BANDWIDTH SELECTION FOR LEVEL SET ESTIMATION IN THE CONTEXT OF REGRESSION AND A SIMULATION STUDY FOR NON PARAMETRIC LEVEL SET ESTIMATION WHEN THE DENSITY IS LOG-CONCAVE

GABRIELA GONZALEZ MARTINEZ

A DISSERTATION SUBMITTED TO
THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN MATHEMATICS AND STATISTICS
YORK UNIVERSITY
TORONTO, ONTARIO

January 2022

Abstract

Bandwidth selection is critical for kernel estimation because it controls the amount of smoothing for a function's estimator. Traditional methods for bandwidth selection involve optimizing a global loss function (e.g. least squares cross validation, asymptotic mean integrated squared error). Nevertheless, a global loss function becomes suboptimal for the level set estimation problem which is local in nature. For a function g, the level set is the set $LS_\lambda = \{x : g(x) \geq \lambda\}$.

In the first part of this thesis we study optimal bandwidth selection for the Nadaraya-Watson kernel estimator in one dimension. We present a local loss function as an alternative to L_2 metric and derive an asymptotic approximation of its corresponding risk. The level set optimal bandwidth (h_{opt}) is the argument that minimizes the asymptotic approximation. We show that the rate of h_{opt} coincides with the rate from traditional global bandwidth selectors. We then derive an algorithm to obtain the practical bandwidth and study its performance through simulations. Our simulation results show that in general, for small samples and small levels, the level set optimal bandwidth shows improvement in estimating the level set when compared to the cross validation bandwidth selection or the local polynomial kernel estimator. We illustrate this new bandwidth selector on a decompression sickness study on the effects of duration and pressure on mortality during a dive.

In the second part, motivated by our simulation findings and the relationship of the level set estimation to the highest density region (HDR) problem, we study via simulations the properties of a plug-in estimator where the density is estimated with a log-concave mixed model. We focus in particular on univariate densities and compare this method against a kernel plug-in estimator. The bandwidth for the kernel plug-in estimator is chosen optimally for the HDR problem. We observe through simulations that when the number of components in the model is correctly specified, the log-concave plug-in estimator performs better than the kernel estimator for lower levels and similarly for the rest of the levels considered. We conclude with an analysis on the daily maximum temperatures in Melbourne, Australia.

Acknowledgements

This thesis would not have been possible without the support of my supervisor Dr. Hanna K. Jankowski who introduced me to this topic. I am thankful for her insightful suggestions and invaluable feedback throughout this project, as well as her guidance during my graduate studies. I would also like to thank my committee members Dr. Xin Gao and Dr. Xiaogang Wang for their helpful suggestions.

I was very fortunate to form part of the Statistical Consulting Service group at the University. I am thankful to all the consultants at the SCS group for being my mentors and always give me their advice. In particular, I will always be indebted to Dr. Georges Monette for all his generosity both with his time and his knowledge. I thank him for all the stimulating conversations and for transmitting me his passion for Statistics.

Finally, I would like to thank my parents for their love and support throughout the years. I also thank my partner for his encouragement and understanding. This thesis is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A problem related to regression is that of estimating a subset of the domain of the regression function $g(x)$. For an arbitrary level λ, consider the level set LS_λ as

$$\mathrm{LS}_\lambda \quad = \quad \Big\{x : g(x) \geq \lambda\Big\},$$

where x is the covariate(s) and g is the mean response function.

Level sets have important applications in many disciplines. In medicine, a practitioner is interested in determining the amount of drug that will produce a positive effect in a specific proportion of the population. A recent example related to the current SARS-CoV-2 pandemic is the estimation of the infective dose. The infective dose is the number of particles that allow a practitioner to detect the infection. This is an important step to understand viral pathology and its correlation with disease severity (see Karimzadeh et al. [2021]). The process to estimate this dose is a level set problem. In imaging, a correct estimation of a level set is useful to determine the location of a tumour. Level sets can also have applications in the change point problem in a regression model. Assume that there is at most one change point. Here, a sequence of random vectors $(X_1, Y_1), \ldots, (X_n, Y_n)$ is observed. The goal is to determine the point τ where the regression model changes. Under a parametric approach

for example, we test if the regression model $Y = X^T\beta + \epsilon$ changes to $Y = X^T\widetilde{\beta} + \epsilon$ for $Y_i \geq \tau$ or some values of X. Generally the null hypothesis, H_0*: there is no change point*, is rejected if the cumulative sum of a sequence of random variables crosses a boundary as discussed in Bhattacharya [1994]. This problem is related to a more complex version of a level set. We can exchange the function g for a random field $X(t)$. The corresponding set is called the excursion A set of $X(t)$ defined as $A = \{t \in T : X(t) \geq u\}$ [Adler, 1976]. This is a more complex type of set than the one we study in this thesis. We refer the interested reader into the papers of Kratz [2006], Adler [1976] and Adler [2010]. The later includes good introductory references to excursion sets. For the reader interested in bandwidth selection for the change point problem we refer them to the papers of Gijbels and Goderniaux [2004] and Yang and Zhang [2020]. Here the emphasis is not in estimating well the regression function everywhere, but in detecting the points where there is a change in the regression function. If the change comes from a discontinuity, that is the regression function is smooth except at a finite number of points, Gijbels and Goderniaux [2004] propose a bootstrap procedure to select the bandwidth. If the change is a structural change, that is the regression function changes at a point in the domain of the covariate, Yang and Zhang [2020] study a data driven method to choose the optimal bandwidth.

Although the level set problem is of interest in multiple disciplines, it has received less attention in the field of statistics. In this thesis we study how to optimally estimate such sets using a kernel plug-in estimator. We could broadly summarize the existing work available into two categories, a direct level set estimation (Willett and Nowak [2007], Cavalier [1997], Scott and Davenport [2007]), and an indirect approach (Laloë and Servien [2013],Jankowski et al. [2014]). Direct methods do not require estimation for the function g. For this type of estimator a loss or risk function is specified and one can optimize such function over all possible sets, which is computationally intensive. Typically, to ease the search, assumptions are made on the general shape of the level set (see Cavalier [1997]). However, these assumptions are hard to check in practice as discussed in Scott and Davenport [2007]. The second category consists of indirect methods. Here, the regression function is estimated and then

thresholded. This type of estimators are called plug-in estimators. Common methods to estimate g range from stronger assumptions such as parametric regression, to more flexible ones: kernel regression, splines, additive models, etc. These methods do not require extra assumptions on the set. We thus study a less computationally intensive, easy to understand for practitioners nonparametric kernel based plug-in estimator.

Arguably, kernels are the most popular tool for nonparametric estimators since they do not require strong assumptions on g. In addition, there is a vast literature available on this topic.

Choosing to work with kernels implies selecting a parameter called the bandwidth. This parameter controls the amount of smoothing for the estimate of g. Figure 1.1 shows how the choice of the bandwidth affects the shape of a kernel estimator. Bandwidth selection is a non trivial problem that generally is focussed on optimizing a global loss function (e.g. least squares cross validation, asymptotic mean integrated squared error). Nevertheless, a global loss function becomes suboptimal for the level set estimation problem which is local in nature. Figure 1.2 shows how the level set optimal bandwidth is different from a traditional bandwidth selector. In order to minimize a global loss function, the estimator for g has to fit well the sharp peak of the function. This results in a small bandwidth. However, the peak has less relevance when we estimate the level set $\widehat{LS}_{0.156} = [-1.036, 1.036]$. In this case the optimal bandwidth is visibly larger. This implies that current bandwidth selection methods do not work well for plug-in level set estimation problems.

The first to study bandwidth selection in the context of level sets are Samworth and Wand [2010]. The authors study bandwidth selection for the highest density region (HDR), a level set estimation problem where g is a density and the level λ depends on a pre-specified coverage probability p. The optimal bandwidth for the HDR is the one that minimizes the asymptotic approximation of a risk proposed by the authors. More recently, Doss and Weng [2018] extend the ideas in Samworth and Wand [2010] to higher dimensions where bandwidth selection is considerably harder.

This thesis consists of two parts. In the first, and main part of this thesis, our work

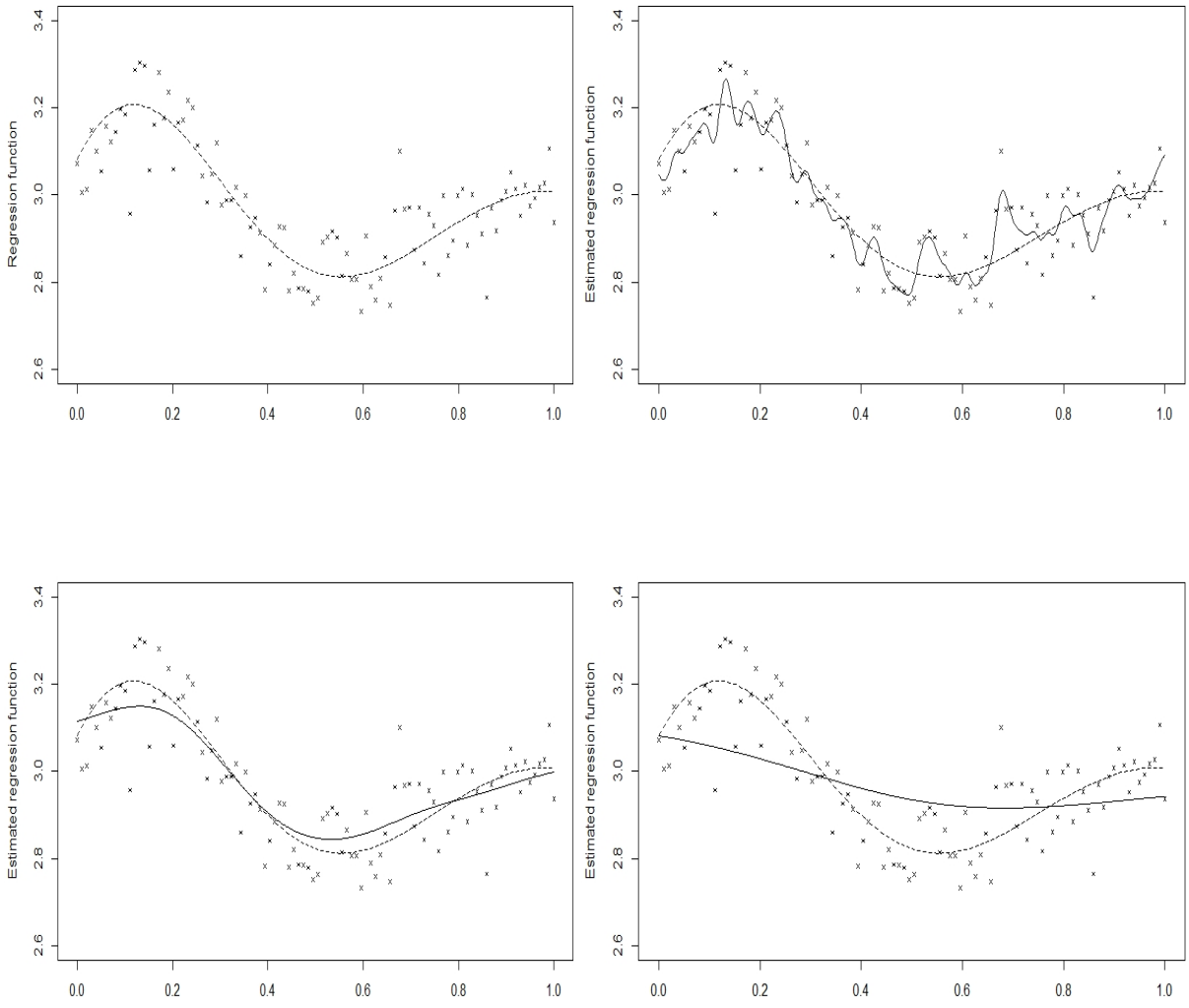

Figure 1.1: Three Nadaraya-Watson kernel estimators for three different bandwidths. The top left plot shows a sample of 100 observations and the true regression function in a dotted line. All other plots show the Nadaraya-Watson kernel estimator in a solid line and the true function in a dotted line. For the top right plot a bandwidth (h) that is too small is selected ($h = .01$). The lower right plot shows the kernel estimator with a bandwidth that is too large ($h = .25$). Finally, the lower left plot is the kernel estimator with an appropriate bandwidth ($h = .09$).

is similar in spirit as Samworth and Wand [2010]; Doss and Weng [2018], but adapted to regression. We study optimal bandwidth selection for the Nadaraya-Watson kernel estimator in one dimension. The ratio form of the Nadaraya-Watson estimator presents additional difficulties in deriving the theory not encountered in Samworth and Wand [2010]; Doss and Weng [2018]. We present a local loss function as an alternative to the L_2 metric and derive an asymptotic approximation of its corresponding risk. The level set optimal bandwidth is then the argument that minimizes this asymptotic approximation. We show that the rate of the level set optimal bandwidth, $h_{opt} = c_{opt} n^{-1/5}$, coincides with the rate from traditional global bandwidth selectors, although the constant c_{opt} is different from them. The level set optimal bandwidth has no closed solution, therefore we derive an algorithm to estimate the constant c_{opt} and obtain the practical bandwidth selector. We then study the performance of this practical bandwidth selector through simulations.

In our simulations we compare our proposed level set bandwidth selector with the least squares cross validation bandwidth selector. This is a data driven estimator for the mean squared error (the most commonly used L_2 metric in kernel estimation). The least squares cross validation bandwidth selector was considered in Samworth and Wand [2010] and Doss and Weng [2018]. In addition, we consider a plug-in estimator where the regression function $g(x)$ is estimated using the local polynomial kernel estimator in Fan et al. [1995]. This is a kernel estimator for which the Nadaraya-Watson estimator is a special case. We then compare its performance against the plug-in estimator proposed in this thesis. Our simulation results show that in general, for small samples and small values of λ, the level set optimal bandwidth shows improvement in estimating LS_λ when compared to the cross validation bandwidth selection. Compared to the local polynomial plug-in estimator, the level set optimal bandwidth also offers an advantage for small values of λ. However, as discussed in Samworth and Wand [2010]; Doss and Weng [2018], this improvement is not observed for all values of λ, across all functions $g(x)$. In addition, we find that although optimal bandwidth estimation is less intensive than direct methods, it still requires considerable computing time. We conclude the section with an analysis of the decompression sickness study data set from

the University of Wisconsin, Madison.

In the second part of this thesis, motivated by our simulation findings and the relationship of the level set estimation to the HDR, we study via simulations the properties of a plug-in estimator where the density is estimated with the log-concave mixed model. This is a mixture model with marginal densities that are log-concave. In particular, the log-concave maximum likelihood estimator (mle) (Dümbgen and Rufibach [2009]; Cule et al. [2010]) requires no estimation of tuning parameters. Moreover, the family of log-concave distributions is robust in the sense that it contains many commonly used densities and it is a parsimonious model. We thus consider it was important to compare this method to those developed in Samworth and Wand [2010]; Doss and Weng [2018].

In our simulation study we focus on univariate densities and compare the optimal bandwidth selector in Samworth and Wand [2010] with the nonparametric mle of a log-concave mixture model. We observe that when the number of components in the model is correctly specified, the log-concave plug-in estimator performs better than the kernel estimator for lower levels and similarly for the rest of the levels considered. We thus see that adding shape-constraints to the density removes the computational burden of bandwidth selection without sacrificing the accuracy. We finish the section with a real data example. We analyse the data on daily maximum temperatures in Melbourne, Australia. This data set is available in the **hdrcde** R-package and was analysed in Samworth and Wand [2010].

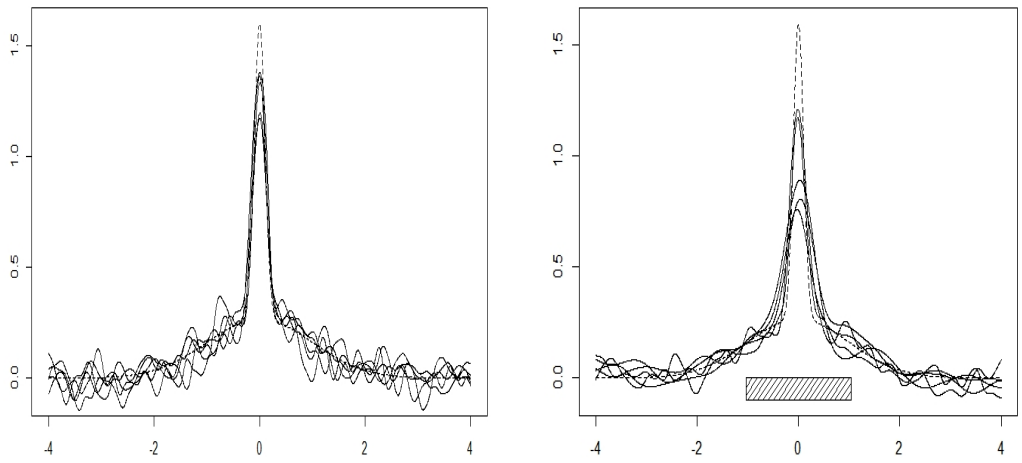

Figure 1.2: Five samples of 1000 observations were generated. The true function is shown on both panels with a dashed curve. The plot on the left shows five Nadaraya-Watson kernel estimates where the bandwidth was selected by minimizing the asymptotic mean integrated squared error. The panel on the right shows five Nadaraya-Watson kernel estimates where the bandwidth selected minimizes the asymptotic risk approximation for the $\mathrm{LS}_{0.156}$ level set. The true $\mathrm{LS}_{0.156} = [-1.036, 1.036]$ is shown.

1.2 Outline

Chapter 2 is dedicated to develop the necessary background for this thesis. Note that our work on level set estimation addresses only the case where $X \in \mathbb{R}$. Therefore the topics covered in the background section mainly focus on one dimension settings.

This thesis is divided in two main topics. The first is studied in Chapter 3 and the second in Chapter 4. Chapter 3 discusses how to optimally select the bandwidth for a plug-in kernel level set estimator. In Section 3.1 we present a local loss function inspired in the work of Samworth and Wand [2010] as an alternative to the traditional integrated squared error loss function. In Section 3.2 we develop the asymptotic theory for our bandwidth selector. We present an asymptotic approximation to the loss function introduced in Section 3.1. The optimal bandwidth for the level set estimation is thus the argument that minimizes the asymptotic approximation.

The remainder of Chapter 3 develops practical implementations of our theoretical results and studies their performance computationally. In Sections 3.3 and 3.4 we develop an algorithm to obtain the optimal bandwidth for level set estimation. This is necessary because the optimal bandwidth has no closed solution and the risk function depends on unknown quantities. In Section 3.5 we study the performance of our bandwidth selector. We divide the section into two since the tools available to compute the practical bandwidth are different depending on the response. First, we consider the problem with Y Gaussian. We emulate the simulation study in Samworth and Wand [2010]; Doss and Weng [2018].We compare the level set bandwidth selector proposed in this thesis against the least squares cross validation and the local polynomial kernel estimator. The latter is a kernel estimator that has been recommended by Wand and Jones [1994] based on their simulation evidence. Next, we consider the problem with Y binary. In this part of the section we compare the oracle bandwidth from our bandwidth selector against the local polynomial's oracle bandwidth. That is, we assume that all the functions needed to compute both bandwidths are known. We follow this approach because both bandwidths depend on the derivatives of the regression function and the literature on bandwidth selection for these functionals is, up to our knowledge, scarce

for Y not Gaussian. We conclude the chapter with an analysis of the decompression sickness study data set from the University of Wisconsin, Madison.

Chapter 4 covers the second part of this thesis. In this chapter we revisit the highest density region problem. We start with a plug-in estimator using a local polynomial kernel estimator, as we did in the context of regression. We analyse its performance against the bandwidth selector outlined in Samworth and Wand [2010]. We observe similar results as in the simulations in Section 3.5. The bandwidth selector in Samworth and Wand [2010] offers an advantage over the local polynomial kernel estimator. However, this is not uniformly across all densities considered. We conclude this section with a simulation study of a plug-in estimator for the highest density region where the density is estimated with the log-concave mixture model. We notice that when the number of components in the mixture model is correctly specified, the log-concave plug-in estimator performs better than the kernel estimator for lower levels and similarly for the rest of the levels considered. The log-concave plug-in estimator also offers the advantage that it is less computationally intensive than the bandwidth selector. In Section 4.3 we study a real data example. We analyse the data on daily maximum temperatures in Melbourne, Australia.

Lastly, we conclude with a discussion section and defer the proofs for our theorems and corollary to the appendix. We include sample code for our simulations and real-data examples in the appendix.

Chapter 2

Background

2.1 Density estimation

Given a set of n independent and identically distributed (IID) observations $X_1, X_2, \ldots, X_n$, a common question in statistics is to find the distribution $f(x)$ of these data points. The statistical model used to solve this question depends on the prior assumptions that we are able to make. In this thesis, we review three statistical models, all having different degree of robustness (flexibility). We cover the parametric model (least robust), shape constrained density estimation and kernel density estimation (most robust). We start with the parametric model.

2.1.1 Parametric density estimation

Assume that our observations $X_1, X_2, \ldots, X_n$ are IID and follow a known distribution $f(x; \theta_0)$ where the true parameter θ_0 is fixed but unknown. The density estimation problem then reduces to the estimation of the parameter, or a finite set of parameters if θ_0 is a vector.

One method to estimate θ_0 is the maximum likelihood estimation (MLE) (Fisher and Russell [1922], Fisher [1925]). The likelihood is the function defined as

$$L(\mathbf{x}; \theta) \quad = \quad \prod_{i=1}^{n} f(x_i; \theta)$$

where $x_1, \ldots, x_n$ are the realizations of the n random variables.

The maximum likelihood estimator $\widehat{\theta}_{\text{mle}}$ is the parameter that makes the observed sample most likely to be observed. That is

$$\widehat{\theta}_{\text{mle}} = \arg\max_{\theta \in \Theta} L(\mathbf{x}; \theta)$$

where Θ is the parameter space. Under regularity conditions, $\widehat{\theta}_{\text{mle}}$ is consistent [Wald, 1949], that is,

$$\widehat{\theta}_{\text{mle}} \xrightarrow{P} \theta_0.$$

It is efficient and asymptotically normal [Cramér, 1946]

$$\sqrt{n}\left(\widehat{\theta}_{\text{mle}} - \theta_0\right) \xrightarrow{d} N\left(0, I^{-1}\right)$$

where I is the Fisher's information matrix.

An alternative method, although less efficient than the MLE, to estimate θ_0 is the method of moments. Suppose θ_0 is d-dimensional. Consider the first d theoretical moments

$$M_i(\theta) = \mathrm{E}_\theta[X^i] = \int x^i dF_\theta(x)$$

for $i = 1, \ldots, d$ and F_{θ_0}, the probability distribution that depends on the true parameter θ_0. The sample moments are then

$$\widehat{M_i} = \frac{1}{n}\sum_{j=1}^{n} X_j^i$$

for $i = 1, \ldots, d$. The method of moments estimator $\widehat{\theta}_{\text{MOM}}$ is the solution to the set of d equations

$$M_i(\theta) = \widehat{M_i} \qquad \text{for i=1,...,d.}$$

Under regularity conditions, $\widehat{\theta}_{\text{MOM}}$ is consistent and converges to a Gaussian distribution as $n \to \infty$ [see Wasserman [2010]].

Although the MLE has attractive properties, it relies on the assumption that the true density is indeed f. More robust methods are needed for studies where less is known about the form of f. As pointed out in White [1982], a misspecification of the density can be problematic. More robust methods are needed for studies where less is known on the form of f.

Next, we review the log-concave density estimation method. This is also a maximum likelihood method with the difference that less is known about the sample's underlying density.

2.1.2 Log-concave density estimation

The development of maximum likelihood estimation (MLE) for shape-constrained density estimation was first discussed by Grenander [1956] for monotonic, non-increasing densities. Rao [1969] studied the maximum likelihood estimator for unimodal densities with known mode. The author proved the consistency and asymptotic distribution for the maximum likelihood estimator. Later, Groeneboom et al. [2001] proved consistency and the asymptotic distribution at a fixed point for the maximum likelihood estimator of convex (and decreasing) density functions. However, shape constrained MLE might not be a feasible method for all shape constrained densities. For example, for a unimodal density with unknown mode, the mle does not exist [see Birgé [1997]]. As discussed in Walther [2009], the additional assumption of log-concavity can alleviate this problem since log-concave densities belong to the family of uni-modal densities (although the reverse is not true). In particular, Balabdaoui et al. [2009] proposed the mode of the log-concave MLE as an estimator of the mode of the true density and studied its properties.

A log-concave density f_0 has the form

$$f_0(x) \quad = \quad \exp \varphi(x)$$

where $\varphi(x)$ is a concave function such that $-\infty \leq \varphi(x) < x$ for all x. That is for any

$x_1, x_2 \in \mathbb{R}$ and any $\lambda \in (0,1)$

$$\varphi\left(\lambda x_1 + (1-\lambda)x_2\right) \geq \lambda\varphi\left(x_1\right) + (1-\lambda)\varphi\left(x_2\right).$$

Let $X_1, X_2, \ldots, X_n$ be an IID sample from the distribution function F where $X_i \in \mathbb{R}$ for $i = 1, 2, \ldots, n$. Let $X_{(1)} < X_{(2)} < \ldots < X_{(n)}$ be the corresponding ordered statistics. The non-parametric log-likelihood function is, based on f_0,

$$l_n(f) \quad = \quad \int \log\left(f_0(x)\right) d\mathbb{F}_n(x) = \int \varphi(x) d\mathbb{F}_n(x).$$

The function $\mathbb{F}_n$ is the empirical distribution function defined as

$$\mathbb{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} 1_{X_i \leq x}.$$

The log-concave maximum likelihood estimator is then

$$\widehat{f}_{\text{mle}} \quad = \quad \underset{f \in F}{\arg\max} \int \log\left(f_0(x)\right) d\mathbb{F}_n(x)$$

where the family F denotes the set of all log-concave densities. This constrained optimization problem is equivalent to maximizing

$$\widetilde{l}_n\left(\varphi\right) \quad = \quad \int \varphi(x) d\mathbb{F}_n(x) - \int \exp\varphi(x) dx$$

over the set of all concave functions. Theorem 3.1 in Silverman [1982] guarantees that the log-concave maximum likelihood estimator

$$\widehat{f}_{\text{mle}} \quad = \quad \exp\widehat{\varphi}_{\text{mle}}$$

is indeed a density. The family of log-concave densities is robust in the sense that it includes most of the commonly used densities in practice such as Gaussian, exponential, gamma with shape parameter greater than or equal to one, logistic, Gumbel and Laplace to name a few,

while keeping a parsimonious model. An added advantage for this type of shape constrained estimation is that, unlike kernel estimators, we do not need to select tuning parameters which has been shown to be non-trivial.

For $d = 1$, Walther [2002] proved existence and uniqueness of the log-concave maximum likelihood estimator while Pal et al. [2007] proved its consistency. Theorem 2.1 in Dümbgen and Rufibach [2009] showed that φ_{mle} is linear on $[X_{(1)}, X_{(n)}]$ with knots in the set of unique observations $\{x_1, \ldots, x_m\}$ for $m \leq n$ and $\widehat{\varphi}_{\text{mle}} = -\infty$ on $\mathbb{R} \cap [X_{(1)}, X_{(n)}]^c$. For higher dimensions, Cule et al. [2010] proved existence and uniqueness of the log-concave maximum likelihood estimator and Cule and Samworth [2010] proved strong consistency of $\widehat{f}_{\text{mle}}$ under exponentially weighted supremum norms.

Let $\widehat{F}_n$ be the estimator of the distribution function F_0 where

$$\widehat{F}_n(x) \quad = \quad \int_{-\infty}^{x} \widehat{f}_{\text{mle}}(s)ds$$

Dümbgen and Rufibach [2009] derived two important properties of this estimator

$$\mu\left(\widehat{F}_n\right) = \mu\left(\mathbb{F}_n\right) \qquad\qquad \text{Var}\left(\widehat{F}_n\right) \leq \text{Var}\left(\mathbb{F}_n\right)$$

That is, the mean of the estimated distribution function is the same as that for the empirical distribution function and the variance of $\widehat{F}_n$ is at most the variance of the empirical distribution.

Balabdaoui et al. [2009] is one of the first works on convergence rates for the log-concave maximum likelihood estimator. Consider a point x_0 where $f_0(x_0) > 0$ for a log-concave density $f_0 \in F$. Balabdaoui et al. [2009] show that $(\widehat{f}_{\text{mle}}(x_0) - f_0(x_0))$ converges, pointwise, to a non-Gaussian distribution at a rate $O\left(n^{-2/5}\right)$.

Doss and Wellner [2016], and Kim and Samworth [2016] investigated global convergence rates for the log-concave maximum likelihood estimator using different approaches. Let

$H^2(f_1, f_2)$ be the square of the Hellinger distance between densities f_1 and f_2 where

$$H^2(f_1, f_2) \;=\; \frac{1}{2}\int \left(\sqrt{f_1(x)} - \sqrt{f_2(x)}\right)^2 dx.$$

Doss and Wellner [2016] developed global convergence rates for $\widehat{f}_{\text{mle}}$. The authors proved that $\widehat{f}_{\text{mle}}$ has uniform almost sure convergence to f_0, it is bounded uniformly and $\widehat{f}_{\text{mle}}$ is Hellinger-consistent and has a convergence rate of $O_p\left(n^{-2/5}\right)$. Kim and Samworth [2016] also presented global convergence rates with a focus on a minimax approach. We recall that for a minimax estimator, let θ be a parameter in a parameter space Θ_n. Denote as $\widehat{\theta}_n$ an estimator. Given a loss function $L(\widehat{\theta}_n, \theta)$, the minimax risk R_n is commonly defined as

$$R_n \;\equiv\; R(\Theta_n) \;=\; \inf_{\widehat{\theta}} \sup_{\theta \in \Theta_n} \mathbb{E}\left[L\left(\widehat{\theta}, \theta\right)\right].$$

Theorem 4 in Kim and Samworth [2016] showed that no estimator $\widehat{f}_n$ computed from a sample of size n can achieve a supremum risk (under $H^2\left(\widehat{f}_n, f_0\right)$) with a rate faster than $O\left(n^{-4/5}\right)$. That is

$$\inf_{\widehat{f}_n} \sup_{f_0 \in F} \mathrm{E}\left[H^2\left(\widehat{f}_n, f_0\right)\right] \geq O\left(n^{-4/5}\right).$$

Moreover, Theorem 5 in Kim and Samworth [2016] demonstrated that $\widehat{f}_{\text{mle}}$ reaches the minimax optimal rate.

Kim et al. [2018] also investigate on the adaptability of the log-concave maximum likelihood estimator. Kim et al. [2018] then showed that if $\log f_0$ is affine, or close to an affine function with respect to an appropriate distance metric, they achieved a faster convergence rate than that in Doss and Wellner [2016] and Kim and Samworth [2016]. In fact, under some conditions on f_0, $\widehat{f}_{\text{mle}}$ reaches parametric or near parametric convergence rates. Consider a density f, not necessarily log-concave, with support on an interval $\mathcal{I} = [s_1, s_2]$ such that $\log f$ is affine (that is $\log f(x) = ax + b$ for some constants a, b) on $\mathcal{I}$. Let $\mathcal{A}$ be the class of such

log-affine densities parametrized as

$$f_{\alpha,s_1,s_2}(x) = \begin{cases} \frac{1}{s_2-s_1}\mathbb{1}_{\{x\in\mathcal{I}\}} & \text{for } \alpha = 0 \\ \frac{\alpha}{\exp(\alpha s_2)-\exp(\alpha s_1)}\exp(\alpha x)\mathbb{1}_{\{x\in\mathcal{I}\}} & \text{for } \alpha \neq 0 \end{cases}$$

For instance, the uniform density with compact support belongs to $\mathcal{A}$.

Let $d_{\mathrm{TV}}(f_1, f_2)$ be the total variation metric where

$$d_{\mathrm{TV}}(f_1, f_2) \quad = \quad \int_{\mathbb{R}} \big|f_1(x) - f_2(x)\big| dx$$

Kim et al. [2018, Theorem 1] showed that if $f_0 = f_{\alpha,s_1,s_2}$ such that $|\alpha|(s_2 - s_1) < \log n$

$$\mathrm{E}\left[d_{\mathrm{TV}}\left(\widehat{f}_{\mathrm{mle}}, f_0\right)\right] \leq \frac{c}{n^{1/2}}$$

for some constant c independent of n. An example where this is satisfied is if f_0 is Uniform $[s_1, s_2]$ for $-\infty < s_1 < s_2 < \infty$. If $|\alpha|(s_2 - s_1)$ is large, the rate above slows down to $O\left(\frac{\log n}{n^{1/2}}\right)$.

Let $d_{KS}^{(n)}(f_0, f_{\alpha,s_1,s_2})$ be a metric on the distribution function of f_0 and f_{α,s_1,s_2} (F_0 and F_{α,s_1,s_2} respectively)

$$d_{KS}^{(n)}(f_0, f_{\alpha,s_1,s_2}) \quad = \quad \sup_{x\in\mathbb{R}} \big|F_{\alpha,s_1,s_2}^n(x) - F_0^n(x)\big| + \sup_{x\in\mathbb{R}} \big|(1 - F_{\alpha,s_1,s_2}(x))^n - (1 - F_0(x))^n\big|.$$

Kim et al. [2018] showed that if $f_0 = f_{\alpha,s_1,s_2}$ where $|\alpha|(s_2 - s_1)$ is large and f_{α,s_1,s_2} is a member of the class of log-concave densities with support on an interval $\mathcal{I}$ where $\log f$ is affine,

$$\inf_{f_{\alpha,s_1,s_2}\in F}\left(d_{\mathrm{TV}}(f_0, f_{\alpha,s_1,s_2}) + d_{KS}^{(n)}(f_0, f_{\alpha,s_1,s_2})\right) \quad = \quad o\left(\frac{n^{-2/5}}{\log n}\right).$$

This guarantees that the convergence rate of $\widehat{f}_{\mathrm{mle}}$ to f_0 is the parametric rate up to a logarithmic factor. Kim et al. [2018, Theorem 3] extended these results to the class of log-concave

densities that are close to a k affine log-concave density in the sense that,

$$\inf_{f_k \in F^k} d_{\mathrm{KL}}\left(f_0, f_k\right) \quad = \quad \left(\frac{k}{n}\log^{5/4} n\right)$$

where F^k is the class of log-concave densities for which $\log f$ is k affine and d_{KL} is the Kullback-Leibler divergence. Under these conditions, the convergence rate for the log-concave maximum likelihood estimator is of the order $O\left(\frac{k}{n}\log^{5/4} n\right)$ when $k = o\left(n^{1/5}\log^{-5/4} n\right)$. This is a faster rate than the minimax rate $O\left(n^{-4/5}\right)$ of all log-concave densities.

Besides its adaptability property, the log-concave maximum likelihood estimator is also robust. As discussed in Dümbgen et al. [2011], the class of log-concave densities F is not convex. It is then not obvious that a density $f \in F$ is close to the true density f_0. Dümbgen et al. [2011] proposed the log-concave projection ψ^*. Let $\mathcal{P}$ be the class of unidimensional probability measures P satisfying $\mathrm{E}[|X|] < \infty$ and $P\left(X = x\right) < 1$ for any point x (the probability measures with all mass at one point are excluded). The log-concave projection is thus a map $\psi^* : \mathcal{P} \to F$ given by

$$\psi^*(P) = \arg\max_{f \in F} \int_{\mathbb{R}} \log f dP$$

Note that when $P = \mathbb{F}_n$, the empirical distribution, this coincides with the log-concave maximum likelihood estimator. However, if P has a density f_0 such that $\int_{\mathbb{R}} f_0 |\log f_0| < \infty$, then $\psi^*(P)$ is the closest log-concave density to f_0 in the Kullback-Leibler divergence sense. Cule and Samworth [2010, Theorem 4] proved that $\widehat{f}_{\mathrm{mle}} \overset{a.s.}{\to} \psi^*(P)$ in case of misspecification.

From a practical perspective, Walther [2002] proposed an iterative convex minorant algorithm to compute $\widehat{f}_{\mathrm{mle}}$ with good performance when compared to other algorithms available at the time [seeRufibach [2007]]. Duembgen et al. [2007] later proposed an alternative algorithm called the active set algorithm which seems to be more efficient [Dümbgen and Rufibach, 2009]. Moreover, Rufibach and Dümbgen [2006] implemented the iterative convex minorant algorithm and the active set algorithm in the **logcondens** R-package available via "CRAN" making the computation of $\widehat{f}_{\mathrm{mle}}$ easily accessible. Later, Cule et al. [2009] imple-

mented the algorithm for the multivariate log-concave maximum likelihood estimator in the **LogConcDEAD** R-package.

Chang and Walther [2007] used log-concave densities for unsupervised classification via a mixture model where the number of m components is known. For the model

$$f(x) = \sum_{m=1}^{k} \pi_m f_m(x),$$

Chang and Walther [2007] assumed that the marginal densities f_m belonged to the class of log-concave densities and $\sum_{m=1}^{k} \pi_m = 1$. The goal was to estimate π_m and f_m non-parametrically using the EM-algorithm. Chang and Walther [2007] showed, through a simulation study, that this non-parametric classification approach outperformed the parametric algorithm based on the normal mixture model for a large (500 observations) and a small (50 observations) sample size when there are $k = 2$ components and these are not Gaussian (marginally). This is expected because of the robustness of log-concave densities. However, for a correctly specified model with Gaussian marginal densities, the non-parametric classification approach was not compromised. Its performance was very similar to the fully parametric approach for a large and small sample size.

2.1.3 Kernel density estimation

The kernel density estimator is defined as $\widehat{f}_{n,h}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)$ where K is a function called kernel. The kernel is symmetric about zero and satisfies $\int K(s)ds = 1$. In most cases, K is a density itself. This non-parametric estimator depends on the choice of a parameter called the bandwidth. This parameter, denoted by h, controls the smoothing degree of the density estimator $\widehat{f}_{n,h}$.

There exist different approaches to select the appropriate bandwidth. We present some of them to Section 2.2.2.4 where we discuss bandwidth selection with more detail in the context of regression. However, all the methods that we discuss there can be easily applied to the context of density estimation. In this section we only review the bandwidth selection based

on the mean integrated squared error (MISE) criteria. The MISE will help us to compare kernel density estimation with parametric estimation.

In general, the bandwidth is chosen to minimize an appropriate risk function. The loss function that is most commonly used in practice is the L_2 distance

$$\int \left(\widehat{f}_{n,h}(x) - f(x)\right)^2 dx.$$

The average of this integral is known as the risk for the estimator $\widehat{f}_{n,h}$, and is referred to as the mean integrated squared error (MISE)

$$\text{MISE}(h) \quad = \quad \text{E}\left[\int \left(\widehat{f}_{n,h}(x) - f(x)\right)^2 dx\right].$$

The MISE as a function of h has a complex structure. An asymptotic approximation of the MISE (AMISE) eases the dependence on h and it is therefore typically used [see Wasserman [2006]]. It can be shown, by decomposing the MISE into an integral on the variance and bias terms and using Taylor expansion, that

$$\text{MISE}(h) \quad = \quad \frac{1}{nh}\int K^2(s)ds + \frac{h^4}{4}\int s^2K(s)ds \int \left(f''(x)\right)^2 dx + o\left(\frac{1}{nh} + h^4\right)$$

and therefore

$$\text{AMISE}(h) \quad = \quad \frac{1}{nh}\int K^2(s)ds + \frac{h^4}{4}\int s^2K(s)ds \int \left(f''(x)\right)^2 dx.$$

The later one is minimized at

$$\begin{aligned} h_{\text{AMISE}} &= \arg\min_h \text{AMISE}(h) \\ &= \left(\frac{\int K^2(s)ds}{(\int s^2K(s)ds)^2 \int (f''(x))^2 dx}\right) n^{-1/5}. \end{aligned}$$

However, the functions that h_{AMISE} depends on are unknown in practice. Wand and Jones [1994] proposed a *quick and simple* bandwidth selector. Assume that the true density f

follows a Gaussian distribution. We thus obtain a simpler form of h_{AMISE} that is a good alternative if the data is close to Gaussian (see section 3.2.1 in Wand and Jones [1994]). For densities that are not Gaussian Wand and Jones [1994] propose *an l-stage bandwidth selector*. This is a plug-in estimator for h_{AMISE}. In this bandwidth selection technique, we substitute the unknown functional in h_{AMISE} for a kernel estimator that depends on its respective optimal bandwidth. This, at the same time, depend on further unknown functionals that we can estimate using kernels that will again depend on their respective bandwidths. This is iteratively repeated until the l^{th} kernel estimator is a simple function that does not depend on any future bandwidth. The unknown component of h_{AMISE} is the integral of the type

$$S\left(f^{(r)}\right) = \int_{\Omega_X} f^{(r)}(x)^2 dx.$$

where Ω_X denotes the support of X. Wand and Jones [1994] showed that under some smoothness assumptions

$$S\left(f^{(r)}\right) = (-1)^r \int_{\Omega_X} f^{(2r)}(x) f(x) dx.$$

It is then sufficient to study the functions of the type

$$\psi_r = \int_{\Omega_X} f^{(r)}(x) f(x) dx.$$

This is the expectation of the r^{th} derivative of f. Wand and Jones [1994], and the references that there appear, proposed thus the estimator

$$\begin{aligned} \widehat{\psi}_r &= n^{-1} \sum_{i=1}^{n} \widehat{f}^{(r)}\left(X_i; \widetilde{h}\right) \\ &= n^{-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \widetilde{h}^{-1} K^{(r)}\left(\frac{X_i - X_j}{\widetilde{h}}\right). \end{aligned}$$

The kernel K and bandwidth $\widetilde{h}$ for this estimator may be different to those used in $\widehat{f}_{n,h}$. Under conditions in Section 3.5 in Wand and Jones [1994], the bandwidth that minimizes the

asymptotic mean squared error for $\widehat{\psi}_r$ is

$$\widetilde{h}_{\text{AMSE}} = \left(\frac{k! L^{(r)}(0)}{-\mu_k(L)\psi_{r+k} n} \right)^{1/(r+k+1)}.$$

Here $k = 2, 4, \ldots$, is the order of the kernel and $\mu_k(L) = \int z^k L(z) dz$. As discussed earlier, this bandwidth depends on the function ψ_{r+k} that is estimated with a kernel and requires a further bandwidth. The process continues until we use an estimator that does not depend on any further bandwidth. One option is to assume f is Gaussian, which yields a simpler estimator for ψ that does not depend on any bandwidth.

From the rate of h_{AMISE} we can infer that, for a kernel density estimator, the best possible rate of the MISE is $O\left(n^{-4/5}\right)$. This in comparison to the MISE of a parametric model which rate is $O\left(n^{-1}\right)$. The kernel density estimator is a more flexible model because it does not need a pre-specified form on f. However, the price that this estimator pays (assuming that the parametric model is correctly specified) is in its efficiency.

2.2 Regression

In this section we review a class of statistical models that are used to describe a data set. As discussed in McCullagh and Nelder [1989], regression is a statistical model that help us to summarize data in terms of the systematic effect and a summary of random variation.

A data set is a collection of variables of interest. It is generally assumed that the independent variables, also known as the covariates or regressors, have an effect on the dependent variable or outcome variable. Typically, the way to describe the relationship between the covariate and the outcome is through the regression function $g(x) = \Psi\left(E[Y|X]\right)$ for a known function Ψ.

We can generalize the methodology for regression problems in two categories, parametric and nonparametric regression. For parametric regression, the function g is described by a finite set of parameters. For nonparametric regression, assumptions on g are less strict and the problem is a curve estimation problem.

2.2.1 Parametric regression

We start this section with a review on parametric regression models. We first discuss linear regression (i.e., Ψ is the identity function) and later we review a more general class of models, the generalized linear models.

2.2.1.1 Linear regression

Suppose there is a collection of independent and identically distributed points (X_i, Y_i) for $i = 1, \ldots, n$. $X_i \in \mathbb{R}$ is called the covariate, also known as the independent variable or regressor. $Y_i \in \mathbb{R}$ is the response, also known as the dependent variable or outcome variable. Assume there is an underlying true model that fully describes the relationship between X and Y

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i.$$

From this model, X and Y have a linear relationship with some added noise ε_i. We further assume that all errors ε_i have a Gaussian distribution with $\mathrm{E}[\varepsilon_i] = 0$ and constant variance σ^2.

Linear regression is a methodology to fit the line that best describes the relationship between X and Y on average. Let $g(x) \equiv \mathrm{E}[Y|X = x]$. The goal of linear regression is to estimate the unknown parameters (β_0, β_1) in

$$g(x_i) \quad = \quad \mathrm{E}[Y_i|X_i = x_i] = \beta_0 + \beta_1 x_i. \tag{2.1}$$

McCullagh and Nelder [1989] summarized the historical development of least squares, first introduced by Gauss. Least squares is a method to estimate the parameters (β_0, β_1). The estimators are chosen to minimize the square sum of the vertical distances between the

observed and predicted outcomes. The optimal parameters for the model specified in (2.1)are

$$\begin{aligned}(\widehat{\beta}_0, \widehat{\beta}_1) &= \underset{(\beta_0,\beta_1)}{\arg\min} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i\right)^2 \\ &= \underset{(\beta_0,\beta_1)}{\arg\min} \sum_{i=1}^{n} \left(Y_i - \beta_0 - \beta_1 x_i\right)^2\end{aligned}$$

The solution to the minimization problem is derived by solving the system of equations given by the partial derivatives

$$\frac{\partial \widetilde{L}}{\partial \beta_i} = 0 \qquad \text{for } i = \{0, 1\}$$

where $\widetilde{L}(\beta) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 x_i)^2$ for a vector of parameters $\beta = (\beta_0, \beta_1)$. This yields the following solution to the system of equations

$$\begin{aligned}\widehat{\beta}_0 &= \frac{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} Y_i - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \\ \widehat{\beta}_1 &= \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}\end{aligned}$$

which can be rewritten in the more traditional matrix form

$$\begin{aligned}\widetilde{L}(\beta) &= \left(\mathbf{Y} - \mathbf{X}^T\beta\right)^T \left(\mathbf{Y} - \mathbf{X}^T\beta\right) \\ \widehat{\beta} &= \left(\mathbf{X}^T\mathbf{X}\right)^{-1} \mathbf{X}^T\mathbf{Y}.\end{aligned}$$

Here

- $\mathbf{Y}$ is the vector of observed responses,
- $\mathbf{X}$ is the design matrix. For a problem with one covariate, $\mathbf{X}$ is a $n \times 2$ matrix. The first column is a column vector of ones to account for the line intercept in the model. The second column records the value of the covariate for each observation,
- β is the vector of unknown parameters (β_0, β_1), and

$\widehat{\beta}$ is the vector of estimated parameters $\widehat{\beta} = \left(\widehat{\beta}_0, \widehat{\beta}_1\right)^T$.

If σ^2 is known, under the assumptions on the error described previously, it is a general result that the least squares estimator $\widehat{\beta}$ follows a Gaussian distribution. In particular

$$\widehat{\beta} \sim N\left(\beta, \sigma^2_{\beta}\right).$$

Note that $\left(\widehat{\beta}_0, \widehat{\beta}_1\right)$ are unbiased estimators for (β_0, β_1). The variance-covariance matrix σ^2_{β} of our estimators is given by

$$\sigma^2_{\beta} \quad = \quad \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\sigma^2.$$

If σ^2 is unknown, we can estimate it with the errors of the predicted Y_i. Let $r_i = Y_i - \widehat{\beta}_0 - \widehat{\beta}_1 X_i$ be the residual for the i^{th}observation. A known estimator for σ^2 is

$$\widehat{\sigma}^2 = \frac{\sum_{i=1}^{n} r_i^2}{n-2}.$$

The number two in the denominator is to account for the two parameters (β_0, β_1) that were estimated through least squares. We can replace σ^2 with $\widehat{\sigma^2}$ in the variance-covariance matrix to obtain estimates for the variances and the covariance. Let $s^2_{b_i}$ be the estimated variance for $\widehat{\beta}_i$. It is a known result that the following statistic has a t-distribution and it becomes useful to do statistical inference on the regression parameters.

$$\frac{\widehat{\beta}_i - \beta_i}{s_{b_i}} \sim t_{(n-2)}.$$

In this section, we addressed the problem of estimating the function that describes the relationship between X and Y. We assumed that this function, the conditional expectation $g(x)$, is a linear function with unknown parameters. We reviewed how to estimate these parameters and stated their statistical properties.

