Kernel deconvolution density estimation

by

Guillermo Basulto-Elias

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Statistics

Program of Study Committee:

Alicia L. Carriquiry, Co-major Professor

Kris De Brabanter, Co-major Professor

Daniel J. Nordman, Co-major Professor

Kenneth J. Koehler

Yehua Li

Lily Wang

Iowa State University

Ames, Iowa

2016

DEDICATION

To Martín. I could not have done without your love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I want to express my deepest gratitude to my advisers, Dr. Alicia Carriquiry, Dr. Daniel Nordman and Dr. Kris De Brabanter. They imparted on me their infinite wisdom, encouraged me, were patient, and gave much of their time so that I could succeed.

I acknowledge my committee members, Dr. Ken Koehler, Dr. Yehua Li and Dr. Lily Wang. Their comments and suggestions on this dissertation were very helpful.

Even the steepest road is easy when you have people on your side. I have always counted on my family and friends - especially on my mother, my sisters and my husband, Lety, Diana, Viry and Martín. My friends in Ames were a valuable source of support.

Every one of my professors at Iowa State University has taught something beyond the syllabus. My undergraduate and masters advisers, Dr. Víctor Pérez-Abreu and Dr. Miguel Nakamura have mentored me all along the way.

Finally, I want to thank the Department of Statistics and the National Council of Science and Technology of Mexico (CONACyT) for their support.

CHAPTER 1. INTRODUCTION

In many areas of application, like medical sciences, variables of interest are not directly observable but only through some error contamination. These cases are often referred as "measurement error problems".

Kernel deconvolution density estimation (KDDE) is a measurement error problem which consists in estimating the density of a population that is observable only with some error contamination. For sake of clarity, we will use a simple example, suppose that we wish to get an estimate of the distribution of the number of steps taken by a particular population. Portable pedometers do the job, but number of steps may be very different from device to device (even between two devices of the same brand). Thus, the measurement error induced by the device should be taken into account.

Assume for the moment that the distribution of the measurement error is known among the pedometers used. Then, assume that we have a sample $Y_1, \ldots, Y_n$ that come from the model

$$Y = X + \epsilon, \tag{1.1}$$

where X is the (unobservable) distribution of the number of steps and ϵ measurement error of the device.

KDDE is a modification of the kernel density estimator (KDE). Recall that the KDE of the sample $X_1, \ldots, X_n$ from the random variable X is given by

$$\tilde{f}_X(x) = \sum_{i=1}^{n} \frac{1}{nh} K\left(\frac{x - X_j}{h}\right), \qquad x \in \mathbb{R}, \tag{1.2}$$

where K is a symmetric function that integrates to one, called the kernel function, and h is called the bandwidth parameter. Using the Fourier inversion theorem we see that the estimator

$\tilde{f}$ in (1.2) is mathematically equivalent to

$$\tilde{f}_X(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(-\imath x t)\hat{\phi}_{X,n}(t)K^{ft}(ht)dt, \qquad x \in \mathbb{R}, \tag{1.3}$$

where $\hat{\phi}_{X,n}(t) = n^{-1}\sum_i \exp(\imath t X_i)$ is the empirical characteristic function of the sample and $K^{ft}(s) = \int \exp(\imath s x)K(x)dx$ is the Fourier transform of the kernel K.

We cannot use (1.3) with the contaminated sample (even less if the error is not negligible). The KDDE is a modification of (1.3) that depends directly of the contaminated sample:

$$\hat{f}_X(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(-\imath x t)\hat{\phi}_{Y,n}(t)\frac{K^{ft}(ht)}{\phi_\epsilon(t)}dt, \qquad x \in \mathbb{R}, \tag{1.4}$$

where $\phi_\epsilon(t) = (2\pi)^{-1}\int \exp(-\imath t x)f_\epsilon(x)dx$ is the characteristic function of ϵ, which has density f_ϵ. Therefore, the KDDE and its properties are mostly studied in the frequency domain.

Several challenges arise with the KDDE. We will list next a few of these.

1. Observe in (1.4) that taking K with Fourier transform that is compactly supported would ensure that the integral exists and it would be easier to compute it. Moreover, convergence rates are easier to find if K^{ft} has compact support.

2. The integral in (1.4) is hard to obtain due to the oscillating nature of the integrand. It is important to have effective and fast numerical procedures to do it.

3. It is unrealistic most of the times to assume that the error distribution (its characteristic function in particular) is known. In real life it can be approximated by providing an independent sample of errors based on repeated measures. In our pedometer example, we can imagine designing an experiment actually counting the differences in steps to approximate the error distribution. For the latter, every individual could wear several pedometers at the same time, thus there would be repeated observations for the same individual.

4. In the case of having repeated measurements, the observations are not independent. There are several ways to obtain a (contaminated) sample and to obtain observations to approximate the error.

5. Equation 1.4 can be generalized to higher dimensions. The literature in that case is not extensive.

6. A challenging problem in KDE is the bandwidth selection. In order to generalize any of the existing methods to KDDE, it is necessary to take it first to the frequency domain. This is not always possible or straightforward.

In this work we approach several of this problems. In particular, in Chapter 2 we conduct a numerical investigation of bivariate KDDE to investigate item 1 by comparing several kernels whose Fourier transform has compact support. We pay special attention to a class of kernels referred as "flat-top kernels" (cf. Politis and Romano (1999)). We also study several aspects of items 3 and 4 in Chapter 2 and compare two nonparametric ways to approximate the (characteristic function of the) error distribution.

Chapter 3 describes an R package to compute univariate and bivariate continous Fourier transforms, which is versatile enough to perform integration using any definition of continuous Fourier transform and its inverse. The implementation for the regularly spaced grid is an algorithm that uses the so-called Fast Fourier Transform that it is known to reduce drastically the computation time. We actually use the C++ library called Armadillo to speed up computations.

We use the R package described in Chapter 3 to create an R package specialized in fast computation of KDDE under several scenarios including the ones described in items 3 and 4 for univariate and bivariate samples. This R package also allows the user to approximate the error in a parametric or nonparametric fashion with ease. The bandwidth selection method is based on cross-validation (cf. Youndjé and Wells (2008)) and it allows the user to visualize the function that had to be minimized to find the bandwidth.

Finally, in Chapter 4 we propose a bandwidth selection which does not require non-trivial iterative steps and does not require knowledge of the smoothness level of the target distribution (cf. Delaigle and Gijbels (2004b) and Fan (1991a)). This bandwidth selection method is inspired by Politis (2003a) in the context of spectral density estimation and KDE.

CHAPTER 2. A NUMERICAL INVESTIGATION OF BIVARIATE KERNEL DECONVOLUTION WITH PANEL DATA

A paper to be submitted

Guillermo Basulto-Elias, Alicia L. Carriquiry, Kris De Brabanter and Daniel J. Nordman

Abstract

We consider estimation of the density of a multivariate response, that is not observed directly but only through measurements contaminated by additive error. We focus on the realistic sampling case of bivariate panel data (repeated contaminated bivariate measurements on each sample unit) with an unknown error distribution. Several factors can affect the performance of kernel deconvolution density estimators, including the choice of the kernel and the estimation approach of the unknown error distribution. We show that the choice of the kernel function is critically important and that the class of flat-top kernels has advantages over more commonly implemented alternatives. We describe different approaches for density estimation with multivariate panel responses, and investigate their performance through simulation. We examine competing kernel functions and describe a flat-top kernel that has not been used in deconvolution problems. Moreover, we study several nonparametric options for estimating the unknown error distribution. Finally, we also provide guidelines to the numerical implementation of kernel deconvolution in higher sampling dimensions.

2.1 Introduction

A challenge in many areas of application (e.g., medical sciences) is that responses of interest are not observable; instead, responses are measured with error contamination. Such cases are

frequently referred to as "measurement error problems" and arise frequently in applications. An example from nutrition epidemiology is the following. Suppose that we wish to estimate the joint density of the "usual" or long-run average of serum iPTH (intact parathyroid hormone) and 25(OH)D (25-hydroxy vitamin D), both of which are associated with bone health. While daily values for these two substances can be measured reliably, they are noisy estimates of their long-run averages, which are the quantities of interest. Ignoring such measurement error can lead to biased inference. Kernel deconvolution density estimation provides an alternative way to remove measurement error in estimating a target density, without the need for stringent parametric assumptions. Deconvolution estimation typically involves translation to the frequency domain to approximate the characteristic function of a target variable through the estimated characteristic functions of noisy measurements and errors. Although kernel deconvolution has received much attention for univariate observations, the case of multivariate observations has received less consideration, especially for the more realistic case where little is known or can be immediately assumed about the error distribution and where the observations consist of one or more (correlated) measurements on subjects (i.e., panel data structure). In particular, there has been limited theoretical development for the bivariate panel data case and even less for the multivariate case.

In this paper, we focus on a simulation-based study to explore the performance of deconvolution methods, in particular when applied to bivariate panel data. The performance of kernel deconvolution estimation in the multivariate setting depends critically on a combination of factors that have not been well studied. These include the choice of kernel, the way in which panel data are pre-processed, the number of measurements obtained for each sample unit, and the approach to estimating the unknown error distribution. Through simulation, we study the impact of these potentially important factors in order to better understand the practical issues associated with implementation of kernel deconvolution and to inform possible future theoretical developments in this area.

In addressing bivariate kernel deconvolution, we consider five candidate kernel functions for which the Fourier transform has bounded support, include the sinc kernel (a popular choice in the univariate case) and a trapezoidal flat-top (where the latter is novel in deconvolution

problems). Additionally, we compare different approximations to the (unknown) characteristic function of the errors with panel data. Such approximations involve choices on how to incorporate replicate measurements and combine these with either a direct empirical estimator or an indirect kernel density-based estimator of the characteristic function of the error. Perhaps surprisingly, our findings suggest that in contrast to some claims in the univariate setting, the way in which panel data are processed (i.e., differenced to estimate the error distribution) has little impact, and that kernels commonly used with univariate measurements do not perform as well as other kernels (e.g., "flat top") studied here for the multivariate case. Further, kernel deconvolution estimators for multivariate responses pose more computational challenges, in particular for numerical integration. To overcome the computational challenges, we use an algorithm based on a Fast Fourier Transform (FFT) that improves efficiency without compromising accuracy .

We end this section with a brief review of the literature. An extensive discussion of the univariate deconvolution problem can be found in Meister (2009). For the multivariate setting, asymptotic convergence properties when the error distribution is assumed to be known can be found in Masry (1991, 1993a,b). The selection of a single bandwidth parameter (diagonal bandwidth matrix with equal diagonal entries) using cross-validation was proposed by Youndjé and Wells (2008) while Comte and Lacour (2013) who also focused diagonal bandwidth matrices, proposed estimating its diagonal elements as the minimum of an empirical approximation to the asymptotic integrated mean squared error. Johannes (2009) and Comte and Lacour (2011) considered the situation where the error distribution is unknown, but a sample of pure errors is available. None of these papers considered multivariate panel data (i.e., repeated multivariate measurements on each subject, contaminated with error). In the univariate panel data setting, with unknown error distribution, several approaches have been proposed for estimating the characteristic function of the error. Delaigle and Meister (2008) suggested using all possible measurement differences available for each individual with deconvolution applied to the measurement averages per subject. Stirnemann et al. (2012b) and Comte et al. (2014) proposed taking simple pairwise differences of the replicate measurements to approximate the characteristic function of the error; only the first noisy measurement on each subject is then used in the

deconvolution steps that follow. Ridge terms or truncation, as in Neumann and Hössjer (1997), are typically applied to control the estimated characteristic function of errors for small values. For the vast majority of the literature discussing deconvolution with univariate panel data, the most frequently chosen kernel is the sinc kernel because it has been shown to have good theoretical properties (e.g., its Fourier transform is flat around zero as suggested by Devroye (1989)) and because its form permits applying the FFT for numerical integration.

This paper is organized as follows. Section 2.2 describes the general density estimation problem and how it can be approached with kernel methods, with special attention to the panel data case introduced in Section 2.2.2. In Section 2.3, we outline several potentially important attributes of the deconvolution method that may impact performance. In particular, Section 2.3.1 discusses how to approximate the characteristic function of errors from a panel data structure, and Section 2.3.2 describes candidate kernel functions. Section 2.3.3 provides numerical integration tools required for bivariate kernel density deconvolution. In Section 2.4 we present the design of a large simulation study and discuss the numerical results. Finally, conclusions are presented in Section 2.5.

2.2 Problem description

We first outline the estimation method for multivariate kernel density deconvolution in general. Section 2.2.1 introduces some initial background in the context of a simplified version of kernel deconvolution density estimation in the multivariate setting. Section 2.2.2 then discusses the case of multivariate panel data and some related estimation issues that arise in this situation. For illustration purposes, we focus on the case of bivariate measurements $\boldsymbol{X} \in \mathbb{R}^d$, $d = 2$, although generalizations to any dimension d of the observation vector are possible.

2.2.1 A simplified estimation framework

Suppose that we are interested in estimating the distribution of a two-dimensional random variable $\boldsymbol{X}$ that cannot be directly observed. Instead we observe $\boldsymbol{X}$ convoluted with random noise, say, $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \in \mathbb{R}^2$ is the random noise vector. The standard deconvolution estimation problem consists of estimating the density of $\boldsymbol{X}$ using a random sample of

measurements

$$\boldsymbol{Y}_i = \boldsymbol{X}_i + \boldsymbol{\epsilon}_i, \tag{2.1}$$

for $i = 1, \ldots, n$. In (2.1), the collection $\{\boldsymbol{X}_i\}_{i=1}^n$ is independent and identically distributed (iid) and so is the variable set $\{\boldsymbol{\epsilon}_i\}_{i=1}^n$. Furthermore, for each measurement $\boldsymbol{Y}_i$, the response $\boldsymbol{X}_i$ of interest is assumed to be independent of the error $\boldsymbol{\epsilon}_i$.

We require some notation before introducing the deconvolution formulas. Let $\phi_{\boldsymbol{Y}}(\cdot)$ denote the characteristic function (CF) of the random sample $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$, and let $\hat{\phi}_{\boldsymbol{Y},n}(\cdot)$ denote the empirical CF. That is, for any $\boldsymbol{t} \in \mathbb{R}^2$,

$$\phi_{\boldsymbol{Y}}(\boldsymbol{t}) = \mathbb{E}\left(e^{\imath \boldsymbol{t}\cdot\boldsymbol{Y}_1}\right) \text{ and } \hat{\phi}_{\boldsymbol{Y},n}(\boldsymbol{t}) = \frac{1}{n}\sum_{j=1}^{n} e^{\imath \boldsymbol{t}\cdot\boldsymbol{Y}_j},$$

where the dot product $\cdot$ above represents the usual inner product in $\mathbb{R}^2$ and $\imath = \sqrt{-1}$. For a real-valued function $g : \mathbb{R}^2 \to \mathbb{R}$, denote its Fourier transform by g^*, where

$$g^*(\boldsymbol{t}) = \int e^{\imath \boldsymbol{t}\cdot\boldsymbol{x}} g(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}. \tag{2.2}$$

When g is a (multivariate) density function, then g^* represents its CF.

By the independence of $\boldsymbol{X}_i$ and $\boldsymbol{\epsilon}_i$ in (2.1), the CF $\phi_{\boldsymbol{X}}(\cdot)$ of the uncontaminated measurements $\boldsymbol{X}$ can be obtained from that of $\phi_{\boldsymbol{Y}}(\cdot)$, the CF of the observed measurements, as

$$\phi_{\boldsymbol{X}}(\boldsymbol{t}) = \frac{\phi_{\boldsymbol{Y}}(\boldsymbol{t})}{\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t})}, \ \boldsymbol{t} \in \mathbb{R}^2,$$

whenever the CF of the errors $\phi_{\boldsymbol{\epsilon}}(\cdot)$, is non-zero (i.e., $\boldsymbol{t} \in \mathbb{R}^2$ with $\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}) \neq 0$). Consequently, by applying the Fourier inversion theorem, we can compute the density of $\boldsymbol{X}$ at $\boldsymbol{x} \in \mathbb{R}^2$ as

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{(2\pi)^2} \int \boldsymbol{e}^{-\imath \boldsymbol{t}\cdot\boldsymbol{x}} \frac{\phi_{\boldsymbol{Y}}(\boldsymbol{t})}{\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t})} \mathrm{d}\boldsymbol{t}, \tag{2.3}$$

which is the distribution of the uncontaminated response $\boldsymbol{X}$. Because $\phi_{\boldsymbol{Y}}$ is typically unknown, we may replace it with its empirical version, $\hat{\phi}_{\boldsymbol{Y},n}$, but the resulting function in (2.3) might not be integrable. However, by substituting a smoothed estimator version instead, say $\hat{\phi}_{\boldsymbol{Y},n}(\mathbf{t})K^*(B_n\mathbf{t})$, we obtain a valid (kernel) estimator of the density of $\boldsymbol{X}$ as

$$\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x}) = \frac{1}{(2\pi)^2} \int \boldsymbol{e}^{-\imath \boldsymbol{t}\cdot\boldsymbol{x}} \hat{\phi}_{\boldsymbol{Y},n}(\mathbf{t}) \frac{K^*(B_n\boldsymbol{t})}{\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t})} \mathrm{d}\boldsymbol{t}, \quad \boldsymbol{x} \in \mathbb{R}^2. \tag{2.4}$$

Here K^* denotes the Fourier transform of a smoothing kernel K and B_n denotes a 2×2 (positive definite) bandwidth matrix.

Note that (2.4) assumes that the CF of the underlying error terms, $\phi_{\boldsymbol{\epsilon}}$, is known. Under the general data model (2.1), it is frequently assumed that either the error distribution is known or that there exists a sample of pure errors, obtained independently of $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$, that can be used to estimate the error CF (see Johannes (2009) and Comte and Lacour (2011)). These assumptions are often unrealistic in many applications. Therefore, in this paper we will not make such assumptions when describing the estimation with panel data in Section 2.2.2.

2.2.2 Kernel deconvolution estimation with panel data

We now discuss kernel density deconvolution estimation for bivariate panel data. For such data, an additive noise model is given by

$$\boldsymbol{Y}_{ij} = \boldsymbol{X}_i + \boldsymbol{\epsilon}_{ij}, \tag{2.5}$$

for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. Here $\boldsymbol{Y}_{ij}$ represents the $d = 2$ observations for the i-th individual on the j-th measurement occasion. As in model (2.1), the iid uncontaminated response variables $\{\boldsymbol{X}_i\}_{i=1}^n$ are independent of the iid errors $\{\boldsymbol{\epsilon}_{ij} : i = 1, \ldots, n; j = 1 \ldots, m\}$.

For simplicity, and without loss of generality, we assume that the number m of observations is the same for every sample unit (individual). Note that while sample units are assumed to be independent, the components among the d-dimensional measurements for each unit are typically not independent, and the replicate measurements obtained on each unit are also correlated.

We wish to estimate the joint density $f_{\boldsymbol{X}}$ of the uncontaminated variables $\{\boldsymbol{X}_i\}_{i=1}^n$. Under a panel data structure, the standard deconvolution density estimator (2.4) needs to be modified to account for the dependence among replicate measurements on each sample unit. That is, measurements made on the same subject are no longer independent, as was the case in Section 2.1 (i.e., instead two measurements $\boldsymbol{Y}_{ij}$ and $\boldsymbol{Y}_{i'j'}$ are independent only if $i \neq i'$).

