objective function as follows

$$M_{\mathbf{X},m}(\boldsymbol{\mu}) = \int_{\theta_{n-1}=0}^{2\pi} \int_{\theta_{n-2}=0}^{\pi} \cdots \int_{\theta_1=0}^{\pi} m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a})^2 \sin^{n-2}(\theta_1) \sin^{n-3}(\theta_2) \cdots \sin(\theta_{n-2})\, d\theta_1 \cdots d\theta_{n-2}\, d\theta_{n-1}. \tag{2.36}$$

To minimise $M_{\mathbf{X},m}(\boldsymbol{\mu})$ we require

$$\frac{\partial}{\partial \mu} M_{\mathbf{X},m}(\boldsymbol{\mu}) = 0. \tag{2.37}$$

After some manipulations similar to the two-dimensional case, we obtain

$$\begin{aligned}&\int_0^{2\pi}\int_0^{\pi}\cdots\int_0^{\pi}\operatorname{med}\left(\mathbf{X}_{\theta_1,\ldots,\theta_{n-1}}\right)\sin^{n-2}(\theta_1)\cdots\sin(\theta_{n-2})d\theta_1\ldots d\theta_{n-2}\,d\theta_{n-1}\\&=\int_0^{2\pi}\int_0^{\pi}\cdots\int_0^{\pi}\mathbf{a}(\boldsymbol{\mu}^T\mathbf{a})\sin^{n-2}(\theta_1)\cdots\sin(\theta_{n-2})d\theta_1\ldots d\theta_{n-2}\,d\theta_{n-1}.\end{aligned}\tag{2.38}$$

Plugging the projection vector $\mathbf{a}$ into the right hand side of Equation (2.38), we have

$$\begin{aligned}&\int_0^{2\pi}\int_0^{\pi}\cdots\int_0^{\pi}\operatorname{med}\left(\mathbf{X}_{\theta_1,\ldots,\theta_{n-1}}\right)\sin^{n-2}(\theta_1)\cdots\sin(\theta_{n-2})d\theta_1\ldots d\theta_{n-2}\,d\theta_{n-1}\\&=n^{-1}\Big\{\int_0^{\pi}\sin^{n-2}(\theta_1)d\theta_1\cdots\int_0^{\pi}\sin(\theta_{n-2})d\theta_{n-2}\int_0^{2\pi}d\theta_{n-1}\Big\}\,\boldsymbol{\mu},\end{aligned}\tag{2.39}$$

which is the definition of the projection median in higher dimensions. This means the shift vector $\boldsymbol{\mu}$ minimising our objective function $M_{\mathbf{X},m}(\boldsymbol{\mu})$ is the projection median in $\mathbb{R}^n$ with $n \geq 2$.

□

2.3.3 Yamm behaviour on a bivariate normal mixture

To gain insight about the theoretical behaviour of yamm we study the case of yamm applied to a mixture of two bivariate normals, where one is thought of as the bulk and the other as the outlier of the distribution. Such a setup enables us to evaluate the robustness of yamm. We numerically and theoretically assess the influence curve when moving the outlier far from the bulk.

The bivariate mixture is a toy example. It is probably the simplest one that allows us to evaluate the response of the yamm to outliers, which does not

include any other free parameters that may affect the behaviour. Our example provides insight on the behaviour of the yamm, but it is not representative of real life. Even in some two-dimensional real problems, there are sometimes other behaviours such as multiple clusters.

2.3.3.1 Bivariate mixture setup

Let $\mathbf{X}_1 \sim N(\boldsymbol{\nu}_1, \Sigma_1)$ and $\mathbf{X}_2 \sim N(\boldsymbol{\nu}_2, \Sigma_2)$ be independent bivariate normal random variables, where $\mathbf{X}_1 = (X_{11}, X_{12})^T$, $\mathbf{X}_2 = (X_{21}, X_{22})^T$ with mean vector $\boldsymbol{\nu}_1 = (\nu_{11}, \nu_{12})^T$ and $\boldsymbol{\nu}_2 = (\nu_{21}, \nu_{22})^T$. Let $\mathbf{R}(\theta)$ be a rotation matrix with angle θ given by

$$\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{2.40}$$

We are interested in the first row of this matrix, which describes the projection onto direction θ. Let $\mathbf{Y}_i = (Y_{i1}, Y_{i2})^T = \mathbf{R}(\mathbf{X}_i - \boldsymbol{\mu})$ for $i = 1, 2$ respectively, where $\boldsymbol{\mu} = (\mu_1, \mu_2)^T$ is a shift vector mentioned in (2.15). Basic multivariate theory shows that

$$\mathbf{Y}_i \sim N\Big\{\mathbf{R}(\boldsymbol{\nu}_i - \boldsymbol{\mu}),\, \mathbf{R}\Sigma_i\mathbf{R}^T\Big\}, \qquad \text{for } i = 1, 2. \tag{2.41}$$

Denote $\mathbf{Y}_i = (Y_{i1}, Y_{i2})^T$, Y_{i1} is the first entry of $\mathbf{Y}_i$ for $i = 1, 2$. Then, it is immediate that $Y_{i1} \sim N(s_i, \sigma_i^2)$, where

$$s_1 = (\nu_{11} - \mu_1)\cos\theta - (\nu_{12} - \mu_2)\sin\theta \text{ and } \sigma_1^2 = (\mathbf{R}\Sigma_1\mathbf{R}^T)_{1,1}, \tag{2.42}$$

$$s_2 = (\nu_{21} - \mu_1)\cos\theta - (\nu_{22} - \mu_2)\sin\theta \text{ and } \sigma_2^2 = (\mathbf{R}\Sigma_2\mathbf{R}^T)_{1,1}. \tag{2.43}$$

The mixture distribution that we study is

$$f_W(w_1, w_2) = (1-\epsilon) f_{\mathbf{X}_1}(w_1, w_2) + \epsilon f_{\mathbf{X}_2}(w_1, w_2), \tag{2.44}$$

where $f_{\mathbf{X}_i}$ is the density of X_i, and $\epsilon \in [0,1]$, is typically small. Here, $f_{\mathbf{X}_1}$ is considered to be the bulk of the distribution and $f_{\mathbf{X}_2}$ the outlier.

2.3.3.2 Projected distribution

Based on the bivariate setup above, the projected distribution is

$$f_Y(y) = (1-\epsilon)\phi_{s_1,\sigma_1^2}(y) + \epsilon\phi_{s_2,\sigma_2^2}(y), \tag{2.45}$$

where $s_1, s_2, \sigma_1^2, \sigma_2^2$ are as above and ϕ is the standard normal density.

The distribution function of the projected $Y(\theta)$ is

$$F_Y(y) = (1-\epsilon)\Phi_{s_1,\sigma_1^2}(y) + \epsilon\Phi_{s_2,\sigma_2^2}(y), \tag{2.46}$$

where Φ is the standard normal distribution function. We require the median of the projected distribution, i.e. find

$$y_m(\epsilon, \theta, s_1, s_2, \Sigma_1, \Sigma_2) \text{ such that } F_Y(y_m) = 1/2. \tag{2.47}$$

Finding an analytic exact solution for y_m is difficult. Hence, we will simplify the problem and assume that $\Sigma_1 = \Sigma_2 = I_2$, the identity matrix. Since $\mathbf{R}(\theta)$ is an orthogonal matrix, this means that $\sigma_1^2 = \sigma_2^2 = 1$ and (2.46) becomes

$$F_Y(y) = (1-\epsilon)\Phi(y - s_1) + \epsilon\Phi(y - s_2). \tag{2.48}$$

For small ϵ, we know that the median should be close to the median of the bulk, so the median of F_Y should be close to s_1, the median of the first component of the mixture in Equation (2.48).

2.3.3.3 Theoretical approximation of Yamm on the mixture

We derive a theoretically based approximation to the empirical influence function. We proceed by using a Taylor series expansion of $F_Y(y)$ around s_1, the quantity we know is close to our median:

$$\begin{aligned} F_Y(y) \approx & \left[1+\epsilon-\epsilon \operatorname{Erfc}\{(s_1-s_2)/\sqrt{2}\}\right]/2 \\ & +(2\pi)^{-1/2}\left[1-\epsilon+\epsilon\exp\{-(s_1-s_2)^2/2\}\right](y-s_1) \\ & +O\{(y-s_1)^2\}, \end{aligned} \tag{2.49}$$

where $\operatorname{Erfc}(y) = 2\pi^{-1/2}\int_y^\infty e^{-t^2}\,dt$. When y is close to s_1, Equation (2.49) is approximately equal to 1/2 when ϵ is small, which is the behaviour we expect.

To find an approximation to the median we solve $F_Y\{y_m(\theta)\} = 1/2$. Ignoring remainders, subtracting 1/2 off both sides of Equation (2.49) gives

$$\frac{\epsilon}{2}\left[\operatorname{Erfc}\{(s_1-s_2)/\sqrt{2}\}-1\right] = \frac{\left[1-\epsilon+\epsilon\exp\{-(s_1-s_2)^2/2\}\right](y_m-s_1)}{\sqrt{2\pi}}, \tag{2.50}$$

and then

$$y_m(\theta) \approx s_1 + \frac{\epsilon\sqrt{\pi/2}\left[\operatorname{Erfc}\{(s_1-s_2)/\sqrt{2}\}-1\right]}{\left[1-\epsilon+\epsilon\exp\{-(s_1-s_2)^2/2\}\right]}. \tag{2.51}$$

Now using

$$\text{Erfc}\{(s_1 - s_2)/\sqrt{2}\} = 2\Phi\{(s_2 - s_1)/\sqrt{2}\}, \tag{2.52}$$

and $\exp\{-(s_1 - s_2)^2/2\} = \sqrt{2\pi}\phi(s_1 - s_2)$, we can write

$$y_m(\theta) \approx s_1 + \frac{\epsilon\sqrt{\pi/2}\left(2\Phi\{(s_2 - s_1)/\sqrt{2}\} - 1\right)}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(s_2 - s_1)}. \tag{2.53}$$

For small ϵ the denominator is close to 1. From Equations (2.42) and (2.43), we can write:

$$s_2 - s_1 = (v_{21} - v_{11})\cos\theta - (v_{22} - v_{12})\sin\theta = \delta_1\cos\theta - \delta_2\sin\theta, \tag{2.54}$$

where $\delta_1 = v_{21} - v_{11}$ and $\delta_2 = v_{22} - v_{12}$. Thus

$$\begin{aligned} y_m(\theta) \approx & \left\{(v_{11} - \mu_1)\cos\theta - (v_{12} - \mu_2)\sin\theta\right\} \\ & + \frac{\epsilon\sqrt{\pi/2}\left[2\Phi\{(\delta_1\cos\theta - \delta_2\sin\theta)/\sqrt{2}\} - 1\right]}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta - \delta_2\sin\theta)}. \end{aligned} \tag{2.55}$$

According to Equation (2.17), our job is to find the optimal $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)^T$, which minimises

$$M = \int_0^{2\pi} y_m^2(\theta)\, d\theta. \tag{2.56}$$

The integrand involves the standard normal distribution function, which is tricky to handle analytically. Hence, we use the approximation, $\phi(z) \approx (1 + \cos z)/2\pi$, for $-\pi < z < \pi$, for the standard normal density, Johnson et al. (1995), which enables the following proposition.

Proposition. *Let* $\boldsymbol{X}_1 = (X_{11}, X_{12})^T$ *and* $\boldsymbol{X}_2 = (X_{21}, X_{22})^T$. *Suppose that* $\boldsymbol{X}_1 \sim N(\boldsymbol{\nu}_1, \Sigma_1)$ *and* $\boldsymbol{X}_2 \sim N(\boldsymbol{\nu}_2, \Sigma_2)$ *independently, where* $\boldsymbol{\nu}_1 = (\nu_{11}, \nu_{12})^T$ *and* $\boldsymbol{\nu}_2 = (\nu_{21}, \nu_{22})^T$, *respectively. Let the mixture, W, of* $\boldsymbol{X}_1$ *and* $\boldsymbol{X}_2$ *be*

$$f_W(w_1, w_2) = (1-\epsilon) f_{\boldsymbol{X}_1}(w_1, w_2) + \epsilon f_{\boldsymbol{X}_2}(w_1, w_2),$$

where $\epsilon \in [0,1]$ *is considered small.*

An approximation of the yamm estimator, $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)$, *is*

$$\begin{aligned} \mu_1^* &= \nu_{11} + \pi^{-1/2} R\epsilon\Big(1 - R^2/32 + R^4/1536\Big)\cos\alpha, \\ \mu_2^* &= \nu_{12} + \pi^{-1/2} R\epsilon\Big(1 - R^2/32 + R^4/1536\Big)\sin\alpha, \end{aligned} \tag{2.57}$$

where $R^2 = (\delta_1^2 + \delta_2^2)$, $\delta_1 = \nu_{21} - \nu_{11}$, $\delta_2 = \nu_{22} - \nu_{12}$ *and* $\alpha = \arctan(\delta_2/\delta_1)$. *The approximation we use is valid whenever* $\big|R\cos(\theta + \alpha)\big| < \sqrt{2}\pi$, *where* θ *is the projection direction when computing yamm. This inequality is true for all* θ *whenever* $R < \sqrt{2}\pi$.

Proof. According to Section 2.3.3.3, we need to find the optimal $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)^T$ minimising

$$M = \int_0^{2\pi} y_m^2(\theta) d\theta, \tag{2.58}$$

where

$$\begin{aligned} y_m \approx{}& \Big\{(\nu_{11} - \mu_1)\cos\theta - (\nu_{12} - \mu_2)\sin\theta\Big\} \\ &+ \frac{\epsilon\sqrt{\pi/2}\left[2\{\Phi\{(\delta_1\cos\theta - \delta_2\sin\theta)/\sqrt{2}\} - 1\right]}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta - \delta_2\sin\theta)}. \end{aligned} \tag{2.59}$$

We do it by splitting the integrand into three terms:

$$J_1(\theta) = \Big\{(\nu_{11} - \mu_1)\cos\theta - (\nu_{12} - \mu_2)\sin\theta\Big\}^2, \tag{2.60}$$

and

$$\begin{aligned} J_2(\theta) =& 2\Big\{(\nu_{11} - \mu_1)\cos\theta - (\nu_{12} - \mu_2)\sin\theta\Big\} \\ & \frac{\epsilon\sqrt{\pi/2}\left[2\{\Phi\{(\delta_1\cos\theta - \delta_2\sin\theta)/\sqrt{2}\} - 1\right]}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta - \delta_2\sin\theta)}, \end{aligned} \tag{2.61}$$

and

$$J_3(\theta) = \frac{\pi\epsilon^2\left[2\{\Phi\{(\delta_1\cos\theta - \delta_2\sin\theta)/\sqrt{2}\} - 1\right]^2/2}{\left\{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta - \delta_2\sin\theta)\right\}^2}. \tag{2.62}$$

Now, we compute the integration term by term. The first term integration is

$$\begin{aligned} \int_0^{2\pi} J_1(\theta)\,d\theta &= \int_0^{2\pi} \Big\{(\nu_{11} - \mu_1)^2\cos^2\theta - 2(\nu_{11} - \mu_1)(\nu_{12} - \mu_2)\sin\theta\cos\theta \\ &\qquad + (\nu_{12} - \mu_2)^2\sin^2\theta\Big\}\,\mathrm{d}\theta \\ &= \pi\Big\{(\nu_{11} - \mu_1)^2 + (\nu_{12} - \mu_2)^2\Big\}. \end{aligned} \tag{2.63}$$

The key part of the second integrand is the "-1" part of Equation (2.61), which we define

$$J_{2,1}(\theta) = \frac{(\nu_{11} - \mu_1)\cos\theta - (\nu_{12} - \mu_2)\sin\theta}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta - \delta_2\sin\theta)}. \tag{2.64}$$

Now

$$J_{2,1}(\theta + \pi) = \frac{(\nu_{11} - \mu_1)\cos(\theta + \pi) - (\nu_{12} - \mu_2)\sin(\theta + \pi)}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi\{\delta_1\cos(\theta + \pi) - \delta_2\sin(\theta + \pi)\}} \tag{2.65}$$

$$= \frac{-\Big\{(\nu_{11} - \mu_1)\cos\theta - (\nu_{12} - \mu_2)\sin\theta\Big\}}{1 - \epsilon - \sqrt{2\pi}\epsilon\phi\{-(\delta_1\cos\theta - \delta_2\sin\theta)\}} \tag{2.66}$$

$$= -J_{2,1}(\theta), \tag{2.67}$$

as $\cos(\theta+\pi) = -\cos\theta$, $\sin(\theta+\pi) = -\sin\theta$ and $\phi(-x) = \phi(x)$. So,

$$\int_0^{2\pi} J_{2,1}(\theta)\,d\theta = \int_0^{\pi} J_{2,1}(\theta)\,\mathrm{d}\theta + \int_{\pi}^{2\pi} J_{2,1}(\theta)\,d\theta \tag{2.68}$$

$$= \int_0^{\pi} J_{2,1}(\theta)\,d\theta + \int_0^{\pi} J_{2,1}(\theta+\pi)\,d\theta \tag{2.69}$$

$$= \int_0^{\pi} J_{2,1}(\theta)\,d\theta - \int_0^{\pi} J_{2,1}(\theta)\,d\theta = 0. \tag{2.70}$$

Hence, we only need to look at the non-"-1" part of Equation (2.61), and, in fact, redefine J_2 to omit that term. So, we look at

$$J_2(\theta) = 2\{(\nu_{11}-\mu_1)\cos\theta - (\nu_{12}-\mu_2)\sin\theta \times \frac{\epsilon\sqrt{2\pi}\Phi\{(\delta_1\cos\theta - \delta_2\sin\theta)/\sqrt{2}\}}{1-\epsilon-\sqrt{2\pi}\epsilon\phi(\delta_1\cos\theta-\delta_2\sin\theta)} \tag{2.71}$$

$$= 2\epsilon\sqrt{2\pi}\Phi\{(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}J_{2,1}(\theta). \tag{2.72}$$

Note

$$J_2(\theta+\pi) = -2\epsilon\sqrt{2\pi}\Phi\{-(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}J_{2,1}(\theta) \tag{2.73}$$

$$= -2\epsilon\sqrt{2\pi}\left[1-\Phi\{(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}\right]J_{2,1}(\theta) \tag{2.74}$$

$$= 2\epsilon\sqrt{2\pi}\Phi\{(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}J_{2,1}(\theta) - 2\epsilon\sqrt{2\pi}J_{2,1}(\theta) \tag{2.75}$$

$$= J_2(\theta) - 2\epsilon\sqrt{2\pi}J_{2,1}(\theta). \tag{2.76}$$

Hence

$$\int_0^{2\pi} J_2(\theta)\,\mathrm{d}\theta = \int_0^{\pi} J_2(\theta)\,d\theta + \int_{\pi}^{2\pi} J_2(\theta)\,d\theta \tag{2.77}$$

$$= \int_0^{\pi} J_2(\theta)\,d\theta + \int_0^{\pi} J_2(\theta+\pi)\,d\theta \tag{2.78}$$

$$= 2\int_0^{\pi} J_2(\theta)\,d\theta - 2\epsilon\sqrt{2\pi}\int_0^{\pi} J_{2,1}(\theta)d\theta. \tag{2.79}$$

Followed by Equation (2.79), we now consider the integral $2\int_0^{\pi} J_2(\theta)\,d\theta$ and

$2\epsilon\sqrt{2\pi}\int_0^\pi J_{2,1}(\theta)\,d\theta$ separately.

$$2\int_0^\pi J_2(\theta)\,d\theta = 4\epsilon\sqrt{2\pi}\int_0^\pi \Phi\{(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}J_{2,1}(\theta)d\theta. \tag{2.80}$$

So, consider the following integration

$$\int_0^\pi \Phi\{(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}J_{2,1}(\theta)d\theta, \tag{2.81}$$

with

$$J_{2,1}(\theta) = \frac{(\nu_{11}-\mu_1)\cos\theta-(\nu_{12}-\mu_2)\sin\theta}{1-\epsilon\Big\{1+\sqrt{2\pi}\phi(\delta_1\cos\theta-\delta_2\sin\theta)\Big\}} \tag{2.82}$$

$$= \Big\{(\nu_{11}-\mu_1)\cos\theta-(\nu_{12}-\mu_2)\sin\theta\Big\}(1-x)^{-1} \tag{2.83}$$

$$= \Big\{(\nu_{11}-\mu_1)\cos\theta-(\nu_{12}-\mu_2)\sin\theta\Big\}\sum_{i=0}^{\infty}x^i, \tag{2.84}$$

by the Binomial expansion and where $x=\epsilon\{1+\sqrt{2\pi}\phi(\delta_1\cos\theta-\delta_2\sin\theta)\}$.

As we want to investigate the situation when ϵ is small, hence, for simplicity, we consider the linear terms of ϵ and ignore those terms involving higher order of ϵ. Then, we only extract the first term of $J_{2,1}(\theta)$ (i.e. $i=0$) when integrating $J_2(\theta)$. So, Equation (2.81) becomes

$$\int_0^\pi \Phi\{(\delta_1\cos\theta-\delta_2\sin\theta)/\sqrt{2}\}\Big\{(\nu_{11}-\mu_1)\cos\theta-(\nu_{12}-\mu_2)\sin\theta\Big\}d\theta. \tag{2.85}$$

Now, we define

$$\cos\alpha=\frac{\delta_1}{R} \quad\text{and}\quad \sin\alpha=\frac{\delta_2}{R}, \tag{2.86}$$

$$\cos\beta=\frac{\nu_{12}-\mu_2}{R'} \quad\text{and}\quad \sin\beta=\frac{\nu_{11}-\mu_1}{R'}, \tag{2.87}$$

with $R^2 = (\delta_1^2 + \delta_2^2)$, $(R')^2 = \{(\nu_{11} - \mu_1)^2 + (\nu_{12} - \mu_2)^2\}$, $\alpha = \arctan(\delta_2/\delta_1)$ and $\beta = \arctan\{(\nu_{11} - \mu_1)/(\nu_{12} - \mu_2)\}$. Using the addition formulae of trigonometric functions, then integrating by parts, (2.85) becomes

$$\int_0^{\pi} -\Phi\{R\cos(\theta+\alpha)/\sqrt{2}\}\, R' \sin(\theta-\beta) d\theta \tag{2.88}$$

$$= \left[\Phi\{R\cos(\theta+\alpha)/\sqrt{2}\}\, R' \cos(\theta-\beta)\right]_0^{\pi}$$

$$+ \frac{RR'}{\sqrt{2}} \int_0^{\pi} \phi\{R\cos(\theta+\alpha)/\sqrt{2}\} \sin(\theta+\alpha)\cos(\theta-\beta) d\theta. \tag{2.89}$$

After some manipulations, we have

$$\left[\Phi\{R\cos(\theta+\alpha)/\sqrt{2}\}\, R' \cos(\theta-\beta)\right]_0^{\pi} = -(\nu_{12} - \mu_2). \tag{2.90}$$

Considering the latter part of (2.89), we find an approximation to the standard normal density mentioned by Johnson et al. (1995), that is

$$\phi(z) \approx \frac{1}{2\pi}(1 + \cos z), \tag{2.91}$$

with $-\pi < z < \pi$. Then, we expand $\cos z$ at $z = 0$ with Maclaurin series and use the first three terms for further calculation:

$$\phi(z) \approx \frac{1}{2\pi}\left\{1 + 1 - \frac{z^2}{2!} + \frac{z^4}{4!} + O(z^6)\right\} \tag{2.92}$$

$$= \frac{1}{\pi}\left\{1 - \frac{z^2}{4} + \frac{z^4}{48} + O(z^6)\right\}. \tag{2.93}$$

Let $z = R\cos(\theta+\alpha)/\sqrt{2}$, with $\left|R\cos(\theta+\alpha)/\sqrt{2}\right| < \pi$, we have

$$\frac{RR'}{\sqrt{2}}\int_0^{\pi}\phi\{R\cos(\theta+\alpha)/\sqrt{2}\}\sin(\theta+\alpha)\cos(\theta-\beta)d\theta \tag{2.94}$$

$$\approx \frac{RR'}{\sqrt{2}\pi}\int_0^{\pi}\left\{1-\frac{R^2\cos^2(\theta+\alpha)}{8}+\frac{R^4\cos^4(\theta+\alpha)}{192}\right\}\sin(\theta+\alpha)\cos(\theta-\beta)d\theta \tag{2.95}$$

$$\begin{aligned}= \frac{RR'}{\sqrt{2}\pi}\Bigg\{&\int_0^{\pi}\sin(\theta+\alpha)\cos(\theta-\beta)d\theta\\ &-\int_0^{\pi}\frac{R^2\cos^2(\theta+\alpha)\sin(\theta+\alpha)\cos(\theta-\beta)}{8}d\theta\\ &+\int_0^{\pi}\frac{R^4\cos^4(\theta+\alpha)\sin(\theta+\alpha)\cos(\theta-\beta)}{192}d\theta\Bigg\}\end{aligned} \tag{2.96}$$

$$= \frac{RR'}{\sqrt{2}}\left\{\frac{\pi}{2}\sin(\alpha+\beta)-\frac{\pi}{64}\sin(\alpha+\beta)+\frac{\pi}{3072}\sin(\alpha+\beta)\right\} \tag{2.97}$$

$$= \frac{RR'}{2\sqrt{2}}\sin(\alpha+\beta)\left(1-R^2/32+R^4/1536\right). \tag{2.98}$$

Hence,

$$2\int_0^{\pi}J_2(\theta)d\theta \approx 4\epsilon\sqrt{2\pi}\left\{-(\nu_{12}-\mu_2)+\frac{RR'}{2\sqrt{2}}\sin(\alpha+\beta)\left(1-\frac{R^2}{32}+\frac{R^4}{1536}\right)\right\}. \tag{2.99}$$

Now, consider the second part of $\int_0^{2\pi}J_2(\theta)d\theta$, that is

$$2\epsilon\sqrt{2\pi}\int_0^{\pi}J_{2,1}(\theta)d\theta = 2\epsilon\sqrt{2\pi}\int_0^{\pi}\left\{(\nu_{11}-\mu_1)\cos\theta-(\nu_{12}-\mu_2)\sin\theta\right\}\sum_{i=0}^{\infty}x^i d\theta, \tag{2.100}$$

where $x=\epsilon\{1+\sqrt{2\pi}\phi(\delta_1\cos\theta-\delta_2\sin\theta)\}$.

As mentioned before, we only consider the linear terms of ϵ. Hence, Equation (2.100) is approximately

$$2\epsilon\sqrt{2\pi}\int_0^{\pi}\left\{(\nu_{11}-\mu_1)\cos\theta-(\nu_{12}-\mu_2)\sin\theta\right\}d\theta = -4\epsilon\sqrt{2\pi}(\nu_{12}-\mu_2). \tag{2.101}$$

Combining the results of Equation (2.99) and Equation (2.101), we have

$$\int_0^{2\pi} J_2(\theta)d\theta \approx 4\epsilon\sqrt{2\pi}\left\{-(\nu_{12}-\mu_2)+\frac{RR'}{2\sqrt{2}}\sin(\alpha+\beta)\left(1-\frac{R^2}{32}+\frac{R^4}{1536}\right)\right\}$$
$$-\{-4\epsilon\sqrt{2\pi}(\nu_{12}-\mu_2)\} \quad (2.102)$$
$$=2\epsilon\sqrt{\pi}RR'\sin(\alpha+\beta)(1-R^2/32+R^4/1536). \quad (2.103)$$

Finally, we wish to consider the integration of $J_3(\theta)$. When ϵ is small, the denominator of $J_3(\theta)$ is approximate to 1 and the numerator contains the quadratic terms of ϵ. As we only consider the linear terms involving ϵ, we ignore the integral $\int_0^{2\pi} J_3(\theta)\mathrm{d}\theta$. Hence, our approximation of the median becomes

$$M=\pi\left\{(\nu_{11}-\mu_1)^2+(\nu_{12}-\mu_2)^2\right\}+KR'\sin(\alpha+\beta)\epsilon, \quad (2.104)$$

with $K=2\sqrt{\pi}R(1-R^2/32+R^4/1536)$.

Now, we wish to find μ_1 and μ_2 in terms of R (i.e. the distance between the outlier and the bulk) by partial differentiation, which will minimise the value of M.

$$\frac{M(\mu_1,\mu_2)}{\partial\mu_1}=-2\pi(\nu_{11}-\mu_1)+K\epsilon\left\{\frac{\partial R'}{\partial\mu_1}\sin(\alpha+\beta)+\frac{\partial\sin(\alpha+\beta)}{\partial\mu_1}R'\right\}, \quad (2.105)$$
$$\frac{M(\mu_1,\mu_2)}{\partial\mu_2}=-2\pi(\nu_{12}-\mu_2)+K\epsilon\left\{\frac{\partial R'}{\partial\mu_2}\sin(\alpha+\beta)+\frac{\partial\sin(\alpha+\beta)}{\partial\mu_2}R'\right\}. \quad (2.106)$$

After some manipulations, we have

$$\frac{M(\mu_1,\mu_2)}{\partial\mu_1} = -2\pi(\nu_{11}-\mu_1) - K\epsilon\Big\{(\nu_{11}-\mu_1)^2+(\nu_{12}-\mu_2)^2\Big\}^{-1/2} \Big\{(\nu_{11}-\mu_1)\sin(\alpha+\beta)+(\nu_{12}-\mu_2)\cos(\alpha+\beta)\Big\}, \tag{2.107}$$

$$\frac{M(\mu_1,\mu_2)}{\partial\mu_2} = -2\pi(\nu_{12}-\mu_2) - K\epsilon\Big\{(\nu_{11}-\mu_1)^2+(\nu_{12}-\mu_2)^2\Big\}^{-1/2} \Big\{(\nu_{12}-\mu_2)\sin(\alpha+\beta)-(\nu_{11}-\mu_1)\cos(\alpha+\beta)\Big\}. \tag{2.108}$$

Then, we set the partial derivatives of Equation (2.107) and Equation (2.108) to be 0 and solve them. Hence, we obtain the optimal values μ_1^* and μ_2^*, which minimise Equation (2.104), as follows:

$$\mu_1^* = \nu_{11} + \pi^{-1/2}R\epsilon(1-R^2/32+R^4/1536)\cos\alpha, \tag{2.109}$$

$$\mu_2^* = \nu_{12} + \pi^{-1/2}R\epsilon(1-R^2/32+R^4/1536)\sin\alpha, \tag{2.110}$$

Finally, we plug the μ_1^* and μ_2^* into Equation (2.104), and obtain the minimum value of M, that is

$$M^* = R^2\epsilon^2(1-R^2/32+R^4/1536)\{2\sin(\alpha+\beta)+1\}, \tag{2.111}$$

with $R < \sqrt{2}\pi$.

□

Intuitively, the approximation in the Proposition works whenever the two cluster means are close enough together, i.e. when $R^2 = \delta_1^2+\delta_2^2 < 2\pi^2$.

In particular, when $\nu_{11} = \nu_{21}$ or $\nu_{12} = \nu_{22}$ (i.e. when one of the $\delta_i = 0, i = 1,2$), we can form a more accurate approximation. This is because the approximation

for the standard normal distribution function, $\phi(z) \approx (1+\cos z)/2\pi$, is no longer required to find the optimal $\boldsymbol{\mu}^* = (\mu_1^*, \mu_2^*)^T$ minimising Equation (2.56). Without loss of generality, let $\boldsymbol{\nu}_1 = (\nu_{11}, \nu_{12})^T = (0,0)^T$ and $\boldsymbol{\nu}_2 = (\nu_{21}, \nu_{22})^T = (0,d)^T$, we obtain the yamm estimator as follows

$$\begin{aligned} \mu_1^* &= 0, \\ \mu_2^* &= 2^{-1/2}\epsilon d\, e^{-\frac{d^2}{8}}\Big(\text{BesselI}\big[0, d^2/8\big] + \text{BesselI}\big[1, d^2/8\big]\Big), \end{aligned} \tag{2.112}$$

where BesselI$[n,z]$ is the modified Bessel function of the first kind, sometimes denoted $I_n(z)$.

Proof. According to Equation (2.85), since $\delta_1 = 0$, $\delta_2 = d$, we have

$$2\int_0^\pi J_2(\theta)\,\mathrm{d}\theta \approx 4\epsilon\sqrt{2\pi}\int_0^\pi \Phi(-d\sin\theta/\sqrt{2})(-\mu_1\cos\theta + \mu_2\sin\theta)\mathrm{d}\theta. \tag{2.113}$$

Integration by part, we have

$$\begin{aligned} 2\int_0^\pi J_2(\theta)\mathrm{d}\theta \approx 4\epsilon\sqrt{2\pi}\Big\{&\Big[\Phi(-d\sin\theta/\sqrt{2})(-\mu_1\sin\theta - \mu_2\cos\theta)\Big]_0^\pi \\ &- \int_0^\pi \frac{d}{\sqrt{2}}\phi(-d\sin\theta/\sqrt{2})(\mu_1\sin\theta + \mu_2\cos\theta)\cos\theta\,\mathrm{d}\theta\Big\} \qquad (2.114) \\ = 4\epsilon\sqrt{2\pi}&\Big\{\mu_2 - \frac{d}{\sqrt{2}}\,\mu_2\int_0^\pi \phi(-d\sin\theta/\sqrt{2})\cos^2\theta\,\mathrm{d}\theta\Big\} \qquad (2.115) \\ = 4\epsilon\sqrt{2\pi}&\Big\{\mu_2 - \frac{d\sqrt{\pi}}{4}\,\mu_2\, e^{-\frac{d^2}{8}}\Big(\text{BesselI}\big[0, \frac{d^2}{8}\big] + \text{BesselI}\big[1, \frac{d^2}{8}\big]\Big)\Big\}. \qquad (2.116) \end{aligned}$$

Now, we try to compute $2\epsilon\sqrt{2\pi}\int_0^\pi J_{2,1}(\theta)\mathrm{d}\theta$ in the second part of Equation (2.79):

$$\begin{aligned} 2\epsilon\sqrt{2\pi}\int_0^\pi J_{2,1}(\theta)\mathrm{d}\theta &\approx -2\epsilon\sqrt{2\pi}\Big\{\int_0^\pi (-\mu_1\cos\theta + \mu_2\sin\theta)\,\mathrm{d}\theta\Big\} \qquad (2.117) \\ &= 4\epsilon\sqrt{2\pi}\mu_2. \qquad (2.118) \end{aligned}$$

Hence,

$$\int_0^{2\pi} J_2(\theta)\,\mathrm{d}\theta \tag{2.119}$$

$$=4\epsilon\sqrt{2\pi}\left\{\mu_2-\frac{d\sqrt{\pi}}{4}\,\mu_2\,e^{-\frac{d^2}{8}}\left(\text{BesselI}[0,\frac{d^2}{8}]+\text{BesselI}[1,\frac{d^2}{8}]\right)\right\}-4\epsilon\sqrt{2\pi}\mu_2 \tag{2.120}$$

$$=-\sqrt{2}d\pi\epsilon\,\mu_2\,e^{-\frac{d^2}{8}}\left(\text{BesselI}[0,\frac{d^2}{8}]+\text{BesselI}[1,\frac{d^2}{8}]\right). \tag{2.121}$$

When $(\nu_{11},\nu_{12})^T=(0,0)^T$ and $(\nu_{21},\nu_{22})^T=(0,d)^T$, we have $J_1(\theta)=(\mu_1^2+\mu_2^2)\pi$. Ignoring the term $J_3(\theta)$, our approximation of median M is

$$M\approx J_1(\theta)+J_2(\theta) \tag{2.122}$$

$$\approx(\mu_1^2+\mu_2^2)\pi-\sqrt{2}d\pi\epsilon\,\mu_2\,e^{-\frac{d^2}{8}}\left(\text{BesselI}[0,\frac{d^2}{8}]+\text{BesselI}[1,\frac{d^2}{8}]\right). \tag{2.123}$$

To find μ_1^* and μ_2^*, we then compute

$$\frac{M(\mu_1,\mu_2)}{\partial\mu_1}=2\pi\mu_1, \tag{2.124}$$

$$\frac{M(\mu_1,\mu_2)}{\partial\mu_2}=2\pi\mu_2-\sqrt{2}d\pi\epsilon\,e^{-\frac{d^2}{8}}\left(\text{BesselI}[0,\frac{d^2}{8}]+\text{BesselI}[1,\frac{d^2}{8}]\right). \tag{2.125}$$

Setting $\frac{M(\mu_1,\mu_2)}{\partial\mu_1}=\frac{M(\mu_1,\mu_2)}{\partial\mu_2}=0$, we have

$$\mu_1^*=0 \tag{2.126}$$

$$\mu_2^*=\frac{\epsilon}{\sqrt{2}}\,d\,e^{-\frac{d^2}{8}}\left(\text{BesselI}[0,\frac{d^2}{8}]+\text{BesselI}[1,\frac{d^2}{8}]\right) \tag{2.127}$$

□

2.3.3.4 The Yamm influence curve on the mixture

This section numerically computes and plots yamm for the case where $\epsilon = 0.05$, $\mathbf{X}_1 \sim N(\boldsymbol{\nu}_1, I_2)$ and $\mathbf{X}_2 \sim N(\boldsymbol{\nu}_2, I_2)$, with $\boldsymbol{\nu}_1 = (0,0)^T$ and $\boldsymbol{\nu}_2 = (0,d)^T$ for $d \in \mathbb{R}$. We explore how yamm varies as d increases from 0 to 10 in steps of 0.2. If yamm is robust, then it should increase with d, but plateau beyond a certain point.

For each value d we estimate yamm as the mean over five hundred bivariate mixture realizations, with two thousand projections involved for each yamm computation, using methods described below in Section 2.3.4. The numerically computed crosses in Figure 2.2 show that, for this setup, yamm plateaus somewhere between $d = 2$ and $d = 4$.

The solid red line in Figure 2.2 shows our theoretical approximation of the yamm influence curve with the more specific setup, where $\boldsymbol{\mu}^*$ follows Equation (2.112). Under this approximation, the influence curve closely follows the numerically computed crosses. On the other hand, the solid blue line is the approximation of the yamm under the more general setting of Equation (2.57), which exhibits poor approximation after $d > 4.5$, although it performs reasonably well when the inter-cluster mean distance $0 < d < 4.5$, and does not plateau.

This is because, in the setup, $\delta_1 = d, \delta_2 = 0$, and $d > 4.5$ implies $R^2 = \delta_1^2 = d^2 > 2\pi^2$. However, the specific setup approximation of yamm obviously does not work for arbitrary values of $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$, whereas the general approximation

gives a good theoretical idea of the yamm influence curve when the two means of the clusters are close enough together.

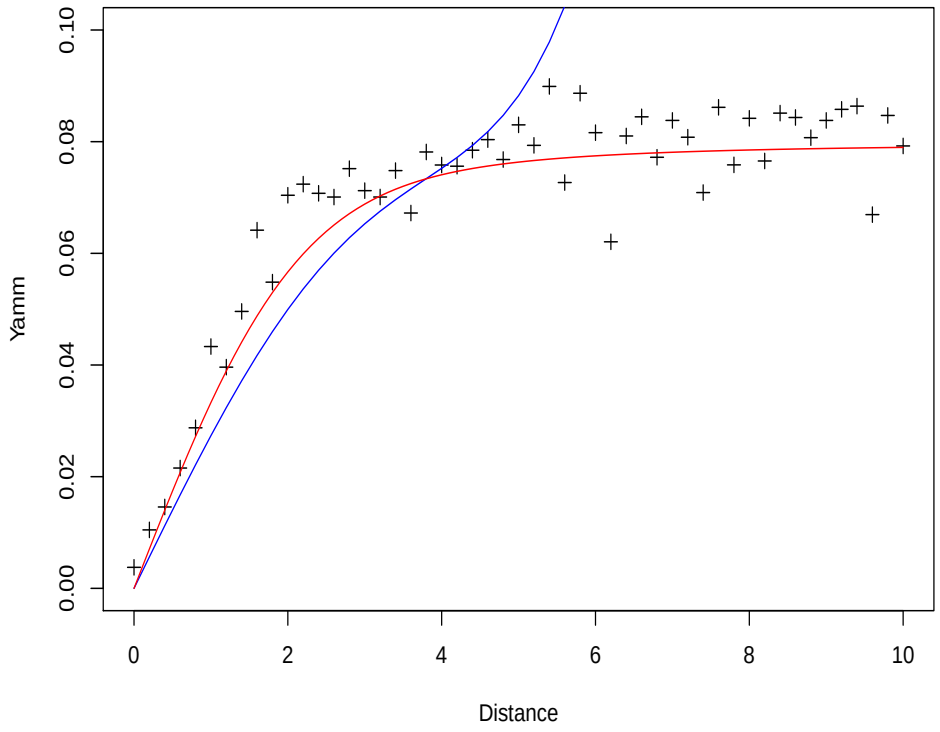


Figure 2.2: Yamm computed on simulated setup, increasing the distance between two bivariate normals. Crosses: numerically computed values; Solid blue line: approximation computed for general $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$; Solid red line: approximation computed when $\boldsymbol{\nu}_1 = (0,0)^T$ and $\boldsymbol{\nu}_2 = (0,d)^T$.

2.3.4 Projection median and Yamm computation

2.3.4.1 Projection median computation

A simple Monte Carlo integration (Robert and Casella, 2005) can be used to compute an approximation of the projection median by

$$\hat{M}_P(\mathbf{X}) = nJ^{-1}\sum_{j=1}^{J}\text{med}(\mathbf{X}_{\mathbf{a}_j}), \tag{2.128}$$

where J represents the number of projections used, and $\{\mathbf{a}_j\}_{j=1}^{J}$ is a set of random, independently-drawn, unit length n-vectors over $\mathbf{X}^{n-1}$.

Calculating approximation (2.128) is relatively straightforward, but a large value of J is required to ensure accuracy. Another approach computes the projection median directly from the definition in (2.12), using the spherical coordinates illustrated in (2.13), where the integral can be obtained by the trapezoidal rule. For example, in the two-dimensional case, we apply the trapezoidal rule once on (2.11). In the three-dimensional case, we have to apply the trapezoidal rule twice for the double integral, and so on. This direct approach is easy to implement when our dataset has a low dimension, but excessive work is required in not that many higher dimensions, even with, e.g. $n = 10$.