For some practical contexts, the assumption of normality in the error's distribution is

restrictive. As an example, if the response Y is a factor variable, linear regression cannot give accurate predictions. Generalized linear models (GLMs) are an extension to linear regression and adopt the methodology to a class of distributions known as the exponential family. Gaussian distribution is one member of this family. In the next section we review exponential families before discussing GLMs.

2.2.1.2 Exponential Family

A random variable Y that is a member of the exponential family distribution has a density of the form

$$f(y;\theta,\zeta) = \exp\left\{\frac{y^T\theta - B(\theta)}{A(\zeta)} + C(y,\zeta)\right\} \tag{2.2}$$

where A, B and $C$are known functions and ζ is the dispersion parameter.

There is no unique way to parametrize an exponential family density. If ζ is known, then the distribution above is called the canonical distribution and θ, the canonical parameter. Examples of the most common members of the exponential family are the Gaussian, exponential, Gamma, Poisson and Bernoulli densities.

From this parametrization we can recover characteristics of the distribution. Let $g = \mathrm{E}[Y]$. We then have, $g = B'(\theta)$ and $\zeta \operatorname{Var}(g) = \zeta B''(\theta)$. That is, the variance is known to be a function of the mean. Thus, GLMs allow the variance to be dependent on the mean.

As discussed previously, the assumptions on linear regression (normality and constant variance) become void for some statistical problems. We had mentioned the example where the response Y is a factor variable. Other scenario common in practice is when Y is a counting variable. Generalized linear models propose a framework where these assumptions get relaxed.

2.2.1.3 Generalized Linear Models

Suppose there is a collection of n independent and identically distributed points (X_i, Y_i) for $i = 1, \ldots, n$. As in linear regression, $x \in \mathbb{R}$ is the covariate and $Y_i \in \mathbb{R}$ is the response

variable. Different from linear regression, the response Y is not constrained to a Gaussian distribution. GLMs assume the distribution of Y_i belongs to the exponential family reviewed in the previous section. Recall that for linear regression

$$g(x) \quad = \quad \mathrm{E}[Y|X = x] = \beta_0 + \beta_1 x.$$

That is, covariate X has a direct additive effect on the conditional mean of Y.

We motivate the need for an extension of linear regression. Suppose that Y has a Bernoulli distribution where the response is a binary factor variable taking values $\{0, 1\}$. Clearly, $\mathrm{E}[Y|X]$ has a range on the interval $[0, 1]$. However, the linear function in linear regression $\beta_0 + \beta_1 X$ can take any value on the real line. For this motive, the linear regression model becomes implausible under this scenario. Let $g(x) = E[Y|X = x]$. A more appropriate model would be,

$$\log\left(\frac{g(x)}{1 - g(x)}\right) \quad = \quad \eta(x) = \mathbf{X}^T\beta.$$

The link function $\Psi(g) = \log\left(\frac{g}{1-g}\right)$ maps the interval $[0, 1]$ into the real line. This mapping is not the only alternative. Other possibilities for $\Psi(g)$ are: $\log\left(-\log(1 - g)\right)$ or $\phi^{-1}(g)$, where ϕ is the Gaussian distribution function.

For GLMs, the effect of the covariates on the conditional mean is modeled as

$$\Psi(g) \quad = \quad \eta = \beta_0 + \beta_1 X.$$

When $\Psi(g)$ coincides with the canonical parameter in an exponential family member, then Ψ is called the canonical link. For the Bernoulli distribution, $\Psi(g) = \log\left(\frac{g}{1-g}\right)$ is the canonical link. For this work we always assume the canonical link. Table 2.1 shows the corresponding canonical link to each of the most common distributions in the exponential family [Nelder and McCullagh, 1989]. Note that for a Gaussian distribution, the link function is the identity function and the model reduces to that of linear regression.

Distribution	**Canonical link**
Gaussian	g
Poisson	$\log(g)$
Binomial	$\log\left(\frac{g}{1-g}\right)$
Gamma	g^{-1}
Inverse Gaussian	g^{-2}

Table 2.1: Canonical link functions of some common univariate distributions in the exponential family.

The goal in GLMs is to find the vector of parameters that maximize the sample's log-likelihood. For a member of the exponential family, the sample's likelihood is

$$L(y;\theta,\zeta) \quad = \quad \prod_{i=1}^{n} \exp\Big\{\frac{y_i\theta_i - B(\theta_i)}{A(\zeta)} + C(y,\zeta)\Big\}.$$

where $A(\cdot)$, $B(\cdot)$ and $C(\cdot)$ are known functions. Maximizing the likelihood is equivalent to maximizing the log-likelihood

$$l(y;\theta,\zeta) \quad = \quad \sum_{i=1}^{n} \Big\{\frac{y_i\theta_i - B(\theta_i)}{A(\zeta)} + C(y,\zeta)\Big\}.$$

In terms of the parameter vector β, assuming the canonical link, the log-likelihood is expressed as follows

$$l(\beta;y,\zeta) \quad = \quad \sum_{i=1}^{n} \Big\{\frac{y_i X_i^T\beta - B(X_i^T\beta)}{A(\zeta)} + C(y,\zeta)\Big\}.$$

The estimated parameter vector $\widehat{\beta}_{\text{mle}}$ is the solution to the score equation $\frac{\partial l}{\partial \beta} = 0$. The solution to this equation has no closed form. It is thus necessary to solve the equation numerically using the scoring method [Nelder and Wedderburn, 1972]. Note that McCullagh and Nelder [1989] discussed that when the canonical link is used, the Newton-Raphson method reduces to the same algorithm as the scoring method. If the canonical link is not used, the

scoring method is typically simpler. This is because the expected value of the Hessian matrix is generally less complex than the actual value of the matrix. In the scoring method, the parameters estimates are computed iteratively by

$$\beta^* = \left(\mathbf{X}^T\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{W}\mathbf{Z}.$$

Here

β^* is the vector of revised estimated parameters $\beta^* = (\beta_0^*, \beta_1^*)^T$

Z is the adjusted dependent variable. It is a vector of length n with components $z_i = \widehat{\eta}_i + (y_i - g_i)\frac{\partial \eta_i}{\partial g_i}$.

X is the design matrix. For a problem with one covariate, **X** is a $n \times 2$ matrix. The first column is a column vector of ones to account for the line intercept in the model. The second column records the value of the covariate for each observation.

W is commonly referred as the matrix of weights. It is an $n \times n$ diagonal matrix with elements $w_i^{-1} = \frac{\partial \eta_i}{\partial g_i} V_i$ where V, the variance of Y, is a function of the mean g. When choosing the canonical link, $\frac{1}{B''(\theta)}\frac{1}{\Psi'(g)} = 1$ and **W** is simplified. Under this condition, each diagonal element is $w_i = V(g_i)$ for $i = 1, \ldots, n$.

Note that **Z** and **W** are initialized with the vectors $\beta^{(0)}$ and thus $\eta^{(0)}$ and $g^{(0)}$. The estimate $\widehat{\beta} = \left(\widehat{\beta}_0, \widehat{\beta}_1\right)$ is the last iteration of β^*.

In this section, we addressed the problem of estimating the function that describes the relationship between X and Y when the response Y belongs to the exponential family. We assumed that the covariate has a linear effect on the response through a known function Ψ known as the link function. In the following section, we relax the assumptions even further and allow a non-linear relationship between the covariate and the response through the canonical link function. This area in statistics is known as nonparametric regression.

2.2.2 Non parametric regression

In the previous section the goal was to estimate a finite number of parameters where we assumed a particular form for the true η. In this section, we relax this assumption. The goal is to estimate a general unknown function η. Under this framework, the regression model for members of the exponential family is

$$\Psi\left(g(x)\right) \quad = \quad \eta(x)$$

for an unknown function $\eta(x)$ and $g(x) = \mathrm{E}[Y|X = x]$.

In this section, we address the estimation problem using kernel estimators. We start the section by introducing the concept of a kernel and its properties. We review two types of regression estimators: the Nadaraya-Watson kernel regression estimator discussed in Section 2.2.2.2 and the local polynomial kernel regression estimator reviewed in Section 2.2.2.3.

2.2.2.1 Kernel functions

A kernel $K(z)$ is a symmetric function that satisfies $\int K(z)dz = 1$ and $\int zK(z)dz = 0$. Although it is not necessary, it is typically assumed that $K(z) \geq 0$ for every z in its support and $\int z^2K(z)dz < \infty$. These two assumptions imply that the kernel is also a density. There exist kernels, called higher order kernels, where $\int z^2K(z)dz = 0$. Under smoothness conditions, higher order kernels can achieve faster convergence rates than kernels that are densities (see section 2.8 in Wand and Jones [1994]). However, this comes at a price in the interpretation of the estimator. The kernel estimator of a density f, computed with a higher order kernel, can take negative values for some x in the support. Therefore, the estimator is not a density. Other concern discussed in Wand and Jones [1994] is that in order to decrease the sample error, higher order kernels require larger sample sizes. In this thesis we work with kernels that are also densities.

Kernels are commonly used to estimate curves under minimal assumptions. Common statistical problems where kernels are applied are density estimation and regression problems.

The latter is of particular interest for this thesis. In the next section we begin by reviewing the simplest kernel regression estimator, the Nadaraya-Watson kernel regression estimator. We then review a more sophisticated kernel regression estimator of which the Nadaraya-Watson is a special case.

2.2.2.2 Nadaraya-Watson kernel regression

Nadaraya [1964] and Watson [1964] developed independently a consistent estimator for the conditional expectation $g(x) = \mathrm{E}[Y|X = x]$. Let (X_i, Y_i) be a collection of n independent and identically distributed points. The Nadaraya-Watson kernel regression estimator is defined as

$$\widehat{g}_{n,h}(x) = \frac{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right) Y_i}{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)}.$$

Therefore,

$$\widehat{\eta}(x) \quad = \quad \Psi\left(\widehat{g}_{n,h}(x)\right)$$

where Ψ is the canonical link.

The kernel K in $\widehat{g}_{n,h}(x)$ is a density and thus satisfies all the properties discussed in the previous section. The parameter h is called the bandwidth and this controls the smoothing of the estimator.

Figure 1.1 shows three Nadaraya-Watson kernel estimators with a different bandwidth each. The top right figure shows the estimator with a bandwidth that is too small. The lower right plot shows the estimator with a bandwidth that is too large. It is thus important to select the appropriate bandwidth for any kernel estimator. Wand and Jones [1994] and Marron and Nolan [1988] have discussed not only the bandwidth selection problem, but also the effect of the choice of the kernel in the estimator. Marron and Nolan [1988] used rescaled kernels to deconfound the effect from the kernel and from the bandwidth. Wand and Jones [1994, Section 2.7] compared the efficiency of five popular kernels used in density estimation.

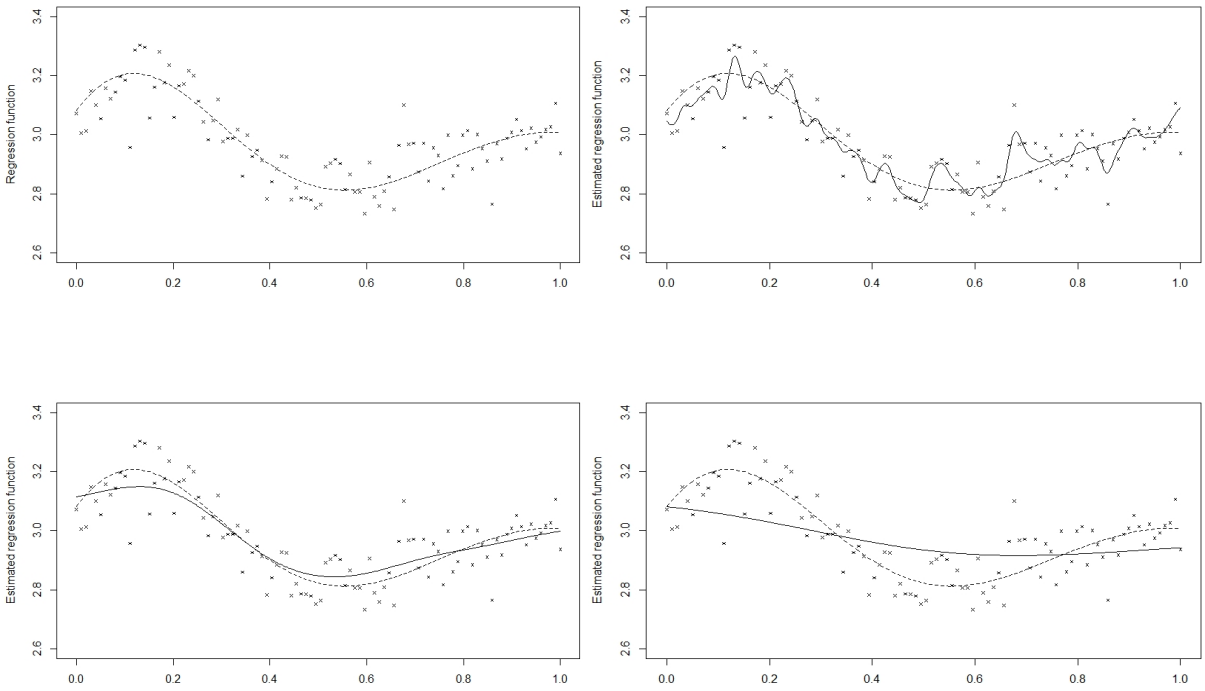

Figure 1.1: Three Nadaraya-Watson kernel estimators for three different bandwidths. The top left plot shows a sample of 100 observations and the true regression function in a dotted line. All other plots show the Nadaraya-Watson kernel estimator in a solid line and the true function in a dotted line. For the top right plot a bandwidth (h) that is too small is selected ($h = .01$). The lower right plot shows the kernel estimator with a bandwidth that is too large ($h = .25$). Finally, the lower left plot is the kernel estimator with an appropriate bandwidth ($h = .09$). (repeated from page 4)

Let f be the true density, then the efficiency of the kernel K, compared to an alternative kernel K^* is the ratio of the sample size needed in order to achieve the same risk (the AMISE) using K^* than using K. We replicate here Table 2.1 in Wand and Jones [1994].

From Table 2.1, an estimator using the Epanechnikov kernel needs 93% of the data to achieve the same performance compared to an estimator with a uniform kernel. Therefore, the choice of the kernel function does not affect significantly the kernel estimator. On the other hand, the bandwidth has a larger effect on the shape of $\widehat{g}_{n,h}$. For a large (small) bandwidth, the estimator is more (less) smooth. The bandwidth also affects the bias and

Kernel	Efficiency of the kernel
Epanechnikov	1.000
Biweight	0.994
Triweight	0.987
Normal	0.951
Uniform	0.930

Table 2.2: We compare the efficiency of various kernels K^*. The efficiency of a kernel K, the Epanechnikov, compared to the kernel K^* is the ratio of the sample size needed in order to achieve the same AMISE using K^* than using K.

variance of the estimator. For a large (small) bandwidth, the bias increases (decreases) while the variance decreases (increases). The question of interest is then how to optimally select the bandwidth for a kernel estimator. This topic will be discussed in Section 2.2.2.4.

The next section will review a different type of kernel estimator. We will see that the Nadaraya-Watson is a special case of this larger group of kernel estimators.

2.2.2.3 Local polynomial kernel regression

Let (X_i, Y_i) for $i = 1, \ldots, n$ be a collection of independent observations with identical distribution and $g(x) = \mathrm{E}[Y|X = x]$. Also assume that Y given X has a density in the exponential family. Let Y be the outcome variable and X the covariate. The relationship between Y and X is modelled by

$$\Psi\left(g(x)\right) \quad = \quad \eta(x)$$

for Ψ the canonical link function.

Next, it is common to approximate the unknown function $\eta(x)$ with a polynomial of degree p

$$\begin{aligned} \eta(x) &= \eta(X_i) + \eta'(X_i)(X_i - x) + \frac{\eta''(X_i)}{2!}(X_i - x)^2 + \ldots + \frac{\eta^{(p)}(X_i)}{p!}(X_i - x)^p \\ &= \beta_0 + \beta_1 (X_i - x) + \beta_2 (X_i - x)^2 + \ldots + \beta_p (X_i - x)^p. \quad (2.3) \end{aligned}$$

One can now maximize the log-likelihood as in Section 2.2.1.3. However, it is desirable to give more emphasis to those observations that are closer to the point x. Let $l_i(\mu, y)$ be the log-likelihood for Y conditional on X. A natural solution is to apply kernel weights to the log-likelihood function

$$l(y,g) \quad = \quad \sum_{i=1}^{n} l_i\left(g(x), y\right) \frac{1}{h} K\left(\frac{X_i - x}{h}\right).$$

We express the log-likehood as a function of the mean and the random variable Y to be consistent with the notation in Fan et al. [1995]. The authors in this paper addressed the problem of local polynomial kernel regression not only for members of the exponential family. Even if the likelihood function is not known, as long as the variance is a known function of the mean $V(g)$, the methodology of the authors explained here holds. The only modification is that the log-likelihood gets replaced by the quasi-likelihood function.

Substituting $\Psi\left(g(x)\right) = \eta(x)$ in the log-likelihood and using the Taylor approximation in (2.3),

$$\begin{aligned} l(y,g) \quad &= \quad \sum_{i=1}^{n} l_i\left(\Psi^{-1}\left(\eta(x)\right), y\right) \frac{1}{h} K\left(\frac{X_i - x}{h}\right) \\ &= \quad \sum_{i=1}^{n} l_i\left(\Psi^{-1}\left(\beta_0 + \beta_1(X_i - x) + \ldots + \beta_p(X_i - x)^p\right), y\right) \frac{1}{h} K\left(\frac{X_i - x}{h}\right). \quad (2.4) \end{aligned}$$

To compute the local polynomial kernel estimate for $\eta(x)$ one maximizes the log-likelihood above on $\beta = (\beta_0, \beta_1, \ldots, \beta_n)$. The estimate for $\eta(x)$ is

$$\widehat{\eta}_{\mathrm{LP},h}(x) \quad = \quad \widehat{\beta_0}$$

where $\widehat{\beta} = \left(\widehat{\beta_0}, \widehat{\beta_1}, \ldots, \widehat{\beta_n}\right)$ is the maximizer of the weighted log-likelihood.

Heuristically, the local polynomial methodology locally fits a polynomial around x. The bandwidth h controls how much an observation contributes to the estimate of η at the point x. A small h implies that less observations will give a weight to the estimator, therefore the estimate will have more variability. For a larger h, the estimate will have less variability, but

will be less accurate (i.e. more biased).

Another important remark is that for the special case when $p = 0$, there is a closed form solution to the estimate of the conditional expectation $g(x)$,

$$\begin{aligned}\widehat{g}_{\mathrm{LP},h} &= \Psi^{-1}\left(\widehat{\eta}_{\mathrm{LP},h}(x; p=0)\right) \\ &= \frac{\frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right) Y_i}{\frac{1}{nh}\sum_{j=1}^{n} K\left(\frac{x-X_j}{h}\right)}\end{aligned}$$

which is the Nadaraya-Watson (or local-constant) kernel regression estimate. We discussed this estimate in Section 2.2.2.2.

Maximizing the weighted log-likelihood not only gives an estimate for the regression function $\eta(x)$, but also for its derivatives. Let $\eta^{(r)}(x)$ be the r^{th} derivative of the regression function. Ruppert et al. [1995] study a local polynomial kernel estimator for the partial derivative given by,

$$\widehat{\eta^{(r)}}(x) = \widehat{\beta}_r r!.$$

There is an obvious advantage in using local polynomial kernel estimators compared to the Nadaraya-Watson estimator. Local polynomial kernel estimators are in general more flexible as the degree increases. Although, as discussed in Fan et al. [1995], a higher degree polynomial increases the variability of $\widehat{\beta}$. In addition, the Nadaraya-Watson estimator suffers from a higher bias on the boundary (we discuss this further in Section 2.2.2.4 where we review measures of performance).

2.2.2.4 Bandwidth selection

In this section, we address the question of bandwidth selection in the context of regression. Wand and Jones [1994] discuss that "unimodal densities perform about the same as each other when used as a kernel". That is when using a kernel estimator, the choice of the kernel density will not have a significant impact on the estimate as the bandwidth will. It is the

bandwidth that can oversmooth or undersmooth important features on the true regression function and therefore, it should be chosen with care.

Let $\widehat{g}_{n,h}(x)$ be the kernel regression estimator for the function g at a point x. The loss function that is most commonly used in practice is the L_2 distance

$$\int \left(\widehat{g}_{n,h}(x) - g(x)\right)^2 dx$$

The average of this integral is known as the risk for the estimator $\widehat{g}_{n,h}$ and is referred to as the mean integrated squared error (MISE). The expectation of the loss function is taken with respect to the data

$$\text{MISE}(h) \quad = \quad \text{E}\left[\int \left(\widehat{g}_{n,h}(x) - g(x)\right)^2 dx\right].$$

It is easy to show that the mean squared error (MSE) can be decomposed into the bias and variance

$$\begin{aligned} \text{MSE}(h) &= \text{E}\left[\left(\widehat{g}_{n,h}(x) - g(x)\right)^2\right] \\ &= \left(\text{E}\left[\widehat{g}_{n,h}(x)\right] - g(x)\right)^2 + \text{Var}\left(\widehat{g}_{n,h}(x)\right). \end{aligned}$$

Therefore, the MISE can be decomposed respectively into

$$\begin{aligned} \text{MISE}(h) &= \int MSE(x;h)dx \\ &= \int \left(\text{E}\left[\widehat{g}_{n,h}(x)\right] - g(x)\right)^2 + \text{Var}\left(\widehat{g}_{n,h}(x)\right) dx \end{aligned}$$

Wasserman [2006] shows, using Taylor expansion and the decomposition of the MSE into bias and variance, that the asymptotic mean integrated squared error (AMISE) for the

Nadaraya-Watson kernel estimator introduced in Section 2.2.2.2 is given by

$$\begin{aligned}\text{AMISE}(h) &= \frac{h^4}{4}\left(\int s^2 K(s)ds\right)^2 \int \left(g''(x) + 2g'(x)\frac{f'(x)}{f(x)}\right)^2 dx \\ &+ \sigma_Y^2 \frac{\int K^2(s)ds}{nh}\int \frac{1}{f(x)}dx\end{aligned}$$

which is easier to minimize. The optimal bandwidth with respect to the AMISE is thus

$$h_{\text{AMISE}} = \left(\frac{\sigma_Y^2 \int K^2(s)ds \int \frac{1}{f(x)}dx}{\left(\int s^2K^2(s)ds\right)^2 \int \left(g''(x) + 2g'(x)\frac{f'(x)}{f(x)}\right)^2 dx}\right)^{1/5} n^{-1/5}.$$

Consider now the local polynomial kernel estimator introduced in Section 2.2.2.3. Further consider the special case of a polynomial of degree one ($p = 1$) and a response variable with density in the exponential family. For a fixed point x, $\widehat{\eta}_h(x;1)$ is the local linear kernel estimator. Fan et al. [1995] derived the AMISE for this estimator

$$\begin{aligned}\text{AMISE}\left(\widehat{\eta}_h(x;1)\right) &= h^4\left(\int z^2 K_1(z)dz\right)^2 \int \left(\frac{\eta^{(2)}(x)}{2}\right)^2 f(x)w(x)dx \\ &\quad + \frac{1}{nh}\int \sigma_1^2(x;K)f(x)w(x)dx,\end{aligned}$$

with minimum at

$$h_{\text{LP}} = \left(\frac{\int \sigma_1^2(x;K)f(x)w(x)dx}{\left(\int z^2K_1(z)dz\right)^2 \int \eta^{(2)}(x)^2 f(x)w(x)dx}\right)^{1/5} n^{-1/5}.$$

Here,

$w(x)$ is any weight function.

$\mathbf{N}$ is a 2×2 matrix with entries $n_{i,j} = \int z^{i+j-2}K(z)dz$.

$\mathbf{M}(z)$ is a 2×2 matrix, identical to $\mathbf{N}$, but with the 1^{st} column replaced by $(1, z)^T$.

$K_1(z)$ is the second order kernel defined as $K_1(z) = \frac{|\mathbf{M}(z)|}{|\mathbf{N}|}K(z)$

$\sigma_1^2(x;K)$, the asymptotic variance of $\widehat{\eta}_h(x;1)$, is given by

$$\sigma_1^2(x;K) \quad = \quad \sigma_{Y|X}^2 \frac{\Psi'\left(g(x)\right)}{f(x)} \int K_1(z)^2 dz.$$

Note that both bandwidths depend on quantities that in practice are unknown. Recall that for density estimation, the l-stage bandwidth selector is a plug-in estimator for the optimal bandwidth with respect to the AMISE criterion. In this bandwidth selection techinique, we substitute the unknown functionals in the optimal bandwidth h_{opt} for kernel estimators that depend on their respective optimal bandwidths. These, at the same time, depend on further unknown functionals that we can estimate using kernels that will, once more, depend on their respective optimal bandwidths. This is iteratively repeated until the l^{th} kernel functional estimator is a simple, ready to use function that does not depend on further bandwidths. Then the method is re-worked backwards to yield the plug-in bandwidth estimator $\widehat{h}_{opt}$. Ruppert et al. [1995] proposed a plug-in estimator for the optimal bandwidth when the outcome is Gaussian and the regression function $g(x)$ is estimated with a local polynomial kernel estimator. However, up to our best knowledge, an approach of this type has not been studied for the more general regression problem where the outcome, conditional on the covariate, has a density in the exponential family.

The idea to use the weighted mean integrated squared error (WMISE), discussed for local polynomial kernel estimators, has also been studied in Signorini and Jones [2004] and Hardle and Marron [1985]. It is worth to note that L_2 is not the only metric that can be used as a loss function. In fact, any L_p distance measure is a valid loss function but is the mathematical simplicity of L_2 (compared to L_p) that makes it widely used.

Other alternatives have been studied to compute a practical bandwidth selector. For the regression problem where the outcome is Gaussian, various bandwidth selection criteria have been proposed as asymptotically equivalent to the MISE. Thus, minimizing these is equivalent to minimize the MISE as the sample size increases. These criteria are: the AIC[Akaike, 1974], the generalized cross-validation [Craven and Wahba, 1975], the finite predictor error [Akaike,

1970], the Shibata selector [Shibata, 1981] and the T selector in Rice [1984]. However, Härdle and Marron [1985] pointed out that these are asymptotically equivalent to MISE (or WMISE) only under specific assumptions on the density f (f is constant on an interval (a, b)) and on the conditional variance and the weight function (for the WMISE criteria) that in practice are typically not satisfied. These bandwidth selectors do not apply to the more general regression problem discussed in this thesis therefore we do not consider them.

Next we discuss the cross-validation criterion [Stone, 1974]. The MISE as a function of h has a complex form as pointed out by Wand and Jones [1994, Chapter 2.5] and Härdle [1986]. To make it easier to work with, a solution is to use a discrete version of the MISE that is mathematically simpler. Härdle [1986] showed that the discretized version of MISE

$$D_{n,\mathrm{MISE}}(h) \quad = \quad \frac{1}{n}\sum_{j=1}^{n}\left[\widehat{g}_{n,h}(x) - g(x)\right]^2,$$

converges uniformly for a sequence of bandwidths h_n to the MISE with probability one. Based on this result, minimizing the cross-validation criterion

$$CV(h) \quad = \quad \frac{1}{n}\sum_{j=1}^{n}\left[\widehat{g}_{n,h}^{(-j)}(x_j) - g(x_j)\right]^2$$

yields the optimal bandwidth from MISE. Other names for this criteria are the "leave-one-out" cross-validation criteria and the least squares cross-validation. We will refer to it as the least squares cross-validation (LSCV) to match the notation in kernel density estimation.

Note that the leave one out estimator

$$\widehat{g}_{n,h}^{(-j)}(X_j) \quad = \quad \sum_{i\neq j}\frac{K\left(\frac{X_j - X_i}{h}\right)Y_i}{\sum_{i=1}^{n}K\left(\frac{X_j - X_i}{h}\right)}$$

is used instead of $\widehat{g}_{n,h}(x_j)$ to avoid under-smoothing (over-fitting) the regression function. However, the LSCV criteria has a drawback. Wand and Jones [1994] point out that this function can have multiple local minima and care needs to be taken in finding the optimal bandwidth. This drawback makes it desirable to investigate alternative bandwidth selectors.

In a regression problem, if the LSCV bandwidth selector requires great computational power and a direct substitution in the optimal bandwidth is not possible, then it is reasonable to use kernel estimators for the unknown quantities and and substitute them in the risk function directly. We follow this approach for bandwidth selection in the context of level sets discussed in Section 3.3.

2.3 Level Sets

In this section, we address the problem of estimating the set $\{x : g(x) \geq \lambda\}$ called the λ-level set. We review and compare two popular approaches for this estimation problem: a direct estimation of the set and a plug-in estimator. Lastly, we discuss a related problem to the λ-level set estimation, called the highest density region. For this problem, we replace the regression function for a density f and the level λ is not fixed or known. Instead, the level depends on a specified area under the density function, which increases the complexity of the problem.Samworth and Wand [2010] and Doss and Weng [2018] studied the highest density problem and proposed a bandwidth selector for the plug-in kernel estimator of the set.

2.3.1 Definition

Consider the function $g(x)$ and a fixed value λ such that $\inf_x g(x) < \lambda < \sup_x g(x)$. The λ-level set of g is the set

$$\mathrm{LS}_\lambda = \{x : g(x) \geq \lambda\}.$$

Two types of level sets are of particular interest in statistics. One is when the function $g(x)$ is a density. The second is when $g(x) = E[Y|X = x]$. In particular, when $g(x)$ comes from a logistic regression model, the level set that results is called the effective dose. In this case, LS_λ is interpreted as the set of covariates such that the conditional probability $g(x)$ reaches a probability not lower than λ.

There exist two main methods for level set estimation. The first are direct methods. These do not require estimation for the function g in the set LS_λ. For this type of estimators a loss or risk function is specified and one optimize such function over all possible sets. The second category consists of indirect methods. Here, the level set LS_λ is estimated by first obtaining an estimate of g and then threshold it.

Before proceeding, we define some notation: S^c denotes the complement of the set S; $S_1 \Delta S_2 = (S_1^c \cap S_2) \cup (S_1 \cap S_2^c)$ is called the symmetric difference set, and μ is the Lebesgue measure or the Hausdorff measure.

2.3.2 Estimate a level set directly

Direct estimation of LS_λ commonly starts with an objective function to be optimized (minimized or maximized) on S, where S is any candidate set. A computational search is then performed among a family of sets $\mathbb{S}$.

Assume $X \in \mathbb{R}^d$ and $Y \in \mathbb{R}$. Cavalier [1997] proposed an estimator

$$\widehat{\mathrm{LS}}_\lambda = \arg\max_S \int_S g(x)dx - \lambda\mu(\mathrm{LS}_\lambda \Delta S)$$

where μ is taken to be either Lebesgue measure or Hausdorff measure. Since g is unknown, Cavalier [1997] assumed that g is such that LS_λ is a star-shaped set, in order to perform the optimization problem required to find $\widehat{\mathrm{LS}}_\lambda$.

Assuming that LS_λ is star-shaped, implies that LS_λ has a very particular form. For simplicity of exposition, consider only the case when $d = 2$. Then, in polar coordinates,

$$\mathrm{LS}_\lambda = \{x = (r, \theta), 0 \leq r < q_\lambda(\theta), 0 \leq \theta < 2\pi\}$$

where $q_\lambda(t)$ is a 2π periodic function, and the first k derivatives exist and satisfy the Hölder condition, see Cavalier [1997]. The main idea in Cavalier's work is to create a partition $\mathcal{P} = (\theta_0, \ldots, \theta_M)$ on $[0, 2\pi]$. On each subinterval $\theta_{l-1} \leq \theta < \theta_l$ for $l = 1, \ldots, M$, the function $q_{\lambda,l}(\theta)$ is approximated with a polynomial of degree k. The coefficients of the polynomial

for each $\theta_{l-1} \leq \theta < \theta_l$ are estimated by maximizing a local version of the objective function described above, which yields a set $\widehat{LS}_{\lambda,l}$ for each subinterval. The final estimate of LS_λ is the intersection of all closed sets containing the union of sets $\widehat{LS}_{\lambda,l}$ for $l = 1, \ldots, M$.

Although the estimate in Cavalier [1997] is minimax, the results are based on assumptions hard to check in practice. Willett and Nowak [2007] also pointed out that this estimator is computationally infeasible.

To alleviate on the computational complexity, Willett and Nowak [2007] proposed to use dyadic trees to speed the search. Dyadic trees "are decision trees that divide the input space by means of axis-orthogonal dyadic splits" [Scott and Nowak, 2006]. The authors also use a penalty parameter to avoid an estimator with non- desirable characteristics. The estimator that they proposed is given by

$$\widehat{LS}_\lambda = \min_S \widehat{R}_n(S) + \psi_n(S)$$

where $\widehat{R}_n(S)$ is an empirical risk measure and ψ_n is a regularization or penalty term. The search is done over a collection $\mathbb{S}$ of all dyadic decision trees with a minimum length.

Next, we review an estimator for the level set where g is a density. Under this setting, Baíllo et al. [2000] studied the estimator

$$\widehat{LS}_\lambda \quad = \quad \bigcup_{i=1}^{n} B\left(X_i, \varepsilon_n\right)$$

where $B\left(X_i, \varepsilon_n\right)$ is the closed ball centred at the i^{th} observation X_i with radius ε_n. The authors proved the characteristics on the sequence of smoothing parameters ε_n to guarantee that $\widehat{LS}_\lambda$ is a consistent estimator. All these estimators are computationally intensive and some require a set of assumptions to alleviate this problem. In the next section, we introduce a simpler alternative estimator for LS_λ called a plug-in estimator.

2.3.3 Plug-in estimators for a level set

A plug-in level set estimator requires an estimate for the function $g(x)$. It is defined as the following set

$$\widehat{\mathrm{LS}}_\lambda = \{x : \widehat{g}_n \geq \lambda\}.$$

We believe a plug-in approach is attractive to practitioners because of its simplicity. Setting up a plug-in estimator in practice has less difficulty, compared to an estimator of the type described in the previous section.

Plug-in estimators for a level set have been widely studied. Mason and Polonik [2009] presented a comprehensive literature review on level set estimation when g is a density. Under this framework, most of the work is centered on the convergence rate of the estimator $\widehat{\mathrm{LS}}_\lambda$. Some contributions include Baíllo et al. [2001] and Baíllo [2003] for $\widehat{\mathrm{LS}}_\lambda$ where $\widehat{g}_n$ is the kernel density estimator. Gayraud and Rousseau [2005] studied the rate of convergence, when $\widehat{\mathrm{LS}}_\lambda$ is the Bayesian posterior estimator. Cuevas et al. [2006] studied consistency and convergence rates for general plug-in level set estimators. That is, g can be assumed to be a density, regression curve, etc. However, this type of estimators have received criticism due to the intermediate step of estimating g. There is a general concern that an estimate of this type can become suboptimal since the majority of tools available to estimate g (traditional kernel methods, splines, and linear smoothers) are created to guarantee a good global fit (see discussion in Willett and Nowak [2007].

In the next section, we review a plug-in estimator in the context of the highest density region (HDR) problem. The HDR is a special type of level set. The main idea of the plug-in estimator for this problem is to estimate g with a kernel and to choose the bandwidth carefully to optimize the estimation of the level set and not of the function g.

2.3.4 Level sets in the context of density estimation

For any fixed $0 < p < 1$ and a density f, the $100(1-p)\%$ highest density region (HDR) is the set

$$\text{HDR}_p \equiv \{x : f(x) \geq \lambda_p\}.$$

This is a special type of level set where the level λ_p depends on f as follows

$$\lambda_p = \inf\left\{\lambda \in (0,\infty) : \int_{-\infty}^{\infty} f(x)\mathbb{1}_{\{f(x)\geq\lambda\}}dx \leq 1-p\right\}.$$

Figure 2.1 shows the 80% HDR for a mixture of Gaussian distributions. Notice that when estimating HDR_p, the level λ_p is not fixed as in the previous section. For the HDR problem, λ_p has to be estimated. Previous research on the HDR estimation problem includes Silverman [1986], Picard and Bar-Hen [2000] and Baíllo [2003]. Wright [1986] studied the related problem of highest posterior density estimation and Hyndman [1996] implemented an algorithm for constructing the HDR. As discussed in the previous section, most of the work in the literature focuses on the convergence rate of estimators of the HDR. Samworth and Wand [2010] studied the HDR problem from a different perspective. Consider the HDR plug-in estimator where $\widehat{f}_h$ is the Nadaraya-Watson kernel density estimator discussed in section 2.2.2.2

$$\widehat{\text{HDR}}_{p,h} = \{x : \widehat{f}_h(x) \geq \widehat{\lambda}_{p,h}\}.$$

The level $\widehat{\lambda}_{p,h}$ is estimated as

$$\widehat{\lambda}_{p,h} = \inf\left\{\lambda \in (0,\infty) : \int_{-\infty}^{\infty} \widehat{f}_h(x)\mathbb{1}_{\{\widehat{f}_h(x)\geq\lambda\}}dx \leq 1-p\right\}.$$

Samworth and Wand [2010] proposed an alternative automatic bandwidth selector specific for $\widehat{\text{HDR}}_{p,h}$ and $\widehat{\lambda}_{p,h}$. To our best knowledge, in the context of the HDR, this is the first

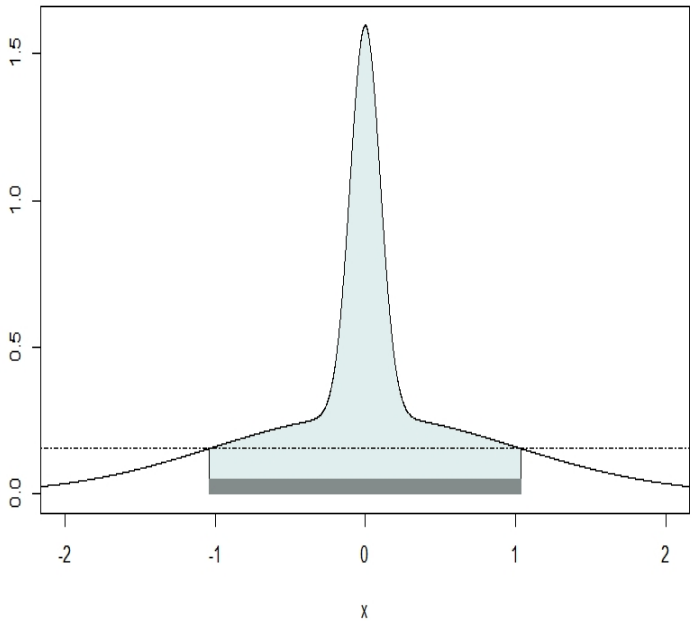

Figure 2.1: We show the 80% HDR for a mixture of Gaussian distributions. This region is shown as a rectangle at the base of the plot.

published work that studies bandwidth selection.

Samworth and Wand [2010] showed that h_{opt}, the optimal bandwidth for the HDR problem, is different from traditional bandwidth selectors such as the AMISE. Consider the following set up. We simulated five random samples from the mixed Gaussian density

$$f(x) \quad = \quad \frac{2}{3}\varphi(x;0,1) + \frac{1}{3}\varphi(x;0,.10), \tag{2.5}$$

where $\varphi(x;m,s)$ is a Gaussian density with mean m and standard deviation s. The upper panel in Figure 2.2 shows five kernel density estimators with bandwidth h_{AMISE} given by,

$$h_{\text{AMISE}} \quad = \quad \left[\frac{\int K^2(s)ds}{\left(\int s^2K(s)ds\right)^2 \int \left(f''(x)\right)^2 dx}\right] n^{-1/5}.$$

To estimate the sharp peak accurately on the density, the bandwidth has to be small enough to avoid over-smoothing. This creates wiggles in regions where the true density is smooth. This behaviour becomes a problem when estimating HDR_p for which perfect estimation of the peak is of lesser interest if, for example, p is sufficiently large or λ_p is sufficiently small.

Recall that $A \Delta B$ denotes the symmetric difference between set A and set B. Samworth and Wand [2010] proposed the loss function

$$\mu_f \left(\text{HDR}_p \Delta \widehat{\text{HDR}}_{p,h} \right) \quad = \quad \int_{\text{HDR}_p \Delta \widehat{\text{HDR}}_{p,h}} f(x) dx.$$

Two important characteristics of this loss function are that it penalizes the misclassification of a point x through the symmetric difference set and it penalizes the likelihood of this point to be observed through the density function. These characteristics make the loss function more appropriate for the context of HDR estimation because only values of x related to the HDR or its estimator contribute to the integral. In contrast to the L_2 norm in MISE where every point on the support of $f(x)$ contributes to the loss function.

Samworth and Wand [2010, Theorem 1] derived an asymptotic approximation of the risk above. The optimal bandwidth for $\widehat{\text{HDR}}_{p,h}$ is thus the bandwidth that minimizes this asymptotic approximation. Assume there exist points x_j such that $f(x_j) = \lambda_p$ for $j = 1, \ldots, 2r$ and a positive integer r. Let φ and ϕ be the Gaussian density, and cumulative distribution function. The asymptotic risk can be shown to be approximately

$$\text{E} \left[\mu_f \left(\text{HDR}_p \Delta \widehat{\text{HDR}}_{p,h} \right) \right] \quad \simeq \quad \frac{1}{n^{2/5}} \sum_{j=1}^{2r} \left[\frac{B_{1,j}}{c^{1/2}} \varphi \left(B_{2,j} c^{5/2} \right) + B_{3,j} c^2 \left\{ 2\phi \left(B_{2,j} c^{5/2} \right) - 1 \right\} \right].$$

The arguments $B_{1,j}, B_{2,j}$ and $B_{3,j}$ are functions that depend on $f(x_j)$ and the derivatives $f'(x_j)$ and $f''(x_j)$. From Samworth and Wand [2010, Corollary 2], the bandwidth that minimizes the asymptotic risk approximation is

$$h_{\text{opt}} \quad = \quad c_{\text{opt}} n^{-1/5}.$$

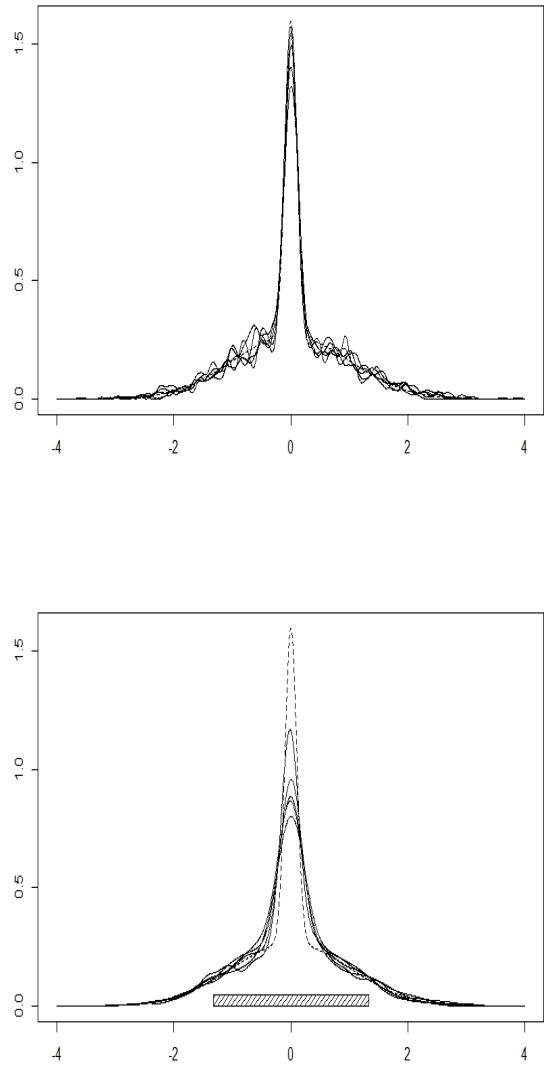

Figure 2.2: Five samples of 1000 observations were generated from a mixed Gaussian density. The true density is shown on both panels with a dashed curve. The panel on top shows five kernel density estimates where the bandwidth was selected by minimizing the AMISE. The lower panel shows five kernel density estimates where the bandwidth selected minimizes the asymptotic risk approximation for the $\text{HDR}_{.20}$ estimation problem. The true $\text{HDR}_{.20}$ is shown.

The constant c_{opt} is independent of n. However, it depends on $f(x_j), f'(x_j)$ and $f''(x_j)$. Notably, h_{opt} is of the same order as the bandwidth that minimizes the AMISE. To compute h_{opt} Samworth and Wand [2010] use further plug-in estimators for any unknown quantities. If c_{opt} would have a closed form, then it would suffice to replace the unknown functions with plug-in estimates. However, because of the complexity of the asymptotic risk, this is not feasible. Therefore, the plug in estimators $\widehat{f}_{h_0}(\widehat{x}_j)$, $\widehat{f}'_{h_1}(\widehat{x}_j)$ and $\widehat{f''}_{h_2}(\widehat{x}_j)$ replace $f(x_j), f'(x_j)$ and $f''(x_j)$ directly in the asymptotic risk approximation to yield a plug-in estimator $\widehat{AR}(c)$ which is then minimized by

$$\widehat{h}_{\text{opt}} \quad = \quad \widehat{c}_{\text{opt}} n^{-1/5}.$$

The quantity $\widehat{c}_{\text{opt}}$ depends on the kernel estimator $\widehat{f}_h(\widehat{x}_j)$; as well as the kernel estimators for the derivatives mentioned before. Results from Samworth and Wand [2010, Theorem 3] show that the relative difference between the asymptotic risk $AR(c)$ and $\widehat{AR}(c)$ is bounded in probability. The same is true for $\widehat{h}_{\text{opt}}$ and h_{opt}.