As in the univariate panel data setting, there are several ways to account for the panel data structure when using the estimator (2.4) of the density $f_{\boldsymbol{X}}$. Comte et al. (2014) use only the first measurement for each subject $\{\boldsymbol{Y}_{i1}\}_{i=1}^n$ to estimate the common CF of the measurements, $\phi_{\boldsymbol{Y}}$.

However, this approach ignores the other $m-1$ measurements available for each subject and therefore may entail some information loss. Instead, we propose making use of all observations obtained on each subject to compute an average measurement for individual; we then reformulate the deconvolution estimator to be based on the CF of the averages, say $\phi_{\bar{\boldsymbol{Y}}}$, rather than on the CF of individual responses $\phi_{\boldsymbol{Y}}$ in (2.4). That is, we first average observations in (2.5) per individual,

$$\bar{\boldsymbol{Y}}_{i\cdot} = \boldsymbol{X}_i + \bar{\boldsymbol{\epsilon}}_{i\cdot}, \; i = 1, \ldots, n. \tag{2.6}$$

When the response variable is the average of m replicate measurements, the error term $\bar{\boldsymbol{\epsilon}}_{i\cdot}$ in (2.6) represents an average of m iid error variables with CF given by $\phi_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t}) = [\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}/m)]^m$ in terms of the CF $\phi_{\boldsymbol{\epsilon}}$ of an original (i.e., single) error term in the panel data model (2.5). Therefore, instead using the kernel density deconvolution estimator (2.4), we use a modified estimator given by

$$\begin{aligned} \hat{f}_{\boldsymbol{X},n}(\boldsymbol{x}) &= \frac{1}{(2\pi)^2} \int \boldsymbol{e}^{-\imath \boldsymbol{t}\cdot\boldsymbol{x}} \hat{\phi}_{\bar{\boldsymbol{Y}},n}(\mathbf{t}) \frac{K^*(B_n\boldsymbol{t})}{\phi_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t})} \mathrm{d}\boldsymbol{t} \\ &= \frac{1}{(2\pi)^2} \int \boldsymbol{e}^{-\imath \boldsymbol{t}\cdot\boldsymbol{x}} \hat{\phi}_{\bar{\boldsymbol{Y}},n}(\mathbf{t}) \frac{K^*(B_n\boldsymbol{t})}{[\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}/m)]^m} \mathrm{d}\boldsymbol{t}, \end{aligned} \tag{2.7}$$

for $\boldsymbol{x} \in \mathbb{R}^2$ based on the CF of error averages, $[\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}/m)]^m$ and the empirical CF of subject averages is $\{\bar{\boldsymbol{Y}}_i\}_{i=1}^n$, $\hat{\phi}_{\bar{\boldsymbol{Y}},n}(\boldsymbol{t}) = n^{-1}\sum_{j=1}^n e^{\imath \boldsymbol{t}\cdot\bar{\boldsymbol{Y}}_j}$, $\boldsymbol{t} \in \mathbb{R}^2$.

However, there are several issues that require examination when implementing an estimator of the form (2.7) with panel data. These can impact the performance of the kernel deconvolution estimator $\hat{f}_{\boldsymbol{X},n}$, which we numerically investigate in Section 4. Firstly, the error distribution is typically unknown so that the associated CF $\phi_{\boldsymbol{\epsilon}}$ in (2.7) needs to be estimated. By exploiting the repeated measurement in the panel data structure, we can construct at least two different estimators, both of which are based on differences between measurements within subjects. We elaborate on this issue in Section 2.3. Potential formulations for constructing differences between measurements in panel data are also outlined in that section. A second issue that arises is that the deconvolution formula (2.7) requires specification of a multivariate kernel function and the bandwidth matrix for that kernel. Section 2.3.2 discusses several types of kernels that may be considered, including "flat-top" kernels that have been successfully used for spectral density estimation but that have not been applied in deconvolution problems (cf. Politis and

Romano (1995), Politis (2003b), Politis and Romano (1999)). Finally, the deconvolution estimator includes integrals that are often numerically approximated; in contrast to the univariate data, the multivariate setting introduces further challenges for the evaluation of a density estimator $\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x})$ at vector points $\boldsymbol{x} \in \mathbb{R}^2$. In Section 2.3.3, we describe some numerical recipes that greatly improve computational efficiency in the multivariate deconvolution setting.

2.3 Challenges in deconvolution estimation with panel data

2.3.1 Estimation of the error distribution

Assume for now that the number m of observations for each individual is even for the panel data (2.5). In the univariate setting, Comte et al. (2014) proposed constructing pairwise differences between repeated measurements (within individuals) as

$$\boldsymbol{Y}_{i,2l-1} - \boldsymbol{Y}_{i,2l} = \boldsymbol{\epsilon}_{i,2l-1} - \boldsymbol{\epsilon}_{i,2l}, \tag{2.8}$$

for $i = 1, \ldots, n$ and $l = 1, \ldots, m/2$. The reason for formulating differences within individuals is that the measurement differences correspond to differences in error terms (i.e., common unobserved variables $\boldsymbol{X}_i$ drop out). Each measurement is used only once to create a difference in (2.8), and therefore the $nm/2$ differences across all individuals are iid. Another possibility is to consider all possible within-individual differences, as in Delaigle and Meister (2008); that is

$$\boldsymbol{Y}_{i,j} - \boldsymbol{Y}_{i,j'} = \boldsymbol{\epsilon}_{i,j} - \boldsymbol{\epsilon}_{i,j'}, \tag{2.9}$$

for $i = 1, \ldots, n$ and $j < j'$. This approach results in a higher number of measurement differences (i.e., $nm(m-1)/2$ differences) which, though identically distributed, are not independent. Finally, we might also consider an intermediate method, which leads to an intermediate sized collection of measurement differences, with a weaker dependence structure relative to (2.9). In this approach, the last $m-1$ measurements for each sample unit are deviated from the first one as follows:

$$\boldsymbol{Y}_{i,j} - \boldsymbol{Y}_{i,1} = \boldsymbol{\epsilon}_{i,j} - \boldsymbol{\epsilon}_{i,1}, \tag{2.10}$$

for $i = 1, \ldots, n$ and $j = 2, 3, \ldots, m$. In this case, there are $n(m-1)$ measurement differences which are not independent. In contrast to (2.8), the differencing strategies (2.9) and (2.10) do not require an even number of observations per individual. The simulation experiments in Section 2.4 compare the performance of deconvolution estimators based on these three types of measurement differences.

Regardless of the chosen approach, the processed data consist of a collection of differenced measurements or error differences, say $\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_k$, where the number of differences k depends on the differencing approach (2.8)-(2.10). We propose a methodology to estimate the error CF $\phi_{\boldsymbol{\epsilon}}$ in (2.7) with these differences. By the iid property of individual error terms, observe that, for $\boldsymbol{t} \in \mathbb{R}^2$ and $j \neq j'$, the CF $\phi_{\boldsymbol{\delta}}$ of a within individual measurement difference is given by

$$\phi_{\boldsymbol{\delta}}(\boldsymbol{t}) = \phi_{\boldsymbol{\epsilon}_{i,j} - \boldsymbol{\epsilon}_{i,j'}}(\boldsymbol{t}) = \phi_{\boldsymbol{\epsilon}_{i,j}}(\boldsymbol{t}) \phi_{\boldsymbol{\epsilon}_{i,j'}}(-\boldsymbol{t}) = \phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}) \phi_{\boldsymbol{\epsilon}}(-\boldsymbol{t}) = |\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t})|^2 . \tag{2.11}$$

If we assume that the distribution of the panel data errors is symmetric and its CF is never zero, then

$$\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}) = \sqrt{\phi_{\boldsymbol{\delta}}(\boldsymbol{t})} \tag{2.12}$$

holds when $\phi_{\boldsymbol{\epsilon}}$ is real-valued (by symmetry) and also positive (by $\phi_{\boldsymbol{\epsilon}}(0) = 1$ and by the assumption that $\phi_{\boldsymbol{\epsilon}}$ is never zero). Thus, under the multivariate panel data structure, an estimator for the CF of an error term, $\phi_{\boldsymbol{\epsilon}}$, is given by

$$\hat{\phi}_{\boldsymbol{\epsilon}}(\boldsymbol{t}) = \sqrt{\left| \Re \left(\hat{\phi}_{\boldsymbol{\delta}}(\boldsymbol{t}) \right) \right|}, \ \boldsymbol{t} \in \mathbb{R}^2, \tag{2.13}$$

where $\Re(.)$ denotes the real part of a complex number and $\hat{\phi}_{\boldsymbol{\delta}}(\boldsymbol{t}) = k^{-1} \sum_{j=1}^{k} \exp(\imath \boldsymbol{t} \cdot \boldsymbol{\delta}_j)$, $\boldsymbol{t} \in \mathbb{R}^2$, is the empirical CF of the measurement differences $\{\boldsymbol{\delta}_j\}_{j=1}^k$.

Note that the estimator (2.13) of the (single) error term does not require stringent assumptions on the distribution of the errors. To our knowledge, empirical CFs seem to be the main nonparametric approach for estimating such error CFs in the deconvolution literature (albeit often in the non-panel data setting where an independent sample of pure errors is assumed to be available). In the simulations of Section 2.4, we also consider an alternative approach for estimating the error CF, for comparison to the empirical CF (2.13) of differences. This alternative approach uses measurement (or error) differences $\{\boldsymbol{\delta}_j\}_{j=1}^k$ to formulate a kernel density

estimator for the distribution of the differences; a Fourier transform of this density then provides an estimator $\hat{\phi}_{\boldsymbol{\delta}}$ of the CF $\phi_{\boldsymbol{\delta}}$. Instead of the empirical CF of differences, this estimator $\hat{\phi}_{\boldsymbol{\delta}}$ can then be substituted in (2.13) to approximate the error term CF $\phi_{\boldsymbol{\epsilon}}$, which generally produces a smoother version of the empirical CF based on differences. Suppose a kernel density estimator is used to approximate the density of $\boldsymbol{\delta}$, using a (bivariate) kernel L and bandwidth matrix Σ. The corresponding CF of the density estimator is then given by

$$\hat{\phi}_{\boldsymbol{\delta}}(\boldsymbol{t}) = \frac{1}{M}\sum_{j=1}^{M} \exp\left(\imath \boldsymbol{t}\cdot\boldsymbol{\delta}_j\right) L^*\left(\Sigma^{1/2}\boldsymbol{t}\right),\ \boldsymbol{t}\in\mathbb{R}^2. \tag{2.14}$$

In particular, if L is the product kernel (see Section 2.3.2) generated by the normal distribution, we have

$$\hat{\phi}_{\boldsymbol{\delta}}(\boldsymbol{t}) = \frac{1}{M}\sum_{j=1}^{M} \exp\left(\imath \boldsymbol{t}\cdot\boldsymbol{\delta}_j - \frac{1}{2}\boldsymbol{t}'\Sigma\boldsymbol{t}\right),\ \boldsymbol{t}\in\mathbb{R}^2. \tag{2.15}$$

We choose a normal kernel above because its Fourier transform (2.15) has an efficient, closed expression. Another possibility, not considered here, would be to use the CF of a normal mixture with a data-driven number of components. In Section 2.4, we specify how we select the bandwidth matrix Σ.

Ultimately, an estimated CF $\hat{\phi}_{\boldsymbol{\epsilon}}$ of the error (or an estimator $\hat{\phi}_{\bar{\boldsymbol{\epsilon}}}$) appears in the denominator of the deconvolution formula (2.7). This means that numerical approximations or estimates of this CF can potentially lead to instabilities for small values of the CF. Several authors have proposed ways to avoid computational issues when using an estimated error CF in the denominator of the deconvolution estimator. For example, Meister (2009) describes two regularization methods: either use a ridge parameter or truncate the denominator in the deconvolution formula when the value of the estimated CF is very small. In our simulations (see Section 2.4), we truncate the integrand when the denominator in (2.7) has small values, as in Neumann and Hössjer (1997). Namely, the term $1/\phi_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t}) = 1/[\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}/m)]^m$ is approximated by

$$\frac{1}{\hat{\phi}_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t})} I_{\{|\hat{\phi}_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t})| > b_{n,m}\}}, \tag{2.16}$$

for $\hat{\phi}_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t}) = [\hat{\phi}_{\boldsymbol{\epsilon}}(\boldsymbol{t}/m)]^m$ based on the estimated error CF $\hat{\phi}_{\boldsymbol{\epsilon}}$ (i.e., using one of the approaches described above) and for $b_{n,m} = 1/\sqrt{k}$ based on the number k of differences used to determine $\hat{\phi}_{\boldsymbol{\epsilon}}$ (i.e., k is equal to $nm(m-1)/2$, $n(m-1)$ or $nm/2$).

2.3.2 Choice of kernel function

In standard kernel density estimation, it is common to construct multivariate kernel functions starting with univariate kernels. The simplest approach is to define a multivariate kernel as a product of univariate kernels

$$K(\boldsymbol{x}) = K_1(x_1)K_1(x_2),\ \boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2, \tag{2.17}$$

where K_1 is a univariate kernel function. Such multivariate kernels (2.17) are called *product kernels* and their corresponding Fourier transform is given by

$$K^*(\boldsymbol{t}) = K_1^*(t_1)K_1^*(t_2),\ \boldsymbol{t} = (t_1, t_2) \in \mathbb{R}^2. \tag{2.18}$$

Another alternative, not pursued here, is the family of radial-symmetric kernels (cf. Wand and Jones (1995)).

Computation of the deconvolution kernel density estimator (2.4) or (2.7) involves integration over the domain $\mathbb{R}^2$ (i.e., for bivariate data). If a bivariate kernel K is chosen in such way that its Fourier transform K^* has bounded support, then the integral in (2.7) is restricted to a finite region. In what follows, we choose univariate kernels $K_1 = K_2$ that have Fourier transforms with support on an finite interval, say $[-1, 1]$. Then, without loss of generality, the resulting bivariate product kernel has Fourier transform K^* supported on $[-1, 1] \times [-1, 1]$. Because this bounded-support property simplifies integration and is also associated with kernels that have good properties, this type of kernel function is commonly used in deconvolution applications. Table 2.1 displays various univariate kernel functions K_1 and their respective Fourier transforms K_1^* for application in (2.18) and then (2.7).

Among univariate kernels K_1 with a Fourier transform with bounded support, the most common is the sinc kernel, whose Fourier transform K_1^* corresponds to a simple indicator function. The sinc kernel has well-known theoretical advantages (see below), but is not non-negative. Consequently, density estimates based on the sinc kernel tend to exhibit wiggly behavior at the tails, with an unattractive appearance; this behavior is illustrated wth numerical examples in Section 2.4.

As pointed out by Devroye (1989), kernels K_1 which perform well in estimation can often be associated with corresponding Fourier transforms that are flat around zero. The sinc kernel

Table 2.1 Kernel functions with their Fourier transforms, where $x, t \in \mathbb{R}$ and K_1 is continuously extended at zero in every case.

Name	Kernel (K_1)	Fourier Transform (K_1^*)
Sinc	$\frac{\sin(x)}{\pi x}$	$I_{[-1,1]}(t)$
de la Vallée-Poussin	$\frac{1-\cos(x)}{\pi x^2}$	$(1-\lvert t\rvert)\, I_{[-1,1]}(t)$
Triweight Ft	$\frac{48(1-15x^{-2})\cos x}{\pi x^4} - \frac{144(2-5x^{-2})\sin x}{\pi x^5}$	$\left(1-t^2\right)^3 I_{[-1,1]}(t)$
Tricube Ft	$\frac{648(-3x^4+90x^2-560)\sin(x)}{x^9\pi} + \frac{162(x^6-80x^4+1120x^2-2240)\cos(x)}{x^{10}\pi} + \frac{18(20160-x^6)}{x^{10}\pi}$	$(1-\lvert t\rvert^3)^3 I_{[-1,1]}(t)$
Flat-top Ft	$\frac{2}{\pi x^2}\left[\cos\left(\frac{x}{2}\right)-\cos(x)\right]$	$I_{\{\lvert t\rvert\le 1/2\}} + 2(1-\lvert\, t\,\rvert) I_{\{1/2\le\lvert t\rvert\le 1\}}$

has this feature, but other kernels are often adopted because they produce smoother estimates of the target density. An alternative to the sinc kernel is the de la Vallée Poussin kernel (VP kernel), defined in Table 2.1. This kernel is actually a probability density (i.e., is nonnegative), but its Fourier transform is sharp around zero and the numerical findings in Section 2.4 suggest that this kernel exhibits poor performance for bivariate deconvolution density estimation.

Between the sinc and VP kernels, there are other kernel functions whose Fourier transforms are flat around zero to varying degrees. We consider three of them (triweight, tricube, trapezoidal flat-top), described in Table 2.1. Figure 2.1 displays all the kernels we have mentioned and their Fourier transforms. The kernel with Fourier transform proportional to the triweight kernel was used by Delaigle and Gijbels (2004b,a), and resembles the kernel whose Fourier transform is proportional to the tricube kernel in Table 2.1. The flat-top kernel has been seemingly unknown in the deconvolution literature but has established properties in density estimation for time series (cf. Politis (2003b)). Section 2.4 demonstrates that the flat-top kernel produces density estimates with attractive shape properties while maintaining good (integrated squared error) performance relative to the other commonly used kernels for deconvolution estimation.

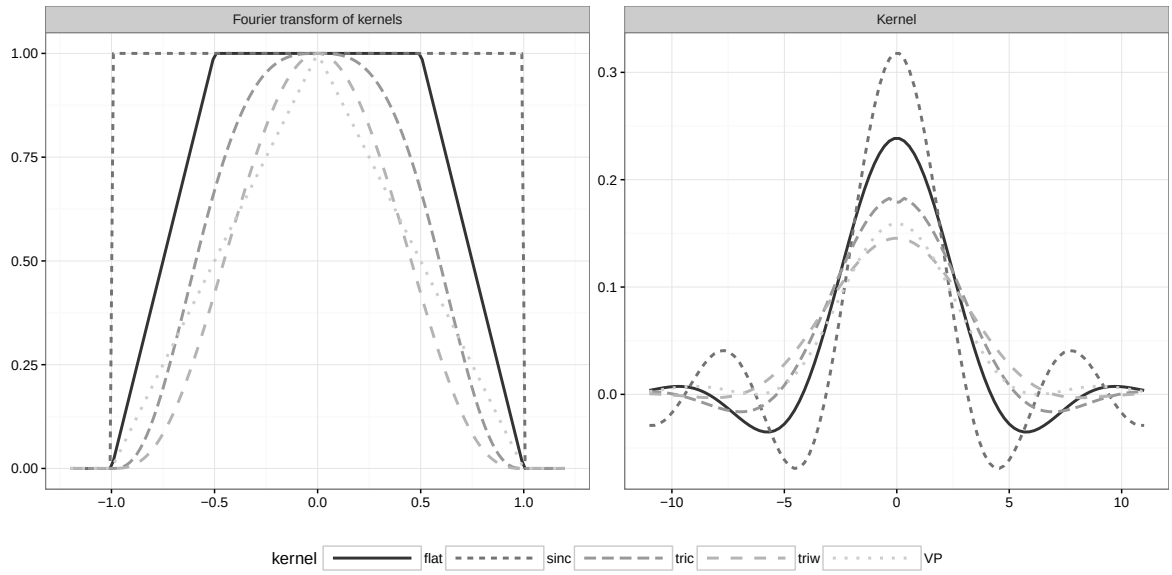

Figure 2.1 Plots of kernel functions and their Fourier transforms: Sinc, de la Vallée-Poussin (VP), triweight Fourier transform (triw), tricube Fourier transform (triw) and Flat-top Fourier transform kernels. The kernel function that oscillates the most is the sinc kernel.