2.3.4.2 Computing Yamm

To compute an approximation to yamm, we can also use Monte Carlo integration together with an optimiser. Let $J \in \mathbb{N}$ be the number of projections, $\{\mathbf{a}_j\}_{j=1}^{J}$ be a set of independent random unit length n-vectors, an estimator for $M_{\mathbf{X},m}(\boldsymbol{\mu})$

is given by

$$\hat{M}_{\mathbf{X},m}(\boldsymbol{\mu}) = J^{-1} \sum_{j=1}^{J} m_{\mathbf{X}}(\boldsymbol{\mu}, \mathbf{a}_j)^2. \tag{2.129}$$

We then numerically minimise $\hat{M}_{\mathbf{X},m}(\boldsymbol{\mu})$ over $\boldsymbol{\mu}$ to obtain our estimated location measure, using the BFGS optimization method (Broyden; Fletcher; Goldfarb; Shanno, 1970). BFGS is a quasi-Newton algorithm searching for a stationary point of a function via local quadratic approximation. Parallel versions such as `optimParallel` (Gerber and Furrer, 2019) exist as easy to use packages in `R`.

After extensive simulation, we find that with reasonable starting values, such as the mean or other multivariate medians, yamm provides accurate results with a considerably smaller number of projections than used by the Monte Carlo projection median method.

In conclusion, projection median computation via the trapezoidal rule is fast and accurate in low dimensions, but increasingly onerous in higher dimensions, as progressively more multidimensional integration is required. For higher dimensions, we prefer the Monte Carlo method and prefer yamm over the projection median as it does not require such a large number of projections, particularly if the optimiser is given a good starting solution.

Overall, approximating the projection median by the trapezoidal rule is a good choice in $\mathbb{R}^2$ and $\mathbb{R}^3$, and either of the other two methods can be used in higher dimensions.

2.4 Empirical Performance for Different Medians

This section reviews the theoretical computational complexity for a variety of medians and computes some running times for real implementations of several medians computed in R. We then present some results for accuracy of estimation for these medians.

2.4.1 Computational complexity and empirical speed

For a dataset in $\mathbb{R}^n$ with k observations, the computational complexity for the Spatial median is $O(nk)$ (Bose et al., 2003), which is the same for the exact computation of the component-wise median. The projection median can be obtained in $O(k^{4/3}\log^{1+\epsilon}k)$ time in $\mathbb{R}^2$ (Durocher and Kirkpatrick, 2005), and $O(k^{5/2+\epsilon})$ time in $\mathbb{R}^3$ (Basu et al., 2012). In $\mathbb{R}^n$, with $n>3$, Basu et al. (2012) showed that $O[k^{n\{1-\delta_n/(n+1)\}+\epsilon}]$ time is required to compute the projection median, where $\delta_n=(4n-3)^{-n}$ and ϵ is a fixed small constant. Several algorithms for other multivariate medians have been developed or the bivariate case. The current best algorithms for Oja's and Liu's medians require $O(k\log^3 k)$ and $O(k^4)$ time, respectively (Aloupis et al., 2003), whereas that for the fastest bivariate Tukey median is $O(k\log^3 k)$ (Langerman and Steiger, 2003). The calculation of these three multivariate medians in higher dimensions is more complicated and approximate computation is often preferred/required.

To provide empirical assessment of the real computation speed, we apply sev-

eral R software medians to simulated data. There are several R functions using different algorithms to compute one median. For example, `spatial.median` from the library `ICSNP` (Nordhausen et al., 2018) estimates the median with the algorithm developed by Vardi and Zhang (2000), while `Gmedian` developed by Cardot et al. (2013) is faster but, perhaps, less accurate. In addition, `l1median` (Croux et al., 2006) from library `pcaPP` (Filzmoser et al., 2021) and `med` from `depth` (Genest et al., 2019) also provide opportunities to compute the spatial median. Hence, after some experiments, we choose the best function (evaluated in terms of speed and accuracy) for each multivariate median in $\mathbb{R}^2$ and $\mathbb{R}^3$ shown in Table 2.1. Much of the software for multivariate medians in R only works in low numbers of dimensions.

Median	Function	Package	Source
Spatial	`l1median`	`pcaPP`	Croux et al. (2006)
CWmed	`med`	`depth`	—
Liu's	`med`	`depth`	Rousseeuw and Ruts (1996)
Tukey's	`med`	`depth`	Rousseeuw et al. (1999) Struyf and Rousseeuw (2000)
Oja's	`ojaMedianEvo`	`OjaNP`	Fischer et al. (2016)
Projection	`PmedTrapz`	`Yamm`	Ours

Table 2.1: R functions used for analysing different multivariate medians.

The `med` function can only calculate the bivariate Liu's median, which is considerably more challenging in higher dimensions. The calculation of Tukey's median is exact in one and two dimensions, and approximate in higher dimensions. We use the approximate Tukey's median computation in the `med` function, due to numerical errors that sometimes surface when using the exact algorithm. For Oja's median, the approximate method (evolutionary algorithm)

is used instead of the exact one, as it is faster and can deal with high dimensions.

Table 2.2 displays mean computation times and their standard deviations across 1000 simulated datasets from the two-dimensional Laplace distribution with different numbers of observations (k) for each set. The results are produced by running R on a single core of an Intel i7-8750h processor with 2.20 GHz base clock using 16Gb RAM. For small k, Liu's median is fastest, but its speed is not as fast as others for higher k. In this experiment, Oja's median is the slowest for small k values, but its speed does not appear to be particularly sensitive to k. Hence, its speed is faster than Tukey's median when $k = 200$. The projection median is one of the quickest when k is below 100, while for large k values, the component-wise median and the Spatial median are faster.

The results in Table 2.2 are produced by only one possible R function for one median. However, other functions can be used. For example, the `med` function from the `depth` package can also be used to calculate the spatial median and provides accurate answers. It is extremely fast for small k and lower dimensions, but it becomes slower than `l1median` for larger k. Hence, we use `l1median` to compute the spatial median, whose performance for small k is also good.

Median		$k=10$	$k=25$	$k=50$	$k=100$	$k=200$
Spatial	mean	27	28	30	29	28
	s.d.	44	45	57	45	45
Component-wise	mean	24	21	25	25	24
	s.d.	42	41	43	43	43
Liu's	mean	3	6	14	49	190
	s.d.	18	24	35	66	250
Tukey's	mean	67	210	510	970	1890
	s.d.	47	28	40	56	100
Oja's	mean	1430	1400	1460	1410	1410
	s.d.	410	190	270	190	160
Projection	mean	7	12	18	31	60
	s.d.	26	32	39	46	49

Table 2.2: Mean and standard deviation (s.d.) of the operation time ($\times 10^{-5}$) in seconds for data in $\mathbb{R}^2$.

2.4.2 Mean squared error for some medians

We assess the accuracy of some of the medians via empirical mean squared error. If $\hat{\mathbf{X}}$ is an estimator in $\mathbb{R}^n$ with respect to the unknown parameter $\boldsymbol{\mu} \in \mathbb{R}^n$, then the mean squared error is

$$\text{MSE}(\hat{\mathbf{X}}) = n^{-1} E(||\hat{\mathbf{X}} - \boldsymbol{\mu}||_2^2), \tag{2.130}$$

where $n^{-1}||\hat{\mathbf{X}} - \boldsymbol{\mu}||_2^2$ represents the squared Euclidean distance between $\hat{\mathbf{X}}$ and $\boldsymbol{\mu}$, normalized by the vector length. Smaller $\text{MSE}(\hat{\mathbf{X}})$ values are better.

Table 2.3 shows mean squared error results based on the same simulations as used for Table 2.2. Not surprisingly, for this long-tailed data, all medians perform better than the sample mean. The spatial median and the projection median have smaller mean squared error, the latter performing better for small

k values. On the other hand, Liu's median always produces a very high mean squared error.

Location Estimator	$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial	67	21	9.7	4.6	2.3
Component-wise	74	26	12.0	5.7	2.9
Liu's	110	31	14.0	6.3	3.2
Tukey's	73	21	10.0	4.8	2.3
Oja's	75	22	11.0	5.6	3.2
Projection	66	21	9.8	4.7	2.3
Mean	110	39	20.0	9.9	5.0

Table 2.3: Mean squared error ($\times 10^{-2}$) for data as in Table 2.2.

Based on these simulations, for the `R` functions listed in Table 2.1, the spatial and projection medians always have the lowest mean squared error, but also fast running speeds. Although Liu's median has the shortest computation time, for small k, it is the most inaccurate, and its computation time becomes long for large datasets. Similarly, the component-wise median is fast, even when k increases, but it has a large mean squared error. Hence, the spatial and projection medians are good choices when computing two-dimensional robust measures of location in this case, and the latter is preferred for small datasets. The computational results for high-dimensional simulations ($n = 3, 5, 10$) can be found in the appendix A.1.

2.4.3 2D projection median computation functions

The R package `DurocherProjectionMedian` can be downloaded from Github at

`https://github.com/12ramsake/DurocherProjectionMedian`

The `DurocherProjectionMedian` package provides functions to compute the projection median via the Monte Carlo integration method as described in Durocher et al. (2017) using `projectionMedianMC`) and an exact method for two dimensions proposed by Ramsay (2017) using `projectionMedian2D`. Tables 2.4 and 2.5 show the performance of the different functions computing the two-dimensional projection median of 1000 simulated datasets from the Laplace distribution with different k.

R Function		k				
		10	25	50	100	200
`PmedTrapz`	mean	7	12	18	31	60
	s.d.	26	32	39	46	49
`projectionMedian2D`	mean	320	1020	3930	11640	44830
	s.d.	50	99	420	970	2690
`PmedMCInt`	mean	250	320	490	870	1670
	s.d.	40	39	33	50	58
`projectionMedianMC`	mean	930	970	1010	1130	1280
	s.d.	49	57	65	60	55

Table 2.4: Mean and standard deviation (s.d.) of the operation time ($\times 10^{-5}$) in seconds for different R functions to produce the projection median.

R Function	k				
	10	25	50	100	200
`PmedTrapz`	656	207	98.2	47.1	23.2
`projectionMedian2D`	656	206	97.4	46.9	22.9
`PmedMCInt`	659	205	97.8	47.0	23.0
`projectionMedianMC`	659	205	97.6	47.0	23.0

Table 2.5: Mean squared error ($\times 10^{-3}$) for 1000 sets of data in $\mathbb{R}^2$ generated from Laplace distribution.

For the Monte Carlo Integration method, when k is small (e.g. under 150 in $\mathbb{R}^2$), the computation time of `projectionMedianMC` is longer than our `PmedMCInt` under the same number of projections in both $\mathbb{R}^2$ and high dimensions, whereas both implementations have almost the same MSE.

Although the `projectionMedian2D` provides a slightly smaller MSE, its running time is slow. Our `PmedTrapz` is faster and its MSE performance is comparable to `projectionMedian2D`, and, hence, the former might be recommended as the best choice for $\mathbb{R}^2$.

2.5 The Yamm R Package

Our `Yamm` R package (Chen and Nason, 2020a) provides users with functions to compute the projection median according to the different methods mentioned in section 2.3.4. `PmedMCInt` computes the projection median using the Monte Carlo approximation; `PmedTrapz` uses the trapezoidal rule and currently, it is only valid in two and three dimensions; `yamm` computes the projection median using the Monte Carlo approximation to find the shift vector $\boldsymbol{\mu}$ minimising our objective function `yamm.obj`. The package also includes functions `Plot2dMedian` and `Plot2dMedian` to plot different multivariate medians for data in both $\mathbb{R}^2$ and $\mathbb{R}^3$. Most functions in our package are implemented internally using C code. This section provides some brief illustrations of the use of `Yamm`.

2.5.1 Yamm projection medians

The function `PmedMCInt` computes the projection median for any multivariate data, `x`, by invoking

```
PmedMCInt(x, nprojs = 20000)
```

Since this function uses Monte Carlo integration, we need to choose the number of projections J, which has a default value of 20000. Typically, a large J is required to obtain a stable answer, which means the result will not change much if recomputed under the same conditions. This function returns the projection median estimate vector.

The function `PmedTrapz` computes the projection median in $\mathbb{R}^2$ and $\mathbb{R}^3$ and is invoked by

```
PmedTrapz(x, no.subinterval)
```

`PmedTrapz` applies the trapezoidal rule once in $\mathbb{R}^2$ and twice in $\mathbb{R}^3$ on each entry of the vector med($\mathbf{X_a}$), mentioned in section 2.3.1.2, and returns a vector of the projection median estimate.

The argument `no.subinterval` determines the number of subintervals for the trapezoidal rule. For the bivariate case the `no.subinterval` argument is a single number that controls the number of subdivisions for the one-dimensional integration; for the trivariate case the argument is a vector of length two that controls the number of subdivisions for the two integrals. In general, it is better to use at least 36 subintervals, which typically produces accurate results

without excessive running time.

More subintervals may be appropriate for more complex datasets. For some unusual data sets it would be ideal to have a high resolution of the interval of integration in one particular region, and a relatively low resolution elsewhere, but this is beyond the scope of the current research. A small number of partitions, e.g. below 15, is not recommended for reasons of accuracy.

The `yamm` is valid for data of any dimension. It uses an optimiser to provide another method to compute the projection median. The arguments are

```
yamm(x, nprojs = 2000, reltol = 1e-06,
        xstart = l1median(x), opt.method = "BFGS",
        doabs = 0, full.results = FALSE).
```

The `yamm` function is a wrapper to minimise the the objective function `yamm.obj`, which uses the Monte Carlo method to approximate the squared or absolute value of the univariate median of the projection of the shifted data matrix. The `nprojs` argument controls the number of projections in the Monte Carlo approximation and `doabs` is an indicator, where 1 uses the absolute value of the univariate median and 0 forces the use of the squared value. The arguments `reltol`, `xstart`, `opt.method` are supplied directly to the R optimisation function `optim`: `reltol` is the tolerance for the optimiser, with default value of 10^{-6}. Usually, we set a larger value (e.g. 10^{-3}) to this argument, which will reduce the running time, whilst maintaining accuracy. The argument `opt.method` controls the selection of optimisation methods, which can be cho-

sen from any of the four options, "BFGS", "Nelder-Mead" (Nelder and Mead, 1965), "CG" (Fletcher and Reeves, 1964), "L-BFGS-B" (Nocedal and Wright, 1999), and "SANN" (Byrd et al., 1995). The default choice "BFGS" is relatively fast and stable in our case. See the help page of the function `optim` in R for further details about the different optimisation methods. The `xstart` argument provides the initial value for the parameters to optimise over, which plays an important role in the function `yamm`. A good starting point will reduce the running time and provide a more accurate result, so we use the spatial median as the default value. Other multivariate medians could be used, but they need to be fast. If `full.results=TRUE`, the output of this function involves a list with components obtained from the `optim` function, otherwise, it returns a vector containing the multivariate median estimate.

2.5.2 Some real examples

We now exhibit results for the projection medians applied to some real datasets. Our plots show different multivariate medians and the sample mean value for two simulated datasets in $\mathbb{R}^2$ and $\mathbb{R}^3$, respectively, allowing the methods to be compared.

2.5.2.1 Beetle data

The famous `beetle` data (Lubischew, 1962) takes six measurements on 74 flea-beetles, with each belonging to one of three different species. We apply `yamm` and obtain the following output:

```
yamm(beetle, nprojs=1000, reltol=1e-3, doabs=0,
     full.results=TRUE)

[1] 180.19194 123.73920  49.97819 135.87913  13.62603  95.49062

$value
[1] 5.585139

$counts
function gradient
     90        4

$convergence
[1] 0

$message
NULL
```

The yamm results show that the optimiser executed 90 calls to the objective function yamm.obj and constructed 4 gradients. The par component contains the estimate of the yamm for the beetle data. These results are not that different from the output generated by PmedMCInt, which is

```
PmedMCInt(beetle, nprojs=100000)

[1] 179.54428 124.72128  50.56934 137.47363  13.23372  94.80188
```

For the beetle data, we chose the number of projections in `yamm` to be 1000, while many more projections were required (e.g. 100000) in `PmedMCInt` to obtain a similar and consistent result; although `yamm` requires optimisation. Fewer projections for the function `PmedMCInt` may lead inaccurate results for some components of the multivariate median. `PmedTrapz` is not valid in this six-dimensional case, but we will show that it has a similar output when computing projection median in two- and three-dimensions.

2.5.2.2 Simulated data in $\mathbb{R}^2$ with three clusters

We now use the function `Plot2dMedian` in the package `Yamm` to generate and display different multivariate medians for the simulated data set `clusters2d`. This set contains three clusters, which are generated randomly from different independent normal distributions, and two outliers.

Here, we display the three different estimates of the projection median. When computing other multivariate medians, we use functions from R packages listed in section 2.4.1. The actual data points is plotted with grey dots. The first plot in Figure 2.3 is producing excluding the two outliers, whilst the second one includes them. The projection medians produced with different estimators are very close to each other, and not far from the other median estimators also. Figure 2.3 also shows that the multivariate medians are not particularly affect by the outliers, whilst the mean value is.

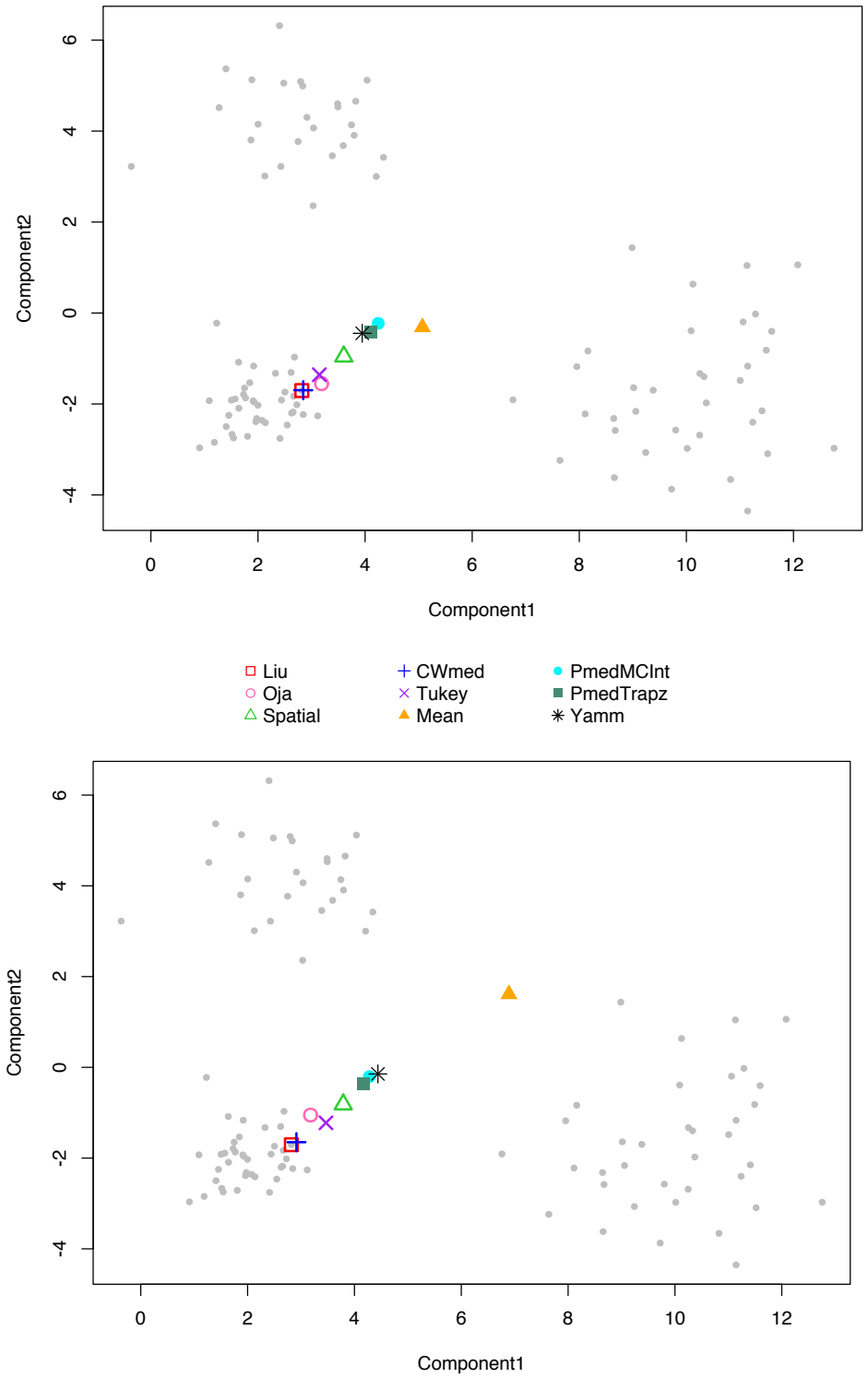


Figure 2.3: Bivariate medians and mean for three cluster two-dimensional set. Top: without outliers; Bottom: with outliers (out of plot area).

2.5.2.3 Simulated data in $\mathbb{R}^3$ with four clusters

The function `Plot3dMedian` in `Yamm` plots the three-dimensional medians. The dataset `clusters3d` has four clusters, each generated from different independent normal distributions, as well as five outliers. Figure 2.4 is produced with the dataset `clusters3d`, whose outliers have been removed. It shows that apart from the Oja's median, the other medians are located close to each other. Again, the three approximations of the projection median almost coincide in every component.

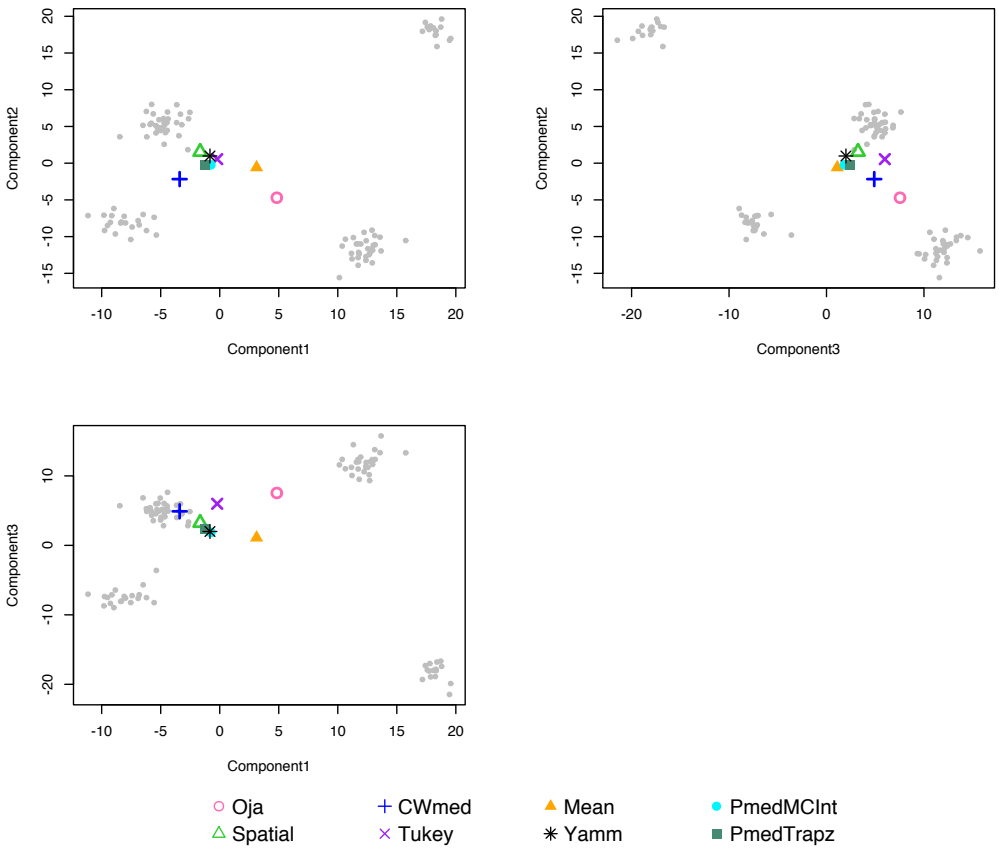


Figure 2.4: Trivariate medians & mean for four cluster three-dimensional set.

2.5.3 The Muqie plot and some examples

As well as obtaining a robust location measure, we can use projections to provide information on the spread and configuration of the data. Obtaining true multivariate quantiles can be computationally challenging, and what we produce are not true multivariate quantiles, but they do enable us to gain useful understanding about multivariate data. The muqie (MUltivariate QuantIlE) plots are constructed as follows.

First choose a unit-length direction vector, u. Then project our yamm-centred multivariate data onto u to obtain a univariate set. The muqie point, $Q(\alpha, u)$, is merely the vector u rescaled to have length equal to the α-quantile of the univariate set. A muqie plot is the collection of all muqie points, $Q(\alpha, u)$ over all unit-length direction vectors u. In practice, we construct our plot by choosing a number of directions and joining the points. The basic concept, and plots, are not new, Section 2 of Fraiman and Pateiro-Lopez (2012) introduces the concept based on mean-centred data and is related to ideas in Kong and Mizera (2012). Our main addition to this body of work is to (i) centre using yamm, or other robust median and (ii) presenting the muqie plots as dynamic videos of increasing α.

Figure 2.5 shows two muqie plots for $\alpha = 0.4$ and $\alpha = 0.8$. The latter indicates the three cluster nature. Surprisingly, this also shows up clearly in the $\alpha = 0.4$ plot with the 0.4 quantile for, e.g. the bottom-left cluster appearing in a "north-easterly" direction and coloured red in our plot. The movie `Animation` shows an animated plot, which includes both the plots in Figure 2.5 and many

of the others for increasing values of α.

These plots were produced by the `muqie()` function in the `Yamm` package. For the animated plot, the package includes the `makeplot()` function, which calls `muqie()` for multiple values of α. Then we use the CRAN package `animation` to produce an animated GIF using

```
saveGIF(makeplot(clusters2d[,-c(102,103)], nprojs=4000),
    diff.col=3, interval=0.1, width=500, height=500).
```

The movie can be found at

`https://doi.org/10.1371/journal.pone.0229845.s005`

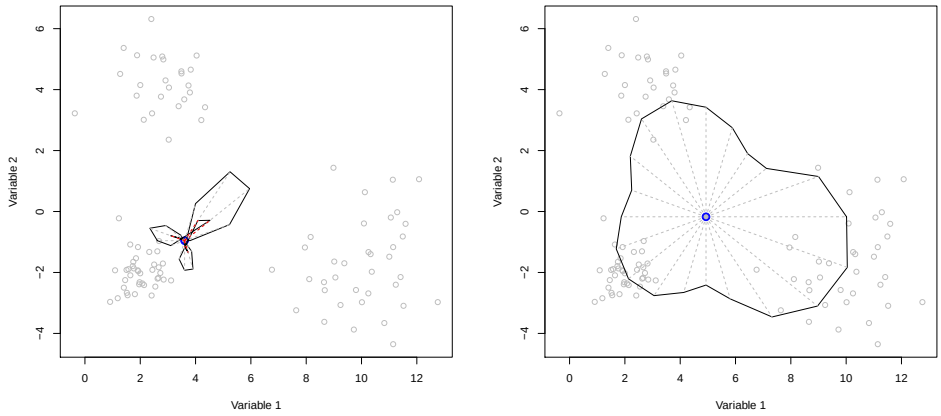


Figure 2.5: Muqie plot for the three cluster two-dimensional data set without outliers for different values of pseudo-quantile α. The centre point (in blue) in each plot is the yamm median. Left: $\alpha = 0.4$, Right: $\alpha = 0.8$.

The movie `beetle` shows a three-dimensional Muqie plot using three variables from the beetle data. The R commands used were:

```
saveGIF(makeplot3D(beetle, dm=c(1,3,6)), diff.col=3,
    interval=0.2, width=500, height=500).
```

The corresponding animated GIF is at

https://doi.org/10.1371/journal.pone.0229845.s006

2.6 Conclusions

We have introduced a new method, yamm, to compute the projection median, for data in $\mathbb{R}^n$ with $n \geq 2$. We have proved the theoretical equivalence of yamm and the projection median. Through theoretical and numerical investigations we demonstrate the robustness of yamm on a simple, but illuminating, bivariate setup.

Then, we illustrated three computation methods for the projection median, which can be best deployed in different situations. Approximating the projection median by the Monte Carlo method is valid in any dimensions but requires a large number of projections to ensure accuracy, while using the trapezoidal rule is computationally fast and accurate in two and three dimensions, but requires more integration on the projection vector in the higher dimensions, which becomes rapidly more complex. The yamm approximation can also compute the median in any dimensions. Its computational speed is not as quick as the other two, under the same conditions (e.g. the number of projections). However, thanks to the optimiser, a small number of the projections can be chosen to obtain an accurate median with a reasonable starting point (e.g. other multivartiate medians or mean value), which can be a distinct advantage.

Our research also documents the simulated empirical performance for different medians in terms of the computation time and the mean squared error. Using different R functions to calculate different multivariate medians, we find that the spatial median and the projection median are always accurate with relatively fast speed using the existing R functions. The performance of other multivariate medians either exhibits slow speed or large mean squared error.

Finally, we introduce our R package, `Yamm`, that contains our three methods to compute the projection median. We show that our methods coincide with each other in $\mathbb{R}^2$ and $\mathbb{R}^3$, and all multivariate medians are not affected by the outliers in the dataset, but the location of the mean value varies a lot. Currently, the function `PmedTrapz` in the R package is only valid in $\mathbb{R}^2$ and $\mathbb{R}^3$, further investment can be conducted on extending this function to higher dimensions.

The `Yamm` package also introduces our Muqie plots, which are capable of producing animated plots of two- and three-dimensional sets' projected quantiles. The animated "growth" of these “quantile” plots give a vivid picture of the extent, spread and configuration of data in the sets.

CHAPTER 3

DENSITY AND HAZARD RATE ESTIMATION USING A BAYESIAN WAVELET APPROACH

3.1 Introduction

Wavelet methods are useful statistical techniques that have been applied in density and hazard rate function estimation. One of the earliest papers proposed by Antoniadis et al. (1999), estimates the hazard rate for right-censored data via linear wavelet smoothers. Based on this article, we try to improve the accuracy of hazard rate estimation for right-censored data displayed in section 3.3. Section 3.3.2 demonstrates the method to estimate the density function, where we use the Bayesian threshold method proposed by Silverman and Johnstone (2005b), which uses a mixture prior of a point mass at zero with a heavy-tailed distribution. Meanwhile, the survival function estimator is computed using a non-parametric Bayesian method with Dirichlet process prior shown in section 3.3.3. In addition, we implement a bootstrap aggregating approach to the density function for the right-censored data in section 3.3.2.1 to increase its accuracy somewhat.

Herrick et al. (2001) present some non-linear, thresholded wavelet estimators, that exploits the non-stationary variance structure of the wavelet coefficients. By considering the covariance structure of empirical wavelet coefficients, in section 3.4, we propose a new multivariate version of the "mixture of Gaussians" prior distribution for the unknown "true" wavelet coefficients while using the Bayesian wavelet shrinkage method to estimate density function. Previous methods have modelled independence between coefficients, even though we know this is not true. Our method models correlations between the coefficients, to attempt to improve performance.

3.2 Literature Review

3.2.1 Survival analysis definitions

3.2.1.1 Censoring

Censored data is commonly used in survival analysis, which only provides partial information about the time to event. A survival time is said to be censored if the exact lifetime has not been observed. There are three types of censored data, described as follows.

- A survival time is *right-censored* if the censoring mechanism prematurely terminates observation of the individual, before the event has actually occurred.
- An survival time is *left-censored* if the event occurred before observation of the individual began.

- An survival time is *interval-censored* when we only know the lower and upper bounds of an interval that the event occurred.

Our survival analysis in this chapter only focus on the right-censored data. Let $X_1, X_2, ..., X_n$, denote lifetimes for the n items under investigation, and let $C_1, C_2, ..., C_n$, be the corresponding censoring times, the observed random variables Z_i and δ_i are defined as

$$Z_i = \min(X_i, C_i) \quad \text{and} \quad \delta_i = \mathbb{I}_{[X_i \leq C_i]}, \tag{3.1}$$

where $\mathbb{I}_A$ is the universal indicator function of set A such that

$$\mathbb{I}_A(x) = \begin{cases} 1\,, & \text{if } x \in A, \\ 0\,, & \text{if } x \notin A\,. \end{cases}$$

Hence, $\delta_i = 0$ indicates that the i-th item's observed time is *right-censored*.

3.2.1.2 Hazard and cumulative hazard function

Let T denote the positive random variable representing time to event. The *cumulative distribution function* is defined as

$$F(t) = P(T \leq t), \tag{3.2}$$

with probability density function $f(t) = F'(t)$, if it exists. The *survival function* is

$$S(t) = P(T > t) = 1 - F(t). \tag{3.3}$$

and the *hazard function* is

$$\lambda(t) = \lim_{\delta_t \to 0} \frac{P(t \leq T < t + \delta t | T \geq t)}{\delta t} = \frac{f(t)}{S(t)}. \tag{3.4}$$

Hence, the *cumulative hazard function* is

$$\Lambda = \int_0^t \lambda(s)\mathrm{d}s. \tag{3.5}$$

3.2.2 Wavelets and Bayesian wavelet shrinkage

Let $f(x)$ be a probability density function. Let $X_1,...,X_n$ be an independent and identically distributed sample from f. For some $M \in \mathbb{Z}$, the wavelet representation of the density function can be written as

$$f(x) \sim \sum_{k\in\mathbb{Z}} c_{Mk}\phi_{Mk}(x) + \sum_{j=M}^{\infty}\sum_{k\in\mathbb{Z}} d_{jk}\psi_{jk}(x), \tag{3.6}$$

where $\{d_{jk}\}$ and $\{c_{jk}\}$ are the wavelet and scaling coefficients respectively, such that

$$d_{jk} = \int_{-\infty}^{\infty} \psi_{jk}(x)f(x)\mathrm{d}x, \tag{3.7}$$

$$c_{jk} = \int_{-\infty}^{\infty} \phi_{jk}(x)f(x)\mathrm{d}x. \tag{3.8}$$

Let $L^2(\mathbb{R})$ be the space of square integrable functions. For some $j \in \mathbb{Z}$, we also define the space V_j as the collection of functions with detail up to some finest scale of resolution j. Hence, for larger value of j, V_j will contain functions with finer scales, so that $V_j \subset V_l$ for $l > j$. Then we have orthonormal basis functions $\{\psi_{jk}(x)\}$ in $L^2(\mathbb{R})$ and $\{\phi_{jk}(x)\}$ in V_j constructed from a "mother wavelet" $\psi(x)$ and "father wavelet" (also known as scaling function) $\phi(x)$ respectively. By dilations and translations, we have

$$\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k), \tag{3.9}$$

$$\phi_{jk}(x) = 2^{j/2}\phi(2^j x - k). \tag{3.10}$$

The simplest wavelet is probably the Haar wavelet, which is defined by

$$\psi(x) = \begin{cases} 1 & \text{if } x \in [0, 0.5), \\ -1 & \text{if } x \in [0.5, 1), \\ 0 & \text{otherwise,} \end{cases} \tag{3.11}$$

and, its associated father wavelet, which is defined by

$$\phi(x) = \begin{cases} 1 & \text{if } x \in [0, 1), \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

The corresponding orthonormal basis functions $\{\psi_{jk}(x)\}$ and $\{\phi_{jk}(x)\}$ are,

$$\psi_{jk}(x) = \begin{cases} 2^{j/2} & \text{if } 2^{-j}k \le x < 2^{-j}(k + \frac{1}{2}), \\ -2^{j/2} & \text{if } 2^{-j}(k + \frac{1}{2}) \le x < 2^{-j}(k + 1), \\ 0 & \text{otherwise,} \end{cases} \tag{3.13}$$

$$\phi_{jk}(x) = \begin{cases} 2^{j/2} & \text{if } 2^{-j}k \le x < 2^{-j}(k + 1), \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

Including the Haar wavelet basis, Daubechies (1988) characterised all orthonormal compactly supported wavelet bases and illustrates this with a family of examples, called *extremal phase wavelets*. She also introduced another family

known as the *least asymmetric wavelets* (Daubechies, 1992). In our investigation, we only consider these two families while implementing our wavelet shrinkage method.

3.2.2.1 Discrete wavelet transform

The Discrete Wavelet Transform (DWT), first proposed by Mallat (1989), decomposes a signal into a set of mutually orthogonal wavelet basis functions, which is also invertible, so that the original signal can be completely recovered from its DWT representation. Starting from the finest level, the basic idea of DWT is to apply transformations recursively on the scaling coefficients to obtain the wavelet and scaling coefficients at successive levels until the desired number of iterations is reached.

As mentioned before, $\{\phi_{1n}(x)\}$ is a basis for V_1, and $\phi(x) \in V_0$, V_0 is a subspace of V_1. Hence, there exists a low-pass filter h_n and a high-pass filter g_n satisfying

$$\psi(x) = \sum_{n\in\mathbb{Z}} g_n \phi_{1n}(x), \tag{3.15}$$

$$\phi(x) = \sum_{n\in\mathbb{Z}} h_n \phi_{1n}(x), \tag{3.16}$$

where

$$g_n = \int \psi(x)\phi_{1n}(x)\mathrm{d}x, \tag{3.17}$$

$$h_n = \int \phi(x)\phi_{1n}(x)\mathrm{d}x. \tag{3.18}$$

Together with Equations (3.9) and (3.10), for all $j,k \in \mathbb{Z}$, we can then obtain

$$\psi_{j-1,k}(x) = \sum_{n\in\mathbb{Z}} g_{n-2k}\phi_{jn}(x), \tag{3.19}$$

$$\phi_{j-1,k}(x) = \sum_{n\in\mathbb{Z}} h_{n-2k}\phi_{jn}(x). \tag{3.20}$$

Substituting Equations (3.19) and (3.20) into Equations (3.7) and (3.8) respectively, we have

$$d_{j-1,k} = \sum_{n} g_{n-2k}c_{jn}, \tag{3.21}$$

$$c_{j-1,k} = \sum_{n} h_{n-2k}c_{jn}. \tag{3.22}$$

Hence, we are able to compute the coarser-level wavelet and scaling coefficients from finer ones with the above formulae for the DWT. When implementing DWT and inverse DWT in practice, the R function `wd` and `wr` from the package `wavethresh` (Nason, 2016) can be used.

3.2.2.2 Wavelet shrinkage and thresholding

Wavelet shrinkage was introduced to the literature by Donoho and Johnstone (1994). The basic model setup is as follows. Suppose n noisy observations $\{y_i\}_{i=1,\ldots,n}$ are obtained from a function f, such that

$$y_i = f(x_i) + e_i, \tag{3.23}$$

where $x_i = i/n$, for $i = 1,\ldots,n$, and $e_i \sim N(0,\sigma^2)$ are an independent sequence. Using the noisy observations y_i, the aim is to estimate the unknown function $f(x)$, for $x \in [0,1]$. Wavelet shrinkage consists of the following three main steps:

- Transform the data $\{y_i\}$ using the *discrete wavelet transform* (DWT) proposed by Mallat (1989).

- Modify the wavelet coefficients. Here, we focus on using the *thresholding* method (Donoho and Johnstone, 1994).

- Apply the inverse transform to the modified coefficients to obtain the estimate $\hat{f}(x)$ of the unknown function $f(x)$.

Applying DWT to the model expressed in Equation (3.23), we can obtain the wavelet-transformed model of it as

$$\tilde{d}_{j,k} = d_{j,k} + \epsilon_{j,k}, \tag{3.24}$$

where $\{\tilde{d}_{j,k}\}$ and $\{d_{j,k}\}$ are the "observed" and "true" wavelet coefficients respectively, and $\{\epsilon_{j,k}\}$ are the noise wavelet coefficients, which correspond to $\{y_i\}$, $\{f(x_i)\}$ and $\{e_i\}$ respectively. Hence, the wavelet thresholding approach illustrated later will be applied on this wavelet-transformed model.

Wavelet thresholding

The thresholding idea is to form estimates $\{\hat{d}_{j,k}\}$ for the "true" wavelet coefficients $\{d_{j,k}\}$ by removing coefficients in $\{\tilde{d}_{j,k}\}$ that are smaller than some threshold. Donoho and Johnstone (1994) defined two types of thresholding functions, that are

- *hard thresholding:* $\hat{d}_{j,k} = \tilde{d}_{j,k}\, \mathbb{I}\{|\tilde{d}_{j,k}| > \xi\}$,

- *soft thresholding:* $\hat{d}_{j,k} = \text{sgn}(\tilde{d}_{j,k})(\tilde{d}_{j,k} - \xi)\, \mathbb{I}\{|\tilde{d}_{j,k}| > \tau\}$,

where $\mathbb{I}$ is an indicator function and ξ is the *threshold*.

There are many different choices for the threshold level ξ. Donoho and Johnstone (1994) proposed the universal threshold, and also suggested another SURE thresholding method (Donoho and Johnstone, 1995) based on Stein's (1981) unbiased risk estimation. Nason (1996) used a cross validation approach to choose the threshold.

Among all thresholding methods, we are interested in the Bayesian wavelet method. In a typical Bayesian wavelet shrinkage method, a prior distribution is chosen for the "true" wavelet coefficients, $\{d_{j,k}\}$. Using Bayes' theorem, with known distribution of $\{\epsilon_{j,k}\}$, the posterior distribution of $\{d_{j,k}\}$ on $\{\tilde{d}_{j,k}\}$ can then be computed, and then we can get the posterior mean or median of the wavelet coefficients. Our work, explained in section 3.3.2, based on the method proposed by Silverman and Johnstone (2004, 2005a,b). Silverman and Johnstone's work, which we call the JS model, basically uses a mixture prior of a point mass at zero with a heavy-tailed distribution, and works well and also demonstrates excellent theoretical properties.

In the JS model, the probability density function of the prior is defined as

$$f_{prior}(d_{j,k}) = w\gamma(d_{j,k}) + (1-w)\delta_0(d_{j,k}), \tag{3.25}$$

where γ represents a heavy-tailed distribution, δ_0 is a point mass (Dirac delta) at zero, and w is the mixing weight with $0 \leq w \leq 1$.