Figure 2.3 compares the performance of the LSCV bandwidth selector against $\widehat{h}_{\text{opt}}$ for the HDR problem. Let f be the density in (2.5). We simulated 250 Monte Carlo samples of 1000 observations each. For a fixed $0 < p < 1$, the HDR estimate when choosing the bandwidth according to the LSCV is the set

$$\widehat{\text{HDR}}_{p,h_{\text{LSCV}}} \quad = \quad \left\{x : \widehat{f}_{h_{\text{LSCV}}}(x) \geq \widehat{\lambda}_{p,h_{\text{LSCV}}}\right\}.$$

Similarly, the HDR estimate for the bandwidth selector in Samworth and Wand [2010] is

$$\widehat{\text{HDR}}_{p,h_{\text{opt}}} \quad = \quad \left\{x : \widehat{f}_{\widehat{h}_{\text{opt}}}(x) \geq \widehat{\lambda}_{p,h_{\text{opt}}}\right\}.$$

Each point in Figure 2.3 represents the errors from both estimates in one particular sample. The x coordinate refers to the error $\mu_f\left(\text{HDR}_p \Delta \widehat{\text{HDR}}_{p,\widehat{h}_{\text{opt}}}\right)$, while the y coordinate refers to $\mu_f\left(\text{HDR}_p \Delta \widehat{\text{HDR}}_{p,h_{\text{LSCV}}}\right)$. From the plot we observe that, for small values of p, h_{opt} offers

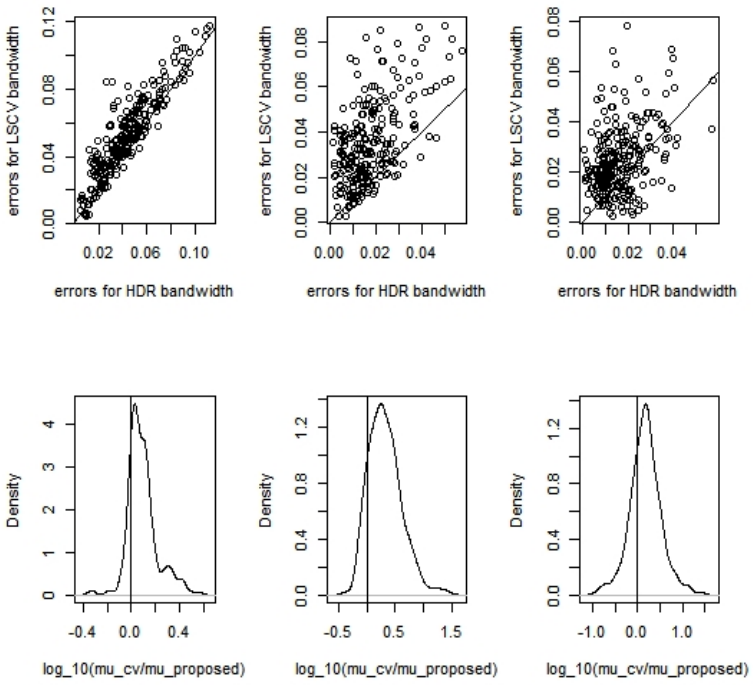

Figure 2.3: Pairwise comparisons between the errors $\mu_f\left(\text{HDR}_p\Delta\widehat{\text{HDR}}_{p,\widehat{h}_{\text{opt}}}\right)$ and $\mu_f\left(\text{HDR}_p\Delta\widehat{\text{HDR}}_{p,h_{\text{LSCV}}}\right)$ for 250 Monte Carlo samples of 1,000 observations each. We assume that the true density is the mixed gaussian density $f(x) = \frac{2}{3}\varphi(x;0,1) + \frac{1}{3}\varphi(x;0,.10)$.

an advantage when compared to the bandwidth h_{LSCV}. When $p = 0.8$, this advantage over the LSCV bandwidth diminishes. This occurs because when $p = 0.8$, estimating the peak of f accurately is important for $\text{HDR}_{0.8}$ and for the mean integrated squared error (the risk function for the LSCV).

Recently, Doss and Weng [2018] generalized the HDR estimation problem in the multivariate setting extending the approach of Samworth and Wand [2010]. Let $X \in \mathbb{R}^d$ be the random variable with distribution f. Under the same loss function as in the univariate case, Doss and Weng [2018] use an asymptotic approximation of the risk function to derive the optimal bandwidth.

For simplicity, Doss and Weng [2018] use a set of simplifying assumptions to present their

theoretical results. The density f is assumed to be unimodal and spherically symmetric. The matrix of bandwidths is assumed $\mathbf{H} = h^2\mathbf{I}$, where $\mathbf{I}$ is the $d \times d$ identity matrix. The goal is then to minimize the asymptotic approximation of the risk with respect to h. From Doss and Weng [2018, Corollary 2.2]

$$h_{\text{opt}} = c_{\text{opt}} n^{-1/(4+d)}$$

The constant c_{opt} is independent of n but, as in the univariate case, depends on the unknown density f. To compute h_{opt} in practice, Doss and Weng [2018] frame the methodology of Samworth and Wand [2010] in the multivariate setting. Doss and Weng [2018] replace f, ∇f and $\nabla^2 f$ in the asymptotic risk approximation with the plug-in kernel estimates $\widehat{f}_{n,\mathbf{H}_0}$, $\widehat{f}_{n,\mathbf{H}_1}$ and $\widehat{f}_{n,\mathbf{H}_2}$ respectively. This yields an estimate for the asymptotic risk ($AR(H)$) that is then minimized with respect to $\mathbf{H}$ using Newton's method. This is an iterative method to find the root of a differentiable function q with values in the real numbers. At the i^{th} iteration, the approximation of the root is

$$x_i \quad = \quad x_{i-1} - \frac{q(x_{i-1})}{q'(x_{i-1})}.$$

For the set of assumptions in Doss and Weng [2018, Corollary 2.2] described previously, Doss and Weng [2018, Corollary 3.2] show that the relative distance between $\widehat{h}_{\text{opt}}$, the minimizer of $\widehat{AR}(h)$ and h_{opt} is bounded in probability at a rate $O_p\left(n^{2/(d+8)}\right)$.

A related problem to the HDR is the density level set (DLS) estimation. For this set, the level λ_p is exchanged for a fixed arbitrary value λ where $\inf_{x\in\mathbb{R}^d} f(x) < \lambda < \sup_{x\in\mathbb{R}^d} f(x)$. Doss and Weng [2018] study a plug-in kernel estimator for the DLS. This estimator is the set

$$\widehat{\text{DLS}}_{\lambda,h} \quad = \quad \left\{x : \widehat{f}_{n,h}(x) \geq \lambda\right\}.$$

The loss function for the DLS problem is the same function as for the HDR problem

$$\mu_f\left(\mathrm{DLS}_\lambda \Delta \widehat{\mathrm{DLS}}_{\lambda,h}\right) \quad = \quad \int_{\mathrm{DLS}_\lambda \Delta \widehat{\mathrm{DLS}}_{\lambda,h}} f(x)dx.$$

The optimal bandwidth for $\widehat{\mathrm{DLS}}_{\lambda,h}$ is thus the argument that minimizes the asymptotic approximation of the risk function. Results from Doss and Weng [2018, Corollary 2.1] show that the optimal bandwidth rate for DLS coincides with that in HDR. The practical bandwidth selection methodology and results are identical to those for the multivariate HDR.

Chapter 3

PART I: Theoretical and applied section for the regression level set

3.1 Optimal Bandwidth selection

In the last section, we reviewed the highest density region (HDR) problem. We discussed this is a special type of level set and we reviewed how to select the appropriate bandwidth when HDR_p is estimated with a plug-in kernel estimator.

In this section, we translate the main ideas of the HDR problem to the context of regression. The density f in HDR_p is exchanged for the conditional expectation g where $g(x) = E[Y|X = x]$. In addition, we predefine a fixed level λ. Thus, for the outcome variable Y and the covariate X, we are interested in the following level set

$$LS_\lambda \quad = \quad \{x : g(x) \geq \lambda\}.$$

In this thesis, we study how to choose the appropriate bandwidth when LS_λ is estimated with a plug-in kernel estimator.

To our knowledge, there is no published work addressing bandwidth selection for a plug-in kernel estimator for LS_λ in the context of regression. Published work on level set estimation in the context of regression includes Cavalier [1997], Willett and Nowak [2007] and Cuevas

et al. [2006]. These contributions mainly study the convergence rates for estimators of LS_λ. Cavalier [1997] and Willett and Nowak [2007] studied optimal rates of convergence for direct estimators of LS_λ. Cuevas et al. [2006] studied consistency and rates of convergence for plug-in estimators of LS_λ. More recently, Laloë and Servien [2013] studied a kernel plug-in estimator for LS_λ. However, their approach is different from the approach in this thesis. Laloë and Servien [2013] studied the conditions that guarantee the consistency of their estimator and established its convergence rate. The bandwidth selection problem was not addressed on their paper. Laloë and Servien [2013, page 5] considered that research on bandwidth selection is "a remaining and crucial problem" for this type of estimator.

3.1.1 Introduction

Kernel methods have been researched extensively and are a popular tool in practice. As pointed out in Section 2.2.2.4, a kernel estimator depends heavily on the choice of the bandwidth. Typically, a larger bandwidth will decrease the variance of the estimator at the expense of increasing the bias. The situation is reversed for a smaller bandwidth. In similar spirit to Samworth and Wand [2010] and Doss and Weng [2018], we focus on the Nadaraya-Watson kernel regression estimator introduced in Section 2.2.2.2

$$\widehat{\mathrm{g}}_{\mathrm{n,h}} = \quad = \quad \frac{\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right) Y_i}{\sum_{j=1}^{n} K\left(\frac{x-X_j}{h}\right)}.$$

Our proposed plug-in estimator for a regression λ-level set is thus

$$\widehat{\mathrm{LS}}_\lambda = \{x : \widehat{g}_{n,h}(x) \geq \lambda\}. \tag{3.1}$$

The main goal for this thesis is then how to optimally select the bandwidth for $\widehat{\mathrm{LS}}_\lambda$. Traditional bandwidth selection methods, as those discussed in Section 2.2.2.4 become inefficient for a level set estimation problem that is by nature local (see Willett and Nowak [2007] and Scott and Davenport [2007]). A loss function such as the mean integrated squared error, although mathematically tractable, it chooses the bandwidth that minimizes a global error.

The same is true if the cross-validation approach is preferred.

Bandwidth selection is a problem that has also been encountered in the context of the HDR estimation. Samworth and Wand [2010] and Doss and Weng [2018] show in their simulation study that choosing the bandwidth according to a more appropriate local loss function improved, in general, the highest density region estimation.

3.1.2 The loss function

We start by defining a loss function, similar to Samworth and Wand [2010], that considers the local nature of the level set problem. Let $q(x)$ be a non-negative function that depends on a covariate. We propose the loss function

$$\mu_q(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda) \quad = \quad \int_{\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda} q(x)dx, \tag{3.2}$$

where $\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda$ is the symmetric difference set between the true level set and our plug-in estimator. This set contains all the points x that are elements of either LS_λ or $\widehat{\mathrm{LS}}_\lambda$, but not both. Therefore, we can express the loss function in the following equivalent way

$$\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right) \quad = \quad \int_{\Omega_X} q(x)|\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx.$$

The optimal bandwidth for our estimator $\widehat{\mathrm{LS}}_\lambda$ is the bandwidth that minimizes an asymptotic approximation of the risk $E\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right]$. We start by stating the assumptions we make to derive our asymptotic approximation of the risk.

3.2 Asymptotic Risk Analysis

3.2.1 Assumptions

The following are the set of conditions on the conditional expectation, the kernel (K), the bandwidth (B), the covariate's density (F) and the function q that we need to derive the asymptotic approximation of the risk. The conditions in (G) are in the same spirit as those

found in Samworth and Wand [2010], Doss and Weng [2018]. Let $g = E[Y|X = x]$ with $g_2(x) = E[Y^2|X = x]$. In order to prove our results, we make the following assumptions.

(F) The support of the density f, C_X is compact. Furthermore, $\inf_{x \in C_X} f(x) \geq b_f > 0$.

(G) For an arbitrary λ, we assume there exist finitely many points $x_1 < \ldots < x_m$ such that $g(x_j) = \lambda$ and $g'(x_j) \neq 0$ and that these x_j (for $j = 1, ..., m$) all lie in the interior of C_X. To simplify the exposition in the proof, we assume that there are an even number of points. That is $m = 2r$ with $g'(x_{2j-1}) > 0$ and $g'(x_{2j}) < 0$. We also define $x_0 = \inf\{x \in C_X\}$ and $x_{2r+1} = \sup\{x \in C_X\}$.

(K) The kernel K is a non-negative, symmetric density and, defining $v_{k,m} = \int |s^m| K^k(s)ds$, satisfies $v_{1,3}, v_{2,2}, v_{3,4}, v_{4,0}$ are all finite.

(B) We assume that the bandwidth $h = h_n \to 0$ as $n \to \infty$. In addition, it is such that nh^5 is at most $O(1)$, and that $n^3 h^{11} \log^{-8} n \to \infty$.

(S) We make the following smoothness assumptions on the underlying functions.

local: Let $\widetilde{B} = \cup_{j=1}^{m} B_\eta(x_j)$ where $B_\eta(x)$ denotes a ball of size η around x. There exists an $\eta > 0$ such that the function q is $C_1(\widetilde{B})$, while f, g_2 are $C_1(\widetilde{B})$ and g is $C_2(\widetilde{B})$.

global: Both f and g are continuous on C_X, and q is an integrable function on C_X.

moment: Define the function $m_k(x) = E[(Y - \lambda)^k | X = x] f(x)$ and assume that there exists an $M_k(x)$ such that for $k = 2, 4$

$$\lim_{h \to 0} \sup_{s \in (x - C_X)/h} m_k(x - hs) \quad \leq \quad M_k(x)$$

with $\int_{C_X} M_2^2(x) q(x) dx < \infty$ and $\int_{C_X} M_4(x) q(x) dx < \infty$.

We wanted assumption (S) to be general in order to accommodate for common loss functions in level set estimation problems e.g. the excess mass loss function assumed in Willett and Nowak [2007]. Assumption (K), on the other hand, lists standard kernel properties. The

remaining assumptions, for the most part, are similar to those of Samworth and Wand [2010], Doss and Weng [2018]. In particular, we note that the smoothness assumptions (S1) on the functions g_2, g, f are all one derivative stronger than those needed in the statement of the theorem. This was also required in Samworth and Wand [2010], Doss and Weng [2018]. This assumption, as well as assumption (B) are necessary in order to achieve the correct integrated error in the final expression.

Using assumption (G), we can specify the regions where the loss function takes positive values

$$\begin{aligned}
\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right) &= \int_{\Omega_X} q(x)|\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx \\
&= \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)|\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx \\
&\quad + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)|\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx \\
&= \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)|\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\}} - 0|dx + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)|\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\}} - 1|dx.
\end{aligned}$$

It thus follows that

$$\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right) = \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\}}dx + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)\mathbb{1}_{\{\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)<\lambda\}}dx,$$

The goal is then to select the bandwidth h that minimizes the following risk function

$$\begin{aligned}
\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] &= \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)P\big(\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)\geq\lambda\big)dx \\
&\quad + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)P\big(\widehat{\mathrm{g}}_{\mathrm{n,h}}(x)<\lambda\big)dx. \qquad (3.3)
\end{aligned}$$

3.2.2 Risk computation and asymptotic risk approximation

The main result of our thesis derives the asymptotic risk expansion of the level set estimator $\widehat{\mathrm{LS}}_\lambda$. Let ϕ be the probability density of a standard normal and Φ its corresponding cumulative distribution function.

Theorem 1. *Suppose that assumptions (B), (F), (G), (K), and (S) hold. Then*

$$\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] = \sum_{j=1}^{m} q(x_j) B_{1,j}(x_j) \left\{ 2\,\frac{\phi(\sqrt{nh^5}B_{2,j})}{\sqrt{nh}} + h^2 B_{2,j}\left[2\Phi(\sqrt{nh^5}B_{2,j}) - 1\right]\right\} + o((nh)^{-1/2} + h^2)$$

where

$$A(x_j) = -\left\{\frac{g'f}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j), \quad B_{1,j} = \frac{1}{|A(x_j)|}, \quad B_{2,j} = -\sigma_K^2\left\{\frac{g''f/2 + g'f'}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j),$$

where $\sigma_K^2 = \int s^2 K(s)ds$ *and* $\sigma_Y^2(x) = \mathrm{Var}(Y|X = x)$.

See Apendix A.1 for Theorem 1 proof. The proof is handled similarly to Samworth and Wand [2010], Doss and Weng [2018]. The asymptotic risk expansion results from a careful sequence of Taylor series type of arguments. The main idea is to first show that the integral in the risk function is dominated by a neighbourhood of each point x_j. Second, in the neighbourhood of each point x_j, the formula is derived by applying the central limit theorem, which allows us to replace the probabilities with Gaussian distribution functions and the final result follows from integration by parts. Our result obtained the same rate of convergence as in Samworth and Wand [2010, Theorem 1]. The leading terms of the sum correspond to the order of the variance and bias as in Samworth and Wand [2010, Theorem 1] and Doss and Weng [2018, Theorem 1] for $d = 1$.

Let $c = hn^{1/5}$, for a sequence of bandwidths of order $O\left(n^{-1/5}\right)$ we can write

$$\lim_{n\to\infty} n^{2/5}\,\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] = \sum_{j=1}^{m} q(x_j)B_{1,j}(x_j) \left\{2\,\frac{\phi(c^{5/2}B_{2,j})}{c^{1/2}} + c^2 B_{2,j}\left[2\Phi(c^{5/2}B_{2,j}) - 1\right]\right\}.$$

It is easy to see that if $(nh^5) \to c_1$ for a constant $0 < c_1 < \infty$, then $\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] = O\left(n^{-2/5}\right)$. This is a faster rate compared to the rate obtained in Laloë and Servien [2013] when $d = 1$. It has to be noted, however, that the approach in Laloë and Servien [2013] is focused on the rate of convergence of the kernel estimator and not on bandwidth selection. The assumptions on the bandwidth are also different from those in this paper. In particular, the authors assumed that $nh^5 = o\left(\frac{1}{\log(n)}\right)$ and, for their simulation study, the practical bandwidth is chosen via cross-validation.

In addition, we observe that the equation above is a continuous function for $c \in (0, \infty)$. Assuming that there is at least one j such that $B_{2j} \neq 0$, it can be shown in Corollary 2 that there exists a unique minimizer for the function. A situation where B_{2j} would be zero for every j is if the true function $g(x)$ is linear and the covariate is uniformly distributed. When this occurs, the second term in the sum is zero and the numerator in the first term is constant on c. It thus follow that under this scenario, the minimizer of the risk function does not exist. It has been discussed (see Gasser and Engel [1990]) that when g is assumed linear, the Nadaraya-Watson estimator is not the best estimator to use. The Nadaraya-Watson kernel estimator tends to deviate from a linear trend as seen in Figure 1 in Gasser and Engel [1990], regardless of the density of the covariate.

Corollary 2. *Assume the conditions in Theorem 1 hold. Furthermore, assume that* $nh^5 \to c$ *for* $0 < c < \infty$. *Then, there exists a unique* $c_{opt} \in (0, \infty)$ *depending on* g, f *and* K; *but not on* n, *such that* $h_{opt} = \arg\min_{h\in(0,\infty)} E[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)]$ *and it satisfies*

$$h_{opt} = c_{opt} n^{-1/5}$$

From Corollary 2 we observe that the level set optimal bandwidth has the same rate as the bandwidth that minimizes the AMISE. Unfortunately, our optimal bandwidth has no closed form. Therefore, a practical bandwidth selector such as the *l-stage bandwidth selector* is not feasible. Instead, we propose to approximate the asymptotic risk and minimize this instead. The result from Theorem 1 gives an asymptotic approximation of the risk (AR). Again, let $c = hn^{1/5}$, then

$$AR(c) = n^{-2/5} \sum_{j=1}^{m} q(x_j) B_{1,j}(x_j) \left\{ 2 \frac{\phi(c^{5/2} B_{2,j})}{c^{1/2}} + c^2 B_{2,j} \left[2\Phi(c^{5/2} B_{2,j}) - 1 \right] \right\}. \quad (3.4)$$

In the section that follows we introduce our practical bandwidth selector and describe our algorithm to obtain it.

3.3 Practical computation for the asymptotic risk

From our results in Theorem 1 and Corollary 2 we show that there is a minimizer to the risk proposed at the beginning of this chapter. Different from other bandwidth selectors with loss function in L_2, the local function here proposed does not yield a close solution to the optimal bandwidth. Therefore, we cannot use a direct plug-in rule into the formulae for the optimal bandwidth as is common for the AMISE (see Wand and Jones [1994]). A natural solution is to replace the unknown functions in the asymptotic risk with kernel estimators and minimize this instead.

A plug-in estimator for the asymptotic risk

Let $c = n^{1/5}h$, and substitute all unknown functions in $AR(c)$ with kernel estimates. Then the plug-in estimate for the asymptotic risk is

$$\widehat{AR}_n(c) = \frac{1}{n^{2/5}} \sum_{j=1}^{2r} \left\{ \frac{\widehat{B}_{1j} \varphi\left(2c^{5/2} \sigma_K^2 \frac{\widehat{B}_{2j}}{\widehat{B}_{1j}} \right)}{c^{1/2}} + c^2 \sigma_K^2 \widehat{B}_{2j} \left[2\phi\left(2c^{5/2} \sigma_K^2 \frac{\widehat{B}_{2j}}{\widehat{B}_{1j}} \right) - 1 \right] \right\} \quad (3.5)$$

In order to estimate the k^{th} derivative of g, we employ the local polynomial estimator $\widehat{g^{(k)}}_{h_k}(x;p)$ that appears in Fan et al. [1995]. It estimates the k^{th} derivative of g locally, using a polynomial of degree p where $p \geq k$. For simplicity, we work with the cases where $p-k$ is odd. In particular, we restrict the degree of the polynomial to be $p = k+1$.

3.4 A practical bandwidth selector

Recall the asymptotic risk $AR(c)$ is minimized at a constant c_{opt} which depends on q, f, f', g', g'', as well as the points $x_1, \ldots, x_{2r}$ such that $g(x_j) = \lambda$ for $j = 1, \ldots, 2r$. The optimal bandwidth for the plug-in level set kernel estimator proposed in Section 3.2.2 depends on this constant through $h_{opt} = c_{opt} n^{-1/5}$. In this section we describe how to obtain the estimator

$$\widehat{c}_{opt} = \arg\min \widehat{AR}_n(c)$$

where the function $\widehat{AR}_n(c)$ is the asymptotic risk with q, f, f', g', g'' and x_j replaced with kernel estimators.

3.4.1 Algorithm description

Assume an independent and identically distributed sample $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$.

(i) We replace f and f' with kernel estimators $\widehat{f}_{h_{f_0}}$ and $\widehat{f'}_{h_{f_1}}$ described in Chapters 2.2 and 2.12 in Wand and Jones [1994].

(ii) We estimate h_{f_0} and h_{f_1} with the two-stage plug-in bandwidth selector described in Chapter 3.6 in Wand and Jones [1994]. Recall from Section 2.2.2.4 that the direct plug-in rule for bandwidth selection starts with the bandwidth that minimizes the asymptotic mean integrated squared error. This optimal bandwidth depends on unknown functionals that are replaced with kernel estimators. However, these depend on a bandwidth that would have to be computed as well. This methodology creates then a number of levels that is stopped when the last bandwidth is computed by what Fan et al. [1995]

call a "rough and ready" method; a simpler bandwidth that does not depend on further kernel estimators (e.g. the optimal bandwidth assuming that the function is Gaussian). Then the method is re-worked backwards.

(iii) We take the least squares cross validation bandwidth (h_{LSCV}) and estimate g with the Nadaraya-Watson kernel estimator $\widehat{g}_{h_{\text{LSCV}}}$. Thus, the estimators for the points x_j are the points $\widehat{x_j}$ such that $\widehat{g}_{h_{LSCV}}(\widehat{x_j}) = \lambda$. Note that it is possible that $\max_x \widehat{g}_{h_{LSCV}}(x) < \lambda$. When this happens, we reduce the bandwidth h_{LSCV} in 10% intervals until $\max_x \widehat{g}_{h_{\text{LSCV}}}(x) > \lambda$.

(iv) We replace g' and g'' with the local polynomial kernel estimators $\widehat{g'}_{h_{g_1}}$ and $\widehat{g''}_{h_{g_2}}$ studied in Fan et al. [1995] and reviewed in Section 2.2.2.3.

(v) We let $h_{g_1} = h_{g_2}$ to ease the computation complexity in the code. For the first two derivatives of g we use the bandwidth that minimzes the asymptotic MISE for the local polynomial kernel estimator reviewed in Section 2.2.2.3. If Y conditional on X has a Gaussian distribution, the function **locpoly** in **KernSmooth** R-package returns this bandwidth.

(vi) We estimate the variance σ_Y^2 with the sample variance and substitute (i)-(v) in the asymptotic risk to yield $\widehat{AR}(c)$. The estimator for c_{opt} is thus

$$\widehat{c}_{opt} = \arg\min_c AR\left(c; \widehat{x_j}, \widehat{f}_{h_{f_0}}, \widehat{f'}_{h_{f_1}}, \widehat{g'}_{h_{g_1}}, \widehat{g''}_{h_{g_2}}\right)$$

and

$$\widehat{h}_{opt} = \widehat{c}_{opt} n^{-1/5}$$

(vii) The optimal bandwidth $\widehat{h}_{opt}$ in step (vi) yields the plug-in λ-level set kernel estimator

$$\widehat{LS}_\lambda = \{x : \widehat{g}_{n,\widehat{h}_{opt}} \geq \lambda\}$$

Relative rate between the asymptotic risk and its practical estimator

Since the practical bandwidth selector is based on a plug-in estimator of the asymptotic risk, it is necessary to investigate how close this approximation is to the true value and how the minimizer of $\widehat{AR}$ compared to that of AR. We present these results in Theorem 3.

Theorem 3. *Assume the conditions in Theorem 6 hold. In addition, assume*

(S.2) f *is* $C_2(\widetilde{B})$ *and* g *is* $C_3(\widetilde{B})$.

(Q) The function q *is a non-negative function such that* $|\widehat{q}(x) - q(x)|$ *is at most* $O_p\left(n^{-2/9}\right)$ *in a neighbourhood of* x_j.

Let $c = n^{1/5}h$ *and define the asymptotic risk function in terms of* c *as*

$$AR(c) \quad = \quad n^{-2/5} \sum_{j=1}^{m} q(x_j) B_{1,j} \left\{ 2\, \frac{\phi(c^{5/2} B_{2,j}}{c^{1/2}} + c^2 B_{2,j} \left[2\Phi(c^{5/2} B_{2,j}) - 1 \right] \right\}.$$

Also define $\widehat{AR}_n(c)$ *as the asymptotic risk function where* x_j,q,f,f',g' *and* g'' *are replaced with kernel estimators. Then*

$$\left| AR(c) - \widehat{AR}_n(c) \right| = O_p\left(n^{-2/9}\right).$$

and

$$\frac{\widehat{h}_{opt}}{h_{opt}} \quad = \quad 1 + O_p\left(n^{-2/9}\right)$$

The relative difference between the oracle and practical bandwidth presented in Theorem 3 yield the same rate of convergence as that in the HDR problem (see Samworth and Wand [2010, Theorem 3]).

3.5 Level set bandwidth selection against least-squares cross validation bandwidth selection and the local polynomial kernel estimator: A simulation study.

In this section we compare the performance of the practical bandwidth selector described in Section 3.3 with the least-squares cross validation (LSCV) bandwidth selector reviewed in Section 2.2.2.4 and the local polynomial kernel estimator reviewed in Section 2.2.2.3. We divide this section into two parts. For the first part we focus on the regression problem where the outcome (Y) is Gaussian. For the second part we consider the situation where Y is binary. We choose to divide this section because the tools available for each are different and thus our approach will differ.

3.5.1 A Gaussian outcome

Consider the regression model

$$Y = g(x) + \varepsilon \qquad \varepsilon \sim N\left(0, \sigma_Y^2\right)$$

where $g(x) = \mathrm{E}[Y|X = x]$. For our simulation study, we consider three scenarios described in Table 3.1 below. In Figures 3.1,3.4 and 3.7 we show the plot for g in each example and the true level sets considered.

Recall that for a fixed level λ and the conditional expectation g, the corresponding λ-level set is

$$\mathrm{LS}_\lambda = \{x : g(x) \geq \lambda\}.$$

Also, recall from Section 2.2.2.2 that the Nadaraya-Watson kernel regression estimator is

given by

$$\widehat{g}_{n,h}(x) \quad = \quad \frac{\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right)}.$$

Let h_{opt} be the bandwidth that results from the practical bandwidth selector that we proposed in Section 3.3. Thus, our estimator for LS_λ is the plug-in estimator

$$\widehat{LS}_{\lambda,opt} \quad = \quad \{x : \widehat{g}_{n,h_{opt}}(x) \geq \lambda\}$$

where $\widehat{g}_{n,h_{opt}}$ is the Nadaraya-Watson estimator with h_{opt} as its bandwidth. For the alternative estimators, let $\widehat{g}_{n,h_{\mathrm{LSCV}}}$ be the Nadaraya-Watson estimator where h_{LSCV} is the bandwidth that minimizes the LSCV criteria. In addition, let $\widehat{g}_{n,h_{\mathrm{LP}}}$ be the local polynomial kernel es-

Example 1	$g(x) = \frac{2}{3}\frac{1}{\sqrt{2\pi}}\exp^{-\frac{1}{2}x^2} + \frac{10}{3}\frac{1}{\sqrt{2\pi}}\exp^{-50x^2}$ for $x \in [-4,4]$ $\sigma_Y^2 = 0.10$	$\lambda = \begin{cases} 0.156, \\ 0.261, \\ 1.325 \end{cases}$
Example 2	$g(x) = \begin{cases} .7\phi(x;4,.15) + .45\phi(x;6,.5) & \text{for } \ 2 \leq x \leq 8 \\ -\frac{1}{2}(x-8.5)^2 + c_0 & \text{for } \ 8 < x \leq 10 \end{cases}$ $\sigma_Y^2 = 0.01$	$\lambda = \begin{cases} 0.05, \\ 0.10, \\ 1.00, \\ 1.30, \\ 1.40, \\ 1.50 \end{cases}$
Example 3	$g(x) = \sqrt{x(1-x)}\sin\left(\frac{2.1\pi}{x+0.5}\right)$ for $x \in [0.1,1]$ $\sigma_Y^2 = 0.10$	$\lambda = \begin{cases} 0.18, \\ 0.25, \\ 0.30, \\ 0.40 \end{cases}$

Table 3.1: Three different conditional expectation functions that are considered in the simulation study. The constant $c_0 = .7\phi(8;4,.15) + .45\phi(8;6,.5) + \frac{1}{2}.25$ in Example 2 is calculated to guarantee continuity on the function.

timator of degree one. It is given by

$$\widehat{g}_{n,h_{\mathrm{LP}}} \quad = \quad \mathbf{e}^T \left(\mathbf{X}_x^T \mathbf{W} \mathbf{X}_x\right)^{-1} \mathbf{X}_x^T \mathbf{W} \mathbf{Y}$$

where

$\mathbf{X}$ is the design matrix $\begin{bmatrix} 1 & (X_1 - x) \\ \vdots & \vdots \\ 1 & (X_n - x) \end{bmatrix}$

$\mathbf{W}$ is an $n \times n$ diagonal matrix with entries $\frac{1}{h} K\left(\frac{X_1 - x}{h}\right), \ldots, \frac{1}{h} K\left(\frac{X_n - x}{h}\right)$.

$\mathbf{Y}$ is the vector of n observed responses.

$\mathbf{e}$ is the column vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

The bandwidth h_{LP} is the argument that minimizes the asymptotic mean integrated squared error (with parameters $r = 0$ and $p = 1$) described in Section 2.2.2.3. The bandwidth selector for h_{LP} is implemented in the function *dpill*, included in the R-package **KernSmooth**. We use this tool for this part of the study.

The corresponding plug-in kernel estimators for a λ-level set are respectively

$$\begin{aligned} \widehat{LS}_{\lambda,\mathrm{LSCV}} &= \{x : \widehat{g}_{n,h_{\mathrm{LSCV}}}(x) \geq \lambda\}, \\ \widehat{LS}_{\lambda,\mathrm{LP}} &= \{x : \widehat{g}_{n,h_{\mathrm{LP}}}(x) \geq \lambda\}. \end{aligned}$$

For each function $g(x)$ presented in Table 3.1 we conducted a simulation study with small and large sample sizes. In each, 250 Monte Carlo samples are generated with 1000 observations and 100,000 observations respectively. The covariate X is generated from a uniform distribution according to the support of $g(x)$ in Table 3.1.

The estimation error for an estimator $\widehat{LS}_\lambda$ is the loss function studied in Section 3.1.2

$$\mu_q\left(LS_\lambda \triangle \widehat{LS}_\lambda\right) \quad = \quad \int_{LS_\lambda \triangle \widehat{LS}_\lambda} q(x) dx$$

where $A \triangle B = (A \cap B^c) \cup (A^c \cap B)$. For the three scenarios studied, we specify $q(x) = f(x)$, the density of the covariate X. In addition, we use the following notation for each estimator considered

$$\begin{aligned} \mu_{opt} &= \mu_f \left(LS_\lambda \triangle \widehat{LS}_{\lambda,opt} \right) \\ \mu_{LSCV} &= \mu_f \left(LS_\lambda \triangle \widehat{LS}_{\lambda,\text{LSCV}} \right) \\ \mu_{LP} &= \mu_f \left(LS_\lambda \triangle \widehat{LS}_{\lambda,\text{LP}} \right). \end{aligned}$$

For each scenario we consider, we visually compare the errors from each estimator and include Wilcoxon tests. Throughout our simulation study we assume a 5% significance level.

The function on example one serves as a comparison to the related work of Samworth and Wand [2010]; Doss and Weng [2018]. This is a function that has a sharp peak, but it is smooth otherwise. Next, we take an example from toxicology. Calabrese and Baldwin [2002] presented common dose-response curves found in studies with different chemical agents and experimental models. For example two, we considered $g(x)$ to be a slightly exaggerated version of the multi-modal curve in Calabrese and Baldwin [2002, Figure 2(j)]. We allowed the slope of g to change at different rate in each mode on the function. Lastly, for example three we consider a truncated version of the doppler function. The doppler function is a common example for nonparametric regression, see [Wasserman, 2006].

Example one

We consider the function

$$g(x) = \frac{2}{3}\frac{1}{\sqrt{2\pi}} \exp\left(-0.50x^2\right) + \frac{10}{3}\frac{1}{\sqrt{2\pi}} \exp\left(-50x^2\right).$$

This is a function that is smooth except for a sharp peak in the center. We consider this function to serve as a comparison to the related work of Samworth and Wand [2010] and Doss and Weng [2018]. Figure 3.1 shows the plot for g, the levels considered and the corresponding level sets. We consider the levels $\lambda = 0.156, 0.261, 1.325$. In Figure 3.2 we

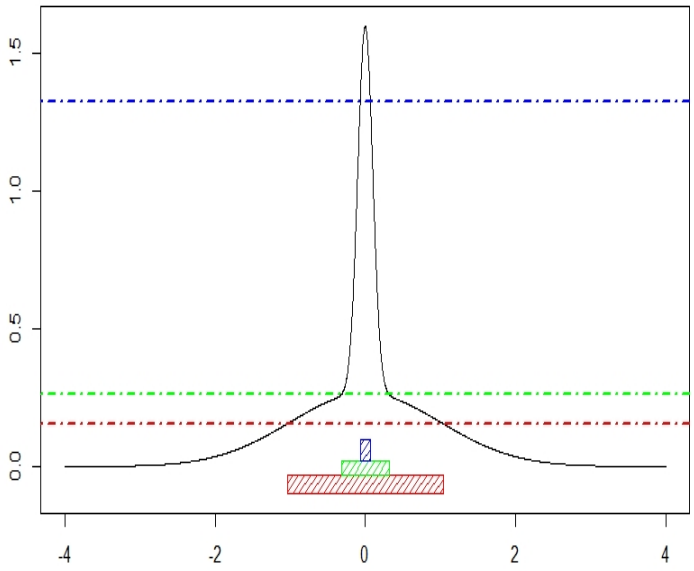

Figure 3.1: True g function for example one. We included the true level sets for $\lambda = 0.156$ (red), $\lambda = 0.261$ (green) and $\lambda = 1.325$ (blue).

show the performance of the estimators $\widehat{LS}_{\lambda,opt}$ and $\widehat{LS}_{\lambda,\text{LSCV}}$ on 250 Monte Carlo samples of $1,000$ and $100,000$ observations each. Every circle in the plot is a sample by sample comparison between the estimators where μ_{opt} is on the x-axis and μ_{LSCV} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LSCV}}/\mu_{opt}$. In Figure 3.3 we show similar comparisons between the performance of our bandwidth selector with the local polynomial kernel estimator. Note that the local polynomial estimator is different from the Nadaraya-Watson estimator in $\widehat{LS}_{\lambda,opt}$ and $\widehat{LS}_{\lambda,\text{LSCV}}$. We decide to include the local polynomial estimator as it is also used in practice and offers an advantage in the boundary bias as discussed in Fan [1992]. In Figure 3.3, every circle in the plot is a sample by sample comparison between the estimators where μ_{opt} is on the x-axis and μ_{LP} on the y-axis. As before, we consider samples of $1,000$ and $100,000$ observations. We observe that for samples

with $1,000$ observations, some of the estimated level sets return as empty sets. Our estimator $\widehat{LS}_{\lambda,opt}$ returns the smallest proportion of level sets (16.4%). It then follows the estimator $\widehat{LS}_{\lambda,\text{LSCV}}$ (36.8%) and finally $\widehat{LS}_{\lambda,\text{LP}}$ with 99.6% of the estimated sets being empty.

	vs LSCV		vs LP	
λ	proportion $\mu_{opt} < \mu_{\text{LSCV}}$	adj. p-val	proportion $\mu_{opt} < \mu_{\text{LP}}$	adj. p-val
0.156	0.936	3.440e-40	0.656	5.106e-09
0.261	0.552	1.179e-02	0.620	8.577e-07
1.325	0.840	3.042e-19	1.000	1.211e-35

	vs LSCV		vs LP	
λ	proportion $\mu_{opt} < \mu_{\text{LSCV}}$	adj. p-val	proportion $\mu_{opt} < \mu_{\text{LP}}$	adj. p-val
0.156	0.792	1.763e-07	0.700	0.035
0.261	0.608	0.303	0.296	0.073
1.325	0.888	6.6e-16	0.992	6.6e-16

Table 3.2: Summary of the results for example one. On top we summarize the results for 250 Monte Carlo samples of 1000 observations each. We present the proportion of the samples for which the error from the LSCV bandwidth selector (local polynomial estimator) is greater than the error yielded from our bandwidth selector. We also include the p-values for one-sided Wilcoxon rank tests for the paired errors. We applied the Holm-Bonferroni correction to account for multiple comparisons. The bottom table presents the same results, but for 250 Monte Carlo samples of 100,000 observations each.

In addition, we present a summary of these results in Table 3.2. In this table we include the proportion of the samples where our bandwidth selector yielded a smaller error than the LSCV bandwidth selector or the local polynomial kernel estimator. Note that when the estimated level set returned is an empty set, $LS_\lambda \triangle \emptyset = LS_\lambda$. Then, we calculate the error as the integral of f on the true level set. We also include the results from Wilcoxon rank tests for paired errors and we use the Holm-Bonferroni correction to account for multiple comparisons.

Our bandwidth selector has an advantage compared to the LSCV bandwidth selector

and local polynomial estimator for smaller samples across all three levels. Next, we take an example from toxicology.

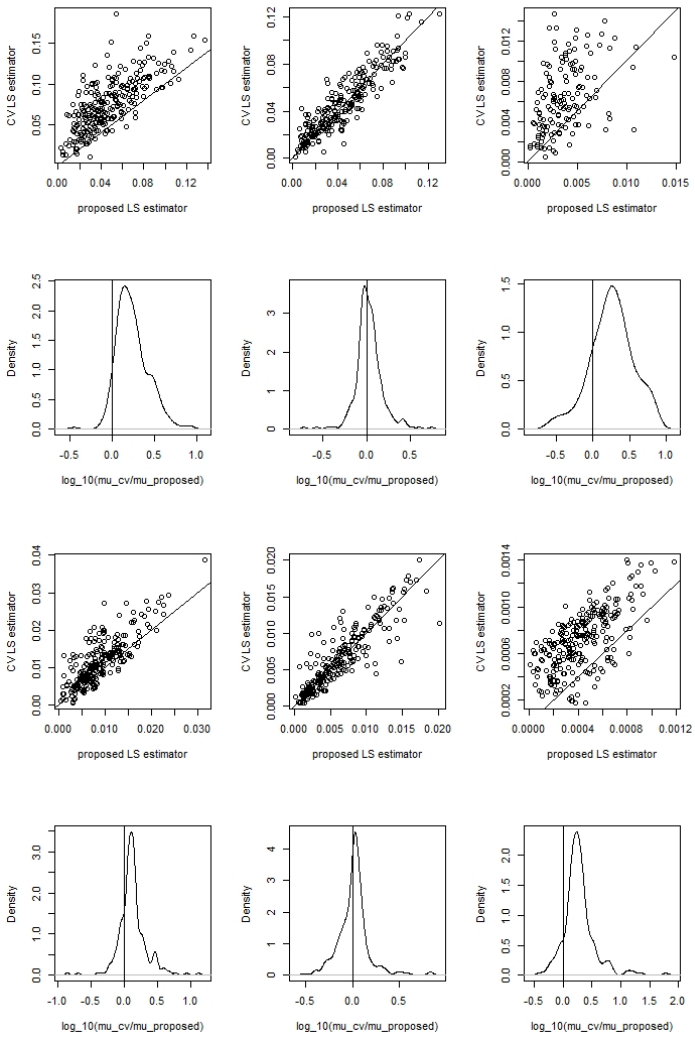

Figure 3.2: Sample by sample comparison between h_{opt} and h_{LSCV} for example one. We graph μ_{opt} on the x-axis and μ_{LSCV} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LSCV}}/\mu_{opt}$. We considered (from left to right) the levels $\lambda = 0.156, 0.261, 1.325$. The first two rows correspond to the results from 250 Monte Carlo samples of 1,000 observations each. The last two rows show the results from 250 Monte Carlo samples of $100,000$ observations each.

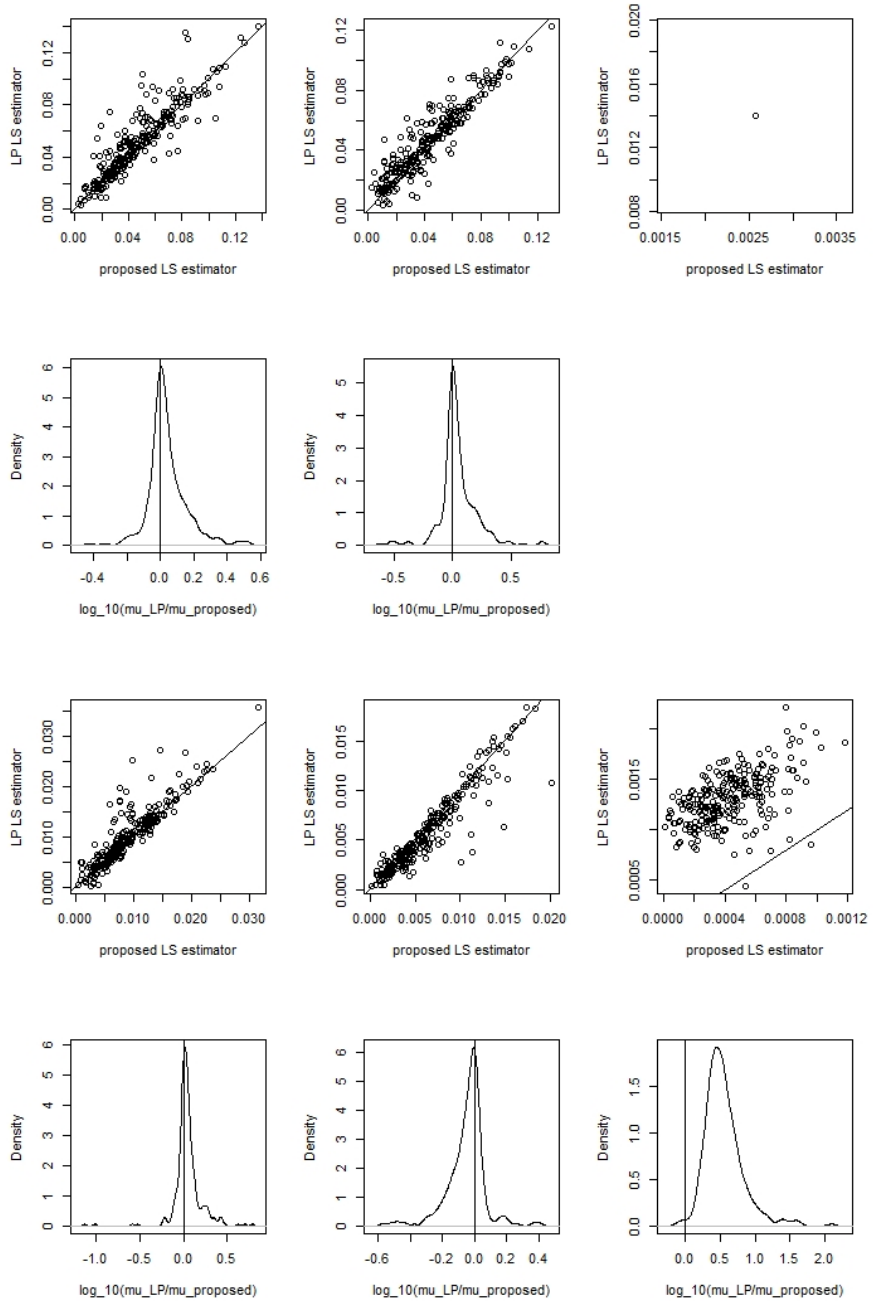

Figure 3.3: Sample by sample comparison between $\widehat{LS}_{\lambda,h_{opt}}$ and $\widehat{LS}_{\lambda,h_{LP}}$ for example one. We graph μ_{opt} on the x-axis and μ_{LP} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LP}}/\mu_{opt}$. We considered (from left to right) the levels $\lambda = 0.156, 0.261, 1.325$. The first two rows correspond to the results from 250 Monte Carlo samples of 1,000 observations each. For $\lambda = 1.325$ the estimator $\widehat{LS}_{\lambda,h_{LP}}$ is always empty except for one sample. The last two rows show the results from 250 Monte Carlo samples of $100,000$ observations each.

Example two

The term hormesis is defined in Mattson [2008] as "a process in which exposure to a low dose of a chemical agent or environmental factor that is damaging at higher doses induces an adaptive beneficial effect on the cell or organism". Calabrese and Baldwin [2002] present common dose-response curves found in toxicology studies with different chemical agents and experimental models. These are typically characterized by a U-shaped or J-shaped dose-response relationship.

We considered $g(x)$ to be a multi-modal curve similar to Calabrese and Baldwin [2002, Figure 2(j)]. Let

$$\varphi(x; m, s) \quad = \quad \frac{1}{s\sqrt{2\pi}} \exp^{-\frac{1}{2}\left(\frac{x-m}{s}\right)^2}.$$

then, for example two:

$$g(x) = \begin{cases} .7\varphi(x; 4, .15) + .45\varphi(x; 6, .5) & 2 \leq x \leq 8 \\ -\frac{1}{2}(x - 8.5)^2 + c_0 & 8 < x \leq 10 \end{cases}$$

where $c_0 = .7\varphi(8; 4, .15) + .45\varphi(8; 6, .5) + \frac{1}{2}(0.5)^2$ is calculated to guarantee continuity on the function. We allow the slope of g to change at different rate in each mode on the function to make the level set estimation more difficult. Moreover, these sharp changes are observed in different graphs that Calabrese and Baldwin [2002] reference. We show in Figure 3.4 the plot for g, the levels considered and the corresponding sets LS_λ.