2.3.3 Numerical Implementation

The deconvolution kernel density estimator $\hat{f}_{\boldsymbol{X},n}$ in (2.7) requires the computation of an integral that does not have a closed expression, and the oscillatory nature of the integrand introduces challenges for numerical approximations. We provide numerical integration recipes to address this practical issue. General-purpose software and algorithms for evaluating integrals in multiple dimensions do exist (mostly Monte Carlo methods); an example is the package called CUBA, described in Hahn (2005). However, when applying such methods to compute the estimator $\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x})$ at different values $\boldsymbol{x} \in \mathbb{R}^2$, the exponential term in the integrand in (2.7) is evaluated repeatedly for the same $\boldsymbol{x}$ values, which is time-consuming and computationally wasteful. Therefore, we propose approximating the integrals in (2.4) using faster and computationally more efficient approaches.

One simple approximation to the integral (2.7) is based on Riemann sums as

$$\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x}) = \frac{1}{(2\pi)^2}\int_{[-1,1]^2} e^{-\imath \boldsymbol{t}\cdot\boldsymbol{x}}\hat{\phi}_{\bar{\boldsymbol{Y}},n}(\mathbf{t})\frac{K^*(\boldsymbol{B}_n\boldsymbol{t})}{\phi_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t})}\mathrm{d}\boldsymbol{t} \approx \frac{\Delta}{(2\pi)^2}\sum_{\boldsymbol{t}\in G} e^{-\imath \boldsymbol{t}\cdot\boldsymbol{x}}\hat{\phi}_{\bar{\boldsymbol{Y}},n}(\mathbf{t})\frac{K^*(\boldsymbol{B}_n\boldsymbol{t})}{\phi_{\bar{\boldsymbol{\epsilon}}}(\boldsymbol{t})}, \tag{2.19}$$

where G denotes a regular grid on $[-1,1]^2$ of $L_1 \cdot L_2$ (L_i points in the i-th coordinate) and Δ denotes the area between grid points. Note that the integration limits are $[-1,1]^2$ rather than $\mathbb{R}^2$ when we focus on kernel functions with Fourier transforms supported on $[-1,1]^2$. (Furthermore, in practice, one would estimate $1/\phi_{\bar{\boldsymbol{\epsilon}}}(\cdot)$ as described in Section 3.1.)

In application, we also need to evaluate the density estimator $\hat{f}_{\boldsymbol{X},n}(\cdot)$ over a grid of points of $\boldsymbol{X}$ so that probability estimates can be obtained by numerically integrating $\hat{f}_{\boldsymbol{X},n}(\cdot)$. This is a different matter than the problem of computing $\hat{f}_{\boldsymbol{X},n}(\cdot)$ at a single point $\boldsymbol{x}$, see Comte and Lacour (2010). For that purpose, we carry out density estimation $\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x})$ on a regular grid of points $\boldsymbol{x}$, say on $[c_1,d_1]\times[c_2,d_2]$ with $L_3 \cdot L_4$ points; here c_1, d_1, c_2, d_2 denote the bounds of a rectangular region for computing density estimators, selected by the user.

We considered two approaches for computing the Riemann sums in (2.19). The first approach computes all quantities in (2.19) which do not depend on $\boldsymbol{x}$ (requiring evaluation only on the subgrid $G \subset [-1,1]^2$) and then evaluates exponential terms at $\boldsymbol{x}$-points on the $[c_1,d_1]\times[c_2,d_2]$ to determine the Riemann sums. Another alternative is to use the Fast Fourier Transform (FFT) to compute the approximating sums in (2.19). We used the FFT method developed in Inverarity (2002a), which is the generalization to the multivariate case of Bailey and Swarztrauber (1994).

The FFT forces $L_1 = L_3$ and $L_2 = L_4$, which is a restriction compared to the plain Riemann sum method. However, the FFT approach is much faster, as can be seen in Figure 2.2. Figure 2.2 compares the time required to obtain density estimates on a grid (using simulated panel data described in Section 2.4) based on both the FFT and plain Riemann sum approaches. (For the sake of simplicity, to produce the Figure 2.2, we considered the same resolution for evaluating both methods, that is, $L_1 = L_2 = L_3 = L_4$). Note that both approaches have the same numerical accuracy as each computes the same Riemann approximation (2.19) to the integral. However, more accurate approximations to this integral require higher resolution grids $G \subset$

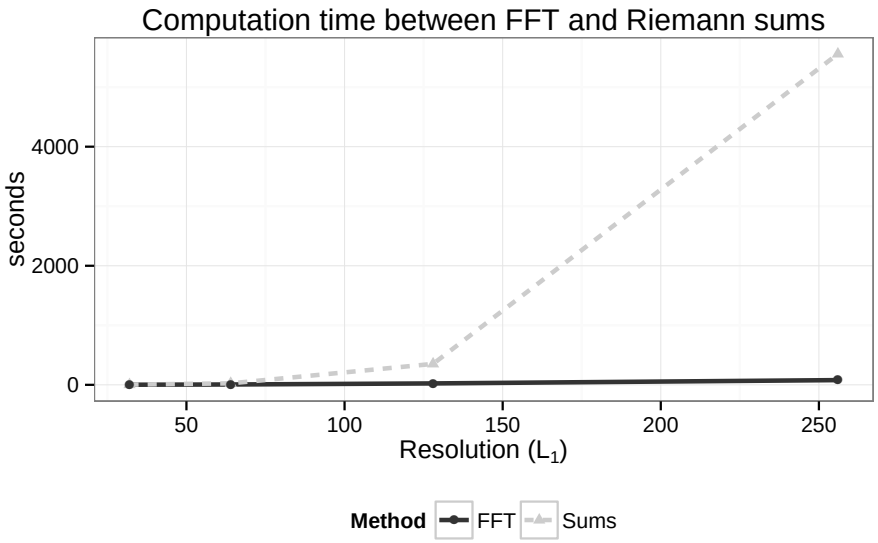

Figure 2.2 Speed comparison of integrating procedures. The Fast Fourier Transform (FFT) is substantially faster than plain Riemann sums for increasingly larger approximation grids.

$[-1, 1]^2$, creating larger computational demands where the FFT has speed advantages. Hence, we recommend the use of the FFT based approach in practice to approximate integrals for bivariate deconvolution estimators. In what follows, we implemented the FFT approximation to carry out all calculations in the simulation studies to follow.

We have also developed a software package called `fourierin`, made available in R, for computing multivariate deconvolution estimators on a grid through the FFT-based numerical approximations of the required integrations; see Basulto-Elias et al. (2016).

2.4 Simulation study

2.4.1 Design of the simulation study

Next, we examine several aspects which impact the performance of bivariate kernel deconvolution through simulation. We begin by describing distributions for data generation and for the underlying target density. Fan (1991b) characterizes distributions according to the *s*moothness of their CF. In our context, the smoother the error distribution, the harder it becomes to esti-

mate the target density of the underlying signal variable. Formally, a (possibly multivariate) random variable $\boldsymbol{X}$ is said to be *ordinary smooth* (OS) if there exist positive constants β, d_0 and d_1 such that

$$d_0|\boldsymbol{t}|^{-\beta} \leq |\phi_{\boldsymbol{X}}(\boldsymbol{t})| \leq d_1|\boldsymbol{t}|^{-\beta}, \qquad \text{as } |\boldsymbol{t}| \to \infty,$$

and it is said to be *supersmooth* (SS) if there exist positive constants β, δ, γ, d_0 and d_1 such that

$$d_0|\boldsymbol{t}|^{-\beta} \exp\left(-\gamma|\boldsymbol{t}|^{\delta}\right) \leq |\phi_{\boldsymbol{X}}(\boldsymbol{t})| \leq d_1|\boldsymbol{t}|^{-\beta} \exp\left(-\gamma|\boldsymbol{t}|^{\delta}\right), \qquad \text{as } |\boldsymbol{t}| \to \infty,$$

where $|\cdot|$ is the Euclidean norm. Unlike OS variables, SS random variables have densities which are infinitely differentiable. In the univariate case, examples of OS random variables include those with the gamma distribution and the Laplace distribution while examples of SS distributions include Gaussian and Cauchy densities. To understand the impact of distribution smoothness on deconvolution estimators here, note that the empirical characteristic function (CF) of the contaminated samples appears in the numerator of (2.4), while the CF of the error appears in the denominator. This same ratio of CFs also arises in asymptotic expansions for mean squared error of deconvolution density estimators. As a consequence, supersmooth error terms then lead to slower convergence rates in density estimation, because they introduce an additional exponential term into such expansions.

For the simulation, we then considered the following design factors related to data-generating process (item 1 below), number of available replicates (item 2) and estimation implementation (items 3-5) in order to examine their impact on the deconvolution estimators with panel data.

1. We considered four combinations of OS/SS signal/error distributions along with three varying levels of noise to signal variance ratios $(0.2, 1, 2)$, as described further below.

2. We assumed either $m = 2$ or $m = 8$ replicated measurements for each individual in a panel data model (2.5) with $n = 150$ individuals.

3. We compared the five kernel functions described in Table 2.1.

4. We implemented the three procedures described Section 2.3.1 for creating "error differences" to approximate the CF of the error.

Table 2.2 Distributions used in the study for signal and noise. Both signal distributions have the same mean and covariance matrix and both error distributions have the same covariance matrix. The bivariate gamma is defined here through a Gaussian copula with density is given $f_{\boldsymbol{G}}(g_1, g_2) = \frac{f_{G_1}(g_1)}{\phi\left(\Phi^{-1}(F_{G_1}(g_1))\right)} \cdot \frac{f_{G_2}(g_2)}{\phi\left(\Phi^{-1}(F_{G_2}(g_2))\right)} \times f_{\boldsymbol{Z}}\left(\Phi^{-1}\left(F_{G_1}(g_1)\right), \Phi^{-1}\left(F_{G_2}(g_2)\right)\right)$ for $g_1, g_2 > 0$ depending on parameters, α_1, α_2, β_1, $\beta_2 > 0$ and $-1 < r < 1$. For $i = 1, 2$, F_{G_i} and f_{G_i} denote the distribution and density functions of a gamma variable with shape α_i and rate β_i parameters; Φ^{-1} and ϕ are the quantile function and the density of a standard normal distribution; and $f_{\boldsymbol{Z}}$ denotes a bivariate normal density with correlation r and standard normal marginal distributions.

Type	Distribution
OS signal	Bivariate gamma with mean $(10, 2)$ and cov. $\begin{bmatrix} 5 & -1.75 \\ -1.75 & 2 \end{bmatrix}$, whose marginals are gamma with shape parameters 20 and 2 and rate paremeters 2 and 1, respectively.
SS signal	Bivariate normal with mean $(10, 2)$ and cov. $\begin{bmatrix} 5 & -1.75 \\ -1.75 & 2 \end{bmatrix}$.
OS noise	Laplace with mean $(0, 0)$ and correl. matrix $\begin{bmatrix} 1 & .5534 \\ .5534 & 1 \end{bmatrix}$.
SS noise	Normal with mean $(0, 0)$ and correl. matrix $\begin{bmatrix} 1 & .5534 \\ .5534 & 1 \end{bmatrix}$.

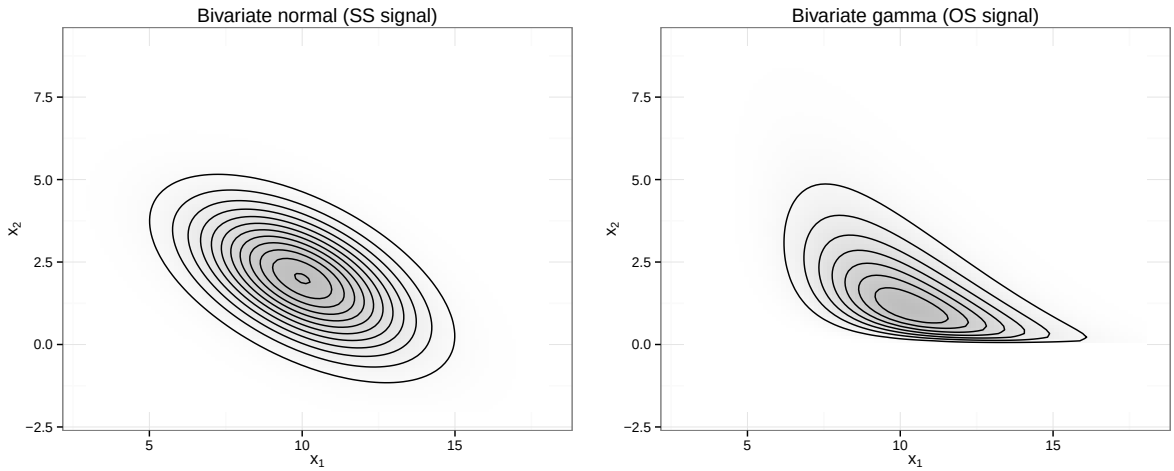

Figure 2.3 Contour plots considered for the study. Both distributions have the same mean and covariance matrix. Observe that the coordinates are correlated in both distributions. The bivariate gamma was generated according to Table 2.2.

5. We implemented two different approximations to the characteristic function based on error differences: either the empirical CF from differences or the Fourier transform of a kernel density estimator applied to differences, as described in Section 2.3.1.

Hence, we include four combinations of ordinary smooth (OS) and supersmooth (SS) distributions across the signal and the noise (item 1), where random errors are generated either from a bivariate normal distribution (SS error) or a bivariate Laplace distribution with the same covariance matrix (OS error), and where signal random variable is generated from either a bivariate normal distribution (SS signal) or from a bivariate gamma distribution (OS signal). Table 2.2 shows two signal distributions (OS/SS) have the same mean and variance properties, while Figure 2.3 shows the contour plot of the distribution of the signal. With regard to noise to signal ratios (item 1), we considered three ratios $0.2, 1, 2$ to represent low to high levels of error. These ratios were component-wise fixed for the marginal distributions; that is, for a noise to signal ratio of 0.2, the first components of the error and signal have a variance ratio of 0.2, and the same holds for the second component. For the kernel density-based approximation to the CF of the errors (item 5), we used a normal kernel, as described in Section 2.3.1, with a diagonal bandwidth matrix obtained with a plug-in method computed using the R package `ks` (see Duong et al. (2007)).

In computing the deconvolution estimators, we used a grid of 64×64 points to approximate the integral in (2.7) via fast Fourier transform (as described in Section 2.3.3) and also for evaluating the density of the signal variable. There was no significant improvement in precision for larger grid sizes. Moreover, the region for the density estimation was fixed at $[3, 18] \times [-2, 9]$, which contains at least 99% of the mass of both OS/SS signal distributions considered (see Figure 2.3). For each simulated panel data set and deconvolution estimator, we evaluated the corresponding the integrated squared error (ISE) through the approximation

$$\int \left[\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x}) - f_{\boldsymbol{X}}(\boldsymbol{x})\right]^2 \mathrm{d}\boldsymbol{x} \approx \Delta \sum_{ij} \left[\hat{f}_{\boldsymbol{X},n}(\boldsymbol{x}_{ij}) - f_{\boldsymbol{X}}(\boldsymbol{x}_{ij})\right]^2,$$

where $\Delta = \tau_1\tau_2/64^2$ is the area of every grid rectangle for $\tau_1 = 11$ and $\tau_2 = 15$, and $\boldsymbol{x}_{ij} = (-2 + i\tau_1, 3 + j\tau_2)$, $i, j \in \{0, \ldots, 63\}$, are the grid points. ISEs for all deconvolution estimators

were computed over 200 simulation runs per data-generating process, and are summized in figures to follow (e.g., boxplots).

Finally, we make some comments about bandwidth effects, which we sought to control in order to quantify the impact of the factors mentioned above (e.g., type of kernel) for multivariate panel data with unknown errors. Bandwidth selection is a challenging problem in kernel deconvolution with a rich literature for univariate data (see Delaigle and Gijbels (2004b)), but far less is known for multivariate data. To minimize the compounding effect of bandwidth choice, we used the diagonal bandwidth matrix for minimizing the integrated squared error (ISE) for deconvolution kernel density estimators based on known (not estimated) measurement and error CFs. That is, we employ a diagonal bandwidth matrix for each kernel by choosing the best approximate bandwidths for these kernels, but we do not assume that the distributions of the errors or the signal are known in the simulation study.

2.4.2 Simulation results

To present the simulation findings, we mention two reference estimators that we adopted in order to help frame the performances of bivariate deconvolution estimators. These reference estimators are obtained from kernel density estimators (without deconvolution) applied to samples defined by either (2.20) or (2.21) as

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \qquad \text{(Error free)} \tag{2.20}$$

$$\bar{\boldsymbol{Y}}_{1\cdot}, \ldots, \bar{\boldsymbol{Y}}_{n\cdot} \qquad \text{(Contaminated)} \tag{2.21}$$

using the normal kernel and a plug-in bandwidth matrix (from the `R` package `decon`). The first estimation reference, which serves as a type of lower bound in estimation error (*error-free* in plots), is the usual bivariate kernel density estimation applied to the sample $\{\boldsymbol{X}_i\}_{i=1}^n$ without measurement error, which is unobservable in practice. However, one wishes for the ISE of a deconvolution estimator to be close to the ISE of this reference estimator, which can be generally expected to exhibit better performance than deconvolution counterparts based on data with true errors in variables. The second reference estimator (*contaminated* in plots) aims to provide an upper bound on estimation error and corresponds to the usual bivariate kernel

density estimator applied to the measurement averages per individual $\{\bar{\boldsymbol{Y}}_i\}_{i=1}^n$ from (2.6). Note that this reference estimator *ignores* measurement errors inherent to the averages (2.21). Hence, as the purpose of kernel deconvolution is precisely to account for measurement error, one would like for devoncolution density estimators to have smaller ISEs than this reference.

Figure 2.4 provides a comparison of average ISEs for the different kernel choices in deconvolution estimation with panel data across the four distributional cases of ordinary smooth/ supersmooth (OS/SS) signal and OS/SS noise (with $m = 2$ replicates per individual and noise to signal ratio of one). The figure also considers the type of estimation for the error/noise characteristic function, based on either the empirical characteristic function (ecf) or the characterstic function of a kernel density estimator (kde) (formed by differences between measurement replicates). As expected, SS error distributions tend to induce larger ISEs on average compared to OS errors and the smoothness of the underlying signal also impacts performance of deconvolution estimators. However, the effect of the kernel choice is qualitatively similar in Figure 2.4, where the sinc kernel has the best performance closely followed by the flat-top kernel in every scenario. The estimaton approach for the error charactersitic function (i.e., ecf or kde) typically resulted in only small differences in performance, with a small but consistent edge for the kde-based method. Note also that deconvolution estimators often substantially improve upon the contaminated reference in Figure 2.4 (upper horizontal bar) which ignores measurment error in density estimation, but as expected perform worse than the estimation reference when no measurment error is present (lower horizontal bar).