There are some conditions on the types of heavy-tailed distribution (Silverman and Johnstone, 2005b, p. 1710), where γ must be symmetric, unimodal, have tails as heavy as, or heavier than, exponential, but not heavier than the Cauchy

distribution, and satisfy a regularity condition such that, for some $\kappa \in [1,2]$,

$$y^{1-\kappa}\gamma(y)^{-1}\int_y^{\infty}\gamma(u)\mathrm{d}u \tag{3.26}$$

is bounded above and below away from zero for sufficiently large y. A popular example is the Laplace distribution, which is also used in our simulation in section 3.3.4, specified by

$$\gamma_a(d_{j,k}) = \frac{a}{2}\exp(-a|d_{j,k}|), \tag{3.27}$$

where $d_{j,k} \in \mathbb{R}$ and a is a positive scale parameter.

To choose appropriate hyper-parameters for the model, "empirical Bayes" was introduced by the JS paper, which estimated parameters directly from the data using a marginal maximum likelihood technique.

For example, let g be the density obtained by forming the convolution of the heavy-tailed density γ with the standard normal density ϕ, and given the prior in Equation (3.25) and the conditional distribution of the "observed" coefficients $\{\tilde{d}_{j,k}\}$ such that

$$\tilde{d}_{j,k}|d_{j,k} \sim N(d_{j,k},\sigma^2). \tag{3.28}$$

Then, the marginal density of the "observed" wavelet coefficients $\{\tilde{d}_{j,k}\}$ is

$$wg(\tilde{d}_{j,k}) + (1-w)\phi(\tilde{d}_{j,k}). \tag{3.29}$$

Since g, ϕ and $\{\tilde{d}_{j,k}\}$ are known at this point but the w is not, in order to estimate w, JS maximizes the marginal log-likelihood

$$l(w_j) = \sum_k \log\left\{w_j g(\tilde{d}_{j,k}) + (1-w_j)\phi(\tilde{d}_{j,k})\right\}, \tag{3.30}$$

where they estimate a separate mixing weight, w_j for each scale level. Then the estimated mixing weights are substituted back into the prior model to obtain a posterior distribution by the Bayesian procedure. Similarly, other parameters in the prior distribution could be estimated in a similar way. This thresholding approach is incorporated into the R package `EbayesThresh` (Silverman and Johnstone, 2005a), which can be executed by function `ebayesthresh.wavelet`. In section 3.3.4, we will use Bayesian wavelet shrinkage to conduct experiments and present results.

3.2.3 The basic Dirichlet process model

This section provides basic information about the Dirichlet process model discussed by Ferguson (1973). We start with the definitions of the Dirichlet distribution and the Dirichlet process, denoted by DP. A Dirichlet distribution of order $K \geq 2$ on $x_1, ..., x_K \geq 0$ with parameters $\alpha_1, ..., \alpha_K > 0$ has a probability density function defined as

$$f(x_1, ..., x_K | \alpha_1, ..., \alpha_K) = \frac{\Gamma(\alpha_1 +, ..., +\alpha_K)}{\Gamma(\alpha_1), ..., \Gamma(\alpha_K)} \prod_{k=1}^{K} x_k^{\alpha_k - 1}, \tag{3.31}$$

where Γ denotes gamma function and we have $\sum_{k=1}^{K} x_k = 1$.

The Dirichlet process is a family of stochastic processes whose realizations are probability distributions. It is widely used in Bayesian inference to describe the prior knowledge about the distribution of random variables. Let α be a positive real number, G_0 be a distribution over some probability space Θ and $A_1, ..., A_r$ be any finite measurable partitions of Θ, then a random distribution G is distributed according to a Dirichlet process with the base distribution G_0

and concentration parameter α controlling how tightly the distribution G is around G_0, written $G \sim DP(\alpha, G_0)$, if

$$\Big(G(A_1),...,G(A_r)\Big) \sim Dir\left(\alpha G_0(A_1),...,\alpha G_0(A_r)\right), \tag{3.32}$$

where Dir is a Dirichlet distribution and the vector $\Big(G(A_1),...,G(A_r)\Big)$ is random since G is random.

Let $X_1,...,X_n$ be a sample from an unknown CDF F. To estimate F from a Bayesian perspective, we can use a prior π on the set of all CDF $\mathscr{F}$ and then we compute the posterior distribution on $\mathscr{F}$ given $X_1,...,X_n$. Hence, the basic Dirichlet process model is defined when the Dirichlet process prior is used (Ferguson, 1973), which has the following form

$$\begin{aligned} x_1,...,x_n &\sim F, \\ F &\sim \pi, \\ \text{with } \pi &= DP(\alpha, G_0), \end{aligned} \tag{3.33}$$

where G_0 and α are defined same as Equation (3.32).

In Bayesian perspective, if a posterior distribution is in the same probability distribution family as the prior distribution, the prior and posterior are then called conjugate distributions, and the prior is called the conjugate prior. The Dirichlet process prior is a conjugate prior, as the posterior distribution is in the same probability distribution family as the prior probability distribution, such that

$$F|x_1,...,x_n \sim DP\left(\alpha + n, \frac{\alpha G_0 + \sum_{i=1}^{n} \delta_{x_i}}{\alpha + n}\right), \tag{3.34}$$

where $\{\delta_{x_i}\}$ is a point-mass at $\{x_i\}$. Hence, the posterior distribution $F|x_1,...,x_n$ is also a distribution of probability distributions, which is the weighted sum of the empirical distribution of the data and the base measure, with the weighting controlled by α.

As the Dirichlet process is a distribution over the space of probability distributions, thus samples from a Dirichlet process are probability distributions. The stick-breaking representation is introduced by Sethuraman (1994) to show what such samples look like.

Suppose that $F \sim DP(\alpha, G_0)$ is a random probability distribution sampled from a Dirichlet process. Then, with probability 1,

$$F = \sum_{k=1}^{\infty} w_k \delta_{\phi_k}, \;\; \phi_k \sim G_0, \tag{3.35}$$

where

$$w_k = z_k \prod_{i=1}^{k-1} (1 - z_i), \;\; z_i \sim \text{Beta}(1, \alpha). \tag{3.36}$$

Combining the above results, we can draw a sample probability distribution from the posterior $F|x_1,...,x_n$ as follows

$$F|x_1,...,x_n = \sum_{k=1}^{m} w_k \delta_{\phi_k}, \;\; \phi_k \sim \frac{\alpha G_0 + \sum_{i=1}^{n} \delta_{x_i}}{\alpha + n}, \tag{3.37}$$

where large m is usually chosen to ensure the accuracy of the estimation, and

$$w_1 = z_1 \;\text{ and }\; w_k = z_k \prod_{i=1}^{k-1} (1 - z_i), \;\text{ for } k = 2, 3, ..., m, \tag{3.38}$$

$$\text{with } \; z_i \sim \text{Beta}(1, \alpha) \;\text{ for } i = 1, 2, ..., m. \tag{3.39}$$

The Dirichlet process model can be used to estimate the survival function, which will be illustrated in section 3.3.3 in detail.

3.2.4 The presmoothed method

Presmoothed versions of the classical non-parametric estimators are motivated by dealing with the heavily-censored data sets, which becomes more frequent nowadays due to increasing lifetimes.

The main idea behind the presmoothed estimators of the functions in survival analysis, such as survival, density, hazard rate and cumulative hazard function, is that they are computed by giving mass to all the data, including the censored observations. Here, we focus on the presmoothed hazard estimator. Therefore, more information on the local behavior of the lifetime distribution is provided to increase the accuracy of estimation. Hence, presmoothed estimators have been shown to have a smaller asymptotic variance and, therefore, a better performance in terms of mean squared error when the bandwidth defined later in Equation (3.41) is suitably chosen (Lopez-de Ullibarri and Jacome, 2013).

Let K be a non-negative real-valued integrable kernel function, which satisfies the following two conditions such that

- $\int_{-\infty}^{+\infty} K(u)\,\mathrm{d}u = 1$,
- $K(u) = K(-u)$ for all values of u.

Suppose the observation at time t is not censored, the Nadaraya-Watson (NW) kernel estimator (Nadaraya, 1964; Watson, 1964) with bandwidth b_1 can be defined as

$$\hat{p}_{b_1}(t) = \frac{\sum_{i=1}^{n} K_{b_1}(t - Z_i)\delta_i}{\sum_{i=1}^{n} K_{b_1}(t - Z_i)}, \tag{3.40}$$

where $\{Z_i\}$ and $\{\delta_i\}$ are described as in Equation (3.1), and $K_{b_1}(t) = b_1^{-1}K(t/b_1)$ is a rescaled kernel.

The presmoothed version of hazard $\hat{\lambda}_{b_1,b_2}(t)$ (Cao and Lopez-de Ullibarri, 2007), with the presmoothing bandwidth b_1 for estimating $\hat{p}_{b_1}(t)$ and a second smoothing bandwidth b_2, is

$$\hat{\lambda}_{b_1,b_2}(t) = \frac{1}{n}\sum_{i=1}^{n} \frac{K_{b_2}(t - Z_i)\,\hat{p}_{b_1}(Z_i)}{1 - H_n(Z_i) + 1/n}, \tag{3.41}$$

where H_n is the empirical estimator of the distribution function of Z. The quantity $\hat{\lambda}_{b_1,b_2}(t)$ is obtained by minimising the mean integrated squared error,

$$\mathrm{MISE}(b_1, b_2) = E\Big[\int_0^{\infty} \{\hat{\lambda}_{b_1,b_2}(t) - \lambda_{b_1,b_2}(t)\}^2 \omega(t)\,\mathrm{d}t\Big], \tag{3.42}$$

where $\omega(t)$ is a non-negative weight function with user-defined support $[-\tau, \tau]$ such that $\int_{-\tau}^{\tau} \omega(t)\,\mathrm{d}t = 1$.

Since the MISE depends on the unknown function $\lambda_{b_1,b_2}(t)$, the optimal bandwidth (b_1, b_2) is obtained in practice by minimizing an approximation of the MISE. Different bandwidth selectors are provided in package `survPresmooth` (Lopez-de Ullibarri and Jacome, 2013), where the function `presmooth` specifies methods of bandwidth selection. In our experiments, plug-in bandwidth selection is used. There are some other methods of bandwidth selection as options, but we do not explore these further.

3.3 Hazard Rate Estimation

This section focuses on hazard rate estimation for right-censored data, especially when the lifetimes (time to failure) of the right-censored data are generated from a piecewise-continuous density function.

3.3.1 Model set-up

We firstly reproduce the model setup by Antoniadis et al. (1999). Assuming that:

- $X_1, X_2, ..., X_n$, are non-negative and independent and identically distributed (IID) lifetimes with common continuous cumulative distribution function (CDF) F and continuous density f,
- $C_1, C_2, ..., C_n$, are non-negative and IID censoring times with common continuous CDF G and continuous density g,
- the lifetimes and censoring times are independent,
- $(Z_1, \delta_1), (Z_2, \delta_2), ..., (Z_n, \delta_n)$ are defined same as Equation (3.1).

According to the definition of the hazard function defined in Equation (3.4), in the censored case, if $G(t) < 1$, we have

$$\lambda(t) = \frac{f(t)\{1 - G(t)\}}{S(t)\{1 - G(t)\}}. \tag{3.43}$$

Let $f^*(t) = f(t)\{1 - G(t)\}$ and $L(t) = P\{Z_i \leq t\}$, we have

$$1 - L(t) = S(t)\{1 - G(t)\}, \tag{3.44}$$

$$\lambda(t) = \frac{f^*(t)}{1 - L(t)}, L(t) < 1. \tag{3.45}$$

Suppose our estimates of $\lambda(t)$ are computed over a finite interval $[\tau_{\min}, \tau_{\max}]$ and the length of the interval is $\tau = \tau_{\max} - \tau_{\min}$. Let N be an integer and $\Delta = \tau 2^{-N}$ defines a dyadic grid (evaluation points) as

$$t_k = \tau_{\min} + k\Delta,\ k = 0, ..., K = 2^N - 1. \tag{3.46}$$

Under the definition above, the time axis is the interval $[\tau_{\max} - \tau_{\min}]$ divided into $K+1$ (i.e. 2^N) subintervals of length Δ centred on t_k. Hence, for $k = 0, ..., K$, the k^{th} subinterval $J_k = [\tau_k, \tau_{k+1}]$ with midpoint t_k is defined as

$$\begin{aligned} \tau_0 &= \tau_{\min} - \frac{\Delta}{2}, \\ \tau_k &= t_k - \frac{\Delta}{2},\ k = 1, ..., K, \\ \tau_{K+1} &= \tau_{\max}. \end{aligned} \tag{3.47}$$

Now define a data set of $(K+1)n$ records consisting of $\left(Y_{ik}, t_k\right)$ as

$$Y_{ik} = \mathbb{I}_{J_k}(Z_i)\delta_i,\ i = 1, ..., n,\ k = 0, ..K, \tag{3.48}$$

which indicates that a non-censored event for item i falls within the subinterval J_k.

Finally, let U_k be the proportion of failures observed to fail in the interval J_k, that is

$$U_k = \frac{1}{n} \sum_{i=1}^{n} Y_{ik},\ k = 0, ..., K. \tag{3.49}$$

Here, U_k/Δ are crude estimators of the subdensity values $f^*(t_k)$, which is

further smoothed using a discrete fast wavelet method (Antoniadis et al., 1999) to obtain an estimate $\hat{f^*}(t_k)$ for every evaluation point t_k.

As Equation (3.45) states, we still need an appropriate estimator of $L(t)$ to compute the hazard rate. Given the set of IID observations $Z_1, ..., Z_n$, the empirical distribution function L_n, which is the standard non-parametric estimator of L, is defined as

$$L_n(t) = \frac{1}{n} \sum_{i}^{n} \mathbb{I}_{[Z_i \leq t]}. \tag{3.50}$$

Since L_n does not take fully into account the smoothness of L, Antoniadis et al. (1999) use a traditional histogram-type estimator $\hat{l}_n$ of the density l of L in their paper. The integral of $\hat{l}_n$ will give an estimator of L, that is

$$\hat{L}_n(t) = \int_{\tau_{\min}}^{t} \hat{l}_n(x)\mathrm{d}x, \ \ t \in [\tau_{\min}, \tau_{\max}], \tag{3.51}$$

where $\hat{l}_n(t)$ is the Haar wavelet transform of the data and $\hat{L}_n(t)$ can be viewed as a wavelet estimator of the survival function. Hence, after estimating $\hat{L}_n(t_k)$ at the dyadic grid $\{t_k\}_{k=0,...,K}$ defined in Equation (3.46), we can then obtain the hazard rate estimation as follows,

$$\hat{\lambda}(t_k) = \frac{\hat{f}^*(t_k)}{1 - \hat{L}_n(t_k)}. \tag{3.52}$$

3.3.2 Density estimation by Bayesian wavelet thresholding and the bootstrap aggregating approach

When estimating $f^*(t)$ defined in Equation (3.45), Antoniadis et al. (1999) proposed a linear wavelet thresholding method, which basically sets to zero all

coefficients that are finer in resolution than some scale j, and keeps the values of the coarser wavelets up to that scale j (inclusive). Although this method retains the computational advantages as compared to other wavelet methods, it is not as flexible as nonlinear methods (Nason, 2008, p. 110). Hence, our method is based on Bayesian wavelet thresholding described in section 3.2.2.2, which uses a mixture prior that contains a point mass at zero mixed with a heavy-tailed distribution. To estimate $f^*(t)$, we also use the bootstrap aggregating approach described in section 3.3.2.1.

3.3.2.1 Bootstrap aggregating approach

Bootstrap aggregating, also known as bagging (Breimann, 1996), is a machine learning ensemble algorithm designed to raise stability of algorithms by reducing variance and lowering the bias. It creates a required number of different sets of the same size with replacement from the original training data set. Using the same machine learning scheme, it builds a model for each set. To deal with a regression problem, new predictions are made by averaging the predictions from the individual models, while in the classification context, predictions are combined by voting a nominal target or averaging the estimated class probabilities together.

To attempt to give a more accurate hazard rate, we apply bagging to estimate the density $f^*(t)$. To achieve this computationally, we use the R function `censboot` to generate b sets of right-censored survival data from the original set, then compute $\hat{f}_i^*(t)$ by Bayesian wavelet shrinkage for $i = 1, ..., b$, which is the estimate of the density $f^*(t)$ for the i-th data set. The bagged version of

the estimate of $f^*(t)$ is as follows,

$$\hat{f^*}_{bagging}(t) = \frac{1}{b}\sum_{i=1}^{b} \hat{f_i^*}(t). \tag{3.53}$$

Although bootstrap aggregating can be highly accurate, it can be computationally expensive, which may discourage its routine use.

3.3.3 Survival function estimation using a Dirichlet process model

In section 3.3.1, we mentioned that Antoniadis et al. (1999) integrate the traditional histogram-type estimator $\hat{l}_n$ to obtain the estimator $\hat{L}_n$ in the denominator of the hazard function, which is reasonably fast and accurate. In an attempt to improve the accuracy of the hazard rate, we use a Dirichlet process model to compute the estimator, which we call $\hat{L}_{DP}$, as the $L(t)$ can be treated as a CDF.

Let $\mathbf{x} = \{x_1, ..., x_n\}$ be a vector of realisations sampled from an unknown distribution L, with $L \sim DP(\alpha, G_0)$, where α and G_0 are defined in section 3.2.3. We are able to sample the posterior probability distributions with N realisations from a Dirichlet process using the following algorithm.

Algorithm 3.1: Sample From a Dirichlet Process Posterior

Function `post.DP`(N, $\boldsymbol{x}$, α, G_0, m)**:**

Let $i = 1$ and n be the length of input vector $\mathbf{x}$.

while $i \leq N$ **do**

Draw $\phi_1, ..., \phi_m$ independently from the distribution, such that $\frac{\alpha G_0 + \sum_{j=1}^{n} \delta_{x_j}}{\alpha + n}$, where δ_{x_j} is a point-mass at x_j.

Draw $z_1, ..., z_m$ independently from Beta $(1, \alpha + n)$.

Let $w_1 = z_1$ and compute $w_k = z_k \prod_{j=1}^{k-1} (1 - z_i)$, for $k = 2, ..., m$

Puts mass w_k at δ_{ϕ_k} and obtain a probability distribution, that is $\sum_{k=1}^{m} w_k \delta_{\phi_k}$.

Let $i = i + 1$

return a set of probability distributions with sample size of N.

Therefore, the sample posterior mean or median obtained by the algorithm can be used to estimate the CDF $L(t)$ (i.e. $\hat{L}_{DP}$) . Hence, the denominator of the hazard ratio is computed by 1 minus this estimate $\hat{L}_{DP}(t)$.

3.3.4 Simulation and results comparison

An advantage of using simulations is that we know the true hazard ratio, which can be used to compare the truth with the results of our approaches. Here, we carry out two simulation experiments, where the observations are from a density containing a discontinuity and a smooth Weibull distribution, respectively, and compare our new approaches with the established non-parametric presmoothed method described in section 3.2.4, and the true hazard rate.

As mentioned above, we compute hazard rate estimators using Bayesian wavelet shrinkage and bagging to obtain $\hat{f}^*(t)$, meanwhile using the Dirichlet process model to produce $\hat{L}_{DP}(t)$. Since our estimates are not guaranteed to be non-negative, $\hat{f}^*(t)$ and $\hat{L}_{DP}(t)$ are replaced by zero if their values are negative in the simulation. Also, while computing the hazard rate, we target the points where $L(t) < 0.9$, as the hazard rate is unstable when $L(t)$ is close to 1. For example, Figures 3.2 and 3.3 in section 3.3.4.2 simulation 1 show the hazard estimation using different methods, where the observations are from a density containing a discontinuity. These figures are computed with $L(t) < 0.9$, whereas Figure 3.1 demonstrates the situation where $L(t)$ is up to approximately 0.99. We can see that the hazard rate, both the truth and estimates, shoot up when $t > 0.7$ (i.e. $L(t)$ large), which is usually less meaningful and in the tails of the distribution of $L(t)$, which is not well-estimated with a finite set of data.

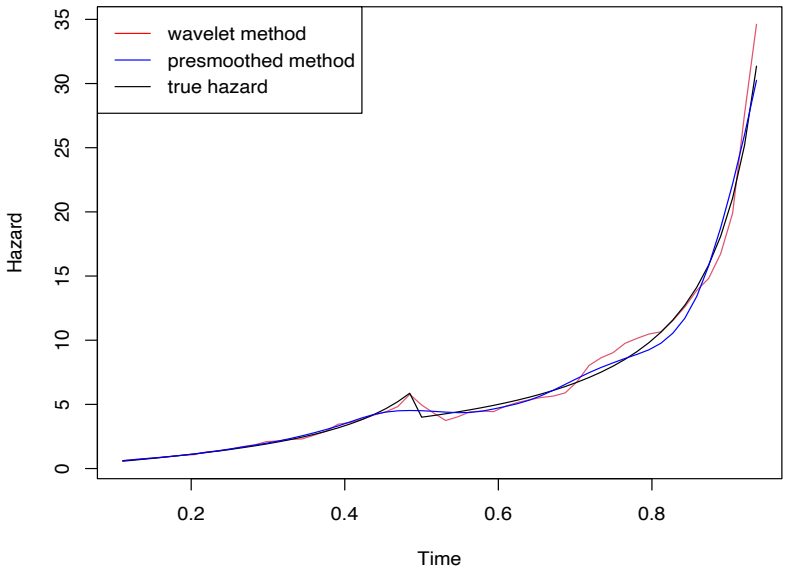


Figure 3.1: Hazard plot when $L(t)$ is up to approximately 0.99. Details and the parameters used to produce the curves are explained in section 3.3.4.2.

3.3.4.1 Suggestions for parameter choices

Practical guidelines for the choice of wavelet method, filter, and length etc. in wavelet-based methods are important components of any methodology. For example, Zhang et al. (2016) explicitly explore the effects of these essential parameters on the estimated values of graph metrics and in their sensitivity to alterations in psychiatric disease. In our studied cases, we can choose the parameters in estimating f^* by minimising the mean squared error, which cannot be done in real data situations as we do not know the truth. In addition, this may be computationally expensively in practice as every combination of the parameters would need to be considered. Hence, after much practical experimentation, this section provides suggestions on how to approach choosing the parameters for different data.

Our suggestions mainly focus on the following parameters: wavelet filter (Daubechies Extremal Phase or Daubechies Least Asymmetric), wavelet length (from one to ten for Daubechies Extremal Phase and four to ten for Daubechies Least Asymmetric), and the number of bins (i.e. the dyadic grid defined in Equation (3.46)). We have observed that no noticeable differences in estimation from different wavelet families of the same wavelet filter length, while a much larger factor impacting estimation was the wavelet filter length. Larger wavelet filter lengths provide smoother wavelets, but they increase the computational burden. Hence, we suggest choosing a wavelet of a relatively large length at the first trial, to ensure the results are relatively robust to small perturbations in wavelet length, then decrease the wavelet length if the larger one is too computationally expensive. Regarding the number of bins, we find

that larger datasets can benefit from a relatively larger number of bins. Usually, we suggest not to use this method if the number of observations is below 200, which may not result in enough data in each bin for obtaining an accurate estimation under this model. Suppose there are n observations, we find that a good number of bins is approximately $2^{\text{floor}\{\log(n)-2\}}$. For example, if the number of observations is around or below 1000, then 16 might be a good choice for the number of bins; if the number of observations is around 2000, then we can try 32; if the number of observations is above 3000, 64 or even larger number of bins can be used. A question for future work is to study more precisely, via theory, a good number of bins.

While estimating the CDF $L(t)$ using the Dirichlet process model, we find in practice, larger N and m mentioned in Algorithm 3.1 will result in a more accurate estimate of $\hat{L}_{DP}$, which will also increase the running time. A good choice for N and m to balance the estimation performance and computational cost is 1000. In terms of the hyperparameter choices of the Dirichlet process prior (i.e. the concentration parameter α and the baseline distribution G_0 explained in Equation (3.33)), it is difficult to give an optimal guess of the prior to obtain the most accurate estimation in reality, as users usually do not have too much knowledge about the dataset. However, after much experimentation, we find that different hyperparameters do not significantly affect the accuracy of the estimation. Our simulation examples displayed below use simple normal distributions as the baseline, which follow Ishwaran and James's (2001) work and give an improvement in estimation. The optimal choices of hyperparameters are not the focus of this project, further study should be carried out on this area.

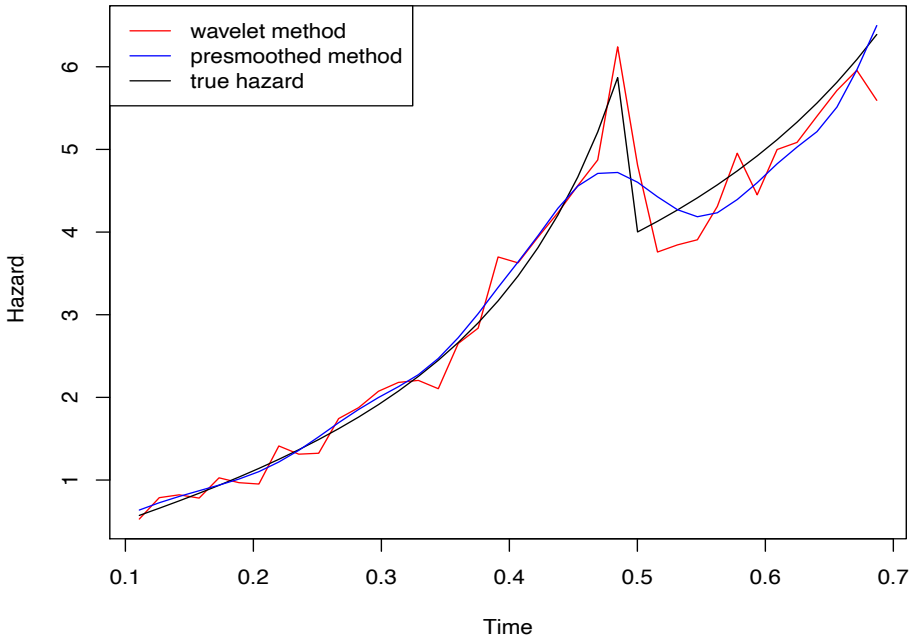
3.3.4.2 Simulation 1: density with a discontinuity

Suppose $X_1,...,X_n$ are non-negative IID, simulated from the density function f, such that

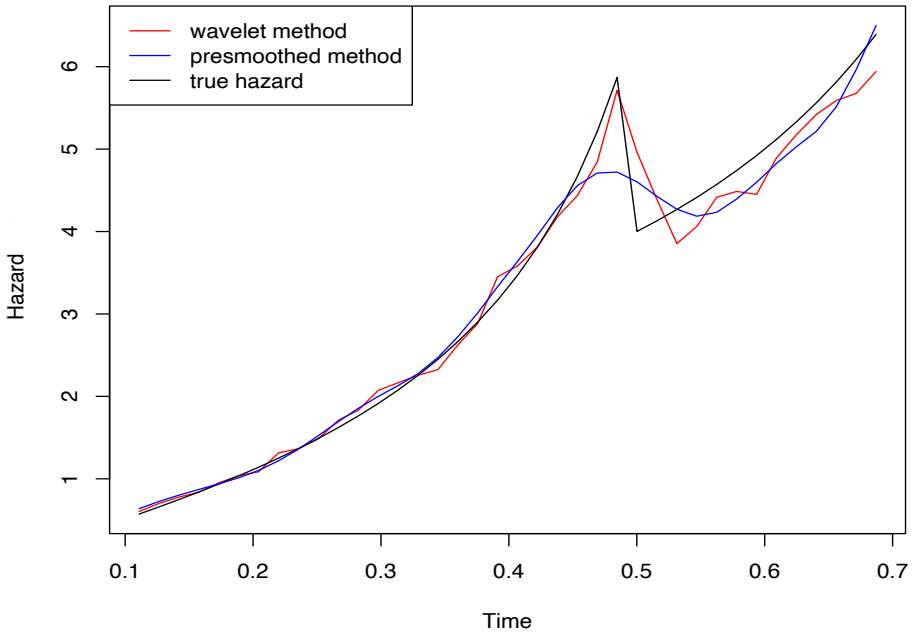
$$f(x) = \begin{cases} 5x & \text{if } x \in [0, 0.5), \\ -3x + 3 & \text{if } x \in [0.5, 1). \end{cases} \tag{3.54}$$

The corresponding censoring times $C_1,...,C_n$ are simulated from the Exp(0.8) distribution. Thus, we can obtain a set of observed random variables (Z_i, δ_i) defined as Equation (3.1).

Figures 3.2 and and 3.3 display the hazard estimate obtained with one realisation of $n = 5000$ observations, where the black line in both figures represents the true hazard corresponding to the density function as Equation (3.54). In our experiment, we choose the number of bins to be 64 and use Daubechies' least asymmetric wavelets with `filter.number` equal to 7, which minimises the mean squared error among all other options of different wavelets. When using our bagging method, to improve the estimation of f^* in both figures, the number of bootstrap replicates is 200. Our hazard estimates in the two plots of Figure 3.2 are computed by the histogram method, $\hat{L}_n$ (Equation (3.51)), which is replaced by the Dirichlet process model, $\hat{L}_{DP}$ (section 3.3.3), in Figure 3.3. While estimating $\hat{L}_{DP}$, as suggested in section 3.3.4.1, the concentration parameter α is 10, the base measure G_0 is a normal distribution $N(1,1)$ and the parameters N and m from a Dirichlet process model described in Algorithm 3.1 are chosen to be 1000.



(a) $\hat{f}^*$ estimated by Bayesian wavelet shrinkage



(b) $\hat{f}^*$ estimated by Bayesian wavelet shrinkage combined with bagging approach

Figure 3.2: Hazard rate estimation: the red line is produced using our wavelet method with $\hat{L}_n$ and $\hat{f}^*$; the blue line is produced using presmoothed method and the black line is the true hazard rate.

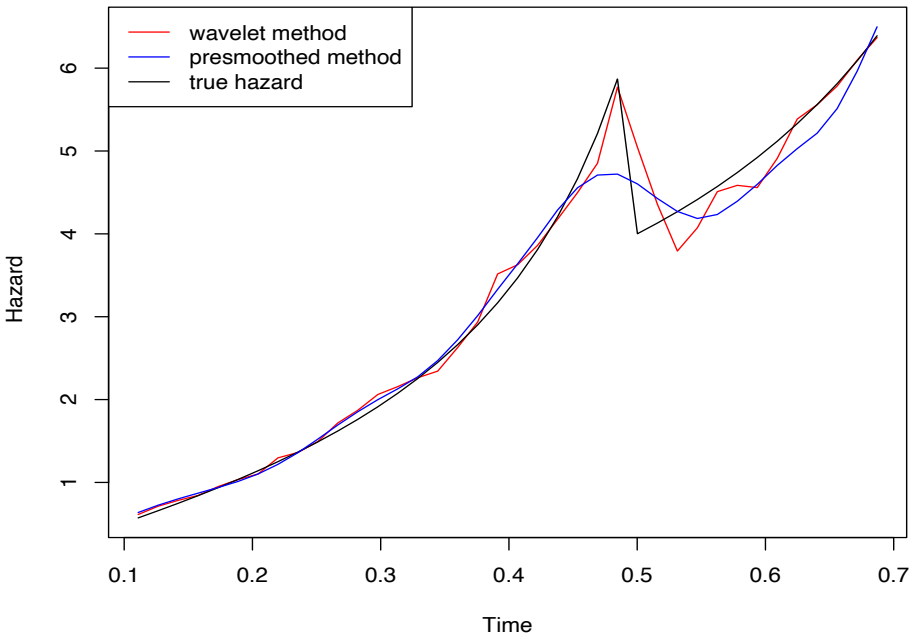


Figure 3.3: Hazard rate estimation: the red line is produced using our wavelet method with $\hat{L}_{DP}$ and bagging for $\hat{f}^*$; the blue line is produced using presmoothed method and the black line is the true hazard rate.

From Equation (3.54), the density function has a discontinuity at 0.5, which results in a 'jump' in the estimated hazard rate. The presmoothed estimate fits the truth well during the interval $(0.1, 0.4)$. However, it is oversmoothed near to $x = 0.5$. Also, the presmoothed estimate for $x \in (0.5, 0.7)$ is not as accurate as in $(0.1, 0.4)$. This is why we prefer our wavelet method to estimate the hazard rate when the density function has discontinuities. Although our wavelet estimate is not smooth, all figures shows that they capture the main feature of the true hazard, especially for the region around the discontinuity.

According to the figures produced for this particular dataset, we can see that our approach, using Bayesian wavelet shrinkage and bagging to estimate f^*

and Dirichlet process modelling to obtain $\hat{L}_{DP}$, performs the best and mimics the truth almost everywhere in our target interval. However, although the Dirichlet process model is accurate and allows us to estimate the survival function L at any time point, which means it is not necessary to use a dyadic grid, it is time consuming. In addition, bagging indeed improves the estimation of f^*, which smooths our hazard estimation considerably compared to using Bayesian wavelet shrinkage only (without bagging). We also compute the mean squared error for our different approaches for this dataset, shown in Table 3.1. The mean squared error for the presmoothed method is 7.83×10^{-2}, which is larger than our wavelet worst case. This may be because the presmoothed method does not perform well at the discontinuity. From this experiment, we can see that wavelet method may be a better choice for density functions with discontinuities.

Method to Estimate Hazard Rate			MSE ($\times 10^{-2}$)
Presmoothed Method			7.83
(a)	f^* L	Bayesian Wavelet Shrinkage Antoniadis et al. (1999) Method (i.e. $\hat{L}_n$)	7.64
(b)	f^* L	Bayesian Wavelet Shrinkage and Bagging Antoniadis et al. (1999) Method (i.e. $\hat{L}_n$)	6.61
(c)	f^* L	Bayesian Wavelet Shrinkage and Bagging Dirichlet Process Model (i.e. $\hat{L}_{DP}$)	5.47

Table 3.1: Mean squared error for one realisation using different approaches to estimate hazard rate, where the parameters and hyperparameters are described at the beginning of section 3.3.4.2.

The mean squared error produced in Table 3.1 is when the number of observations $n = 5000$, which is a reasonable choice in this case. As mentioned in section 3.3.4.1, the number of bins chosen is related to the number of obser-

vations n and the distribution of the dataset. For this particular setup stated on page 77, we produce a plot showing the mean squared errors for different number of observations n. Figure 3.4 shows that when n is greater than 5000, the mean squared error produced using method (c) in Table 3.1 is reasonably small. Although an increase in the number of observations will lower the mean squared error, it also becomes more time-consuming, which may not be considered in practice.

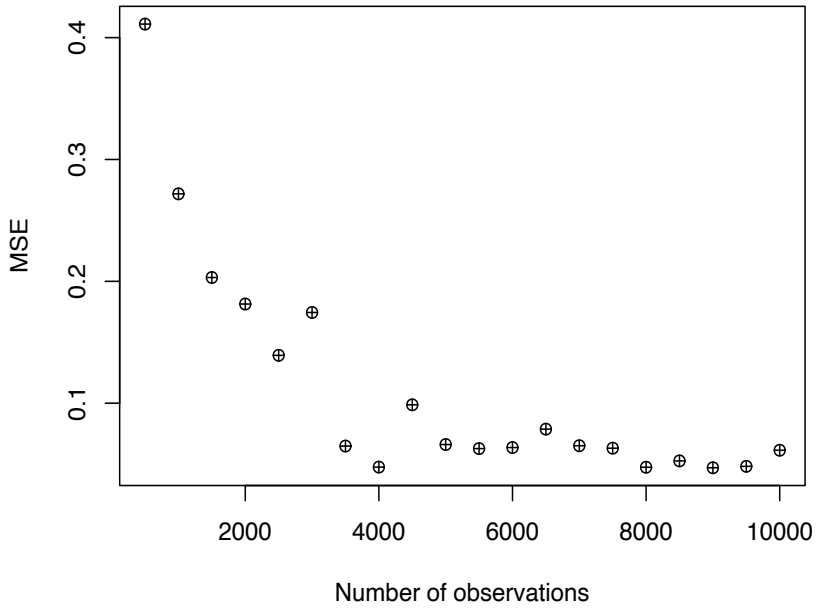


Figure 3.4: Mean squared error for different number of observations n , where method (c) is used in Table 3.1 in the wavelet approach.

To make our results more convincing, we repeat our experiment 200 times with different datasets simulated by Equation (3.54) and produce a boxplot for the MSE shown in Figure 3.5, where we find that our wavelet method clearly has better performance than the presmoothed approach.

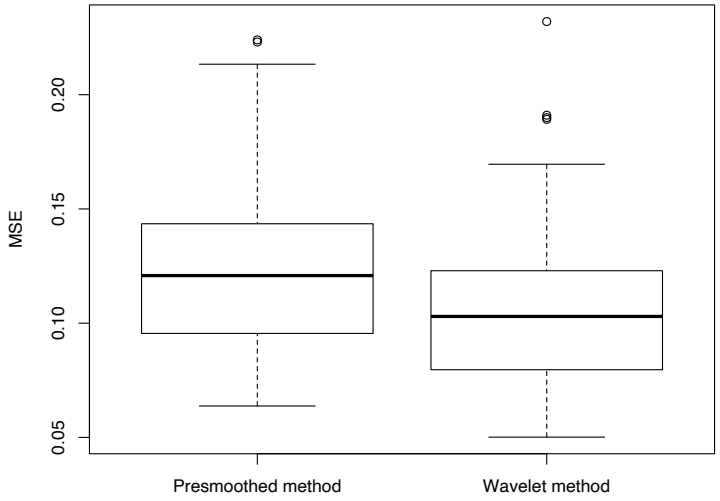


Figure 3.5: Boxplot of the mean square error for hazard rate estimation with 200 datasets of $n = 5000$, where method (c) in Table 3.1 is used in the wavelet approach.

3.3.4.3 Simulation 2: Weibull distribution

Considering now the situation where observations are from a continuous distribution, our method also performs reasonably well. Suppose $X_1,...,X_n$ are non-negative IID, simulated from a Weibull distribution with density

$$f(x;k,\lambda) = \begin{cases} \frac{k}{\lambda}\left(\frac{k}{x}\right)^{k-1} e^{-(k/x)^k} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0, \end{cases} \tag{3.55}$$

where the shape parameter $k > 0$ is chosen to be 3.5 and the scale parameter $\lambda > 0$ is 1.8 in this experiment. The corresponding censoring times $C_1,...,C_n$ are simulated from the Exp(0.4) distribution.

Figure 3.6 displays the hazard estimate computed with one dataset of $n = 2000$ observations, where we can see both wavelet and presmoothed approach mimic the truth until $x = 1.3$. However, the presmoothed method has better performance when $x \in (1.3, 1.8)$, while our wavelet method has lower mean squared error after $x = 1.8$.

While using our wavelet method to estimate f^*, we use the Bayesian wavelet shrinkage method mentioned previously plus bagging to increase the accuracy. In this experiment, the number of bins is 32, and Daubechies' least asymmetric wavelets with `filter.number` equal to 7 is chosen, and the number of bootstrap replicates is 200. Also, the survival function L is estimated by $\hat{L}_{DP}$, where the concentration parameter α is 5, the base measure G_0 is a normal distribution $N(1.5, 1)$ and the sample size N in (3.37) is chosen to be 2000. After computing the mean squared error for this dataset, we find that our wavelet estimate surprisingly has the lower value (i.e. 0.0075) compared to the presmoothed result, that is 0.013.

Figure 3.7 is a boxplot produced with 200 different datasets simulated by the same distribution as described in Equation (3.55), where we find that both methods have similar performance although the presmoothed method has two more outliers than our wavelet method for the 200 datasets. This may be because our wavelet method does not have significant advantages when the datasets are drawn from a continuous distribution.

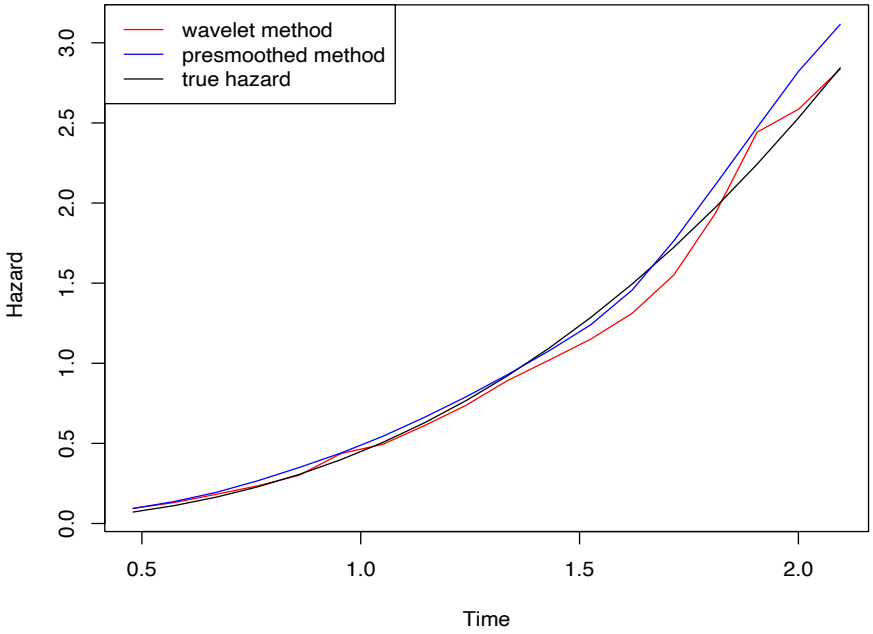


Figure 3.6: Hazard rate estimation: the red line is produced using our wavelet method with $\hat{L}_{DP}$ and bagging for $\hat{f}^*$; the blue line is produced using presmoothed method and the black line is the true hazard rate.

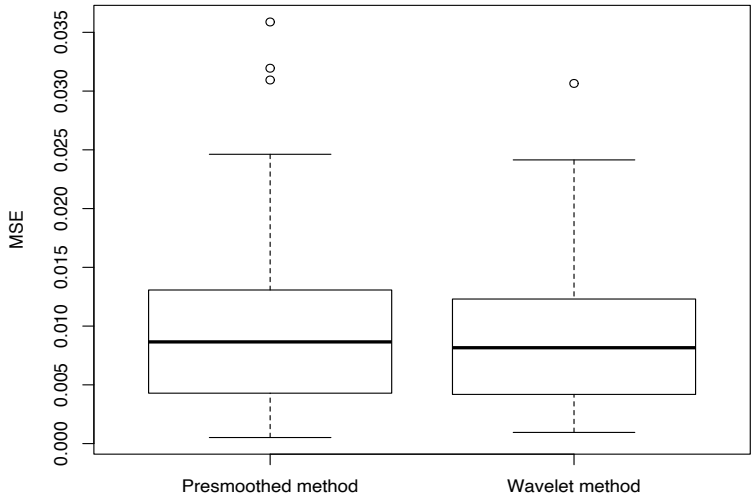


Figure 3.7: Boxplot of mean square error for hazard rate estimation with 200 datasets of $n = 5000$ from the Weibull distribution, where method (c) in Table 3.1 is used in the wavelet approach.