We consider the levels $\lambda = 0.05, 0.10, 1.00, 1.30, 1.40, 1.50$. As for example one, we include the local polynomial estimator and the LSCV bandwidth selector as alternative approaches to estimate LS_λ. In Table 3.3 we show the proportion of samples where $h_{opt}(\widehat{LS}_{\lambda,opt})$ yielded a smaller error than h_{LSCV} ($\widehat{LS}_{\lambda,\mathrm{LSCV}}$) or $\widehat{LS}_{\lambda,\mathrm{LP}}$. We include the adjusted p-values for the Wilcoxon rank tests for paired errors. We use the Holm-Bonferroni correction to account for multiple comparisons. The top table refers to the results from 250 Monte Carlo samples of $1,000$ observations each. The lower table refers to 250 Monte Carlo samples of $100,000$

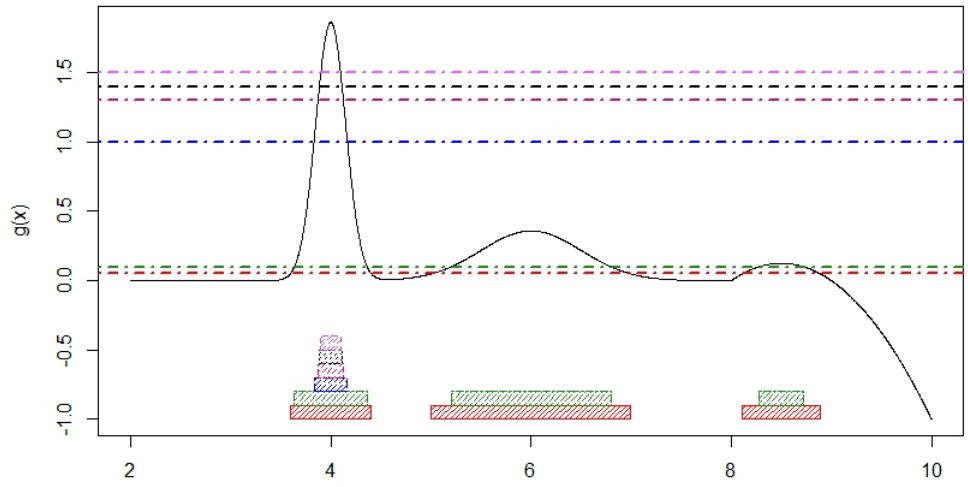

Figure 3.4: True g function for example two. We included the true level sets for $\lambda = 0.05$ (red), $\lambda = 0.10$ (green), $\lambda = 1.00$ (blue), $\lambda = 1.30$ (dark pink),$\lambda = 1.40$ (black) and $\lambda = 1.50$ (purple).

observations each.

Next, we take the levels $\lambda = 0.05, 1.00, 1.50$ as these summarize what we observed for all levels in Table 3.3. We compare the performance of our bandwidth selector and the LSCV bandwidth selector in Figure 3.5. Every circle in the plot is a sample by sample comparison between the estimators where μ_{opt} is on the x-axis and μ_{LSCV} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LSCV}}/\mu_{opt}$. The top two rows in Figure 3.5 show the results for 250 Monte Carlo samples of $1,000$ observations each, while the lower two show 250 Monte Carlo samples of $100,000$ observations each. In Figure 3.6 we compare the plug-in estimator with our bandwidth selector ($\widehat{LS}_{\lambda,opt}$) against $\widehat{LS}_{\lambda,\text{LP}}$. Figure 3.6 shows the comparisons for 250 Monte Carlo samples of $1,000$ observations (two top rows) and $100,000$ observations (last two rows).

In general, our bandwidth selector offers an advantage to the LSCV bandwidth selector

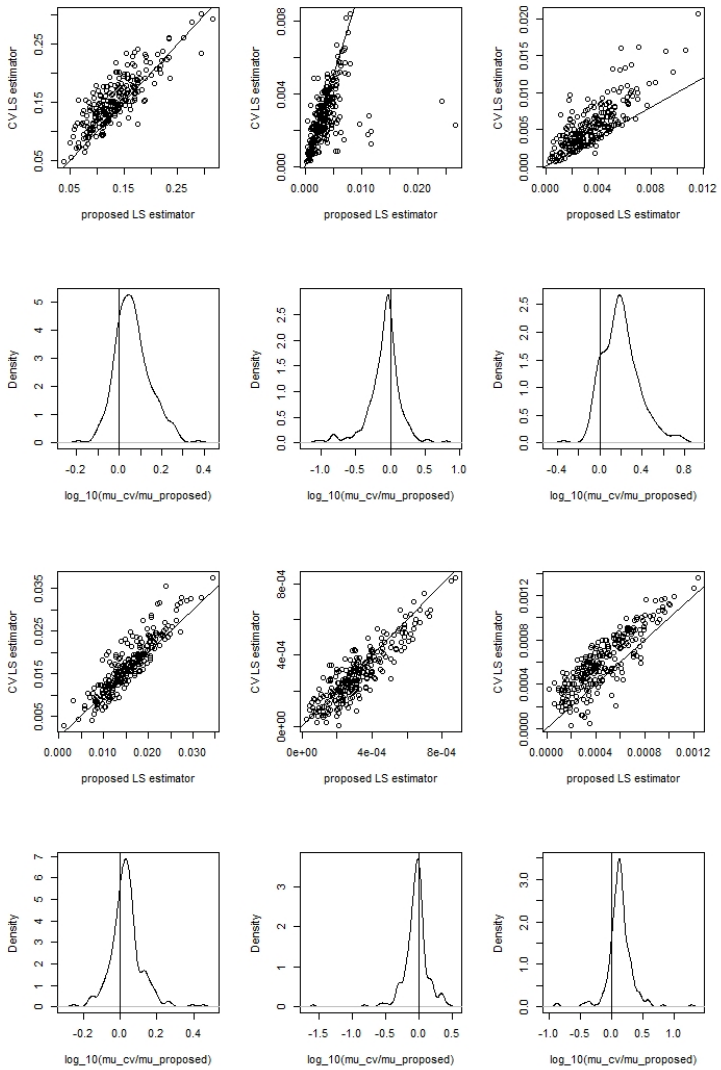

Figure 3.5: Sample by sample comparison between h_{opt} and h_{LSCV} for example two. We graph μ_{opt} on the x-axis and μ_{LSCV} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LSCV}}/\mu_{opt}$. We considered (from left to right) the levels $\lambda = 0.05, 1.00, 1.50$. The first two rows correspond to the results from 250 Monte Carlo samples of 1,000 observations each. The last two rows show the results from 250 Monte Carlo samples of $100,000$ observations each.

and the local polynomial estimator except for $\lambda = 1.00$. This behaviour on the plug-in estimator was also observed in Samworth and Wand [2010]; Doss and Weng [2018]. As

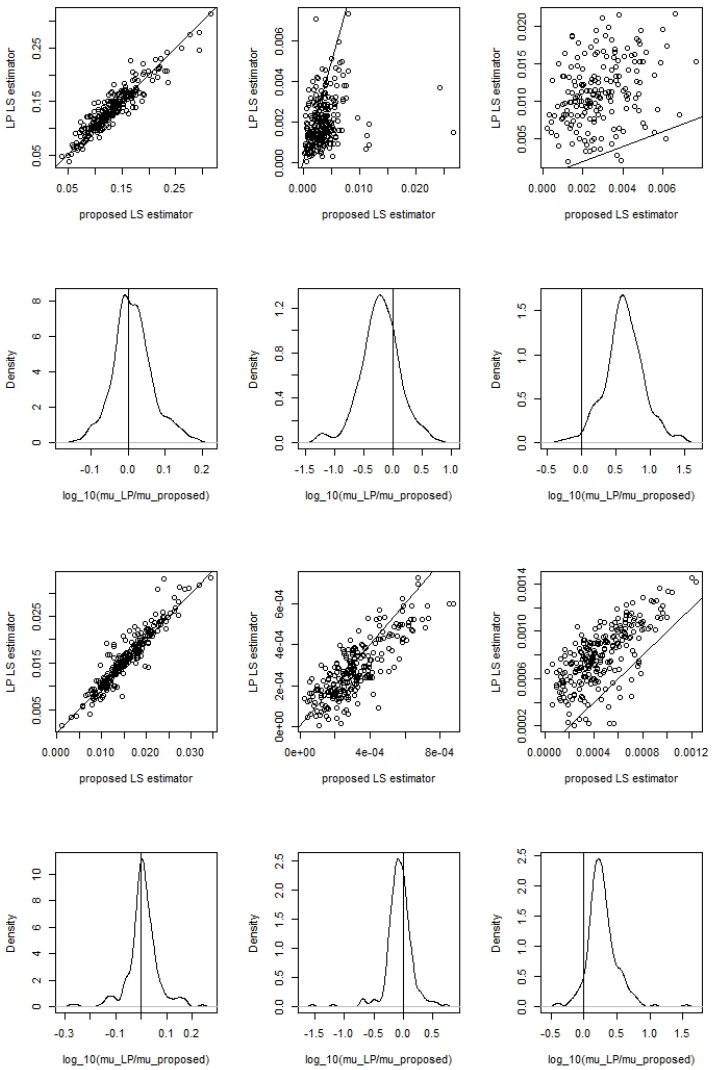

Figure 3.6: Sample by sample comparison between $\widehat{LS}_{\lambda,h_{opt}}$ and $\widehat{LS}_{\lambda,h_{LP}}$ for example two. We graph μ_{opt} on the x-axis and μ_{LP} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LP}}/\mu_{opt}$. We considered (from left to right) the levels $\lambda = 0.05, 1.00, 1.50$. The first two rows correspond to the results from 250 Monte Carlo samples of 1,000 observations each. The last two rows show the results from 250 Monte Carlo samples of $100,000$ observations each.

observed here, the authors noted that their plug-in estimator did not offer an advantage across all levels and densities when compared to the cross-validation bandwidth selector.

	vs LSCV		vs LP	
λ	proportion $\mu_{opt} < \mu_{\text{LSCV}}$	adj. p-val	proportion $\mu_{opt} < \mu_{\text{LP}}$	adj. p-val
0.05	0.792	1.324e-23	0.564	8.881e-04
0.10	0.756	5.211e-18	0.556	3.272e-02
1.00	0.304	1	0.240	1
1.30	0.700	1.019e-15	0.856	8.761e-33
1.40	0.844	8.504e-31	0.944	1.380e-40
1.50	0.880	2.829e-37	0.988	7.546e-42

	vs LSCV		vs LP	
λ	proportion $\mu_{opt} < \mu_{\text{LSCV}}$	adj. p-val	proportion $\mu_{opt} < \mu_{\text{LP}}$	adj. p-val
0.05	0.696	7.952e-12	0.576	0.00513
0.10	0.652	2.990e-09	0.548	0.024
1.00	0.388	1	0.352	1
1.30	0.460	1	0.560	0.061
1.40	0.780	2.64e-15	0.836	2.64e-15
1.50	0.852	2.64e-15	0.936	2.64e-15

Table 3.3: Summary of the results for example two. On top we summarize the results for 250 Monte Carlo samples of 1000 observations each. We present the proportion of the samples for which the error from the LSCV bandwidth selector (or local polynomial estimator) is greater than the error yielded from our bandwidth selector. We also include the p-values for one-sided Wilcoxon rank tests for the paired errors. We applied the Holm-Bonferroni correction to account for multiple comparisons. The table below presents the same results, but for 250 Monte Carlo samples of 100,000 observations each.

Example three

Here we consider the doppler function. This function is a common example for nonparametric regression, see [Wasserman, 2006]. In particular, we consider a truncated version of the doppler function. Let

$$g(x) \quad = \quad \sqrt{x(1-x)}\sin\left(\frac{2.1\pi}{x+0.5}\right) \text{ for } x \in [0.1, 1].$$

We show in Figure 3.7 the plot for g, the levels considered and the corresponding sets LS_λ.

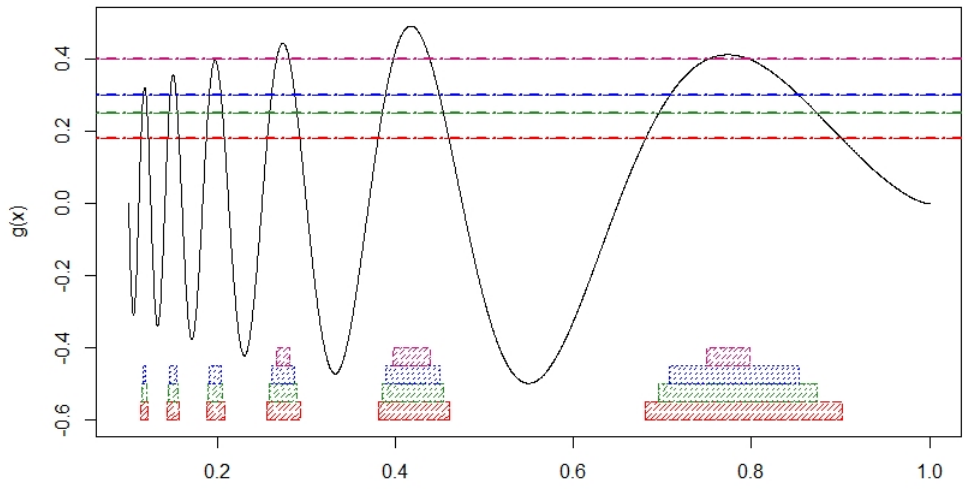

Figure 3.7: True g function for example three. We included the true level sets for $\lambda = 0.18$ (red), $\lambda = 0.25$ (green), $\lambda = 0.30$ (blue) and $\lambda = 0.40$ (deep pink).

In example three, we consider the levels $\lambda = 0.18, 0.28, 0.30, 0.40$. As before, we compare our bandwidth selector against the LSCV bandwidth selector and the local polynomial kernel estimator. In Figure 3.8 we compare the performance of our bandwidth h_{opt} against h_{LSCV}. We consider two sets of 250 Monte Carlo samples. The results in the top two rows in the figure refer to samples of $1,000$ observations each. In the last two rows, the samples have

	vs LSCV		vs LP	
λ	proportion $\mu_{opt} < \mu_{\text{LSCV}}$	adj. p-val	proportion $\mu_{opt} < \mu_{\text{LP}}$	adj. p-val
0.18	0.244	1	0.632	1.233e-07
0.25	0.336	1	0.684	2.537e-09
0.30	0.468	1	0.576	0.021
0.40	0.532	0.096	0.436	1

	vs LSCV		vs LP	
λ	proportion $\mu_{opt} < \mu_{\text{LSCV}}$	adj. p-val	proportion $\mu_{opt} < \mu_{\text{LP}}$	adj. p-val
0.18	0.348	1	0.96	1.76e-15
0.25	0.244	1	0.936	1.76e-15
0.30	0.372	1	0.912	1.76e-15
0.40	0.868	1.76e-15	0.368	1

Table 3.4: Summary of the results for example three. On top we summarize the results for 250 Monte Carlo samples of 1000 observations each. We present the proportion of the samples for which the error from the LSCV bandwidth selector (or local polynomial estimator) is greater than the error yielded from our bandwidth selector. We also include the p-values for one-sided Wilcoxon rank tests for the paired errors. We applied the Holm-Bonferroni correction to account for multiple comparisons. The table below presents the same results, but for 250 Monte Carlo samples of 100,000 observations each.

$100,000$ observations each. Every circle in the plot is a sample by sample comparison between the estimators where μ_{opt} is on the x-axis and μ_{LSCV} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LSCV}}/\mu_{opt}$. Next we contrast our bandwidth selector with the local polynomial kernel estimator. We compare the performance of these two estimators in the same spirit as we did with the LSCV bandwidth selector. Figure 3.9 shows the results. As before, we consider samples of $1,000$ and $100,000$ observations. We observe that our estimator $\widehat{LS}_{\lambda,opt}$ offers an advantage compared to the local polynomial estimator $\widehat{LS}_{\lambda,\text{LP}}$ for both sample sizes.

We summarize the results from our simulations in Table 3.4. We present the proportion of samples for which our bandwidth selector had a smaller error than the LSCV bandwidth

selection and local polynomial estimator. Our bandwidth selector offers an advantage to the local polynomial estimator. However, this is not the case when compared to the LSCV. We observe that only for the highest level $\lambda = 0.40$ our bandwidth selector performs better than the LSCV. Figure 3.10 shows that h_{opt} yields an estimator for g that has less variability mainly seen in the last mode of the function. We see that since h_{opt} is chosen by minimizing a local loss function, $\widehat{g}_{h_{opt}}$ fits better the true function g on the regions closer to the level $\lambda = 0.40$ as compared to the estimator with the LSCV bandwidth.

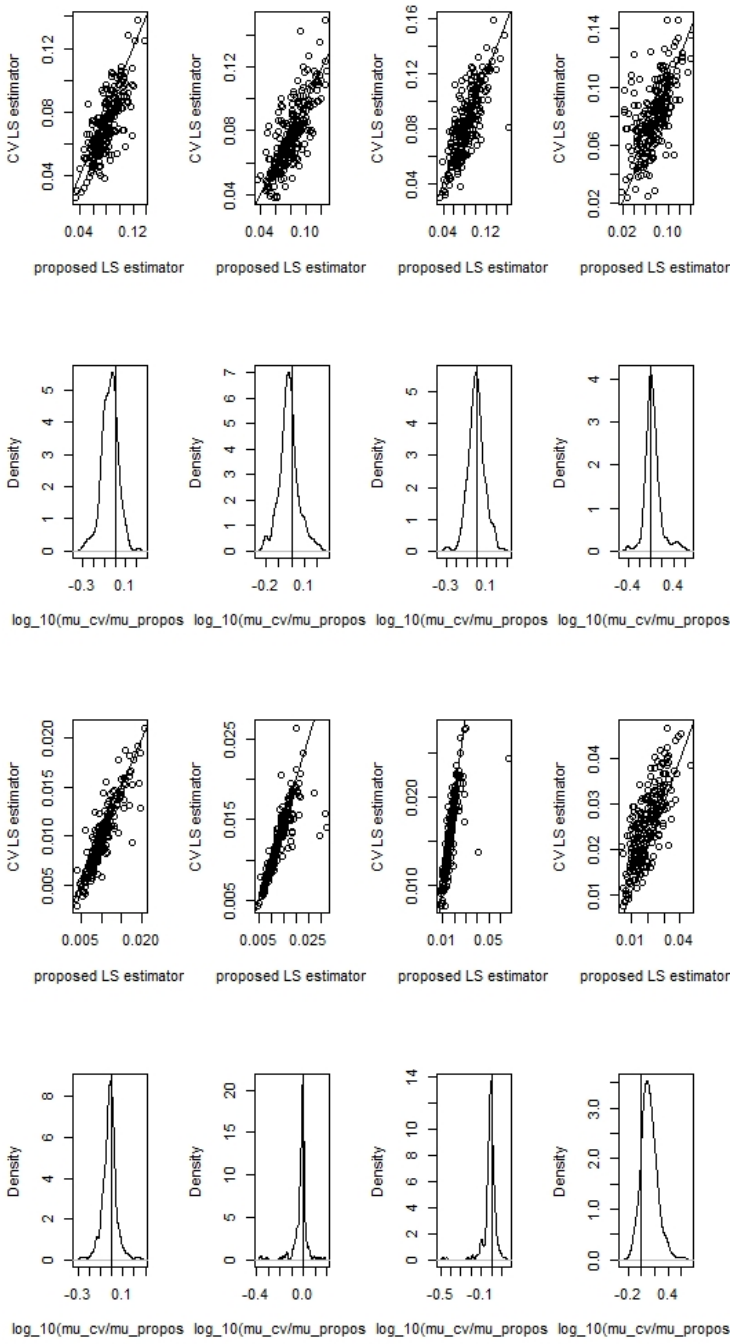

Figure 3.8: Sample by sample comparison between h_{opt} and h_{LSCV} for example three. We graph μ_{opt} on the x-axis and μ_{LSCV} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LSCV}}/\mu_{opt}$. We considered (from left to right) the levels $\lambda = 0.18, 0.25, 0.30, 0.40$. The first two rows correspond to the results from 250 Monte Carlo samples of 1,000 observations each. The last two rows show the results from 250 Monte Carlo samples of $100,000$ observations each.

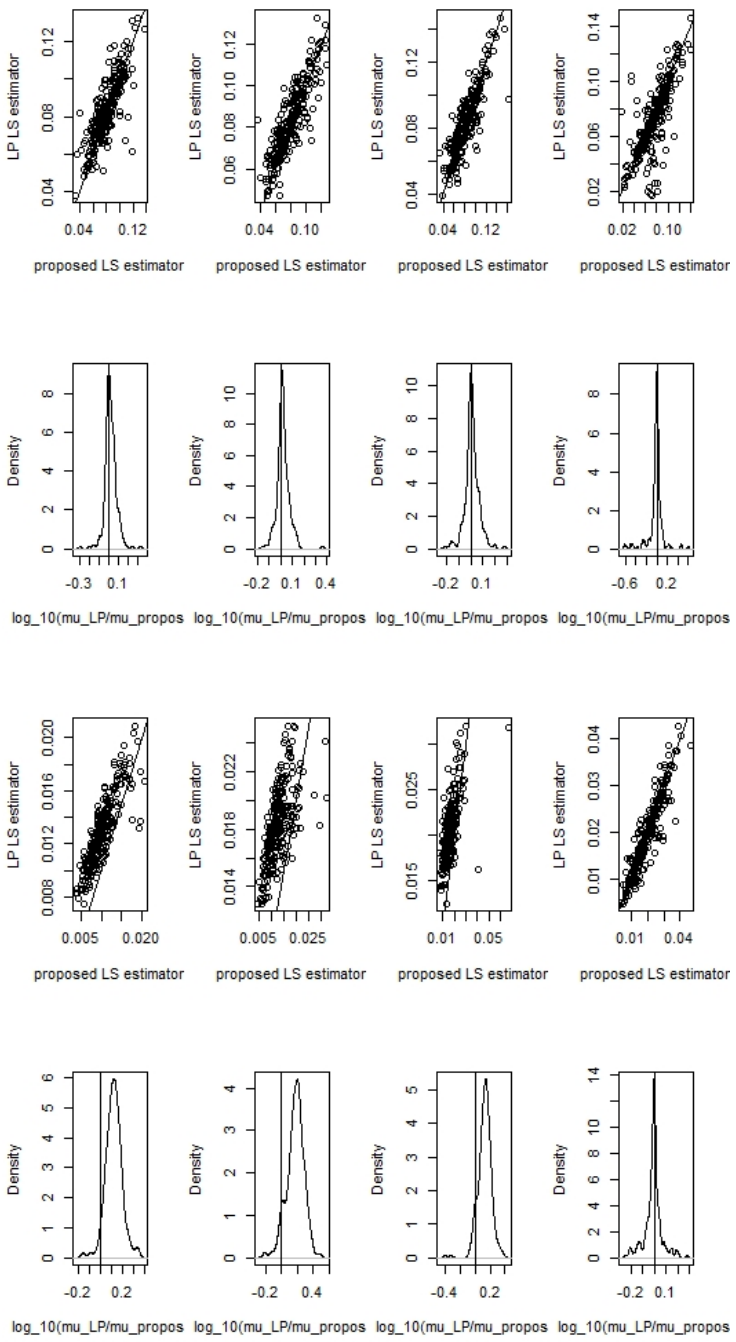

Figure 3.9: Sample by sample comparison between $\widehat{LS}_{\lambda,h_{opt}}$ and $\widehat{LS}_{\lambda,h_{LP}}$ for example three. We graph μ_{opt} on the x-axis and μ_{LP} on the y-axis. We also include density plots for the logarithm of the ratio $\mu_{\text{LP}}/\mu_{opt}$. We considered (from left to right) the levels $\lambda = 0.18, 0.25, 0.30, 0.40$. The first two rows correspond to the results from 250 Monte Carlo samples of 1,000 observations each. The last two rows show the results from 250 Monte Carlo samples of $100,000$ observations each.

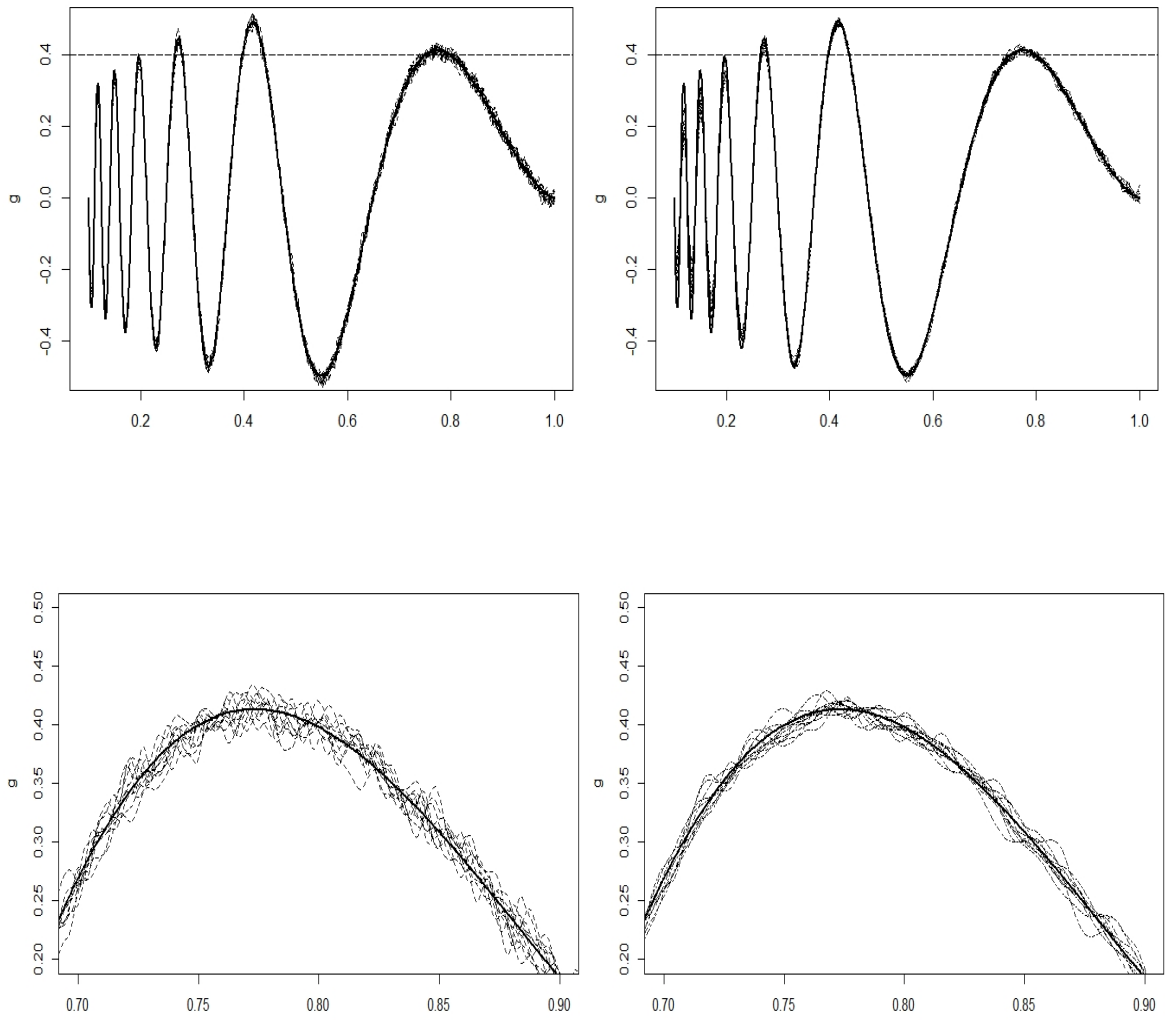

Figure 3.10: We randomly select ten Monte Carlo samples of 100,000 observations each. The solid line in each plot represents the truncated doppler function in example three. The plot at the top left corner shows ten estimators $\widehat{g}_{h_{\text{LSCV}}}$ (dotted curves) for the truncated doppler function. The plot at the top right corner shows ten estimators $\widehat{g}_{h_{opt}}$ (dashed curves) with the bandwidth chosen optimally for the level $\lambda = 0.40$. The plots at the bottom show close up images of the function.

Next, we study the case when the distribution of the outcome Y, conditional on the covariate X, is a member of the exponential family. We consider the special case when Y is binary.

3.5.2 A binary outcome

Let Y take values $\{0, 1\}$ and $x \in \mathbb{R}$. Assume the conditional density $f_{Y|X}(y)$ is a member of the exponential family reviewed in Section 2.2.1.2. Consider the regression model

$$\Psi\left(g(x)\right) \quad = \quad \eta(x)$$

where $g(x) = \mathrm{E}[Y|X = x]$ and Ψ is the canonical link function for the Bernoulli distribution

$$\Psi(s) \quad = \quad \log\left(\frac{s}{1-s}\right).$$

Recall that the asymptotic risk and therefore our bandwidth h_{opt} depend on the unknown derivatives g' and g''. One can replace them with kernel estimators, as we did for Y Gaussian. Let h_1 and h_2 be the bandwidths for the derivatives of g. Recall that for Y Gaussian we let $h_1 = h_2 = h_{\mathrm{LP}}$, the optimal bandwidth for the local polynomial kernel estimator of degree one proposed in Fan et al. [1995] and reviewed in Section 2.2.2.3

$$h_{\mathrm{LP}} \quad = \quad \left[\frac{\int \sigma_{0,1}^2(x;K) f(x) w(x) dx}{\{\int z^2 K_{0,1}(z) dz\}^2 \{\int \eta^{(2)}(x)^2 f(x) w(x) dx\}}\right]^{1/5} n^{-1/5}.$$

However, computing h_{LP} for Y not Gaussian is more complex. Fan et al. [1995] suggests a "rough-and-ready" bandwidth. The authors suggest to fit η with a polynomial which degree is at least two. Nevertheless, this bandwidth has no asymptotic properties and therefore we will use the oracle bandwidths here.

Let c_{opt}^* be the argument that minimizes the asymptotic risk where we assume all functions are known. Thus, the oracle bandwidth for our bandwidth selector discussed in Section 3.3 is $h_{opt}^* = c_{opt}^* n^{-1/5}$. We do the same for the local polynomial optimal bandwidth. We assume

all functions in h_{LP} are known and obtain the oracle bandwidth h^*_{LP}. The corresponding plug-in estimators for LS_λ are thus

$$\begin{aligned}\widehat{LS}^*_{\lambda,opt} &= \{x : \widehat{g}_{n,h^*_{opt}} \geq \lambda\} \\ \widehat{LS}^*_{\lambda,\text{LP}} &= \{x : \widehat{g}_{n,h^*_{\text{LP}}} \geq \lambda\}.\end{aligned}$$

These estimators yield the errors

$$\begin{aligned}\mu^*_{opt} &= \mu_f\left(LS_\lambda \triangle \widehat{LS}^*_{\lambda,opt}\right) \\ \mu^*_{\text{LP}} &= \mu_f\left(LS_\lambda \triangle \widehat{LS}^*_{\lambda,\text{LP}}\right).\end{aligned}$$

For this simulation study we transform g in example one. Let

$$\widetilde{g}(x) = \frac{2}{3}\frac{1}{\sqrt{2\pi}}\exp^{-\frac{1}{2}x^2} + \frac{10}{3}\frac{1}{\sqrt{2\pi}}\exp^{-50x^2} \qquad \text{for x in [-4,4].}$$

We now consider

$$g(x) = 10\widetilde{g}(x) - 3 \qquad \text{for x in [-4,4].}$$

Figure 3.11 shows the plot for g and the true level sets in this simulation study. As before, we simulate 250 Monte Carlo samples of $1,000$ observations each. We take into account the levels $\lambda = 0.1908, 0.4034, 0.80, 0.90$. In Figure 3.12 we compare the performance of the estimators $\widehat{LS}^*_{\lambda,opt}$ and $\widehat{LS}^*_{\lambda,\text{LP}}$. We refer to the level set estimators and not the bandwidths since the local polynomial kernel estimator, reviewed in Section 2.2.2.3, is different to the Nadaraya Watson kernel estimator used for $\widehat{LS}^*_{\lambda,opt}$. Each point represents one sample. The error μ^*_{opt} is on the x-axis and μ^*_{LP} is on the y-axis. We also include density plots for the logarithm of the ratio $\mu^*_{\text{LP}}/\mu^*_{opt}$. We summarize the results from our simulation in Table 3.5. We include the proportion of the samples where our estimator yielded a smaller error than the local polynomial estimator. To compute the necessary bandwidths for both estimators, we assume all functions are known. We observe that our estimator gives an advantage over

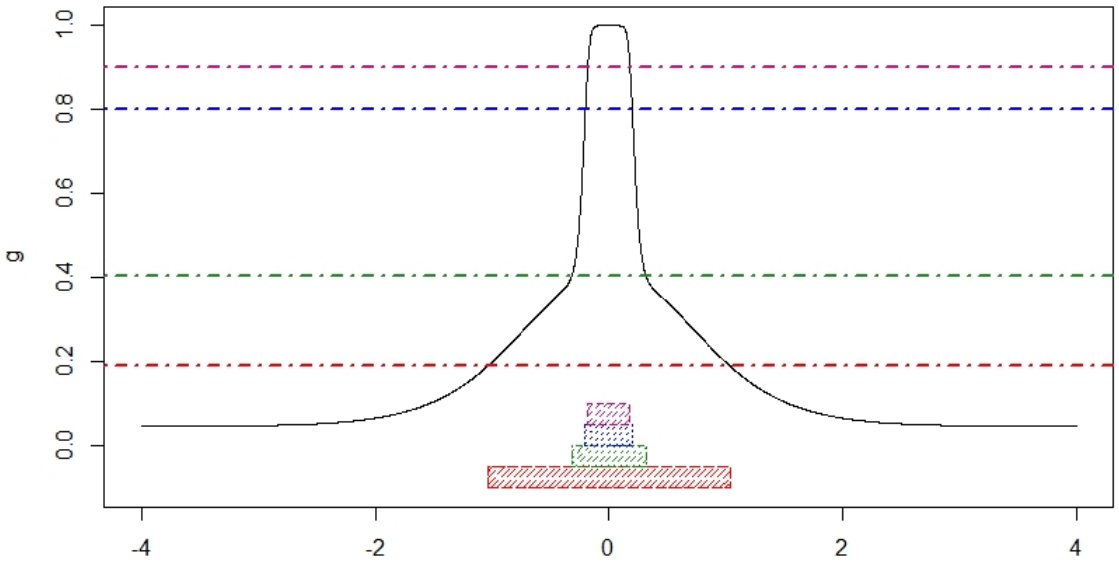

Figure 3.11: True g function for example with Y binary. We included the true level sets for $\lambda = 0.1908$ (long-dashed line), $\lambda = 0.4034$ (dot-dashed line), $\lambda = 0.80$ (dotted line) and $\lambda = 0.90$ (dashed line).

the local polynomial for three of the four level considered.

λ	proportion $\mu^*_{opt} < \mu^*_{\text{LP}}$	adj. p-val
0.1908	0.332	1
0.4034	0.968	4.925e-42
0.80	0.852	8.927e-36
0.90	0.840	2.758e-35

Table 3.5: Summary of the results for the example with Y binary. We summarize the results for 250 Monte Carlo samples of 1000 observations each. We present the proportion of the samples for which the error from the local polynomial estimator with bandwidth h^*_{LP} is greater than the error yielded from our oracle bandwidth selector (h^*_{opt}). We also include the p-values for one-sided Wilcoxon rank tests for the paired errors. We applied the Holm-Bonferroni correction to account for multiple comparisons.

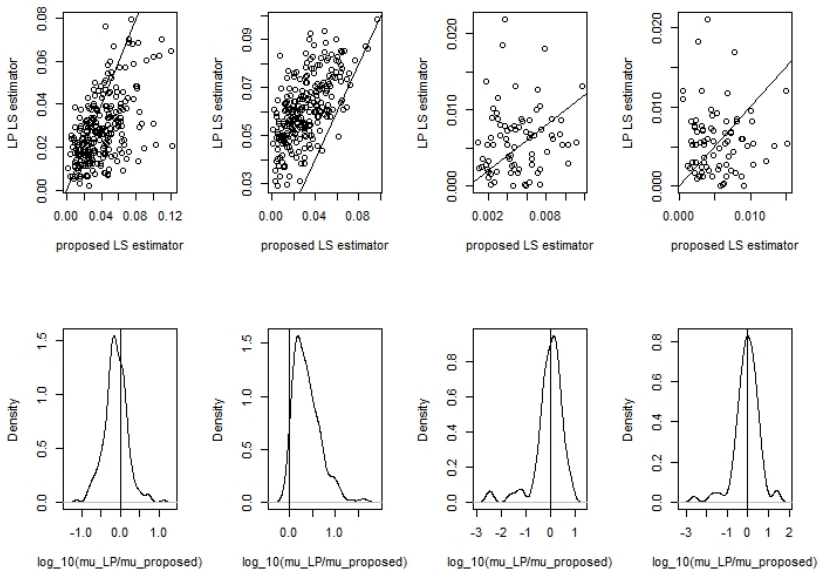

Figure 3.12: Sample by sample comparison between $\widehat{LS}^*_{\lambda,opt}$ and $\widehat{LS}^*_{\lambda,\mathrm{LP}}$. We graph μ^*_{opt} on the x-axis and μ^*_{LP} on the y-axis. We also include density plots for the logarithm of the ratio $\mu^*_{\mathrm{LP}}/\mu^*_{opt}$. We considered (from left to right) the levels $\lambda = 0.1908, 0.4034, 0.80, 0.90$. We considered 250 Monte Carlo samples of 1,000 observations each for the example with Y binary.

To conclude, in this simulation study we observe that our bandwidth selector offers some advantage to the alternative LSCV or local polynomial plug-in estimator. However, as it was discussed in the highest density region problem (Samworth and Wand [2010]; Doss and Weng [2018]), our bandwidth selector does not perform better than the alternative estimators for all regression functions (or density functions for the HDR problem) or across all levels. Recall x_j are the points such that $g(x_j) = \lambda$. On the cases where our estimator did not improve, we observe that the estimators for $g'(x_j)$ and $g''(x_j)$ were not accurate therefore affecting the results.

In this thesis we consider the bandwidth selector that we propose as an initial step into level set estimation with a plug-in estimator. Our approach is centred into optimizing the smoothing parameter h, but this is not the only approach possible. Studying an estimator

different from the kernel estimator is a worth future project. One could estimate g with splines [Schoenberg, 1946] or wavelets. The later one being functions that are locally adaptive (see Wasserman [2006]) and therefore could be an interesting estimator to study in the context of level set estimation. In the next section we present an example of application considering data on decompression sickness.

3.6 A decompression sickness study

Decompression sickness (DCS) occurs when pressure around the body decreases rapidly. It can happen in the air or in the water. During the decompression, dissolved gases form bubbles that travel to any part of the body. Therefore, the DCS can provoke different symptoms and produce a damage of various degrees from pain, to paralysis, or death. Examples of situations where DCS can occur are: diving, working in a caisson, flying in an unpressurised aircraft and extra vehicular activity from space (Cooper and Hanson [2021]).

Researchers from the University of Wisconsin conducted an experiment to study the effects of the DCS in the human body. They used sheep as the experimental subjects and simulated dives in a pressure chamber. The data set contains the results for $n = 1108$ sheep. For each, the designed pressure (in absolute atmospheres) and duration (in minutes) was recorded. The outcomes for the central nervous system, respiratory disease, limb bends and mortality were also included.

This data set has been studied in the context of the ED problem in Li et al. [2008] and more recently in Jankowski et al. [2014]. We refer the reader to these papers and the references that there appear for a more thorough discussion on the data collection for this experiment.

Here we consider pressure and duration as the covariates and mortality as the outcome. Because our theory is developed for a one dimension problem, we perform a princial component analysis (PCA) on pressure and duration. Let $Z \in \mathbb{R}$, the PCA score, be the new covariate and note the outcome is binary with values $\{0, 1\}$. A value of 1 in the outcome represents that the sheep died. Thus, for $g(z) = \mathrm{E}[Y|Z = z]$ and a fixed level $\lambda \in (0, 1)$, the

LS_λ level set is

$$\mathrm{LS}_\lambda \quad = \quad \{z : g(z) \geq \lambda\}.$$

Our goal then is to select the optimal bandwidth for the plug-in estimator

$$\widehat{LS}_{n,\lambda} \quad = \quad \{z : \widehat{g}_{n,h}(z) \geq \lambda\}.$$

Recall that our asymptotic risk function depends on the first two derivatives of g. We code the algorithm in Fan et al. [1995] to compute kernel estimators for g' and g'' with bandwidths h_1 and h_2 respectively. To reduce the computational complexity, we set $h_1 = h_2 = h_{LP}$. The bandwidth h_{LP} is the optimal bandwidth of the local polynomial kernel regression estimator (with degree one) for g. In addition, our asymptotic risk depends on an initial estimator $\widehat{g}^{(0)}$ and points z_j such that $\widehat{g}^{(0)}(z_j) = \lambda$ for $j = 1, \ldots, 2r$ for some $r > 0$. We observed that for some points z_j we had that $\widehat{f}_{h_0}(z_j) = 0$, which brings numerical errors in our asymptotic risk. To avoid this situation, we transformed z_j by adding a difference $\delta > 0$ such that $\widehat{f}_{h_0}(z_j + \delta) \neq 0$.

Let h_{LS_λ} be the optimal bandwidth for the estimator of the level set LS_λ. In addition, let $\widehat{g}_{n,h}$ be the Nadaraya-Watson kernel regression estimator with bandwidth h. We show in Figure 3.13 the estimators $\widehat{g}_{n,h_{\mathrm{LS}_\lambda}}$ and the corresponding level sets $\widehat{LS}_{n,\lambda}$ for levels $\lambda = 0.20, 0.50, 0.80$. The estimator $\widehat{g}_{n,h}$ is smoother in the bottom plot of the figure. This happens because for the level set $\lambda = 0.80$, the fluctuations of the function in the interval $[-2, 1)$ have lesser importance. Therefore, the optimal bandwidth for this level set is visibly larger than the optimal bandwidth for $\lambda = 0.20$ where the fluctuations of the function are more important to estimate $LS_{0.20}$ accurately.

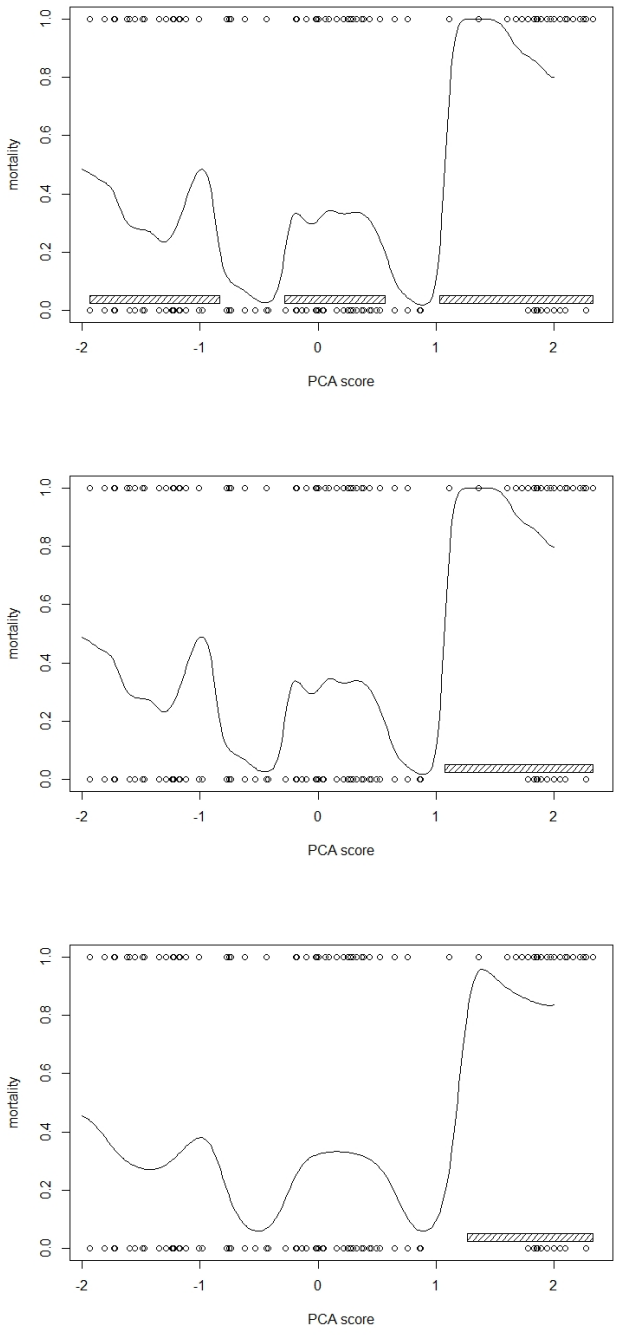

Figure 3.13: In every plot we show the observed outcomes and PCA scores. From top to bottom we considered the levels $\lambda = 0.20, 0.50, 0.80$. We included the Nadaraya-Watson estimator $\widehat{g}_{n,h_{\mathrm{LS}_\lambda}}$ with the bandwidth that is optimal for every level set. The shaded rectangles in each plot show the actual estimated level set $\widehat{LS}_{n,\lambda}$.

Chapter 4

PART II: Alternative estimators for the highest density region, two simulation studies

In this section we revisit the simulation study for the highest density region (HDR) in Samworth and Wand [2010]. For a density f and a fixed $0 < p < 1$, recall from Section 2.3.4 that the $100(1-p)\%$ HDR is the set defined as

$$\text{HDR}_p \quad = \quad \big\{x : f(x) \geq \lambda_p\big\}.$$

This is a special type of a level set. The level λ_p depends on f as follows

$$\lambda_p \quad = \quad \inf\Big\{\lambda \in (0,\infty) : \int_{-\infty}^{\infty} f(x)\mathbb{1}_{\{f(x)\geq\lambda\}}dx \leq 1-p\Big\}.$$

Here we consider two estimators of f. Each will yield a plug-in estimator for the HDR. In the following section we start with the local polynomial kernel density estimator. We will compare its approximation error against the bandwidth selector in Samworth and Wand [2010]. Then we will study the log-concave plug-in estimator for the HDR. We estimate f with the log-concave mixture model proposed in Boonpatcharanon [2019]. An advantage of

this type of estimator is that it does not require an extra smoothing parameter as the kernel estimator does.

4.1 A simulation study with a local polynomial kernel density estimator

For a bandwidth h and data points $X_1, \ldots, X_n$, recall the kernel density estimator

$$\widehat{f}_h(x) \quad = \quad \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right).$$

A plug-in kernel estimator for the HDR is thus

$$\widehat{\mathrm{HDR}}_{p,h} \quad = \quad \left\{x : \widehat{f}_h(x) \geq \widehat{\lambda}_{p,h}\right\}$$

where the level λ_p is estimated as

$$\widehat{\lambda}_{p,h} \quad = \quad \inf\left\{\lambda \in (0,\infty) : \int_{-\infty}^{\infty} \widehat{f}_h(x) \mathbb{1}_{\{\widehat{f}_h(x) \geq \lambda\}} dx \leq 1 - p\right\}.$$

Samworth and Wand [2010] proposed an algorithm to select the optimal bandwidth for $\widehat{\mathrm{HDR}}_{p,h}$. We summarize the authors' method in Section 2.3.4. We recall the loss function studied in Samworth and Wand [2010] for the estimator $\widehat{\mathrm{HDR}}_{p,h}$

$$\mu_f\left(\widehat{\mathrm{HDR}}_{p,h} \triangle \mathrm{HDR}_p\right) \quad = \quad \int_{\widehat{\mathrm{HDR}}_{p,h} \triangle \mathrm{HDR}_p} f(x) dx.$$

where $A \triangle B = (A \cap B^c) \cup (A^c \cap B)$. The optimal bandwidth h_{opt} is thus the bandwidth that minimizes the asymptotic expansion of $\mathrm{E}\left[\mu_f\left(\widehat{\mathrm{HDR}}_{p,h} \triangle \mathrm{HDR}_p\right)\right]$ where the expectation is taken with respect to the data. We compare the estimator $\widehat{\mathrm{HDR}}_{p,h_{opt}}$ with the local polynomial estimator available in the *locpoly* function within the **KernSmooth** R-package. Let $\widehat{\mathrm{HDR}}_{p,h_{\mathrm{LP}}}$ be the latter and h_{LP} its optimal bandwidth. We compute h_{LP} with the function *dpik* also from the **KernSmooth** R-package. The method for selecting the bandwidth h_{LP} is

a direct-plug in estimator where the unknown functionals are replaced with kernel estimators as described in Chapter 3 in Wand and Jones [1994]. The corresponding loss function for the local polynomial kernel estimator is thus

$$\mu_f\left(\widehat{\mathrm{HDR}}_{p,h_{\mathrm{LP}}} \triangle \mathrm{HDR}_p\right) \;=\; \int_{\widehat{\mathrm{HDR}}_{p,h_{\mathrm{LP}}} \triangle \mathrm{HDR}_p} f(x)dx.$$

We consider the ten densities in Marron and Wand [1992b] and studied in Samworth and Wand [2010]. For every density we simulated 250 Monte Carlo samples of $1,000$ observations each. For a fixed density f, let $\widehat{\mathrm{HDR}}^{[i]}_{p,h_{\mathrm{LP}}}$ and $\widehat{\mathrm{HDR}}^{[i]}_{p,h_{opt}}$ be the HDR estimators for the i^{th} sample with errors

$$\begin{aligned}
\mu^{\mathrm{LP}}_{f,p} &= \mu_f\left(\widehat{\mathrm{HDR}}^{[i]}_{p,h_{\mathrm{LP}}} \triangle \mathrm{HDR}_p\right) \\
\mu^{opt}_{f,p} &= \mu_f\left(\widehat{\mathrm{HDR}}^{[i]}_{p,h_{opt}} \triangle \mathrm{HDR}_p\right).
\end{aligned}$$

We consider the values $p = 0.20, 0.50, 0.80$ for all ten densities and compare the error $\mu^{\mathrm{LP}}_{f,p}$ against $\mu^{opt}_{f,p}$. In particular, consider density four in Marron and Wand [1992b]

$$f_4(x) \;=\; \frac{2}{3}\varphi(x;0,1) + \frac{1}{3}\varphi(x;0,.10).$$

This is a mixture of two normal distributions $\varphi(x;\mu,\sigma)$ with mean μ and standard deviation σ. Figure 4.1 shows the density and the HDRs considered here.