A similar pattern emerges in Figure 2.5, which considers the effect of both differing noise to signal ratios and varying amounts of replication ($m = 2, 8$). From the figure, the expected estimation error (ISE) decreases when the number of replicates grows from $m = 2$ to $m = 8$, and errors also increase as the noise to signal ratio increases. However, regardless of these factors, the results are again qualitatively similar across types of kernel choices, and the flat-top and sinc kernels emerge as the best performers. In particular, with higher numbers of replicates and low noise to signal ratios (ideal inference situations with panel data), deconvolution estimators based on flat-top and sinc kernels achieve error rates similar to the oracle type reference of kernel estimation based on observations with no measurement error (lower horizontal bar in

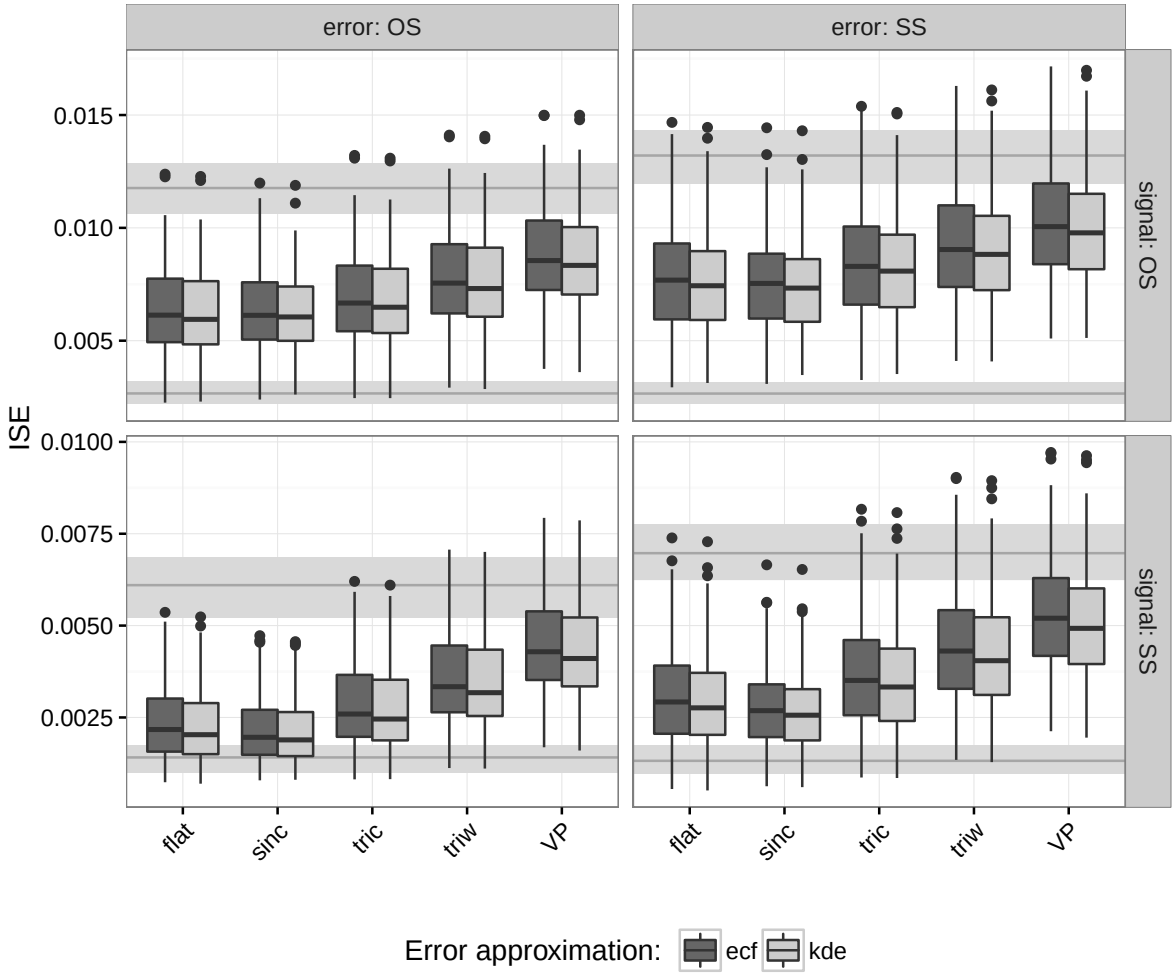

Figure 2.4 Comparison of estimators for different combinations of smoothness when the noise to signal ratio is equal to 1. Observe that the SS signal and OS noise has the closest performance to the error free case when sinc and flat-top kernels are used. On the other hand, the worst performance overall occurs in the OS signal and SS noise, which is expected. The two background bands represent the three quartiles of the ISE computed for the kernel density estimator of the error free observations (lower band) and the kernel density estimator of the contaminated sample ignoring the noise (upper band). This plot was generated for $m = 2$ replicate observations per individual and noise to signal ratio of one.

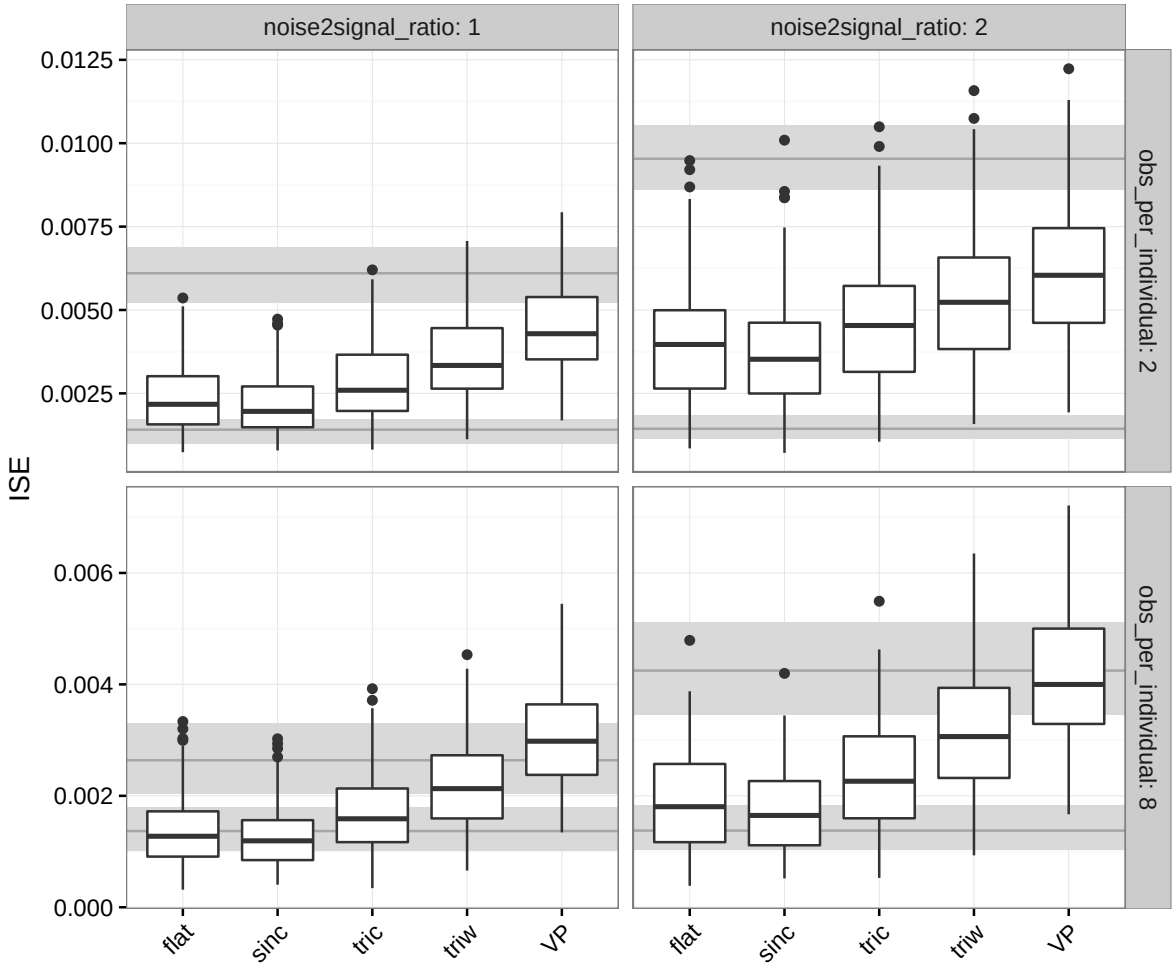

Figure 2.5 Comparison of estimators for different noise-to-signal ratio and number of observations per individuals when the empirical characteristic function has been used for approximating the error with SS signal and OS noise. The two background bands represent the three quartiles of the ISE computed for the kernel density estimator of the error free observations (lower band) and the kernel density estimator of the contaminated sample ignoring the noise (upper band).

Figure 2.5), which also supports these kernel choices. Again, ignoring measurement error (simply averaging observations over replicates and applying ordinary kernel density estimation techniques) is seen to create larger estimaton errors compared to deconvolution approaches (upper horizontal bar in Figure 2.5), as particularly evident for small replication $m = 2$.

We mentioned in Section 2.3.1 that there are several strategies to obtain differences to approximate the characteristic function of the error. Specifically, taking all the possible difference of observations for the same individual (cf. (2.9)), which leads to many correlated observations of differences of errors; taking differences of paired columns, obtaining fewer but independent error differences (cf. (2.10)), and everything minus the first repetition for individual, which is an compromise between number of observations and independence (cf. (2.8)). Figure 2.6 compares performances for deconvolution estimators based on this type of pre-processing used for creating error differences from replicate measurements in the panel data structure. This illustration considers the $m = 8$ replicate case (as the type of differencing is not an issue with $m = 2$ replications). Essentially from this figure, the type of pre-processing has surprisingly little effect, which remains true as the noise ratio increases.

2.4.3 Kernel choices

Among the previous simulation findings, the choice of kernel emerges as the single most influential factor for the performance of deconvolution estimators to be considered in application. While the sinc kernel (sinc) has been widely used in the literature for univariate deconvolution and also exhibited consistently good performance in our simulations, the flat-top kernel emerged with a performance similar to the sinc kernel in terms of average ISE. In fact, in many simulation cases, the mean ISEs of deconvolution estimators from both sinc and flat-top kernels were close to the reference estimator based on error free samples. However, an additional potential advantage of the flat-top kernel is that this tends to produce density estimators without the "wiggly" (and sometimes heavily negative) surfaces assoicated with the sinc kernel. Such artifacts are a well-known and undesirable feature of sinc kernel-based estimates, which becomes a compounding issue in the multivariate setting.

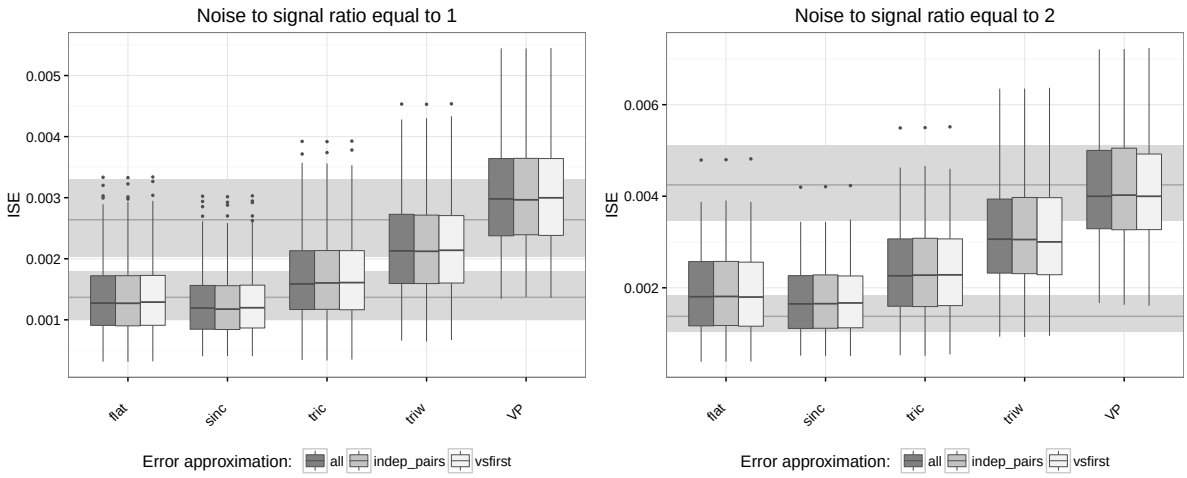

Figure 2.6 The estimation performances based on pre-processing mechanism used to compute the differences among $m = 8$ replicates per individual (for approximating the characteristic function of noise). The left plot considers a noise to signal ratio of 1 and the right plot involves a ratio of 2. The two background bands represent the three quartiles of the ISE computed for the kernel density estimator of the error free observations (lower band) and the kernel density estimator of the contaminated sample ignoring the noise (upper band). This plot was generated using the empirical characteristic function to estimate the error (results were similar the characterstic function of a kernel density estimator, cf. (2.15), Section 3.1) with OS signal/SS noise data generation.

To illustrate, Figure 2.7 presents the deconvolution density estimates based on different kernel types, produced from a simulated data set of OS noise and OS signal with a noise to signal ratio of 0.2, two replicates per individual ($m = 2$) and 150 individuals ($n = 150$). From the figure, the sinc kernel yields an estimate with the most pronounced oscillations, leading to a substantial negative part in the density estimate unlike the other kernel approaches shown. In contrast, the flat-top kernel provides a density estimate with less wavy and negative artifacts. While less prominent, some negative portions in the density estimate do still appear with the flat-top kernel in Figure 2.7, but this aspect is difficult to completely remove. For comparison and perspective, among the kernels in Figure 2.7, only the de la Vallé-Poussin kernel (VP) produces a non-negative density estimate. However, from the previous simulations with deconvolution estimators, the VP kernel also induced the worst performances in terms of average ISE, sometimes no better than the contaminated reference estimator that ignored measurment error. In this sense, the flat-top kernel in multivariate deconvolution estimation may lead to comparable performances in terms of ISE compared to the sinc kernel (being the best among all the kernel functions considered here), but with more attractive properties in the shape and appearance of its density estimator.

2.5 Conclusions and promising research avenues

We examined the performance of several different methods for deconvolution density estimation with bivariate panel data (repeated measurements for individual subject to distortion by measurement error). While deconvolution estimation has received much attention for univariate data, there has been less consideration for multivariate data and, in particular, for panel data structures which arise in scientific studies. For such data, we examined the impact of several choices available to the practitioner when implementing a density estimator for the underlying signal variable of interest. These include the type of kernel, the approach for processing replicate measurements in order to estimate error distributions, and the type of estimator of the error characteristic function. One general observation is that the performance of deconvolution density estimators could be improved by using kernel density estimators (and their characteristic functions) formed from error terms rather than the empirical characteristic

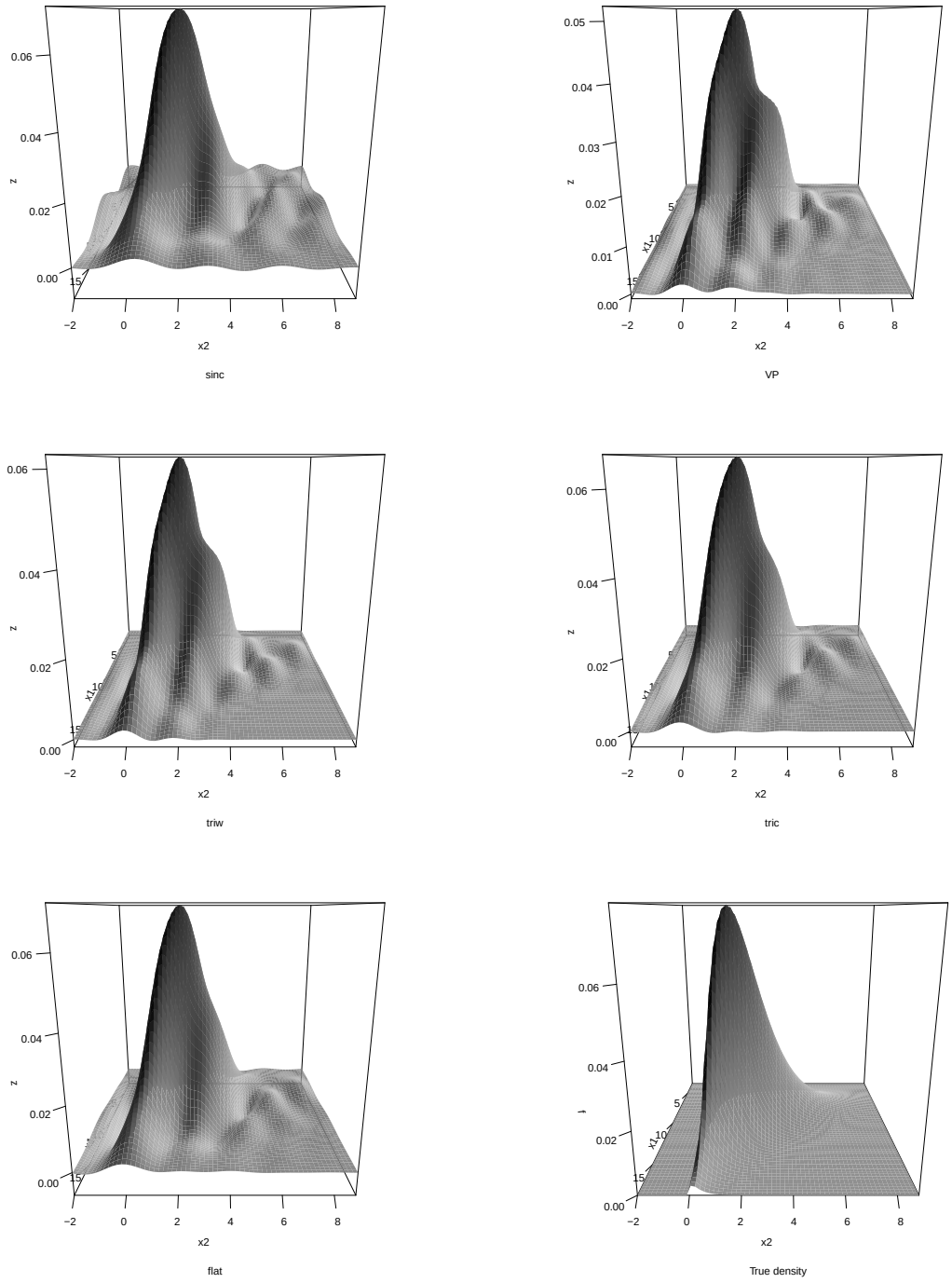

Figure 2.7 Perspective plots of the true density and kernel deconvolution density estimates based on a sample size of $n = 150$ varying the kernel function. The sinc kernel (upper left corner) possess the greatest oscillations

functions from errors; yet the latter is most commonly proposed in the literature. However, among possibilities considered, we found that the most important and substantial feature impacting bivariate kernel deconvolution is type of the kernel function. Traditionally, the sinc kernel has been popular for univariate deconvolution, because its Fourier transform is very simple and exhibits good properties in terms of integrated squared error; we show that those features continue to be important for bivariate panel data. However, we also found that flat-top kernels result in similar performance to the sinc kernel in terms of estimation error, while producing bivariate density estimates with more attractive geometries and fewer wavy artifacts. We have also described numerical integration recipes based on FFTs in order to quickly and practically implement deconvolution estimators in the multivariate data setting; R software and illustrations of implementation are available in a Github repository in Basulto-Elias (2016b).