These simulation studies indicate that in terms of estimation accuracy, our wavelet method, which uses the Dirichlet process model to estimate CDF, has promise in that it works well in situations where the underlying density has a discontinuity, but also competitive in the underlying smooth case. However, this method has highly variable run times, which depends on the initial hyper-parameter settings. If hyperparameters N and m in Algorithm 3.1 are large, then this can result in extended execution times, whereas if they are small, then the execution times are reduced. Of course, different hyperparameter settings will also impact on the accuracy of our overall result. It is difficult to be specific about how the hyperparameters interact to produce specific execution times and accuracy and further work would be required to elicit this. Roughly speaking, in our experience with simulations, our wavelet method that uses the Dirichlet process model can take up to ten to a thousand times longer than the presmoothed method. Users could decide which method to use according to their primary goal, speed or accuracy. Usually, if high standard of accuracy is not required, our wavelet method with bagging approach and the traditional histogram-type method to estimate CDF could be a good choice.

3.4 New Method for Density Estimation

One limitation of using the Bayesian wavelet method mentioned previously is that our evaluation points must be on a dyadic grid. Hence, we introduce a new approach to solve this problem while estimating the density function. Our main idea is based on Herrick et al.'s (2001) work, which exploits the non-stationary variance structure of the wavelet coefficients. For the first

time, we consider the detailed covariance structure of the empirical wavelet coefficients and propose a multivariate version of the "mixture of Gaussians" prior distribution in the Bayesian wavelet thresholding approach.

3.4.1 Prior mixture of Gaussians

Let $f(x)$ be a probability density function. Let $X_1, ..., X_n$ be an independent and identically distributed (IID) sample from f. Recall Equation (3.6): for some $M \in \mathbb{Z}$, the wavelet representation of the density function is

$$f(x) \sim \sum_{k \in \mathbb{Z}} c_{Mk} \phi_{Mk}(x) + \sum_{j=M}^{\infty} \sum_{k \in \mathbb{Z}} d_{jk} \psi_{jk}(x), \tag{3.56}$$

where the coefficients c_{Mk} and d_{jk}, as described in Equations (3.8) and (3.7), are scaling and wavelet coefficients respectively.

By considering the coefficients as expectations $c_{jk} = \mathbb{E}\Big[\phi_{jk}(X)\Big]$ and $d_{jk} = \mathbb{E}\Big[\psi_{jk}(X)\Big]$, we can compute the empirical coefficients $\widetilde{c}_{jk}$ and $\widetilde{d}_{jk}$ by

$$\widetilde{c}_{jk} = \frac{1}{n} \sum_{i=1}^{n} \phi_{jk}(X_i), \tag{3.57}$$

$$\widetilde{d}_{jk} = \frac{1}{n} \sum_{i=1}^{n} \psi_{jk}(X_i), \tag{3.58}$$

which are calculated for resolution levels M up to some large primary resolution level J, which is the coarsest level to which thresholding is applied (Hall and Patil, 1995). The empirical coefficients $\widetilde{d}_{jk}$ can then be thresholded using our Bayesian wavelet shrinkage method to obtain the estimated coefficients $\widehat{d}_{jk}$. Then, the estimated density is

$$\widehat{f}(x) = \sum_{k \in \mathbb{Z}} \widetilde{c}_{Mk} \phi_{Mk}(x) + \sum_{j \in \mathscr{J}_M} \sum_{k \in \mathscr{K}} \widehat{d}_{jk} \psi_{jk}(x), \tag{3.59}$$

where $\mathcal{J}_M = \{j \in \mathbb{Z} : M \leq j < J\}$, $\mathcal{K} = \{k \in \mathbb{Z} : k_{min} \leq k \leq k_{max}\}$, k_{min} and k_{max} are the minimum and maximum values of k needed, so that the coefficients "cover" the data.

After computing all the empirical coefficients by Equations (3.57) and (3.58) by the wavelet transform, we can apply the Bayesian wavelet shrinkage for the empirical coefficients using the "mixture of Gaussians" prior distribution (Chipman et al., 1997) for each unknown "true" wavelet coefficient d_{jk}:

$$d_{jk}|\gamma_{jk} \sim \gamma_{jk} N(0, c_j^2 \tau_j^2) + (1 - \gamma_{jk}) N(0, \tau_j^2), \tag{3.60}$$

where γ_{jk} is a Bernoulli random variable with prior distribution of

$$P(\gamma_{jk} = 1) = 1 - P(\gamma_{jk} = 0) = p_j, \tag{3.61}$$

and p_j, c_j, and τ_j are all hyperparameters to be chosen. The prior parameter τ_j is typically set to be small and the hyperparameter c_j should be set to be much larger than one.

Hence, let $\mathbf{d} = \{d_{jk}\}^T_{j \in \mathcal{J}_M, k \in \mathcal{K}}$ be a vector of the true wavelet coefficients and $\boldsymbol{\gamma} = \{\gamma_{jk}\}^T_{j \in \mathcal{J}_M, k \in \mathcal{K}}$, according to Equation (3.60), we have the multivariate normal distribution

$$\mathbf{d}|\boldsymbol{\gamma} \sim N(\mathbf{0}, \Psi), \tag{3.62}$$

where $\mathbf{0}$ is a vector of zeros and Ψ is a *diagonal* covariance matrix with diagonals defined as $\gamma_{jk} c_j^2 \tau_j^2 + (1 - \gamma_{jk}) \tau_j^2$ for $j \in \mathcal{J}_M, k \in \mathcal{K}$.

Similar to the wavelet-transformed model defined in Equation (3.24), we have the multivariate version such that

$$\widetilde{\mathbf{d}} = \mathbf{d} + \boldsymbol{\epsilon}, \tag{3.63}$$

where $\widetilde{\mathbf{d}} = \{\widetilde{d}_{jk}\}^T_{j\in\mathcal{J}_M,k\in\mathcal{K}}$ is a vector of noisy wavelet coefficients, and $\boldsymbol{\epsilon} = \{\epsilon_{jk}\}^T_{j\in\mathcal{J}_M,k\in\mathcal{K}}$ is a vector of noise.

In Chipman et al.'s (1997) model, they considered the likelihood of the observed wavelet coefficients, which follows a IID normal distribution. In our work, we compute the empirical coefficients, which are not mutually independent, as each observation contributes to many different empirical coefficients, as shown by Equations (3.57) and (3.58) from Herrick et al. (2001). Hence, we assume the likelihood of the empirical wavelet coefficients is given by

$$\widetilde{\mathbf{d}} \sim N(\mathbf{d}, \Sigma), \tag{3.64}$$

where Σ is the covariance matrix defined by

$$\mathrm{Cov}\left[\widetilde{d}_{j_1k_1}, \widetilde{d}_{j_2k_2}\right] = \frac{1}{n}\Big\{\mathbb{E}\left[\psi_{j_1k_1}(X)\,\psi_{j_2k_2}(X)\right] - \mathbb{E}\left[\psi_{j_1k_1}(X)\right]\mathbb{E}\left[\psi_{j_2k_2}(X)\right]\Big\}. \tag{3.65}$$

Then, by the Bayes' theorem, we can compute the posterior mean of $\mathbf{d}$ given $\widetilde{\mathbf{d}}$, which can be chosen as our "estimate" $\left(\text{i.e. } \widehat{\mathbf{d}} = \{\widehat{d}_{jk}\}^T_{j\in\mathcal{J}_M,k\in\mathcal{K}}\right)$ of the "true" wavelet coefficients.

From Bayes theorem, the density of the posterior distribution is

$$f(\mathbf{d}|\widetilde{\mathbf{d}}, \boldsymbol{\gamma}) = f(\widetilde{\mathbf{d}}|\mathbf{d}, \boldsymbol{\gamma}) f(\mathbf{d}|\boldsymbol{\gamma}) / f(\widetilde{\mathbf{d}}|\boldsymbol{\gamma}) \propto f(\widetilde{\mathbf{d}}|\mathbf{d}, \boldsymbol{\gamma}) f(\mathbf{d}|\boldsymbol{\gamma}), \tag{3.66}$$

where

$$f(\widetilde{\mathbf{d}}|\mathbf{d}, \boldsymbol{\gamma}) = f(\widetilde{\mathbf{d}}|\mathbf{d}) = |\det(2\pi\Sigma)|^{-1/2} \exp\Big\{-\frac{1}{2}(\widetilde{\mathbf{d}} - \mathbf{d})^T \Sigma^{-1} (\widetilde{\mathbf{d}} - \mathbf{d})\Big\}, \tag{3.67}$$

$$f(\mathbf{d}|\boldsymbol{\gamma}) = |\det(2\pi\Psi)|^{-1/2} \exp\Big(-\frac{1}{2}\mathbf{d}^T \Psi^{-1} \mathbf{d}\Big). \tag{3.68}$$

Hence, we have

$$f(\mathbf{d}|\tilde{\mathbf{d}},\boldsymbol{\gamma}) \propto \exp\Big\{-\frac{1}{2}(\tilde{\mathbf{d}}-\mathbf{d})^T\Sigma^{-1}(\tilde{\mathbf{d}}-\mathbf{d})\Big\}\exp\Big(-\frac{1}{2}\mathbf{d}^T\Psi^{-1}\mathbf{d}\Big). \tag{3.69}$$

$$= \exp\Big[-\frac{1}{2}\Big\{\tilde{\mathbf{d}}^T\Sigma^{-1}\tilde{\mathbf{d}}-\tilde{\mathbf{d}}^T\Sigma^{-1}\mathbf{d}-\big(\mathbf{d}^T\Sigma^{-1}\tilde{\mathbf{d}}\big)^T+\mathbf{d}^T\Sigma^{-1}\mathbf{d}+\mathbf{d}^T\Psi^{-1}\mathbf{d}\Big\}\Big] \tag{3.70}$$

$$= \exp\Big[-\frac{1}{2}\Big\{\tilde{\mathbf{d}}^T\Sigma^{-1}\tilde{\mathbf{d}}-2\tilde{\mathbf{d}}^T\Sigma^{-1}\mathbf{d}+\mathbf{d}^T\big(\Sigma^{-1}+\Psi^{-1}\big)\mathbf{d}\Big\}\Big]. \tag{3.71}$$

Deleting terms that do not depend on $\mathbf{d}$, we obtain

$$f(\mathbf{d}|\tilde{\mathbf{d}},\boldsymbol{\gamma}) \propto \exp\Big[-\frac{1}{2}\Big\{\mathbf{d}^T\big(\Sigma^{-1}+\Psi^{-1}\big)\mathbf{d}-2\tilde{\mathbf{d}}^T\Sigma^{-1}\mathbf{d}\Big\}\Big]. \tag{3.72}$$

We want to find $\boldsymbol{\mu}$ and Φ, such that

$$\mathbf{d}|\tilde{\mathbf{d}},\boldsymbol{\gamma} \sim N(\boldsymbol{\mu},\Phi), \tag{3.73}$$

which means

$$f(\mathbf{d}|\tilde{\mathbf{d}},\boldsymbol{\gamma}) \propto \exp\Big\{-\frac{1}{2}(\mathbf{d}-\boldsymbol{\mu})^T\Phi^{-1}(\mathbf{d}-\boldsymbol{\mu})\Big\} \tag{3.74}$$

$$= \exp\Big\{-\frac{1}{2}\Big(\mathbf{d}^T\Phi^{-1}\mathbf{d}-2\mathbf{d}^T\Phi^{-1}\boldsymbol{\mu}+\boldsymbol{\mu}^T\Phi^{-1}\boldsymbol{\mu}\Big)\Big\}. \tag{3.75}$$

Hence, by Equation (3.72), we obtain

$$\Phi = \big(\Sigma^{-1}+\Psi^{-1}\big)^{-1}, \tag{3.76}$$

$$\boldsymbol{\mu} = \big(\Sigma^{-1}+\Psi^{-1}\big)^{-1}\Sigma^{-1}\tilde{\mathbf{d}}. \tag{3.77}$$

Recall Ψ depends on $\boldsymbol{\gamma}$ and since $\boldsymbol{\gamma}$ is a Bernoulli random variable, it is difficult to analytically derive the full posterior $f(\mathbf{d}|\tilde{\mathbf{d}})$ integrating out $\boldsymbol{\gamma}$. Hence, we use Gibbs sampling (Geman and Geman, 1984) to obtain a sequence of observations

from the joint distribution of $\boldsymbol{\gamma}, \mathbf{d}|\widetilde{\mathbf{d}}$. Our algorithm, which enables us to find a sample of $\mathbf{d}|\widetilde{\mathbf{d}}, \boldsymbol{\gamma}$, is described by the following steps:

- Start with some initial value $\left(\boldsymbol{\gamma}^{(i)}, \mathbf{d}^{(i)}\right)$.
- For the next sample $\left(\boldsymbol{\gamma}^{(i+1)}, \mathbf{d}^{(i+1)}\right)$, we have

$$\boldsymbol{\gamma}^{(i+1)} \sim f\left(\cdot \,|\, \mathbf{d}^{(i)}, \widetilde{\mathbf{d}}\right),$$
$$\mathbf{d}^{(i+1)} \sim f\left(\cdot \,|\, \boldsymbol{\gamma}^{(i+1)}, \widetilde{\mathbf{d}}\right).$$

- Continue until the required number of samples are produced.

According to the procedures above, we need to find the distribution of $\boldsymbol{\gamma}|\mathbf{d}, \widetilde{\mathbf{d}}$. By the Bayes theorem, we find

$$f(\boldsymbol{\gamma}|\mathbf{d}, \widetilde{\mathbf{d}}) \propto f(\widetilde{\mathbf{d}}|\mathbf{d}, \boldsymbol{\gamma}) f(\mathbf{d}|\boldsymbol{\gamma}) f(\boldsymbol{\gamma}) \tag{3.78}$$
$$\propto f(\mathbf{d}|\boldsymbol{\gamma}) f(\boldsymbol{\gamma}). \tag{3.79}$$

Hence, for individual $\gamma_{jk}|d_{jk}, \widetilde{d}_{jk}$, we have

$$f(\gamma_{jk}|d_{jk}, \widetilde{d}_{jk}) = f(\gamma_{jk} = 1|d_{jk}, \widetilde{d}_{jk}) + f(\gamma_{jk} = 0|d_{jk}, \widetilde{d}_{jk}) \tag{3.80}$$
$$\propto f(d_{jk}|\gamma_{jk} = 1) f(\gamma_{jk} = 1) + f(d_{jk}|\gamma_{jk} = 0) f(\gamma_{jk} = 0). \tag{3.81}$$

This gives us the distribution of $\boldsymbol{\gamma}|\mathbf{d}, \widetilde{\mathbf{d}}$ to enable the Gibbs sampling work, which is the mixture parameter between the two Gaussians in the model defined in Equation (3.61).

3.4.2 Hyperparameter estimation

Like the JS model, we can obtain the estimates of hyperparameters in our model (e.g. c_j, τ_j and p_j explained in Equations (3.60) and (3.61)) by maximising marginal likelihood of $\widetilde{\mathbf{d}}$. Firstly, we compute the joint distribution of $\widetilde{\mathbf{d}}, \mathbf{d}, \boldsymbol{\gamma}$ as follows

$$f(\widetilde{\mathbf{d}}, \mathbf{d}, \boldsymbol{\gamma}) = f(\widetilde{\mathbf{d}}|\mathbf{d}, \boldsymbol{\gamma}) f(\mathbf{d}|\boldsymbol{\gamma}) f(\boldsymbol{\gamma}) \tag{3.82}$$

$$= C \exp\Big\{ -\frac{1}{2}(\widetilde{\mathbf{d}} - \mathbf{d})^T \Sigma^{-1} (\widetilde{\mathbf{d}} - \mathbf{d}) - \frac{1}{2} \mathbf{d}^T \Psi^{-1} \mathbf{d} \Big\} f(\boldsymbol{\gamma}) \tag{3.83}$$

$$= C \Big\{ \widetilde{\mathbf{d}}^T \Sigma^{-1} \widetilde{\mathbf{d}} - 2\widetilde{\mathbf{d}}^T \Sigma^{-1} \mathbf{d} + \mathbf{d}^T \big(\Sigma^{-1} + \Psi^{-1}\big) \mathbf{d} \Big\} f(\boldsymbol{\gamma}) \tag{3.84}$$

$$= C \exp\Big[-\frac{1}{2} \Big\{ \mathbf{d}^T \big(\Sigma^{-1} + \Psi^{-1}\big) \mathbf{d} - 2\widetilde{\mathbf{d}}^T \Sigma^{-1} \mathbf{d} \Big\} \Big] \exp\big(-\frac{1}{2} \widetilde{\mathbf{d}}^T \Sigma^{-1} \widetilde{\mathbf{d}} \big) f(\boldsymbol{\gamma}) \tag{3.85}$$

with $C = \Big| \det(2\pi\Sigma) \det(2\pi\Psi) \Big|^{-1/2}$. Integrating $f(\widetilde{\mathbf{d}}, \mathbf{d}, \boldsymbol{\gamma})$ with respect to $\mathbf{d}$ and $\boldsymbol{\gamma}$, we can find the marginal distribution of $f(\widetilde{\mathbf{d}})$. From Equations (3.73) to (3.76), we have

$$\int f(\widetilde{\mathbf{d}}, \mathbf{d}, \boldsymbol{\gamma}) \, \mathrm{d}\mathbf{d} = \frac{\Big| \det(2\pi\Phi) \Big|^{1/2}}{\Big| \det(2\pi\Sigma) \det(2\pi\Psi) \Big|^{1/2}} \exp\big(-\frac{1}{2} \widetilde{\mathbf{d}}^T \Sigma^{-1} \widetilde{\mathbf{d}} \big) f(\gamma_{jk}). \tag{3.86}$$

Then

$$f(\widetilde{\mathbf{d}}) = \sum_{j,k} f(\widetilde{\mathbf{d}}, \boldsymbol{\gamma}) = \sum_{j,k} \int f(\widetilde{\mathbf{d}}, \mathbf{d}, \boldsymbol{\gamma}) \, \mathrm{d}\mathbf{d} \tag{3.87}$$

$$= \sum_{j,k} \frac{\Big| \det(2\pi\Phi) \Big|^{1/2}}{\Big| \det(2\pi\Sigma) \det(2\pi\Psi) \Big|^{1/2}} \exp\big(-\frac{1}{2} \widetilde{\mathbf{d}}^T \Sigma^{-1} \widetilde{\mathbf{d}} \big) f(\gamma_{jk}). \tag{3.88}$$

In practice, we can find the estimates of the hyperparameters by maximising $f(\widetilde{\mathbf{d}})$ numerically using R function `optim`. However, it will be too computationally expensive as we need to sum over all the combinations of possible values

of γ_{jk} for $j \in \mathscr{J}_M = \{j \in \mathbb{Z} : M \leq j < J\}$ and $k \in \mathscr{K} = \{k \in \mathbb{Z} : k_{min} \leq k \leq k_{max}\}$ described in Equation (3.59) to obtain $f(\widetilde{\mathbf{d}})$.

3.4.3 Simulation and results comparison

Again, we carry out two simulations, where observations are from a Weibull distribution and a density with a discontinuity respectively. Here, we use our new method to compare with the JS model rather than the approach proposed by Chipman et al. (1997). This is because the JS model is a more recent development, a thorough robust underlying asymptotic theory, attention to detail and also has a user-friendly R package.

Suppose $X_1, ..., X_n$ are simulated from the Weibull distribution defined in (3.55) with $k = \lambda = 2$. As mentioned in section 3.3.4.1, a relatively larger filter number is a good choice to try, and Chipman et al. (1997) suggest that the hyperparameter c_j should be much greater than one and τ_j is typically small. Hence, in this experiment, we choose Daubechies' least asymmetric wavelets with the filter number of 7, and the hyperparameters in our new method are set to be $p_j = 0.01$, $c_j = 1000$, and $\tau_j = 0.001$ for $j = 1, ..., 4$. Also, as under the JS model, the number of evaluation points must be on a dyadic grid, the number of bins is chosen to be 32 for both wavelet models. Figure 3.8 shows estimation using our new method (red line) and using the JS model (blue line) for one particular data set with 2000 observations, where we can see both methods are close to the truth despite some inaccurate estimates when X_i are around 1.5.

We also plot a kernel density estimate (green line) in Figure 3.8, which looks

smoother than the estimation produced by the wavelet methods. However, the kernel estimation may be oversmoothed in the situation where the observations are generated from a distribution with discontinuities, which will be displayed in the next example.

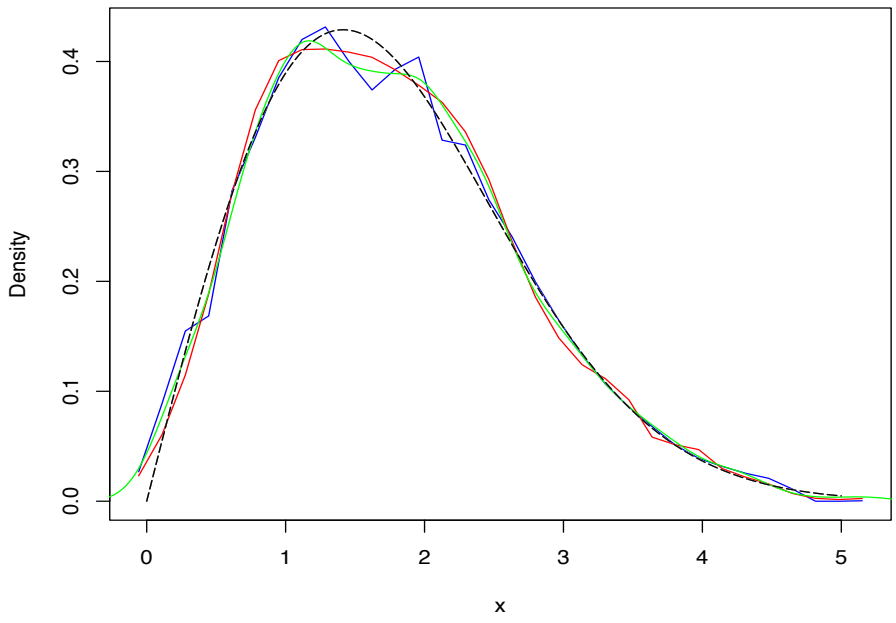


Figure 3.8: Density estimation: the red line represents the density estimated using our new method; the blue line is the density estimate using the R function `ebayesthresh.wavelet`; the green line is the kernel estimate; the black dash line represents the true density of Weibull $(2,2)$ distribution.

In order to compare the accuracy of the two wavelet methods, we generate 200 different sets of data from Weibull $(2,2)$ distribution and compare the mean squared error of these two wavelet-based estimates. We find that our new method has a better performance, as about 68% of the datasets have the lower mean square error using our new method rather than the JS model for this

particular distribution. Figure 3.9 shows a boxplot of the mean square error computed by the kernel method, JS model using R function `ebayesthresh` and our new approach respectively. In general, our new approach has the lower mean square error, although it has more outliers.

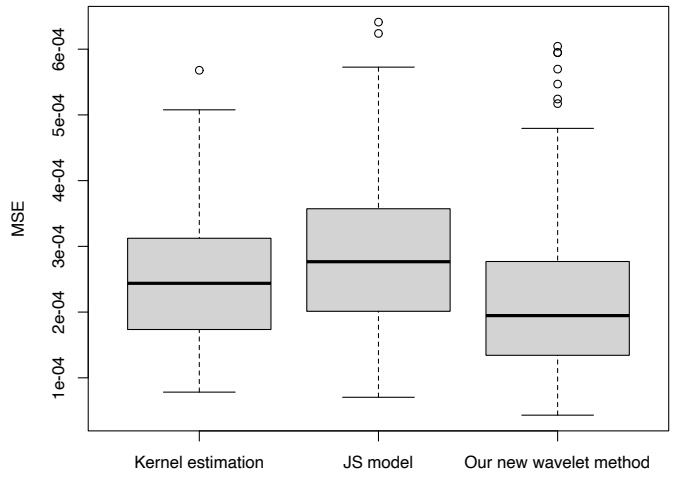


Figure 3.9: Boxplot of mean square error for density estimation with 200 datasets.

Now, let us focus on a second simulation, where the X_is are simulated from the distribution with density f defined as

$$f(x) = \begin{cases} \frac{2x}{75} & \text{if } x \in [0,5), \\ \frac{40-4x}{75} & \text{if } x \in [5,10). \end{cases} \tag{3.89}$$

Again we choose Daubechies' least asymmetric wavelets with the filter number of 7, and the number of bins is chosen to be 32 . The hyperparameters in our new method are set to be $p_j = 0.01$, $c_j = 1000$, and $\tau_j = 0.001$ for $j = 1,...,4$. Figure 3.10 shows the density estimates using different methods for one dataset

with 2000 observations. As we can see, there is a discontinuity at $x = 5$. However, compared to the other two wavelet-based methods, the kernel estimate does not capture this feature for this particular dataset.

Now we generate 200 different datasets from this distribution, which shows that about 54% and 58% of ours have the lower mean square error than the JS model and kernel estimation respectively. The boxplots of the mean square error for the kernel method and two wavelet models in Figure 3.11 also show that both wavelet methods have similar performance, where our new wavelet method has the slightly smaller mean squared errors in general. In addition, the kernel method does not produce extreme mean squared errors, as it oversmoothes the density function, especially near the discontinuity point.

From the two simulations, we may conclude that in terms of the accuracy, our new wavelet method gives an improvement, although there is not always an overwhelming superiority. Comparing to the JS model, the other main advantage of our new method is that it does not require a dyadic grid for the evaluation points. However, our new method is time consuming as it requires multivariate computation.

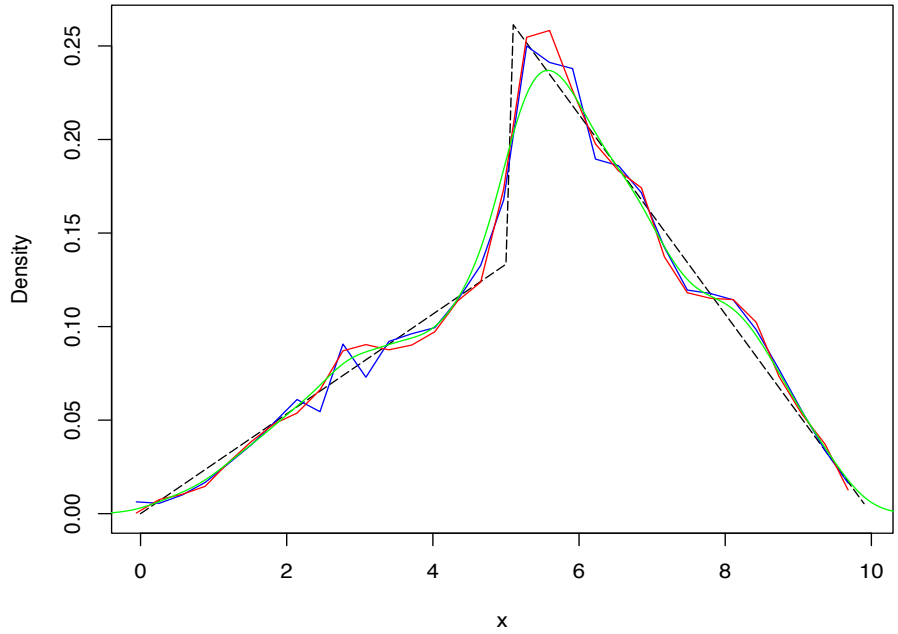


Figure 3.10: Density estimation: the red line represents the density estimated using our new method; the blue line is the density estimate using the R function `ebayesthresh.wavelet`; the green line is the kernel estimate; the black dash line represents the true density.

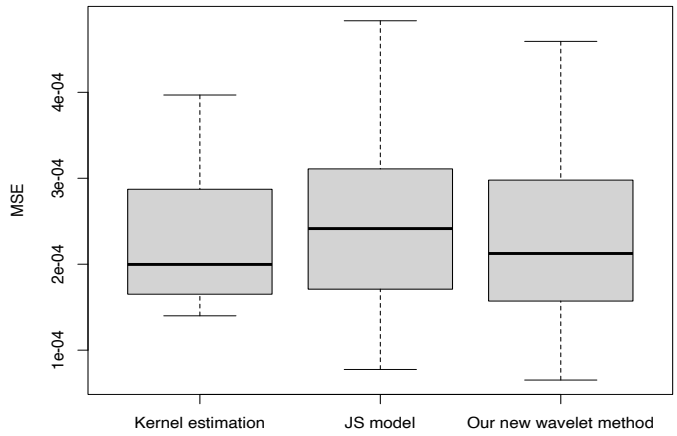


Figure 3.11: Boxplot of mean square error for density estimation with 200 datasets.

3.5 Conclusions

This chapter presents the methods to improve the estimates of the density and hazard rate using the wavelet approaches. For hazard rate estimation, we use the Bayesian threshold approach to estimate the density function, and a non-parametric Bayesian method with Dirichlet process prior to estimate the survival function. We display two simulation examples, where the observations are from a density containing a discontinuity and a smooth Weibull distribution, respectively. Both simulations show that our method works well compared to the presmoothed method (Lopez-de Ullibarri and Jacome, 2013).

For density function estimation, we propose a multivariate version of the "mixture of Gaussians" prior distribution in the Bayesian wavelet thresholding approach using the detailed covariance structure of the empirical wavelet coefficients. The two simulation experiments show that comparing to the JS model (Silverman and Johnstone, 2004, 2005a,b) and the kernel density estimation, our new wavelet method, which does not require a dyadic grid for the evaluation points, gives an improvement but not always an overwhelming superiority.

Our wavelet methods is useful in the real applications. This is because using smoothed methods such as kernel estimation or presmoothed method to estimate the hazard or density functions will never show a discontinuity even if there was one. Our wavelet methods working as an "insurance", will give users the capability to show this structure, especially in the situation when there is a high possibility to have a discontinuity. Antoniadis et al. (1999) show an employment example with discontinuities in hazard estimate of employment

time by three months, as people could only get benefit from the government for the first three months of the unemployed period. Actually, any sharp cut-off in policy or rules may cause discontinuities in hazard or density functions.

Further investigation could be carried out on choosing the hyperparameters efficiently and accurately for our new wavelet method, as our current work determines the hyperparameters based on the rough suggestions of Chipman et al.'s (1997) work. Although in section 3.4.2, we derive the marginal likelihood of the empirical wavelet coefficients and try to maximum the likelihood numerically to find the estimates of the hyperparameters, it is computationally inefficient to be applied in real situations.

CHAPTER 4

CLUSTERED RECURRENT LIFETIMES ANALYSIS

4.1 Introduction

Survival analysis is one of the most important statistical methods, which analyses the lifetime of individuals until an event occurs. It is widely used in many areas such as biological research or mechanical systems, where the event can be defined as death or failure respectively. Traditional survival analysis usually only contains a single event occurring for each individual: death or survival. Our work focuses on recurrent survival times, which allows us to have several lifetimes for each individual. Epilepsy, malaria, bladder cancer, etc. are common situations with recurrent survival times.

Our goal is to use methods to form groups of individuals and estimate survival functions for each group, which is useful when we only have few lifetimes per individual. Section 4.3 illustrates our methods to group individuals, which can be constructed using the dissimilarity matrix of the individuals from separate covariate information. For example, if the information of the individuals comes with their geographical location, then the dissimilarity matrix can be computed

directly, or we can compute the matrix according to the covariates associated with each individual. Assuming the lifetimes of individuals in the same group are from the same independent distribution, we can improve the estimates of the survival function by grouping lifetimes of individuals from each cluster. We can then use the estimates to predict lifetimes of individuals with similar covariates.

4.2 Literature Review

This chapter uses a basic knowledge of survival data analysis which has been reviewed in Chapter 3.

4.2.1 Methods for modelling lifetimes

There are many methods for modelling lifetimes, such as using parametric or non-parametric distributions and regression models. For parametric methods, the exponential and the Weibull distributions are commonly considered. Each makes a different assumption about the nature of the hazard rate. The exponential distribution has a constant hazard function, which reflects a lack of memory property. However, the constant hazard rate is often untenable. The Weibull distribution generalises the exponential distribution, and it is probably the most widely used parametric distribution in survival analysis. This may be because it covers a wide variety of distributional shapes with simple survival and hazard functions, which has been found to be useful in many contexts.

For non-parametric survival function estimation, the Kaplan-Meier method

(Kaplan and Meier, 1958) is often used. The estimator is computed by

$$\hat{S}(t) = \prod_{i:t_i \leq t} \left(1 - \frac{d_i}{n_i}\right), \tag{4.1}$$

where t_i is the time when at least one event happened, d_i is the number of events (e.g. deaths) that happened at time t_i, and n_i is the number of individuals that have not yet had an event or been censored up to time t_i.

One of the most popular regression models to analyse survival data is *Cox's proportional hazard model* (Cox, 1972). Suppose there are p explanatory variables, denoted by vector z, which may affect the time until an event. Standardising these explanatory variables with $z = 0$, we will obtain some standard set of conditions, which is called baseline hazard shown in formula below. In reality, such standard set can be the control group in a clinical trial. Cox's model proposes that the hazard rates of individuals are related via the relationship

$$\lambda(t|z) = \lambda_0(t)\exp(\beta \cdot z), \tag{4.2}$$

where $t > 0$, $\beta \in \mathbb{R}^p$ is a vector of regression parameters, and $\lambda_0(t)$ is the baseline hazard representing the hazard of a individual with $z = 0$. Equation (4.2) also shows a multiplicative effect on the hazard of any deviation away from zero in each explanatory variable.

The reason why the model is called proportional hazard regression is because of the constant ratio of hazard functions of two individuals with explanatory variable z_1 and z_2 at all times, that is,

$$\frac{\lambda(t|z_1)}{\lambda(t|z_2)} = \frac{\lambda_0(t)\exp(\beta \cdot z_1)}{\lambda_0(t)\exp(\beta \cdot z_2)} = \exp\{\beta \cdot (z_1 - z_2)\}. \tag{4.3}$$

The original Cox's proportional hazard regression only collects the data for the first event, whereas the later events are ignored. More appropriate models for recurrent events, which are generalizations of the Cox's proportional hazard regression, are the Andersen and Gill (AG) model (Andersen and Gill, 1982), the Wei, Lin and Weissfeld (WLW) marginal model (Wei et al., 1989), the Prentice, Williams and Peterson (PWP) conditional model (Prentice et al., 1981), and the frailty models (McGilchrist and Aisbett, 1991), whose hazard functions are shown later in Table 4.1.

The AG model is a counting process model, which assumes that the recurrent events for individuals are independent and have the same baseline hazard, $\lambda_0(t)$, for all individuals. Although this method can be used to evaluate repeated occurrence of hospitalizations for all individuals, the assumptions may be too strong in practice.

The WLW model employs total time and assumes a event-specific baseline hazard. Therefore, each individual is considered to be at risk of all recurrent events from the start of the observation period, which means that each recurrence is regarded as a separate process and there is no ordering for all events within each individual. This model will give reliable estimates when data do not have any ordering. A drawback of the WLW method is that it takes no account of the strong information contained in the order of the recurrent survival lifetimes, just their values. Other methods that do take account of the order information have the potential to do better.

Prentice, Williams and Peterson proposed two models. One is a counting process model, called PWP-CP, the other is a gap time (PWP-GT) model, both of which assume that the recurrent events within each individual are related and the baseline hazard always changes from event to event. For instance, if a person had a bladder cancer and has been cured by a treatment, the risk for the cancer to reappear may increase, as the body may be damaged during the treatment. The only difference between these two models is that the former is based on a counting process time interval, but the latter one is based on a gap time interval, which is the time since the previous event. However, (Yadav et al., 2018) finds that, for higher order events, the estimates may be unreliable for both models, as the number of individuals at risk decreases.

The frailty model (McGilchrist and Aisbett, 1991) is a random effects model, which considers the excess risk or frailty for distinct individuals caused by the unknown factors that are not listed in the observed explanatory variables. The assumption of the baseline hazard allows it to vary within each individual as the heterogeneity is directly incorporated via the random effect (frailty). This model can be used both for modelling lifetimes of individuals, like twins or family members, and for repeated events for the same individual. The most common frailty model assumes the random effects follow a gamma distribution with mean equal to one and unknown variance, but this may be a restriction implying that the dependence is most important for late events (Hougaard, 1995).

We remind the reader that $\lambda_0(t)$ in Equation (4.2) represents the common baseline hazard for the regression model and $\beta \in \mathbb{R}^p$ is a vector of regression

parameters. For each individual, suppose there are K recurrent events, then t_{k-1} with $k = 1,2,...,K$ is the time for the k-th event occurring. Let λ_{ik} denote the hazard function for the k-th event of the i-th individual at time t, hence λ_{0k} represents an event-specific baseline hazard for the k-th event. Let $z_{ik}(t) \in \mathbb{R}^p$ denote the covariate vector for the i-th individual with respect to the k-th event. Also, Y_i represents the random effect (frailty term) for the i-th individual in the frailty model. Table 4.1 summarizes the hazard functions for the regression models mentioned previously.

Regression model	Hazard function
AG model	$\lambda_{ik}(t\|z_{ik}) = \lambda_0(t)\exp(\beta \cdot z_{ik})$
WLW model	$\lambda_{ik}(t\|z_{ik}) = \lambda_{0k}(t)\exp(\beta \cdot z_{ik})$
PWP-CP model	$\lambda_{ik}(t\|z_{ik}) = \lambda_{0k}(t)\exp(\beta \cdot z_{ik})$
PWP-GT model	$\lambda_{ik}(t\|z_{ik}) = \lambda_{0k}(t - t_{k-1})\exp(\beta \cdot z_{ik})$
frailty model	$\lambda_{ik}(t\|z_{ik}) = \lambda_{0k}(t)Y_i \exp(\beta \cdot z_{ik})$

Table 4.1: Hazard function for different regression models

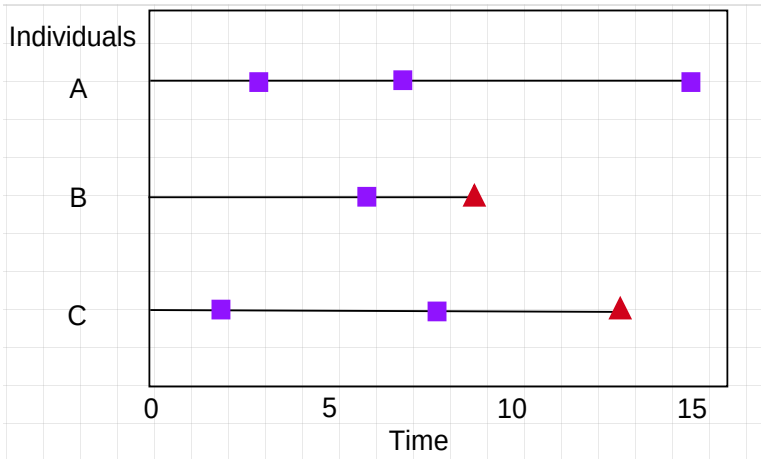
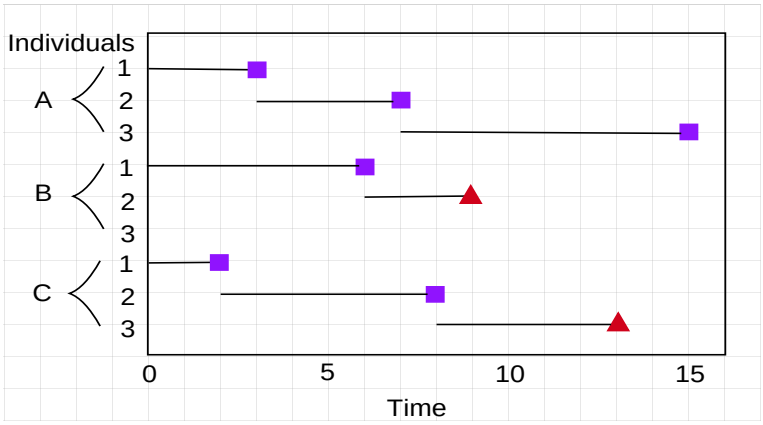
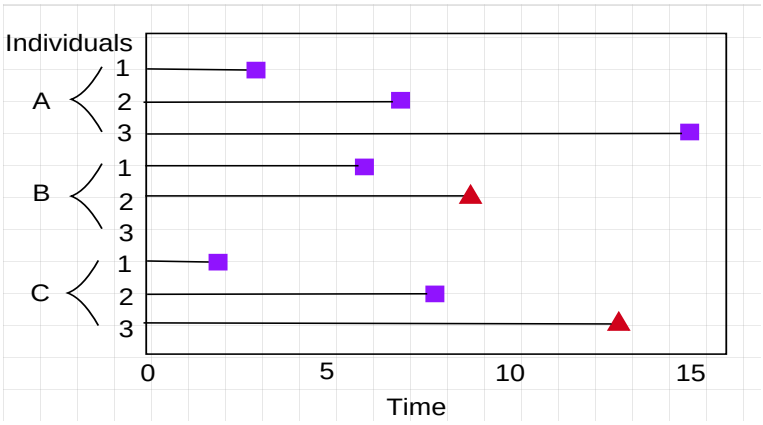


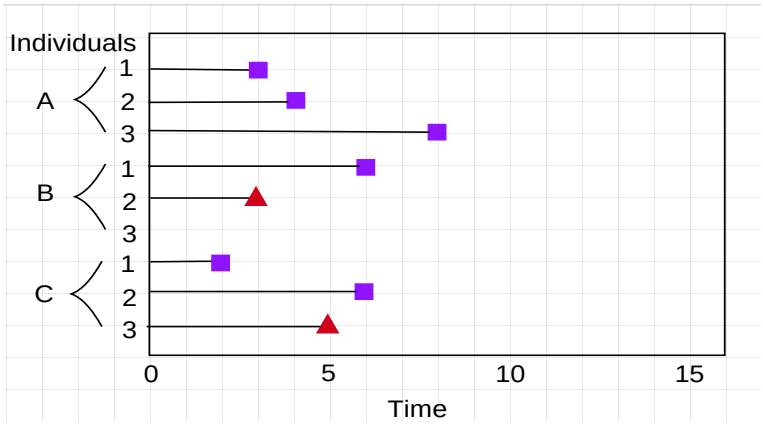
Figure 4.1: Illustrative recurrent times for three individuals, where a square represents an event and a triangle means that the individual is right-censored.