Next, we simulate 250 Monte Carlo samples of $1,000$ observations each from density f_4. We then compare the performance of the estimators $\widehat{\mathrm{HDR}}_{p,h_{\mathrm{LP}}}$ and $\widehat{\mathrm{HDR}}_{p,opt}$. We show the results in Figure 4.2. Every circle in the plot is a sample by sample comparison between the estimators where $\mu^{\mathrm{LP}}_{f,p}$ is on the x-axis and $\mu^{opt}_{f,p}$ on the y-axis. In the figure we also include density plots for the logarithm of the ratio $\mu^{\mathrm{LP}}_{f,p}/\mu^{opt}_{f,p}$. For each p we ran a Wilcoxon test for the paired errors. We observed that for $p = .20$ the errors are not statistically different. For $p = 0.50, 0.80$, the estimator $\widehat{\mathrm{HDR}}_{p,opt}$ yields a statistically lower error compared to $\widehat{\mathrm{HDR}}_{p,h_{\mathrm{LP}}}$. These results coincide with those from example one in Section 3.5. The local polynomial

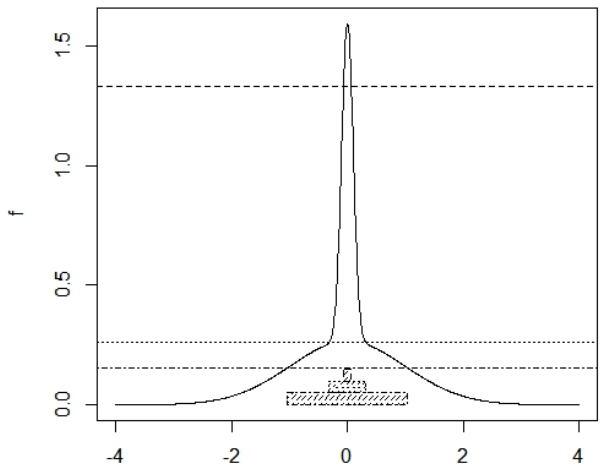

Figure 4.1: Plot for density $f_4(x)$. We included the true HDRs for $p = 0.20, 0.50, 0.80$.

kernel estimator does not perform well for larger values of p (or λ in the regression problem). In the HDR problem we observe larger errors for $\widehat{\text{HDR}}_{p,h_{\text{LP}}}$. In the level set estimation problem we observe that the local polynomial returns an empty set in almost all samples considered. In both cases we notice that the bandwidth h_{LP} returns an estimator that does not capture the peak in the density (or regression function) correctly.

We carry out similar simulations to the remaining nine densities and compare both estimators in every case. We consider $p = 0.20, 0.50, 0.80$ as before. In Table 4.1 we present all the Wilcoxon tests that we conducted. We applied the Holm-Bonferroni correction to account for multiple comparisons within each density. The estimator $\widehat{\text{HDR}}_{p,opt}$ proposed in Samworth and Wand [2010] offers an advantage in 15 out of the 30 paired comparisons. Samworth and Wand [2010] reached a similar conclusion when comparing the plug-in estimator using their bandwidth selector h_{opt} against the plug-in estimator using the least-squares cross validation optimal bandwidth. In Section 3.5 we apply the ideas in Samworth and Wand [2010] to the regression setting. Instead of a density, we consider a regression function. We observed that

the optimal bandwidth h_{opt} proposed in Section 3.3 offers an advantage for some functions, but this does not occur uniformly.

In the next section we study an alternative plug-in estimator. Instead of estimating the density in HDR_p with a kernel estimator, we estimate it with the log-concave mixture model. This is a non parametric estimator that does not require a smoothing parameter as is the case for the kernel estimator. For this estimator we assume that the true density comes from a mixed model where the marginal densities are log-concave.

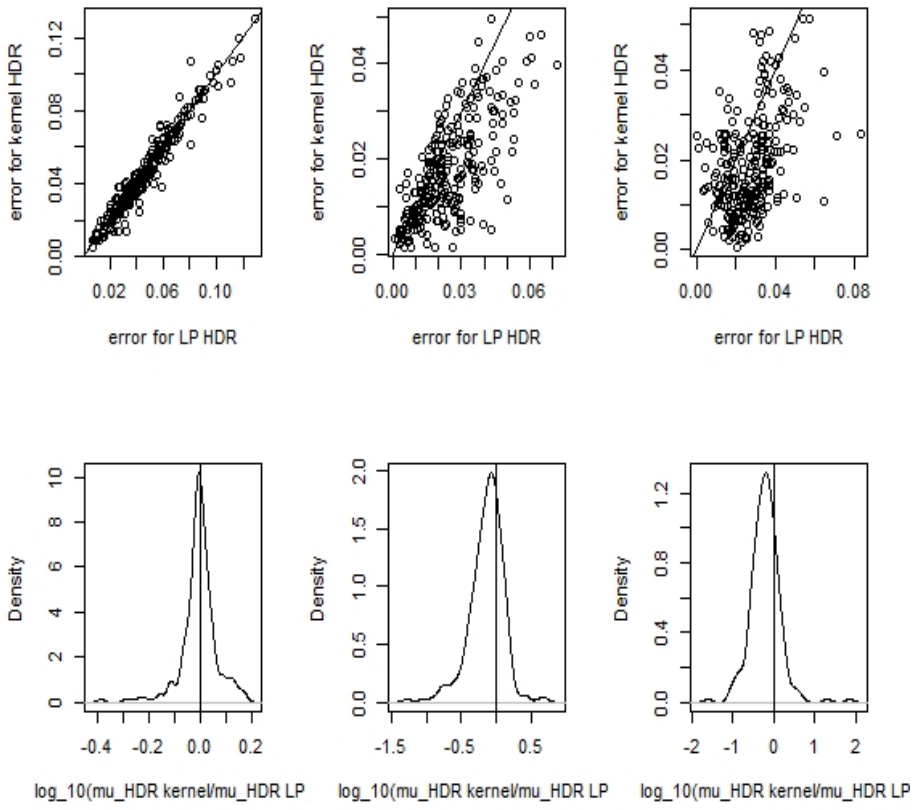

Figure 4.2: Sample by sample comparison between $\widehat{\text{HDR}}_{p,h_{opt}}$ and $\widehat{\text{HDR}}_{p,h_{\text{LP}}}$ for $f_4(x)$. We graph $\mu^{\text{LP}}_{f,p}$ on the x-axis and $\mu^{opt}_{f,p}$ on the y-axis. In the figure we also include density plots for the logarithm of the ratio $\mu^{\text{LP}}_{f,p}/\mu^{opt}_{f,p}$. We considered (from left to right) the values of $p = 0.20, 0.50, 0.80$. Here we simulate 250 Monte Carlo samples of 1,000 observations each.

4.2 A simulation study on the log-concave highest density region estimator

Consider the model

$$f(x) = \sum_{j=1}^{m} \pi_j f_j(x)$$

where an observation $x \in \mathbb{R}$ comes from a mixture of m densities $f_j(x)$ for $j = 1, \ldots, m$. Assume that each of these densities is log-concave. This is called the log-concave mixture model (LCMM)

$$f(x) = \sum_{j=1}^{m} \pi_j \exp \varphi_j(x)$$

The goal is to estimate the number of components m, the mixing proportions $\pi_1, \ldots, \pi_m$ such that $\sum_{j=1}^{m} \pi_j = 1$ and the concave functions $\varphi_1, \ldots, \varphi_m$.

Estimation of the log-concave mixed model

For a fixed number of components, the mixing proportions $\pi_1, \ldots, \pi_m$, as well as the concave functions are estimated using the EM algorithm. In the expectation step $\widehat{\pi}_j$ are obtained as,

$$\begin{aligned} \widehat{w}_{ij}^{(t)} &= \frac{\widehat{\pi}_j^{(t-1)} f_j\left(x_i | \widehat{\lambda}^{(t-1)}\right)}{\sum_{j=1}^{m} \widehat{\pi}_j^{(t-1)} f_j\left(x_i | \widehat{\lambda}^{(t-1)}\right)} \\ \widehat{\pi}_j^{(t)} &= \frac{\sum_{i=1}^{n} \widehat{w}_{ij}^{(t)}}{n} \end{aligned}$$

where $f_j\left(x | \widehat{\lambda}^{(t-1)}\right)$ is the log-concave maximum likelihood estimator at the previous iteration. In the maximization step, the set of parameters for the log-concave density, that is $\widehat{\varphi}_j$ are updated using the log-concave maximum likelihood estimator which can be computed using the **logcondens** R-package. Note that we need initial values for the mixing proportions $\pi_j^{(0)}$ $j = 1, \ldots, m$ to run the EM algorithm.

One popular criteria for model selection is the Bayesian information criteria (BIC). It is derived from the posterior probability of the model given the data set. Let λ be the parameters of the model, the BIC criterion is written as

$$\text{BIC} \quad = \quad -2l(\widehat{\lambda}_{\text{MLE}}) + |\lambda| \ln(n)$$

where $\widehat{\lambda}_{\text{MLE}}$ is the maximum likelihood estimator of λ and $|\lambda|$ is the number of components. Under the LCMM, Boonpatcharanon [2019] proposed a new criterion called the log-concave BIC

$$\text{LCBIC} \quad = \quad -2l(\widehat{\varphi}_{\text{MLE}}) + |\theta| \ln(n) - \sum_{j=1}^{k} \ln(\pi_j) + \sum_{j=1}^{k} \ln |I(\widehat{\varphi}_j)|$$

where $|\theta|$ is the number of free parameters in the model and $I(\widehat{\varphi}_j)$ is the Fisher's information matrix of the j^{th} component. The authors thus use this criterion to choose the number of components m and then estimate the parameters iteratively with the EM algorithm summarized above. We use this technique to estimate the highest density region discussed in Section 2.3.4.

A Simulation study

Recall that for a density f and a probability p, the highest density region (HDR) is the set

$$\text{HDR}_p \quad = \quad \{x : f(x) > \lambda_p\}$$

where

$$\lambda_p \quad = \quad \inf \Big\{ \lambda \in (0, \infty) : \int_{-\infty}^{\infty} f(x) \mathbb{1}_{\{f(x) \geq \lambda\}} dx \leq 1 - p \Big\}.$$

We study the plug-in estimator where f is estimated with the LCMM estimator

$$\widehat{f}_{\mathrm{LC}}(x) \quad = \quad \sum_{j=1}^{\widehat{m}} \widehat{\pi}_j \widehat{f}_j(x).$$

Here $\widehat{f}_j(x)$ is the log-concave maximum likelihood estimator for $j = 1, \ldots, \widehat{m}$. The log-concave HDR estimator is thus the set

$$\widehat{\mathrm{HDR}}_{\mathrm{LC},p} \quad = \quad \{x : \widehat{f}_{\mathrm{LC}}(x) > \widehat{\lambda}_p\}$$

with

$$\widehat{\lambda}_p \quad = \quad \inf\left\{\lambda \in (0,\infty) : \int_{-\infty}^{\infty} \widehat{f}_{\mathrm{LC}}(x) \mathbb{1}_{\{\widehat{f}_{\mathrm{LC}}(x) \geq \lambda\}} dx \leq 1 - p\right\}.$$

In the simulation study we consider density four and density six in Marron and Wand [1992b]

$$\begin{aligned} f_4(x) &= \frac{2}{3} N(0,1) + \frac{1}{3} N\left(0, \frac{1}{10}\right) \\ f_6(x) &= \frac{1}{2} N\left(-1, \frac{2}{3}\right) + \frac{1}{2} N\left(1, \frac{2}{3}\right). \end{aligned}$$

Both densities are a mixture of log-concave densities. In addition, we study a misspecification case. We consider a mixture of T densities. Let $T(x; d)$ be the t-distribution with d degrees of freedom. We consider

$$f_T(x) \quad = \quad \frac{2}{3} T(x; 5) + \frac{1}{3} T(x; 2).$$

Here we consider the probabilities $p = 0.20, 0.50, 0.80$ in accordance to the levels considered in Samworth and Wand [2010]. The true HDRs for densities f_4, f_6 and f_T appear in Figure 4.3. For every level p, we simulated 250 Monte Carlo samples. For density f_4 and f_T, the size of each sample is of 1000 observations. For density f_6, 5000. We increase the sample size for f_6

to improve the estimation of m.

Let $\widehat{\text{HDR}}_{h_{opt},p}$ be the kernel estimator in Samworth and Wand [2010], reviewed in Section 2.3.4. Recall that this is the plug-in estimator for the set HDR_p where f is estimated with a kernel density estimator and h_{opt} is the optimal bandwidth proposed in Samworth and Wand [2010]. For the i^{th} sample, let $\widehat{\text{HDR}}^{[i]}_{\text{LC},p}$ be the log-concave HDR estimator. Then we compute its error

$$\mu_f\left(\widehat{\text{HDR}}^{[i]}_{\text{LC},p} \triangle \text{HDR}_p\right) \quad = \quad \int_{\widehat{\text{HDR}}^{[i]}_{\text{LC},p} \triangle \text{HDR}_p} f(x)dx.$$

We repeat this step but with $\widehat{\text{HDR}}^{[i]}_{h_{opt},p}$ instead. Next, we compare both errors for each of the 250 Monte Carlo samples. We present the results in Figure 4.4, Figure 4.5 and Figure 4.6 where each point represents a sample by sample comparison between the two estimators. Each figure also includes density plots for the logarithm of the estimators errors' ratio. We observe that for some samples, the number of components in the LCMM was incorrectly estimated. Therefore, we mark these points in red and included an additional curve in each density plot. This additional curve (dashed line) contains only the samples for which the number of components are correctly estimated. The greatest difference between the curve that contains all samples and the curve that contains only those with the correct estimation of m is observed for f_T in Figure 4.6 as expected. For f_4 only in 6 out of 250 samples m was incorrectly estimated. We complement the plots just described with boxplots. We divide again the results from the log-concave HDR estimator depending if m was correctly estimated (LCHDR (c)) or not (LCHDR (i)).

From the plots we observe that when m is correctly estimated, the performance of the log-concave HDR estimator has significant improvement. We thus repeat the simulations described previously. For all densities considered we exchange $\widehat{m}$ for the true value $m = 2$. We also consider samples with 300 observations and 1000 observations. We present the results for density f_4 in Figure 4.9, for density f_6 in Figure 4.10 and for the misspecified model f_T in Figure 4.11.

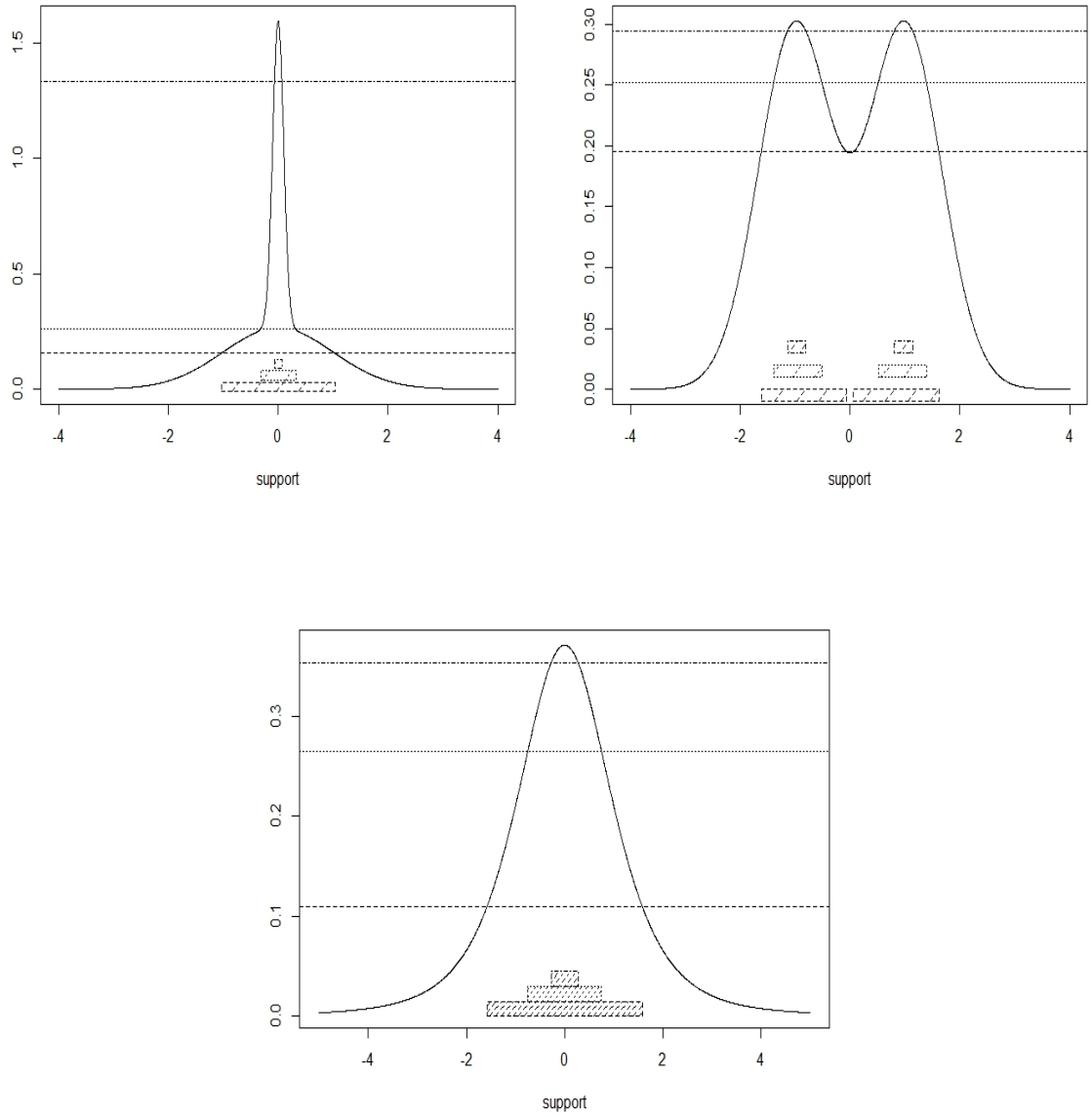

Figure 4.3: HDR for three densities. At the top of the figure: $f_4(x)$ and $f_6(x)$. At the bottom, the mixture of t-distributions f_T. We show the HDRs when $p = .20$ (dashed line), $p = .50$ (dotted line) and $p = .80$ (dash-dotted line).

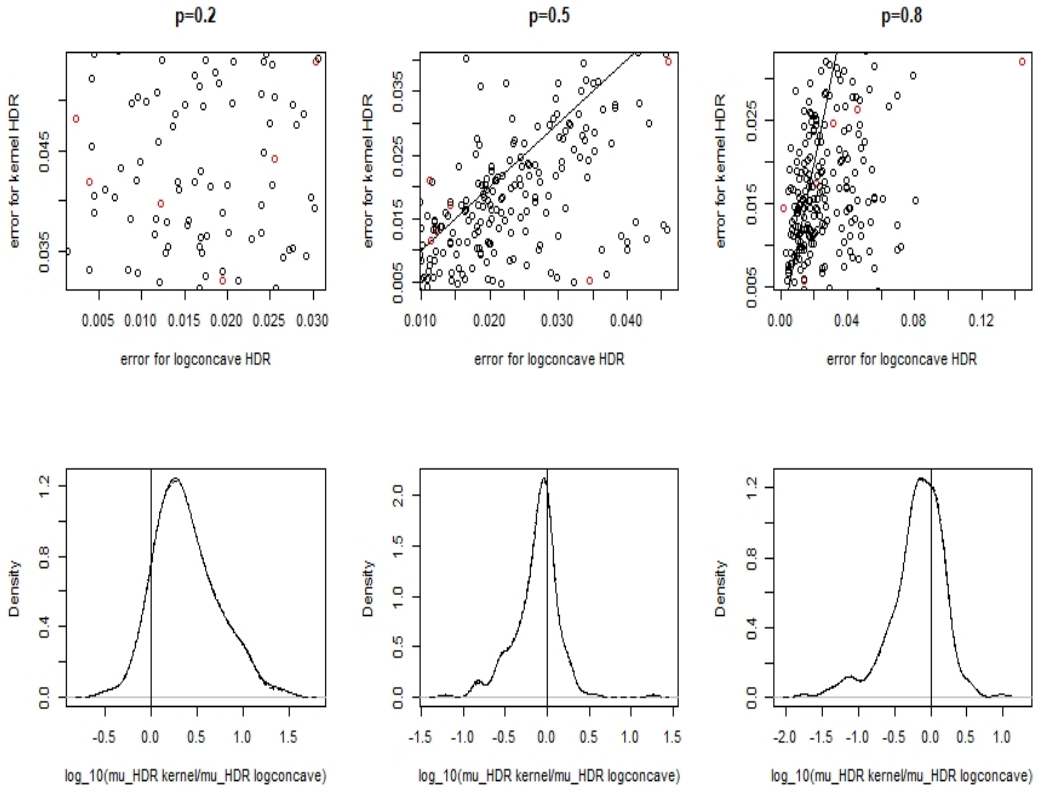

Figure 4.4: Comparison between the log-concave HDR and the kernel HDR estimators for $f_4(x)$. We classify the errors for the log-concave HDR estimators based on the estimator of the number of components in the LCMM ($\widehat{m}$). The label LCHDR (c) and the points in black refer to the log-concave HDR with m correctly estimated, while LCHDR (i) and red points refer to the log-concave HDR with m incorrectly estimated. We show the results for 250 Monte Carlo samples of 1000 observations each and three probabilities $p = 0.20, 0.50, 0.80$. For each probability we include at the bottom a density plot for the logarithm of the ratio of the kernel HDR estimate and the LCHDR estimate.

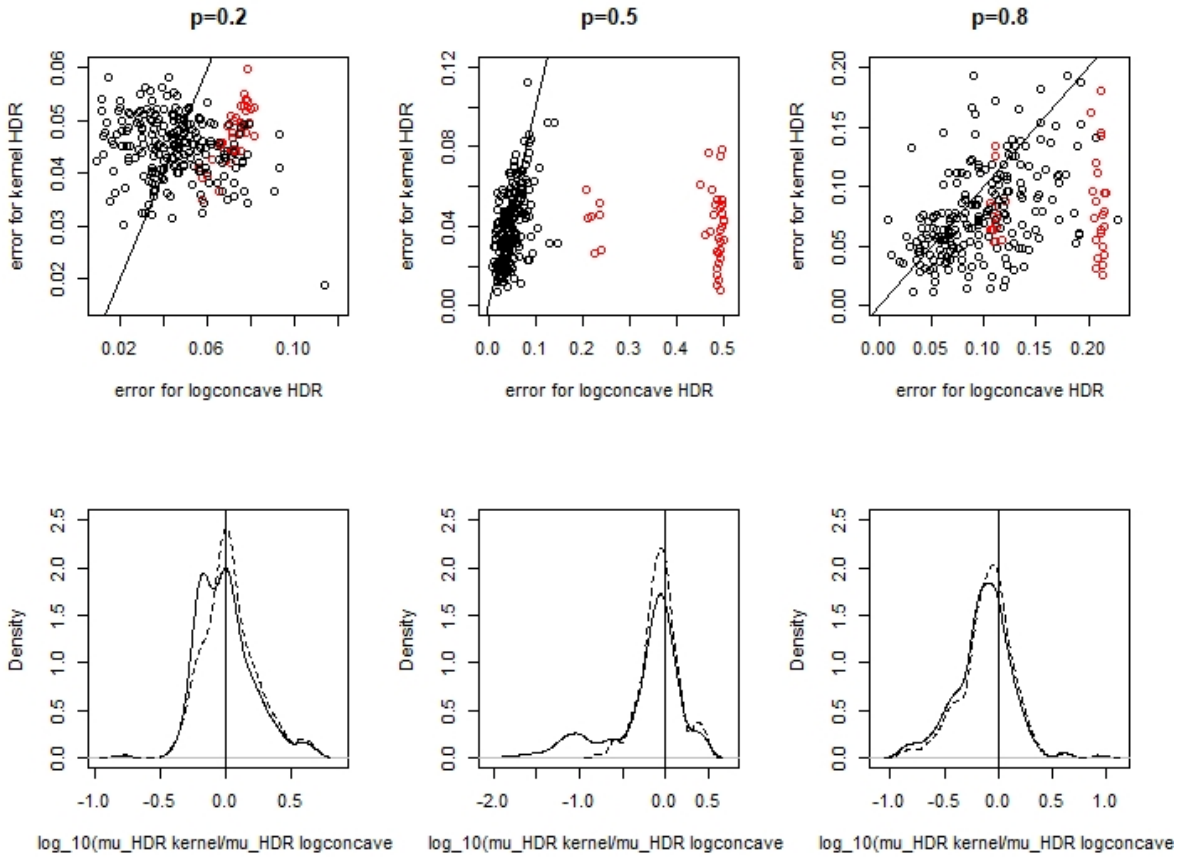

Figure 4.5: Comparison between the log-concave HDR and the kernel HDR estimators for $f_6(x)$. We classify the errors for the log-concave HDR estimators based on the estimator of the number of components in the LCMM ($\widehat{m}$). The label LCHDR (c) and the points in black refer to the log-concave HDR with m correctly estimated, while LCHDR (i) and red points refer to the log-concave HDR with m incorrectly estimated. We show the results for 250 Monte Carlo samples of 5000 observations each and three probabilities $p = 0.20, 0.50, 0.80$.For each probability we include at the bottom a density plot for the logarithm of the ratio of the kernel HDR estimate and the LCHDR estimate.

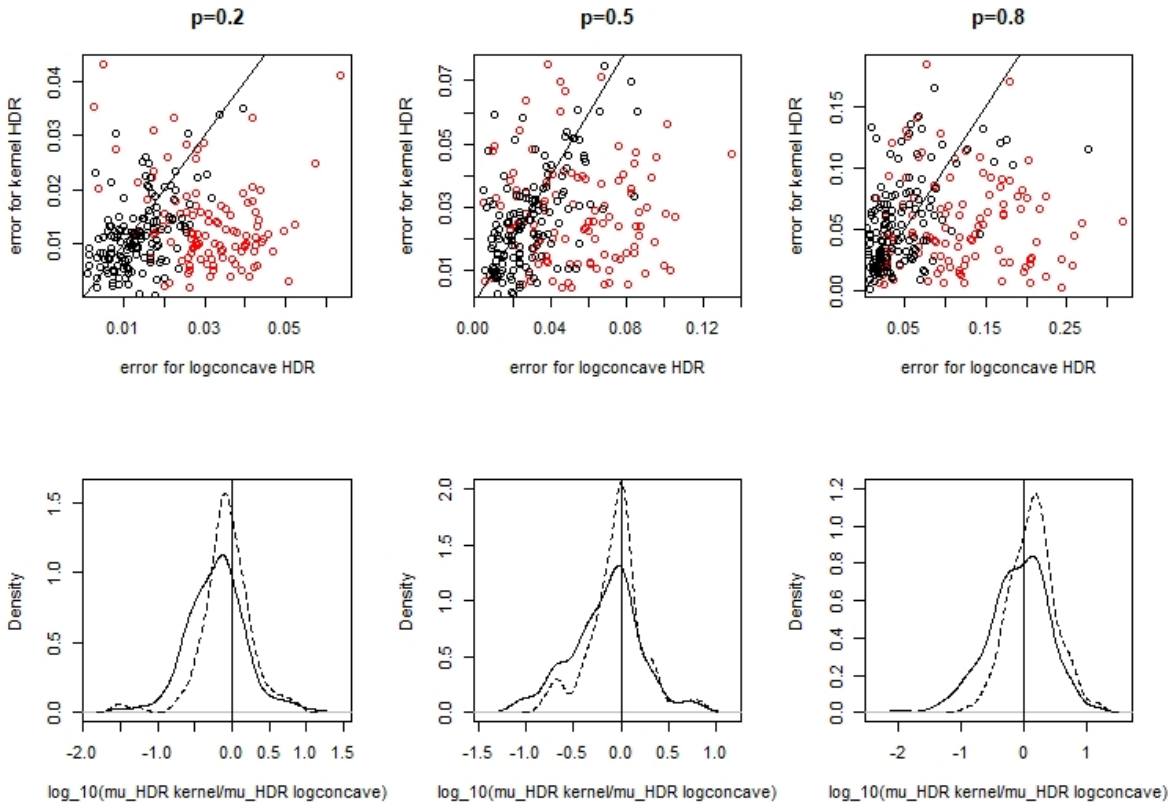

Figure 4.6: Comparison between the log-concave HDR and the kernel HDR estimators for the misspecification case f_T. We classify the errors for the log-concave HDR estimators based on the estimator of the number of components in the LCMM ($\widehat{m}$). The label LCHDR (c) and the points in black refer to the log-concave HDR with m correctly estimated, while LCHDR (i) and red points refer to the log-concave HDR with m incorrectly estimated. We show the results for 250 Monte Carlo samples of 1000 observations each and three probabilities $p = 0.20, 0.50, 0.80$.For each probability we include at the bottom a density plot for the logarithm of the ratio of the kernel HDR estimate and the LCHDR estimate.

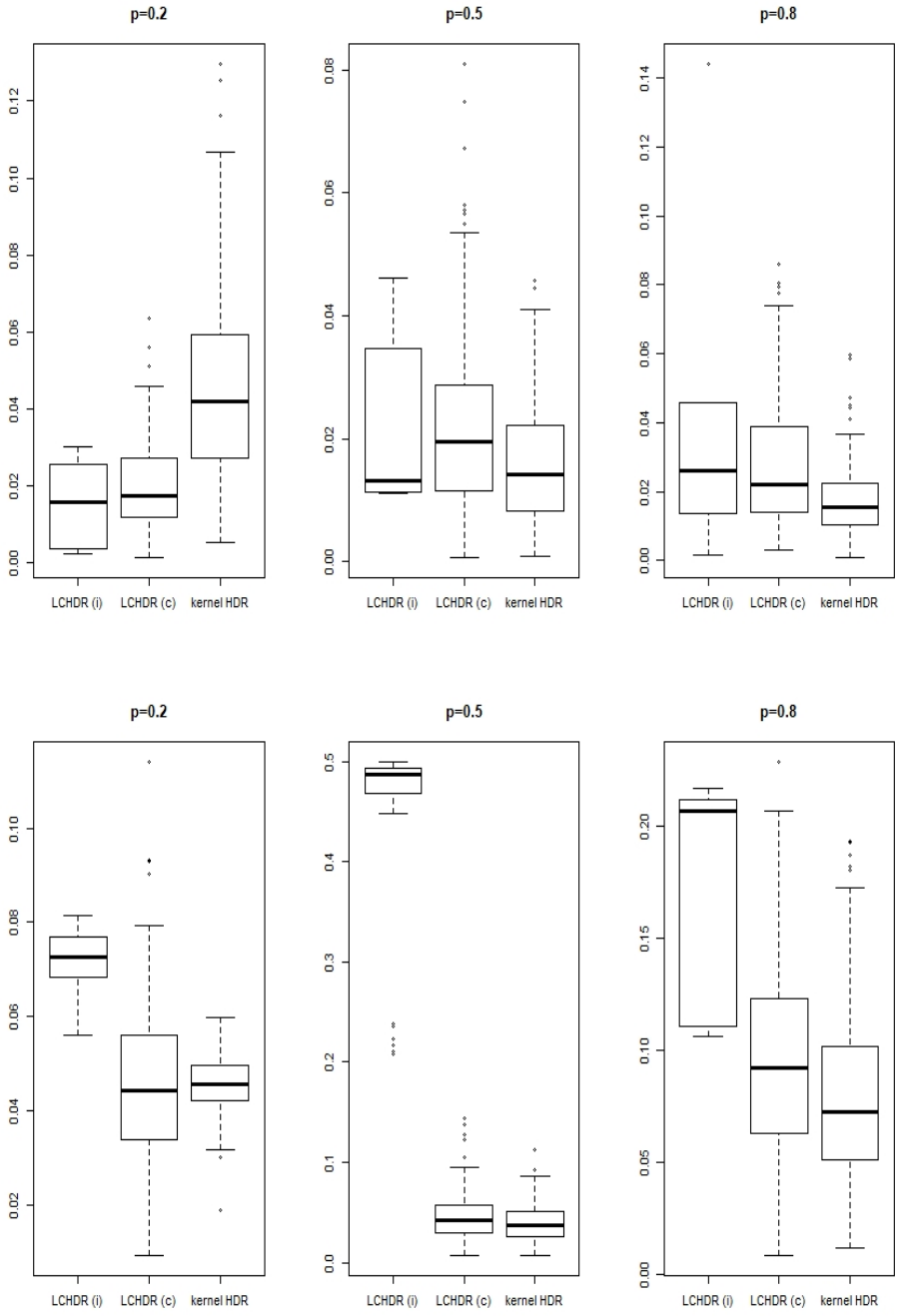

Figure 4.7: Boxplots for the errors that result from the estimators $\widehat{\text{HDR}}_{LC,p}$ and $\widehat{\text{HDR}}_{h_{opt},p}$. The plot at the top shows the results for 250 Monte Carlo samples of 1000 observations each from density f_4. The plot at the bottom shows the results for 250 Monte Carlo samples of 5000 observations each from density f_6. For both cases we consider the probabilities $p = 0.20, 0.50, 0.80$. We divide the results for the log-concave HDR estimator into two groups: LCHDR(c) represents the errors when m in the corresponding LCMM is correctly estimated, the label LCHDR(i) represents the errors when it is not.

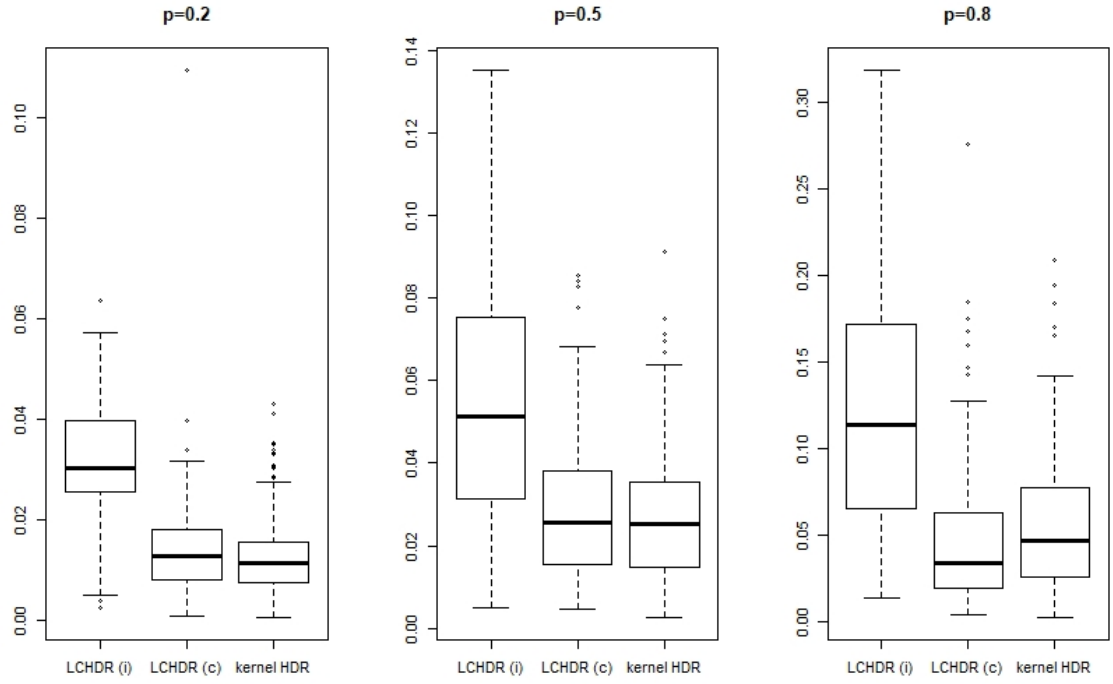

Figure 4.8: Boxplots for the errors that result from the estimators $\widehat{\text{HDR}}_{LC,p}$ and $\widehat{\text{HDR}}_{h_{opt},p}$. The plot shows the results for 250 Monte Carlo samples of 1000 observations each from the misspecified model f_T. We consider the probabilities $p = 0.20, 0.50, 0.80$. We divide the results for the log-concave HDR estimator into two groups: LCHDR(c) represents the errors when m in the corresponding LCMM is correctly estimated, the label LCHDR(i) represents the errors when it is not.

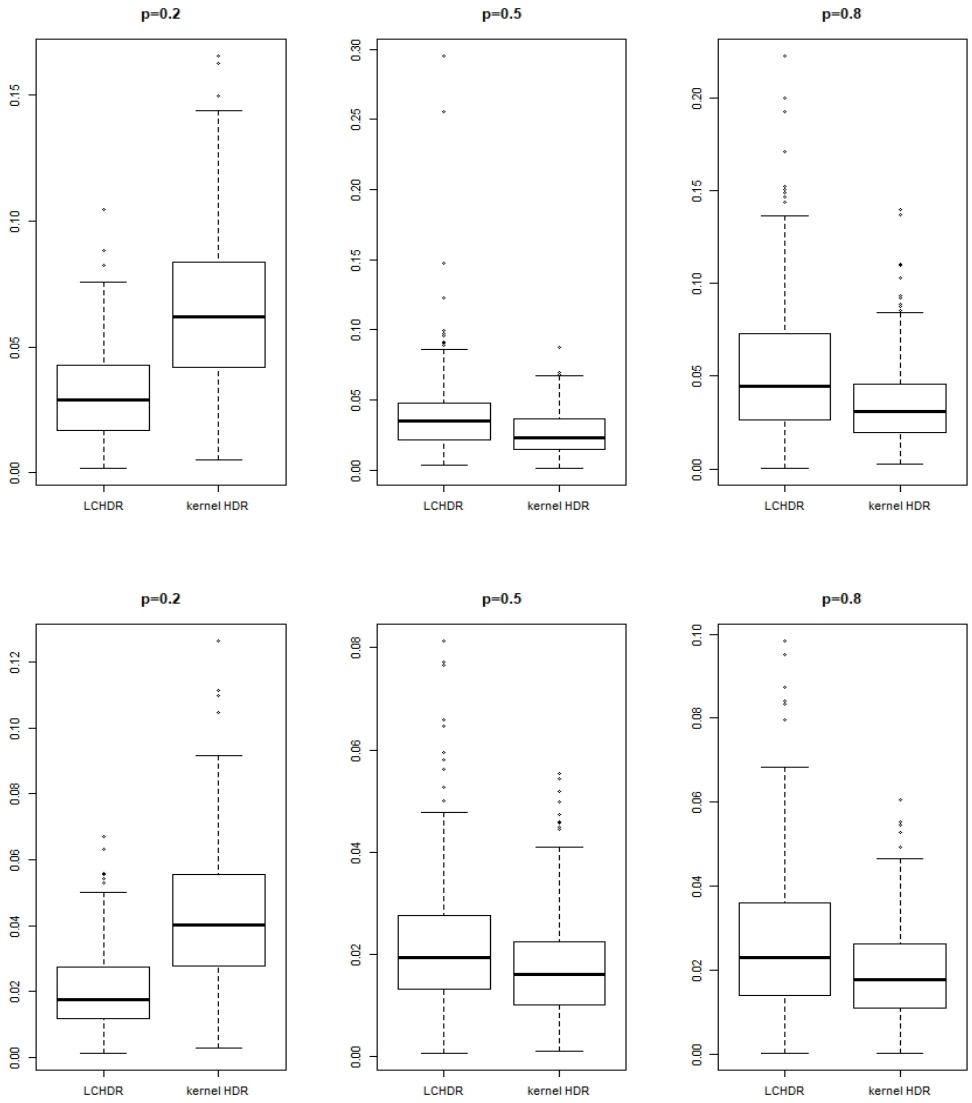

Figure 4.9: Boxplot comparisons between the log-concave HDR estimator (LCHDR) and the kernel HDR estimator for f_4. We assume the number of components m is known. The top plot summarizes the estimation errors for 250 Monte Carlo samples with 300 observations. The plot below summarizes the estimation errors for 250 Monte Carlo samples with 1000 observations. We considered the probabilities $p = 0.20, 0.50, 0.80$ (from left to right).

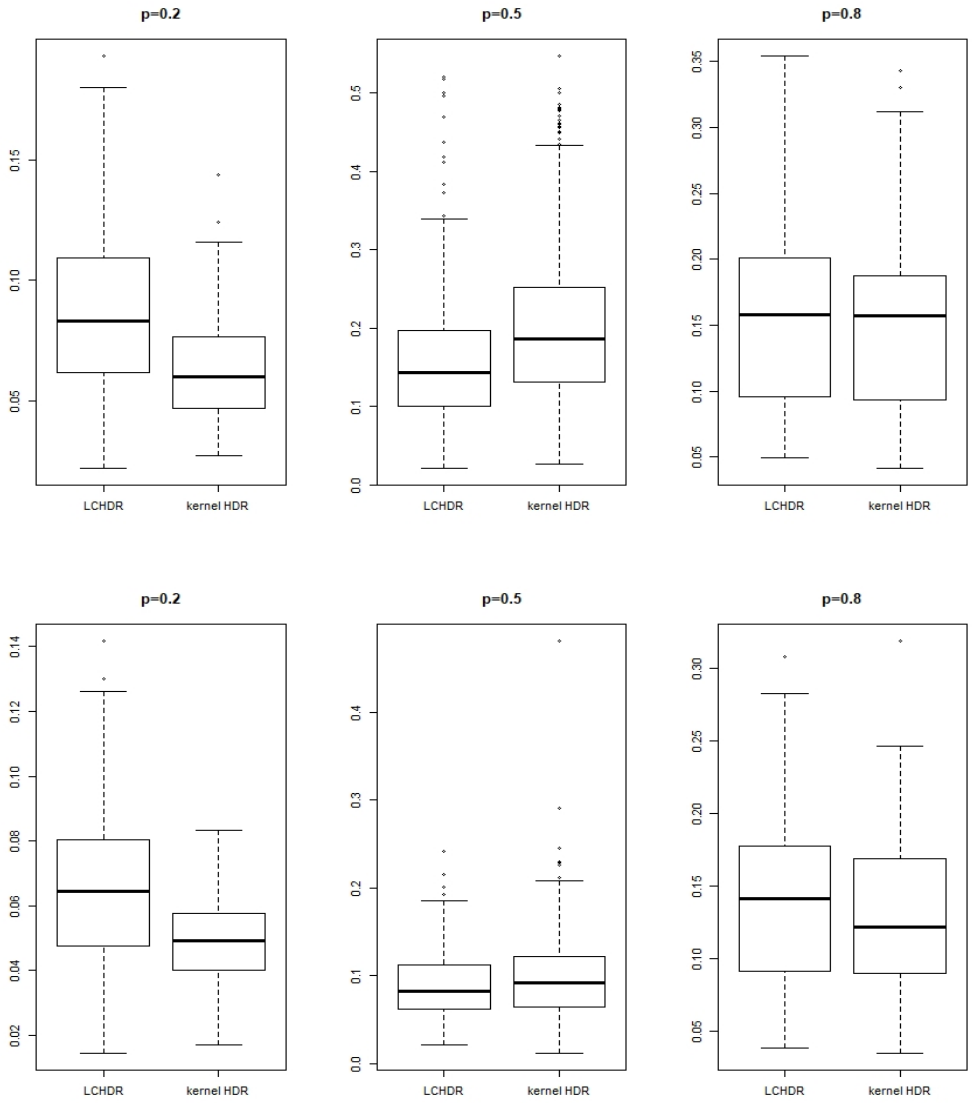

Figure 4.10: Boxplot comparisons between the log-concave HDR estimator (LCHDR) and the kernel HDR estimator for f_6. We assume the number of components m is known. The top plot summarizes the estimation errors for 250 Monte Carlo samples with 300 observations. The plot below summarizes the estimation errors for 250 Monte Carlo samples with 1000 observations. We considered the probabilities $p = 0.20, 0.50, 0.80$ (from left to right).

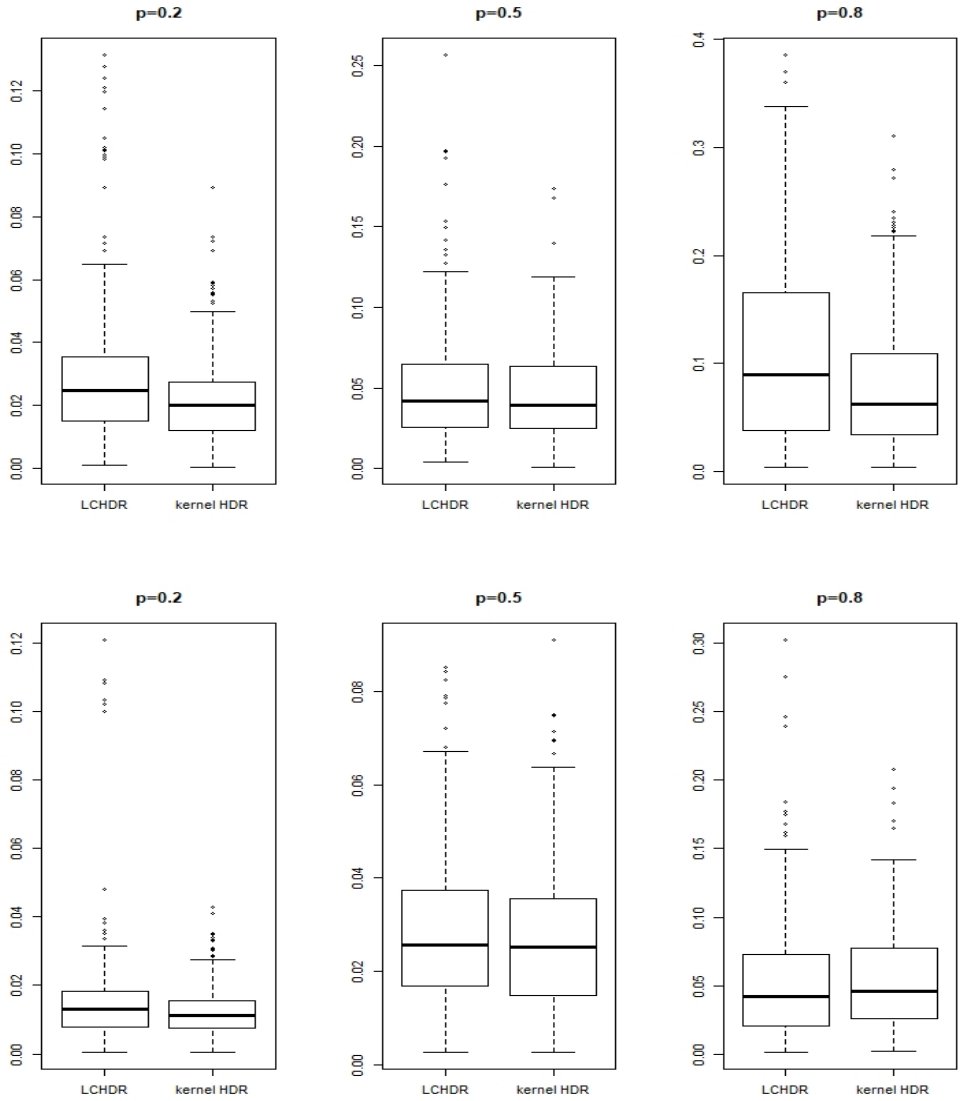

Figure 4.11: Boxplot comparisons between the log-concave HDR estimator (LCHDR) and the kernel HDR estimator for the misspecified model f_T. We assume the number of components m is known. The top plot summarizes the estimation errors for 250 Monte Carlo samples with 300 observations. The plot below summarizes the estimation errors for 250 Monte Carlo samples with 1000 observations. We considered the probabilities $p = 0.20, 0.50, 0.80$ (from left to right).