Open research issues remain mostly associated with the development of theory for multivariate deconvolution estimation with panel data. Two specific open problems are determining rates of convergence of the kernel density estimators and defining optimal bandwidth choices. A related and important issue for investigation concerns data-driven approaches for selecting bandwidths that are suitable for panel data in the multivariate case.

CHAPTER 3. "FOURIERIN": AN R PACKAGE TO COMPUTE FOURIER INTEGRALS

A paper to be submitted to The R Journal

Guillermo Basulto-Elias, Alicia L. Carriquiry, Kris De Brabanter and Daniel J. Nordman

Abstract

We present our `R` package "fourierin" (cf. (Basulto-Elias et al., 2016)) intended to numerically compute Fourier-type integrals of functions of one or two variables and finite support in several points. If the points belong to a regular grid, the algorithm described in Inverarity (2002a) is implemented making significant improvements in the speed.

3.1 Introduction

Continuous Fourier transforms commonly appear in several areas, such as physics and statistics. In probability theory, for example, continuous Fourier transforms are related to the characteristic function of a distribution.

The definition of a continuous Fourier transform may change from one context to another. It is desirable then to have a function that easily adapts to every definition of continuous Fourier transform as well as its inverse.

Fourier integrals cannot always be solved in elementary terms and the oscillating nature of the makes typical numerical integration recipes to fail. Bailey and Swarztrauber (1994) presents an algorithm that expresses the mid-point integration rule in terms of the discrete Fourier transform, which allows the use of the Fast Fourier transform. Inverarity (2002a) extended such characterization to the multivariate case.

Since `R` is one of the most popular programming languages among statisticians, it is useful to have tools to compute Fourier integrals. We have not found an `R` package that does this specific type of integrals in general, although it is required by some statistical procedures. See Wang and Wang (2011a) for an application in kernel deconvolution; Fourier integrals are implemented for particular univariate cases.

The package "fourierin" has a main function (`fourierin`) that performs Fourier-type integrals and it has arguments that easily adapt to several definitions of continuous Fourier transform and its inverse (cf. Inverarity (2002b)). The heavy computations are done in `C++` via "RcppArmadillo" package (cf. Eddelbuettel and Sanderson (2014)).

The rest of the paper has four sections. First, we describe the Fourier integral and the numerical approximation to be used in "Fourier Integrals and FFT". Then, in "Speed Comparison" we compare the fast and the slow implementations in one and two dimensions at several grid sizes. Then we show how the function must be used. Finally, in "Summary", we present conclusions and upcoming extensions to the `R` package.

3.2 Fourier Integrals and FFT

Our package aims to compute Fourier integrals at several points simultaneously. This is, the computation of the integral

$$\left[\frac{|s|}{(2\pi)^{1-r}}\right]^{n/2}\int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_n}^{b_n} f(\boldsymbol{t})\exp\{\imath s\langle \boldsymbol{w},\boldsymbol{t}\rangle\}\mathrm{d}\boldsymbol{t}, \tag{3.1}$$

at more than one point $\boldsymbol{w}\in\mathbb{R}^n$, where f is a n-variate function that takes real or complex values, for $n\in\{1,2\}$, $a_j<b_j$ for $j=1,\ldots,n$ and $s,r\in\mathbb{R}$. Observe that the constants s and r allow flexibility on the definition of continuous Fourier transforms and its inverse. We are particularly interested in fast computation of the integral (3.1) evaluated in a regular grid, say, at $\boldsymbol{w}^{(j_1,\ldots,j_n)}=(w_{j_1},\ldots,w_{j_n})$ and $w_{j_k}=c_k+j_k\Gamma_k$, with $\Gamma_k=(d_k-c_k)/n_k$ for $k=1,\ldots,n$.

We approximate the integral in (3.1) with a sum using the mid-point rule:

$$\prod_{j=1}^{n}(b_j-a_j)\cdot\left[\frac{|s|}{(2\pi)^{1-r}}\right]^{n/2}\sum_{i_1=0}^{m_1-1}\sum_{i_2=0}^{m_2-1}\cdots\sum_{i_n=0}^{m_n-1} f(\boldsymbol{t})\exp\{\imath s\langle \boldsymbol{w},\boldsymbol{t}^{(i_1,\ldots,i_n)}\rangle\}, \tag{3.2}$$

where $\boldsymbol{t}^{(i_1,\ldots,i_n)} = (t_{i_1},\ldots,t_{i_n})$ and $t_{i_k} = a_k + i_k\Delta_k + \Delta_k/2$, with $\Delta_k = (b_k - a_k)/n_k$ for $k = 1,\ldots,n$.

The sum (3.2) can be written in terms of discrete Fourier transforms and inverses of discrete Fourier transforms, details of this derivation can be found in Inverarity (2002a). We are using the notation in that paper. It is well known that using the FFT can improve the complexity of computing a discrete Fourier transform from $O(m^2)$ to $O(m \log m)$ in the univariate case, where m is the size of the vector. The complexity of computing the multivariate discrete Fourier transform of an n-dimensional array using the FFT is $O(M \log M)$, where $M = m_1 \cdots m_n$ and m_j is the size of the j-th dimension, $j = 1,\ldots,n$.

3.3 Speed Comparison

Here we compare the differences in execution times for univariate and bivariate cases. Figure 3.1 is a univariate example of the integral in (3.1) evaluated on a regular grid and Figure 3.2 is a bivariate example. Note that the time units are different, since computing the bivariate case takes longer.

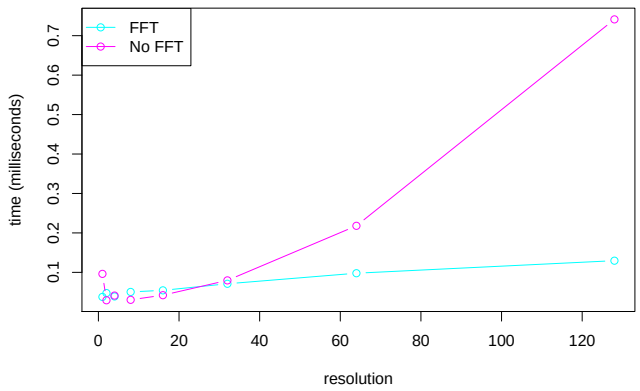

Figure 3.1 Example of a univariate Fourier integral over grids of several (power of two) sizes. Specifically, the standard normal density is being recovered using the Fourier inversion formula. Time is in milliseconds. The Fourier integral has been applied five times for every resolution and each dot represents the mean for the corresponding grid size and method.

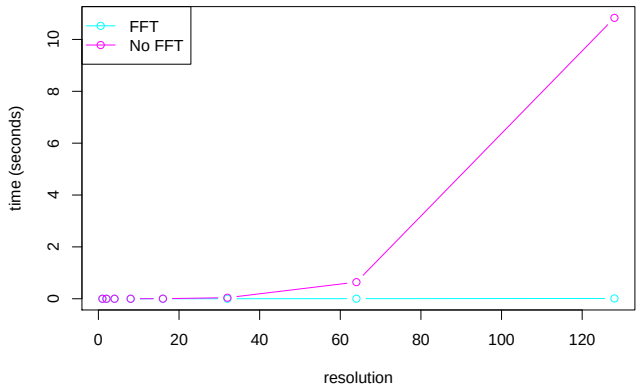

Figure 3.2 Example of a bivariate Fourier integral over grids of several (power of two) sizes. Both axis have the same resolution. Specifically, the characteristic function of a bivariate normal distribution is being computed. Time is in seconds (unlike 3.1). The Fourier integral has been applied five times for every resolution and each dot represents the mean for the corresponding grid size and method.

3.4 Examples

In this section we will present examples of usage of the "fourierin" package. First, we will start with a univariate example which further illustrates two cases in which the FFT is not used (and it is slower as a consequence). The second example is two dimensional and we compare the real and the imaginary parts of an approximation using Fourier integrals to the true values.

The next code shows how the package can be used for univariate cases. The example we consider is to recover a normal density, f from its characteristic function, ϕ, where,

$$f(x) = \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-3)^2\right] \qquad \text{and} \qquad \phi(t) = \exp\left(3t\imath - \frac{1}{2}t^2\right), \tag{3.3}$$

for all $t, x \in \mathbb{R}$.

```
## ------------------------------------------------------------------
##              1D example with complex-valued function
## ------------------------------------------------------------------


library(fourierin)
```

```
                                        # Parameters
mu <- 3
lower <- -4 + mu
upper <- 4 + mu
resolution <- 64
                                        # Compute integral
out <- fourierin(f = function(t) exp(1i*mu*t - t^2/2),
                 a = -5, b = 5, c = lower, d = upper,
                 r = -1, s = -1, resol = resolution,
                 use_fft = FALSE)
                                        # Extract grid and values
grid <- out$w
values <- Re(out$values)
true <- dnorm(grid, mu)
                                        # Plot parameters
colors <- c("cyan", "magenta")
legends <- c("approximated", "true")
line_type <- 2:3
line_width <- 1.5
                                        # Generate and plot
plot(range(grid), range(values, true), type = "n",
     xlab = "x", ylab = "values")
lines(grid, values, col = colors[1], lty = line_type[1], lwd = line_width)
lines(grid, true, col = colors[2], lty = line_type[2], lwd = line_width)
legend("topleft", legend = legends, col = colors,
       lty = line_type, lwd = line_width)

                                        # Evaluate in a non-regular
                                        # grid
```

```
new_grid <- c(0, 3, 4)
fourierin(f = function(t) exp(1i*mu*t - t^2/2),
          a = -5, b = 5, r = -1, s = -1, resol = resolution,
          w = new_grid)
## fourierin(f = function(t) exp(1i*mu*t - t^2/2),
##           a = -5, b = 5, r = -1, s = -1, resol = resolution,
##           w = new_grid)
##                [,1]
## [1,] 0.004432043-0i
## [2,] 0.398942058+0i
## [3,] 0.241970623+0i
```

Note that the first call of the `fourierin` function has the argument `use_fft = FALSE`. This could be set to be equal to `TRUE` to use the algorithm described in Inverarity (2002a) for regular grids, which uses the FFT and is substantially faster (see Figure 3.1).

On the other hand, at the bottom of the code we show how `fourierin` works when a (non-regular) grid is provided. Observe that if this is the case, only the Fourier integral values are returned; unlike the case where a regular grid is used, in which a list is returned with the Fourier integral values and an integration grid is returned. The function f must be able to be evaluated at vectors.

Figure 3.3 is the plot generated in the latter code. A low resolution has been set in order to observe differences between the true density and its recovered version using Fourier integrals.

Finally, we use a bivariate normal density, f, to find its characteristic function ϕ. This illustrates how `fourierin` function works for bivariate functions. In particular,

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \quad \text{and} \quad \phi(\boldsymbol{t}) = \exp\left(\imath\boldsymbol{x}'\boldsymbol{\mu} - \frac{1}{2}\boldsymbol{t}'\Sigma\boldsymbol{t}\right), \tag{3.4}$$

for all $\boldsymbol{t}, \boldsymbol{x} \in \mathbb{R}^2$, with $\boldsymbol{\mu} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $\Sigma = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$.

```
## ------------------------------------------------------------------
```

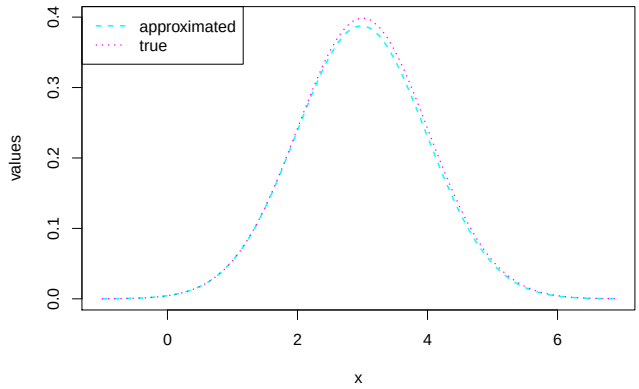

Figure 3.3 Example of `fourierin` function for univariate function at resolution 64: Recovering a normal density from its characteristic function. See Equation 3.3.

```
##              2D example with real-valued function
## ------------------------------------------------------------------


library(fourierin)


## Parameters of bivariate normal distribution
mu <- c(-1, 1)
sig <- matrix(c(3, -1, -1, 2), 2, 2)
resolution <- 128


## Multivariate normal density. x is n x d
f <- function(x) {
                                        # Auxiliar values
    d <- ncol(x)
    z <- sweep(x, 2, mu, "-")


                                        #Get numerator and denominator
                                        #of normal density
```

```
    num <- exp(-0.5*rowSums(z * (z %*% solve(sig))))
    denom <- sqrt((2*pi)^d*det(sig))

    return(num/denom)
}

## Characteristic function. s is n x d
phi <- function(s) {
    complex(modulus = exp(-0.5*rowSums(s*(s %*% sig))),
            argument = s %*% mu)
}

## Compute values
                                        # Approximate cf using Fourier
                                        # integrals
eval <- fourierin(f, a = c(-8, -6), b = c(6, 8),
                  c = c(-4, -4), d = c(4, 4),
                  r = 1, s = 1, resol = c(resolution, resolution))
t1 <- eval$w1
t2 <- eval$w2
t <- as.matrix(expand.grid(t1 = t1, t2 = t2))
approx <- eval$values
true <- matrix(phi(t), resolution, resolution)        # Compute true
                                                      # values

## Generate plots
                                        # Plot parameters
colors <- c("cyan", "magenta")
legends <- c("approximated", "true")
```

```
line_type <- 2:3
line_width <- 1.5


                                        # Generate and plots
                                        # Real part section
i <- resolution/2 + 1
plot(t2, Re(approx[i, ]), type = "l", col = colors[1],
     lty = line_type[1], lwd = line_width,
     ylab = "",
     xlab = expression(t[2]),
     main = expression(paste("Real part section at ",
                             t[1], "= 0")))
lines(t2, Re(true[i, ]), col = colors[2],
     lty = line_type[2], lwd = line_width)
legend("topleft", legend = legends,
       col = colors, lwd = line_width)


                                        # Another section, now of the
                                        # imaginary part
plot(t1, Im(approx[i, ]), type = "l", col = colors[1],
     lty = line_type[1], lwd = line_width,
     ylab = "",
     xlab = expression(t[1]),
     main = expression(paste("Real part section at ",
                             t[2], "= 0")))
lines(t1, Im(true[i, ]), col = colors[2],
     lty = line_type[2], lwd = line_width)
legend("topleft", legend = legends,
       col = colors, lwd = line_width)
```

In the latter code, we can observe that the output is a list containing three elements: two vectors with the corresponding evaluation grid values on every direction and a complex matrix containing the Fourier integral values. If we do not want to evaluate the Fourier integrals on a regular grid and we desire to evaluate on, say l points, then we must pass a $l \times 2$ matrix in the argument `w` and the function will return a vector of size l rather than a list. The function f must be able to receive a two-column matrix with m rows, where m is the number of points where the Fourier integral will be evaluated.

Figure 3.4 has the two plots generated with the previous code. Recall that the output was a complex matrix, thus we are showing only the a section of the real values (at $t_1 = 0$) and a section of the imaginary values (at $t_2 = 0$).

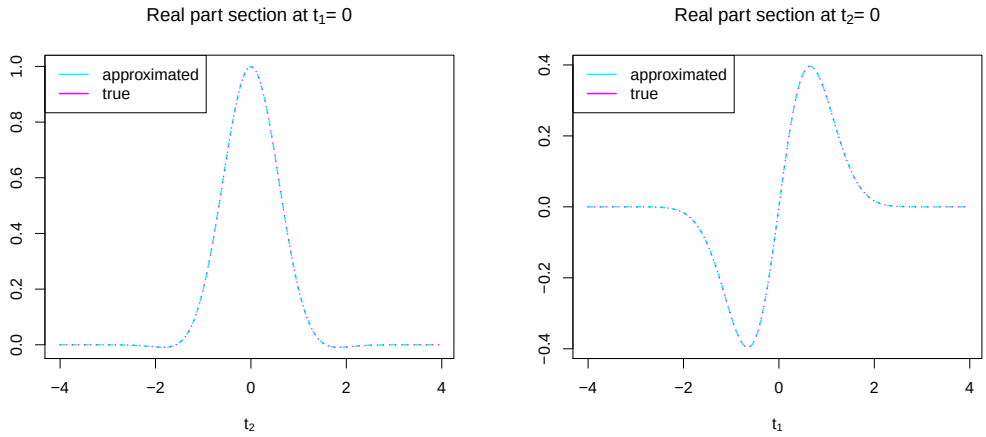

Figure 3.4 Example of `fourierin` function for univariate function at resolution 128×128: Obtaining the characteristic function of a bivariate normal distribution from its density. See Equation 3.4. The left plot is the section of the real part of the characteristic function at $t_1 = 0$; the right plot is the section of the characteristic function at $t_2 = 0$.

3.5 Summary

We have presented univariate and bivariate examples, which are the ones implemented in version 0.0.2. We will extend this numerical procedure up to dimension 5, since this procedure is reliable up to that dimension.

CHAPTER 4. KERNEL DECONVOLUTION DENSITY ESTIMATION WITH R PACKAGE KERDEC

A paper to be submitted to the Journal of Statistical Software

Guillermo Basulto-Elias, Alicia L. Carriquiry, Kris De Brabanter and Daniel J. Nordman

Abstract

Kernel deconvolution density estimation is a nonparametric method to estimate a density of observations contaminated with some error. The **R** package *kerdec* covers several kernel deconvolution sampling scenarios for univariate and bivariate samples. In particular, (i) when the error distribution is known, (ii) when an independent sample of errors is available and (iii) when repeated observations are available. *RcppArmadillo* has been used speed-up heavy computations.

4.1 Introduction

Data with measurement error are common in nature. Ignoring such error produces erroneous inferences. Kernel density estimation, which is perhaps the most common and best studied density estimator cannot be applied directly, since the error must be taken into account.

Several challenges come along with the use of kernel deconvolution density estimator (KDDE), for example bandwidth selection gets a lot more difficult, kernels whose Fourier transform have compact support are better for theoretical and practical reasons.

An extra complication could come from the error distribution, since in real life it is rarely known. There are two alternatives to approximate it. The first is to have, besides the contaminated sample, a sample or errors. The second option arises when more than one contaminated

sample per individual is available, thus, by taking differences of contaminated samples per individuals. Differences of errors are obtained and they can ultimately be used to approximate the error distribution.

There are two **R** packages for deconvolution: *decon* (Wang and Wang, 2011b) and *deamer* (Stirnemann et al., 2012a). The *decon* package performs KDDE with known error allowing the user to pick among several bandwidth selection methods summarized in Delaigle and Gijbels (2004b). The *deamer* package allows different sampling scenarios when the error distribution is unknown with a fix kernel function. The bandwidth selection used there is detailed in Comte and Lacour (2011).