(a) Counting Process



(b) Total Time



(c) Gap time

Figure 4.2: Illustrations of the risk intervals: a) counting process b) total time c) gap time, derived from the times of the individuals in Figure 4.1. A Square represents an event and a triangle means that the individual is right-censored.

As mentioned before, the AG and PWP-CP are counting process models, the WLW model uses total time, whereas the PWP-GT uses a gap or waiting-time scale. Figure 4.1 shows the recurrent lifetimes (e.g. recurrence of cancer) for three individuals A, B, C, whereas Figure 4.2 shows three types of intervals (i.e. counting process, total time and gap time intervals) for the same three individuals in Figure 4.1. For example, Figure 4.1 shows that individual A has three event times at $t = 3, 7, 15$ respectively. The corresponding time intervals shown in Figure 4.2 for the counting process are $(0,3],(3,7],(7,15]$, which are $(0,3],(0,7],(0,15]$ and $(0,3],(0,4],(0,8]$ when using the total time and gap time respectively. These types of intervals are considered as the risk intervals in the survival analysis, which is defined when an individual is at risk of having an event along a given time scale.

4.2.2 Dissimilarity matrix and hierarchical cluster analysis

The relationships between individuals are often considered in data analytic and data mining tasks, which can be represented numerically using the idea of similarities or dissimilarities. The similarity between two objects is a numerical measure of the degree to which the two individuals are alike, which is non-negative and often between 0 and 1, representing no similarity and complete similarity respectively. By contrast, the dissimilarity measure describes pairwise distinction between individuals. Intuitively, the greater distinction between two individuals, the larger the value of the measure of dissimilarity.

The dissimilarity matrix is always a square matrix with the diagonal entries set to zero. The non-diagonal entries represent pairwise dissimilarities between two individuals, which can be metric or non-metric as described next. Here, we only consider the metric dissimilarity matrix as follows. A metric dissimilarity function, d, maps any two individuals, x_i and x_j, into a real number, which satisfies the following three properties (Mendelson, 1990):

1. d is non-negative, that is

$$d(x_i, x_j) > 0 \text{ if and only if } x_i \neq x_j,$$
$$d(x_i, x_j) = 0 \text{ if and only if } x_i = x_j;$$

2. d is symmetric, that is,

$$d(x_i, x_j) = d(x_j, x_i);$$

3. d satisfies the triangle inequality, that is for any individuals x_i, x_j, x_k,

$$d(x_i, x_j) \leq d(x_i, x_k) + d(x_k, x_j).$$

In our work, we only consider metric dissimilarity matrices. Commonly used metrics to compute the dissimilarities for numerical data are the Minkowski distance, Manhattan distance, Euclidean distance, maximum distance, and Mahalanobis distance (Mahalanobis, 1936), while the Hamming distance (Hamming, 1950), Jaccard distance (Jaccard, 1912) and simple matching similarity are suitable for the categorical data. For mixed data, with both numerical and categorical covariates, Gower's similarity (Gower and Legendre, 1986) are quite often considered in practice. We explain these below.

Suppose x_i and x_j are two individuals with p covariates/attributes, Table 4.2 summarizes the formulae for various metric distances, where we find the Manhattan and Euclidean distances are special cases of the Minkowski distance with $r = 1, 2$ respectively, and the Minkowski distance is known as the maximum distance when $r = \infty$. If there are correlations of the attributes between individuals, the Mahalanobis distance is often preferred to other metrics (Mclachlan, 1999).

Metric distance	Equation of formula
Minkowski Distance	$d(x_i, x_j) = \left(\sum_{k=1}^{p} \lvert x_{ik} - x_{jk} \rvert^r \right)^{1/r}$
Manhattan Distance	$d(x_i, x_j) = \sum_{k=1}^{p} \lvert x_{ik} - x_{jk} \rvert$
Euclidean Distance	$d(x_i, x_j) = \left(\sum_{k=1}^{p} \lvert x_{ik} - x_{jk} \rvert^2 \right)^{1/2}$
Maximum Distance	$d(x_i, x_j) = \max_{k=1}^{p} \lvert x_{ik} - x_{jk} \rvert$
Mahalanobis Distance	$d(x_i, x_j) = \left\{ (x_i - x_j) \Sigma^{-1} (x_i - x_j)^T \right\}^{1/2}$

Table 4.2: Formulae for the metric distances: x_{ik} and x_{jk} are the k-th attributes of x_i and x_j respectively and Σ^{-1} is the inverse of the covariance matrix of the data.

For categorical data, the dissimilarity function, d, for the k-th attributes of two individuals x_i and x_j , can be

$$d(x_{ik}, x_{jk}) = \begin{cases} 0 & \text{iff } x_{ik} = x_{jk} \\ 1 & \text{otherwise,} \end{cases} \tag{4.4}$$

Hence, the Hamming distance can be defined as the number of differences in

the binary attributes, such that

$$d(x_i, x_j) = \sum_{k=1}^{p} d(x_{ik}, x_{jk}), \tag{4.5}$$

which is equivalent to the definition of Manhattan distance.

For the individuals x_i and x_j and $a, b, c, d \geq 0$, we define

- a to be the number of times that x_i and x_j both share an attribute,
- b to be the number of times that x_i does not have an attribute but x_j does,
- c to be the number of times that x_i have an attribute but x_j does not,
- d to be the number of times that x_i and x_j do not have an attribute.

Then the Jaccard distance (Jaccard, 1912) is

$$d(x_i, x_j) = \frac{b + c}{a + b + c}, \tag{4.6}$$

and the simple matching similarity, $s(x_i, x_j)$, is

$$s(x_i, x_j) = \frac{a + d}{a + b + c + d}. \tag{4.7}$$

For mixed data, Gower's similarity (Gower and Legendre, 1986) is defined as

$$s_G(x_i, x_j) = \frac{\sum_{k=1}^{p} w_k \delta(x_{ik}, x_{jk})}{\sum_{k=1}^{p} w_k}, \tag{4.8}$$

where $w_k \geq 0$ quantifies the confidence/weight in variable k, and $\delta(x_{ik}, x_{jk})$ is

the similarity between x_i, x_j on variable k.

For categorical data,

$$\delta(x_{ik}, x_{jk}) = 1 - d(x_{ik}, x_{jk}), \tag{4.9}$$

with $d(x_{ik}, x_{jk})$ defined in Equation (4.4).

For numerical data,

$$\delta(x_{ik}, x_{jk}) = 1 - \frac{x_{ik} - x_{jk}}{r_k}, \tag{4.10}$$

with r_k equal to the range of values on the k-th variable.

The Gower's dissimilarity is simply defined as $\sqrt{1 - s_G(x_i, x_j)}$.

4.3 New Methods for Analysing Recurrent Events using Clusters

This section describes our new methods using cluster structures to analyse recurrent lifetimes, where covariate information is also used. Basically, suppose there are n individuals. We construct the cluster structure for these individuals with the help of a dissimilarity matrix. In the situation when the dissimilarity matrix or the distance between pairs of individuals is unknown, we compute it based on the given covariates. Using hierarchical cluster analysis, we then merge the individuals into clusters according to the closeness between individuals, which can be presented as a dendrogram. Then we group the recurrent lifetimes of each individual, where we combine the individuals of the similar

survival/hazard functions into the same cluster. As the number of the recurrent lifetimes for each individual is usually not numerous, with this method, we can have more recurrent times to estimate the survival functions, by assuming individuals similar survival/hazard functions could be grouped together, as stated on page 100.

As mentioned previously, our methods could be applied in biological research. For example, the survival estimates of epilepsy, which is a recurrent chronic noncommunicable disease of the brain. According to the WHO, there are around 50 million people worldwide who have suffered from this disease. Usually, these people tend to have more physical and psychological problems such as a brain tumour, brain malformations etc., which could be used in our methods to construct the dissimilarity matrix and produce dendrograms. However, using these characteristics only to group patients may be a bit inaccurate, so based on the dendrograms, recurrent lifetimes could be used to carry out goodness-of-fit tests to adjust the initial clusters shown on the dendrograms. Because of the low recurrence of this disease, it is usually difficult to estimate survival curves individually. Using all lifetimes within a cluster to conduct analysis may provide a more accurate estimate. Other similar examples, which have few recurrent data individually, are also suitable to use our proposed methods, as our main goal is to group lifetimes reasonably to increase the number of data to make estimation. The details of our methods will be illustrated below.

To analyse clustered measurements, mixed models are also popular alternatives, which contain both fixed effects and random effects. Clayton (1994) reviews different approaches to analyse recurrent event data, including the

generalised linear mixed model. Sun et al. (2011) propose a class of mixed models for recurrent event data and give a clinic study on chronic granulomatous disease. Rathbun and Shiffman (2018) develop a mixed effects version of a recurrent events model to describe variation among smokers in how they respond to some time-varying covariates. The main difference between our model and a mixed effect model is that ours combines all possible factors to compute the dissimilarities between individuals, which are used to construct initial clusters. Then, the initial clusters will be adjusted using recurrent lifetimes to obtain the final clusters. Hence, all factors are not directly included in the model. On the other hand, mixed models include each factor directly, either in fixed or random effects, which gives information on the influence of individual factors.

4.3.1 Initial clusters construction

For different types of data, choosing one of the methods listed in section 4.2.2 to compute the dissimilarity matrix, we can then use the complete-linkage clustering method (Everitt et al., 2011) and display the clusters using a dendrogram, which shows the hierarchical relationship between individuals. At the beginning of the complete-linkage clustering process, each element is a cluster, which is then sequentially combined into larger clusters until all elements being in one cluster finally. At each step, the two clusters with the shortest distance are combined. Suppose C_1, C_2 are two clusters and c_1, c_2 are any individuals in these two clusters respectively, the distance between these two clusters $D(C_1, C_2)$ is defined as

$$D(C_1, C_2) = \max_{c_1 \in C_1, c_2 \in C_2} d(c_1, c_2), \tag{4.11}$$

where $d(c_1, c_2)$ is the distance between the individuals c_1 and c_2.

In practice, we use `daisy` function in R `cluster` package (Maechler et al., 2021) to compute the pairwise dissimilarities with Euclidean (the default), Manhattan and Gower metrics. Some other functions are also available in R, for example, function `distance` in `philentropy` package (Drost, 2018) is able to compute 46 different distances or similarities, which includes all the methods mentioned in this section. To apply the complete-linkage clustering method in R software, we use `hclust` function in package `stats`.

4.3.2 Clusters adjustment

After obtaining the dendrogram of the individuals constructed according to their covariates by the hierarchical clustering method, we then adjust the clusters using the *recurrent survival times* for each individual. Basically, we carry out a non-parametric goodness-of-fit test, the Kolmogorov-Smirnov test (Massey, 1951), for the recurrent lifetimes to compare whether the two samples of individuals from different clusters are actually from the same distribution.

Here, we use the two-sample Kolmogorov-Smirnov (K-S) test to check whether two underlying probability distributions differ, which can be applied in R software by using function `ks.test` in package `stats`. The Kolmogorov-Smirnov

test statistic is defined as

$$D_{n,m} = \sup_x |\hat{F}_{1,n}(x) - \hat{F}_{2,m}(x)|, \tag{4.12}$$

where sup is the supremum function, and $\hat{F}_{1,n}$, $\hat{F}_{2,m}$ are the empirical distribution functions of the two samples with size n and m respectively. The null hypothesis for the test is that both samples come from a population with the same underlying distribution. Hence, we reject the null hypothesis at significance level α if

$$D_{n,m} > D_{n,m,\alpha}, \tag{4.13}$$

where $D_{n,m,\alpha}$ is the critical value.

Loosely speaking, using the Kolmogorov-Smirnov test, our two methods of adjusting the existing clusters based on the dendrogram are as follows.

Method 1 Top-down adjustment: Assuming all the individuals should not be grouped in the same cluster, we separate them into two clusters using the existing dendrogram. For each cluster, at each step, we check whether the two children of each cluster should be separated or not using the Kolmogorov-Smirnov test including the recurrent lifetimes of all individuals in each branch. The process stops when the further division is not possible.

Method 2 Bottom-up adjustment: Assuming every individual is a cluster, we then check whether the current cluster is able to combined with the cluster next to it according to the dendrogram. If we do not have the

enough evidence to reject the null hypothesis of the Kolmogorov-Smirnov test, we group the clusters together. The process stops when further combination is impossible.

We now propose one algorithm for the top-down approach and two algorithms for the bottom-up approach defined next.

4.3.3 Clusters adjusting algorithms

The following algorithms illustrate in detail how our methods work practically. The top-down adjustment explained in Algorithm 4.1 on page 116 applies a recursive function to divide all individuals into clusters. For the bottom-up adjustment, Algorithm 4.2 on page 117 considers whether the consecutive individuals of the dendrogram should be clustered one by one. Hence, for a number of n individuals, we start with $n-1$ pairwise Kolmogorov-Smirnov (K-S) tests. However, sometimes Algorithm 4.2 may result in too many small clusters. Hence, we introduce Algorithm 4.3 on page 117, which attempts some improvement, where further combinations of the Algorithm 4.2 existing clusters are possible by conducting Kolmogorov-Smirnov tests for all consecutive clusters again. Larger clusters are made from the smaller ones, and the process stops until no more larger clusters can be made. For all three algorithms, the inputs and output are listed as follows.

Input:

- $vec.t$: a vector of recurrent lifetimes for all individuals
  ordered from the left of the dendrogram
- $dend$: the current dendrogram
- $in.list$: a list of input clusters of individuals
- sig: the significance level set for the K-S test

Output:

- $out.list$: a list of output clusters of individuals

Algorithm 4.1: Top-down Adjustment

Function `TopDown.adjust`($vec.t$, $dend$, sig)**:**

- **if** *members of dendrogram* ≥ 2 **then**
  - Cut $dend$ into two branches (clusters) at its highest height.
  - Use the K-S test on the two clusters and compute the p-value.
  - **if** *p-value* < *sig* **then**
    - For each branch, run `TopDown.adjust`($vec.t$, $dend$, sig)
    - **return**
  - **else**
    - Add the two separate clusters into $out.list$
    - **return**
- **else**
  - Add the individual as a cluster into $out.list$
  - **return**

Algorithm 4.2: Bottom-up Adjustment (Method One)

Let each individual ordered from left of $dend$ to be an input cluster.

Function `BottomUp.adjust1(`*vec.t, in.list, sig*`)`**:**

- Define n to be the number of clusters of $in.list$.
- Let the current cluster be the first cluster of $in.list$.
- Let $out.list$ be an empty list.
- **for** $i = 1, 2, ..., n-1$ **do**
  - Use K-S test to compute the p-value of the recurrent times between the i-th and $(i+1)$-th clusters of $in.list$.
  - **if** *p-value* > *sig* **then**
    - Include the $(i+1)$-th cluster into the current cluster.
    - **if** $i = n-1$ **then**
      - Add the current (last) cluster into $out.list$.
  - **if** *p-value* < *sig* **then**
    - Add the current cluster into $out.list$.
    - **if** $i = n-1$ **then**
      - Add the $(i+1)$-th cluster into $out.list$.
    - **else**
      - Set the current cluster to be the $(i+1)$-th cluster.
- **return**

Algorithm 4.3: Bottom-up Adjustment (Method Two)

Function `BottomUp.adjust2(`*vec.t, in.list, sig*`)`**:**

- Let $in.list$ be an empty list.
- Let $out.list$ be the individual clusters ordered from left of $dend$.
- **while** $out.list \neq in.list$ **do**
  - $in.list = out.list$
  - Run `BottomUp.adjust1(`*vec.t, in.list, sig*`)` and obtain $out.list$.
- **return**

4.3.4 Improve the survival estimates

Our main goal is, of course, to obtain good estimates of our survival functions. After obtaining clusters according to the dendrogram obtained by the hierarchical clustering method, we can then estimate the survival/hazard function for each cluster by using *all* lifetimes in that cluster. This may provide better estimation as we have more recurrent survival times to analyse for each cluster, which may not be possible if we estimate the survival/ hazard function using the recurrent lifetimes for each individual separately. Sometimes, the number of observations for each individual is fewer than ten in reality, for example with the bladder cancer recurrent lifetime data (Byar, 1980). Hence, it can be hard to get accurate estimates for these individuals.

4.4 Weibull Distribution Simulation Examples

This section contains simulation examples, where the recurrent lifetimes for each individual are generated from different Weibull distributions. For simplicity, we only produce uncensored lifetimes. Treating the two parameters of each Weibull distribution as the geographical coordinates for the corresponding individual, we can obtain a network of clustered individuals and compute a Euclidean distance matrix. Using the algorithms mentioned in section 4.3.3, our aim is to classify the clusters according to the dendrogram produced by the distance matrix, and the recurrent lifetimes of individuals. The individuals in the same clusters share the same distribution, we can obtain the estimates of the survival/hazard functions from them. We will then compare our method with the regular (generalised) Cox's Proportional Hazard regression method in

terms of mean squared error performance.

Our Weibull simulation setup is biased in favour of our approach as we use covariates as the parameters to generate the recurrent lifetimes. However, even when the covariates have an indirect effect on the recurrent lifetimes, we still believe our method will be useful in getting information out. In real life, there are cases where the covariates do not influence the recurrent lifetimes. In that case, our method will not extract good estimators, but other methods that rely on the covariates to get better estimators, such as the (generalised) Cox's Proportional Hazard regression, will not necessarily work well either.

4.4.1 Data generation

The situation we focus on is when the number of recurrent lifetimes for each individual is small. Hence, in this example, we generate 70 individuals each with eight event times from different Weibull distributions. The individuals are designed to be clustered into six groups according to their values of their Weibull distribution parameters. Let $\eta > 0$, $\alpha > 0$ be the shape and scale parameters of the Weibull distribution, respectively, for the lifetime T, the probability density $f(t)$, survival $S(t)$ and hazard $\lambda(t)$ functions are:

$$f(t) = \eta\alpha^{-\eta}t^{\eta-1}\exp\{-(t/\alpha)^{\eta}\}, \tag{4.14}$$

$$S(t) = \exp\{-(t/\alpha)^{\eta}\}, \tag{4.15}$$

$$\lambda(t) = \eta\alpha^{-\eta}t^{\eta-1}. \tag{4.16}$$

When generating the lifetimes and distance matrix for the individuals, we follow the steps below:

- Divide a circle in the Cartesian plane into six equal parts, and simulate six cluster centers with associated x and y coordinates. (See Figure 4.3)
- Simulate x and y coordinates of the individuals in each cluster by defining the distance between each cluster center and its individuals, to follow a zero-mean normal distribution. The standard deviation of the normal distribution affects the tightness of the clusters. A large value will result in a dispersed distribution of the individuals around the cluster centre and vice versa. In the example, demonstrated later, we set the standard deviation to be 0.3. The number of individuals in each cluster is random: 70 in total.
- Simulate eight recurrent lifetimes for every individual using a Weibull distribution. The shape and scale parameters chosen for the individual are their geographical coordinates in the $x - y$ plane.

Once have obtained the x and y coordinates for all individuals, then we can directly compute the distance matrix, and use it to produce a dendrogram. Figure 4.3 shows the geographical locations/network for all individuals, where different clusters are assigned different colours. Figure 4.4 is the dendrogram produced using the complete-linkage clustering method (Everitt et al., 2011), which does not know the true clusters.

4.4.2 Cluster classification

We then apply the three algorithms mentioned in section 4.3.3 to this particular dataset and produce the dendrograms shown in Figure 4.5 and 4.6, where the recurrent lifetimes of individuals are used to classify the new clusters

represented by different colours. When applying the three algorithms, we set the significance level to be 0.05 in the K-S tests. The dendrograms in Figure 4.5 and 4.6 show that the new clusters produced do not exactly match the clusters we designed when generating data.

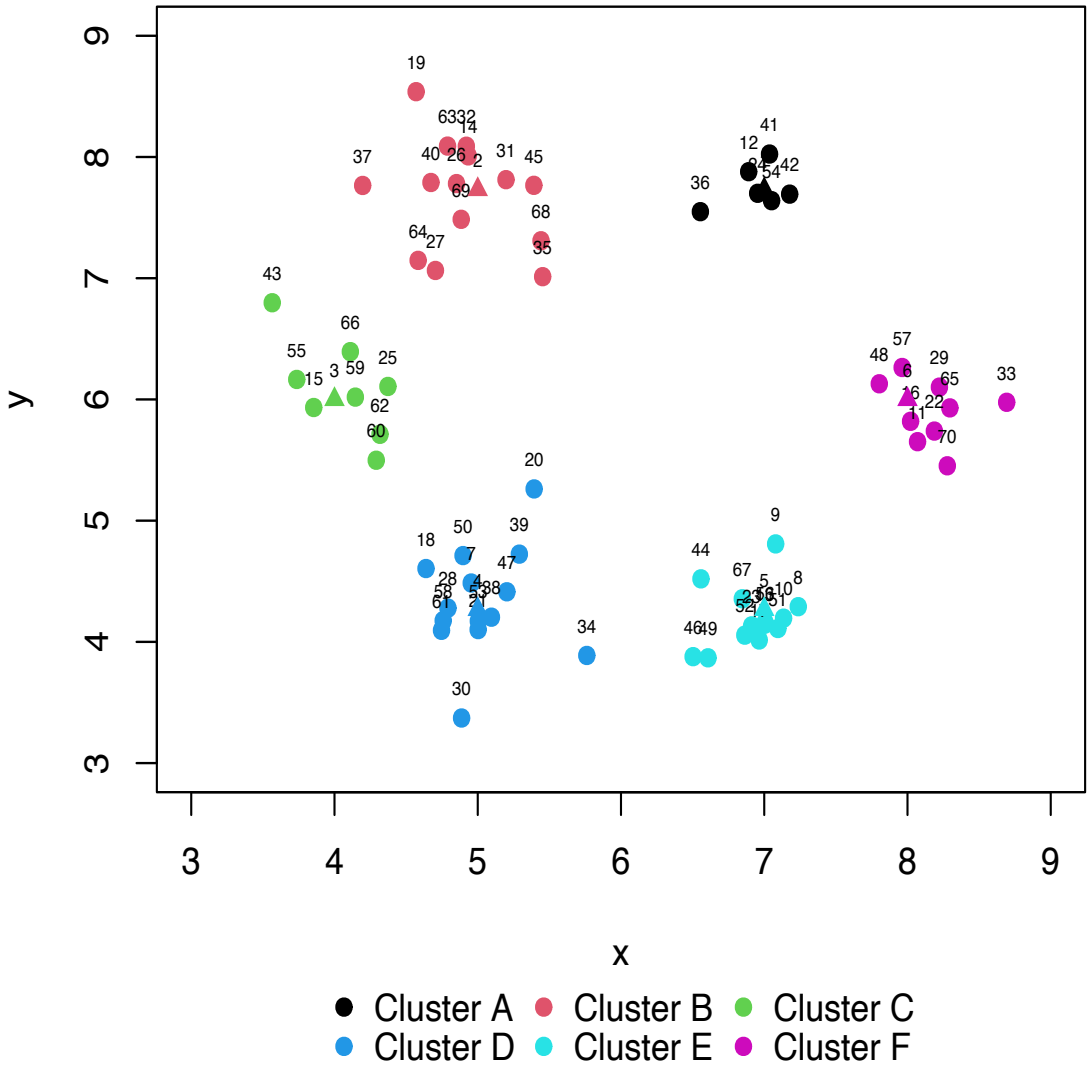


Figure 4.3: Geographical locations for 70 individuals, where triangles represent the cluster centers.

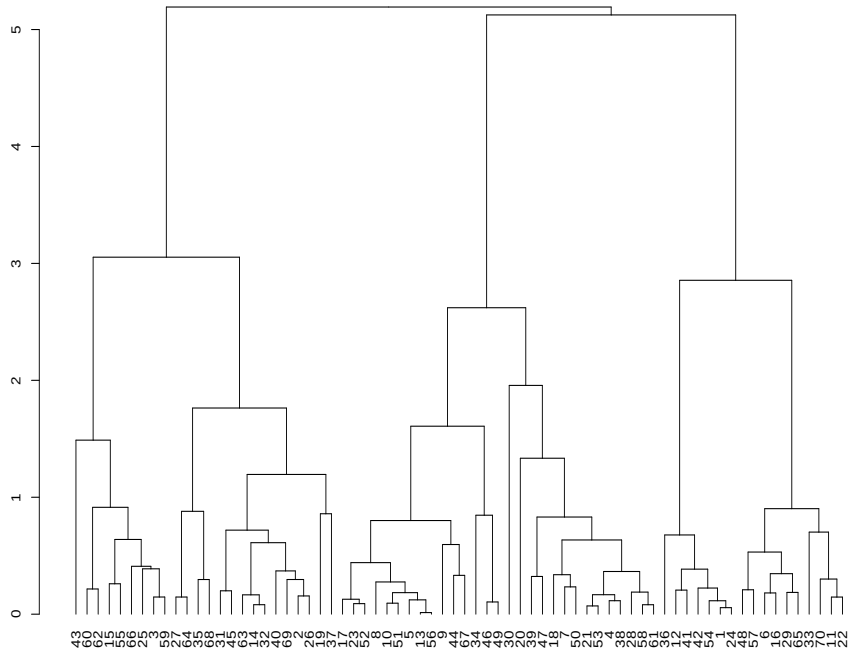


Figure 4.4: Dendrogram produced for the 70 individuals under the complete-linkage clustering method.

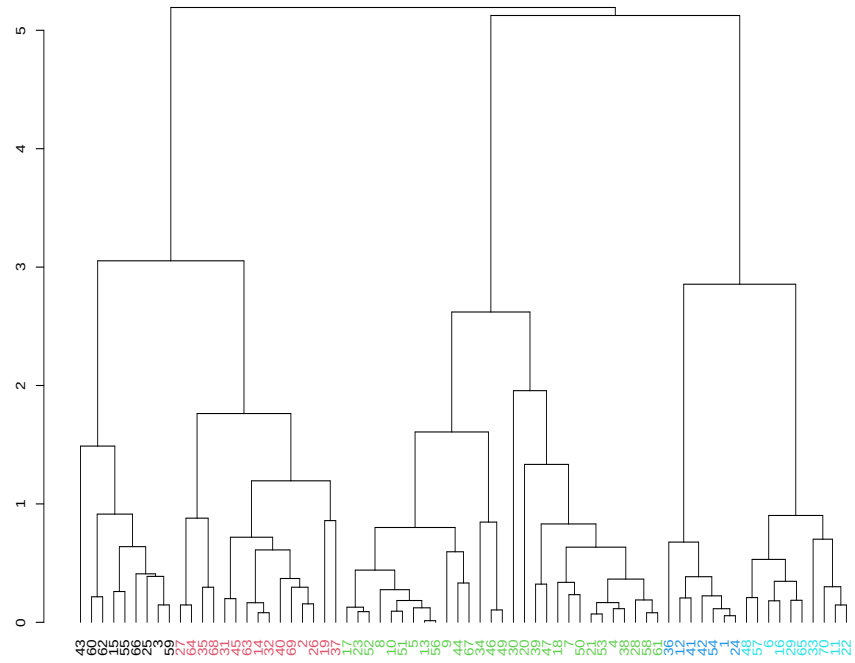
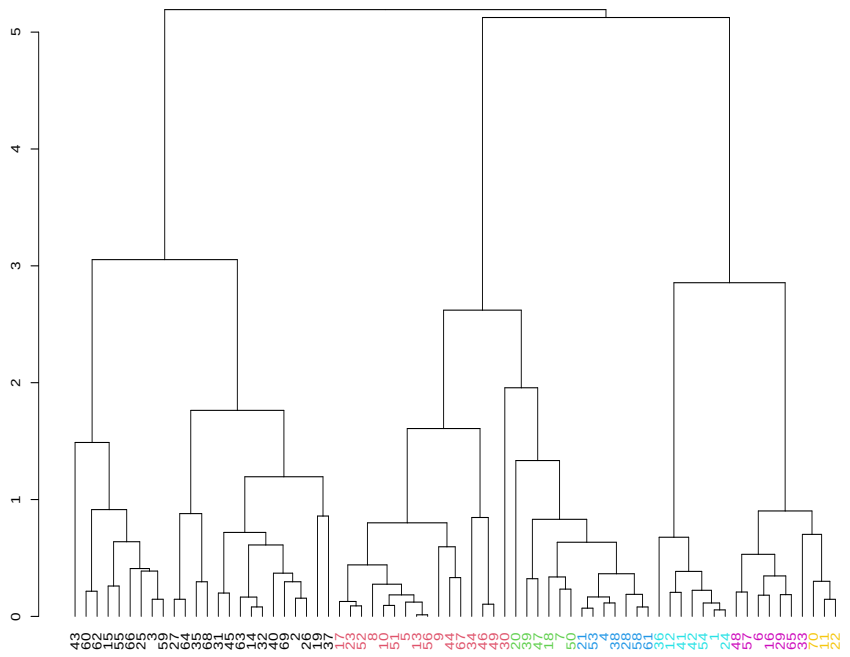
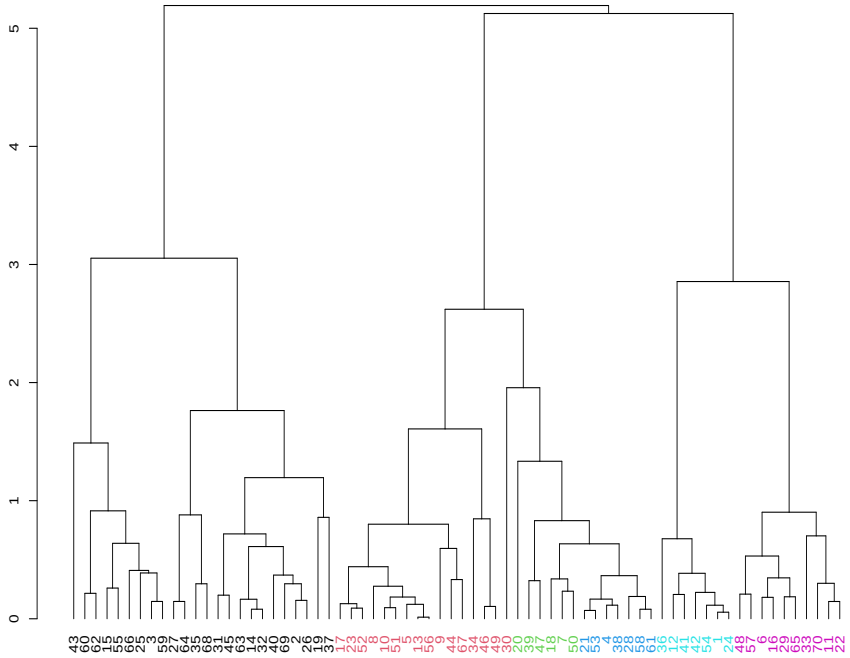


Figure 4.5: Dendrogram produced using the top-down adjustment algorithms, where different colours represent different clusters



(a) Algorithm 4.2



(b) Algorithm 4.3

Figure 4.6: Dendrogram produced using the bottom-up adjustment algorithms, where different colours represent different clusters

From Figure 4.5, we can see that Algorithm 4.1 groups Clusters D and E in Figure 4.3 together. This may be because it is difficult to choose the significance level to cluster the individuals exactly as they were designed. Also, some outliers of Cluster D (e.g. individual numbered 34) may be influence the clustering results.

The problem with the bottom-up method is that sometimes well-separated individuals are clustered. For example, Figure 4.6 shows that individual 30 is grouped into Cluster E, which is far from that cluster center. This may be because when conducting the K-S tests pairwise, we try to include an additional cluster to the current cluster, which eventually causes the K-S tests performed between one very large cluster (current cluster) and one relatively small cluster (the additional cluster). This may lead to increased Type II error.

Comparing with two methods, the top-down adjustment seems more coincident with the designed clusters. For the bottom-up adjustment, Algorithm 4.3 usually combines some small clusters in dendrogram produced under Algorithm 4.2 together. For example, Figure 4.6 (b) shows that Algorithm 4.3 groups the last two clusters in Figure 4.6 (a). We also compute the contingency tables as well as the adjusted Rand index values (Rand, 1971) to compare the performance of estimated clusters for these three proposed algorithms. The results show that our top-down approach has relatively higher similarities with a score of 0.7412 between the true and estimated clusters for this particular data set, which agrees with our guess from the dendrograms obtained in Figure 4.5 and Figure 4.6. The next step is to estimate the survival functions for all three algorithms using the recurrent lifetimes of clustered individuals.

		True Clusters						
		A	B	C	D	E	F	Sum
Estimated Clusters	1	0	0	9	0	0	0	9
	2	0	15	0	0	0	0	15
	3	0	0	0	14	15	0	29
	4	7	0	0	0	0	0	7
	5	0	0	0	0	0	10	10
	Sum	7	15	9	14	15	10	70
Adjusted Rand Index		0.7412						

Table 4.3: Contingency table and adjusted Rand index produced using the top-down approach explained in Algorithm 4.1.

		True Clusters						
		A	B	C	D	E	F	Sum
Estimated Clusters	1	0	15	9	0	0	0	24
	2	0	0	0	1	15	0	16
	3	0	0	0	6	0	0	6
	4	0	0	0	7	0	0	7
	5	7	0	0	0	0	0	7
	6	0	0	0	0	0	7	7
	7	0	0	0	0	0	3	3
	Sum	7	15	9	14	15	10	70
Adjusted Rand Index		0.6865						

Table 4.4: Contingency table and adjusted Rand index produced using the bottom-up approach explained in Algorithm 4.2.

		True Clusters						
		A	B	C	D	E	F	Sum
Estimated Clusters	1	0	15	9	0	0	0	24
	2	0	0	0	1	15	0	16
	3	0	0	0	6	0	0	6
	4	0	0	0	7	0	0	7
	5	7	0	0	0	0	0	7
	6	0	0	0	0	0	10	10
	Sum	7	15	9	14	15	10	70
Adjusted Rand Index		0.7210						

Table 4.5: Contingency table and adjusted Rand index produced using the bottom-up approach explained in Algorithm 4.3.

As all individuals are generated from different Weibull distributions with known parameters (geographical locations), we can compare the estimated parameters of the clustered individuals for three algorithms in section 4.3.3 by the following steps.

- According to the new clusters produced by our algorithms, we combine the recurrent lifetimes for the individuals in the same new cluster, as we assume these individuals are from the same distribution.
- Using R function `fitdist` in the package `fitdistrplus` (Delignette-Muller and Dutang, 2015), we estimate the shape and scale parameters in the Weibull distribution based on these combined recurrent lifetimes.
- We compute the mean squared error for each of these two parameters. The parameter errors are calculated by taking difference between the actual individual parameters generated and the estimates.

Figures 4.5 and 4.6 show that our three algorithms produce five, seven and six new clusters respectively for this particular dataset. Table 4.6 displays the shape and scale parameters (i.e. the geographical locations) of the cluster centers shown in Figure 4.3. As individuals in the same clusters share the same Weibull distribution, the "true" parameters of the Weibull distribution for each cluster are represented by these cluster centers. Tables 4.7 shows the parameter estimates for each new cluster. The results of Tables 4.6 and 4.7 are plotted in Figure 4.7.

Table 4.8 presents the mean squared errors for different algorithms, where

we find that the two algorithms using bottom-up approach have similar mean squared errors for both parameters, as plots in Figure 4.6 show that the first five clusters produced are the same. Algorithm 4.1 has a larger mean square error in shape estimate. This is because Algorithm 4.1 combines two horizontal clusters D and E in Figure 4.3, which are well separated. On the other hand, the scale estimate produced by Algorithms 4.2 and 4.3 have a greater mean square error, as these two algorithms combines the two left-top clusters (i.e. B and C in Figure 4.3) together.

	Cluster A	Cluster B	Cluster C	Cluster D	Cluster E	Cluster F
Shape	7.00	5.00	4.00	5.00	7.00	8.00
Scale	7.73	7.73	6.00	4.27	4.27	6.00

Table 4.6: True parameters of Weibull distributions for the original clusters.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Shape	4.39	4.62	5.43	7.18	7.77
Scale	6.21	7.48	4.29	7.72	5.91

(a) Estimates computed by Algorithm 4.1

	Cluster1	Cluster2	Cluster3	Cluster4	Cluster5	Cluster6	Cluster7
Shape	4.27	6.36	4.82	5.30	7.18	7.91	8.73
Scale	7.04	4.11	4.78	4.23	7.72	6.03	5.59

(b) Estimates computed by Algorithm 4.2

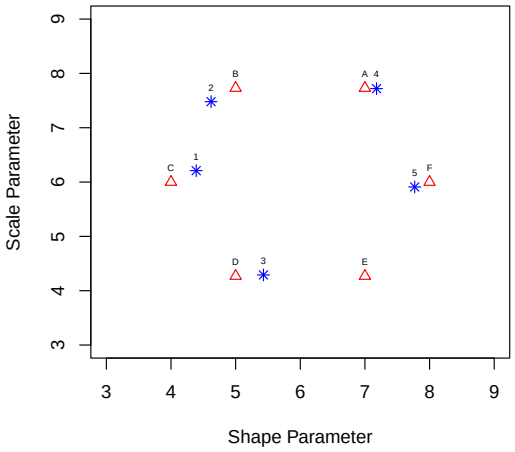
	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Shape	4.27	6.36	4.82	5.30	7.18	7.77
Scale	7.04	4.11	4.78	4.23	7.72	5.91

(c) Estimates computed by Algorithm 4.3

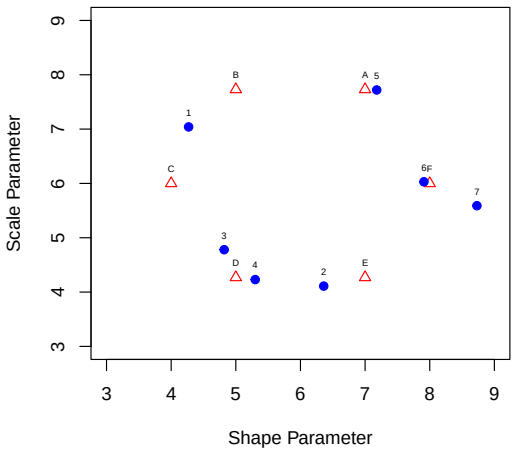
Table 4.7: Estimated parameters for Weibull distributions using different algorithms explained in section 4.3.3, which coincide with points in Figure 4.7 .

	Algorithm 1	Algorithm 2	Algorithm 3
Shape	0.781	0.453	0.436
Scale	0.123	0.298	0.304

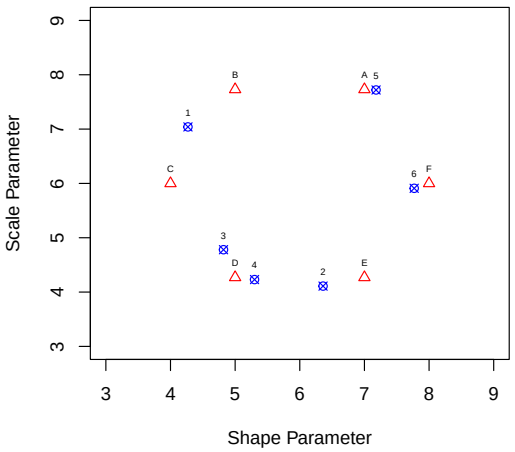
Table 4.8: Mean square errors of the two estimates in Weibull distribution produced by different algorithms.



(a) Algorithm 4.1



(b) Algorithm 4.2



(c) Algorithm 4.3

Figure 4.7: Geographical location (i.e. parameters) of the cluster centers (red triangles) shown in Table 4.6 and the parameter estimates (blue symbols) in Table 4.7, which are computed using algorithms explained in section 4.3.3.

In terms of the mean squared error performance, it is difficult to say which algorithm is better for this particular dataset, but, in general, the top-down adjustment is more coincident with the original clusters we set up and will not have the issues such as clustering two individuals far away from each other. Hence, we will use the top-down adjustment to estimate the survival function and compare with other estimation methods in later sections. Further study could focus on other alternative methods to improve the cluster classification procedures, such as clustering individuals using top-down and bottom-up approach together at the same time, which will end up at some point at middle of the dendrogram.

4.4.3 Methods comparisons

As mentioned in the literature review, the generalised Cox proportional hazard regressions are suitable for modelling recurrent lifetimes. For the datasets generated from the Weibull distribution, this section will compare our method with the generalised Cox model in terms of mean squared error performance.

We use the function `coxph` in R package `survival` (Therneau, 2021) to fit an AG model explained in section 4.2.1, as the AG model is the simplest generalised Cox proportional hazard regression, which assumes the independence of recurrent events within the individuals. Our data generated from Weibull distributions is consistent with the AG model assumptions. Then `survfit` function is applied to estimate the survival function under the AG model. Using our network approach, we can also obtain the survival estimates for each adjusted cluster. As we know the true survival function for each individual,

we can compute the mean squared error of the survival function for each individual, and take average according to the number of individuals. Suppose there are n individuals being considered, and for an individual j at time t, let $S_j(t)$ be the true survival function, $\hat{S}_j(t)$ be the survival estimate using the AG regression or our network approach, then the mean squared errors, MSE, can be calculated by

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}\Big[\int\Big\{\hat{S}_j(t) - S_j(t)\Big\}^2 \mathrm{d}t\Big] \tag{4.17}$$

$$\approx \frac{1}{n}\sum_{j=1}^{n}\Big[\frac{1}{m}\sum_{i=1}^{m}\Big\{\hat{S}_j(t_i) - S_j(t_i)\Big\}^2\Big], \tag{4.18}$$

where t_i with $i = 1,...,m$ is a time point in a given interval (t_1, t_m) that has been divided into $m-1$ equal parts.