We notice from the results in Figure 4.9 and Figure 4.10 that when the model was correctly specified, the log-concave HDR estimator offers some advantage to the kernel HDR estimator for small values of p. For larger values of p the log-concave HDR estimator performs similar to the kernel estimator. For the misspecified model (the mixture of t-distributions), when m is known, the log-concave HDR estimator's performance decreased for samples of 300 observations. However, for larger samples (1000 observations), the performance of the log-concave HDR is similar to the kernel HDR. Our results suggest that there is a need to investigate on other estimators for m (the number of components in the LCMM). However, we think this problem is less complex compared to the bandwidth selection problem for kernel estimators. For this reason, we consider $\widehat{\text{HDR}}_{LC,p}$ as a valid alternative estimator to the HDR plug-in kernel estimator.

4.3 A study on daily temperatures in Melbourne, Australia

We study the data set *maxtemp* available in the **hdrcde** R-package. The data set consists of the daily maximum temperatures observed in Melbourne Australia between 1981 and 1990.

Conditional on today's maximum temperature, we consider tomorrow's temperature for analysis. We group the data based on today's temperature. We divide the range of today's maximum temperature into seven intervals. For the first six we create intervals of 5 degrees each, starting at 5 degrees Celsius. That is $[5, 10), [10, 15), \ldots, [30, 35)$. For the seventh interval we consider today's temperatures within $[35, 45)$ degrees to guarantee sufficient observations.

Figure 4.12 compares the log concave estimates for the $20\%, 50\%$ and 80% HDRs against the kernel HDR estimates using the optimal bandwidth proposed in Samworth and Wand [2010] and reviewed in Section 2.3.4. The kernel HDR shows that tomorrow's temperature, conditional on today's has a bimodal density when today's temperature is within $[30, 35)$ or $[35, 45)$ degrees Celsius. The log-concave HDR does not show this bimodality. However, the corresponding support for each HDR is visibly longer than the kernel HDR estimate. This suggests that the density estimator from the LCMM is smoother than the kernel density

estimator with the bandwidth proposed in Samworth and Wand [2010].

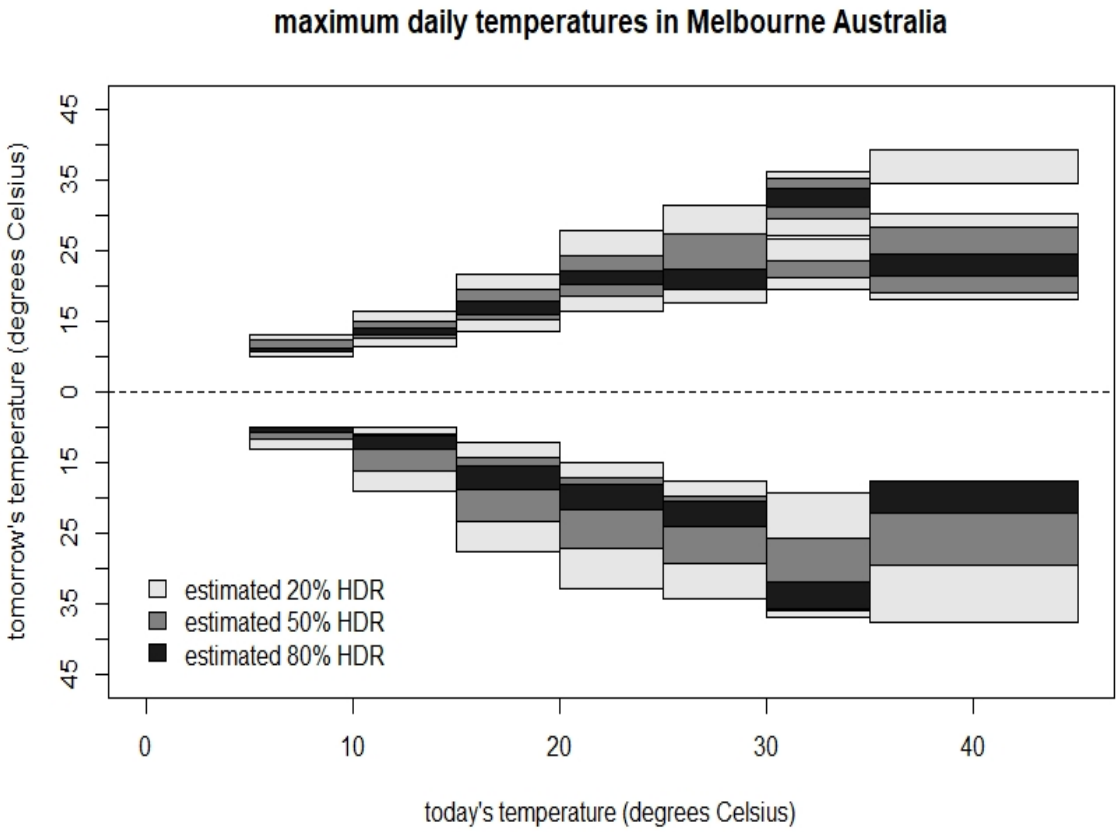

Figure 4.12: Comparison between the kernel and log-concave estimates for the 20%, 50% and 80% HDRs for the conditional density of tomorrow's maximum temperatures given today's. The plot at the top corresponds to the kernel HDRs. The plot at the bottom shows the log-concave HDRs. We reflect the latter about the x-axis.

density	p	proportion $\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	Wilcoxon test H_o: $\mu_{f,p}^{\text{LP}} = \mu_{f,p}^{opt}$ alternative hypothesis	adj. p-val
1	0.20	0.268	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	1.191e-12
1	0.50	0.412	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	1.410e-04
1	0.80	0.712	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	9.708e-12
2	0.20	0.188	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	6.6e-16
2	0.50	0.340	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	1.117e-09
2	0.80	0.420	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	4.093e-04
3	0.20	0.556	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	3.991e-04
3	0.50	0.400	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	1.27e-05
3	0.80	0.696	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
4	0.20	0.468	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	0.663
4	0.50	0.272	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	6.6e-16
4	0.80	0.236	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	6.6e-16
5	0.20	0.288	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	7.884e-12
5	0.50	0.428	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	5.453e-04
5	0.80	0.852	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
6	0.20	0.716	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	8.862e-12
6	0.50	0.692	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	1.053e-14
6	0.80	0.512	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	0.081
7	0.20	0.260	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	6.6e-16
7	0.50	0.336	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	9.892e-07
7	0.80	0.644	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	1.802e-04
8	0.20	0.396	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	5.447e-05
8	0.50	0.844	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
8	0.80	0.304	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	1.139e-10
9	0.20	0.956	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
9	0.50	0.748	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
9	0.80	0.552	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	1.965e-04
10	0.20	1.000	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
10	0.50	0.772	$\mu_{f,p}^{\text{LP}} < \mu_{f,p}^{opt}$	6.6e-16
10	0.80	0.308	$\mu_{f,p}^{opt} < \mu_{f,p}^{\text{LP}}$	1.882e-09

Table 4.1: Summary of the results for density $f_4(x)$. We summarize the results for 250 Monte Carlo samples of 1000 observations each. We present the proportion of the samples for which the error from the local polynomial estimator is less than $\mu_{f,p}^{opt}$. We also include the p-values for one-sided Wilcoxon rank tests for the paired errors. We applied the Holm-Bonferroni correction to account for multiple comparisons within each density.

Chapter 5

Conclusions

In the first part of this thesis we study the optimal bandwidth selector for the plug-in kernel estimator of a level set adapted to regression. In particular, we focus on the Nadaraya-Watson kernel regression estimator in one dimension. We present a local loss function as an alternative to the L_2 metric in the same spirit as Samworth and Wand [2010] and Doss and Weng [2018]. We then derive an asymptotic approximation of its corresponding risk and show that the rate of the level set optimal bandwidth $h_{opt} = c_{opt} n^{-1/5}$ coincides with the rate from traditional global bandwidth selectors. However, the constant c_{opt} is far different from them. Because the optimal bandwidth has no closed solution, we derive an algorithm to estimate the constant c_{opt}. Thus, the estimator $\widehat{c}_{opt}$ yields our practical bandwidth selector. We then study the performance of this practical bandwidth selector through simulations. Our simulations show that in general, for small samples and small values of λ, the level set optimal bandwidth shows improvement in estimating the set LS_λ when compared to the cross validation bandwidth selection or the local polynomial kernel estimator.

As in Samworth and Wand [2010]; Doss and Weng [2018], there are occasions where improvement is not seen with our method.

In the second part of this thesis, motivated by our simulation findings and the relationship of the level set estimation to the HDR, we study via simulations the properties of a plug-in estimator where the density is estimated with the log-concave mixed model. This is a

mixture model with marginal densities that are log-concave. In particular, the log-concave maximum likelihood estimator (Dümbgen and Rufibach [2009]; Cule et al. [2010]) requires no estimation of tuning parameters which makes it attractive in practice. Our simulations show that when the number of components in the log-concave mixed model is correctly specified, the log-concave plug-in estimator performs better than the kernel estimator for lower levels and similarly for the rest of the levels considered. Therefore, adding shape-constrains to the density removes the computational burden of bandwidth selection without decreasing the accuracy.

There are interesting alternative approaches for extending the work here presented. First, for level set estimation in the context of regression, one could consider an alternative estimator for g such as wavelets. Wavelets are locally adaptive functions that adjust well to sudden changes in smoothness in g. Although it is true that this type of estimator needs a large signal to noise ratio, for the sample sizes considered in Samworth and Wand [2010], this assumption is reasonable. For a multivariate model, a semiparametric estimator is a feasible path for extending the ideas in this thesis. Moreover, a semiparametric estimator is easier to interpret because of its parametric component. Next, in the context of the HDR problem adding shape constrains decreases the computational complexity and avoids the problem of bandwidth selection. However, from our simulations we observe there is an opportunity to develop a new methodology to estimate the number of components in a log-concave mixture model more accurately. As we discussed before, when the number of components is estimated correctly, the log-concave plug-in estimator offers an advantage over the kernel estimator. In addition, we believe that selecting the number of components in the log-concave mixture model is a less complex problem than that of bandwidth selection.

Appendix A

Proofs of main results

A.1 Proof of Theorem 1

Assumptions

Let $g = E[Y|X = x]$ with $g_2(x) = E[Y^2|X = x]$. In order to prove our results, we make the following assumptions.

(F) The support of the density f, C_X is compact. Furthermore, $\inf_{x \in C_X} f(x) \geq b_f > 0$.

(G) For an arbitrary λ, we assume there exist finitely many points $x_1 < \ldots < x_m$ such that $g(x_j) = \lambda$ and $g'(x_j) \neq 0$ and that these x_j for $j = 1, ..., m$ all lie in the interior of C_X. To simplify the exposition in the proof, we assume that there are an even number of points. That is $m = 2r$ with $g'(x_{2j-1}) > 0$ and $g'(x_{2j}) < 0$. We also define $x_0 = \inf\{x \in C_X\}$ and $x_{2r+1} = \sup\{x \in C_X\}$.

(K) The kernel K is a non-negative, symmetric density and, defining $v_{k,m} = \int |s^m| K^k(s) ds$, satisfies $v_{1,3}, v_{2,2}, v_{3,4}, v_{4,0}$ are all finite.

(B) We assume that the bandwidth $h = h_n \to 0$ as $n \to \infty$. In addition, it is such that nh^5 is at most $O(1)$, and that $n^3 h^{11} \log^{-8} n \to \infty$.

(S) We make the following smoothness assumptions on the underlying functions.

local: Let $\widetilde{B} = \cup_{j=1}^{m} B_\eta(x_j)$ where $B_\eta(x)$ denotes a ball of size η around x. There exists an $\eta > 0$ such that the function q is $C_1(\widetilde{B})$, while f, g_2 are $C_1(\widetilde{B})$ and g is $C_2(\widetilde{B})$.

global: Both f and g are continuous on C_X, and q is an integrable function on C_X.

moment: Define the function $m_k(x) = E[(Y-\lambda)^k | X = x] f(x)$ and assume that there exists an $M_k(x)$ such that for $k = 2, 4$

$$\lim_{h \to 0} \sup_{s \in (x - C_X)/h} m_k(x - hs) \quad \leq \quad M_k(x)$$

with $\int_{C_X} M_2^2(x) q(x) dx < \infty$ and $\int_{C_X} M_4(x) q(x) dx < \infty$.

The proof of our main result is handled similarly to Samworth and Wand [2010], Doss and Weng [2018]. The main idea is to first show that the integral in the risk function is dominated by a neighbourhood of each point x_j. Second, in the neighbourhood of each point x_j, the formula is derived by applying the central limit theorem, which allows us to replace the probabilities with Gaussian distribution functions and the final result follows from integration by parts. For the most part, our assumptions are similar to those of Samworth and Wand [2010], Doss and Weng [2018]. In particular, we note that the smoothness assumptions (S1) on the functions g_2, g, f are all one derivative stronger than those needed in the statement of the theorem. This was also required in Samworth and Wand [2010], Doss and Weng [2018], and here it is necessary in order to achieve the correct integrated error in the final expression.

The main difference in our approach is that in Samworth and Wand [2010], Doss and Weng [2018] the first step of the proof (as described above) relies on a concentration inequality for kernel density estimators. The counterpart of this for the Nadaraya-Watson estimator appears in Vogel and Schettler [2013]. Alas, the theorem in Vogel and Schettler [2013] requires that the response variable is bounded. In order to remove this assumption, we used a slightly different approach and instead transform the inequality $\widehat{g}_{n,h}(x) \geq \lambda$ which involves a ratio to the inequality $(nh)^{-1} \sum_{i=1}^{n} (Y_i - \lambda) K((x - X_i)/h) \geq 0$ which does not. The key consequence of this is the assumption (F) where we need to assume that the density f is bounded below, which necessitates that the covariate X be bounded. We feel, however, that in practice the

assumption of bounded X is a more desirable assumption than that of a bounded response. In addition to (F), we also now require some additional assumptions on the kernel K, as well as the smoothness assumptions (S) on the fourth and second moments of Y. The bandwidth requirements (B) also change slightly. Note that the (S) moment assumption is not difficult to satisfy since C_X is compact. We choose to leave the current formulation in order to emphasize where the need for (F) arises.

Theorem 1. *Suppose that assumptions (B), (F), (G), (K), and (S) hold. Then*

$$\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] = \sum_{j=1}^{m} q(x_j)B_{1,j}(x_j)\left\{2\,\frac{\phi(\sqrt{nh^5}B_{2,j})}{\sqrt{nh}} + h^2 B_{2,j}\left[2\Phi(\sqrt{nh^5}B_{2,j}) - 1\right]\right\} + o((nh)^{-1/2} + h^2)$$

where

$$B_{1,j} = \left\{\frac{(\sigma_Y^2 f v_0)^{1/2}}{g'f}\right\}(x_j), \qquad B_{2,j} = -\sigma_K^2\left\{\frac{g''f/2 + g'f'}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j),$$

where $\sigma_K^2 = \int s^2 K(s)ds$ *and* $\sigma_Y^2(x) = \mathrm{Var}(Y|X = x)$.

Proof of Theorem 1. Recall that, $\widehat{g}_{n,h}(x)$ denotes the Nadaraya-Watson kernel estimator

$$\widehat{g}_{n,h}(x) = \frac{\sum_{i=1}^n K\left(\frac{x-X_i}{h}\right)Y_i}{\sum_{j=1}^n K\left(\frac{x-X_j}{h}\right)}.$$

The key difficulty in working with this estimator, is that it is defined as a ratio. We therefore switch out of this formulation. Let $R_{n,i}(x) = h^{-1}K((x - X_i)/h)Y_i$ and $S_{n,i}(x) = h^{-1}K((x - X_i)/h)$ with $R_n = R_n(x) = n^{-1}\sum_{i=1}^n R_{n,i}(x)$ and $S_n = S_n(x) = n^{-1}\sum_{i=1}^n S_{n,i}(x)$. Note that with this notation we have $\widehat{g}_{n,h}(x) = R_n(x)/S_n(x)$. We also let $D_{n,i}(x) = (Y_i - \lambda)S_{n,i}(x)$ with $D_n(x) = n^{-1}\sum_{i=1}^n D_{n,i}(x)$.

First, we compute

$$\begin{aligned}
\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right) &= \int_{C_X} q(x)|\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx \\
&= \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)|\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx \\
&\quad + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)|\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}} - \mathbb{1}_{\{g(x)\geq\lambda\}}|dx \\
&= \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)|\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}} - 0|dx + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)|\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}} - 1|dx.
\end{aligned}$$

It thus follows that

$$\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right) = \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)\mathbb{1}_{\{\widehat{g}_{n,h}(x)\geq\lambda\}}dx + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)\mathbb{1}_{\{\widehat{g}_{n,h}(x)<\lambda\}}dx,$$

so that the risk function for our estimator is hence

$$\begin{aligned}
\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] &= \sum_{j=0}^{r}\int_{x_{2j}}^{x_{2j+1}} q(x)P\big(\widehat{g}_{n,h}(x)\geq\lambda\big)dx \\
&\quad + \sum_{j=1}^{r}\int_{x_{2j-1}}^{x_{2j}} q(x)P\big(\widehat{g}_{n,h}(x)<\lambda\big)dx \equiv T_1 + T_2
\end{aligned}$$

where the expectation is taken with respect to the data (X, Y) in the kernel estimator. We separate the details for each of the two terms defined above, and begin with T_1.

For any $\delta > 0$, we write

$$T_1 = T_{11}(\delta) + T_{12}(\delta) + T_{13}(\delta)$$

where

$$\begin{aligned}
T_{11}(\delta) &= \sum_{j=1}^{r} \int_{x_{2j}}^{x_{2j}+\delta} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx \\
T_{12}(\delta) &= \int_{x_0}^{x_1-\delta} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx + \sum_{j=1}^{r-1} \int_{x_{2j}+\delta}^{x_{2j+1}-\delta} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx \\
&\quad + \int_{x_{2r}+\delta}^{x_{2r+1}} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx \\
T_{13}(\delta) &= \sum_{j=0}^{r-1} \int_{x_{2j+1}-\delta}^{x_{2j+1}} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx.
\end{aligned}$$

Similarly, we define

$$T_2 = T_{21}(\delta) + T_{22}(\delta) + T_{23}(\delta)$$

where

$$\begin{aligned}
T_{21}(\delta) &= \sum_{j=1}^{r} \int_{x_{2j}-\delta}^{x_{2j}} q(x) P\big(\widehat{g}_{n,h}(x) < \lambda\big) dx \\
T_{22}(\delta) &= \sum_{j=1}^{r} \int_{x_{2j-1}+\delta}^{x_{2j}-\delta} q(x) P\big(\widehat{g}_{n,h}(x) < \lambda\big) dx \\
T_{23}(\delta) &= \sum_{j=0}^{r-1} \int_{x_{2j+1}}^{x_{2j+1}+\delta} q(x) P\big(\widehat{g}_{n,h}(x) < \lambda\big) dx.
\end{aligned}$$

The proof now proceeds via the following steps.

1. Show $T_{12}(\delta) = o(1/\sqrt{nh})$ for an appropriate choice of δ and sufficiently large n.

2. Replace δ in step one with δ_n and write $T_1 = T_{11}(\delta_n) + T_{12}(\delta_n) + T_{13}(\delta_n)$ instead. Extending step one, we show that

$$T_{12}(\delta_n) \leq \frac{C}{(nh)^2 \delta_n^4},$$

for some constant C, where $\delta_n = h/\log n$ so that $\delta_n = o(h)$. We assume that $(nh)^3 \delta_n^8 = n^3 h^{11} \log^{-8} n \to \infty$ which implies that $T_{12}(\delta_n) = o((nh)^{-1/2})$.

3. Similarly, write $T_2 = T_{21}(\delta_n) + T_{22}(\delta_n) + T_{23}(\delta_n)$ and repeat steps one and two above to show that $T_{22}(\delta_n) = o(1/\sqrt{nh})$.

4. Show that

$$\begin{aligned} & T_{11}(\delta_n) + T_{21}(\delta_n) \\ &= \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} \left\{ q(x_{2j}) \int_{-(nh)^{1/2}\delta_n}^{(nh)^{1/2}\delta_n} |P(D_n(x_{2j}^t) < 0) - 1_{\{t<0\}}| dt \right\} + O(\delta_n^2) \\ &\equiv \widetilde{T}_1(\delta_n) + O(\delta_n^2), \end{aligned}$$

$$\begin{aligned} & T_{13}(\delta_n) + T_{23}(\delta_n) \\ &= \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} \left\{ q(x_{2j-1}) \int_{-(nh)^{1/2}\delta_n}^{(nh)^{1/2}\delta_n} |P(D_n(x_{2j-1}^t) < 0) - 1_{\{t\geq 0\}}| dt \right\} + O(\delta_n^2) \\ &\equiv \widetilde{T}_3(\delta_n) + O(\delta_n^2) \end{aligned}$$

where $x_j^t = x_j + (nh)^{-1/2}t$. Since we assume that $\delta_n = o(h)$, we have that the error $O(\delta_n^2) = o(h^2)$.

5. We next reduce the region of integration in $\widetilde{T}_1$ from $[-(nh)^{1/2}\delta_n, (nh)^{1/2}\delta_n]$. Let $t_n \to \infty$ more slowly than $(nh)^{1/2}\delta_n$ and write $B_{n,j} = [t_j^* - t_n, t_j^* + t_n]$. The value of t_j^* will be defined within the detailed proof. In this step, we define

$$\widetilde{T}_{11}(t_n) = \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} \left\{ q(x_{2j}) \int_{B_{n,j}} |P(\widehat{g}_{n,h}(x_{2j}^t) < \lambda) - 1_{\{t<0\}}| dt \right\}$$

and show that $\widetilde{T}_1(\delta_n) - \widetilde{T}_{11}(t_n) = o((nh)^{-1/2})$. Repeating the argument shows that

$$\widetilde{T}_{31}(t_n) = \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} \left\{ q(x_{2j-1}) \int_{B_{n,j}} |P(\widehat{g}_{n,h}(x_{2j-1}^t) < \lambda) - 1_{\{t\geq 0\}}| dt \right\}$$

with $\widetilde{T}_3(\delta_n) - \widetilde{T}_{31}(t_n) = o((nh)^{-1/2})$.

6. Define the terms

$$A(x_j) = -\left\{\frac{g'f}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j), \qquad B(x_j) = -\sqrt{nh^5}\left\{\frac{g''f/2+g'f'}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j)\sigma_K^2,$$

where $v_{2,0} = \int K^2(s)ds$ and $\sigma_K^2 = \int s^2K^2(s)ds$. In this penultimate step we show that

$$\begin{aligned}
&\widetilde{T}_{11}(t_n) + \widetilde{T}_{31}(t_n) \\
&= \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j})t + B(x_{2j})\right) - 1_{\{t\geq 0\}}|dt\right\} \\
&\quad + \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j-1})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j-1})t + B(x_{2j-1})\right) - 1_{\{t<0\}}|dt\right\} \\
&\quad + o((nh)^{-1/2}) + O\left(t_n h^3 + \frac{t_n^2h^2}{\sqrt{nh}} + \frac{t_n^3}{nh}\right),
\end{aligned}$$

where Φ denotes the standard normal cumulative distribution function. The main tool here is the Barry-Esseen theorem, although the extra smoothness assumptions on the functions g, f, g_2 are also used in this step to obtain the required bounds. We therefore need $t_n = o((nh)^{1/6})$ and $t_n = o(h^{-1})$. We choose $t_n = (nh)^{1/6}\log^{-1} n$. Since we assumed that $(nh)^3 \geq n^3h^{11}\log^{-8} n \to \infty$ we also have that $ht_n = (nh^7)^{1/6}\log^{-1} n \to 0$ as long as $nh^5 \leq O(1)$ as required. Then

$$\begin{aligned}
\widetilde{T}_{11}(t_n) + \widetilde{T}_{31}(t_n) &= \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j})t + B(x_{2j})\right) - 1_{\{t\geq 0\}}|dt\right\} \\
&\quad + \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j-1})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j-1})t + B(x_{2j-1})\right) - 1_{\{t<0\}}|dt\right\} \\
&\quad + o((nh)^{-1/2}).
\end{aligned}$$

7. We are now in a position to derive the final expression. Applying Lemma B.3 of Doss

and Weng [2018], we have

$$\begin{aligned}
\widetilde{T} &\equiv \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j})t+B(x_{2j})\right)-1_{\{t\geq 0\}}|dt\right\} \\
&\quad +\frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j-1})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j-1})t+B(x_{2j-1})\right)-1_{\{t<0\}}|dt\right\} \\
&= \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j})\int_{-\infty}^{\infty}|\Phi\left(-A(x_{2j})t-B(x_{2j})\right)-1_{\{t<0\}}|dt\right\} \\
&\quad +\frac{1}{(nh)^{1/2}}\sum_{j=1}^{r}\left\{q(x_{2j-1})\int_{-\infty}^{\infty}|\Phi\left(A(x_{2j-1})t+B(x_{2j-1})\right)-1_{\{t<0\}}|dt\right\} \\
&= \frac{1}{(nh)^{1/2}}\sum_{j=1}^{2r}q(x_j)\frac{2\phi(B(x_j))+2B(x_j)\Phi(B(x_j))-B(x_j)}{|A(x_j)|} \\
&= \sum_{j=1}^{2r}q(x_j)B_{1,j}\left\{2\,\frac{\phi(\sqrt{nh^5}B_{2,j})}{\sqrt{nh}}+h^2B_{2,j}\left[2\Phi(\sqrt{nh^5}B_{2,j})-1\right]\right\},
\end{aligned}$$

letting $B_{1,j}=1/|A(x_j)|$ and $B_{2,j}=B(x_j)/\sqrt{nh^5}$. Collecting the error terms through steps two to six, we have thus established that

$$\mathrm{E}\left[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)\right] = \widetilde{T}+o\left(\frac{1}{\sqrt{nh}}+h^2\right).$$

To complete the proof it therefore remains to fill in the details of steps 1-6.

STEP 1. By the local smoothness assumptions in (S) g is C^2 in the neighbourhood of each x_j with $g(x_j)=\lambda$ and hence, by the inverse function theorem, $g(x)$ is invertible in a neighbourhood of each x_j for $j=1,\ldots,2r$. Therefore, for all sufficiently small $\varepsilon>0$, we have that

$$g^{-1}(\lambda-\varepsilon) = x_j-\varepsilon\frac{1}{g'(x_j)}+O_j(\varepsilon^2).$$

Define $\delta_{\varepsilon,j}=\varepsilon/|g'(x_j)|+O_j(\varepsilon^2)$. It follows that we can write

$$\{x:g(x)\leq\lambda-\varepsilon\} = \bigcup_{j=0}^{r}[x_{2j}+\delta_{\varepsilon,2j},x_{2j+1}-\delta_{\varepsilon,2j+1}]. \tag{A.1}$$

It follows that for any $\delta \geq \max_{j\in\{0,\ldots,2r\}} \delta_{\varepsilon,j} > 0$, we have

$$\bigcup_{j=0}^{r} [x_{2j} + \delta, x_{2j+1} - \delta] \subset \{x : g(x) \leq \lambda - \varepsilon\}.$$

Next, consider an x such that $g(x) \leq \lambda - \varepsilon$, and note that this implies that $(g(x) - \lambda)f(x) \leq -\varepsilon f(x) \leq -\widetilde{\varepsilon} < 0$, where $\widetilde{\varepsilon} = b_f \varepsilon$ and $b_f > 0$ is such that $\inf_{x\in C_X} f(x) \geq b_f$ from assumption (F). Let $\mu(x) = (g(x) - \lambda)f(x)$. Then

$$\begin{aligned}
P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) &= P\big(R_n(x) - \lambda S_n(x) \geq 0\big) \\
&= P\big(R_n(x) - \lambda S_n(x) + \widetilde{\varepsilon} \geq \widetilde{\varepsilon}\big) \quad = \quad P\big(D_n(x) + \widetilde{\varepsilon} \geq \widetilde{\varepsilon}\big) \\
&\leq P\big(D_n(x) - \mu(x) \geq \widetilde{\varepsilon}\big) \\
&\leq P\big(|D_n(x) - E[D_n(x)]| \geq \widetilde{\varepsilon}/2\big) + P\big(\sup_{x\in C_X} |E[D_n(x)] - \mu(x)| \geq \widetilde{\varepsilon}/2\big)
\end{aligned}$$

noting that the probability on the right hand side is either zero or one. We then show that this second term above is always zero, as long as n is chosen sufficiently large.

$$\begin{aligned}
&E[D_n(x)] - \mu(x) \\
&= \int (g(t) - \lambda)\frac{1}{h}K\left(\frac{x-t}{h}\right) f(t)dt - \mu(x) \\
&= \int (g(x - sh) - \lambda)f(x - sh)K(s)ds - \mu(x) \\
&= \int \Big[\{g(x - sh) - g(x)\}f(x) + g(x)\{f(x - sh) - f(x)\}\Big]K(s)ds \to 0
\end{aligned}$$

as $n \to \infty$ since g, f are continuous on C_X and hence bounded and uniformly continuous and we apply the dominated convergence theorem. We have thus shown that for all n large enough

$$P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) \leq P\big(|D_n(x) - E[D_n(x)]| \geq \widetilde{\varepsilon}/2\big).$$

We now return to the term T_{12}. For convenience, we write

$$T_{12}(\delta) = \sum_{j=0}^{r} \int_{x_{2j}+\delta}^{x_{2j+1}-\delta} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx$$

using the simplifying convention that $x_0 + \delta = x_0$ and $x_{2r+1} - \delta = x_{2r+1}$. Consider now $\delta \geq \max_{j\in\{0,\ldots,2r\}} \delta_{\varepsilon,j}$ as defined above. Then, for any $j = 0, \ldots, r$ and $x \in [x_{2j}+\delta, x_{2j+1}-\delta]$, $g(x) \leq \lambda - \varepsilon$. We thus have that for sufficiently large n

$$\begin{aligned} T_{12}(\delta) &\leq \sum_{j=0}^{r} \int_{x_{2j}+\delta}^{x_{2j+1}-\delta} q(x) P(|D_n(x) - E[D_n(x)]| \geq \widetilde{\varepsilon}/2)\, dx \\ &\leq \sum_{j=0}^{r} \int_{x_{2j}+\delta}^{x_{2j+1}-\delta} q(x) \frac{\mathrm{E}\left[(D_n(x) - E[D_n(x)])^4\right]}{\widetilde{\varepsilon}^4}\, dx \\ &\leq \frac{1}{(nh)^2\widetilde{\varepsilon}^4} \int_{C_X} \big(M_4(x) + M_2^2(x)\big)\, q(x) dx \end{aligned}$$

by Markov's inequality and Lemma 4, for sufficiently large n.

STEP 2. We next show that the result of step one continues to hold for any appropriate sequence $\delta_n \to 0$. For such a δ_n, let ε_n in (A.1) be such that $\varepsilon_n = \min_j(|g'(x_j)|)\delta_n/2 > 0$. For all sufficiently large n and for any $j = 0, \ldots, r$ we then have that $x \in [x_{2j} + \delta_n, x_{2j+1} - \delta_n]$, $g(x) \leq \lambda - \varepsilon_n$. To see this, note that if $g'(x_j) > 0$, then

$$\begin{aligned} g^{-1}(\lambda - \varepsilon_n) &= x_j - \frac{\varepsilon_n}{g'(x_j)} + C_j^* \varepsilon_n^2 \\ &= x_j - \frac{\min_j(|g'(x_j)|)\delta_n}{2g'(x_j)} + C_j^* \delta_n^2 \\ &\geq x_j - \delta_n(1/2 + C_j^* \delta_n) \quad \geq \quad x_j - \delta_n. \end{aligned}$$

Similarly, if $g'(x_j) < 0$, then

$$\begin{aligned} g^{-1}(\lambda - \varepsilon_n) &= x_j - \frac{\varepsilon_n}{g'(x_j)} + C_j^* \varepsilon_n^2 \\ &= x_j + \frac{\min_j(|g'(x_j)|)\delta_n}{2|g'(x_j)|} + C_j^* \delta_n^2 \\ &\leq x_j + \delta_n(1/2 + C_j^* \delta_n) \quad \leq \quad x_j + \delta_n. \end{aligned}$$

In the above, the constants are uniformly bounded, though may change from line to line.

We may thus repeat the argument in step one to show that

$$T_{12}(\delta_n) \quad \leq \quad \frac{C}{(nh)^2 \delta_n^4} \int_{C_X} \left(M_4(x) + M_2^2(x) \right) q(x) dx,$$

for some finite constant C. It follows that $T_{12}(\delta_n) = o(1/\sqrt{nh})$ as long as $(nh)^3 \delta_n^8 \to \infty$.

STEP 3. We now repeat steps one and two to obtain a bound for term $T_{22}(\delta_n)$. Again, by assumption (S) for all sufficiently small $\varepsilon > 0$, we have that

$$g^{-1}(\lambda + \varepsilon) \quad = \quad x_j + \varepsilon \frac{1}{g'(x_j)} + O_j(\varepsilon^2).$$

Define $\delta_{\varepsilon,j} = \varepsilon / |g'(x_j)| + O_j(\varepsilon^2)$. It follows that we can write

$$\{x : g(x) \geq \lambda + \varepsilon\} \quad = \quad \bigcup_{j=1}^{r} [x_{2j-1} + \delta_{\varepsilon,2j}, x_{2j} - \delta_{\varepsilon,2j+1}].$$

As in step two, let ε_n be such that $\varepsilon_n = \min_j(|g'(x_j)|)\delta_n/2 > 0$. For all sufficiently large n and for any $j = 0, \ldots, r$ we then have that $x \in [x_{2j-1} + \delta_n, x_{2j} - \delta_n]$ implies $g(x) \geq \lambda + \varepsilon_n$. To see this, note that if $g'(x_j) > 0$, then

$$\begin{aligned} g^{-1}(\lambda + \varepsilon_n) &= x_j + \frac{\varepsilon_n}{g'(x_j)} + C_j^* \varepsilon_n^2 \\ &= x_j + \frac{\min_j(|g'(x_j)|)\delta_n}{2g'(x_j)} + C_j^* \delta_n^2 \\ &\leq x_j + \delta_n(1/2 + C_j \delta_n) \quad \leq \quad x_j + \delta_n. \end{aligned}$$

Similarly, if $g'(x_j) < 0$, then

$$\begin{aligned} g^{-1}(\lambda+\varepsilon_n) &= x_j + \frac{\varepsilon_n}{g'(x_j)} + C_j^*\varepsilon_n^2 \\ &= x_j - \frac{\min_j(|g'(x_j)|)\delta_n}{2|g'(x_j)|} + C_j^*\delta_n^2 \\ &\geq x_j - \delta_n(1/2 + C_j^*\delta_n) \quad \geq \quad x_j - \delta_n. \end{aligned}$$

In the above, the constants C_j^* are uniformly bounded, though may change from line to line. Therefore, arguing as before, for sufficiently large n, we obtain the same bound as in step two.

STEP 4.

$$\begin{aligned} T_{11}(\delta_n) + T_{21}(\delta_n) &= \sum_{j=1}^{r} \int_{x_{2j}}^{x_{2j}+\delta_n} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx + \sum_{j=1}^{r} \int_{x_{2j}-\delta_n}^{x_{2j}} q(x) P\big(\widehat{g}_{n,h}(x) < \lambda\big) dx \\ &= \sum_{j=1}^{r} \int_{x_{2j}}^{x_{2j}+\delta_n} q(x) \left\{1 - P\big(\widehat{g}_{n,h}(x) < \lambda\big)\right\} dx + \sum_{j=1}^{r} \int_{x_{2j}-\delta_n}^{x_{2j}} q(x) P\big(\widehat{g}_{n,h}(x) < \lambda\big) dx \\ &= \sum_{j=1}^{r} \int_{x_{2j}-\delta_n}^{x_{2j}+\delta_n} q(x) \left| P\big(\widehat{g}_{n,h}(x) < \lambda\big) - 1(x \geq x_{2j})\right| dx \\ &= \sum_{j=1}^{r} q(x_{2j}) \int_{x_{2j}-\delta_n}^{x_{2j}+\delta_n} \left| P\big(D_n(x) < 0\big) - 1(x \geq x_{2j})\right| dx + O(\delta_n^2), \end{aligned}$$

since

$$\sum_{j=1}^{r} \int_{x_{2j}-\delta_n}^{x_{2j}+\delta_n} \{q(x) - q(x_{2j})\} \left| P\big(D_n(x) < 0\big) - 1(x \geq x_{2j})\right| dx \quad = \quad O(\delta_n^2),$$

using assumption (S). Next, let $t = (nh)^{1/2}(x - x_{2j})$ to obtain

$$\begin{aligned} &\sum_{j=1}^{r} q(x_{2j}) \int_{x_{2j}-\delta_n}^{x_{2j}+\delta_n} \left| P\big(D_n(x) < 0\big) - 1(x \geq x_{2j})\right| dx \\ &= \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} q(x_{2j}) \int_{-(nh)^{1/2}\delta_n}^{(nh)^{1/2}\delta_n} \left| P\big(D_n(x_{2j}^t) < 0\big) - 1(t \geq 0)\right| dt \quad \equiv \quad \widetilde{T}_1(\delta_n), \end{aligned}$$

where $x_j^t = x_j + (nh)^{-1/2}t$. A similar approach shows that

$$\begin{aligned}
&T_{13}(\delta_n) + T_{23}(\delta_n) \\
&= \sum_{j=1}^{r} \int_{x_{2j-1}-\delta_n}^{x_{2j-1}} q(x) P\big(\widehat{g}_{n,h}(x) \geq \lambda\big) dx + \sum_{j=1}^{r} \int_{x_{2j-1}}^{x_{2j-1}+\delta_n} q(x) P\big(\widehat{g}_{n,h}(x) < \lambda\big) dx \\
&= \sum_{j=1}^{r} q(x_{2j-1}) \int_{x_{2j-1}-\delta_n}^{x_{2j-1}+\delta_n} \big|P\big(D_n(x) < 0\big) - 1(x < x_{2j-1})\big| \, dx + O(\delta_n^2), \\
&= \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} \left\{ q(x_{2j-1}) \int_{-(nh)^{1/2}\delta_n}^{(nh)^{1/2}\delta_n} |P(D_n(x_{2j-1}^t) < 0) - 1_{\{t<0\}}| dt \right\} + O(\delta_n^2).
\end{aligned}$$

STEP 5. Each of the r integrals comprising $\widetilde{T}_1(\delta_n)$ has region of integration given by $A_n = [-(nh)^{1/2}\delta_n, (nh)^{1/2}\delta_n]$. Our next step is to reduce this to $B_{j,n} = [t_j^* - t_n, t_j^* + t_n]$ for an appropriate choice of $t_j^*, t_n \to \infty$. To this end, define

$$\widetilde{T}_{11}(t_n) = \frac{1}{(nh)^{1/2}} \sum_{j=1}^{r} q(x_{2j}) \int_{B_{j,n}} \big|P\big(D_n(x_{2j}^t) < 0\big) - 1_{\{t \geq 0\}}\big| \, dt.$$

In Step 5, we show that $\widetilde{T}_1(\delta_n) = \widetilde{T}_{11}(t_n) + o(1/\sqrt{nh})$. Let $\mu_D(x) = E[D_n(x)]$. From Lemma 5 we know that

$$\mu_D(x_j^t) = a(x_j)(nh)^{-1/2}t + b(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right] + O((nh)^{-3/2}t^3 + h^3).$$

Since $\{g''f/2 + g'f'\}(x_j)$ is bounded, $f \geq 0$ and g is locally monotone near each x_j, it follows that for sufficiently large n the function $\mu_D(x_j^t)$ is monotone as a function of t. Let t_j^* denote the unique value such that $\mu_D(x_j^t) = 0$ for $t \in A_n$ and $t_n \to \infty$ more slowly than $\sqrt{nh}\delta_n$. Recall also that Lemma 5 tells us that $t_j^* = O(\sqrt{nh^5} + \sqrt{nh}\delta_n^2)$. Let $\eta = (nh)^{-1/2}t - sh$ and note that we can write $\mu_D(x_j^t) = \int (g(x_j+\eta) - \lambda) f(x_j+\eta) K(s) ds$. Let $m(x) = (g(x) - \lambda)f(x)$, and let $\eta^* = (nh)^{-1/2}t_j^* - sh$. Then we have

$$\begin{aligned}
\mu_D(x_j^t) - \mu_D(x_j^{t_j^*}) &= \int \Big[m(x_j + \eta) - m(x_j + \eta^*)\Big] K(s) ds \\
&= \int m'(x_j^*)(\eta - \eta^*) K(s) ds = (nh)^{-1/2}(t - t_j^*) \int m'(x_j^*) K(s) ds,
\end{aligned}$$

where x_j^* is a value between $x_j + \eta$ and $x_j + \eta^*$. Now, $m'(x) = \{g'f + (g - \lambda)f'\}(x)$ with $m'(x_j) = \{g'f\}(x_j)$ and is C^1 by our assumptions on g and f. Hence $|m'(x_j^*) - m'(x_j)| = O(|\eta^* - \eta|) = O((nh)^{-1/2}|t_j^* - t|)$. It follows that for sufficiently large n we have

$$|\mu_D(x_j^t)| \quad \geq \quad c_\mu (nh)^{-1/2}|t - t_j^*|,$$

for some strictly positive constant c_μ.

Define $I_{j,n} = A_n \setminus B_{j,n}$. Note that

$$|P(D_n(x_{2j}^t) < 0) - 1_{\{t \geq 0\}}| = \begin{cases} P(D_n(x_{2j}^t) \geq 0), & \text{if } t \geq 0 \\ P(D_n(x_{2j}^t) < 0), & \text{if } t < 0 \end{cases}$$

For $j = 1, \ldots, r$ and as long as n is large enough we have that $\mu_D(x_{2j}^t) > 0$ for $t < t_{2j}^*$ and $t \in I_{j,n}$ and $\mu_D(x_{2j}^t) < 0$ for $t \geq t_{2j}^*$ and $t \in I_{2j,n}$. It follows that when $t \geq 0$ and $t \in I_{2j,n}$,

$$\begin{aligned} P(D_n(x_{2j}^t) \geq 0) &= P\left(D_n(x_{2j}^t) - \mu_D(x_{2j}^t) \geq -\mu_D(x_{2j}^t)\right) \\ &\leq P\left(|D_n(x_{2j}^t) - \mu_D(x_{2j}^t)| \geq |\mu_D(x_{2j}^t)|\right) \\ &\leq \frac{\mathrm{Var}(D_n(x_{2j}^t))}{\mu_D(x_{2j}^t)^2} \\ &\leq \frac{\mathrm{Var}(D_n(x_{2j}^t))}{c_\mu^2 (nh)^{-1}(t - t_{2j}^*)^2} \quad = \quad \frac{O(1)}{(t - t_{2j}^*)^2}, \end{aligned}$$

since by Lemma 5 we know that $\mathrm{Var}(D_n(x_{2j}^t)) = \sigma_D^2(x_{2j}^t) = (nh)^{-1/2}O(1)$ on $I_{2j,n}$. Similarly, when $t < 0$ and $t \in I_{2j,n}$,

$$\begin{aligned} P(D_n(x_{2j}^t) < 0) &\leq P\left(|D_n(x_{2j}^t) - \mu_D(x_{2j}^t)| \geq |\mu_D(x_{2j}^t)|\right) \\ &\leq \frac{\mathrm{Var}(D_n(x_{2j}^t))}{c_\mu^2 (nh)^{-1}(t - t_{2j}^*)^2} \quad = \quad \frac{O(1)}{(t - t_{2j}^*)^2}. \end{aligned}$$

It follows that

$$\begin{aligned}(nh)^{1/2}\left\{\widetilde{T}_1(\delta_n)-\widetilde{T}_{11}(t_n)\right\} &= \sum_{j=1}^{r} q(x_{2j})\int_{I_{j,n}}\left|P\big(D_n(x_{2j}^t)<0\big)-1_{\{t\geq 0\}}\right|dt \\ &\leq \sum_{j=1}^{r} q(x_{2j})\int_{I_{2j,n}}\frac{O(1)}{(t-t_{2j}^*)^2}dt\to 0,\end{aligned}$$

by the dominated convergence theorem since $|t-t_{2j}^*|\geq t_n\to\infty$ on $I_{2j,n}$ and $(t-t_{2j}^*)^{-2}$ is integrable. It follows that $\widetilde{T}_1(\delta_n)=\widetilde{T}_{11}(t_n)+o(1/\sqrt{nh})$.