The **R** package that we describe in this work (*kerdec*) performs KDDE under several sampling scenarios and kernel functions for univariate and bivariate data. The bandwidth selection procedure is described in Youndjé and Wells (2008). The heavy computations were are done in **C++** via *RcppArmadillo* for a better time performance. It also uses the **C++** functions included in the **R** *fourierin* package (Basulto-Elias, 2016a).

The major differences of *kerdec* from *decon* and *deamer* is that the *kerdec* also works for bivariate samples, several sampling scenarios and kernel functions, but it is exclusive for KDDE. It also offers parametric and nonparametric approximations to the error distribution.

In Section 4.2, we define the kernel deconvolution formula and possible generalizations and challenges in a more technical fashion. In Section 4.3 we provide details on the possible sampling scenarios. We describe the kernel functions included in the *kerdec* package in Section 4.4 while we show how the bandwidth can be selected and calculated in Section 4.5. In Section 4.6 we show how to use our package with bivariate data. We display specific code examples in Section 4.7. Some functions of *kerdec* can be use in contexts other than kernel deconvolution and they are described in Section 4.8. Finally, the possible features and generalizations that our package will include are described in Section 4.9.

4.2 Kernel Deconvolution Density Estimation

The aim of kernel deconvolution density estimation (KDDE) is to provide a nonparametric density estimator for a distribution whose samples has been contaminated with additive error.

KDDE is a generalization of the usual kernel density estimator where the Fourier inversion theorem is used. The use of this theorem and the fact that the sample is not directly observable impede straightforward generalization. In this section we provide a formula for the estimator and describe some of the challenges produced by this estimator. In Sections 4.3, 4.4 and 4.5 we describe these problems in more detail and show how *kerdec* may be applied.

Let $\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_n$ be a sample of independent and identically distributed (*iid*) d-sized vectors from the model

$$\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{\epsilon}, \tag{4.1}$$

where $\boldsymbol{X}$ and $\boldsymbol{\epsilon}$ are independent. Assume for that $\boldsymbol{\epsilon}$ is a symmetric random vector with mean zero and known variance Σ. The kernel deconvolution density estimation formula is

$$\hat{f}_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{(2\pi)^d} \int \exp\left(-\imath \boldsymbol{x} \cdot \boldsymbol{t}\right) \hat{\phi}_{\boldsymbol{Y},n}(\boldsymbol{t}) \frac{K^{ft}(h\boldsymbol{t})}{\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t})} d\boldsymbol{t}, \qquad \boldsymbol{x} \in \mathbb{R}^d, \tag{4.2}$$

where $\boldsymbol{x} \cdot \boldsymbol{t} = \sum_{i=1}^d x_i t_i$ is the usual inner product in $\mathbb{R}^d$, $K^{ft}(\boldsymbol{t}) = \int \exp\left(\imath \boldsymbol{u} \cdot \boldsymbol{t}\right) K(\boldsymbol{u}) d\boldsymbol{u}$ is the Fourier transform of a kernel K, $\phi_{\boldsymbol{\epsilon}}(\boldsymbol{t}) = \int \exp\left(\imath \boldsymbol{x} \cdot \boldsymbol{t}\right) f_{\boldsymbol{\epsilon}}(\boldsymbol{x}) d\boldsymbol{x}$ is the characteristic function of $\boldsymbol{\epsilon}$ and $\hat{\phi}_{\boldsymbol{Y},n}(\boldsymbol{t}) = \sum_{i=1}^n \exp\left(\imath \boldsymbol{Y}_i \cdot \boldsymbol{t}\right)$ is the empirical characteristic function of the contaminated sample.

The main functions in *kerdec* are `kerdec_dens` and `kerdec_dens2D`. The former for univariate KDDE and the latter for the bivariate case. We will unfold the parameters required in the upcoming sections.

Observe that in formula 4.2 it was assumed that the characteristic function of the error, $\phi_{\boldsymbol{\epsilon}}(\cdot)$ is known. In many real-life cases, it is not possible to have perfect knowledge of the error distribution, thus an approximation to its characteristic function must be used. Moreover it is not clear what the contaminated sample should be in order to use formula 4.2 in case of repeated measurements, since ideally they should be iid. In Section 4.3 we describe how the *kerdec* package handles those situations.

Furthermore, it is convenient to select the Fourier transform of the kernel function K in formula 4.2 having compact support, since this ensures integrability. In Section 4.4 we describe the kernel functions included in *kerdec*.

On the other hand, the integral in deconvolution formula (4.2) needs to be computed numerically, but the integrand is a product of oscillating functions. An effective and fast procedure to do this is using the so-called fast Fourier transform (FFT). *kerdec* uses the **R** package *fourierin*, which computes continuous Fourier integrals using FFT's.

Finally, the integral in deconvolution formula (4.2) cannot be computed with ease, because it cannot be reduced to a simple elementary function.

4.3 Sampling Setting

Formula 4.2 can be applied to contaminated samples when the error distribution is completely known. When the error distribution is not known, extra information is required to approximate its characteristic function.

The simplest sampling scenario for a contaminated sample and full knowledge of the error distribution is detailed in Section 4.3.1. The case when the error distribution is unknown but an extra sample of errors is available is detailed in Section 4.3.2. Finally, if the data contain repeated measurements, also referred as panel data, differences can be used to approximate the error distribution. This is explained in Section 4.3.3. In the two latter cases, the error distribution can be approximated assuming that the errors follow a normal or Laplace distribution. A nonparametric approximation is also provided in *kerdec*.

It is also important to mention that when the data contain repeated measurements, the first repetition can be taken as the contaminated sample but also it would be possible to take the average of the repeated measurements. Moreover, the error distribution may be approximated by taking differences of observations of the same individual; although, there is not a unique way to take these differences. In Section 4.3.3 we describe the possibilities and applications of the *kerdec* package.

For sake of simplicity all the examples considered here are univariate ($d = 1$). In Section 4.6 some bivariate examples will be provided.

4.3.1 Contaminated Sample and Known Error Distribution

Function `kerdec_dens()` computes the KDDE. The simplest scenario in KDDE occurs when the error distribution is known. The normal and Laplace distributions are allowed in this case. Therefore, besides passing the contaminated sample in the parameter `smp`, it is necessary to specify the error distribution in the parameter `error_dist` and the scale parameter of such distribution in `error_scale_par`. The simplest call of this scenario would be:

```
kerdec_dens(smp = my_sample, error_dist = "laplace", error_scale_par = 0.6,
            lower = 0.5, upper = 10)
```

The parameters `lower` and `upper` are the extremes where the density is going to be estimated.

Figure 4.1 is an example of this case. Code to generate this plot and details about the example are shown in Section 4.7.

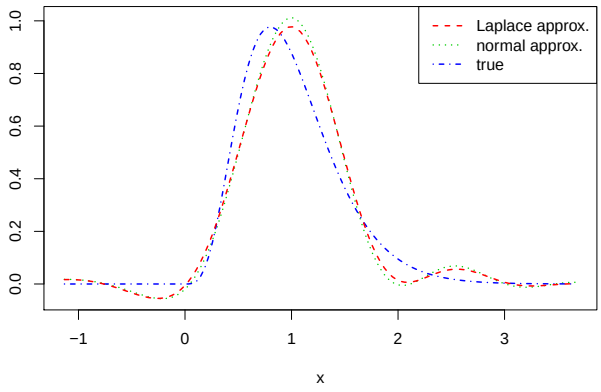

Figure 4.1 Gamma sample contaminated with Laplacian errors. The error distribution is assumed to be known. This plot shows when the correct distribution, Laplace, was given (long dashed red line) and the incorrect distribution (normal) was given (dashed green line).

4.3.2 Contaminated Sample Plus Sample or Errors

In Section 4.3.1 the error distribution was known, thus, the parameters `smp`, `error_dist` and `error_scale_par` were passed to `kerdec_dens()`. In real life it is hard to know beforehand the scale parameter, but it can be estimated if a sample of errors, preferably independent from the contaminated sample, is available. The function `kerdec_dens()` handles this case by providing the sample of errors as the parameter `error_smp` rather than passing the scale parameter directly (`error_scale_par`)). In this case, the maximum likelihood estimate (MLE) of the scale parameter of the error is computed. An example of a call for this scenario is:

```
kerdec_dens(smp = my_sample, error_dist = "normal",
            error_smp = my_error_sample, lower = 0.5, upper = 10)
```

Moreover, observe in Formula 4.2 that the error is used via its characteristic function. The function `kerdec_dens()` can also approximate this characteristic function nonparametrically by using the real part of the empirical characteristic function of the sample of errors (it makes sense to use the real part if we assume that the error distribution is symmetric). Figure 4.2 was generated under this scheme and the code is shown in Section 4.7. The code below is a call of KDDE when the empirical characteristic function is used.

```
kerdec_dens(smp = my_sample, error_dist = "none",
            error_smp = my_error_sample, lower = 0.5, upper = 10)
```

4.3.3 Panel Data

Repeated observations scheme, also referred as panel data, occurs with high frequency in sciences. For example, suppose that we are interested in the distribution of the weight in certain population and a the weight of 200 hundred was obtained each in three different scales.

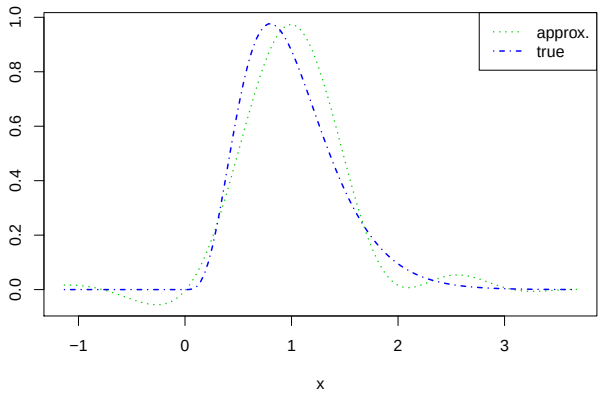

Figure 4.2 Contaminated sample plus a sample of independent errors. The characteristic function of the error has been approximated with the empirical characteristic function. The error distribution was approximated with the empirical characteristic function. The code that generated this plot can be found in Section 4.7.

Therefore our model would be

$$Y_{ij} = X_i + \epsilon_{ij} \qquad i = 1, \ldots, n,\ j = 1, \ldots, l, \tag{4.3}$$

where $X_1, \ldots, X_n$ the corresponding weights of the n individuals, and ϵ_{ij} the measurement errors induced by the scales. Assume that $X_1, \ldots, X_n$ are independent and ϵ_{ij} are independent. The values in the example are $n = 200$ and $j = 3$.

In this sampling scheme, it is still possible to use KDDE, but it is not clear what the sample is in order to apply Formula 4.2. Meister (2009) discusses pros and cons of using only the first column, that is, to use $Y_{1,1}, \ldots, Y_{n,1}$ as the sample or to use the averaged observations instead, that is, $\bar{Y}_1, \ldots, \bar{Y}_n$, where $\bar{Y}_i = n^{-1} \sum_j Y_{ij}$. The parameter `panel_proc` of `kerdec_dens()` can process both by setting it to `'keep_first'` or `'take_aver'`, respectively. `panel = 'keep_first'` is the default value.

On the other hand, the error distribution can be approximated by taking differences of the contaminated observations, since $Y_{i,j_1} - Y_{i,j_2} = \epsilon_{i,j_1} - \epsilon_{i,j_2}$. If there are only two repetitions, the sample of differences would be $Y_{i,2} - Y_{i,1}$, but if there are more than two repetitions, there are more possible differences. In particular, (i) we can consider all pairwise differences $Y_{i,j_1} - Y_{i,j_2}$ for $j_1 > j_2$, which leads to a sample of correlated differences of errors; (ii) we can

take differences by pairs of columns, $Y_{i,2} - Y_{i,1}$, $Y_{i,4} - Y_{i,3}$, ..., $Y_{i,2\lfloor l/2 \rfloor} - Y_{i,2\lfloor l/2 \rfloor - 1}$, which would lead to fewer samples of differences of errors, but independent, and (iii) a compromise between them by subtracting the first repetition from the rest, $Y_{i,l} - Y_{i,1}, Y_{i,l-1} - Y_{i,1}, \ldots, Y_{i,2} - Y_{i,1}$. The parameter `'error_proc'` in `kerdec_dens()` can be set to `'all'`, `'indep_pairs'` and `'vs_first'` to obtain the corresponding case. The default option is `error_proc = 'all'`.

We emphasize the fact that with repeated measurements it is not necessary to specify the scale parameter of the error distribution or the error distribution. Thus, `error_dist` can be set to `'laplace'`, `'normal'` or `'none'`. The latter is the default choice. Again, if a parametric approach is taken, the MLE of the scale parameter is computed.

An example of KDDE for panel data is given below. We have changed the default options.

```
kerdec_dens(smp = my_panel_matrix, error_dist = "laplace",
            panel_proc = "take_aver", error_proc = "indep_pairs",
            lower = 0.5, upper = 10)
```

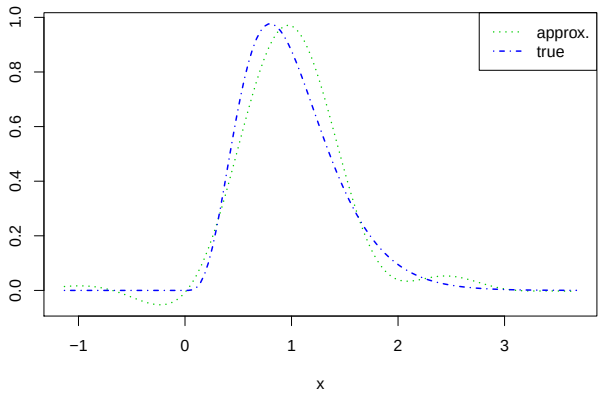

Figure 4.3 KDDE on panel data. Code is shown in Section 4.7

4.4 Kernel Functions

We mentioned in Section 4.2 that it is convenient to restrict the kernel functions to those that have a Fourier transform with bounded support. The most popular kernels in usual kernel density estimation, e.g., Gaussian, Epanechnikov, triweight and tricube, do not meet this criteria. On the other hand, the so-called sinc kernel is the most popular choice due to the simplicity of its Fourier transform. Recall that the Fourier transform of a kernel is what is used in Formula 4.2 and not the kernel itself. Delaigle and Gijbels (2004b) suggest the use of a second order kernel for which the Fourier transform is proportional to the triweight kernel.

The type of kernel can change the form of the estimator. For instance, the sinc kernel has good theoretical properties, but the estimators that it produces are usually very wiggly. *kerdec* allows the user to select among five kernels with compactly bounded Fourier transforms by specifying the parameter `kernel` of function `kerdec_dens()`. The choices are:

- `"sinc"` The sinc kernel. Its Fourier transform is $K^{ft}(t) = I_{[-1,1]}(t)$, where $I_A(\cdot)$ is the indicator function in the set A.
- `"triangular"` Its Fourier transform is a triangle, $K^{ft}(t) = (1 - |t|)I_{[-1,1]}(t)$.
- `"triw"` Its Fourier transform is proportional to the triweight kernel, $K^{ft}(t) = (1 - |t|^2)^3 I_{[-1,1]}(t)$.
- `"tric"` Its Fourier transform is proportional to the tricube kernel, $K^{ft}(t) = (1 - |t|^3)^3 I_{[-1,1]}(t)$.
- `"flat"` Its Fourier transform is a trapeze,

$$K^{ft}(t) = \begin{cases} 1 & |t| < 1/2, \\ 2(1 - |t|) & 1/2 \leq |t| < 1, \cdot \\ 0 & \text{otherwise} \end{cases}$$

  $I_{[-1/2,\,1/2]}(t) + (1 - |t|^3)^3 I_{[-1,1]}(t)$.

Figure 4.4 displays the kernel functions and their Fourier transforms that have been implemented in *kerdec*.

A simple example of usage is the following. The default value is `kernel = "flat"`

```
kerdec_dens(smp = my_sample, error_dist = "none", kernel = "sinc",
            error_smp = my_error_sample, lower = 0.5, upper = 10)
```

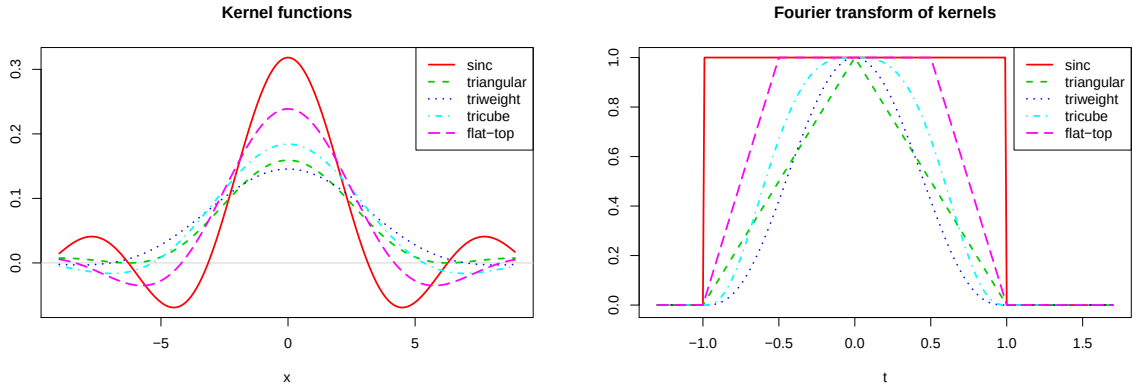

Figure 4.4 Kernel functions (left) and their Fourier transforms (right) included in package *kerdec*.

Finally, we specify that for the bivariate case, *kerdec* provides only so-called product kernels, which are generated by the product of univariate kernels. Figure 4.5 shows the Fourier transform of the flat-top kernel.

4.5 Bandwidth Selection

Bandwidth selection is a crucial part of KDDE. Although in univariate KDDE there are several options of bandwidth selection (cf. Delaigle and Gijbels (2004b)), in the multivariate case there are only two that we are aware of. The first is a cross-validation approach in Youndjé and Wells (2008), and the second is an empirical approximation to the mean integrated squared error in Comte and Lacour (2013). We have implemented the cross-validation based bandwidth selector.

It is known that cross-validation might have more than one local minimum (cf. Delaigle and Gijbels (2004b)). If the user wants to see the actual cross-validation function, it can be

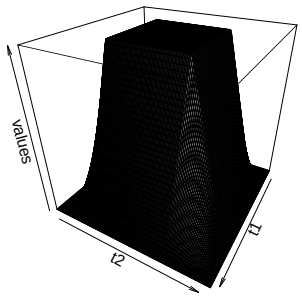

Figure 4.5 Fourier transform of product kernel generated by the kernel whose Fourier transform is a trapeze.

displayed with the function `select_bw()`. This function receives basically the same arguments as `kerdec_dens()`, except for the limits where the density would be estimated. The user can set the value where the optimization function will start to look of a minimum, the parameter is `h0` in `kerdec_dens()`.