For the examples displayed in sections 4.4.1 and 4.4.2, the mean squared error obtained using the AG model and our approach are 3.41×10^{-3} and 4.05×10^{-3} respectively, where we can see our approach has slightly larger error. However, when we decrease the number of individuals being considered, then our approach usually has better performance. For example, now we simulate a dataset of 40 individuals with eight recurrent lifetimes each, which are originally designed to be clustered into five groups randomly using the approach described in section 4.4.1. Figure 4.8 shows the original designed geographical locations of the 40 individuals according to the parameters of their Weibull distributions and Figure 4.9 is the dendrogram of clusters produced using our top-down approach, where the significance level is also 0.05 when applying K-S test.

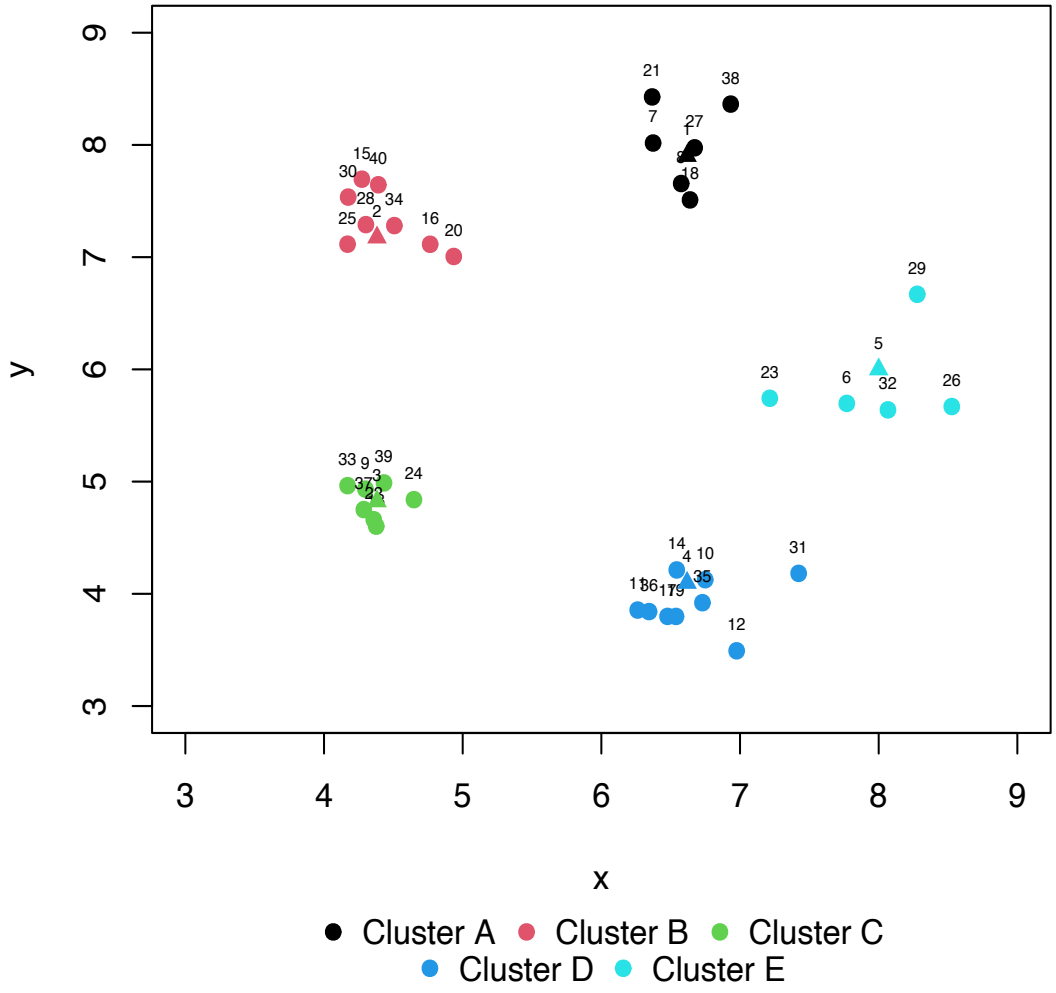


Figure 4.8: Geographical locations for 40 individuals, where triangles represent the cluster centers.

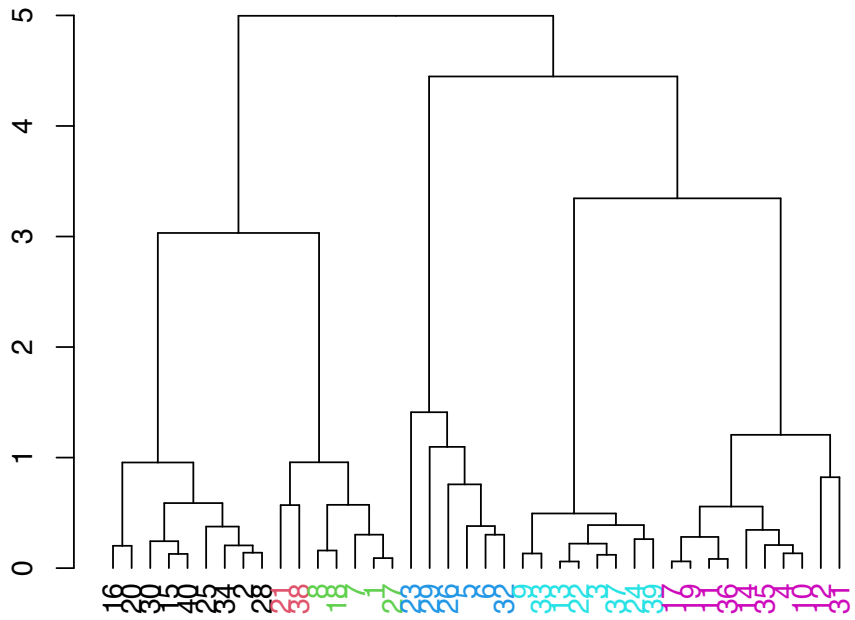


Figure 4.9: Dendrogram produced using the top-down approach, where different colours represent different clusters

The two figures show that, apart from individual 21 and 38, the clusters produced by our top-down approach coincide with the data generated originally, which results in a high adjust Rand index (i.e. 0.9565). Then we calculate the mean squared errors for our approach and AG model, which are now 3.03×10^{-3} and 3.57×10^{-3} respectively. For this particular dataset and setup, our approach has a better performance than the generalised Cox model.

Now we repeat our simulations using the above setup for 200 times, where 40 individuals with eight recurrent survival lifetimes for each grouped in five clusters are generated every time. We compute the mean squared error of the estimated survival functions for every simulation and obtain the following boxplots for both our method and AG model, where we find our method gen-

erally has better performance with fewer outliers and extreme small values of the mean squared error (i.e. lower than 2×10^{-3}). In addition, these 200 simulations show that there are 66.5% of the generated datasets having the lower mean squared error using our method to estimate the survival functions compared to the AG model.

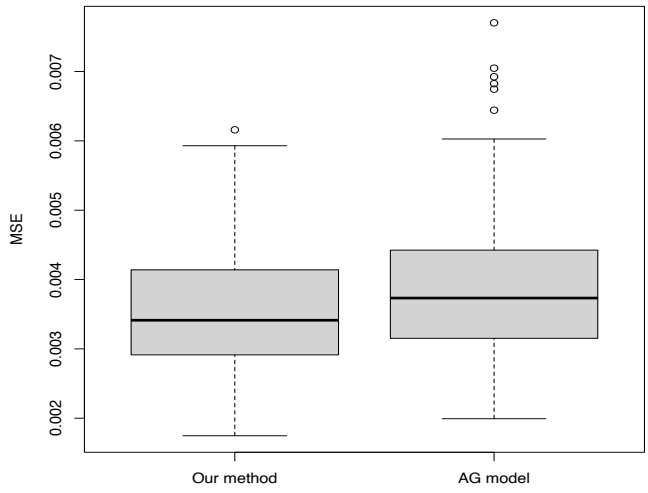


Figure 4.10: Boxplots of the mean squared errors for 200 simulations using our method and the AG model

From the two examples displayed above and some more trials, we find the mean squared error for our method and the generalised Cox model are similar. However, for a large number of data points, the generalised Cox model has the better performance, which has different versions for different types of datasets, whereas our method may perform better when the number of data points are not large enough for using regression models especially when there exists lots of covariates. In fact, our method combines the influence of all covariates, and

focuses on improving the survival/ hazard estimates using a network based on the covariates, on the other hand, the (generalised) Cox models focus on examining how specified factors influence the hazard rate of a event happening at a particular point in time.

However, as our simulated data is Weibull-distributed, a Weibull distribution for survival estimation is probably optimistic. Actually, once we get the survival times in clusters, we can use any method such as the Kaplan-Meier method etc. to estimate the survival/hazard function. The main idea of our method is to build clusters and put more lifetimes in each cluster to improve the estimates.

4.5 Conclusions

This chapter introduces a new method to estimate survival functions for groups of individuals with recurrent survival lifetimes using their network structure, where we construct the initial clusters of the network using a dissimilarity matrix and adjust these clusters with our proposed top-down and bottom-up approaches. After obtaining new clusters, assuming the underlying survival functions for each group are the same, we obtain these estimates by using all lifetimes in that group. In addition, we conduct the Weibull distribution simulations and compare the performance of our methods with the (generalised) Cox models, where we find that our methods have advantages when there are small number of data but many covariates for regression models.

Both this chapter and the previous one focus on estimating the survival (density/hazard) function, further investigation could work on the possibility of combining methods in these two chapters to improve the estimation process of recurrent lifetimes. To be specific, individuals could be grouped using the methods discussed in this chapter and then, survival (density/hazard) estimates could be conducted using our wavelet methods explained previously with all lifetimes in each group.

CHAPTER 5

NETWORK AUTOREGRESSIVE PROCESS WITH EXOGENOUS NETWORK TIME SERIES

5.1 Introduction

Multivariate time series are widely available in various fields, so modelling this kind of data well is important for making decisions. In the past, the *vector autoregressive model* (Sims, 1980) is one of the most commonly used models to deal with the multivariate time series. The use of network data has increased in recent years, so modelling multivariate time series data with the use of any underlying network structure becomes more and more popular. This kind of modelling not only incorporates the influence of the previous time lags of its own and its neighbours, but also reduces the VAR model to a more efficient one as usually fewer parameters are required to be estimated. For example, Knight et al. (2016) proposed the *network autoregressive (integrated) moving average processes* (NARIMA). Based on the network lifting (wavelet) transform, they also introduced network differencing to remove trend. Zhu et al. (2017) investigated conditions for the strict stationarity and the asymptotic properties

of the *network autoregressive model* (NAR), as well as developing own ordinary least squares type estimator. Knight et al. (2020) generalised the NAR model and made a R package called `GNAR` (Leeming et al., 2020), which is available on the `CRAN` repository. Nason and Wei (2021) extend these network models to include node-specific time series exogenous variables, which is called GNARX model.

Our model adds another network time series as exogenous regressors to the NAR model, which we call the *network autoregressive process with exogenous network time series* (NAREN). Basically, our target multivariate time series is modelled by the previous time lags of itself and its neighbours according to the network structure, *as well as another* network time series, which is related of our target time series. With our proposed model, we demonstrate how to choose the model order by optimising AIC values in section 5.3.2. To compare the forecasting performance between different multivariate time series models, we use simple GNAR and our NAREN models to fit one simulation example and two real data sets, where model orders are chosen by subjective judgement, and the results are shown in section 5.4.

5.2 Literature Review

In this section, some useful definitions from time series analysis books including Shumway and Stoffer (2011) and Box et al. (2015) are presented. In addition, we introduce the *network autoregressive model* (NAR), originally proposed by Knight et al. (2016), which is then extended to the *generalised network*

autoregressive model (GNAR) (Knight et al., 2020). The recent published work proposed by Nason and Wei (2021), the *generalised network autoregressive model with exogenous node-specific regressors* (GNARX), is also reviewed.

5.2.1 The univariate autoregressive model

To begin, we define the term *(weakly) stationary*. The stochastic process $\{X_t\}_{t\in T}$ is said to be *weakly stationary* if for all $n \geq 1$, and for any j such that $t_1, t_2, \ldots, t_n$ and $t_1 + j, t_2 + j, \ldots, t_n + j$ belong to some index set $T \subset \mathbb{N}$, all the joint moments of order one and two of $X_{t_1}, X_{t_2}, \ldots, X_{t_n}$ exist, are finite, and equal to the corresponding joint moments of $X_{t_1+j}, X_{t_2+j}, \ldots, X_{t_n+j}$. Hence, $\mathbb{E}\{X_t\} \equiv \mu$ and $\text{Var}\{X_t\} \equiv \sigma^2$ are constants independent of t.

The weak stationarity defined above allows to introduce the *autocovariance* sequences for the process $\{X_t\}$. As the covariance between X_{t+j} and X_t, separated by j intervals of time or by lag j is the same for all t under the stationary assumption, the *autocovariance* for a stationary time series at lag j can be defined as

$$\gamma(j) = \text{Cov}(X_t, X_{t+j}) = \mathbb{E}\{(X_t - \mu)(X_{t+j} - \mu)\}. \tag{5.1}$$

Similarly, the *autocorrelation* at lag j is

$$\rho(j) = \text{Cor}(X_t, X_{t+j}) = \frac{\gamma(j)}{\gamma(0)}. \tag{5.2}$$

Now, the *white noise process* and the *autoregressive process* (AR) are defined as follows.

- **white noise process**

  A sequence of uncorrelated random variable $\{X_t\}$ is a white noise process if

$$\mathbb{E}\{X_t\} = 0, \quad \text{Var}\{X_t\} = \sigma^2 \quad \forall t, \tag{5.3}$$

  and

$$\rho(j) = \begin{cases} 1 & j = 0, \\ 0 & j \neq 0. \end{cases}$$

- **p-th order autoregressive process AR(p)**

  $\{X_t\}$ is a AR(p) if it can be expressed in the form

$$X_t = \mu + \sum_{j=1}^{p} \phi_j X_{t-j} + \epsilon_t, \tag{5.4}$$

  where $\{\phi_j\}_{j=1}^{p}$ and μ are real constants with $\phi_p \neq 0$, and ϵ_t is a white noise process.

  For a zero mean AR(p) process, rearranging the expression in (5.4), we have

$$\epsilon_t = X_t - \sum_{j=1}^{p} \phi_j X_{t-j} = \big(1 - \sum_{j=1}^{p} \phi_j L^j\big) X_t =: \phi(L) X_t \tag{5.5}$$

  where $\phi(L)$ is the lag polynomial and L is the lag operator where $L^j X_t = X_{t-j}$ for $j = 1, \ldots, p$. For an AR process, stationarity is not always guaranteed. Briefly speaking, an AR process is stationary if all roots of the lag polynomial $\phi(L)$ are outside the unit circle.

5.2.2 The vector autoregressive model with exogenous variables (VARX)

A *Vector autoregressive model with exogenous variables* (VARX) (Hamilton, 1994) is an extension of the *Vector autoregressive model* (VAR) (Sims, 1980), both of which are commonly used for multivariate time series. For $i = 1,...,N$, $j = 1,...,M$ and $t = 1,...,T$, let $\{X_{i,t}\}$ and $\{Y_{j,t}\}$ be the time series for variable i and j at time t respectively, we have the following definitions.

- **p-th order vector autoregressive process VAR(p)**

  Let $\mathbf{X}_t = (X_{1,t},...,X_{N,t})^T \in \mathbb{R}^N$ be a vector of time series observations, for t=1,2,...,T, $\{\mathbf{X}_t\}$ can be expressed in the form

  $$\mathbf{X}_t = \boldsymbol{\mu} + \boldsymbol{\Phi}_1\mathbf{X}_{t-1} + ... + \boldsymbol{\Phi}_p\mathbf{X}_{t-p} + \boldsymbol{\epsilon}_t, \tag{5.6}$$

  where $\{\boldsymbol{\Phi}_i\}_{i=1}^p$ are $N \times N$ matrices of coefficients, $\boldsymbol{\mu} = (\mu_1,...,\mu_N)^T$ is a vector mean, and $\boldsymbol{\epsilon}_t = (\epsilon_{1,t},...,\epsilon_{N,t})^T$ is a white noise process with zero-mean and time-invariant positive definite covariance matrix $\mathbb{E}(\boldsymbol{\epsilon}_t\boldsymbol{\epsilon}_t^T) = \Sigma_\epsilon$.

- **p-th order vector autoregressive process with exogenous variables of lag q VARX(p,q)**

  Let $\mathbf{Y}_t = (Y_{1,t},...,Y_{M,t})^T \in \mathbb{R}^M$ be an another vector of time series observations, for t=1,2,...,T, we have

  $$\mathbf{X}_t = \boldsymbol{\mu} + \boldsymbol{\Phi}_1\mathbf{X}_{t-1} + ... + \boldsymbol{\Phi}_p\mathbf{X}_{t-p} + \boldsymbol{\Theta}_1\mathbf{Y}_{t-1} + ... + \boldsymbol{\Theta}_q\mathbf{Y}_{t-q} + \boldsymbol{\epsilon}_t, \tag{5.7}$$

  where $\mathbf{X}_t$, $\{\boldsymbol{\Phi}_i\}$, $\boldsymbol{\mu}$, $\boldsymbol{\epsilon}_t$ are as defined in the VAR(p) model, and $\{\boldsymbol{\Theta}_j\}_{j=1}^q$ are $N \times M$ coefficient matrices.

The definitions above show that the general mean-zero VAR model has up to pN^2 parameters for the $\{\mathbf{\Phi}_i\}_{i=1}^{p}$ matrices, and the VARX model will include another maximum of qNM variables for the $\{\mathbf{\Theta}_j\}_{j=1}^{q}$ matrices. Therefore, a downside of the VAR and VARX models is that when N or M is large compared to T, only few orders of the full model can be usually fitted.

5.2.3 The generalised network autoregressive model (GNAR)

The *network autoregressive model* (NAR) (Knight et al., 2016), as well as the *generalised network autoregressive model* (GNAR) (Knight et al., 2020), describes a multivariate time series process that uses the network structure of associated variables, where all time series data are collected on the network $\mathscr{G} = (\mathscr{K}, \mathscr{E})$, with a set of nodes, $\mathscr{K}$, which are connected by edges from the set of edges $\mathscr{E}$. Each node in a network represents a variable and follows an autoregressive model with additional dependence defined by the neighbourhood structure of each node.

Suppose $\mathscr{K} = \{1, ..., K\}$, we use the notation $i \nsim j$ if nodes $i, j \in \mathscr{K}$ are connected by an (undirected) edge. Hence, we can define the set of edges by $\mathscr{E} = \{(i, j) : i \nsim j; i, j \in \mathscr{K}\}$. Suppose $\mathscr{A} \subset \mathscr{K}$ is a subset of nodes, the neighbourhood set of $\mathscr{A}$ is defined by $\mathscr{N}(\mathscr{A}) = \{j \in \mathscr{K}/\mathscr{A} : i \nsim j; i \in \mathscr{A}\}$. Hence, the set of rth-stage neighbours of a node $i \in \mathscr{K}$ is

$$\mathscr{N}^{(r)}(i) = \mathscr{N}\{\mathscr{N}^{(r-1)}(i)\}/\left[\left\{\cup_{s=1}^{(r-1)} \mathscr{N}^{(s)}(i)\right\} \cup \{i\}\right], \tag{5.8}$$

for $r = 2,3,...$ and $\mathcal{N}^{(1)}(i) = \mathcal{N}(\{i\})$ is the set of immediate neighbours of node i. Then for time $t = 1,...,T$ and node $i = 1,...,K$, the NAR(p,[**s**]) model is defined as

$$X_{i,t} = \sum_{j=1}^{p} \left(\alpha_j X_{i,t-j} + \sum_{r=1}^{s_j} \sum_{q \in \mathcal{N}^{(r)}(i)} \beta_{j,r,q} X_{q,t-j} \right) + \epsilon_{i,t}, \tag{5.9}$$

where $\{\alpha_j\}$ and $\{\beta_{j,r,q}\}$ are parameter sets, $\mathcal{N}^{(r)}(i)$ is defined in (5.8), p is the maximal time lag, $\mathbf{s} = (s_1,...,s_p)$ is a vector indicating the number of neighbour stages to include at each time lag, and $\{\epsilon_{i,t}\}$ are mutually uncorrelated white noise processes. The NAR model assumes that the $\{\alpha_j\}$ and $\{\beta_{j,r,q}\}$ do not depend on time t, neither do they depend on node i.

To incorporate distance information into the network structure, the connection weight of a node $i \in \mathcal{K}$ and its rth-stage neighbour $j \in \mathcal{N}^{(r)}(i)$ can be defined as

$$w_{i,j} = d(i,j)^{-1} \Big\{ \sum_{k \in \mathcal{N}^{(r)}(i)} d(i,k)^{-1} \Big\}^{-1}, \tag{5.10}$$

where $d(i,j)$ is the distance from node i to node j.

Then, the GNAR(p,[**s**]) (Knight et al., 2020) is defined as follows, which additionally allows for different autoregressive parameters at each node and also includes the covariates on edges or nodes.

$$X_{i,t} = \sum_{j=1}^{p} \left(\alpha_{i,j} X_{i,t-j} + \sum_{c=1}^{C} \sum_{r=1}^{s_j} \beta_{j,r,c} \sum_{q \in \mathcal{N}^{(r)}(i)} w_{i,q,c}^{(t)} X_{q,t-j} \right) + \epsilon_{i,t}, \tag{5.11}$$

where p, **s** and $\{\epsilon_{i,t}\}$ are defined in the NAR(p,[**s**]) model, $\{\alpha_{i,j}\}$ and $\{\beta_{j,r,c}\}$ are parameter sets, $C \in \mathbb{N}$ are the covariates splitting edges or nodes into different types. As defined in (5.10), $w_{i,q,c}^{(t)}$ is the connection weight between nodes i and q at time t for type c.

5.2.4 The generalised network autoregressive model with exogenous node-specific regressors (GNARX)

Let $\{X_{i,t}\}$ be a network time series based on the network $\mathscr{G} = (\mathscr{K}, \mathscr{E})$, and $\{Y_{h,i,t}\}$ be the h-th real-valued stationary exogenous node-specific regressor series for node i at time t, where $h = 1, ..., H \in \mathbb{N}$ is the number of such regressors. Based on the GNAR model, Nason and Wei (2021) proposed the *generalised network autoregressive model with exogenous node-specific regressors* (GNARX $(p, \mathbf{s}, \mathbf{p}')$), such that

$$X_{i,t} = \sum_{j=1}^{p} \left(\alpha_{i,j} X_{i,t-j} + \sum_{c=1}^{C} \sum_{r=1}^{s_j} \beta_{j,r,c} \sum_{q \in \mathscr{N}^{(r)}(i)} w_{i,q,c}^{(t)} X_{q,t-j} \right) + \sum_{h=1}^{H} \sum_{j'=0}^{p'_h} \lambda_{h,j'} Y_{h,i,t-j'} + \epsilon_{i,t}, \tag{5.12}$$

where $\{\alpha_{i,j}\}$, $\{\beta_{j,r,c}\}$, $\{\lambda_{h,j'}\}$ are real parameter sets , $p, \mathbf{s}, \{\epsilon_{i,t}\}$ and $\{w_{i,q,c}^{(t)}\}$ are defined in the GNAR$(p, [\mathbf{s}])$ model, and p'_h, the h-th element of $\mathbf{p}' = (p'_1, ..., p'_H)$ denotes the maximum lag of the h-th exogenous regressor involved in the model.

The GNARX model, which is stationary under the assumption of stationarity of $\{Y_{h,i,t}\}$ for all h, i and parameter constraints

$$\sum_{j=1}^{p} \left(|\alpha_{i,j}| + \sum_{c=1}^{C} \sum_{r=1}^{s_j} |\beta_{j,r,c}| \right) < 1, \tag{5.13}$$

shares the same advantages with respect to missing data as the GNAR model.

5.3 Network Autoregressive Process with Exogenous Network Time Series (NAREN)

5.3.1 Model specification

Suppose that $\{X_{i,t}\}$ and $\{Y_{i,t}\}$ are two *network* time series associated with the same graph $\mathcal{G}$, the *network autoregressive process with exogenous network time series*, NAREN $(p_x, [\mathbf{s_x}], p_y, [\mathbf{s_y}])$, models $X_{i,t}$ in terms of both $\{X_{i,t}\}$ and $\{Y_{i,t}\}$, where $\{X_{i,t}\}$ and $\{Y_{i,t}\}$ are two separate NAR processes. Our proposed model can be expressed as

$$\begin{aligned} X_{i,t} = &\sum_{j=1}^{p_x} \left(\alpha_{i,j} X_{i,t-j} + \sum_{r=1}^{s_{x,j}} \beta_{j,r} \sum_{q \in \mathcal{N}^{(r)}(i)} w_{i,q}^{(t)} X_{q,t-j} \right) \\ &+ \sum_{j=1}^{p_y} \left(\theta_j Y_{i,t-j} + \sum_{r=1}^{s_{y,j}} \phi_{j,r} \sum_{q \in \mathcal{N}^{(r)}(i)} w_{i,q}^{(t)} Y_{q,t-j} \right) + \epsilon_{i,t}, \end{aligned} \tag{5.14}$$

where p_x and p_y are maximal lags for $\{X_{i,t}\}$ and $\{Y_{i,t}\}$ respectively, $[\mathbf{s_x}] = (s_{x,1}, ..., s_{x,p_x})$ and $[\mathbf{s_y}] = (s_{y,1}, ..., s_{y,p_y})$ are corresponding neighbour stage vectors, $\mathcal{N}^{(r)}(i)$, $\{w_{i,q}^{(t)}\}$ and $\{\epsilon_{i,t}\}$ are the same as the GNAR model . Also, we assume that the parameter sets $\{\alpha_{i,j}\}$, $\{\beta_{j,r}\}$, $\{\theta_j\}$ and $\{\phi_{j,r}\}$ do not depend on time t.

Compared to the GNAR and GNARX models, our proposed model includes the exogenous regressors of *another network time series*, which is really useful when our target variable $X_{i,t}$ is not only affected by its own and its neighbours' previous lags, but also some other network time series. In the NAREN model, we do not include node-specific autoregressive parameters of $\{Y_{i,t}\}$, as $\{Y_{i,t}\}$ are usually not the main factors that may affect the response and using the global

parameter, θ_j, of $\{Y_{i,t}\}$ for every node i will simplify our model and reduce the computing time. Similarly with the GNAR model, our NAREN model allows for different autoregressive parameters of $\{X_{i,t}\}$, but we do not include the covariates C for simplicity. Compared to the GNARX, our model includes more information as the exogenous regressors are another network time series rather than node-specific time series. Compared to the VARX model, our model has fewer parameters to estimate, that is up to a maximum of

$$Kp_x + \sum_{1}^{p_x} s_{x,j} + p_y + \sum_{1}^{p_y} s_{y,j} \tag{5.15}$$

parameters.

Therefore, fewer data points are required to fit our model, which also can have good performance as the network structure of the two time series provides extra information that VAR-like models do not have access to.

5.3.2 Model selection and parameter estimation

This section illustrates how model order is chosen in general. Indeed, we first need to determine the values of $p_x, \mathbf{s_x}, p_y$ and $\mathbf{s_y}$, and then estimate parameters for that particular model. By treating our NAREN model as a linear regression, we use the least squares approach to fit the NAREN models with different combinations of $p_x, \mathbf{s_x}, p_y$ and $\mathbf{s_y}$, and obtain the Akaike information criterion (AIC) values (Akaike, 1974) for each model. Let k be the number of estimated parameters, and $\hat{L}$ be the maximum value of the likelihood function for the model, the AIC value of the model is defined by

$$\text{AIC} = 2k - 2\log(\hat{L}). \tag{5.16}$$

The definition above shows that the AIC not only takes goodness of fit into consideration, but also includes a penalty of increasing number of estimated parameters. It is shown by Shibata (1980) that, for an infinite-order autoregressive process, the order selection by minimising AIC is asymptotically optimal for prediction in large samples, which provides the smallest possible one-step mean squared prediction error. Also, let n be the number of time series, some simulations given in Bhansali (1984) show that AIC will consistently estimate the order if some finite constant, that is dependent on n, varies slowly with n. Hence, we always prefer the NAREN model with the minimum AIC value for a set of candidate models.

However, it is computationally inefficient to fit the NAREN models with all possible combinations of $p_x, \mathbf{s_x}, p_y$ and $\mathbf{s_y}$ to choose the set of parameters with the smallest AIC value. To mitigate this issue, we fit a *univariate autoregressive model with exogenous regressors* (UARE) for each node and find the time lags for the predictors $\{X_{i,t}\}$ and $\{Y_{i,t}\}$ leading a minimum AIC. To be specific, the UARE model for node i at time t is defined by

$$X_{i,t} = \sum_{j=1}^{p_{i,x}} \alpha_{i,j} X_{i,t-j} + \sum_{j=1}^{p_{i,y}} \theta_{i,j} Y_{i,t-j} + \epsilon_{i,t}, \tag{5.17}$$

where $\epsilon_{i,t}$ are zero-mean white noise process, $\{\alpha_{i,j}\}$ and $\{\theta_{i,j}\}$ are parameter sets, $\{p_{i,x}\}$ and $\{p_{i,y}\}$ are time lags of $X_{i,t}$ and $Y_{i,t}$ respectively. (Although the UARE model is written using the similar parameter notations with our NAREN model in (5.14), the fitted values of these parameters might be different between the two models.)

For each node i, we will obtain the values of $p_{i,x}$ and $p_{i,y}$ that result in the

minimal AIC. The most frequent $p_{i,x}$ and $p_{i,y}$ are chosen to be the estimates of p_x and p_y in our NAREN model, which is not the only way to determine the parameters of our NAREN model. Other methods, such as choosing the mean or median of $p_{i,x}$ and $p_{i,y}$, will also be alternatives, and further study on setting appropriate time lags for the two network time series in our NAREN model can be carried out. After choosing the appropriate p_x and p_y, we then use the linear regression to fit the NAREN models with all possible combinations of $\mathbf{s_x}$ and $\mathbf{s_y}$ to obtain the lowest AIC value, which will be much more efficient. Here, the limit of the entries of vectors $\mathbf{s_x}$ and $\mathbf{s_y}$ are the minimum of the maximum neighbour stages of all nodes in the network, which are set to be five for those larger than five, as further stage-neighbours have less influence on the time series of the current node.

5.3.3 Forecasting performance measurements

In many situations, a model that has a good in-sample fit will not necessarily ensure that the out-of-sample performance of the model is also good. Hence, we use the *pseudo-out-of-sample forecasting* (POOSF) method to measure the forecasting performance of our NAREN model. To see how our NAREN model performs out of the sample, we exclude some of our network time series data, and use our model to predict the outcome. Doing this iteratively for a number of periods, we can obtain a set of forecasting errors, which can be used to compare the performance for different time series models. The following steps describe how the method works.

Step 1: Split our network time series and exclude some observations

as the out-of-sample, which is usually the last 10% to 20% of the time period. Denote the number of period excluded by P.

Step 2: Use the remaining data (i.e. the first $r = T - P$ periods of data, where T is the full length of our time series) to estimate our NAREN model.

Step 3: For node i at time $r+1$ given the time series of the first r periods, compute our pseudo-forecast time series, $\hat{X}_{i,r+1|r}$. Hence, using the actual time series, $X_{i,r+1|r}$, the forecast error, $\hat{e}_{i,r+1|r}$, is obtained by

$$\hat{e}_{i,r+1|r} = X_{i,r+1|r} - \hat{X}_{i,r+1|r}. \tag{5.18}$$

Step 4: Move one time period forwards and repeat the process of fitting the NAREN model and computing the forecast error until the final period. So the number of out-of-sample decreases by one and the sample used to fit the model increase by one every time. Hence, we can obtain a set of pseudo-out-of-sample errors and the *root mean squared forecast error* (RMSFE) for node i can be estimated by the following formula

$$\text{RMSFE}_i = \sum_{j=r}^{T-1} \sqrt{\frac{\hat{e}^2_{i,j+1|j}}{P}}, \tag{5.19}$$

where $\{\hat{e}_{i,j+1|j}\}$ is defined in (5.18).

After obtaining the RMSFE for all nodes in a given network, we can take mean to represent the forecasting error for our NAREN model and compare the forecasting performance with other models.

5.4 Some Examples

Using one simulation example and two real applications, we want to see the forecast performance of different multivariate time series models, where simple possible NAREN/GNAR models are chosen to make comparison. For the five-node network example in 5.4.1, we compare the forecast performance between the simplest GNAR and NAREN models, as it is unnecessary to apply complex network models to such a simple network with maximum neighbour-stages of two or three for all nodes. When analysing the two real network time series applications, we increase the numbers of time lags and neighbour stages considered, as the network structures of these two examples are more complicated. Taking into the information provided by including some extra parameters improves the forecast performance for these two real examples. To compare the forecasting performance of the GNAR and NAREN models, we choose the simple comparative benchmarks rather than models with the lowest AIC values for the following two reasons.

- For complex networks, models with the lowest AIC values always accompany with many parameters (e.g. the wind example in section 5.4.2), which is computational expensive to conduct the POOSF described in section 5.3.3 and compute the RMSFE.

- The simple comparative models sometimes have better forecasting performance, as models with the lowest AIC values do not guarantee the lowest RMSFE in practice.

Further study could be worked on determining network models based on different criteria for comparing the forecasting performance for different models.

5.4.1 Simulation example

This section displays a simulation example, where the simple five-node network from the R package `GNAR` (Leeming et al., 2020) shown in Figure 5.1 is used.

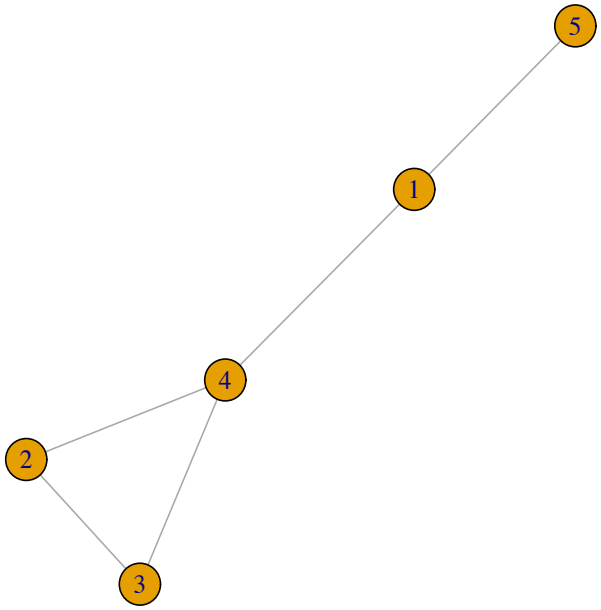


Figure 5.1: Five-node Network.

We generate a particular sample with 500 time series observations for both $\{X_{i,t}\}$ and $\{Y_{i,t}\}$ using a NAREN (1,[1],1,[1]) model. For $i = 1,...,5$ and $t = 1,...,500$, the model can be written as

$$X_{i,t} = \alpha_1 X_{i,t-1} + \beta_{1,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{i,q}^{(t)} X_{q,t-1} + \theta_1 Y_{i,t-1} + \phi_{1,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{i,q}^{(t)} Y_{q,t-1} + \epsilon_{i,t}, \tag{5.20}$$

where we use the global autoregressive parameter α_1 instead of different parameters for each node. Here, the parameters chosen when generating data

are shown in Table 5.1.

Parameter	α_1	$\beta_{1,1}$	θ_1	$\phi_{1,1}$
Value	0.5	0.3	0.4	0.2

Table 5.1: Parameters chosen in NAREN $(1,[1],1,[1])$ model for generating time series data.

We now fit the GNAR $(1,0)$, GNAR $(1,[1])$, and NAREN $(1,[1],1,[1])$ models using the data generated from (5.20) for model comparison. For this particular data set, we compute the AIC values of these three models to see the goodness of fit and use the method described in section 5.3.3 to compute the RMSFE to show their forecasting performance. The results are shown in Table 5.2.

As our dataset $\{X_{i,t}\}$ is generated from (5.20) with parameters stated in Table 5.1, the parameters estimated using NAREN $(1,[1],1,[1])$ model are almost the same as the truth. In addition, if the multivariate time series $\{X_{i,t}\}$ depends on some exogenous variables, fitting a GNAR model with network structure (i.e. GNAR $(1,[1])$ model) will only have small improvement on both goodness of fit and forecasting performance compared to the one not using the neighbourhood information (i.e. GNAR $(1,0)$ model), whereas our NAREN model has a significant decrease in terms of AIC and RMSFE values (i.e. 33.5% and 30.5% decrease in AIC value, 52.5% and 46.7% decrease in RMSFE compared to the GNAR $(1,0)$ and GNAR $(1,[1])$ model respectively).

	GNAR (1,0)			GNAR (1,[1])			NAREN (1,[1],1,[1])		
	Estimate	s.e.($\times 10^{-3}$)	95% CI	Estimate	s.e.($\times 10^{-3}$)	95% CI	Estimate	s.e.($\times 10^{-3}$)	95% CI
$\hat{\alpha}_1$	0.964	5.06	(0.954, 0.974)	0.533	19.7	(0.494, 0.571)	0.507	10.9	(0.485, 0.528)
$\hat{\beta}_{1,1}$	-	-	-	0.457	20.3	(0.417, 0.497)	0.296	11.2	(0.274, 0.317)
$\hat{\theta}_1$	-	-	-	-	-	-	0.397	5.72	(0.386, 0.408)
$\hat{\phi}_{1,1}$	-	-	-	-	-	-	0.198	7.33	(0.183, 0.212)
AIC	10610			10149			7053		
RMSFE	2.0653			1.8389			0.9800		

Table 5.2: Simulation results of GNAR (1,0), GNAR (1,[1]) and NAREN (1,[1],1,[1]) models for our simulated data, where "s.e." and "CI" in the table represents the standard error and confidence interval respectively.

5.4.2 Wind data

5.4.2.1 Exploratory data analysis

The dataset we use here records the wind speed (metres per second) and direction (in radians) hourly at 102 different locations in the UK over 30 days. The hourly measurements were taken by a pilot anemometer at a height of 10m at the target locations. These target wind speeds and directions are related to contemporaneous measurements taken at a nearby reference site (i.e. a Meteorological Office station in the UK), so our models and predictions are based on the reference wind speeds and directions.

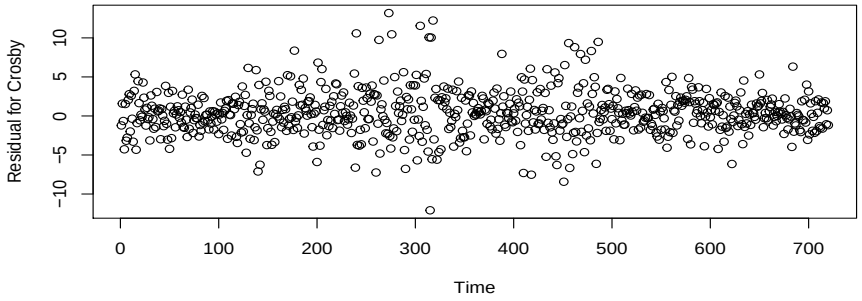
Our aim is to model the wind speed for each target location according to its previous speed and direction as well as its neighbours', which means that the exogenous variable in this example is the wind direction. Hence, we are able to compare the forecasting performance of our NAREN model with other time series models. Before analysing the wind data, we clean up the dataset by deleting the nodes (locations) with missing values, so only 75 out of 102 target locations remain.

We then apply a variance stabilizing logarithmic transform to the wind speed using the following equation:

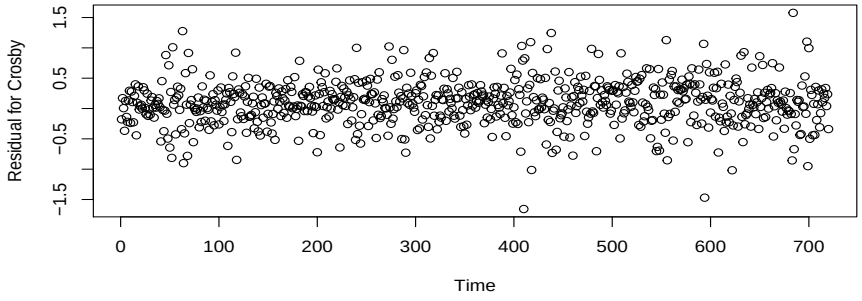
$$\text{new data} = \log(1 + \text{original data}). \tag{5.21}$$

After applying the transformation to the wind speed, we find the residual plots obtained when fitting a NAREN model usually show slightly better adherence to variance constancy, with fewer outliers presented. Figure 5.2 shows the residual plots before and after transformation for the location **CROSBY** when

fitting a NAREN(1,[1],1,[1]) model with a network linking all nodes less 80km, where we find that plot (b) has a slightly better performance.



(a) Residual plots for CROSBY before transformation



(b) Residual plots for CROSBY after transformation

Figure 5.2: Residual plots for CROSBY after fitting a NAREN(1,[1],1,[1]) model with a network linking all nodes less 80km

5.4.2.2 Network construction and selection

The wind system does not come with a network, so we create one to execute our further analysis. Given the geographical position for each target location, we are able to construct different networks with the following methods:

- **Method 1:** Minimum spanning tree (MST), which links all nodes without any cycles and with the minimum possible total distance;
- **Method 2:** Connect all nodes less than certain distance apart, and link the closest node for those isolated ones, if applicable.

Figure 5.3 displays the MST network, where we can see that some nodes are quite close geographically but not connected. For example, **ABERP** and **SENNY** are really close to each other, but the MST method does not link them. This is why we introduce the second method, which tries to include all influence of the wind speed and direction within a certain distance from the target location.