A similar approach establishes that $\widetilde{T}_3(\delta_n)=\widetilde{T}_{31}(t_n)+o(1/\sqrt{nh})$, where

$$\widetilde{T}_{31}(t_n) = \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r} q(x_{2j-1})\int_{B_{2j-1,n}}\left|P\big(D_n(x_{2j-1}^t)<0\big)-1_{\{t<0\}}\right|dt.$$

STEP 6. Recall that $x_j^t=x_j+(nh)^{-1/2}t$, and let Φ denote the cumulative distribution function of the standard normal distribution. Define

$$\begin{aligned}\widetilde{T}_{111}(t_n) &= \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r} q(x_{2j})\int_{B_{n,j}}\left|\Phi\left(A(x_{2j})t+B(x_{2j})\right)-1(t\geq 0)\right|dt \\ &= \frac{1}{(nh)^{1/2}}\sum_{j=1}^{r} q(x_{2j})\int_{-\infty}^{\infty}\left|\Phi\left(A(x_{2j})t+B(x_{2j})\right)-1(t\geq 0)\right|dt+o(1/\sqrt{nh}),\end{aligned}$$

since $t_n\to\infty$. Now,

$$\begin{aligned}&(nh)^{1/2}|\widetilde{T}_{11}(t_n)-\widetilde{T}_{111}(t_n)| \\ &\leq \sum_{j=1}^{r} q(x_{2j})\int_{B_{j,n}}\left|P\left(\left\{\frac{D_n-\mu_D}{\sigma_D}\right\}(x_{2j}^t)<-\left\{\frac{\mu_D}{\sigma_D}\right\}(x_{2j}^t)\right)-\Phi\left(-\left\{\frac{\mu_D}{\sigma_D}\right\}(x_{2j}^t)\right)\right|dt \\ &\qquad+\sum_{j=1}^{r} q(x_{2j})\int_{B_{j,n}}\left|\Phi\left(-\left\{\frac{\mu_D}{\sigma_D}\right\}(x_{2j}^t)\right)-\Phi\left(A(x_{2j})t+B(x_{2j})\right)\right|dt \\ &\leq \sum_{j=1}^{r} q(x_{2j})\int_{B_{j,n}}\left\{\frac{\rho_1(x_{2j}^t)}{\sigma_1^3(x_{2j}^t)\sqrt{n}}+\phi(0)\left|-\left\{\frac{\mu_D}{\sigma_D}\right\}(x_{2j}^t)-A(x_{2j})t-B(x_{2j})\right|\right\}dt\end{aligned}$$

by the Berry-Esseen theorem for the first term and using the fact that the standard normal

density is bounded above by $\phi(0)$. In the notation above we have

$$\rho_1(x) \;=\; E[|D_{n,1}(x) - \mu_D(x)|^3], \qquad \sigma_1^2(x) \;=\; \mathrm{Var}(D_{n,1}(x)),$$

with $hD_{n,1}(x) = (Y_1 - \lambda)K((x - X_1)/h) = \{R_{n,1} - \lambda S_{n,1}\}(x)$. Applying Lemma 5 we have that $\{\rho_1/\sigma_1^3\}(x_j^t) = h^{-1/2}O(1)$ and hence

$$\sum_{j=1}^{r} q(x_{2j}) \int_{B_{j,n}} \frac{\rho_1(x_{2j}^t)}{\sigma_1^3(x_{2j}^t)\sqrt{n}} dt \;\leq\; \sum_{j=1}^{r} q(x_{2j}) |B_{j,n}| O(1) \frac{1}{\sqrt{nh}} \;\leq\; C \frac{t_n}{\sqrt{nh}} \to 0.$$

For the second term, we again apply Lemma 5,

$$\begin{aligned} \int_{B_{j,n}} \left| - \left\{ \frac{\mu_D}{\sigma_D} \right\} (x_{2j}^t) - A(x_{2j-1})t - B(x_{2j-1}) \right| dt &= \int_{B_{j,n}} O(th^2 + (nh)^{-1/2}t^2 + \sqrt{nh^7}) dt \\ &= t_n \left(\sqrt{nh^7} \right) + \frac{t_n^3}{\sqrt{nh}} + t_n^2 h^2. \end{aligned}$$

Therefore,

$$|\widetilde{T}_{11}(t_n) - \widetilde{T}_{111}(t_n)| \;=\; o((nh)^{-1/2}) + O\left(t_n h^3 + \frac{t_n^2 h^2}{\sqrt{nh}} + \frac{t_n^3}{nh} \right),$$

since $|t_n| \leq \sqrt{nh}\delta_n$ and $\delta_n = o(1)$.

□

Lemma 4. *Let $D_{n,i}(x) = h^{-1}(Y_i - \lambda)K((x - X_i)/h)$ and $\overline{D}_{n,i}(x) = D_{n,i}(x) - E[D_{n,i}(x)]$. Define the function $m_k(x) = E[(Y - \lambda)^k | X = x]f(x)$ and assume that there exists an $M(x)$ such that for $k = 2, 4$*

$$\lim_{h \to 0} \sup_{s \in (x - C_X)/h} m_k(x - hs) \;\leq\; M_k(x)$$

with $\int_{C_X} M_2^2(x)dx < \infty$ *and* $\int_{C_X} M_4(x)dx < \infty$. *Lastly, assume that* $v_{k,0}$ *are finite for* $k = 2, 4$. *Then*

$$E\left[\left\{\frac{1}{n}\sum_{i=1}^{n}\overline{D}_{n,i}(x)\right\}^4\right] \leq \frac{1}{(nh)^2}\left(M_4(x) + M_2^2(x)\right),$$

for large enough n *as long as* $h = o(1)$ *with* $nh \to \infty$.

Proof. We begin with a calculation.

$$E\left[\left\{\frac{1}{n}\sum_{i=1}^{n}\overline{D}_{n,i}(x)\right\}^4\right] = \frac{1}{n^4}\sum_{i_1=1}^{n}\sum_{i_2=1}^{n}\sum_{i_3=1}^{n}\sum_{i_4=1}^{n}E\left[\overline{D}_{n,i_1}(x)\overline{D}_{n,i_2}(x)\overline{D}_{n,i_3}(x)\overline{D}_{n,i_4}(x)\right].$$

Now, since the terms are independent for different subscripts and since each $\overline{D}_{n,i_1}(x)$ is mean zero, it follows that

$$E\left[\left\{\frac{1}{n}\sum_{i=1}^{n}\overline{D}_{n,i}(x)\right\}^4\right] = \frac{1}{n^4}\left\{nE[\overline{D}_{n,1}^4(x)] + n(n-1)E[\overline{D}_{n,1}^2(x)]^2\right\}$$

We now calculate each of the terms.

$$\begin{aligned}
E[\overline{D}_{n,1}^2(x)] &\leq E[D_{n,1}^2(x)] \\
&= \int E[(Y-\lambda)^2|X=y]f(y)\frac{1}{h^2}K^2\left(\frac{x-y}{h}\right)dy \\
&= h^{-1}\int E[(Y-\lambda)^2|X=x-sh]f(x-sh)K^2(s)\,ds \\
&= h^{-1}\int m_2(x-sh)K^2(s)ds \leq h^{-1}M_2(x)v_{2,0}.
\end{aligned}$$

Next, we compute

$$\begin{aligned}
E[\overline{D}_{n,1}^4(x)] &\leq 11E[D_{n,1}^4(x)] \\
&= \int E[(Y-\lambda)^4|X=y]f(y)\frac{1}{h^4}K^4\left(\frac{x-y}{h}\right)dy \\
&= h^{-3}\int E[(Y-\lambda)^4|X=x-sh]f(x-sh)K^4(s)\,ds \\
&= h^{-3}\int m_4(x-sh)K^4(s)ds \leq h^{-3}M_4(x)v_{4,0}.
\end{aligned}$$

Both the last inequalities hold only for sufficiently large n. Then

$$\begin{aligned} E\left[\left\{\frac{1}{n}\sum_{i=1}^{n}\overline{D}_{n,i}(x)\right\}^4\right] &= \frac{1}{n^4}\left\{nE[\overline{D}_{n,1}^4(x)] + n(n-1)E[\overline{D}_{n,1}^2(x)]^2\right\} \\ &\leq \frac{1}{(nh)^3}M_4(x) + \frac{1}{(nh)^2}M_2^2(x) \leq \frac{1}{(nh)^2}\left(M_4(x) + M_2^2(x)\right) \end{aligned}$$

as long as $nh \geq 1$. □

Lemma 5. *Define* $\sigma_Y^2(x) = \mathrm{Var}(Y|X=x)$ *with* $g_2(x) = E[Y^2|X=x]$. *Also, let* $v_{2,0} = \int K^2(s)ds, \sigma_K^2 = \int s^2K(s)ds$ *and assume that* $\int |s^3|K(s)ds, \int s^2K^2(s)ds$ *and* $\int s^4K^3(s)ds$ *are finite. Suppose that there exists an* $\eta > 0$ *such that* f, g_2 *are both* $C^2(B_\eta(x_j))$ *and* g *is* $C^3(B_\eta(x_j))$, *where* $B_\eta(x)$ *denotes a ball of size* η *around* x. *Define* $a(x_j) = \{g'f\}(x_j)$, $b(x_j) = \{g''f/2 + g'f'\}(x_j)$, *and* $c(x_j) = v_{2,0}\{\sigma_Y^2 f\}(x_j)$. *Then*

$$\mu_D(x_j^t) = a(x_j)(nh)^{-1/2}t + b(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right] + O((nh)^{-3/2}t^3 + h^3).$$

Assume also that $|t| \leq \sqrt{nh}\delta_n$ *with* $\delta_n = o(h)$ *and* $h = o(1)$ *and* t_j^* *is such that* $\mu_D(x_j^t) = 0$. *Then*

$$t_j^* = O(\sqrt{nh^5} + \sqrt{nh}\delta_n^2).$$

Note also that for all t *such that* $|t| \leq \sqrt{nh}\delta_n$ *and* $\delta_n = o(1)$

$$\begin{aligned} \sigma_D^2(x_j^t) &= (nh)^{-1}\left[\{\sigma_Y^2 f\}(x_j)v_{2,0} + O((nh)^{-1/2}t + h^2 + n^{-1}t^2)\right] \\ \frac{\rho_1(x_j^t)}{\sigma_1^3(x_j^t)} &= h^{-1/2}O(1), \end{aligned}$$

and

$$\left|-\frac{\mu_D(x_j^t)}{\sigma_D(x_j^t)} - t\left\{\frac{g'f}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j) - \left\{\frac{g''f/2 + g'f'}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(x_j)\right| = O(th^2 + (nh)^{-1/2}t^2 + \sqrt{nh^7}).$$

Throughout, all the little-oh and big-oh terms are uniform on $B_\eta(x_j)$.

Proof. Let $\eta = (nh)^{-1/2}t - sh$ and recall that $x_j^t = x_j + (nh)^{-1/2}t$ and define $\sigma_K^2 = \int s^2 K(s)ds$. Then

$$\begin{aligned}
&\mu_D(x_j^t) \\
&= \int \frac{1}{h}(g(y)-\lambda)K\left(\frac{x_j^t - y}{h}\right) f(y)dy \\
&= \int (g(x_j+\eta)-\lambda) f(x_j+\eta)K(s)ds \\
&= \int \left\{ g'(x_j)\eta + \frac{1}{2}g''(x_j^*)\eta^2 \right\} \left\{ f(x_j) + f'(x_j^*)\eta \right\} K(s)ds \\
&= \{g'f\}(x_j)(nh)^{-1/2}t + \left\{ g''f/2 + g'f' \right\}(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2 \right] + E_1 + E_2 + E_3,
\end{aligned}$$

where the three error terms are derived below.

$$\begin{aligned}
2|E_1| &= f(x_j)\left| \int \left[g''(x_j^*) - g''(x_j) \right] \eta^2 K(s)ds \right| \\
&\leq f(x_j) \int \left| g'''(x_j^*)\eta^3 \right| K(s)ds \\
&= O(1)[(nh)^{-3/2}t^3 + h^3],
\end{aligned}$$

by our assumptions on f and g. Note that the convergence is uniform. Similarly,

$$\begin{aligned}
2|E_2| &= g'(x_j)\left| \int \left[f'(x_j^*) - f'(x_j) \right] \eta^2 K(s)ds \right| \\
&\leq g'(x_j) \int |f''(x_j^*)\eta^3| K(s)ds \\
&= O(1)[(nh)^{-3/2}t^3 + h^3].
\end{aligned}$$

Note that the first line of the calculations to bound both E_1 and E_2 above we can achieve rates of $o(1)[(nh)^{-1}t^2 + h^2]$ assuming only C_1 for f and C_2 for g and applying the dominated convergence theorem. However, it turns out that we require tighter bounds, and hence we move up to C^2 and C^3 respectively. The exact problem arises since a bound of $h^2(1+o(1)) = h^2 + o(h^2)$ is not sufficiently tight on the $o(h^2)$ in step six of the main proof. Adding the extra derivative yields instead $h^3(1+o(1))$, and the move from $o(h^2)$ to the more precise $O(h^3)$ is

sufficient for our purposes. The comment persists for our calculations of the variance since ultimately the bound required is on μ_D/σ_D. Lastly,

$$\begin{aligned} 2|E_3| &= \left| \int g''(x_j^*) f'(x_j^*) \eta^3 K(s) ds \right| \\ &\leq 3C[(nh)^{-3/2} t^3 + h^3]. \end{aligned}$$

Let $a(x_j) = \{g'f\}(x_j)$ and $b(x_j) = \{g''f/2 + g'f'\}(x_j)$. It follows that

$$\mu_D(x_j^t) = a(x_j)(nh)^{-1/2} t + b(x_j) \left[(nh)^{-1} t^2 + h^2 \sigma_K^2 \right] + O((nh)^{-3/2} t^3 + h^3).$$

Again, the error terms are all uniform.

We now calculate t_j^*. The value of t which solves $\mu_D(x_j^t) = 0$ solves

$$t = -\frac{\sqrt{nh}}{a(x_j)} \left\{ b(x_j) \left[(nh)^{-1} t^2 + h^2 \sigma_K^2 \right] (1 + o(1)) + O((nh)^{-3/2} t^3 + h^3) \right\}.$$

Since $|t| \leq \sqrt{nh}\delta_n$ and $\delta_n = o(h)$ with $h = o(1)$ this simplifies to

$$\begin{aligned} t &= -\frac{\sqrt{nh}}{a(x_j)} \left\{ b(x_j) \left[(nh)^{-1} t^2 + h^2 \sigma_K^2 \right] (1 + o(1)) + O((nh)^{-3/2} t^3 + h^3) \right\} \\ &= -\frac{b(x_j)}{a(x_j)} \sigma_K^2 O(\sqrt{nh^5}) + \sqrt{nh}\delta_n^2 O(1) = O(\sqrt{nh^5}). \end{aligned}$$

That is $t_j^* \leq O(1)$, since we assume that $\sqrt{nh^5}$ and $\sqrt{nh}\delta_n^2$ are both at most $O(1)$.

From the above calculations, we find that for $|t| \leq \sqrt{nh}\delta_n$ we have

$$\begin{aligned}
\mu_D^2(x_j^t) &= \left\{a(x_j)(nh)^{-1/2}t + b(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right] + O((nh)^{-3/2}t^3 + h^3)\right\}^2 \\
&= a^2(x_j)(nh)^{-1}t^2 + b^2(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right]^2 + O((nh)^{-3/2}t^3 + h^3)^2 \\
&\quad +2a(x_j)b(x_j)(nh)^{-1/2}t\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right] \\
&\quad +2\left\{a(x_j)(nh)^{-1/2}t + b(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right]\right\}O((nh)^{-3/2}t^3 + h^3) \\
&= a^2(x_j)(nh)^{-1}t^2 + b^2(x_j)\left[(t/\sqrt{nh})^4 + 2h^2\sigma_K^2(t/\sqrt{nh})^2 + h^4\sigma_K^4\right] \\
&\quad +O((t/\sqrt{nh})^6 + (t/\sqrt{nh})^3h^3 + h^6) + O((t/\sqrt{nh})^3 + h^2(t/\sqrt{nh})) \\
&\quad +O((t/\sqrt{nh})^4 + (t/\sqrt{nh})h^3 + (t/\sqrt{nh})^5 + h^5 + h^3(t/\sqrt{nh})^2 + h^2(t/\sqrt{nh})^3) \\
&= a^2(x_j)(nh)^{-1}t^2 + O(\delta_n^3 + \delta_n h^2) \\
&= h^{-1}\left[a^2(x_j)t^2/n + O(h\delta_n^3 + h^3\delta_n)\right] \quad = \quad h^{-1}\left[a^2(x_j)t^2/n + O(h^3\delta_n)\right],
\end{aligned}$$

since $\delta_n = o(h)$. Let $\sigma_D^2(x) = \mathrm{Var}(D_n(x))$ and note that $g_2(x_j) - \lambda^2 = \sigma_Y^2(x_j)$. Then

$$\begin{aligned}
n\sigma_D^2(x_j^t) + \mu_D^2(x_j^t) &= E[D_{n,i}^2(x_j^t)] \\
&= \int \frac{1}{h^2}(g_2(y) - 2\lambda g(y) + \lambda^2)K^2\left(\frac{x_j^t - y}{h}\right)f(y)dy \\
&= h^{-1}\int (g_2(x_j + \eta) - 2\lambda g(x_j + \eta) + \lambda^2)f(x_j + \eta)K^2(s)ds \\
&= h^{-1}\int \left[\sigma_Y^2(x_j) + \{g_2' - 2\lambda g'\}(x_j^*)\eta\right]\left[f(x_j) + f'(x_j^*)\eta\right]K^2(s)ds \\
&= h^{-1}\Bigg[\{\sigma_Y^2 f\}(x_j)v_{2,0} \\
&\qquad + \int \left[\{g_2' - 2\lambda g'\}(x_j^*)f(x_j) + \{\sigma_Y^2\}(x_j)f'(x_j^*)\right]\eta K^2(s)ds \\
&\qquad + \int \{f'(g_2' - 2\lambda g')\}(x_j^*)\eta^2 K^2(s)ds\Bigg]
\end{aligned}$$

We now introduce the additional derivative conditions on f, g, g_2 to tighten up the error

calculations. Continuing from above, we have

$$\begin{aligned}
& n\sigma_D^2(x_j^t) + \mu_D^2(x_j^t) \\
&\leq \; h^{-1}\Bigg[\{\sigma_Y^2 f\}(x_j)v_{2,0} \\
&\qquad + \int \left[\{g_2' - 2\lambda g'\}(x_j)f(x_j) + \{\sigma_Y^2\}(x_j)f'(x_j)\right]\eta K^2(s)ds \\
&\qquad + \int \left[|\{g_2' - 2\lambda g'\}'(x_j^*)f(x_j)| + |\{\sigma_Y^2\}(x_j)f''(x_j^*)|\right]\eta^2 K^2(s)ds \\
&\qquad + \int \{f'(g_2' - 2\lambda g')\}(x_j^*)\eta^2 K^2(s)ds\Bigg] \\
&= \; h^{-1}\left[\{\sigma_Y^2 f\}(x_j)v_{2,0} + O(1)\{(nh)^{-1/2}t\} + O(1)\left\{(nh)^{-1}t^2 + h^2 v_{2,2}\right\}\right],
\end{aligned}$$

where $v_{2,2} = \int s^2 K^2(s)ds < \infty$ and using the fact that $K^2(s)$ is also symmetric (since $K(s)$ is) to remove the $O(h)$ term. Therefore,

$$n\sigma_D^2(x_j^t) + \mu_D^2(x_j^t) \;=\; h^{-1}\left[\{\sigma_Y^2 f\}(x_j)v_{2,0} + O((nh)^{-1/2}t + h^2)\right]$$

where the error is uniform. It follows that

$$\begin{aligned}
\sigma_D^2(x_j^t) &= (nh)^{-1}\Big\{\left[\{\sigma_Y^2 f\}(x_j)v_{2,0} + O((nh)^{-1/2}t + h^2)\right] - \left[a^2(x_j)t^2/n + O(\delta_n h^3)\right]\Big\} \\
&= (nh)^{-1}\left[\{\sigma_Y^2 f\}(x_j)v_{2,0} + O((nh)^{-1/2}t + h^2 + n^{-1}t^2)\right]
\end{aligned}$$

Returning to μ_D, we have that for $|t| \leq \sqrt{nh}\delta_n$ with $\delta_n = o(h)$,

$$\begin{aligned}
\mu_D(x_j^t) &= (nh)^{-1/2}\left[a(x_j)t + b(x_j)\left[(nh)^{-1/2}t^2 + \sqrt{nh^5}\sigma_K^2\right] + O((nh)^{-1}t^3 + \sqrt{nh^7})\right] \\
&= (nh)^{-1/2}\left[a(x_j)t + b(x_j)\sqrt{nh^5}\sigma_K^2 + O((nh)^{-1/2}t^2 + \sqrt{nh^7})\right].
\end{aligned}$$

This implies that

$$\begin{aligned}
-\frac{\mu_D(x_j^t)}{\sigma_D(x_j^t)} &= -\frac{(nh)^{-1/2}\left[a(x_j)t + b(x_j)\sqrt{nh^5}\sigma_K^2 + O((nh)^{-1/2}t^2 + \sqrt{nh^7})\right]}{(nh)^{-1/2}\left[\{\sigma_Y^2 f v_{2,0}\}^{1/2}(x_j) + O((nh)^{-1/2}t + h^2 + n^{-1}t^2)\right]} \\
&= -\frac{a(x_j)}{\{\sigma_Y^2 f v_{2,0}\}^{1/2}(x_j)}t - \frac{b(x_j)\sigma_K^2}{\{\sigma_Y^2 f v_{2,0}\}^{1/2}(x_j)}\sqrt{nh^5} + O(th^2 + (nh)^{-1/2}t^2 + \sqrt{nh^7})
\end{aligned}$$

Lastly, we compute $\rho_1(x_j^t)/\sigma_1^3(x_j^t)$, and note that $\sigma_1^3(x_j^t) = (n\sigma_D^2(x_j^t))^{3/2}$. To complete this last calculation we need to evaluate the numerator,

$$\rho_1(x_j^t) = E[|R_{n,j} - \lambda S_{n,j} - \mu_D(x_j^t)|^3] \leq 4(E[|R_{n,j} - \lambda S_{n,j}|^3] + |\mu_D(x_j^t)|^3).$$

First,

$$\begin{aligned}
&E[|R_{n,j} - \lambda S_{n,j}|^3] \\
&= \int \frac{1}{h^3} E[|Y - \lambda|^3 \big| X = y] K^3\left(\frac{x_j^t - y}{h}\right) f(y)dy \\
&\leq 4\int \frac{1}{h^3}\{E[|Y|^3 \big| X = y] + |\lambda|^3\} K^3\left(\frac{x_j^t - y}{h}\right) f(y)dy \\
&= 4\int \frac{1}{h^3}\{g_3(y) + |\lambda|^3\} K^3\left(\frac{x_j^t - y}{h}\right) f(y)dy \\
&= 4h^{-2}\int \{g_3(x_j + \eta) + |\lambda|^3\} f(x_j + \eta) K^3(s)ds \\
&= 4h^{-2}\int \{g_3(x_j) + g_3'(x_j^*)\eta + |\lambda|^3\}\left\{f(x_j) + f'(x_j^*)\eta\right\} K^3(s)ds.
\end{aligned}$$

Now, since $(nh)^{-1/2}t \leq \delta_n \leq 1$ and $h = o(1) \leq 1$ it follows that $|\eta| \leq 1 + |s|$. It follows that from this and the assumption that $g_3(y) = E[|Y|^3|X = y]$ is locally C_1 that

$$E[|R_{n,j} - \lambda S_{n,j}|^3] \leq O(1)h^{-2}\int \{1 + s^2\} K^3(s)ds = O(h^{-2}).$$

Furthermore,

$$\begin{aligned} |\mu_D(x_j^t)|^3 &= \left| a(x_j)(nh)^{-1/2}t + b(x_j)\left[(nh)^{-1}t^2 + h^2\sigma_K^2\right] + O((nh)^{-3/2}t^3 + h^3)\right|^3 \\ &= O(\delta_n^3 + h^6) = o(h^3). \end{aligned}$$

From before we have that $\sigma_1^2(x_j^t) = n\sigma_D^2(x_j^t) = O(h^{-1})$, and hence

$$\frac{\rho_1(x_j^t)}{\sigma_1^3(x_j^t)} = h^{-1/2}O(1),$$

as required, completing the proof. □

A.2 Proof of Corollary 2

Corollary 2. *Assume the conditions in Theorem 1 hold. Furthermore, assume that* $nh^5 \to c$ *for* $0 < c < \infty$. *Then, there exists a unique* $c_{opt} \in (0, \infty)$ *depending on* g, f *and* K; *but not on* n, *such that* $h_{opt} = \arg\min_{h\in(0,\infty)} E[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)]$ *and it satisfies*

$$h_{opt} = c_{opt}n^{-1/5}$$

Proof. Recall that for $c = (nh^5)^{1/5}$

$$\lim_{n\to\infty} n^{2/5}E[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)] = \sum_{j=1}^m q(x_j)B_{1,j}\left\{2\,\frac{\phi(c^{5/2}B_{2,j})}{c^{1/2}} + c^2B_{2,j}\left[2\Phi(c^{5/2}B_{2,j}) - 1\right]\right\}.$$

For simplicity in exposition let $s = c^{1/2}$ or equivalently $s = (nh^5)^{1/10}$. Then, denote

$$u_j(s) \equiv \left\{2\,\frac{\phi(s^5B_{2,j})}{s} + s^4B_{2,j}\left[2\Phi(s^5B_{2,j}) - 1\right]\right\},$$

and

$$\lim_{n\to\infty} n^{2/5}E[\mu_q\left(\mathrm{LS}_\lambda \triangle \widehat{\mathrm{LS}}_\lambda\right)] = \sum_{j=1}^m q(x_j)B_{1,j}u_j(s) \equiv u(s). \tag{A.2}$$

We now show that there exists an s_{min} where $u'(s = s_{min}) = 0$,

$$u'_j(s) = -2s^{-2}\phi(s^5 B_{2,j}) + 4s^3 B_{2,j}\left[2\Phi(s^5 B_{2,j}) - 1\right]$$

with $u'_j(0_+) = -\infty$ and $u'_j(\infty) = \infty$. Next,

$$u''_j(s) = 4s^{-3}\phi(s^5 B_{2,j}) + 50s^7 B_{2,j}^2\phi(s^5 B_{2,j}) + 12s^2 B_{2,j}\left[2\Phi(s^5 B_{2,j}) - 1\right].$$

Note that $B_{1j} > 0\ \forall j$ and $\text{sign}(B_{2j}) = \text{sign}\left(2\Phi(B_{2j}s) - 1\right)$ thus $u''_j(s) > 0$ for every s. This implies that $u''(s) = \sum_{j=1}^m q(x_j)B_{1,j}u''_j(s) > 0$ for every s and therefore $u'(s) = \sum_{j=1}^m q(x_j)B_{1,j}u'_j(s)$ is monotone with $u'(0) = -\infty$ and $u'(\infty) = \infty$. Then it exists a unique s_{min} such that $u'(s_{min}) = 0$.

□

A.3 Proof of Theorem 3

We start by introducing an auxiliary result.

Theorem 6 (page 144 Fan et al. [1995])**.** *Assume the regression model $\Psi\left(g(x)\right) = \eta(x)$ where $g(x) = E[Y|X = x]$. Let $\eta^{(r)}$ be the r^{th} derivative that is estimated with a p^{th} degree polynomial. Fix $p - r > 0$ to be odd and assume that $h = h_n \to 0$ and $nh^3 \to \infty$ as $n \to \infty$. If x is a fixed point in the interior of the support of f, under the following conditions,*

1.- *The function Ψ is the canonical link.*

2.- *The functions f', $\eta^{(p+2)}$, $\text{Var}(Y|X = x)$, $\text{Var}^{(2)}(Y|X = x)$ and $\Psi^{(3)}$ are continuous.*

3.- *For each $x \in \text{supp}(f)$, the functions $\left\{\Psi'\left(g(x)\right)\text{Var}(Y|X = x)\right\}^{(-1)}$, $\text{Var}(Y|X = x)$ and $\Psi'\left(g\left(x\right)\right)$ are non zero.*

4.- *The kernel K is a symmetric probability density with support $[-1, 1]$.*

5.- *Assumption (F) holds.*

the following holds

$$\sqrt{nh^{2r+1}}\sigma_{r,p}(x;K)^{-1} \times [\widehat{\eta}_r(x;p,h) \\ -\eta^{(r)}(x) - \left\{\int z^{p+1}K_{r,p}(z)dz\right\}\left\{\frac{\eta^{(p+1)}(x)}{(p+1)!}\right\}h^{p-r+1}\{1+O(h)\}\Bigg] \\ \xrightarrow{D} N(0,1)$$

where

$$\begin{aligned}\sigma^2_{r,p}(x;K) &= \mathrm{Var}(Y|X=x)\Psi'\left(g(x)\right)^2 f(x)^{-1}\int K_{r,p}(z)^2dz \\ K_{r,p}(z) &= r!\frac{|M_{r,p}(z)|}{|N_p|}K(z)\end{aligned}$$

where N_p is a $(p+1)\times(p+1)$ matrix with (i,j) entry $N_{i,j}=\int z^{i+j-2}K(z)dz$. The matrix $M_{r,p}$ is the same as N_p but the $(r+1)st$ column is replaced by $(1,z,\ldots,z^p)^T$.

Lemma 7. *Let the conditions in Theorem 6 hold. Fix $p=r+1$ and let b_n be a sequence such that $b_n\to 0$ as $n\to\infty$. Furthermore, assume that $nh_r^{2r+5}\to c$ for some constant c. Then for any fixed $x\in\sup(f)$*

$$|\widehat{\eta}_r(x;p,h_r)-\eta^{(r)}(x)| = O_p\left(n^{-2/2r+5}\right).$$

Proof. Assume the conditions hold. Then, by Markov

$$P\left(\left[\,\widehat{\eta}_r(x;p,h_r)-\eta^{(r)}(x)\,\right]^2 > M^2b_n^2\right) \leq \frac{\mathrm{E}\left[\left(\widehat{\eta}_r(x;p,h_r)-\eta^{(r)}(x)\right)^2\right]}{M^2b_n^2}.$$

Moreover, we can decompose the expectation above as

$$\mathrm{E}\left[\left(\widehat{\eta}_r(x;p,h_r)-\eta^{(r)}(x)\right)^2\right] = \mathrm{Var}\left(\widehat{\eta}_r(x;p,h_r)\right)+\left(\mathrm{E}[\widehat{\eta}_r(x;p,h_r)]-\eta^{(r)}\right)^2.$$

From Theorem 6, $\mathrm{Var}\left(\widehat{\eta}_r(x;p,h_r)\right) = O_p\left(\frac{1}{nh_r^{2r+1}}\right)$ and $\left(\mathrm{E}[\widehat{\eta}_r(x;p,h_r)]-\eta^{(r)}\right)^2 = O_p(h_r^4)$.

Because $nh_r^{2r+5} \to c$,

$$P\left(\left[\,\widehat{\eta}_r(x;p,h_r) - \eta^{(r)}(x)\,\right]^2 > M^2 b_n^2\right) \quad \leq \quad \frac{O\left(n^{-4/2r+5}\right)}{M^2 b_n^2}.$$

We thus deduce that

$$|\widehat{\eta}_r(x;p,h_r) - \eta^{(r)}(x)| \quad = \quad O_p\left(n^{-2/2r+5}\right)$$

completing the proof. □

Theorem 3. *Assume the conditions in Theorem 6 hold. In addition, assume*

(S.2) f is $C_2(\widetilde{B})$ and g is $C_3(\widetilde{B})$.

(Q) The function q is a non-negative function such that $|\widehat{q}(x) - q(x)|$ is at most $O_p\left(n^{-2/9}\right)$ in a neighbourhood of x_j.

Let $c = n^{1/5}h$ and define the asymptotic risk function in terms of c as

$$AR(c) \quad = \quad n^{-2/5}\sum_{j=1}^{m} q(x_j)B_{1,j}\left\{2\,\frac{\phi(c^{5/2}B_{2,j}}{c^{1/2}} + c^2 B_{2,j}\left[2\Phi(c^{5/2}B_{2,j}) - 1\right]\right\}.$$

Also define $\widehat{AR}_n(c)$ as the asymptotic risk function where x_j,q,f,f',g' and g'' are replaced with kernel estimators. Then

$$\left|AR(c) - \widehat{AR}_n(c)\right| = O_p\left(n^{-2/9}\right).$$

and

$$\frac{\widehat{h}_{opt}}{h_{opt}} \quad = \quad 1 + O_p\left(n^{-2/9}\right)$$

Proof. With a slight abuse of notation, let

$$AR(c) \quad \equiv \quad AR\left(c; x_j, q, f, f', g', g''\right).$$

We are thus interested in the difference

$$\begin{aligned}\left|AR(c;x_j,q,f,f',g',g'') - AR(c;\widehat{x}_j,\widehat{q},\widehat{f},\widehat{f}',\widehat{g}',\widehat{g''})\right| \leq \\ \left|AR(c;\widehat{x}_j,q,f,f',g',g'') - AR(c;\widehat{x}_j,\widehat{q},\widehat{f},\widehat{f}',\widehat{g}',\widehat{g''})\right| \\ + \left|AR(c;x_j,q,f,f',g',g'') - AR(c;\widehat{x}_j,q,f,f',g',g'')\right| \\ \equiv D_1 + D_2.\end{aligned}$$

For ease in notation, we omitted the bandwidths for the estimators.

We begin with D_1. We first study the differences of the functions f, f', g' and g'', and their kernel estimators. From the regression model $g(x) = \Psi^{-1}(\eta(x))$. Thus,

$$\begin{aligned} g'(x) &= \frac{\partial}{\partial\eta}\Psi^{-1}(\eta(x))\,\eta'(x) \\ g''(x) &= \frac{\partial^2}{\partial\eta^2}\Psi^{-1}(\eta(x))\left(\eta'(x)\right)^2 + \frac{\partial}{\partial\eta}\Psi^{-1}(\eta(x))\,\eta''(x). \end{aligned}$$

We next use Lemma 7 to compute plug-in estimators for g' and g''. We start with the first derivative

$$\begin{aligned} \widehat{g'}(x) &= \frac{\partial}{\partial\eta}\Psi^{-1}(\widehat{\eta}_{h_0}(x))\,\widehat{\eta'}_{h_1}(x) \\ &= \frac{\partial}{\partial\eta}\Psi^{-1}\left(\eta(x) + O_p\left(n^{-2/5}\right)\right)\left(\eta'(x) + O_p\left(n^{-2/7}\right)\right) \end{aligned}$$

Since the derivative of Ψ^{-1} exists, it then follows that

$$\begin{aligned} \widehat{g'}(x) &= \left[\frac{\partial}{\partial\eta}\Psi^{-1}(\eta(x)) + O_p\left(n^{-2/5}\right)\right]\left[\eta'(x) + O_p\left(n^{-2/7}\right)\right] \\ &= \frac{\partial}{\partial\eta}\Psi^{-1}(\eta(x))\,\eta'(x) + O_p\left(n^{-2/7}\right). \end{aligned}$$

Now, for the second derivative

$$\begin{aligned}
\widehat{g''}(x) &= \frac{\partial^2}{\partial\eta^2}\Psi^{-1}\left(\widehat{\eta}_{h_0}(x)\right)\left(\widehat{\eta'}_{h_1}(x)\right)^2 + \frac{\partial}{\partial\eta}\Psi^{-1}\left(\widehat{\eta}_{h_0}(x)\right)\widehat{\eta''}_{h_2}(x) \\
&= \frac{\partial^2}{\partial\eta^2}\Psi^{-1}\left(\eta(x)+O_p\left(n^{-2/5}\right)\right)\left(\eta'(x)+O_p\left(n^{-2/7}\right)\right)^2 \\
&\qquad + \frac{\partial}{\partial\eta}\Psi^{-1}\left(\eta(x)+O_p\left(n^{-2/5}\right)\right)\left(\eta''(x)+O_p\left(n^{-2/9}\right)\right) \\
&= \frac{\partial^2}{\partial\eta^2}\Psi^{-1}\left(\eta(x)\right)\left(\eta'(x)\right)^2 + \frac{\partial}{\partial\eta}\Psi^{-1}\left(\eta(x)\right)\eta''(x) + O_p\left(n^{-2/9}\right).
\end{aligned}$$

Thus, uniformly

$$\begin{aligned}
|\widehat{g'} - g'(x)| &= O_p\left(n^{-2/7}\right) \\
|\widehat{g''} - g''(x)| &= O_p\left(n^{-2/9}\right).
\end{aligned} \tag{A.3}$$

Using the same type of arguments, from Wand and Jones [1994][Chapter 2], it can be shown that

$$\begin{aligned}
|\widehat{f}_{h00} - f(x)| &= O_p\left(n^{-2/5}\right) \\
|\widehat{f'}_{h11} - f'(x)| &= O_p\left(n^{-2/7}\right).
\end{aligned} \tag{A.4}$$

We then show the plug-in estimates for $A(x_j)$ and $B(x_j)$. For the first,

$$\begin{aligned}
\widehat{A}(\widehat{x_j}) &= -\left\{\frac{\widehat{g'}\widehat{f}}{(\widehat{\sigma}_Y^2\widehat{f}v_{2,0})^{1/2}}\right\}(\widehat{x_j}) \\
&= -\left\{\frac{[g'(\widehat{x_j})+O_p(n^{-2/7})][f(\widehat{x_j})+O_p(n^{-2/5})]}{\left([\sigma_Y^2+O_p(n^{-1/2})][f(\widehat{x_j})+O_p(n^{-2/5})]v_{2,0}\right)^{1/2}}\right\} \\
&= -\frac{\{g'f\}(\widehat{x_j})\left(1+O_p(n^{-2/7})\right)}{(\sigma_Y^2 f(\widehat{x_j})v_{2,0})^{1/2}\left(1+O_p(n^{-2/5})\right)^{1/2}} \\
&= A\left(\widehat{x_j}\right)\{1+O_p\left(n^{-2/7}\right)\}
\end{aligned}$$

and

$$\begin{aligned}
\widehat{B}(\widehat{x_j}) &= -\sqrt{nh^5}\left\{\frac{\widehat{g''}\widehat{f}/2+\widehat{g'}\widehat{f'}}{(\widehat{\sigma}_Y^2\widehat{f}v_{2,0})^{1/2}}\right\}(\widehat{x_j})\sigma_K^2 \\
&= -\sqrt{nh^5}\sigma_K^2\frac{\{g''f/2\}(\widehat{x_j})+O_p\left(n^{-2/9}\right)+\{g'f'\}(\widehat{x_j})+O_p\left(n^{-2/7}\right)}{(\sigma_Y^2 f(\widehat{x_j})v_{2,0})^{1/2}\left(1+O_p(n^{-2/5})\right)^{1/2}} \\
&= -\sqrt{nh^5}\sigma_K^2\left\{\frac{g''f/2+g'f'}{(\sigma_Y^2 f v_{2,0})^{1/2}}\right\}(\widehat{x_j})\{1+O_p\left(n^{-2/9}\right)\}\{1+O_p\left(n^{-2/5}\right)\} \\
&= B\left(\widehat{x_j}\right)\{1+O_p\left(n^{-2/9}\right)\}.
\end{aligned}$$

Hence,

$$AR(c;\widehat{x_j},\widehat{q},\widehat{f},\widehat{f'},\widehat{g'},\widehat{g''}) = \\ n^{-2/5}\sum_{j=1}^{m} q(\widehat{x_j})\widehat{B}_1(\widehat{x_j})\left\{2\,\frac{\phi(c^{5/2}\widehat{B}_2(\widehat{x_j})}{c^{1/2}}+c^2\widehat{B}_2(\widehat{x_j})\left[2\Phi(c^{5/2}\widehat{B}_2(\widehat{x_j}))-1\right]\right\}$$

where

$$\begin{aligned}
\widehat{B}_1(\widehat{x_j}) &= \frac{1}{|\widehat{A}(\widehat{x_j})|} = B_1(\widehat{x_j})\{1+O_p(n^{-2/7})\} \\
\widehat{B}_2(\widehat{x_j}) &= \frac{\widehat{B}(\widehat{x_j})}{\sqrt{nh^5}} = B_2(\widehat{x_j})\{1+O_p(n^{-2/7})\}.
\end{aligned}$$

The density and distribution functions are continuous, thus

$$\begin{aligned}
\phi\left(c^{5/2}\widehat{B}_2(\widehat{x_j})\right) &= \phi\left(c^{5/2}B_2(\widehat{x_j})\right)+O_p(n^{-2/9}) \\
\Phi\left(c^{5/2}\widehat{B}_2(\widehat{x_j})\right) &= \Phi\left(c^{5/2}B_2(\widehat{x_j})\right)+O_p(n^{-2/9}).
\end{aligned}$$

We now plug-in the estimators into the asymptotic risk (as long as $|\widehat{q}(\widehat{x_j})-q(\widehat{x_j})| = O_p(n^{-2/9})$

or faster)

$$
\begin{aligned}
&AR(c;\widehat{x_j},\widehat{q},\widehat{f},\widehat{f'},\widehat{g'},\widehat{g''}) = \\
&\quad n^{-2/5}\sum_{j=1}^{2r}\left[q(\widehat{x_j})+O_p(n^{-2/9})\right]\left[B_1(\widehat{x_j})+O_p(n^{-2/7})\right]\left\{\frac{2\phi(c^{5/2}B_2(\widehat{x_j}))+O_p(n^{-2/9})}{c^{1/2}}\right. \\
&\qquad\left. +c^2\left[B_2(\widehat{x_j})+O_p(n^{-2/9})\right]\left[2\Phi\left(c^{5/2}B_2(\widehat{x_j})\right)+O_p(n^{-2/9})-1\right]\right\} \\
&= n^{-2/5}\sum_{j=1}^{m} q(\widehat{x_j})B_1(\widehat{x_j})\left\{2\,\frac{\phi(c^{5/2}B_2(\widehat{x_j})}{c^{1/2}}+c^2B_2(\widehat{x_j})\left[2\Phi(c^{5/2}B_2(\widehat{x_j}))-1\right]\right\}+O_p(n^{-2/9})
\end{aligned}
$$

We therefore conclude that $D_1 = O_p\left(n^{-2/9}\right)$.

We now look into D_2. Recall from Fan et al. [1995] that

$$
g(\widehat{x_j}) \quad = \quad \widehat{g}(\widehat{x_j}) + O_p(n^{-2/5}). \tag{A.5}
$$

From condition (S), we do a Taylor expansion on the left hand side. This yields

$$
g(x_j) + g'(\xi)\left(\widehat{x_j} - x_j\right) \quad = \quad \widehat{g}(\widehat{x_j}) + O_p\left(n^{-2/5}\right).
$$

Next we show that $g'(\xi) \neq 0$ for some $\xi \in [\widehat{x_j}, x_j]$. We start by showing that for a small ε_0 and a point $\widehat{x_j}$ such that $\widehat{g}\,(\widehat{x_j}) = \lambda$, it is true that $\widehat{x_j} \in [x_j - \varepsilon_0, x_j + \varepsilon_0]$. We prove it by contradiction. Without loss of generality, assume $g(x_j - \varepsilon_0) - \lambda > 0$ and $g(x_j + \varepsilon_0) - \lambda < 0$. Since $\widehat{g}$ has no roots in the interval, assume $\widehat{g}(x) - \lambda > 0\ \forall x \in [x_j - \varepsilon_0, x_j + \varepsilon_0]$. Therefore

$$
\sup_{x\in[x_j-\varepsilon_0,x_j+\varepsilon_0]}\left|\left(\widehat{g}(x)-\lambda\right)-\left(g(x)-\lambda\right)\right| \quad \geq \quad |g(x_j+\varepsilon_0)-\lambda|.
$$

In particular,

$$
\lim_{n\to\infty}\sup_{x\in[x_j-\varepsilon_0,x_j+\varepsilon_0]}\left|\widehat{g}(x)-g(x)\right| \quad \geq \quad |g(x_j+\varepsilon_0)-\lambda|
$$

which contradicts the initial assumption. We thus proved that $\widehat{x_j} \in [x_j - \varepsilon_0, x_j + \varepsilon_0]$ and

therefore $\xi \in [x_j - \varepsilon_0, x_j + \varepsilon_0]$ too. Note that $g'(x) \neq 0$ for $x \in [x_j - \varepsilon_0, x_j + \varepsilon_0]$ since g is monotone locally on x_j. It then follows that $g'(\xi) \neq 0$. Going back to A.5, we obtain that $|\widehat{x_j} - x_j| = O_p(n^{-2/5})$. From condition (S.2)

$$\begin{aligned}
g'(\widehat{x_j}) &= g'(x_j) + (\widehat{x_j} - x_j)\, g''(x_j) \\
g''(\widehat{x_j}) &= g''(x_j) + (\widehat{x_j} - x_j)\, g^{(3)}(x_j) \\
f(\widehat{x_j}) &= f(x_j) + (\widehat{x_j} - x_j)\, f'(x_j) \\
f'(\widehat{x_j}) &= f'(x_j) + (\widehat{x_j} - x_j)\, f''(x_j).
\end{aligned}$$

Thus,

$$\begin{aligned}
\left|g'(\widehat{x_j}) - g'(x_j)\right| &= O_p\left(n^{-2/5}\right) \qquad \left|f(\widehat{x_j}) - f(x_j)\right| = O_p\left(n^{-2/5}\right) \\
\left|g''(\widehat{x_j}) - g''(x_j)\right| &= O_p\left(n^{-2/5}\right) \qquad \left|f'(\widehat{x_j}) - f'(x_j)\right| = O_p\left(n^{-2/5}\right).
\end{aligned}$$

From these results

$$\begin{aligned}
A(\widehat{x_j}) &= -\frac{g'(x_j) f(x_j) \{1 + O_p\left(n^{-2/5}\right)\}}{\left(f \sigma_Y^2 v_{2,0}\right)^{1/2} \{1 + O_p\left(n^{-2/5}\right)\}^{1/2}} \\
&= A(x_j)\{1 + O_p\left(n^{-2/5}\right)\} \\
B(\widehat{x_j}) &= -\sqrt{nh^5} \frac{\{g'' f/2 + g' f'\}(x_j) \left(1 + O_p\left(n^{-2/5}\right)\right)}{\left(f \sigma_Y^2 v_{2,0}\right)^{1/2} \{1 + O_p\left(n^{-2/5}\right)\}^{1/2}} \\
&= B(x_j)\{1 + O_p\left(n^{-2/5}\right)\}
\end{aligned}$$

and

$$\begin{aligned}
B_1(\widehat{x_j}) &= B_1(x_j)\{1 + O_p\left(n^{-2/5}\right)\} \\
B_2(\widehat{x_j}) &= B_2(x_j)\{1 + O_p\left(n^{-2/5}\right)\}.
\end{aligned}$$

After some algebra

$$
\begin{aligned}
&AR(c;\widehat{x}_j,q,f,f',g',g'') = \\
&\qquad n^{-2/5}\sum_{j=1}^{m} q(\widehat{x}_j)B_1(\widehat{x}_j)\left\{2\,\frac{\phi(c^{5/2}B_2(\widehat{x}_j)}{c^{1/2}} + c^2 B_2(\widehat{x}_j)\left[2\Phi(c^{5/2}B_2(\widehat{x}_j))-1\right]\right\} \\
&= n^{-2/5}\sum_{j=1}^{m} q(x_j)B_1(x_j)\left\{2\,\frac{\phi(c^{5/2}B_2(x_j)}{c^{1/2}} + c^2 B_2(x_j)\left[2\Phi(c^{5/2}B_2(x_j))-1\right]\right\} + O_p\left(n^{-2/5}\right).
\end{aligned}
$$

This yields $D_2 = O_p\left(n^{-2/5}\right)$, therefore $\left|\widehat{AR}_n(c) - AR(c)\right| = O_p\left(n^{-2/9}\right)$ uniformly. Moreover

$$
\begin{aligned}
\left|\widehat{AR}_n(\widehat{c}_{opt}) - AR(\widehat{c}_{opt})\right| &= \left|\widehat{AR}_n(\widehat{c}_{opt}) - \left(AR(\widehat{c}_{opt}) - AR(c_{opt})\right)\right| \\
&= \left|\widehat{AR}_n(\widehat{c}_{opt}) - AR(c^*)\left(\widehat{c}_{opt} - c_{opt}\right)\right| \text{ for some } c^* \in (\widehat{c}_{opt}, c_{opt}) \\
&= \left|AR(c^*)\left(\widehat{c}_{opt} - c_{opt}\right)\right| \;=\; O_p\left(n^{-2/9}\right).
\end{aligned}
$$