The next code shows how `select_bw()` can be used to compute the cross-validation bandwidth. In Section 4.7 we include the piece of code (clearly indicated in the comments) that generated Figure 4.6.

```
select_bw(smp = my_sample, error_smp = my_error_sample,
          error_dist = "normal", bw_interval = c(0.1, 0.4))
```

On the other hand, the user has the option of directly specifying the bandwidth in `kerdec_dens()` by setting the parameter `h` to some value. If `h` is provided, then the cross-validation bandwidth will not be computed.

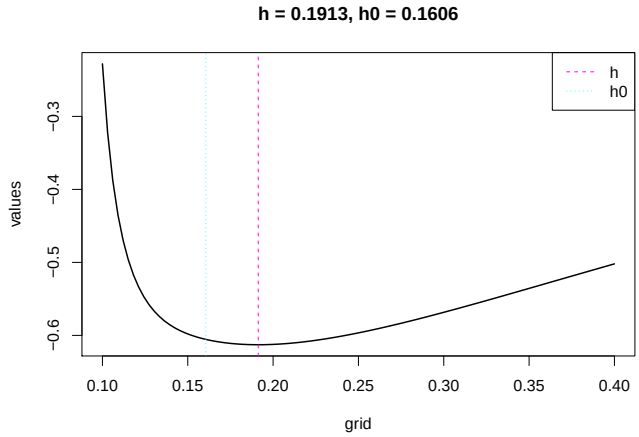

Figure 4.6 Cross-validation function for KDDE. Code and details can be found in Section 4.7.

4.6 Bivariate Case

R package *kerdec* can perform KDDE for bivariate sampling, error distribution options and kernels choices by calling the functions `kerdec_dens2D()` and `select_bw2D()` instead of `kerdec_dens()` and `select_bw()`, respectively. The difference in the arguments is that instead of passing `smp` parameter (as in univariate KDDE), `smp1` and `smp2` must be passed with the first and second coordinates of the observations, respectively. The same is required with parameters `error_smp1`, `error_smp2`, `error_scale_par1` and `error_scale_par2`.

A sample code of bivariate KDDE is shown below.

```
den <- kerdec_dens2D(smp1 = first_coord_of_my_sample,
                     smp2 = second_coord_of_my_sample,
                     error_smp1 = first_coord_of_my_error_sample,
                     error_smp2 = second_coord_of_my_error_sample,
                     lower = c(-1, -2)), upper = c(3, 4))
```

In Section 4.7 we provide a complete example corresponding to the plot displayed in Figure 4.7.

4.7 Examples

In this section we describe several applications of univariate KDDE and bivariate KDDE. The code has enough comments to help the user understand every step.

We first show examples of univariate KDDE under the three sampling schemes described in Section 4.3. For sake of comparison, we have generated samples from the same distributions.

The target density is a gamma distribution with shape and rate parameters equal to 5 and 5, respectively. The sample size has been set to 130. In case of additional error, the sample size of those errors is 140 and for the panel data structure, the number of repetitions is 5.

```
## ------------------------------------------------------------------
## ----- Generate random samples ------------------------------------
## ------------------------------------------------------------------

library(kerdec)                          # Load package
set.seed(666)                            # Set seed

## Settings and samples for all the cases
n <- 130                                 # Sample size
l <- 5                                   # Number of columns
m <- n + 10                              # Error sample size
shape <- 5                               # X distr. shape par.
rate <- 5                                # X distr. rate par.
sd_error <- 0.30                         # std. error of error distr.
X <- rgamma(n, shape, rate)              # Uncontaminated sample
eps_panel <- matrix(rlaplace(n*l, sd = sd_error),
                    n, l)                # Panel of errors
```

```
eps <- rlaplace(m, sd = sd_error)       # Pure errors
Y <- X + eps_panel[, 1]                 # Contaminated sample
Y_panel <- sweep(x = eps_panel, MARGIN = 1,
                 STATS = X, FUN = "+")  # Contaminated in panel

## ------------------------------------------------------------------
## Normal error. Known sigma.
## ------------------------------------------------------------------

normal <- kerdec_dens(smp = Y, error_scale_par = sd_error,
                      error_dist = "normal",
                      lower = min(Y) - 2*sd(Y),
                      upper = max(Y) + 2*sd(Y))
laplace <- kerdec_dens(smp = Y, error_scale_par = sd_error,
                      error_dist = "laplace",
                      lower = min(Y) - 2*sd(Y),
                      upper = max(Y) + 2*sd(Y))

with(normal, {
    true <- dgamma(x_eval, shape, rate)
    plot(range(x_eval), range(c(f_vals, true)), type = "n",
         xlab = "x", ylab = "")
    lines(x_eval, f_vals, col = 3, lty = 3, lwd = 1.5)
    lines(x_eval, true, col = 4, lty = 4, lwd = 1.5)
})
with(laplace, {
    lines(x_eval, f_vals, col = 2, lty = 2, lwd = 1.5)
})
legend("topright", legend = c("Laplace approx.", "normal approx.",
```

```
                              "true"),
       lwd = 1.5, col = 2:4,
       lty = 2:4)

## ------------------------------------------------------------------
## Contaminated sample plus extra sample
## ------------------------------------------------------------------

dens <- kerdec_dens(smp = Y, error_smp = eps,
                    error_dist = "none",
                    lower = min(Y) - 2*sd(Y),
                    upper = max(Y) + 2*sd(Y))

with(dens, {
    true <- dgamma(x_eval, shape, rate)
    plot(range(x_eval), range(c(f_vals, true)), type = "n",
         xlab = "x", ylab = "")
    lines(x_eval, f_vals, col = 3, lty = 3, lwd = 1.5)
    lines(x_eval, true, col = 4, lty = 4, lwd = 1.5)
})
legend("topright", legend = c("approx.", "true"),
       lwd = 1.5, col = 3:4,
       lty = 3:4)

## ------------------------------------------------------------------
## Panel data
## ------------------------------------------------------------------

dens <- kerdec_dens(smp = Y_panel, error_smp = eps_panel,
```

```
                    error_dist = "laplace",
                    lower = min(Y) - 2*sd(Y),
                    upper = max(Y) + 2*sd(Y))

with(dens, {
    true <- dgamma(x_eval, shape, rate)
    plot(range(x_eval), range(c(f_vals, true)), type = "n",
         xlab = "x", ylab = "")
    lines(x_eval, f_vals, col = 3, lty = 3, lwd = 1.5)
    lines(x_eval, true, col = 4, lty = 4, lwd = 1.5)
})
legend("topright", legend = c("approx.", "true"),
       lwd = 1.5, col = 3:4,
       lty = 3:4)
```

In Section 4.5 we mentioned that the function `select_bw()` can find the bandwidth outside of the function `kerdec_dens()` at the same time it plots the cross-validation function to be minimized. Figure 4.6 corresponds to the code below.

```
## ------------------------------------------------------------------
## Contaminated sample plus extra sample. Bandwidth selection
## ------------------------------------------------------------------

library(kerdec)                         # Load package
set.seed(666)

## Settings and samples for all the cases
n <- 150                                # Sample size
```

```
m <- n + 10                              # Error sample size
shape <- 5                               # X distr. shape par.
rate <- 5                                # X distr. rate par.
sd_error <- 0.3                          # std. error of error distr.
X <- rgamma(n, shape, rate)              # Uncontaminated sample
eps <- rlaplace(m, sd = sd_error)        # Error sample
Y <- X + rlaplace(n, sd = sd_error)      # Contaminated sample


select_bw(smp = Y, error_smp = eps,
          error_dist = "normal", bw_interval = c(0.1, 0.4))
```

Finally, the code below illustrates the use of the bivariate KDDE in *kerdec*. The target distribution is a product of gamma densities and the error follows a bivariate Laplace distribution. Observe that the bandwidth has been directly provided.

```
## ------------------------------------------------------------------
## Bivariate case
## ------------------------------------------------------------------

library(kerdec)                          # Load package
set.seed(666)

## Settings and samples for all the cases
n <- 130                                 # Sample size
l <- 5                                   # Number of columns
m <- n + 10                              # Error sample size
shape <- 5                               # X distr. shape par.
```

```
rate <- 5                                # X distr. rate par.
sd_error1 <- 0.2                         # std. error of error distr.
sd_error2 <- 0.3                         # std. error of error distr.

## Generate biv. samples.
X1 <- rgamma(n, shape, rate)             # Uncontaminated sample
X2 <- rgamma(n, shape, rate)             # Uncontaminated sample
eps1 <- rlaplace(m, sd = sd_error1)      # Pure errors
eps2 <- rlaplace(m, sd = sd_error2)      # Pure errors
Y1 <- X1 + rlaplace(n, sd = sd_error1)   # Contaminated sample
Y2 <- X2 + rlaplace(n, sd = sd_error2)   # Contaminated sample


bw <- (0.12)^2
den <- kerdec_dens2D(smp1 = Y1, smp2 = Y2,
                     kernel = "vp",
                     error_smp1 = eps1,
                     error_smp2 = eps2,
                     error_dist = "none",
                     lower = c(min(Y1) - 4*sd(Y1), min(Y2) - 4*sd(Y2)),
                     upper = c(max(Y1) + 4*sd(Y1), max(Y2) + 4*sd(Y2)),
                     h = bw, resolution = 256)
persp(z = Re(den$vals), theta = 30)
```

4.8 Other Features

There are some functions included in *kerdec* that are required by KDDE, but also could be useful out of this context. We briefly present multivariate empirical characteristic functions in Section 4.8.1 and in Section 4.8.2 we present our Laplace distribution functions.

4.8.1 Empirical Characteristic Function

The empirical characteristic function (ECF) it is easy to program in **R**. It is opportune to have it already implemented. We have implemented in *kerdec* a few useful functions that were programmed in **C++** and called via *RcppArmadillo*.

The functions available are `ecf()`, `ecf_real()`, `ecf_imag()` and `ecf_mod()` to compute the ECF, its real and imaginary parts and its module, respectively. The importance of having four different functions is that it avoids unnecessary computations. For instance, calling `ecf_real()` rather than `Re(ecf())` takes about half the time, and calling `ecf_mod()` rather than `Mod(ecf())` avoids the creation of a complex matrix or vector and it is slightly faster. The code below provides a simple example of usage of these functions in both, univariate and multivariate cases.

```
## ------------------------------------------------------------------
## Empirical Characteristic Function
## ------------------------------------------------------------------

## Univariate ------------------------------------------------------

library(kerdec)                          # Load package
set.seed(666)                            # Set seed

## Parameters of Poisson distribution, sample size and grid size
lambda <- 10
n <- 150                                 # Sample size
m <- 200                                 # Grid size
t <- seq(-2, 2, length.out = m)          # Evaluation grid
smp <- rpois(n, lambda)                  # Random sample
```

```
## Compute empirical characteristic values and characteristic function
## values
real <- ecf_real(t, smp)
real[1:5]



> real[1:5]
[1] -0.01652991 -0.01911710 -0.02250165 -0.02628080 -0.02998878



## Multivariate ------------------------------------------------------

## Parameters of bivariate normal distribution
mu <- c(-1, 0, 1)
sig <- diag(1:3)

## Random sample of dimension 3.
rndm <- function(n) {
    cbind(rnorm(n, mu[1], sig[1, 1]),
          rnorm(n, mu[2], sig[2, 2]),
          rnorm(n, mu[3], sig[3, 3]))
}

## Create evaluation grid.
grid_1d <- seq(-3, 3, length.out = 10)
grid <- as.matrix(expand.grid(t1 = grid_1d,
                              t2 = grid_1d,
                              t3 = grid_1d))
```

```
n <- 50
ecf(t = grid, smp = rndm(n))[1:5]
```

```
> ecf(t = grid, smp = rndm(n))[1:5]
[1] -0.05367969-0.08665816i -0.13508830-0.07451102i -0.15411251+0.04830170i
[4] -0.15340938+0.13791242i -0.05158918+0.26974275i
```

4.8.2 Laplace Distribution

The Laplace distribution is frequently used to model error. The *kerdec* package provides the functions `rlaplace()`, `plaplace()`, and `dlaplace()` for generating random samples, computing the cumulative distribution function and the probability density function. These functions work the same way as **R** base distribution functions.

There are several parametrizations of the Laplace distribution. We have selected the one parametrized by its mean and standard deviation, that is, the one whose probability density function is given by

$$f(x) = \frac{\sqrt{2}}{\sigma} \exp\left(-\sqrt{2}\frac{x-\mu}{\sigma}\right), \qquad x \in \mathbb{R}. \tag{4.4}$$

We conclude with a brief example of use of these functions.

```
## ------------------------------------------------------------------
## Laplace Distribution
## ------------------------------------------------------------------
set.seed(666)
sigma <- 3
n <- 10
```

```
dlaplace(x = c(0, 10), mean = 0, sd = sigma)
```

```
> dlaplace(x = c(0, 10), mean = 0, sd = sigma)
[1] 0.235702260 0.002113878
```

```
plaplace(x = c(0, 10), mean = 0, sd = sigma)
```

```
> plaplace(x = c(0, 10), mean = 0, sd = sigma)
[1] 0.5000000 0.9955158
```

```
rlaplace(n = n, mean = 0, sd = sigma)
```

```
> rlaplace(n = n, mean = 0, sd = sigma)
 [1] -5.07708752  3.48028205  0.23964812 -1.28030426 -0.44145466  0.79536979
 [7]  5.06361085  3.00571128  3.81773484  0.02808505
```

4.9 Conclusions and Future Developments

We have presented here the more important functions of *kerdec* package for KDDE. The functions are intended to be simple to use and they have been improved in their speed by doing all the time consuming operations in **C++**. Future versions of the package will include more bandwidth selection methods and perhaps it will also contain functions for local regression with measurement error.

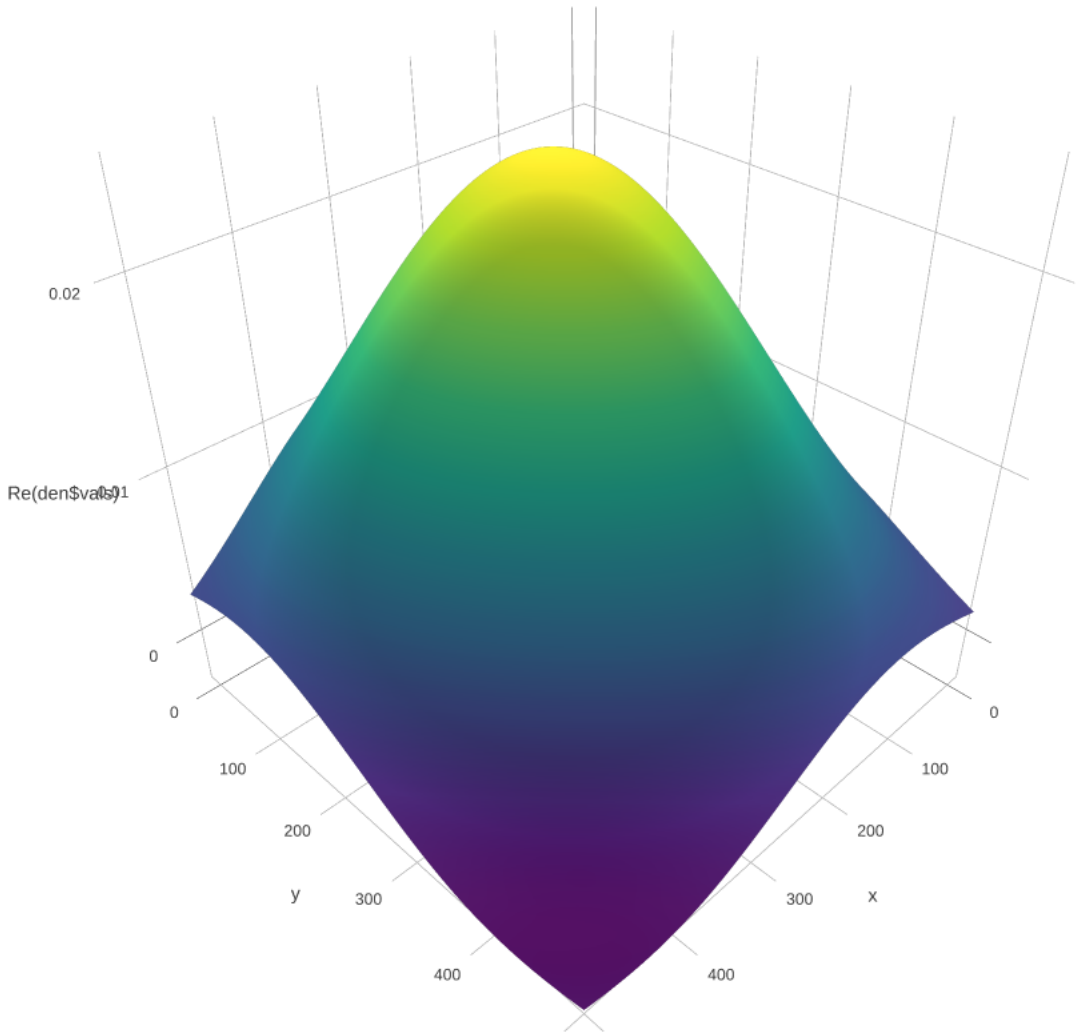

Figure 4.7 Bandwidth for example in Section 4.3.2

CHAPTER 5. ADAPTIVE BANDWIDTH FOR KERNEL DECONVOLUTION DENSITY ESTIMATORS

A paper in preparation

Guillermo Basulto-Elias, Alicia L. Carriquiry, Kris De Brabanter and Daniel J. Nordman

Abstract

Kernel deconvolution density estimation is a nonparametric estimator of a density based on observations that are subject to random noise. We provide a bandwidth that adapts to the smoothness of the target density

5.1 Introduction

Kernel deconvolution density estimation (KDDE) is a nonparametric density estimator for samples subjected to measurement error. It is the natural generalization of the usual kernel density estimation (KDE) to this setting.

In KDDE, samples of the target distribution are only available with measurement error. The basic idea of KDDE is to use the Fourier inversion theorem, thus, the theory is developed almost exclusively in the frequency domain.

In nonparametric problems, the selection of the tuning or bandwidth parameter is one of the toughest steps. KDDE is not the exception; moreover, well-studied bandwidth selection methods must be done in the frequency domain in order to be adapted to KDDE, if this is even possible.

Besides the contaminated sample, some knowledge on the error distribution is required, otherwise the problem is ill-posed. To study properties of the KDDE, it is common to assume

that the error distribution is known. It is possible, however, to approximate the error distribution if there are repeated measurements (Comte et al. (2014)) or if an extra sample of errors is available (cf. Johannes (2009)).

A comparison between several bandwidth selection methods in KDDE can be found in Delaigle and Gijbels (2004b,a). Bootstrap, cross-validation and normal reference are among these methods. Another approach is considered by Comte and Lacour (2011), where an empirical version of the asymptotic mean squared error (MISE) is minimized.

The bootstrap method and the plug-in bandwidth selection methods presented in Delaigle and Gijbels (2004b) require a pilot bandwidth. To provide it, the roughness of the second derivative (that it, the integral of its square) is approximated by an iterative method that starts with a normal reference rule, as described in Delaigle and Gijbels (2002).

In KDDE, the rate at which the tails of the characteristic function (ECF) of the error decay affects the overall convergence rates (of the MISE, for instance). Thus, it is required to study the estimators separately according to this. Fan (1991b) provides a pilot bandwidth that leads to optimal convergence rates, however, it requires knowledge of the smoothness of the target density or the rate at which the CF of the error goes to zero.