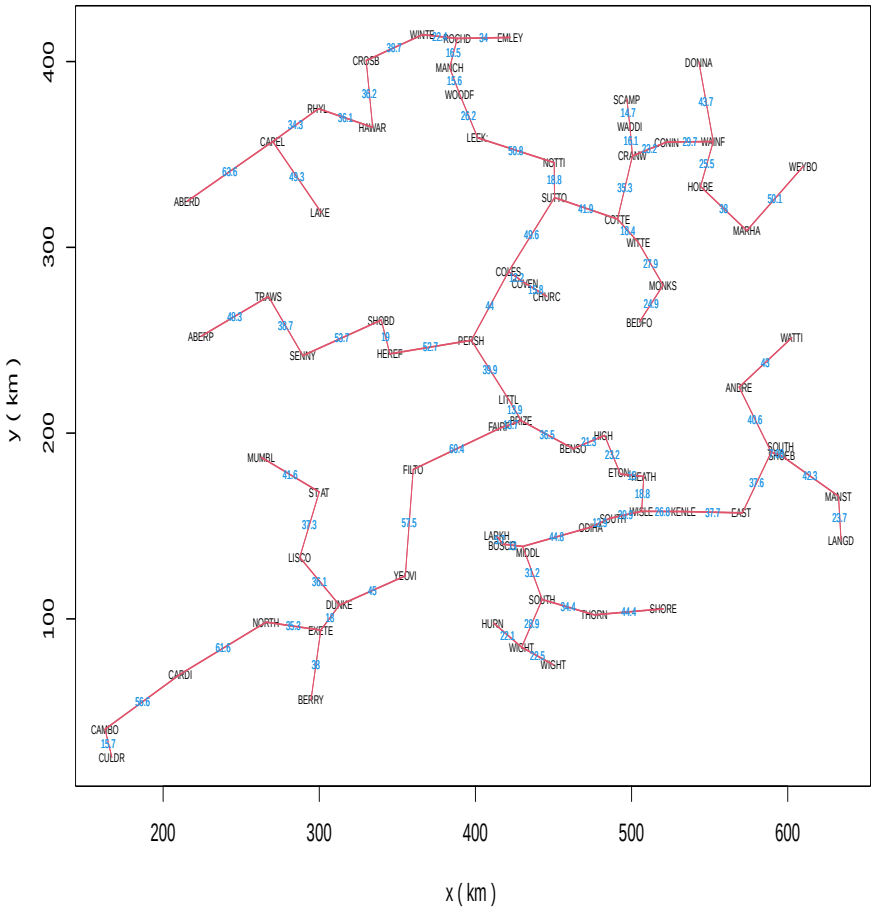


Figure 5.3: Network connection for wind data using MST method.

Figure 5.4 displays two networks that connect nodes less 25km and 65km respectively, where we notice that not all nodes are linked together when we

choose a small value of distance (e.g. 25km).

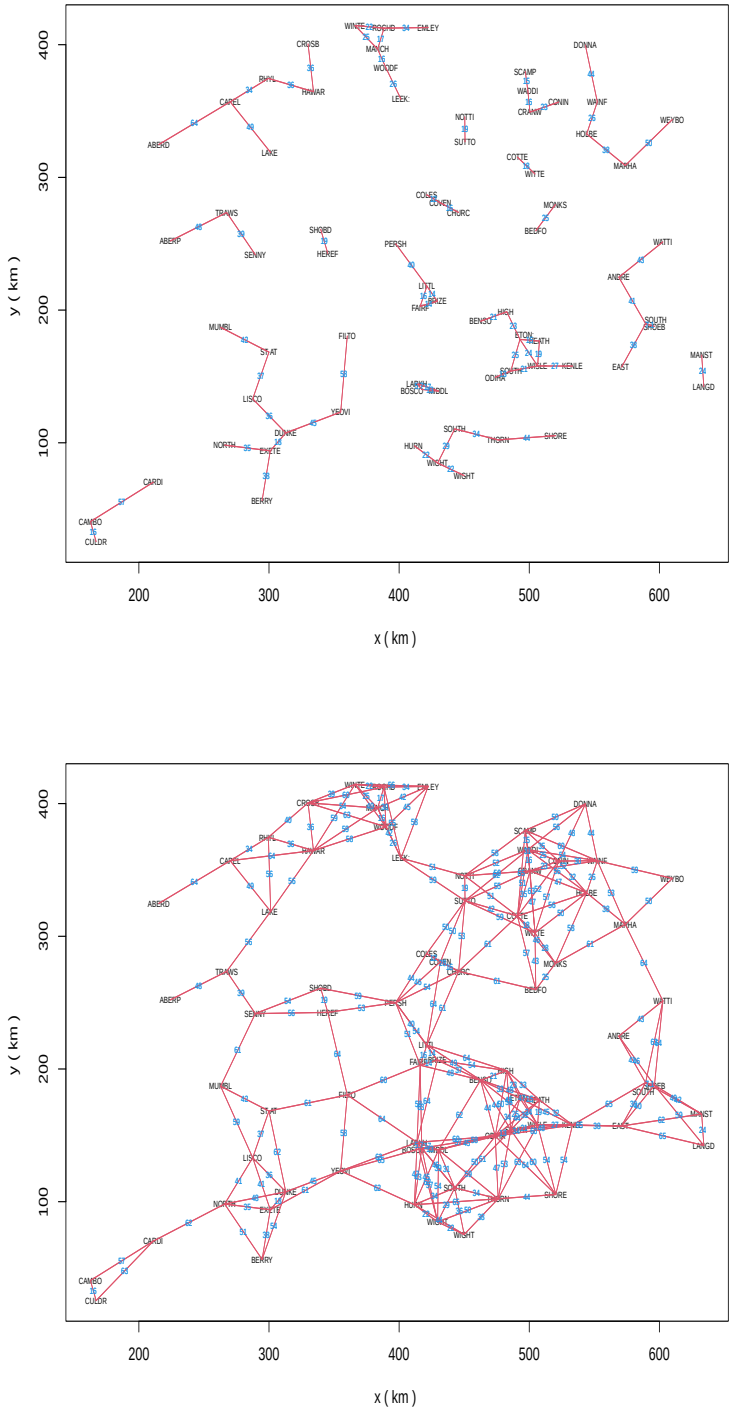


Figure 5.4: Network connection for wind data linking nodes less than 25km (top) and 65km (bottom).

In order to connect all nodes within one network, which can avoid some issues when removing the spatial trend of the network (Nunes et al., 2015), we improve our second construction method. In the modified version of the second method, we connect all nodes less than certain distance apart to form several groups of linked networks. Then, these groups of networks are linked by selecting one node from every group and connecting the selected nodes in order to minimise the possible total distances. Figure 5.5 shows a network constructed using this modified method when the distance is set to be 25km.

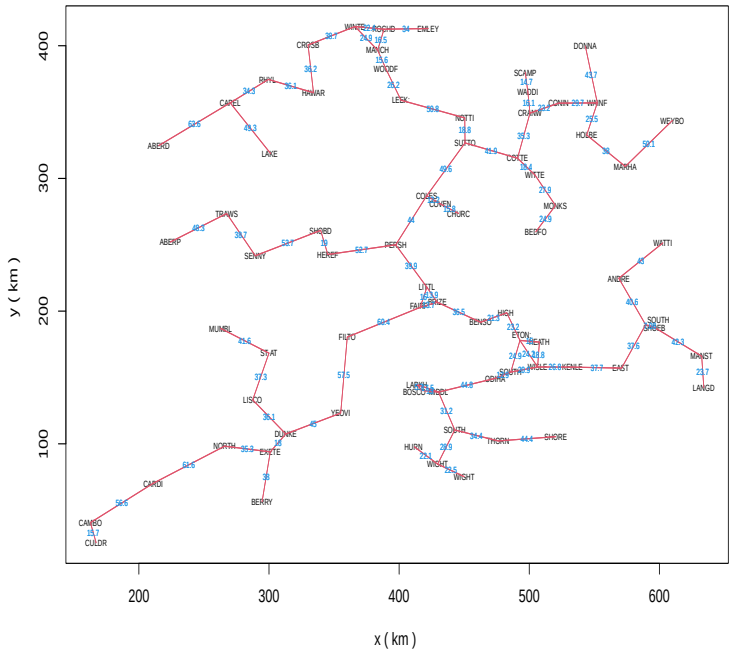


Figure 5.5: Network connecting nodes less than 25km with modified method.

Using the modified network construction method, we choose to construct a network by connecting nodes within 80km for further analysis, as there are some articles based on investigating wind energy from a meteorological per-

spective showing that 80km is a reasonable resolution while constructing a network for the `wind` data. For example, Martin et al. (2015) investigated wind-speed correlations using three extensive datasets spanning continents, durations and time resolution, while Cradden et al. (1974) focus on the study of estimates of hourly aggregate wind power generation for the UK, with wind speed resolution of around $40-50$km. All these studies show that the most appropriate choice of wind speed resolution requires a knowledge of the topography of the target region (e.g. complex or smooth terrain), the frequency of the wind speed data recorded (i.e. every minute, hourly, daily etc.) and the height above the ground level when measurements taking place. Although there is no universal appropriate wind speed resolution, most analysis consider regions smaller than 100km while modelling, which is consistent with our decision while constructing the network.

5.4.2.3 Fitting a NAREN model for the wind data

Based on the method described in section 5.3.2, we fit a UARE model for each node, explained in (5.17), and find that the combination of $p_{i,x}=5$ and $p_{i,y}=1$ results in minimal AIC values in most of the nodes. Let $p_x=5$ and $p_y=1$, and for simplicity, we use the global autoregressive parameter α_j for all nodes to fit NAREN models, with all possible combinations of $\mathbf{s_x}$ and $\mathbf{s_y}$ up to five for each entry, which is the maximum neighbour-stage for this network. The results shows that the NAREN $(5,[1,3,2,5,2],1,0)$ model gives the smallest AIC value. Hence, the best fitted NAREN model of the wind speed $X_{i,t}$ at node i is

$$X_{i,t}=\sum_{j=1}^{5}\left(\alpha_j X_{i,t-j}+\sum_{r=1}^{s_{x,j}}\beta_{j,r}\sum_{q\in\mathcal{N}^{(r)}(i)}w_{r,q}(i)X_{q,t-j}\right)+\theta_1 Y_{i,t-1}+\epsilon_{i,t}, \tag{5.22}$$

where $s_{x,1}=1, s_{x,2}=3, s_{x,3}=2, s_{x,4}=5$, $s_{x,5}=2$, and $Y_{i,t-1}$ represents the wind direction for node i at time $t-1$.

The following table displays the estimated parameters for the fitted model, which indicates how the time is linking the network for the wind speed. For example, the wind speed of immediate neighbours one hour ago and up to the fifth stage-neighbours four hours ago, etc. has influence on the current wind speed. This is reasonable for wind data, as it takes time (e.g. four hours in this case) for wind of the fifth stage-neighbours to come to the target locations (node i). Also, further stage-neighbours (i.e. larger than five) and the stage-neighbours of wind directions will not affect the current wind speed.

	time lag	α_j	$\beta_{j,1}$	$\beta_{j,2}$	$\beta_{j,3}$	$\beta_{j,4}$	$\beta_{j,5}$	θ_j
$X_{i,j}$	$j=1$	513	336	-	-	-	-	-
	$j=2$	78	-80	123	81	-	-	-
	$j=3$	334	-108	-55	-	-	-	-
	$j=4$	-115	-77	-15	-86	18	118	-
	$j=5$	56	14	-36	-	-	-	-
$Y_{i,j}$	$j=1$	-	-	-	-	-	-	-1

Table 5.3: Parameter estimates ($\times 1000$) for NAREN $(5,[1,3,2,5,2],1,0)$ model.

5.4.2.4 Forecasting performance comparisons

When comparing the forecasting performance with other time series models for the wind data, we do not use the NAREN $(5,[1,3,2,5,2],1,0)$ model, as the best-fitted model may not always have the best forecasting performance. Hence, some simpler models will be applied to the wind data. To be specific, we fit the GNAR $(2,\mathbf{0})$, GNAR $(2,[1,1])$ and NAREN $(2,[1,1],2,[2,1])$ models with

the following formula:

$$\begin{aligned} X_{i,t} =& \alpha_1 X_{i,t-1} + \beta_{1,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{1,q}(i) X_{q,t-1} + \\ & \alpha_2 X_{i,t-2} + \beta_{2,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{1,q}(i) X_{q,t-2} + \\ & \theta_1 Y_{i,t-1} + \phi_{1,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{1,q}(i) Y_{q,t-1} + \phi_{1,2} \sum_{q \in \mathcal{N}^{(2)}(i)} w_{2,q}(i) Y_{q,t-1} + \\ & \theta_2 Y_{i,t-2} + \phi_{2,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{1,q}(i) Y_{q,t-2} + \epsilon_{i,t}, \end{aligned} \tag{5.23}$$

where $\mathbf{0}$ represents a vector of two zeros and for node i at time t, $\{X_{i,t}\}$ are the wind speeds and $\{Y_{i,t}\}$ are the exogenous variables representing wind direction. Here, the global autoregressive parameters α_1 and α_2 are used instead of different parameters for different nodes. This is because when we fit the NAREN $(2,[1,1],2,[2,1])$ model using different autoregressive parameters for each node i, we find the estimates $\alpha_{i,j}$ with $j = 1,2$ are concentrated within a small range. Table 5.4 shows the lower and upper quantiles for the estimates $\alpha_{i,1}$ and $\alpha_{i,2}$ respectively.

	$\hat{\alpha}_{i,1}$	$\hat{\alpha}_{i,2}$
Lower quantile	0.5454	0.1038
Upper quantile	0.6299	0.1753

Table 5.4: the lower and upper quantiles for the estimates $\alpha_{i,1}$ and $\alpha_{i,2}$ in NAREN $(2,[1,1],2,[2,1])$ model.

Tables 5.5 shows the estimates with their standard errors and confidence intervals, AIC values and RMSFE for the three models that we use to fit the wind speed, where we can see that adding the neighbourhood speed information will make the model better fitted, as the AIC value decreases by 7.60% when using GNAR $(2,[1,1])$ model instead of GNAR $(2,\mathbf{0})$ model. Also, GNAR

$(2,[1,1])$ model increases the forecasting performance a bit as its RMSFE decreases by 4.88% compared with the GNAR $(2,\mathbf{0})$ model. However, our NAREN $(1,[1,1],1,[2,1])$ model does not have too much improvement in terms of fitness and forecasting performance, with a decrease of 0.45% for AIC and 0.32% for RMSFE. This may be because the previous wind direction in the network does not have too much influence on the current wind speed. It is reasonable as the data records instant wind direction at each node which may change during time and when the wind travels to other locations.

5.4.3 COVID-19 Data Example

Although COVID-19 was discovered fewer than two years (i.e. the first case was detected in the late December 2019), the research related to COVID-19 has become extensive since the outbreak of the pandemic. For example, Omori et al. (2020) investigate the ascertainment rate of novel coronavirus disease in Japan; Pham (2020) estimates the number of deaths related to COVID-19 in United States using a modified logistic fault-dependent detection model; Chu (2021) uses the Susceptible-Infectious-Recovered model and the log-linear regression model to analyse the incidence of the disease in Italy and Spain, etc.

Our second example is based on the coronavirus (COVID-19) data provided by the official UK government website:
`https://coronavirus.data.gov.uk/details/cases`.

As an easily-spread communicable disease, COVID-19 will sometimes cause

	GNAR (2, **0**)			GNAR (2,[1, 1])			NAREN (2,[1, 1],2,[2, 1])		
	Estimate	s.e.($\times 10^{-3}$)	95% CI	Estimate	s.e.($\times 10^{-3}$)	95% CI	Estimate	s.e.($\times 10^{-3}$)	95% CI
$\hat{\alpha}_1$	0.666	4.069	(0.658, 0.673)	0.565	4.228	(0.557, 0.574)	0.578	4.625	(0.569, 0.587)
$\hat{\beta}_{1,1}$	-	-	-	0.326	9.131	(0.308, 0.344)	0.333	9.693	(0.314, 0.352)
$\hat{\alpha}_2$	0.323	4.072	(0.315, 0.331)	0.246	4.225	(0.238, 0.255)	0.241	4.625	(0.231, 0.250)
$\hat{\beta}_{2,1}$	-	-	-	-0.138	9.136	(-0.156, -0.120)	-0.166	9.695	(-0.185, -0.147)
$\hat{\theta}_1$	-	-	-	-	-	-	-0.015	1.747	(-0.018, -0.011)
$\hat{\phi}_{1,1}$	-	-	-	-	-	-	-0.024	3.652	(-0.032, -0.017)
$\hat{\phi}_{1,2}$	-	-	-	-	-	-	0.025	2.563	(0.020, 0.030)
$\hat{\theta}_2$	-	-	-	-	-	-	0.004	1.745	(0.001, 0.008)
$\hat{\phi}_{2,1}$	-	-	-	-	-	-	0.020	3.600	(0.013, 0.027)
AIC	51963			48015			47798		
RMSFE	0.3282			0.3122			0.3112		

Table 5.5: Simulation results of GNAR (2, **0**), GNAR (2,[1, 1]) and NAREN (1,[1, 1], 1, [2, 1]) models for the wind data, where "s.e." and "CI" in the table represents the standard error and confidence interval respectively.

serious respiratory tract infections, although some statistics shows that most infections are mild (Hoseinpour Dehkordi et al., 2020). If there are a large number of serious patients, who are admitted by hospitals, then the national health service in UK will face an increasing burden. Hence, our purpose is to model the numbers of patients in mechanical ventilation beds using our NAREN model, where the exogenous variable is the number of new hospital admissions. A proper multivariate time series model will give an idea of how to allocate medical resources and help make local policies.

5.4.3.1 Exploratory data analysis

The COVID-19 data is available online up to date, however, we only focus on the time period from 01/09/2020 to 01/02/2021, when we can see a large number of new cases reported everyday as shown in Figure 5.6. Although the number of confirmed cases is considerable in some later periods, as vaccinations were developed and administered since late December 2020, more recent cases tend to be mild and the number of patients admitted by hospitals is no longer at such a high level.

To obtain the number of confirmed COVID-19 patients and the number of patients in mechanical ventilation beds, each hospital trust needs to report the number daily at 8am and the UK figure is the sum of the four nations' figures, which can only be calculated when all nations' data are available. The details of the information about the COVID data is provided by the official UK government website:
`https://coronavirus.data.gov.uk/details/about-data`.

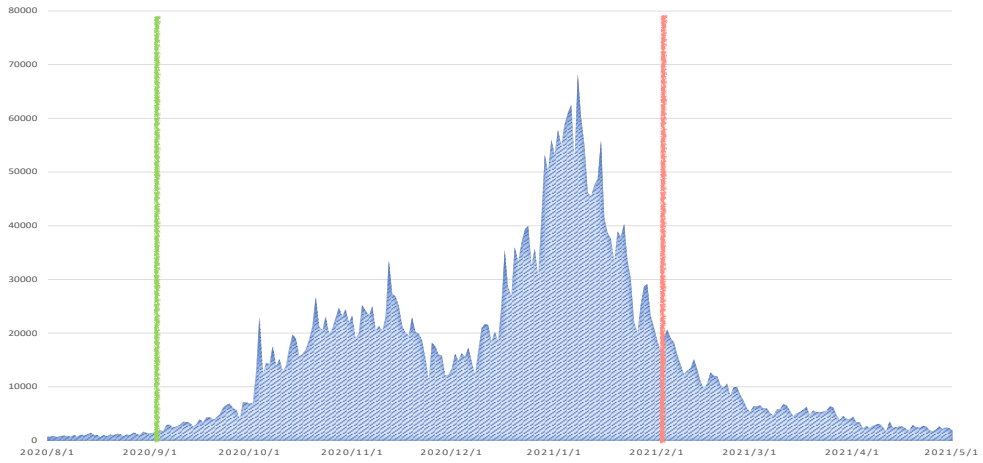


Figure 5.6: Daily new COVID-19 cases reported in UK from 01/08/2020 to 01/05/2021, where the green line and the red line represent the starting and end date of our target network time series.

Before fitting our NAREN model, we have to clean the data by deleting the hospitals with all zero patients for all time periods. Also, some incorrect or misleading information needs to be amended. For example, Rotherham NHS Foundation and Rotherham, Doncs and South Humber are 10 miles away, but they have the same postcode, which will cause problems when constructing a network. Hence, we manually enter the actual distance between these two places. After cleaning the data, we have 132 nodes (NHS trusts) with 154 time periods.

NHS trusts often consist of more than one hospital especially for some large areas, which may have greater number of confirmed patients and patients in mechanical ventilation beds than the small local areas. However, the network structures link all the NHS trusts, where the trusts with small number of patients may also be influential to our NAREN model. When fitting the network

time series models, we use the ordinary number of patients, which could also be normalized by the population size. Further studies could also be carried out in this area.

5.4.3.2 Network construction

The next step is to construct a network for the targeted hospitals according to their geographic locations. We use the minimum spanning tree method described in section 5.4.2.2 to obtain the network shown in Figure 5.7.

5.4.3.3 Forecasting performance comparisons

Now we want to compare the model forecasting performance for different multivariate time series models. The VARX model is not considered here as we do not have enough time observations for such a large network with 132 nodes, which is a major limitation of the VARX model and an advantage of our NAREN model. Let $\mathbf{0}$ be a vector of three zeros, we now fit a GNAR $(3,\mathbf{0})$, GNAR $(3,[1,0,0])$ and NAREN $(3,\mathbf{0},2,[1,0])$ models with the following formula:

$$\begin{aligned} X_{i,t} =& \alpha_1 X_{i,t-1} + \beta_{1,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{1,q}(i) X_{q,t-1} + \alpha_2 X_{i,t-2} + \alpha_3 X_{i,t-3} + \\ & \theta_1 Y_{i,t-1} + \phi_{1,1} \sum_{q \in \mathcal{N}^{(1)}(i)} w_{1,q}(i) Y_{q,t-1} + \theta_2 Y_{i,t-2} + \epsilon_{i,t}, \end{aligned} \tag{5.24}$$

where for node i at time t, $\{X_{i,t}\}$ are the numbers of patients in mechanical ventilation beds and $\{Y_{i,t}\}$ are the exogenous variables representing the new admissions into hospitals.

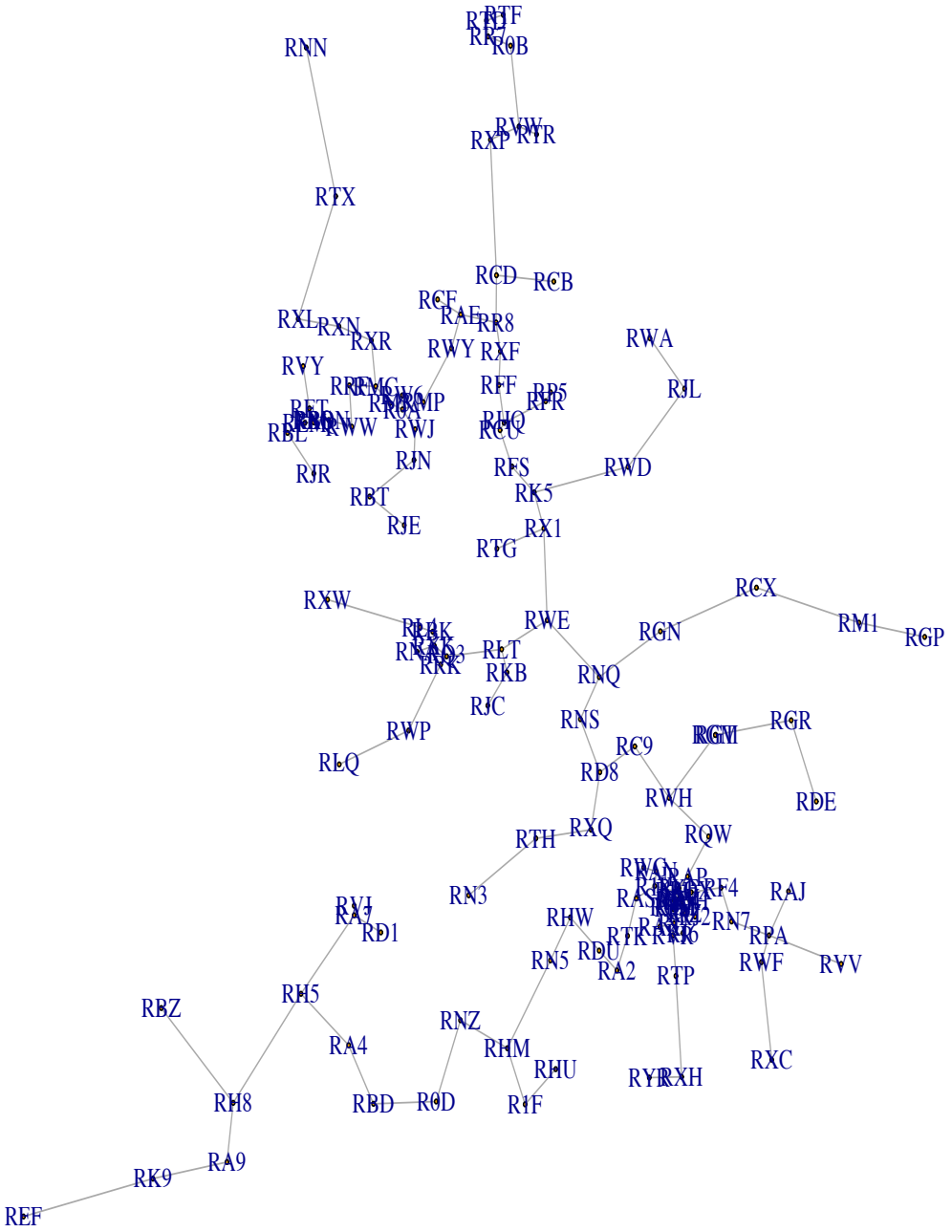


Figure 5.7: Network constructed for the targeted hospitals according to their geographical locations, where the nodes are labelled with their unique internal codes.

The logarithmic transform shown in (5.21) in wind application is applied to our two network time series when fitting network models as the forecasting performance after transformation is better than the non-transformed data.

Here, we also use global autoregressive parameters $\{\alpha_j\}_{j=1}^{3}$ while fitting both GNAR and NAREN models, as the autoregressive parameters for most of the nodes are concentrated within a small range when using different $\{\alpha_{i,j}\}$ parameters to fit models. Table 5.6 shows the lower and upper quantiles for the estimates of $\alpha_{i,1}$, $\alpha_{i,2}$ and $\alpha_{i,3}$ respectively, for $i = 1,...,132$.

	$\hat{\alpha}_{i,1}$	$\hat{\alpha}_{i,2}$	$\hat{\alpha}_{i,3}$
Lower quantile	0.7077	0.01398	0.01496
Upper quantile	0.9347	0.2667	0.2015

Table 5.6: the lower and upper quantiles for the estimates of $\alpha_{i,1}$, $\alpha_{i,2}$ and $\alpha_{i,3}$ in NAREN $(3,\mathbf{0},2,[1,0])$ model.

Tables 5.7 shows the parameter estimates together with standard errors and confidence intervals for the GNAR $(3,\mathbf{0})$, GNAR $(3,[1,0,0])$ and NAREN $(3,\mathbf{0},2,[1,0])$ models. When fitting our NAREN model, including the neighbour-stage parameter $\hat{\beta}_{1,1}$ will not improve the forecast performance, which is only considered in the GNAR $(3,[1,0,0])$ model in order to make comparison with the GNAR $(3,\mathbf{0})$ model. In addition, although the parameter $\hat{\phi}_{1,1}$ is not very significant (i.e. significant under 10%), adding this term will lower the RMSFE for our NAREN model.

Table 5.7 also displays the AIC values and RMSFE for the three models that we use to fit the COVID data. The results indicate that our NAREN $(3,\mathbf{0},2,[1,0])$

	GNAR $(3,\mathbf{0})$			GNAR $(3,[1,0,0])$			NAREN $(3,\mathbf{0},2,[1,0])$		
	Estimate	s.e.($\times 10^{-3}$)	95% CI	Estimate	s.e.($\times 10^{-3}$)	95% CI	Estimate	s.e.($\times 10^{-3}$)	95% CI
$\hat{\alpha}_1$	0.648	7.033	(0.634, 0.661)	0.638	7.038	(0.624, 0.652)	0.601	6.994	(0.587, 0.615)
$\hat{\beta}_{1,1}$	-	-	-	0.026	1.952	(0.022, 0.030)	-	-	-
$\hat{\alpha}_2$	0.216	8.365	(0.200, 0.233)	0.211	8.337	(0.194, 0.227)	0.199	8.166	(0.183, 0.215)
$\hat{\alpha}_3$	0.141	7.158	(0.127, 0.155)	0.134	7.149	(0.119, 0.148)	0.120	7.004	(0.107, 0.134)
$\hat{\theta}_1$	-	-	-	-	-	-	0.017	3.369	(0.010, 0.024)
$\hat{\phi}_{1,1}$	-	-	-	-	-	-	0.003	1.909	(-0.001, 0.007)
$\hat{\theta}_2$	-	-	-	-	-	-	0.064	3.413	(0.057, 0.070)
AIC	4003			3827			2948		
RMSFE	0.1906			0.1892			0.1805		

Table 5.7: Simulation results of GNAR $(3,\mathbf{0})$, GNAR $(3,[1,0,0])$ and NAREN $(3,\mathbf{0},2,[1,0])$ models for the COVID data, where "s.e." and "CI" in the table represents the standard error and confidence interval respectively.

model is better fitted, which includes the number of new admissions in hospitals up to two previous time-lags as well as the one-stage neighbours' new patients. Compared to the GNAR $(3,\mathbf{0})$ and GNAR $(3,[1,0,0])$ model, the AIC value of our NAREN model decreases by 26.4% and 23.4% respectively. Also, in terms of the forecasting performance, our NAREN $(3,\mathbf{0},2,[1,0])$ model reduces the RMSFE values by 5.25% for GNAR $(3,\mathbf{0})$ and 4.54% for GNAR $(3,[1,0,0])$. In general, our NAREN $(3,\mathbf{0},2,[1,0])$ model has an improvement on both fitting and forecasting performance compared to the other two GNAR models.

The results above coincide with the real situation, as an increase in the number of admissions in hospitals will always lead an increase in the numbers of patients in mechanical ventilation beds due to the deterioration of the disease in some people, which always include a time delay. In addition, an increase of previous admissions in neighbours' hospitals may affect the current occupation of the mechanical ventilation beds, as patients may be transferred to other hospitals if the current one's resources are limited and/or the infection speeds.

5.5 Conclusions

This chapter introduces our NAREN model, which extends the generalised network autoregressive model (GNAR) by adding the exogenous network terms. When dealing with the real data, our extended model sometimes has a noticeable improvement compared to the GNAR model, which is heavily dependent on the relationship of the two sets of time series data. As a special case of

the VARX model, our NAREN model not only captures the structure of the network and incorporates the neighbours' information, but also is more efficient and applicable in the real data analysis as fewer parameters are required to be estimated. However, we did not compare the forecast performance between our NAREN model and the GNARX model proposed by Nason and Wei (2021), which is recently published towards the completion of the thesis, further comparison between these two models could be carried out.

CHAPTER 6

CONCLUSIONS AND DISCUSSIONS

This section summarises main ideas and achievements of our new methodology in this thesis, and discusses possible further research of our current work.

6.1 Yet Another Multivariate Median

Key contributions

- Proves theoretical equivalence of yamm and the projection median.
- Demonstrates the robustness of yamm on a simple bivariate setup.
- Illustrates three computation methods for the projection median and introduces the `Yamm` package to compute the projection median.
- Shows empirically that the spatial median and the projection median perform well compared to other medians using the existing R functions.

Further investigation

As we only show the the robustness of yamm on a simple bivariate setup, a more general cases could be investigated in the future both theoretically and

numerically. In addition, the function `PmedTrapz` in our `Yamm` package is only valid for computing the projection median in $\mathbb{R}^2$ and $\mathbb{R}^3$ currently, which uses the trapezoidal rule to produce accurate results without excessive running time. A generalisation of this R function to higher dimensions could be considered, which should be computational efficient in practice as well.

6.2 Density and Hazard Rate Estimation using a Bayesian Wavelet Approach

Key contributions

- Uses Bayesian wavelet method to improve hazard rate estimates, where the survival function is estimated by a Dirichlet process prior. Better performance in simulation examples compared to the presmoothed method.

- Uses the detailed covariance structure of the empirical wavelet coefficients in the Bayesian wavelet thresholding approach to estimate the density function, where the "mixture of Gaussians" prior distribution is applied. An improvement is presented empirically in terms of the mean squared error.

Further investigation

When estimating the hazard rate, the number of bins is suggested to be approximately $2^{\text{floor}\{\log(n)-2\}}$ for n observations, which is a rough good choice. A further work could be based on the theory of choosing the number of bins. When applying the Bayesian wavelet shrinkage for the empirical coefficients

using the "mixture of Gaussians" prior distribution to estimate the density function, we try to estimate the hyperparameters by maximising marginal likelihood distribution of the empirical coefficients numerically. It is too computational expensive and currently, we determine the hyperparameters based on the rough suggestions of Chipman et al. (1997). Further investigation can be carried out on choosing the hyperparameters efficiently and accurately using appropriate theoretical or numerical methods.

6.3 Survival Estimation with Networks

Key contributions

- Introduces a new method to estimate survival functions for groups of individuals with recurrent survival lifetimes using their network structure.
- Shows a better performance in the Weibull distribution simulations when the number of individuals considered is small.

Further investigation

Our simulation examples show that sometimes both our top-down and bottom-up approaches can not obtain groups matching the truth exactly. Further study could focus on other alternative methods to improve the cluster classification procedures, such as clustering individuals using top-down and bottom-up approaches together at the same time. In addition, our new method to improve survival estimates is only applied to the simulation examples, as currently we are not able to find the appropriate real datasets with both enough covariates information to classify different clusters and recurrent lifetimes. It is vital to

apply our method to real situations, which can be considered in the future.

6.4 Network Autoregressive Processes With Exogenous Network Time Series

Key contributions

- Generalises the GNAR model to our NAREN model by including the exogenous variables of another network time series.
- Displays a better POOSF performance in both simulation and real examples compared to the GNAR models.

Further investigation

Like the GNAR model, further work could be adding covariates on edges or nodes of the network in our NAREN model. In section 5.3.2, we fit UARE models for each node and use the modal autoregressive time lags with minimum AIC values as the estimates of the time lags of our NAREN model, further study on using alternative methods to set appropriate time lags can be carried out. When comparing the forecasting performance between network time series models, we use simple comparative benchmarks, which could be invested further on some other alternatives. Also, comparing the performance between our NAREN model and the GNARX model is also an interesting topic for the future work. As mentioned in section 5.4.3.1, using normalised number of patients is also an alternative to analyse the COVID application, which has not been done in our experiment.

APPENDIX A

SUPPLEMENTARY MATERIAL FOR CHAPTER 2

A.1 Simulation Performance for High-Dimensional Medians

Median		$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial	mean	28	30	29	29	35
	s.d.	45	46	45	46	48
Component-wise	mean	27	28	34	31	32
	s.d.	44	45	47	46	47
Tukey's	mean	97	430	670	1170	2160
	s.d.	20	47	50	74	130
Oja's	mean	2240	2270	2220	2290	2650
	s.d.	530	630	470	640	6330
Projection	mean	24	83	200	470	1030
	s.d.	43	37	22	49	52

Table A.1: Mean and standard deviation (s.d.) of the three-dimensional medians' operation time ($\times 10^{-5}$) in seconds using 1000 datasets generated from Laplace distribution with different numbers of observations (k), where R functions stated in Table 2.1 are used.

Location Estimator	$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial Median	57	20	9.0	4.2	2.1
Component-wise Median	70	27	12.0	5.6	2.8
Tukey's Median	66	21	9.6	4.4	2.2
Oja's Median	68	23	11.0	5.6	3.6
Projection Median	58	21	9.1	4.2	2.1
Mean	97	40	19.0	9.7	5.1

Table A.2: Mean squared error ($\times 10^{-2}$) of three-dimensional medians with 1000 sets of data generated from Laplace distribution.

Median		$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial	mean	36	30	39	40	44
	s.d.	48	46	49	49	50
Component-wise	mean	36	31	36	37	40
	s.d.	48	46	48	48	49
Tukey's	mean	200	690	930	1510	2680
	s.d.	22	42	65	110	180
Oja's	mean	6520	4620	4360	4380	4390
	s.d.	4630	1700	1450	1310	1410
Projection	mean	590	670	830	1200	1930
	s.d.	47	49	48	45	47

Table A.3: Mean and standard deviation (s.d.) of the five-dimensional medians' operation time ($\times 10^{-5}$) in seconds using 1000 datasets generated from Laplace distribution with different numbers of observations (k), where R function `PmedMCInt` are used to produce the projection median, since `PmedTrapz` is only valid in $\mathbb{R}^2$ and $\mathbb{R}^3$.

Location Estimator	$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial Median	54	19	8.2	3.9	1.9
Component-wise Median	74	28	11.7	5.6	2.7
Tukey's Median	63	20	8.7	4.1	2.0
Oja's Median	190	23	11.0	6.6	4.8
Projection Median	57	20	8.5	4.0	1.9
Mean	100	42	19	9.9	4.9

Table A.4: Mean squared error ($\times 10^{-2}$) of five-dimensional medians with 1000 sets of data generated from Laplace distribution.

Median		$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial	mean	53	48	60	59	66
	s.d.	50	50	49	49	48
Component-wise	mean	53	47	53	60	64
	s.d.	50	50	50	49	48
Oja's	mean	5250	17550	14780	13110	12750
	s.d.	150	9570	7550	4940	5250
Projection	mean	1150	1240	1390	1770	2540
	s.d.	77	53	62	53	140

Table A.5: Mean and standard deviation (s.d.) of the ten-dimensional medians' operation time ($\times 10^{-5}$) in seconds using 1000 datasets generated from Laplace distribution with different numbers of observations (k), where R function `med` is not able to compute the Tukey's median when $n = 10$.

Location Estimator	$k = 10$	$k = 25$	$k = 50$	$k = 100$	$k = 200$
Spatial Median	52	18	8.0	3.8	1.9
Component-wise Median	73	27	11.7	5.6	2.8
Oja's Median	990	290	23	13	12
Projection Median	55	19	8.2	3.9	1.9
Mean	100	41	20	9.8	5.0

Table A.6: Mean squared error ($\times 10^{-2}$) of ten-dimensional medians with 1000 sets of data generated from Laplace distribution.

A.2 R software

Package 'Yamm'

April 3, 2020

Title Multivariate Methods Based on Projections and Related Concepts

Version 1.3.1

Date 2020-04-02

Depends R (>= 3.0), depth, OjaNP, pcaPP, interp

Suggests animation

Maintainer Guy Nason <g.nason@imperial.ac.uk>

Description Functionality to compute the projection median via several algorithms. This package also provides functions to plot different multivariate medians and multivariate quantiles in two-dimensional and three-dimensional data respectively. See Chen, F. and Nason, G.P. (2020) ``A new method for computing the projection median, its influence curve and techniques for the production of projected quantile plots." PLOS One (accepted for publication).

License GPL-2

NeedsCompilation yes

Author Fan Chen [aut],
Guy Nason [aut, cre]

Repository CRAN

Date/Publication 2020-04-03 16:30:02 UTC

R topics documented:

1

Yamm-package *Multivariate Methods Based on Projections and Related Concepts*

Description

This package provides functions for computing the projection median. PmedTrapz approximates the projection median by the trapezoidal rule, which is only valid for the two- and three-dimensional cases, while PmedMCInt use Monte Carlo approximation, and it is valid for any multivariate median. yamm provides another method to compute the projection median based on an optimiser technique. This package also provides functions for plotting different multivariate medians, such as the Spatial, Component-wise, Tukey's, Oja's median, etc., for randomly generated data sets in both the two-dimensional and three-dimensional cases. In addition, this package also allows users to produce the two-dimensional and three-dimensional quantile plots with function muqie and muqie3D respectively.

Details

The DESCRIPTION file:

Package:	Yamm
Title:	Multivariate Methods Based on Projections and Related Concepts
Version:	1.3.1
Date:	2020-04-02
Author:	Fan Chen [aut], Guy Nason [aut, cre]
Depends:	R (>= 3.0), depth, OjaNP, pcaPP, interp
Suggests:	animation
Maintainer:	Guy Nason <g.nason@imperial.ac.uk>
Description:	Functionality to compute the projection median via several algorithms.
License:	GPL-2

Index of help topics:

Plot2dMedian	Plot Two-dimensional Medians
Plot3dMedian	Plot Three-dimensional Medians
PmedMCInt	Projection Median Approximated by Monte Carlo Integration
PmedTrapz	Projection Median Approximated by Trapezoidal Rule
Yamm-package	Multivariate Methods Based on Projections and

```
                    Related Concepts
beetle              Six Measurements of Beetles
clusters2d          Three Clusters of 2-dimensional Data
clusters3d          Four Clusters of 3-dimensional Data
makeplot            Plot Two-dimensional Quantile
makeplot3D          Plot Three-dimensional Quantile
muqie               Two-dimensional Quantile
muqie3D             Three-dimensional Quantile
yamm                Yet Another Multivariate Median
yamm.obj            Objective Function for Yamm
```

Author(s)

NA

Maintainer: Guy Nason <g.nason@imperial.ac.uk>

References

Basu, R., Bhattacharya, B.B., and Talukdar, T. (2012) The projection median of a set of points in Rd *CCCG.*, **47**, 329-346. doi: 10.1007/s0045401193806

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection median, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

Croux, C., Filzmoser, P., and Oliveira, M., (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, **87**, 218-225.

Durocher, S. and Kirkpatrick, D. (2009), The projection median of a set of points, *Computational Geometry*, **42**, 364-375.

Fischer,D., Mosler, K., Mottonen, J.K., Nordhausen, K., Pokotylo, O., and Vogel, D. (2016) Computing the Oja Median in R: The Package OjaNP, *ArXiv:1606.07620*

Rousseeuw, P.J. and Ruts, I. (1996), Algorithm AS 307: Bivariate location depth, *Appl. Stat.-J. Roy. St. C*, **45**, 516-526.

Rousseeuw, P.J. and Ruts, I. (1998), Constructing the bivariate Tukey median, *Stat. Sinica*, **8**, 828-839.

Rousseeuw, P.J., Ruts, I., and Tukey, J.W. (1999), The Bagplot: A Bivariate Boxplot, *The Am. Stat.*, **53**, 382-387.

Struyf, A. and Rousseeuw, P.J. (2000), High-dimensional computation of the deepest location, *Comput. Statist. Data Anal.*, **34**, 415-436.

See Also

yamm, PmedTrapz, PmedMCInt,

Examples

```
# Load a 2-dimensional data set.
data(clusters2d)
#
# Set seed for reproduction.
```

4 *beetle*

```
set.seed(5)
#
# Projection median approximated by Monte Carlo Integration.
PmedMCInt(clusters2d, nprojs = 30000)
# [1]  4.3369501 -0.1578591
#
#
# Projection median approximated by the trapezoidal rule.
PmedTrapz(clusters2d,no.subinterval=180)
# [1]  4.1556553 -0.3566614
#
#
# Yamm.
set.seed(5)
yamm(clusters2d,nprojs = 2500,reltol=1e-3,doabs=1,full.results=FALSE)
# [1]  4.3871582 -0.1070497
#
#
# Plot 2-D medians
# Remove the outliers of the dataset.
cluster_without_outlier <- clusters2d[c(1:101),]
myxvec <- c(min(cluster_without_outlier[,1]),
          max(cluster_without_outlier[,1]))
myyvec <- c(min(cluster_without_outlier[,2]),
          max(cluster_without_outlier[,2]))
#
# Plot the figure.
set.seed(5)
Plot2dMedian(clusters2d, myxvec, myyvec, yamm.nprojs = 2000,
             PmedMCInt.nprojs = 20000, no.subinterval = 36,
             opt.method = "BFGS", xlab = "Component1",
             ylab = "Component2")
```

`beetle` *Six Measurements of Beetles*

Description

Multivariate dataset containing six measurements on each of three species of flea-beetles: concinna, heptapotamica, and heikertingeri. The original data set contains one column identifying the species of the observations, which is irrelevant and has been deleted here.

Usage

```
data("beetle")
```

clusters2d 5

Format

A data frame with 74 observations on the following 6 variables.

`tars1` Width of the first joint of the first tarsus in microns (the sum of measurements for both tarsi).

`tars2` The same for the second joint.

`head` The maximal width of the head between the external edges of the eyes in 0.01 mm.

`aede1` The maximal width of the aedeagus in the fore-part in microns.

`aede2` The front angle of the aedeagus (1 unit = 7.5 degrees).

`aede3` The aedeagus width from the side in microns.

Source

Lubischew, A.A.(1962) On the Use of Discriminant Functions in Taxonomy, *Biometrics*,**18**, 455-477.

References

Cook, D.H. and Swayne, D.F. (2007). Interactive and Dynamic Graphics for Data Analysis: With Examples Using R and GGobi. http://www.ggobi.org/book/data/flea.xml

Examples

```
data(beetle)
```

`clusters2d` *Three Clusters of 2-dimensional Data*

Description

This dataset with 103 observations contains three clusters, which are generated from different independent normal distributions randomly, and two outliers (located in the last two rows).

Usage

```
data("clusters2d")
```

Format

The first cluster has 26 observations, and the two variables are generated from $N(3,1)$ and $N(4,1)$ respectively. The second cluster has 36 observations, and the two variables are generated from $N(10,1.5)$ and $N(-2,1.5)$ respectively. The third cluster has 39 observations, and the two variables are generated from $N(2,0.5)$ and $N(-2,0.5)$ respectively. The two outliers are $c(100.3,99.1)$ and $c(97.5,98.4)$.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

Examples

```
data(clusters2d)
```

`clusters3d` *Four Clusters of 3-dimensional Data*

Description

This dataset with 105 observations contains four clusters, which are generated from different Laplace distributions randomly, and five outliers (located in the last five rows).

Usage

```
data("clusters3d")
```

Format

The four clusters are generated from different multivariate Laplace distributions. The first cluster has 20 observations, where the mean values μ of the Laplace distribution are equal to $(-8, -8, -8)$ and the covariance matrix Σ is the product of two times identity matrix. The second cluster has 35 observations, where $\mu = (-5, 5, 5)$ and Σ is the identity matrix. The third cluster has 30 observations, where $\mu = (12, -12, 12)$ and Σ is the identity matrix. The fourth cluster has 30 observations, where $\mu = (18, 18, -18)$ and Σ is the identity matrix. The five outliers are from the $\mu = (100, 100, -100)$ and Σ is the product of ten times identity matrix.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

Examples

```
data(clusters3d)
```

`makeplot` *Plot Two-dimensional Quantile*

Description

This function calls `muqie` for multiple values of quantiles from 0.5 to 0.95 and then produces a set of plots with these quantiles for producing an animated GIF using package **animation**.

Usage

```
makeplot(xdata, dm=c(1,2), nsegs=20,
         quantile.increment= 0.001,
         nprojs=2000, reltol=0.001)
```

Arguments

`xdata`	The data as a matrix or dataframe with the number of columns greater than or equal to two, with each row being viewed as one multivariate observation.
`dm`	A numeric vector with two entries representing the selected columns of the data considered. The default value is $c(1, 2)$, which means the first two columns of data are chosen if the dimension of data is greater than two.
`nsegs`	The number of the unit-length direction vectors u, which is computed by dividing a unit circle into `nsegs` equal sectors.
`quantile.increment`	A numeric value specifies the increment of the set of different quantiles.
`nprojs`	The number of projections for the dataset when computing `yamm`. The default value is 2000.
`reltol`	The tolerance of the optimisation process in the function `yamm`. The default value is 0.001.

Value

This function returns a set of plots with various specified quantiles.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

`yamm` `muqie`

Examples

```
# Load a data frame with 103 rows and 2 columns.
# The last two rows of the data are the outliers.
data(clusters2d)
#
# Remove the outliers of the dataset.
cluster_without_outlier <- clusters2d[c(1:101),]
#
# Produce an animation of a set of multivariate quantile plots.

if (requireNamespace("animation")) {

library("animation")
# Generate temporary file
f <- tempfile(fileext=".gif")
#
# Now generate movie into the temporary file.
# Here nprojs=40: for a real example, for production quality you should increase
# it to 1000, 2000 or even higher
#
# Here quantile.increment=0.1, for production quality this should be reduced to
# e.g. 0.01, of even smaller
#
saveGIF(makeplot(cluster_without_outlier, nprojs=40, quantile.increment=0.1),
diff.col=3, interval=0.1,width=500, height=500, movie.name=f)
cat("Movie saved to: ", f, "\n")
}
```

`makeplot3D` *Plot Three-dimensional Quantile*

Description

This function calls `muqie3D` for multiple values of quantiles from 0.5 to 0.95 and then produces a set of perspective plots of a surface over the x-y plane with these quantiles, which are used to produce an animated GIF using package **animation**.

Usage

```
makeplot3D(xdata, dm=c(1,2,3), nsegs=30,
         quantile.increment= 0.005,
         nprojs=2000, reltol=0.001)
```

Arguments

`xdata`	The data as a matrix or dataframe with the number of columns greater than or equal to three, with each row being viewed as one multivariate observation.

`dm` A numeric vector with three entries representing the selected columns of the data considered. The default value is $c(1, 2, 3)$, which means the first three columns of data are chosen if the dimension of data is more than three.

`nsegs` The number of the three-dimensional unit-length direction vectors `u`, which is computed by dividing a unit sphere into `nsegs` equal sectors.

`quantile.increment` A numeric value specifies the increment of the set of different quantiles.

`nprojs` The number of projections for the dataset when computing `yamm`. The default value is 2000.

`reltol` The tolerance of the optimisation process in the function `yamm`. The default value is 0.001.

Value

This function returns a set of perspective plots of a surface over the x-y plane with various specified quantiles.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

`yamm` `muqie3D`

Examples

```
#
data(beetle)
#
# Produce an animation of a set of multivariate quantile plots.
if (requireNamespace("animation")) {

library("animation")
# Generate temporary file
f <- tempfile(fileext=".gif")
#
# Now generate movie into the temporary file.
# Here nprojs=40: for a real example, for production quality you should increase
# it to 1000, 2000 or even higher
#
# Here quantile.increment=0.1, for production quality this should be reduced to
# e.g. 0.01, of even smaller
#
saveGIF(makeplot3D(beetle, dm=c(1,3,6), nprojs=40, quantile.increment=0.1),
diff.col=3, interval=0.1,width=500, height=500, movie.name=f)
cat("Movie saved to: ", f, "\n")
}
```

`muqie` *Two-dimensional Quantile*

Description

This function plots the collection of all MUltivariate QuantIlE points in two dimensions (`muqie`) over all unit-length direction vectors `u`, which projects the `yamm`-centred multivariate data onto the chosen vector `u` to obtain a univariate set. The `muqie` point is merely the vector `u` rescaled to have length equal to the quantile of the univariate set.

Usage

```
muqie(xdata, dm=c(1,2), probs=0.5, nsegs=20,
      nprojs=2000, reltol=0.001, plot.it=FALSE,
      full.return=FALSE, xlab=NULL, ylab=NULL)
```

Arguments

`xdata`	The data as a matrix or dataframe with the number of columns greater than or equal to two, with each row being viewed as one multivariate observation.
`dm`	A numeric vector with two entries representing the selected columns of the data considered. The default value is $c(1, 2)$, which means the first two columns of data are chosen if the dimension of data is more than two.
`probs`	The quantile of the data after projected to obtain a univariate set.
`nsegs`	The number of the two-dimensional unit-length direction vectors `u`, which is computed by dividing a unit circle into `nsegs` equal sectors.
`nprojs`	The number of projections for the dataset when computing `yamm`. The default value is 2000.
`reltol`	The tolerance of the optimisation process in the function `yamm`. The default value is 0.001.
`plot.it`	Logical. If `TRUE`, the function `muqie` will produce a two-dimensional quantile plot.
`full.return`	Logical. If `TRUE`, the function `muqie` will return a list of full results. See "Value".
`xlab`	x-axis label for the quantile plot.
`ylab`	y-axis label for the quantile plot.

Value

If `full.results` = `TRUE`, it returns a list comprising of

`ans`	A data matrix with four rows. The first row represents the angle of the unit-length projection vector `u` to the positive x-axis, while the second and third row are the x- and y-coordinates of the projection vector respectively. The last row is univariate quantile of the projected data matrix.

uvd	A data matrix after projecting the yamm-centred multivariate data onto a set of projection vectors u.
cdata	The yamm-centred multivariate data matrix.
yamm	The yamm value of the multivariate data. See yamm for more details.

If full.results = FALSE (default), it will only return ans.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

yamm

Examples

```
data(beetle)
#
# Compute the 0.7-quantile for the first two columns of the beetle data.
muqie(beetle,dm=c(1,4), probs=0.7)
```

---

muqie3D *Three-dimensional Quantile*

Description

This function plots the collection of all MUltivariate QuantIlE points in three dimensions (muqie3D) over all unit-length direction vectors u, which projects the yamm-centred multivariate data onto the chosen vector u to obtain a univariate set. The muqie3D point is merely the vector u rescaled to have length equal to the quantile of the univariate set.

Usage

```
muqie3D (xdata, dm=c(1,2,3), probs=0.5,
         nsegs=30, nprojs=2000, reltol=0.001,
         plot.it=FALSE, full.return=FALSE)
```

Arguments

xdata	The data as a matrix or dataframe with the number of columns greater than or equal to three, with each row being viewed as one multivariate observation.
dm	A numeric vector with three entries representing the selected columns of the data considered. The default value is $c(1, 2, 3)$, which means the first three columns of data are chosen if the dimension of data is more than three.
probs	The quantile of the data after projected to obtain a univariate set.

`nsegs`	The number of the three-dimensional unit-length direction vectors `u`, which is computed by dividing a unit sphere into `nsegs` equal sectors.
`nprojs`	The number of projections for the dataset when computing `yamm`. The default value is 2000.
`reltol`	The tolerance of the optimisation process in the function `yamm`. The default value is 0.001.
`plot.it`	Logical. If `TRUE`, the function `muqie` will produce a three-dimensional quantile plot.
`full.return`	Logical. If `TRUE`, the function `muqie` will return a list of full results. See "Value".

Value

If `full.results = TRUE`, it returns a list comprising of

`ans`	A data matrix with four rows. The first three rows represent the x-, y- and z-coordinates of the projection vector `u` respectively. The last row is univariate quantile of the projected data matrix.
`uvd`	A data matrix after projecting the `yamm`-centred multivariate data onto a set of projection vectors `u`.
`cdata`	The `yamm`-centred multivariate data matrix.
`yamm`	The yamm value of the multivariate data. See `yamm` for more details.

If `full.results = FALSE` (default), it will only return `ans`.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

`yamm`

Examples

```
data(beetle)
#
# Compute the 0.7-quantile for the first three columns of the beetle data.
muqie3D(beetle, dm=c(1,3,6), probs=0.7)
```

`Plot2dMedian` *Plot Two-dimensional Medians*

Description

This function plots various multivariate medians in the two-dimensional case. The grey dots presented in the figure are the data points and the Spatial, Component-wise (CWmed), Tukey's, Oja's, Liu's, Projection median as well as the mean value of the data set are plotted in the figure.

Usage

```
Plot2dMedian(data, xvec, yvec, yamm.nprojs = 2000,
             PmedMCInt.nprojs = 20000,
             no.subinterval = 36, opt.method = "BFGS",
             xlab = "Component1", ylab = "Component2")
```

Arguments

`data`	The data as a matrix or data frame, with each row being viewed as one multivariate observation.
`xvec`	A numeric vector containing the maximum and minimum values you desire for the x-axis.
`yvec`	A numeric vector containing the maximum and minimum values you desire for the y-axis.
`yamm.nprojs`	The number of projections for the dataset when computing `yamm`. The default value is 2000.
`PmedMCInt.nprojs`	The number of projections for the dataset when computing `PmedMCInt`. The default value is 20000, since `PmedMCInt` requires large number of projections while doing the Monte Carlo integration to ensure accuracy.
`no.subinterval`	The number of subintervals while using the trapezoidal rule to approximate the projection median with `PmedTrapz` function. The default value is 36, and small values (e.g. less than 10) of `no.subinterval` should not be used, to safeguard accuracy.
`opt.method`	The method chosen for the optimiser when computing the `yamm`, with default function "`BFGS`". `optim` is used to minimise the objective function `yamm.obj`. Apart from "`BFGS`", other functions in `optim` like "`Nelder-Mead`", "`CG`", "`L-BFGS-B`", and "`SANN`" can also be used.
`xlab`	Title for x-axis. Must be a character string.
`ylab`	Title for y-axis. Must be a character string.

Details

The Spatial median is obtained using `l1median` in the Rpackage **pacPP**. The Component-wise (CWmed), Liu's and Tukey's median are produced using function `med` in the Rpackage **depth**. Oja's median is produced using function `ojaMedian` in the Rpackage `OjaNP`. When computing the projection median, three approximations are implemented and diplayed in the plot, where `PmedMCInt` uses Monte Carlo method, `PmedTrapz` is computed by the trapezoidal rule, and `yamm` uses an optimiser.

The argument `xvec` and `yvec` are useful when there are outliers in the data set, which are not expected to be shown in the figure in some cases. Determining the x-axis and y-axis allows you to zoom in the plot and see the difference between multivariate medians and mean value.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

`PmedTrapz`, `PmedMCInt`, `yamm`, `yamm.obj`, `optim`.

Examples

```
# Load a data frame with 103 rows and 2 columns.
# The last two rows of the data are the outliers.
data(clusters2d)
#
# Remove the outliers of the dataset.
cluster_without_outlier <- clusters2d[c(1:101),]
myxvec <- c(min(cluster_without_outlier[,1]),
          max(cluster_without_outlier[,1]))
myyvec <- c(min(cluster_without_outlier[,2]),
          max(cluster_without_outlier[,2]))
#
# Plot the figure.
set.seed(5)
Plot2dMedian(clusters2d, myxvec, myyvec, yamm.nprojs = 2000,
             PmedMCInt.nprojs = 20000, no.subinterval = 36,
             opt.method = "BFGS", xlab = "Component1",
             ylab = "Component2")
```

`Plot3dMedian` *Plot Three-dimensional Medians*

Description

This function plots multivariate medians in the three-dimensional case. The grey dots presented in the figure are the data points and the Spatial, Component-wise (CWmed), Tukey's, Oja's, Liu's, Projection medians as well as the mean value of the data set are plotted in the figure.

Usage

```
Plot3dMedian(data, xvec, yvec, zvec, yamm.nprojs = 2000,
             PmedMCInt.nprojs = 20000, no.subinterval = c(18,36),
             opt.method = "BFGS", xlab = "Component1",
             ylab = "Component2", zlab = "Component3")
```

Arguments

`data`	The data as a matrix or data frame, with each row being viewed as one multivariate observation.
`xvec`	A numeric vector containing the maximum and minimum values you desire for the x-axis.
`yvec`	A numeric vector containing the maximum and minimum values you desire for the y-axis.
`zvec`	A numeric vector containing the maximum and minimum values you desire for the z-axis.
`yamm.nprojs`	The number of projections for the dataset when computing `yamm`. The default value is 2000.
`PmedMCInt.nprojs`	The number of projections for the dataset when computing `PmedMCInt`. The default value is 20000, since `PmedMCInt` requires large number of projections while doing the Monte Carlo integration to ensure accuracy.
`no.subinterval`	A numeric vector of two entries which represents the number of subintervals chosen while using the trapezoidal rule to approximate the projection median with `PmedTrapz` function. The default vector is $c(36, 36)$. Note small values (e.g. less than 10) for each entry of `no.subinterval` should not be used, to safeguard accuracy.
`opt.method`	The method chosen for the optimiser when computing the `yamm`, with default function "`BFGS`". `optim` is used to minimise the objective function `yamm.obj`. Apart from `BFGS`, other functions in `optim` like "`Nelder-Mead`", "`CG`", "`L-BFGS-B`", and "`SANN`" can also be used.
`xlab`	Title for x-axis. Must be a character string.
`ylab`	Title for y-axis. Must be a character string.
`zlab`	Title for z-axis. Must be a character string.

Details

The Spatial median is obtained using `l1median` in the Rpackage **pacPP**. The Component-wise (CWmed), and Tukey's median are produced using function `med` in the Rpackage **depth**. Oja's median is produced using function `ojaMedian` in the Rpackage `OjaNP`. Liu's median is not available in higher dimensions (> 2), so it is not shown here. When computing the projection median, three approximations are implemented and diplayed in the plot, where `PmedMCInt` uses Monte Carlo method, `PmedTrapz` is computed by the trapezoidal rule, and `yamm` uses an optimiser.

The argument `xvec`, `yvec` and `zvec` are useful when there are outliers in the dataset, which are not expected to be shown in the figure in some cases. Determining the x-axis y-axis, and z-axis allows you to zoom in the plot and see the difference between multivariate medians and mean value.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

PmedMCInt, PmedTrapz yamm, yamm.obj, optim.

Examples

```
# Load a data frame with 105 rows and 3 columns.
# The last five rows of the data are the outliers.
data(clusters3d)
#
# Remove the outliers of the dataset.
cluster_without_outlier <- clusters3d[c(1:100),]
myxvec <- c(min(cluster_without_outlier[,1]),
          max(cluster_without_outlier[,1]))
myyvec <- c(min(cluster_without_outlier[,2]),
          max(cluster_without_outlier[,2]))
myzvec <- c(min(cluster_without_outlier[,3]),
          max(cluster_without_outlier[,3]))
#
# Plot the figure.
set.seed(15)
Plot3dMedian(cluster_without_outlier, myxvec, myyvec, myzvec,
             yamm.nprojs = 2000, PmedMCInt.nprojs = 15000,
             no.subinterval = c(18,36),opt.method = "BFGS",
             xlab = "Component1",ylab = "Component2",
             zlab = "Component3")
```

PmedMCInt *Projection Median Approximated by Monte Carlo Integration*

Description

This function approximates the projection median using Monte Carlo integration, which can be used for any dimensions. PmedMCInt is implemented internally using C code CPmedMCInt and hence is much faster than coding with R only.

Usage

```
PmedMCInt(x, nprojs = 20000)
```

Arguments

`x`	The data as a matrix or data frame, with each row being viewed as one multivariate observation.
`nprojs`	The number of projections when using the Monte Carlo method to approximate the integration. The default value is 20000, since `PmedMCInt` requires large a number of projections to ensure the accuracy. More projections may increase the accuracy, as well as the running time.

Details

The projection median was introduced by Durocher and Kirkpatrick (2009) and generalised by Basu, Bhattacharya and Talukdar (2012). `PmedMCInt` produces the projection median using Monte Carlo approximation, which is valid in any multi-dimensional data. However, a large number of projections is sometimes required to ensure accuracy, which will also increase the running time. In this case, `PmedTrapz` is preferred for the two- or three-dimensional data, which is fast and accurate in general. In higher dimensions, `yamm` is another alternative for computing the projection median.

Value

A vector of the projection median for n-dimensional data.

References

Durocher, S. and Kirkpatrick, D. (2009), The projection median of a set of points, *Computational Geometry*,**42**, 364-375.

Basu, R., Bhattacharya, B.B., and Talukdar, T. (2012) The projection median of a set of points in Rd *CCCG.*, **47**, 329-346. doi: 10.1007/s0045401193806

See Also

`PmedTrapz`, `yamm`

Examples

```
# Load a 2-dimensional data set
data(clusters2d)
#
# Set seed for reproduction.
set.seed(5)
#
# Projection median approximated by Monte Carlo Integration.
PmedMCInt(clusters2d, nprojs = 50000)
# [1]  4.3246488 -0.1535201
#
#
# Load a 6-dimensional data set
data(beetle)
#
set.seed(5)
PmedMCInt(beetle,nprojs = 150000)
```

```
# [1] 179.92439 125.16939  50.01176 136.55460  13.22277  95.04224
```

PmedTrapz *Projection Median Approximated by Trapezoidal Rule*

Description

This function approximates the projection median using trapezoidal rule, which is only valid for the two- and three-dimensional cases. PmedTrapz is implemented internally using C code CPmedTrapz2D and CPmedTrapz3D and hence is much faster than coding with R only.

Usage

```
PmedTrapz(x, no.subinterval)
```

Arguments

x	The data as a matrix or data frame, with each row being viewed as one multivariate observation.
no.subinterval	A vector of subintervals chosen for implementing the trapezoidal rule. It is a number in the two-dimensional case, and has a length of two for the three-dimensional data, since the trapezoidal rule is only required once in 2D and needs to be applied twice for the double integral in 3D. Small values (e.g. less than 10) for each entry of no.subinterval should not be used, to safeguard the accuracy.

Details

The projection median was introduced by Durocher and Kirkpatrick (2009) and generalised by Basu, Bhattacharya and Talukdar (2012). PmedTrapz produces the projection median directly from the definition using the trapezoidal rule, but current function is only valid in the two-dimensional and three-dimensional case. For more general dimensionalities, you can refer to function PmedMCInt and yamm.

Value

A vector of the projection median in the two or three dimensions.

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

Durocher, S. and Kirkpatrick, D. (2009), The projection median of a set of points, *Computational Geometry*,**42**, 364-375.

Basu, R., Bhattacharya, B.B., and Talukdar, T. (2012) The projection median of a set of points in Rd *CCCG.*, **47**, 329-346. doi: 10.1007/s0045401193806

See Also

PmedMCInt

Examples

```
# Load a 2-dimensional dataset
data(clusters2d)
#
# Projection median approximated by the trapezoidal rule.
PmedTrapz(clusters2d,no.subinterval=180)
# [1]  4.1556553 -0.3566614
#
# Load a 3-dimensional dataset
data(clusters3d)
#
PmedTrapz(clusters3d,c(180,360))
# [1] -0.906680  1.584866  2.695584
```

---

yamm *Yet Another Multivariate Median*

Description

Another method for computing the projection median for any dimensional dataset. Basically, it minimises the objective function yamm.obj over a unit hypersphere and finds the optimal shift vector mu in yamm.obj. optim in the **stats** package is used in this function to minimise yamm.obj.

Usage

```
yamm(x, nprojs = 2000, reltol = 1e-6, abstol=-Inf, xstart = l1median(x),
     opt.method = "BFGS", doabs = 0, full.results=FALSE)
```

Arguments

x	The data as a matrix or data frame, with each row being viewed as one multivariate observation.
nprojs	The number of projections for the shifted data matrix while using the Monte Carlo method to approximate the integration. The default value is 2000, more projections may be required for complicated data to ensure accuracy, which, however, increases the running time.
reltol	The tolerance of the optimisation process gets supplied to control arguments of optim. The default value is $1e-6$. Loosening tolerance will make the running process faster. Generally, $1e-3$ is enough to obtain a good approximation for a short running time.
abstol	The absolute convergence tolerance of the optimisation process gets supplied to control arguments of optim. The default value is negative infinity.

`xstart`	The starting value for the optimiser. The default value is Spatial median of the data using function `l1median`. Other multivariate medians or mean values can also be used. Note, you should be aware of the outliers when using the mean values as a starting point, which may slow down the optimisation process or result in a less accurate median.
`opt.method`	The method chosen for the optimiser when computing the `yamm`, with default function "`BFGS`". Apart from "`BFGS`", other functions in `optim` like "`Nelder-Mead`", "`CG`", "`L-BFGS-B`", and "`SANN`" can also be used.
`doabs`	If 0 (default), the function `yamm.obj` integrates the square of the univariate median of the projection to the shifted data set over a unit hypersphere; if 1, `yamm.obj` integrates the absolute value of the univariate median instead.
`full.results`	Logical. If `FALSE` (default), the function `yamm` only returns the best set of yamm location estimator found; if `TRUE`, a list of full reults from the function `optim` is displayed.

Value

If `full.results = FALSE`, it returns the best set of yamm location estimator found, otherwise, it returns a list comprising of

`par`	The best set of parameters found, which is the yamm location estimator.
`value`	The value of objective function `yamm.obj` corresponding to `par`.
`counts`	A two-element integer vector giving the number of calls to the objective function and gradient of the function respectively. This excludes those calls needed to compute the Hessian, if requested, and any calls to the objective function to compute a finite-difference approximation to the gradient.
`convergence`	An integer code. 0 indicates successful completion (which is always the case for method "`SANN`" and "`Brent`"). Possible error codes are 1 indicates that the iteration limit had been reached. 10 indicates degeneracy of the `Nelder-Mead` simplex. 51 indicates a warning from the "`L-BFGS-B`" method; see component `message` for further details. 52 indicates an error from the "`L-BFGS-B`" method; see component `message` for further details.
`message`	A character string giving any additional information returned by the optimiser, or `NULL`

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

`yamm.obj`, `optim`.

Examples

```
data(beetle)
#
# Set seed for reproduction.
set.seed(5)
#
# Yamm approximated using 1000 projections.
yamm(beetle,nprojs = 1000,reltol=1e-3,doabs=0,full.results=TRUE)
#
# $par
# [1] 180.30601 124.23781  50.16349 135.53947  13.45252  95.64742
#
# $value
# [1] 5.704375
#
# $counts
# function gradient
#      69        4
#
# $convergence
# [1] 0
#
# $message
# NULL
```

`yamm.obj` *Objective Function for Yamm*

Description

The objective function when computing `yamm`, which is the integral of the squared or absolute value of the univariate median of the projection of the shifted data set over a unit hypersphere. It is implemented internally using C code `Cyammobj` and hence is much faster than coding with R only.

Usage

```
yamm.obj(x, mu, nprojs = 2000, doabs = 0)
```

Arguments

`x`	The data as a matrix or data frame, with each row being viewed as one multivariate observation.
`mu`	A shift vector with length n, where n should equal to the number of columns (variables) of the data matrix. Each row of the data matrix `x` is shifted by `mu` to obtain the shifted data matrix.
`nprojs`	The number of projections for the shifted data matrix while using the Monte Carlo method to approximate the integration. The default value is 2000.

doabs — If 0 (default), function yamm.obj integrates square of the univariate median of the projection to the shifted data set over a unit hypersphere; if 1, yamm.obj integrates absolute value of the univariate median instead.

Value

A univariate integral of the squared or absolute value of the median of the projection of the shifted data set over a unit hypersphere is returned from the .C calling function

References

Chen, F. and Nason, Guy P. (2020) A new method for computing the projection medi an, its influence curve and techniques for the production of projected quantile plots. *PLOS One*, (to appear)

See Also

yamm

Examples

```
data(beetle)
#
# Set seed for reproduction.
set.seed(5)
#
# Objective function for yamm with a chosen shift vector.
#
yamm.obj(beetle, mu=rep(10,6), nprojs=5000, doabs=1)
# [1] 88.38346
```

Index

BIBLIOGRAPHY

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

Aloupis, G., Langerman, S., Soss, M., and Toussaint, G. T. (2003). Algorithms for bivariate medians and a Fermat-Torricelli problem for lines. *Computational Geometry*, 26:69–79.

Andersen, P. K. and Gill, R. D. (1982). Cox's regression model for counting processes: A large sample study. *The Annals of Statistics*, 10:1100–1120.

Antoniadis, A., Grégoire, G., and Nason, G. P. (1999). Density and hazard rate estimation for right censored data by using wavelet methods. *Journal of the Royal Statistical Society, Series B*, 61:63–84.

Basu, R., Bhattacharya, B. B., and Talukdar, T. (2012). The projection median of a set of points in $\mathbb{R}^d$. *Discrete and Computational Geometry*, 47:329–346.

Bhansali, R. J. (1984). *Robust and Nonlinear Time Series Analysis*. Springer, New York.

Bose, P., Maheshwari, A., and Morin, P. (2003). Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry: Theory and Applications*, 24:135–146.

Box, G., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. Wiley Series in Probability and Statistics, New Jersey.

Breimann, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms. *The Institute of Mathematics and its Applications*, 6:76–90.

Byar, D. P. (1980). The veterans administration study of chemoprophylaxis for recurrent stage I bladder tumours: Comparisons of placebo, pyridoxine and topical thiotepa. In *Bladder Tumors and Other Topics in Urological Oncology*, pages 363–370. Springer, Boston.

Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16:1190–1208.

Cao, R. and Lopez-de Ullibarri, I. (2007). Product-type and presmoothed hazard rate estimators with censored data. *TEST: An Official Journal of the Spanish Society of Statistics and Operations Research*, 16:355–382.

Cardot, H., Cénac, P., and Zitt, P. A. (2013). Efficient and fast estimation of the geometric median in Hilbert spaces with an averaged stochastic gradient algorithm. *Bernoulli Society for Mathematical Statistics and Probability*, 19:18–43.

Chaudhuri, P. and Sengupta, D. (1993). Sign tests in multidimension: Inference based on the geometry of data cloud. *Journal of the American Statistical Association*, 88:1363–1370.

Chen, F. and Nason, G. (2020a). *Yamm: Multivariate Methods Based on Projections and Related Concepts*. R package version 1.3.1.

Chen, F. and Nason, G. P. (2020b). A new method for computing the projection median, its influence curve and techniques for the production of projected quantile plots. *PLOS ONE e-prints*, 15(5).

Chipman, H. A., Kolaczyk, E. D., and McCulloch, R. E. (1997). Adaptive Bayesian wavelet shrinkage. *Journal of the American Statistical Association*, 92:1413–1421.

Chu, J. (2021). A statistical analysis of the novel coronavirus (COVID-19) in Italy and Spain. *PLOS ONE e-prints*, 16(3).

Clayton, D. (1994). Some approaches to the analysis of recurrent event data. *Statistical Methods in Medical Research*, 3:244–262.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society, Series B*, 34:187–220.

Cradden, L. C., Restuccia, F., Hawkins, S. L., and Harrison, G. P. (1974). Consideration of wind speed variability in creating a regional aggregate wind power time series. *Resources*, 3:215–234.

Croux, C., Filzmoser, P., and Oliveira, M. R. (2006). Algorithms for projection-pursuit robust principal component analysis. *KU Leuven Working Paper No. KBI 0624*, 19:18–43.

Daubechies, I. (1988). Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, 41:909–996.

Daubechies, I. (1992). *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia.

Delignette-Muller, M. L. and Dutang, C. (2015). fitdistrplus: An R package for fitting distributions. *Journal of Statistical Software*, 64(4).

Donoho, D. L. and Johnstone, I. M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81:425–455.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:1200–1224.

Drost, H. (2018). Philentropy: Information theory and distance quantification with R. *Journal of Open Source Software*, 3(26).

Durocher, S. and Kirkpatrick, D. G. (2005). The projection median of a set of points in $\mathbb{R}^2$. *Journal of Computational Geometry*, 42:364–375.

Durocher, S., Leblanc, A., and Skala, M. (2017). The projection median as a weighted average. *Journal of Computational Geometry*, 8:78–104.

Everitt, B. S., Landau, S., Leese, M., and Stahl, D. (2011). *Cluster Analysis*. Wiley, London.

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

Filzmoser, P., Fritz, H., and Kalcher, K. (2021). *pcaPP: Robust PCA by Projection Pursuit*. R package version 1.9-74.

Fischer, D., Mosler, K., Möttönen, J., Nordhausen, K., Pokotylo, O., and Vogel, D. (2016). Computing the Oja median in R: The package OjaNP. *ArXiv e-prints*.

Fletcher, R. (1970). A new approach to variable metric algorithms. *Computer Journal*, 13:317–322.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *Computer Journal*, 7:148–154.

Fraiman, R. and Pateiro-Lopez, B. (2012). Quantiles for finite and infinite dimensional data. *Journal of Multivariate Analysis*, 108:1–14.

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741.

Genest, M., Masse, J.-C., src/depth.f contains eigen, J.-F. P., tql2, tred2 written by the EISPLACK authors, dgedi, dgefa from LINPACK written by Cleve Moler, daxpy, dscal, dswap, idamax from LINPACK written by Jack Dongarra, from NAPACK, V., written by J. C. Gower, A. ., written by F. K. Bedall, A. ., Zimmermann, H., written by P.J. Rousseeuw, A. ., and Ruts., I. (2019). *depth: Nonparametric Depth Functions for Multivariate Analysis*. R package version 2.1-1.1.

Gerber, F. and Furrer, R. (2019). optimParallel: An R package providing a parallel version of the L-BFGS-B optimization method. *The R Journal*, 11:352–358.

Goldfarb, D. (1970). A family of variable metric updates derived by variational means. *Journal of the Mathematics of Computation*, 24:123–126.

Gower, J. and Legendre, P. (1986). Metric and Euclidean properties of dissimilarity coefficients. *Journal of Classification*, 3:5–48.

Hall, P. and Patil, P. (1995). Formulae for mean integrated squared error of nonlinear wavelet-based density estimators. *The Annals of Statistics*, 23:905–928.

Hamilton, J. (1994). *Time Series Analysis*. Princeton University Press, New Jersey.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29:147–160.

Hayford, J. W. (1902). What is the center of an area or the center of a population. *Journal of the American Statistical Association*, 8:47–58.

Herrick, D. R. M., Nason, G. P., and Silverman, B. W. (2001). Some new methods for wavelet density estimation. *Sankhya: The Indian Journal of Statistics, Series A*, 63:394–411.

Hoseinpour Dehkordi, A., Alizadeh, M., Derakhshan, P., Babazadeh, P., and Jahandideh, A. (2020). Understanding epidemic data and statistics: A case study of COVID-19. *Journal of Medical Virology*, 92:868–882.

Hougaard, P. (1995). Frailty models for survival data. *Lifetime Data Analysis*, 1:255–273.

Ishwaran, H. and James, L. (2001). Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96:161–173.

Jaccard, P. (1912). The distribution of the flora in the alpine zone. *New Phytologist*, 11:37–50.

Johnson, N. L., Kotz, S., and Balakrishnan, N. (1995). *Continuous Univariate Distributions*. Wiley & Sons, New York.

Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53:457–481.

Knight, M., Leeming, K., Nason, G., and Nunes, M. (2020). Generalized network autoregressive processes and the GNAR package. *Journal of Statistical Software*, 96(5).

Knight, M. I., Nunes, M. A., and Nason, G. P. (2016). Modelling, detrending and decorrelation of network time series. *ArXiv e-prints*.

Kong, L. and Mizera, I. (2012). Quantile tomography: Using quantiles with multivariate data. *Statistica Sinica*, 22:1589–1610.

Langerman, S. and Steiger, W. (2003). *Optimization in Arrangements*. Springer Berlin Heidelberg, Berlin.

Leeming, K., Nason, G., Knight, M., and Nunes, M. (2020). *GNAR: Methods for Fitting Network Time Series Models*. R package version 1.1.1.

Lopez-de Ullibarri, I. and Jacome, M. (2013). survpresmooth: An R package for presmoothed estimation in survival analysis. *Journal of Statistical Software*, 54(11).

Lubischew, A. A. (1962). On the use of discriminant functions in taxonomy. *Biometrics*, 18:455–477.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2021). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.1.3.

Mahalanobis, P. C. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India*, 2:49–55.

Mallat, S. G. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693.

Martin, C. M., Lundquist, J. K., and Handschy, M. A. (2015). Variability of interconnected wind plants: Correlation length and its dependence on variability time scale. *Environmental Research Letters*, 10(4):044004.

Massey, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46:68–78.

McGilchrist, C. and Aisbett, C. (1991). Regression with frailty in survival analysis. *Biometrics*, 47:461–466.

Mclachlan, G. (1999). Mahalanobis distance. *Resonance*, 4:20–26.

Mendelson, B. (1990). *Introduction to Topology*. Dover, New York.

Nadaraya, E. (1964). On estimating regression. *Theory of Probability and Its Applications*, 9:186–190.

Nason, G. (1996). Wavelet shrinkage using cross-validation. *Journal of the Royal Statistical Society, Series B*, 58:463–479.

Nason, G. (2008). *Wavelet Methods in Statistics with R*. Springer, New York.

Nason, G. (2016). *wavethresh: Wavelets Statistics and Transforms*. R package version 4.6.8.

Nason, G. and Wei, J. (2021). Quantifying the economic response to COVID-19 mitigations and death rates via forecasting purchasing managers' indices using generalised network autoregressive models with exogenous variables. *Journal of the Royal Statistical Society, Series A*, 184: (to appear).

Nelder, J. A. and Mead, R. (1965). A simplex algorithm for function minimization. *Computer Journal*, 7:308–313.

Nocedal, J. and Wright, S. J. (1999). *Numerical Optimization*. Springer, New York.

Nordhausen, K., Sirkia, S., Oja, H., and Tyler, D. E. (2018). *ICSNP: Tools for Multivariate Nonparametrics*. R package version 1.1-1.

Nunes, M. A., Knight, M. I., and Nason, G. P. (2015). Modelling and prediction of time series arising on a graph. In *Modelling and Stochastic Learning for Forecasting in High Dimensions*, pages 183–192. Springer International Publishing, Switzerland.

Oja, H. (1983). Descriptive statistics for multivariate distributions. *Statistics & Probability Letters*, 1:327–332.

Oja, H. (2013). Multivariate median. In Becker, C., Fried, R., and Kuhnt, S., editors, *Robustness and Complex Data Structures*, chapter 1, pages 3–16. Springer, Berlin.

Omori, R., Mizumoto, K., and Nishiura, H. (2020). Ascertainment rate of novel coronavirus disease (COVID-19) in Japan. *International Journal of Infectious Diseases*, 96:673–675.

Pham, H. (2020). On estimating the number of deaths related to COVID-19. *Mathematics e-prints*, 8.

Prentice, R. L., Williams, B. J., and Peterson, A. V. (1981). On the regression analysis of multivariate failure time data. *Biometrika*, 68:373–379.

Ramsay, K. (2017). Computable, robust multivariate location using integrated univariate ranks. Master's thesis, University of Manitoba, Winnipeg.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850.

Rathbun, S. L. and Shiffman, S. (2018). Mixed effects models for recurrent events data with partially observed time-varying covariates: Ecological momentary assessment of smoking. *Biometrics*, 1:46–55.

Robert, C. P. and Casella, G. (2005). *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg.

Rousseeuw, P. J. and Ruts, I. (1996). Algorithm AS 307: Bivariate location depth. *Journal of the Royal Statistical Society, Series C*, 45:516–526.

Rousseeuw, P. J., Ruts, I., and Tukey, J. W. (1999). The bagplot: A bivariate boxplot. *The American Statistician*, 53:382–387.

Sethuraman, J. (1994). A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650.

Shanno, D. F. (1970). Conditioning of quasi-Newton methods for function minimization. *Journal of the Mathematics of Computation*, 24:647–656.

Shibata, R. (1980). Asymptotically efficient selection of the order of the model for estimating parameters of a linear process. *Annals of Statistics*, 8:147–164.

Shumway, R. and Stoffer, D. (2011). *Time Series and its Applications*. Springer, New York.

Silverman, B. and Johnstone, I. (2004). Empirical Bayes estimates of possibly sparse sequences. *Annals of Statistics*, 32:1594–1649.

Silverman, B. and Johnstone, I. (2005a). Ebayesthresh: R programs for empirical Bayes thresholding. *Journal of Statistical Software*, 12(8).

Silverman, B. and Johnstone, I. (2005b). Empirical Bayes selection of wavelet thresholds. *Annals of Statistics*, 33:1700–1752.

Sims, C. A. (1980). Macroeconomics and reality. *Econometrica*, 48:1–48.

Small, C. G. (1990). A survey of multidimensional medians. *International Statistical Review*, 58:263–277.

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9:1135–1151.

Struyf, A. and Rousseeuw, P. J. (2000). High-dimensional computation of the deepest location. *Computational Statistics & Data Analysis*, 34:415–426.

Sun, L., Zhao, X., and Zhou, J. (2011). A class of mixed models for recurrent event data. *The Canadian Journal of Statistics*, 4:578–590.

Therneau, T. M. (2021). *A Package for Survival Analysis in R*. R package version 3.2-11.

Tukey, J. W. (1975). Mathematics and the picturing of data. *In Proceedings of the International Congress of Mathematicians*, 2:523–531.

Vardi, Y. and Zhang, C. (2000). The multivariate L1-median and associated data depth. *Proceedings of the National Academy of Sciences*, 97:1423–1426.

Watson, G. (1964). Smooth regression analysis. *Shankya A*, 26:359–372.

Weber, A. (1929). *Theory of the Location of Industries*. The University of Chicago Press, Chicago.

Wei, L. J., Lin, D. Y., and Weissfeld, L. (1989). Regression analysis of multivariate incomplete failure time data by modeling marginal distributions. *Journal of the American Statistical Association*, 84:1065–1073.

Yadav, D. C., Vishnubhatla, S., Ma, k., and Pandey, R. (2018). An overview of statistical models for recurrent events analysis: A review. *Epidemiology e-prints*, 08(4).

Zhang, Z., Telesford, Q. K., Giusti, C., Lim, K. O., and Bassett, D. S. (2016). Choosing wavelet methods, filters, and lengths for functional brain network construction. *PLOS ONE*, 11:1–24.

Zhu, X., Pan, R., Li, G., Liu, Y., and Wang, H. (2017). Network vector autoregression. *The Annals of Statistics*, 45:1096–1123.