On the other hand, Politis (2003b) describes a pilot bandwidth in the context of the spectral density in time series and KDE that automatically captures the optimal convergence rates with no knowledge of the target density. This pilot bandwidth is obtained by studying the estimators in the frequency domain. Moreover, it is not obtained by an iterative process, but by "cutting" the support of the empirical characteristic function (ECF).

In this paper, we generalize the the pilot bandwidth proposed in Politis (2003b) to KDDE using flat-top kernels, which have been proven to reduce bias (cf. Politis and Romano (1999)).

5.2 Background

Measurement error problems occur often in sciences. Suppose that we want a nonparametric density estimator, the usual kernel density estimator (KDE) would not be appropriate since it would not take the measurement error into account, specially when such error is large. The kernel deconvolution density estimator (KDDE) provides a solution to the latter problem.

Consider a sample from the model

$$Y = X + \epsilon, \tag{5.1}$$

where X and ϵ are independent and ϵ has mean equal to zero, say $Y_1, \ldots, Y_n$. The KDDE of X is defined as

$$\hat{f}_X(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(-\imath t x) \hat{\phi}_{Y,n}(t) \frac{K^{ft}(ht)}{\phi_\epsilon(t)} dt, \tag{5.2}$$

where h is the bandwidth parameter; $\hat{\phi}_{Y,n}(t) = n^{-1} \sum_{i=1}^{n} \exp(\imath t Y_i)$ is the empirical characteristic function (ECF) of the contaminated sample;

$$K^{ft}(s) = \int_{-\infty}^{\infty} \exp(\imath s x) K(x) dx \qquad\qquad s \in \mathbb{R}$$

is the Fourier transform (FT) of the kernel function K (the function K is symmetric and integrates to one), and

$$\phi_\epsilon(t) = \int_{-\infty}^{\infty} \exp(\imath s x) f_\epsilon(x) dx, \qquad\qquad t \in \mathbb{R},$$

where $f_\epsilon(\cdot)$ is the probability density function (PDF) of the error, which we will assume is known.

Note in Equation 5.2 that the kernel itself is not used directly, but rather, its FT. If $K^{ft}(\cdot)$ had compact support, the integral in Equation 5.2 would be easier to compute and we would be certain it is finite. Moreover, the properties of the KDDE are easier to study.

The magnitude of the pointwise bias of the KDDE depends on both, the *order* of the kernel K and the smoothness of the target density $f_X(\cdot)$. We say that K is of order q if there exists $|\alpha| < \infty$ such that

$$\int_{-\infty}^{\infty} x^k K(x) dx = \begin{cases} 0, & k = 1, \ldots q-1, \\ \alpha & k = q. \end{cases} \tag{5.3}$$

The biases of the KDEs and the KDDEs are exactly the same (cf. Meister (2009)). In the context of KDEs, Politis and Romano (1999) define a family of kernels whose Fourier transform is flat around the origin and demonstrate that such kernels produce KDEs with bias depending

only on the smoothness of the target density. This family of kernels is defined via its CF, specifically,

$$K^{ft}(t) = \begin{cases} 1, & |t| < \lambda, \\ g_\lambda(t), & \lambda \leq |t| < 1, \\ 0, & |t| \geq 1, \end{cases} \tag{5.4}$$

where $0 < \lambda \leq 1$ and $g_c(\cdot)$ is a continuous real valued function such that for all $t \in \mathbb{R}$, $g_\lambda(t) = g_\lambda(-t)$, $|g_\lambda(t)| \leq 1$, $g_\lambda(\lambda) = 1$, $g_\lambda(1) = 0$. We will refer to any of these kernels as *flat-top kernels*. The case $\lambda = 1$ is the most popular kernel in KDDE. Note that in this case no function $g_\lambda(\cdot)$ is required. Another example of a kernel that meets these conditions is trapezoidal shaped:

$$K^{ft}(t) = \begin{cases} 1, & |t| < 1/2, \\ 2(1 - |t|), & 1/2 \leq |t| < 1, \\ 0, & |t| \geq 1. \end{cases}$$

In order to study the convergence of KDEs, Meister (2009) considers a sub-class of densities in $L_2(\mathbb{R})$ that ensures that the first $\lfloor \beta \rfloor$ derivatives of the density exist and are bounded.

Definition 5.2.1. *We say that a density satisfies a Sobolev's condition if it belongs to the set*

$$\mathcal{F}_{\beta,C} = \{f \text{ density} : \int |f^{ft}(t)|^2 |t|^{2\beta} dt \leq C\}. \tag{5.5}$$

where $0 < C < \infty$. β *is called the smoothness degree.*

On the other hand, observe that the CF of the error appears in the denominator of Equation 5.2. We may infer that its behavior affects the convergence rates. Indeed, we will discuss compare such convergence rates. Before that, let us define two main classes of tail behavior of CF of the error.

Definition 5.2.2. *A density function* f *is said to be* ordinary smooth (OS) *if there exist* $\alpha > 0$ *and* $0 < C_1 < C_2$ *such that*

$$C_1(1 + |t|)^{-\alpha} \leq |f^{ft}(t)| \leq C_2(1 + |t|)^{-\alpha}, \qquad \text{for all } t \in \mathbb{R}. \tag{5.6}$$

Definition 5.2.3. *A density function f is said to be* supersmooth (SS) *if there exist $\gamma > 0$, $d_2 > d_1 > 0$ and $0 < C_1 < C_2$ such that*

$$C_1 \exp(-d_1|t|) \leq |f^{ft}(t)| \leq C_2 \exp(-d_2|t|), \qquad \text{for all } t \in \mathbb{R}. \tag{5.7}$$

Laplace and gamma distributions are examples of OS distributions, while Gaussian and Cauchy distributions are examples of SS distributions. Gaussian and Laplace distributions are common for modeling measurement errors. There are other special measurement errors that may not be SS or OS; the uniform distribution is an example of this. See Meister and Neumann (2010) for KDDE where the error follows a uniform distribution.

Observe that all the flat-top kernels given by 5.4 satisfy that $K \in L_2(\mathbb{R})$; $K^{(ft)}(\cdot)$ has support on $[-1, 1]$; $|K^{ft}(t)| \leq 1$ for all $t \in \mathbb{R}$, and

$$\sup_{t \neq 0} |t|^{-\beta} |K^{ft}(t) - 1| < \infty,$$

for all $\beta > 1$. Therefore, according to Meister (2009),

$$\sup_{f \in \mathcal{F}_{\beta,C}} MISE\left(\hat{f}, f\right) = O\left(\frac{1}{nh} \max_{|t| \leq 1/h} |\phi_\epsilon(t)|^{-2}\right) + O(h^{2\beta}), \tag{5.8}$$

where

$$MISE\left(\hat{f}, f\right) = E \int \left[\hat{f}(x) - f(x)\right]^2 dx$$

is the mean integrated squared error (MISE).

The first term of Equation 5.8 comes from the variance and the second term from the squared bias. We may identify that the bias term is the same order as in KDE, while the variance differs in the term $\max_{|t| \leq 1/h} |\phi_\epsilon(t)|^{-2}$ (in KDE the order of the variance is $O(n^{-1}h^{-1})$.

Now, in order to reduce the order of convergence of the MISE, the order of the bandwidth ought to be such that the variance and bias terms are the same. It is possible to see (cf. Meister (2009)) that if $f \in \mathcal{F}_{\beta,C}$, then $h \asymp n^{1/(2\beta+2\alpha+1)}$ for the OS case (5.6) and $h = c^{-1/\gamma} (\log n)^{-1/\gamma}$ for some $c \in (0, (2d_1)^{-1})$. The overall bounds are $O(n^{-2\beta/(2\beta+2\gamma+1)})$ for OS error and $O\left((\log n)^{-2\beta/\gamma}\right)$ for SS error, that is, the order of convergence is much slower for SS error.

5.3 Adaptive Bandwidth Using Flat-Top Kernels

Fan (1991b) introduced pilot bandwidth for KDDE in both cases, OS and SS error. It is necessary, however, to know beforehand the number of derivatives in the OS error case. Delaigle and Gijbels (2002) proposed a pilot bandwidth for KDDE which is based on an iterative procedure, which requires minimization of an asymptotic MISE at every iteration. On the other hand, Politis (2003b) suggests another pilot bandwidth

Politis (2003b) proposed a pilot bandwidth in the context KDEs. Such bandwidth is very simple and fast to find by truncating the ECF of the sample. We discuss in this section how this idea also works for KDDEs.

We start by defining the pilot bandwidth defined by Politis (2003b) for KDDEs. This is,

$$\hat{h} = (2\hat{t})^{-1}, \qquad \text{where } \hat{t} = \min\left\{ t > 0 : \ |\hat{\phi}_{Y,n}(t)| = \kappa \sqrt{\frac{\log n}{n}} \right\}, \tag{5.9}$$

where κ is a positive constant. Politis (2003b) suggests practical ways to pick κ. Observe that here we are defining the bandwidth for the ECF of the contaminated sample.

Proposition 5.3.1. *Let $\phi_Y(\cdot)$ be the CF of Y in (5.1).*

(i) Suppose that

$$|\phi_Y(t)| = a|t|^{-b} + o(|t|^{-b}), \qquad \text{as } |t| \to \infty,$$

for some $a > 0$ and $b > 1$ then $\hat{h} \sim^P A\, n^{1/2b} (\log n)^{1/2b}$, for some constant A. $U_n \sim^P V_n$ means $U_n/V_n \to 1$ in probability.

(ii) Suppose that

$$|\phi_Y(t)| = a|t|^{b} e^{-d|t|^e} + o(|t|^{b} e^{-d|t|^e}), \qquad \text{as } |t| \to \infty,$$

for some constant b, positive constants a and d and $e \geq 1$, then $\hat{h} \sim^P A\, (\log n)^{-1/e}$, for $A = (1/2d)^{-1/e}$.

Although the statement of Proposition 5.3.1 is slightly different from Theorem 2.2 from Politis (2003b), the proof is essentially the same.

5.4 Discussion

In Section 5.3 we proposed an adaptive bandwidth using flat-top kernels inspired by Politis (2003b). In such paper, the adaptive bandwidth is further used to find a local bandwidth selection method for KDE's with second order kernels, although it is specified that the same idea would be applied to global density estimation. In this section we do the analogous global bandwidth selection for second order kernels KDDE's using flat-top pilots (5.4).

The asymptotic mean integrated squared error (AMISE) is the following

$$AMISE(h) = \frac{1}{2\pi h}\int |K^{ft}(t)|^2 |\phi_\epsilon(t/h)|^{-2} dt + \frac{h^4}{4}\mu_2^2(K) R(f_X''), \tag{5.10}$$

where K is a second order kernel (5.3), $\mu_2(K) = \left(\int x^2 K(x)dx\right)^2$ and $R(\psi) = \int \psi^2(x)d(x)$. We abuse the notation by making the AMISE depend only on the bandwidth parameter, when the more formal and cumbersome notation would be $AMISE\left[\hat{f}_X, f_X\right]$.

Now, in Equation 5.8, we can approximate $R(f_X'')$ using the pilot bandwidth $\hat{h}$ in Equation 5.9 in the following manner.

$$\widehat{R(f_X'')} = \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\hat{f}_X''(x)\right]^2 dx = \int_{-\infty}^{\infty} t^2 |\hat{\phi}_{Y,n}(t)|^2 \frac{|K_{flat}^{ft}(\hat{h}t)|^2}{|\phi_\epsilon(t)|^2} dt \tag{5.11}$$

where K_{flat} is a flat-top kernel, we used Parseval's identity and

$$\hat{f}_X''(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(-\imath t x) t^2 \hat{\phi}_{Y,n}(t) \frac{K_{flat}^{ft}(\hat{h}t)}{\phi_\epsilon(t)} dt. \tag{5.12}$$

Meister (2009) proves that the optimal bandwidth is $n^{-2\beta/(2\beta+2\alpha+1)}$, which is quite different from $n^{-1/(2\beta+2\alpha+1)}$. In the SS, however, the optimal bandwidth is the same order as before, thus $\hat{h}$ has the same asymptotic behavior. Therefore, we can expect this procedure to work fine in the case where the error is OS.

Finally, $\widehat{R(f_X'')}$ can approximate $R(f_X'')$ in Equation 5.10 and thus it is possible to provide a bandwidth parameter by minimizing this approximation to the AMISE.

CHAPTER 6. CONCLUSIONS

We introduced the interesting topic of kernel deconvolution density estimation and some of its challenges in Chapter 1. In Chapter 2 we explored some methodological aspects of KDDE. Then in Chapter 3 and Chapter 4 we described our statistical software for KDDE in R, which is both fast and flexible. In Chapter 5, we proposed a simple bandwidth selection procedure that has good theoretical properties.

There are many directions in which this work could be extended. To start with, local constant regression is a nonparametric regression method generalized from kernel density estimation. There is a measurement error version of this procedure that could be implemented in our software, as well as other already existent methods.

It would be desirable to extend the R-package to perform Fourier-type integrals for higher dimensions. The current version is univariate and bivariate, which are the dimensions that the C++ used to build our software currently handles. This could be done by using another C++ library integrates well with R and performs fast Fourier transforms in higher dimensions; we currently use the C++ Armadillo library for univariate and bivariate cases.

Flat-top kernels showed promising properties in Chapter 2 and Chapter 5. More comprehensive theoretical and practical studies of this family of kernels could improve the kernel deconvolution density estimation with better bandwidth selection procedures.

BIBLIOGRAPHY

Bailey, D. H. and Swarztrauber, P. N. (1994). A fast method for the numerical evaluation of continuous fourier and laplace transforms. *SIAM Journal on Scientific Computing*, 15(5):1105–1110.

Basulto-Elias, G. (2016a). fourierin: Computes Numeric Fourier Integrals.

Basulto-Elias, G. (2016b). kerdec: An `R` package for deconvolution methods. *Available at https://github.com/gbasulto/kerdec*.

Basulto-Elias, G., Carriquiry, A., Brabanter, K. D., and Nordman, D. J. (2016). "fourierin": An `R` package to compute fourier integrals. Unpublished manuscript.

Comte, F. and Lacour, C. (2010). Pointwise deconvolution with unknown error distribution. *Comptes Rendus Mathematique*, 348(5):323–326.

Comte, F. and Lacour, C. (2011). Data-driven density estimation in the presence of additive noise with unknown distribution. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(4):601–627.

Comte, F. and Lacour, C. (2013). Anisotropic adaptive kernel deconvolution. 49(2):569–609.

Comte, F., Samson, A., and Stirnemann, J. J. (2014). Deconvolution Estimation of Onset of Pregnancy with Replicate Observations. *Scandinavian Journal of Statistics*, 41(2):325–345.

Delaigle, A. and Gijbels, I. (2002). Estimation of integrated squared density derivatives from a contaminated sample. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):869–886.

Delaigle, A. and Gijbels, I. (2004a). Bootstrap bandwidth selection in kernel density estimation from a contaminated sample. *Annals of the Institute of Statistical Mathematics*, 56(1):19–47.

Delaigle, A. and Gijbels, I. (2004b). Practical bandwidth selection in deconvolution kernel density estimation. *Computational Statistics & Data Analysis*, 45(2):249–267.

Delaigle, A. and Meister, A. (2008). Density estimation with heteroscedastic error. *Bernoulli*, 14(2):562–579.

Devroye, L. (1989). Consistent deconvolution in density estimation. *Canadian Journal of Statistics*, 17(2):235–239.

Duong, T. et al. (2007). ks: Kernel density estimation and kernel discriminant analysis for multivariate data in r. *Journal of Statistical Software*, 21(7):1–16.

Eddelbuettel, D. and Sanderson, C. (2014). RcppArmadillo: Accelerating R with high-performance C++linear algebra. *Computational Statistics & Data Analysis*, 71:1054–1063.

Fan, J. (1991a). On the optimal rates of convergence for nonparametric deconvolution problems. *The Annals of Statistics*, pages 1257–1272.

Fan, J. (1991b). On the Optimal Rates of Convergence for Nonparametric Deconvolution Problems. *Source: The Annals of Statistics The Annals of Statistics*, 19(3):1257–1272.

Hahn, T. (2005). Cuba-a library for multidimensional numerical integration. *Computer Physics Communications*, 168(2):78–95.

Inverarity, G. (2002a). Fast computation of multidimensional fourier integrals. *SIAM Journal on Scientific Computing*, 24(2):645–651.

Inverarity, G. W. (2002b). Fast Computation of Multidimensional Fourier Integrals. *SIAM Journal on Scientific Computing*, 24(2):645–651.

Johannes, J. (2009). Deconvolution with unknown error distribution. *The Annals of Statistics*, 37(5A):2301–2323.

Masry, E. (1991). Multivariate probability density deconvolution for stationary random processes. *Information Theory, IEEE Transactions on*, 37(4):1105–1115.

Masry, E. (1993a). Asymptotic normality for deconvolution estimators of multivariate densities of stationary processes. *Journal of multivariate analysis*, 44(1):47–68.

Masry, E. (1993b). Strong consistency and rates for deconvolution of multivariate densities of stationary processes. *Stochastic processes and their applications*, 47(1):53–74.

Meister, A. (2009). *Deconvolution Problems in Nonparametric Statistics*, volume 193 of *Lecture Notes in Statistics*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Meister, A. and Neumann, M. H. (2010). Deconvolution from non-standerd error densities under replicated measurements. *Statistica Sinica*, 20:0–0.

Neumann, M. H. and Hössjer, O. (1997). On the effect of estimating the error density in nonparametric deconvolution. *Journal of Nonparametric Statistics*, 7(4):307–330.

Politis, D. N. (2003a). Adaptive bandwidth choice. *Journal of Nonparametric Statistics*, 15(4-5):517–533.

Politis, D. N. (2003b). Adaptive bandwidth choice. *Journal of Nonparametric Statistics*, 15(4-5):517–533.

Politis, D. N. and Romano, J. P. (1995). Bias-corrected nonparametric spectral estimation. *Journal of time series analysis*, 16(1):67–103.

Politis, D. N. and Romano, J. P. (1999). Multivariate Density Estimation with General Flat-Top Kernels of Infinite Order. *Journal of Multivariate Analysis*, 68(1):1–25.

Stirnemann, J., Samson, A., and Lacour, F. C. C. f. C. (2012a). deamer: Deconvolution density estimation with adaptive methods for a variable prone to measurement error.

Stirnemann, J. J., Comte, F., and Samson, A. (2012b). Density estimation of a biomedical variable subject to measurement error using an auxiliary set of replicate observations. *Statistics in medicine*, 31(30):4154–4163.

Wand, M. P. and Jones, M. C. (1995). *Kernel smoothing*, volume 60. Chapman & Hall/CRC.

Wang, X.-F. and Wang, B. (2011a). Deconvolution estimation in measurement error models: The R package decon. *Journal of Statistical Software*, 39(10):1–24.

Wang, X.-F. and Wang, B. (2011b). Deconvolution Estimation in Measurement Error Models: The R Package decon. *Journal of Statistical Software*, 39(10):1–24.

Youndjé, . and Wells, M. T. (2008). Optimal bandwidth selection for multivariate kernel deconvolution density estimation. *Test*, 17:138–162.