Note $AR''(c) > 0$ and bounded away from zero. Therefore

$$
\frac{\widehat{c}_{opt}}{c_{opt}} \;=\; 1 + O_p\left(n^{-2/9}\right)
$$

completing the proof. □

Appendix B

Code section

B.1 Example code for level set simulations

Common functions

```
######     LIBRARIES      #####

library(KernSmooth)
library(np)

####  GENERATE RANDOM OBSERVATIONS FROM f(X) ####

X.sample<-function(n,min,max){
  xx<-runif(n,min,max)
  return(xx)
}

#####   NADARAYA-WATSON KERNEL ESTIMATOR  USING BINNING    #####

NWkernel_regression_gaussian_binned    <- function(x,h,grid,cl,dlY){
```

```
  subfunction<-function(x1D,h){
    argument<-(x1D-grid)/h
    denom<-sum(dnorm(x=argument)*cl)
    numerator<-dnorm(x=argument)*dlY
    kk<-sum(numerator/denom)
    return(kk)
  }
  ghat_vector<-unlist(lapply(x,function(xx)subfunction(x1D=xx,h=h)))
  return(ghat_vector)
}

####  FUNCTION THAT COMPUTES THE GRID AND WEIGHTS FOR BINNING
   ########
linear_binning_fast  <-  function(xsample,Y.sample,nbins){

  delta             <-  (max(xsample)-min(xsample))/(nbins-1)
  rescale_xsample   <-  xsample/delta
  vector_floors     <-  floor(rescale_xsample)
  vector_ceilings   <-  ceiling(rescale_xsample)

  grid_weights_cl      <-  rep(0,nbins+1)
  grid_weights_dl      <-  rep(0,nbins+1)
  grid_weights_dl_2    <-  rep(0,nbins+1)

  l1     <-  min(floor(min(rescale_xsample)),ceiling(min(rescale_
     xsample)))
  l2     <-  max(floor(max(rescale_xsample)),ceiling(max(rescale_
     xsample)))
  lseq   <-  seq(from=l1,to=l2,by=1)

  lconstant  <- -l1+1
```

```
  for (i in 1:n){

    temp_weight_floor                    <-  1-abs(rescale_xsample[i
       ]-vector_floors[i])
    temp_weight_ceiling                  <-  1-abs(rescale_xsample[i
       ]-vector_ceilings[i])
    grid_weights_cl[vector_floors[i]+lconstant]   <- grid_weights_cl[
       vector_floors[i]+lconstant]+temp_weight_floor
    grid_weights_cl[vector_ceilings[i]+lconstant] <- grid_weights_cl[
       vector_ceilings[i]+lconstant]+temp_weight_ceiling

    grid_weights_dl[vector_floors[i]+lconstant]   <- grid_weights_dl[
       vector_floors[i]+lconstant]+(temp_weight_floor*Y.sample[i])
    grid_weights_dl[vector_ceilings[i]+lconstant] <- grid_weights_dl[
       vector_ceilings[i]+lconstant]+(temp_weight_ceiling*Y.sample[i
       ])

    grid_weights_dl_2[vector_floors[i]+lconstant]   <- grid_weights_
       dl_2[vector_floors[i]+lconstant]+(temp_weight_floor*Y.sample[i
       ]^2)
    grid_weights_dl_2[vector_ceilings[i]+lconstant] <- grid_weights_
       dl_2[vector_ceilings[i]+lconstant]+(temp_weight_ceiling*Y.
       sample[i]^2)
  }

  return(list(grid_points=lseq*delta,cl=grid_weights_cl,dl=grid_
     weights_dl,dl_2=grid_weights_dl_2))

}
```

```
#FUNCTIONS NECESSARY TO ESTIMATE f WITH A KERNEL ESTIMATOR USING
    BINNING

psi_NS_8<-function(sigma.hat){
  psi8<-105/(32*sqrt(pi)*sigma.hat^9)
  return(psi8)
}
d6K<-function(x){
  derivative<-(1/sqrt(2*pi))*exp(-.5*(x^2))*(x^6-15*x^4+45*x^2-15)
  return(derivative)
}
d4K<-function(x){
  derivative<-(1/sqrt(2*pi))*exp(-.5*(x^2))*(x^4-6*x^2+3)
  return(derivative)
}
sigma.hat<-function(sample){
  sigma.sample<-sd(sample)
  IQR.sample<-quantile(sample,.75)-quantile(sample,.25)
  s<-min(sigma.sample,IQR.sample)
  return(s)
}
psi6_binning  <-  function(h_6,grid,counts,n){

  n_grid<-length(grid)
  sum<-0
  for(i in 1:n_grid){
    for(j in 1:n_grid){
      argument<-(grid[j]-grid[i])/h_6
      Lr<-d6K(argument)*counts[j]*counts[i]
      sum<-sum+Lr
    }
  }
```

```
  result<-n^(-2)*h_6^(-7)*sum
  return(result)
}

psi4_binning  <-  function(h_4,grid,counts,n){

  n_grid<-length(grid)
  sum<-0
  for(i in 1:n_grid){
    for(j in 1:n_grid){
      argument<-(grid[j]-grid[i])/h_4
      Lr<-d4K(argument)*counts[j]*counts[i]
      sum<-sum+Lr
    }
  }
  result<-n^(-2)*h_4^(-5)*sum
  return(result)
}

Kernel_density_binning   <-   function(x, n, h, grid,counts){
  K         <-  function(x){
    k_x    <-  (abs(x)<1)*(3/4)*(1-x^2)
    return(k_x)
  }
  subfunction<-function(x1D,h){
    argument<-(x1D-grid)/h
    kk       <- K(x=argument)*counts
    kk       <-sum(kk)*(1/n)*(1/h)
    return(kk)
  }
  fhat_vector <-  unlist(lapply(x,function(xx) subfunction(x1D=xx,h=h
     )))
```

```
  return(fhat_vector)
}


##FUNCTIONS TO ESTIMATE THE DERIVATIVE OF f WITH KERNELS AND BINNING
   ##

df_function      <-     function(x,sigma,mu){
  temporal    <-  -(1/(sqrt(2*pi)*sigma^2))*((x-mu)/sigma)*exp(-.5*((x
     -mu)/sigma)^2)
  return(temporal)
}

d1K              <-     function (x){
  derivative   <-  (abs(x)<1)*(-3/2)*x
  return(derivative)
}
dfhat_binning     <-     function(x,grid,counts,n,h){
  subfunction<-function(x1D,h){
    argument<-(x1D-grid)/h
    kk       <-lapply(argument, d1K)
    kk       <-unlist(kk)*counts
    kk       <-sum(kk)*(1/n)*(1/h^2)
    return(kk)
  }
  dfhat_vector <-  unlist(lapply(x,function(xx) subfunction(x1D=xx,h=
     h)))

  return(dfhat_vector)
}
psi_NS_10     <-   function(sigma.hat){
```

```
  res     <-  -945/(64*sqrt(pi)*sigma.hat^11)
  return(res)
}

d8K           <-   function(x){

  res     <-  (1/sqrt(2*pi))*exp(-(1/2)*x^2)*(x^8-28*x^6+210*x^4-420*x
     ^2+105)
  return(res)
}

psi8_binning  <-  function(h_8,grid,counts,n){

  n_grid<-length(grid)
  sum<-0
  for(i in 1:n_grid){
    for(j in 1:n_grid){
      argument<-(grid[j]-grid[i])/h_8
      Lr<-d8K(argument)*counts[j]*counts[i]
      sum<-sum+Lr
    }
  }
  result<-n^(-2)*h_8^(-9)*sum
  return(result)
}

#########   CROSS VALIDATION  BINNED FUNCTION   ############

#This function is from B.A. Turlach and M.P. Wands paper:
#Fast Computation of Auxiliary Quantities in Local Polynomial
   Regression
```

```
my_CV        <-      function(grid,counts,dY,dY_2,x.sample,Y.sample,h){

  function_h_1D     <-    function(hh){

    d     <-    lapply((1:length(grid)),function(ll) sum(dnorm(x=(grid-
       grid[ll])/hh,mean=0,sd=1)*counts))
    d     <-    unlist(d)
    S     <-    matrix(,nrow=length(grid),ncol=length(grid))

    for(l in 1:length(grid)){
      S[l,]    <-    t(d[l]^(-1)*dnorm(x=(grid-grid[l])/hh,mean=0,sd=1)
          )
    }
    ghat_grid    <-    S%*%dY

    cv_l    <-  function(l){
      temp<-(dY_2[l]-2*ghat_grid[l]*dY[l]+ghat_grid[l]^2*counts[l])/(
          1-S[l,l])^2
      return(temp)
    }

    cv_score    <-    sum(unlist(lapply((1:length(grid)),cv_l)))

    return(cv_score)
  }

  cv_score_vector  <- unlist(lapply(h,function_h_1D))
  return(cv_score_vector)

}

#####    FUNCTIONS TO ESTIMATE THE LEVEL  SETS    ######
```

```
myBFfzero<-function (f, a, b, num = 1000, eps = 1e-05)
{
  h = abs(b - a)/num
  i = 0
  j = 0
  a1 = b1 = 0
  while (i <= num) {
    a1 = a + i * h
    b1 = a1 + h
    if (f(a1) == 0) {
      root<-a1
      f.root<-f(a1)
    }
    else if (f(b1) == 0) {
      root<-b1
      f.root<-f(b1)
    }
    else if (f(a1) * f(b1) < 0) {
      repeat {
        if (abs(b1 - a1) < eps)
          break
        x <- (a1 + b1)/2
        if (f(a1) * f(x) < 0)
          b1 <- x
        else a1 <- x
      }
      #print(j + 1)
      j = j + 1
      root<-(a1 + b1)/2
      f.root<-f((a1 + b1)/2)
    }
```

```
    i = i + 1
  }
  if (j == 0)
    print("finding root is fail")
  else return(list(root=root,f.root=f.root))
}

root_bisection<-function(x0,x1,tol,p,h_estimate,grid,cl,dlY){
  kernel_estim<-function(xx){
    ghat_temp<-NWkernel_regression_gaussian_binned(x=xx,h=h_estimate,
        grid=grid,cl=cl,dlY=dlY)-p
    return(ghat_temp)
  }
  root<-myBFfzero(kernel_estim,x0,x1,eps=tol)$root
  return(root)
}

root_bisectionLP<-function(x0,x1,tol,p,ghatx,ghatLP){
  fhat_interpolation <- splinefun(x=ghatx, y=ghatLP,method="fmm")
  fhat_shifted <- function(xx){
    fhat_temp<-fhat_interpolation(xx)-p
    return(fhat_temp)
  }
  root<-myBFfzero(fhat_shifted,x0,x1,eps=tol)$root
  return(root)
}

grid_search<-function(h,p,xsample,delta,grid,cl,dlY){
  candidates<-seq(from=min(xsample),to=max(xsample),by=delta)

  i<-1
  roots_vector<-NULL
```

```
while(i<length(candidates)){
  x1<-candidates[i]
  x2<-candidates[i+1]

  f.x1<-NWkernel_regression_gaussian_binned(x=x1,h=h,grid=grid,cl=
     cl,dlY=dlY)-p
  f.x2<-NWkernel_regression_gaussian_binned(x=x2,h=h,grid=grid,cl=
     cl,dlY=dlY)-p
  if(f.x1*f.x2>0){
    m<-(x1+x2)/2
    f.m<-NWkernel_regression_gaussian_binned(x=m,h=h,grid=grid,cl=
       cl,dlY=dlY)-p
    if(f.x1*f.m>0){
      i<-i+1
      #print(i)
    }else{
      r1<-root_bisection(x0=x1,x1=m,tol=.00005,p=p,h_estimate=h,
         grid,cl,dlY)
      r2<-root_bisection(x0=m,x1=x2,tol=.00005,p=p,h_estimate=h,
         grid,cl,dlY)
      roots_vector<-c(roots_vector,r1,r2)
      i<-i+1
      #print(i)
    }
  }else{
    r1<-root_bisection(x0=x1,x1=x2,tol=.00005,p=p,h_estimate=h,grid
       ,cl,dlY)
    roots_vector<-c(roots_vector,r1)
    i<-i+1
    #print(i)
  }
}
```

```
  return(roots_vector)
}

grid_searchLP    <-    function(ghatx,ghatLP,p,xsample,delta){

  candidates          <-  seq(from=min(xsample),to=max(xsample),by=
     delta)
  fhat_interpolation  <-  splinefun(x=ghatx,y=ghatLP,method="fmm")

  i<-1
  roots_vector<-NULL
  while(i<length(candidates)){
    x1<-candidates[i]
    x2<-candidates[i+1]
    f.x1<-fhat_interpolation(x1)-p
    f.x2<-fhat_interpolation(x2)-p
    if(f.x1*f.x2>0){
      m<-(x1+x2)/2
      f.m<-fhat_interpolation(m)-p
      if(f.x1*f.m>0){
        i<-i+1
        #print(i)
      }else{
        r1<-root_bisectionLP(x0=x1,x1=m,tol=.00005,p=p,ghatx=ghatx,
           ghatLP=ghatLP)
        r2<-root_bisectionLP(x0=m,x1=x2,tol=.00005,p=p,ghatx=ghatx,
           ghatLP=ghatLP)
        roots_vector<-c(roots_vector,r1,r2)
        i<-i+1
        #print(i)
      }
    }else{
```

```
      r1<-root_bisectionLP(x0=x1,x1=x2,tol=.00005,p=p,ghatx=ghatx,
          ghatLP=ghatLP)
      roots_vector<-c(roots_vector,r1)
      i<-i+1
      #print(i)
    }
  }
  return(roots_vector)
}

myHDR<-function(roots_vector,d,p,h_estimate,grid,cl,dlY){
  r<-length(roots_vector)
  HDR<-list()
  roots_vector<-sort(roots_vector)
  R<-list()
  i<-1
  left_boundary<-NWkernel_regression_gaussian_binned(x=roots_vector[1
      ]-d,h=h_estimate,grid=grid,cl=cl,dlY=dlY)-p
  right_boundary<-NWkernel_regression_gaussian_binned(x=roots_vector[
      r]+d,h=h_estimate,grid=grid,cl=cl,dlY=dlY)-p
  if(left_boundary>0){
    R[[i]]<-c(min(x.sample),roots_vector[1])
    i<-i+1
  }
  if(right_boundary>0){
    R[[i]]<-c(roots_vector[r],max(x.sample))
    i<-i+1
  }
  l<-1
  while(l<length(roots_vector)){
    r1<-roots_vector[l]
    r2<-roots_vector[l+1]
```

```
    rm<-(r1+r2)/2
    f.rm<-NWkernel_regression_gaussian_binned(x=rm,h=h_estimate,grid=
        grid,cl=cl,dlY=dlY)-p
    if(f.rm>0){
      R[[i]]<-c(r1,r2)
      i<-i+1
    }
    l<-l+1
  }
  return(R)
}

myHDRLP<-function(roots_vector,d,p,ghatx,ghatLP,x.sample){
  fhat_interpolation <-  splinefun(x=ghatx,y=ghatLP,method="fmm")
  r<-length(roots_vector)
  HDR<-list()
  roots_vector<-sort(roots_vector)
  R<-list()
  i<-1
  left_boundary<-fhat_interpolation(roots_vector[1]-d)-p
  right_boundary<-fhat_interpolation(roots_vector[r]+d)-p
  if(left_boundary>0){
    R[[i]]<-c(min(x.sample),roots_vector[1])
    i<-i+1
  }
  if(right_boundary>0){
    R[[i]]<-c(roots_vector[r],max(x.sample))
    i<-i+1
  }
  l<-1
  while(l<length(roots_vector)){
    r1<-roots_vector[l]
```

```
    r2<-roots_vector[l+1]
    rm<-(r1+r2)/2
    f.rm<-fhat_interpolation(rm)-p
    if(f.rm>0){
      R[[i]]<-c(r1,r2)
      i<-i+1
    }
    l<-l+1
  }
  return(R)
}

true_roots<-function(p,min,max,delta){

  g_shifted<-function(xx){
    gtemp<-g(x=xx)-p
    return(gtemp)
  }
  candidates<-seq(from=min,to=max,by=delta)

  i<-1
  roots_vector<-NULL
  while(i<length(candidates)){
    x1<-candidates[i]
    x2<-candidates[i+1]
    f.x1<-g(x=x1)-p
    f.x2<-g(x=x2)-p
    if(f.x1*f.x2>0){
      m<-(x1+x2)/2
      f.m<-g(x=m)-p
      if(f.x1*f.m>0){
        i<-i+1
```

```
        #print(i)
      }else{
        r1<-myBFfzero(g_shifted,a=x1,b=m,eps=.00005)$root
        r2<-myBFfzero(g_shifted,a=m,b=x2,eps=.00005)$root
        roots_vector<-c(roots_vector,r1,r2)
        i<-i+1
        #print(i)
      }
    }else{
      r1<-myBFfzero(g_shifted,a=x1,b=x2,eps=.00005)$root
      roots_vector<-c(roots_vector,r1)
      i<-i+1
      # print(i)
    }
  }
  return(roots_vector)
}

#### FUNCTIONS RELATED TO THE ERROR COMPUTATION (\mu_q) ####

auxiliary_error_function<-function(HDR_hat,true_HDR,min=min,max=max){
  n_hdr<-length(HDR_hat)
  counter<-n_hdr
  loss_temp<-0
  integral_regions_type1<-list()

  i_temp<-NULL
  for(i in 1:n_hdr){
    if(HDR_hat[[i]][2]<=true_HDR[[1]][1]){
      integral_regions_type1<-c(integral_regions_type1,HDR_hat[i])
      loss_temp<-loss_temp+F(x=HDR_hat[[i]][2],min=min,max=max)-F(x=
          HDR_hat[[i]][1],min=min,max=max)
```

```
    i_temp<-c(i_temp,i)
    counter<-counter-1
  }
}
if(!is.null(i_temp)){HDR_hat<-HDR_hat[-c(i_temp)]}
n_hdr<-length(HDR_hat)
i_temp<-NULL
if(counter>0){
  for(i in 1:n_hdr){
    if(HDR_hat[[i]][1]>=true_HDR[[1]][2]){
      integral_regions_type1<-c(integral_regions_type1,HDR_hat[i])
      loss_temp<-loss_temp+F(x=HDR_hat[[i]][2],min=min,max=max)-F(x
          =HDR_hat[[i]][1],min=min,max=max)
      i_temp<-c(i_temp,i)
      counter<-counter-1
    }
  }
  if(!is.null(i_temp)){HDR_hat<-HDR_hat[-c(i_temp)]}
}
i_temp<-NULL
if(counter>0){
  n_hdr<-length(HDR_hat)
  if(HDR_hat[[1]][1]<true_HDR[[1]][1]){
    integral_regions_type1<-c(integral_regions_type1,list(c(HDR_hat
        [[1]][1],true_HDR[[1]][1])))
    loss_temp<-loss_temp+F(x=true_HDR[[1]][1],min=min,max=max)-F(x=
        HDR_hat[[1]][1],min=min,max=max)
    HDR_hat[[1]][1]<-true_HDR[[1]][1]
  }
  if(HDR_hat[[n_hdr]][2]>true_HDR[[1]][2]){
    integral_regions_type1<-c(integral_regions_type1,list(c(true_
        HDR[[1]][2],HDR_hat[[n_hdr]][2])))
```

```
      loss_temp<-loss_temp+F(x=HDR_hat[[n_hdr]][2],min=min,max=max)-F
          (x=true_HDR[[1]][2],min=min,max=max)
      HDR_hat[[n_hdr]][2]<-true_HDR[[1]][2]
    }
    loss_temp<-loss_temp+F(x=true_HDR[[1]][2],min=min,max=max)-F(x=
        true_HDR[[1]][1],min=min,max=max)
    for(i in 1:n_hdr){
      loss_temp<-loss_temp-F(x=HDR_hat[[i]][2],min=min,max=max)+F(x=
          HDR_hat[[i]][1],min=min,max=max)
    }
  }
  return(loss_temp)
}

error_function<-function(HDR,HDR_estimate,min,max){
  R<-length(HDR)
  Rhat<-length(HDR_estimate)
  counter<-Rhat
  error<-0
  r<-1
  temp.hat<-NULL
  while(r<=R){
    indicator1<-c(1:Rhat)[unlist(lapply(HDR_estimate,function(x)x[1]<
        HDR[[r]][2] & x[2]<HDR[[r]][2]))]
    indicator2<-c(1:Rhat)[unlist(lapply(HDR_estimate,function(x)x[1]<
        HDR[[r]][2] & x[2]>=HDR[[r]][2]))]
    if(length(indicator1)==0 & length(indicator2)==0){
      error<-error+F(HDR[[r]][2],min=min,max=max)-F(HDR[[r]][1],min=
          min,max=max)
      r<-r+1
    }else{
      if(length(indicator1)>0){
```

```
        temp.hat<-HDR_estimate[(indicator1)]
        HDR_estimate<-HDR_estimate[-indicator1]
        Rhat<-length(HDR_estimate)
        counter<-counter-length(indicator1)
      }
      indicator2<-c(1:Rhat)[unlist(lapply(HDR_estimate,function(x)x[1
          ]<HDR[[r]][2] & x[2]>=HDR[[r]][2]))]
      if(length(indicator2)>0){
        last_interval<-HDR_estimate[indicator2]
        last_interval[[1]][2]<-HDR[[r]][2]
        temp.hat<-c(temp.hat,last_interval)
        HDR_estimate[[indicator2]][1]<-HDR[[r]][2]
        #counter<-counter-length(indicator2)
      }
      temp.error<-auxiliary_error_function(HDR_hat = temp.hat,true_
          HDR = HDR[r],min=min,max=max)
      error<-error+temp.error
      r<-r+1
      temp.hat<-NULL
    }
  }
  if(counter>0){
    sum_error<-sum(unlist(lapply(HDR_estimate[(Rhat-counter+1):Rhat],
        function(x)F(x[2],min=min,max=max)-F(x[1],min=min,max=max))))
    error<-error+sum_error
  }
  return(error)
}

#######         ASYMPTOTIC RISK         ########
AR.hat<-function(c,xj,p,nd,varY,f_h0_bin,df_h0_bin,grid,counts,h1_
    binning_g,h2_binning_g){
```

```
#we assume that q(x)=f(x)
#We assume that the kernel K is the standard normal distribution.

sigmaK <- 1
v_20   <- 1/(2*sqrt(pi))

fhat                  <-Kernel_density_binning(x=xj, n=nd, h=h0_bin,
    grid=grid,counts=counts)
dfhat                 <-dfhat_binning(x=xj,grid=grid,counts=counts,n
   =nd,h=df_h0_bin)
dghat                 <-locpoly(x=x.sample,y=Y.sample,drv=1,degree=2
   ,bandwidth=h1_binning_g)
d2ghat                <-locpoly(x=x.sample,y=Y.sample,drv=2,degree=3
   ,bandwidth=h2_binning_g)

dghat_interpolation   <-  splinefun(x=dghat$x,y=dghat$y,method="fmm
   ")
d2ghat_interpolation  <-  splinefun(x=d2ghat$x,y=d2ghat$y,method="
   fmm")
dghat_xj              <-  dghat_interpolation(xj)
d2ghat_xj             <-  d2ghat_interpolation(xj)

A_xj        <-    -(dghat_xj*fhat)/((varY*fhat*v_20)^(1/2))
B2j.hat     <-    -sigmaK*(.5*fhat*d2ghat_xj+dghat_xj*dfhat)/((varY
   *fhat*v_20)^(1/2))
B1j.hat     <-    1/abs(A_xj)

nT_vector   <-    fhat*B1j.hat*((2*dnorm(x=c^(5/2)*B2j.hat)/c^(1/2)
   )+c^(2)*B2j.hat*(2*pnorm(q=c^(5/2)*B2j.hat)-1))
T_tilde     <-    nd^(-2/5)*sum(nT_vector)

return(T_tilde)
```

```
}
```

Code related to the simulations from example one

```
#########      TRUE FUNCTIONS      ############


g<-function(x){
  gg<-(2/3)*dnorm(x,mean=0,sd=1)+(1/3)*dnorm(x,mean=0,sd=1/10)
  return(gg)
}

dg_function <- function(x){

  sigma2 <- .10
  result <- (-1/sqrt(2*pi))*x*(2/3*exp(-.5*x^2)+(1/3)*(1/sigma2^3)*
     exp(-1/(2*sigma2^2)*x^2))
  return(result)
}

d2g_function <- function(x){
  sigma2 <- .10
  result <- (-1/sqrt(2*pi))*(2/3*exp(-.5*x^2)*(1-x^2)+(1/(3*sigma2^3)
     )*exp(-1/(2*sigma2^2)*x^2)*(1-1/(sigma2^2)*x^2))
  return(result)
}
F<-function(x,min,max){
  temp<-(x-min)/(max-min)
  return(temp)
}

#######        SIMULATIONS      #######
```

```
set.seed(635017)
n          <-  1000
M          <-  250
p          <-  c(g(1.036055),g(.3193359),g(.06738281))
varY       <- .1
minimo     <-  -4
maximo     <-  4
true_hdr   <- function(pp){
  roots      <-true_roots(p=pp,min=minimo,max=maximo,delta=8/1000)
  hroots     <-length(roots)/2
  true_HDR   <-list()
  for(i in 1:hroots){

    true_HDR<-c(true_HDR,list(c(roots[2*i-1],roots[2*i])))
  }
  return(true_HDR)
}
true_HDR   <- lapply(p,true_hdr)
g(unlist(true_HDR))



error_ED1_CV         <- matrix(,nrow=M,ncol=length(p))
error_LP1            <- matrix(,nrow=M,ncol=length(p))
error_LSCV1          <- matrix(,nrow=M,ncol=length(p))

h_LSCV_vec1          <-  rep(0,M)
h_LP_vec1            <-  rep(0,M)
h_ED_vec1_CV         <-  matrix(,nrow=M,ncol=length(p))
h_initial_vec1       <-  matrix(,nrow=M,ncol=length(p))
initial_roots        <-  list()
```

```
Y.matrix                  <-  matrix(,nrow=M,ncol=n)
X.matrix                  <-  matrix(,nrow=M,ncol=n)

for(m in 1:M){
  x   <- X.sample(n=n,min=minimo,max=maximo)
  g.x <- g(x)
  y   <- g.x+rnorm(n,mean=0,sd=sqrt(varY))

  X.matrix[m,]  <- x
  Y.matrix[m,]  <- y
}

m<-1
while(m<=M){

  n            <-  1000
  x.sample     <-  X.matrix[m,]
  Y.sample     <-  Y.matrix[m,]

  grid_points      <-  x.sample
  grid_cl          <-  rep(1,n)
  grid_dlY         <-  Y.sample

  ndiv         <-  1000
  d            <-  (max(x.sample)-min(x.sample))/ndiv

  #bandwidth for fhat
  s.hat         <-  sigma.hat(x.sample)
  h6_book_bin   <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
  h4_book_bin   <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,grid=
     grid_points,counts = grid_cl,n=n)*n))^(1/7)
  h0_bin        <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,grid=
```

```
    grid_points,counts = grid_cl,n=n)*n))^(1/5)

#bandwidth for dfhat
h8_book_binning          <-    (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
    ^(1/11)
h6_book_deriv_binning    <-    (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_
    book_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/9)
h1_binning               <-    (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_
    book_deriv_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/7)

h_LP                    <- dpill(x=x.sample,y=Y.sample)

h_LP_vec1[m]    <- h_LP
ghat_LP         <- locpoly(x=x.sample,y=Y.sample,bandwidth = h_LP)

h_CV            <- npregbw(formula=Y.sample~x.sample,bwmethod="cv.ls
    ")$bw
h_LSCV_vec1[m] <- h_CV

Y_bar          <- sum(Y.sample)/n
varY_hat       <- sum((Y.sample-Y_bar)^2)/(n-1)
for(l in 1:length(p)){
  ####################
  #ROOTS CV
  roots_vec_cv<-grid_search(h=h_CV,p=p[l],xsample=x.sample,delta=d,
     grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  if(!is.null(roots_vec_cv)){
    HDR_CV<-myHDR(roots_vector = roots_vec_cv,d=d,p=p[l],h_estimate
        = h_CV,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    HDR_CV<-HDR_CV[order(sapply(HDR_CV,'[[',1))]
    error_LSCV1[m,l]<-error_function(HDR_estimate = HDR_CV,HDR=true
       _HDR[[l]],min=minimo,max=maximo)
```

```
}else{error_LSCV1[m,l]<-999}

#INITIAL ROOTS LSCV
seqplot       <-  seq(from=min(x.sample),to=max(x.sample),length.
    out=2000)

h_initial    <- h_CV
ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h=h
    _initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
while(max(ghat.initial,na.rm=TRUE)<p[l]){
  h_initial    <- h_initial*(.95)
  ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h
      =h_initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
}

roots_vec_initialCV<-grid_search(h=h_initial,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
copt_hat_CV<-optimize(f=AR.hat,lower=0,upper=100,xj=roots_vec_
    initialCV,p=p[l],nd=n,varY=varY_hat,f_h0_bin=h0_bin,df_h0_bin=
    h1_binning,grid=grid_points,counts=grid_cl,h1_binning_g=h_LP,h
    2_binning_g=h_LP)$minimum
h_ED_withLP_CV<-copt_hat_CV*(n^(-1/5))
h_ED_vec1_CV[m,l]<-h_ED_withLP_CV

roots_vec_ED_CV<-grid_search(h=h_ED_withLP_CV,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

if(!is.null(roots_vec_ED_CV)){
  HDR_ED_CV<-myHDR(roots_vector = roots_vec_ED_CV,d=d,p=p[l],h_
      estimate = h_ED_withLP_CV,grid=grid_points,cl=grid_cl,dlY=
      grid_dlY)
  HDR_ED_CV<-HDR_ED_CV[order(sapply(HDR_ED_CV,'[[',1))]
```

```
      error_ED1_CV[m,l]<-error_function(HDR_estimate =HDR_ED_CV,HDR=
          true_HDR[[l]],min=minimo,max=maximo)
    } else {
      error_ED1_CV[m,l]<-999
    }

    #ROOTS LP
    roots_vec_LP<-grid_searchLP(ghatx=ghat_LP$x,ghatLP=ghat_LP$y,p=p[
        l],xsample=x.sample,delta=d)

    if(!is.null(roots_vec_LP)){
      HDR_LP<-myHDRLP(roots_vector=roots_vec_LP,d=d,p=p[l],ghatx=ghat
          _LP$x, ghatLP=ghat_LP$y,x.sample=x.sample)
      HDR_LP<-HDR_LP[order(sapply(HDR_LP,'[[',1))]
      error_LP1[m,l]<-error_function(HDR_estimate =HDR_LP,HDR=true_
          HDR[[l]],min=minimo,max=maximo)
    } else {
      error_LP1[m,l]<-999
    }
  }

  print(m)
  m<-m+1

}


set.seed(635017)
n          <-  100000
M          <-  250
p          <-  c(g(1.036055),g(.3193359),g(.06738281))
varY       <- .1
```

```
minimo    <-  -4
maximo    <-  4
true_hdr  <- function(pp){
  roots     <-true_roots(p=pp,min=minimo,max=maximo,delta=8/1000)
  hroots    <-length(roots)/2
  true_HDR  <-list()
  for(i in 1:hroots){

    true_HDR<-c(true_HDR,list(c(roots[2*i-1],roots[2*i])))
  }
  return(true_HDR)
}
true_HDR  <- lapply(p,true_hdr)
g(unlist(true_HDR))


error_ED1_CV        <- matrix(,nrow=M,ncol=length(p))
error_LP1           <- matrix(,nrow=M,ncol=length(p))
error_LSCV1         <- matrix(,nrow=M,ncol=length(p))

h_LSCV_vec1         <-  rep(0,M)
h_LP_vec1           <-  rep(0,M)
h_ED_vec1_CV        <-  matrix(,nrow=M,ncol=length(p))
h_initial_vec1      <-  matrix(,nrow=M,ncol=length(p))
initial_roots       <-  list()

Y.matrix                <-  matrix(,nrow=M,ncol=n)
X.matrix                <-  matrix(,nrow=M,ncol=n)

for(m in 1:M){
  x   <- X.sample(n=n,min=minimo,max=maximo)
  g.x <- g(x)
```

```
  y    <- g.x+rnorm(n,mean=0,sd=sqrt(varY))

  X.matrix[m,]  <- x
  Y.matrix[m,]  <- y
}

m<-1
while(m<=M){

  n            <-  100000
  x.sample     <-  X.matrix[m,]
  Y.sample     <-  Y.matrix[m,]

  plot(x.sample,Y.sample,type="p",main="␣n=100,000")
  nbins            <-  1000

  binning_object  <-  linear_binning_fast(xsample=x.sample,Y.sample=Y
     .sample,nbins=nbins)

  grid_points      <-  binning_object$grid_points
  grid_cl          <-  binning_object$cl
  grid_dlY         <-  binning_object$dl
  grid_dlY2        <-  binning_object$dl_2

  ndiv         <-  1000
  d            <-  (max(x.sample)-min(x.sample))/ndiv


  #bandwidth for fhat
  s.hat         <-  sigma.hat(x.sample)
  h6_book_bin   <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
  h4_book_bin   <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,grid=
```

```
    grid_points,counts = grid_cl,n=n)*n))^(1/7)
h0_bin        <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,grid=
    grid_points,counts = grid_cl,n=n)*n))^(1/5)

#bandwidth for dfhat
h8_book_binning          <-   (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
    ^(1/11)
h6_book_deriv_binning    <-   (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_
    book_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/9)
h1_binning               <-   (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_
    book_deriv_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/7)


h_LP                    <- dpill(x=x.sample,y=Y.sample)


h_LP_vec1[m]    <- h_LP
ghat_LP         <- locpoly(x=x.sample,y=Y.sample,bandwidth = h_LP)

h_CV            <- optimize(my_CV,grid=grid_points,counts=grid_cl,dY
    =grid_dlY,dY_2=grid_dlY2,x.sample=x.sample,Y.sample=Y.sample,
    lower=.00001,upper=1)$minimum
h_LSCV_vec1[m] <- h_CV

Y_bar          <- sum(Y.sample)/n
varY_hat       <- sum((Y.sample-Y_bar)^2)/(n-1)
for(l in 1:length(p)){
  ####################

  #ROOTS CV
  roots_vec_cv<-grid_search(h=h_CV,p=p[l],xsample=x.sample,delta=d,
     grid=grid_points,cl=grid_cl,dlY=grid_dlY)
```

```
if(!is.null(roots_vec_cv)){
  HDR_CV<-myHDR(roots_vector = roots_vec_cv,d=d,p=p[l],h_estimate
      = h_CV,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  HDR_CV<-HDR_CV[order(sapply(HDR_CV,'[[',1))]
  error_LSCV1[m,l]<-error_function(HDR_estimate = HDR_CV,HDR=true
     _HDR[[l]],min=minimo,max=maximo)
}else{error_LSCV1[m,l]<-999}

#INITIAL ROOTS LSCV
seqplot       <-  seq(from=min(x.sample),to=max(x.sample),length.
   out=2000)

h_initial    <- h_CV
ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h=h
   _initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
while(max(ghat.initial,na.rm=TRUE)<p[l]){
  h_initial    <- h_initial*(.95)
  ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h
     =h_initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
}

roots_vec_initialCV<-grid_search(h=h_initial,p=p[l],xsample=x.
   sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
copt_hat_CV<-optimize(f=AR.hat,lower=0,upper=100,xj=roots_vec_
   initialCV,p=p[l],nd=n,varY=varY_hat,f_h0_bin=h0_bin,df_h0_bin=
   h1_binning,grid=grid_points,counts=grid_cl,h1_binning_g=h_LP,h
   2_binning_g=h_LP)$minimum
h_ED_withLP_CV<-copt_hat_CV*(n^(-1/5))
h_ED_vec1_CV[m,l]<-h_ED_withLP_CV

roots_vec_ED_CV<-grid_search(h=h_ED_withLP_CV,p=p[l],xsample=x.
   sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
```

```
    if(!is.null(roots_vec_ED_CV)){
      HDR_ED_CV<-myHDR(roots_vector = roots_vec_ED_CV,d=d,p=p[l],h_
          estimate = h_ED_withLP_CV,grid=grid_points,cl=grid_cl,dlY=
          grid_dlY)
      HDR_ED_CV<-HDR_ED_CV[order(sapply(HDR_ED_CV,'[[',1))]
      error_ED1_CV[m,l]<-error_function(HDR_estimate =HDR_ED_CV,HDR=
          true_HDR[[l]],min=minimo,max=maximo)
    } else {
      error_ED1_CV[m,l]<-999
    }

    #ROOTS LP
    roots_vec_LP<-grid_searchLP(ghatx=ghat_LP$x,ghatLP=ghat_LP$y,p=p[
        l],xsample=x.sample,delta=d)

    if(!is.null(roots_vec_LP)){
      HDR_LP<-myHDRLP(roots_vector=roots_vec_LP,d=d,p=p[l],ghatx=ghat
          _LP$x, ghatLP=ghat_LP$y,x.sample=x.sample)
      HDR_LP<-HDR_LP[order(sapply(HDR_LP,'[[',1))]
      error_LP1[m,l]<-error_function(HDR_estimate =HDR_LP,HDR=true_
          HDR[[l]],min=minimo,max=maximo)
    } else {
      error_LP1[m,l]<-999
    }
  }

  print(m)
  m<-m+1

}
```

Code related to the simulations from example two

```
################        TRUE FUNCTIONS
    ############################
#
    ##########################################################################


c0          <-.7*dnorm(8,mean=4,sd=.15)+.45*dnorm(8,mean=6,sd=.5)+.5*(8
    -8.5)^2
g           <- function(x){
  gtemp <- (.7*dnorm(x,mean=4,sd=.15)+.45*dnorm(x,mean=6,sd=.5))*(x<8
      )+(-.5*(x-8.5)^2+c0)*(x>=8)
  return(gtemp)
}
dg_function <- function(x){

  df   <-  function(x,mu,s){
    argument     <-  (x-mu)/s
    result_df    <-  (1/sqrt(2*pi*s^2))*exp(-.5*argument^2)*(-argument
       *(1/s))
    return(result_df)
  }
  result_dg    <-  (0.7*df(x,mu=4,s=.15)+0.45*df(x,mu=6,s=0.5))*(x<8)
     +((-.5*2)*(x-8.5))*(x>=8)
  return(result_dg)
}

d2g_function <- function(x){
  df2     <-   function(x,mu,s){
    argument    <-  (x-mu)/s
    result_df2 <-(1/sqrt(2*pi*s^2))*exp(-.5*argument^2)*((argument*(1
```

```
        /s))^2-(1/s^2))
    return(result_df2)
  }

  result_d2g  <-  (0.7*df2(x,mu=4,s=.15)+0.45*df2(x,mu=6,s=0.5))*(x<8
     )-1*(x>=8)
  return(result_d2g)
}

F<-function(x,min,max){
  temp<-(x-min)/(max-min)
  return(temp)
}

########      THE SIMULATIONS PART      ########
set.seed(635017)
n          <-  1000
M          <-  250
p          <-  c(.05,.10,1,1.3,1.4,1.5)
varY       <- .1
minimo     <- 2
maximo     <- 10
true_hdr   <- function(pp){
  roots      <-true_roots(p=pp,min=minimo,max=maximo,delta=.9/1000)
  hroots     <-length(roots)/2
  true_HDR   <-list()
  for(i in 1:hroots){

    true_HDR<-c(true_HDR,list(c(roots[2*i-1],roots[2*i])))
  }
  return(true_HDR)
}
```

```
true_HDR  <- lapply(p,true_hdr)
g(unlist(true_HDR))


error_ED1_CV        <- matrix(,nrow=M,ncol=length(p))
error_LP1           <- matrix(,nrow=M,ncol=length(p))
error_LSCV1         <- matrix(,nrow=M,ncol=length(p))

h_LSCV_vec1         <-  rep(0,M)
h_LP_vec1           <-  rep(0,M)
h_ED_vec1_CV        <-  matrix(,nrow=M,ncol=length(p))
h_initial_vec1      <-  matrix(,nrow=M,ncol=length(p))
initial_roots       <-  list()

Y.matrix                <-  matrix(,nrow=M,ncol=n)
X.matrix                <-  matrix(,nrow=M,ncol=n)

for(m in 1:M){
  x   <- X.sample(n=n,min=minimo,max=maximo)
  g.x <- g(x)
  y   <- g.x+rnorm(n,mean=0,sd=sqrt(varY))

  X.matrix[m,]  <- x
  Y.matrix[m,]  <- y
}

m<-1
while(m<=M){

  n           <-  1000
  x.sample    <-  X.matrix[m,]
  Y.sample    <-  Y.matrix[m,]
```

```
grid_points      <-  x.sample
grid_cl          <-  rep(1,n)
grid_dlY         <-  Y.sample

ndiv         <-  1000
d            <-  (max(x.sample)-min(x.sample))/ndiv

#bandwidth for fhat
s.hat         <-  sigma.hat(x.sample)
h6_book_bin   <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
h4_book_bin   <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,grid=
   grid_points,counts = grid_cl,n=n)*n))^(1/7)
h0_bin        <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,grid=
   grid_points,counts = grid_cl,n=n)*n))^(1/5)

#bandwidth for dfhat
h8_book_binning          <-   (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
   ^(1/11)
h6_book_deriv_binning    <-   (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_
   book_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/9)
h1_binning               <-   (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_
   book_deriv_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/7)

h_LP                   <- dpill(x=x.sample,y=Y.sample)

h_LP_vec1[m]    <- h_LP
ghat_LP         <- locpoly(x=x.sample,y=Y.sample,bandwidth = h_LP)

h_CV            <- npregbw(formula=Y.sample~x.sample,bwmethod="cv.ls
   ")$bw
h_LSCV_vec1[m] <- h_CV
```

```
Y_bar          <- sum(Y.sample)/n
varY_hat       <- sum((Y.sample-Y_bar)^2)/(n-1)
for(l in 1:length(p)){
  ####################

  #ROOTS CV
  roots_vec_cv<-grid_search(h=h_CV,p=p[l],xsample=x.sample,delta=d,
     grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  if(!is.null(roots_vec_cv)){
    HDR_CV<-myHDR(roots_vector = roots_vec_cv,d=d,p=p[l],h_estimate
        = h_CV,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    HDR_CV<-HDR_CV[order(sapply(HDR_CV,'[[',1))]
    error_LSCV1[m,l]<-error_function(HDR_estimate = HDR_CV,HDR=true
       _HDR[[l]],min=minimo,max=maximo)
  }else{error_LSCV1[m,l]<-999}

  #INITIAL ROOTS LSCV
  seqplot        <-  seq(from=min(x.sample),to=max(x.sample),length.
     out=2000)

  h_initial     <- h_CV
  ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h=h
     _initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  while(max(ghat.initial,na.rm=TRUE)<p[l]){
    h_initial     <- h_initial*(.95)
    ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h
       =h_initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  }

  roots_vec_initialCV<-grid_search(h=h_initial,p=p[l],xsample=x.
     sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
```

```
copt_hat_CV<-optimize(f=AR.hat,lower=0,upper=100,xj=roots_vec_
    initialCV,p=p[l],nd=n,varY=varY_hat,f_h0_bin=h0_bin,df_h0_bin=
    h1_binning,grid=grid_points,counts=grid_cl,h1_binning_g=h_LP,h
    2_binning_g=h_LP)$minimum
h_ED_withLP_CV<-copt_hat_CV*(n^(-1/5))
h_ED_vec1_CV[m,l]<-h_ED_withLP_CV

roots_vec_ED_CV<-grid_search(h=h_ED_withLP_CV,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

if(!is.null(roots_vec_ED_CV)){
  HDR_ED_CV<-myHDR(roots_vector = roots_vec_ED_CV,d=d,p=p[l],h_
      estimate = h_ED_withLP_CV,grid=grid_points,cl=grid_cl,dlY=
      grid_dlY)
  HDR_ED_CV<-HDR_ED_CV[order(sapply(HDR_ED_CV,'[[',1))]
  error_ED1_CV[m,l]<-error_function(HDR_estimate =HDR_ED_CV,HDR=
      true_HDR[[l]],min=minimo,max=maximo)
} else {
  error_ED1_CV[m,l]<-999
}

#ROOTS LP
roots_vec_LP<-grid_searchLP(ghatx=ghat_LP$x,ghatLP=ghat_LP$y,p=p[
    l],xsample=x.sample,delta=d)

if(!is.null(roots_vec_LP)){
  HDR_LP<-myHDRLP(roots_vector=roots_vec_LP,d=d,p=p[l],ghatx=ghat
      _LP$x, ghatLP=ghat_LP$y,x.sample=x.sample)
  HDR_LP<-HDR_LP[order(sapply(HDR_LP,'[[',1))]
  error_LP1[m,l]<-error_function(HDR_estimate =HDR_LP,HDR=true_
      HDR[[l]],min=minimo,max=maximo)
} else {
```

```
      error_LP1[m,l]<-999
    }
  }

  print(m)
  m<-m+1

}

set.seed(635017)
n          <-  100000
M          <-  250
p          <-  c(.05,.10,1,1.3,1.4,1.5)
varY       <- .1
minimo     <- 2
maximo     <- 10
true_hdr   <- function(pp){
  roots      <-true_roots(p=pp,min=minimo,max=maximo,delta=.9/1000)
  hroots     <-length(roots)/2
  true_HDR   <-list()
  for(i in 1:hroots){

    true_HDR<-c(true_HDR,list(c(roots[2*i-1],roots[2*i])))
  }
  return(true_HDR)
}
true_HDR   <- lapply(p,true_hdr)
g(unlist(true_HDR))


error_ED1_CV         <- matrix(,nrow=M,ncol=length(p))
error_LP1            <- matrix(,nrow=M,ncol=length(p))
```

```
error_LSCV1          <- matrix(,nrow=M,ncol=length(p))

h_LSCV_vec1          <-  rep(0,M)
h_LP_vec1            <-  rep(0,M)
h_ED_vec1_CV         <-  matrix(,nrow=M,ncol=length(p))
h_initial_vec1       <-  matrix(,nrow=M,ncol=length(p))
initial_roots        <-  list()

Y.matrix                 <-  matrix(,nrow=M,ncol=n)
X.matrix                 <-  matrix(,nrow=M,ncol=n)

for(m in 1:M){
  x    <- X.sample(n=n,min=minimo,max=maximo)
  g.x <- g(x)
  y    <- g.x+rnorm(n,mean=0,sd=sqrt(varY))

  X.matrix[m,]  <- x
  Y.matrix[m,]  <- y
}

m<-1
while(m<=M){

  n            <-  100000
  x.sample     <-  X.matrix[m,]
  Y.sample     <-  Y.matrix[m,]

  nbins            <-  1000

  binning_object  <-  linear_binning_fast(xsample=x.sample,Y.sample=Y
     .sample,nbins=nbins)
```

```
grid_points      <-  binning_object$grid_points
grid_cl          <-  binning_object$cl
grid_dlY         <-  binning_object$dl
grid_dlY2        <-  binning_object$dl_2

ndiv         <-  1000
d            <-  (max(x.sample)-min(x.sample))/ndiv


#bandwidth for fhat
s.hat         <-  sigma.hat(x.sample)
h6_book_bin   <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
h4_book_bin   <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,grid=
   grid_points,counts = grid_cl,n=n)*n))^(1/7)
h0_bin        <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,grid=
   grid_points,counts = grid_cl,n=n)*n))^(1/5)

#bandwidth for dfhat
h8_book_binning          <-   (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
   ^(1/11)
h6_book_deriv_binning    <-   (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_
   book_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/9)
h1_binning               <-   (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_
   book_deriv_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/7)

h_LP                   <- dpill(x=x.sample,y=Y.sample)

h_LP_vec1[m]    <- h_LP
ghat_LP         <- locpoly(x=x.sample,y=Y.sample,bandwidth = h_LP)

h_CV            <- optimize(my_CV,grid=grid_points,counts=grid_cl,dY
   =grid_dlY,dY_2=grid_dlY2,x.sample=x.sample,Y.sample=Y.sample,
```

```
    lower=.00001,upper=1)$minimum
h_LSCV_vec1[m] <- h_CV

Y_bar          <- sum(Y.sample)/n
varY_hat       <- sum((Y.sample-Y_bar)^2)/(n-1)
for(l in 1:length(p)){
  ####################

  #ROOTS CV
  roots_vec_cv<-grid_search(h=h_CV,p=p[l],xsample=x.sample,delta=d,
      grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  if(!is.null(roots_vec_cv)){
    HDR_CV<-myHDR(roots_vector = roots_vec_cv,d=d,p=p[l],h_estimate
        = h_CV,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    HDR_CV<-HDR_CV[order(sapply(HDR_CV,'[[',1))]
    error_LSCV1[m,l]<-error_function(HDR_estimate = HDR_CV,HDR=true
       _HDR[[l]],min=minimo,max=maximo)
  }else{error_LSCV1[m,l]<-999}

  #INITIAL ROOTS LSCV
  seqplot        <-  seq(from=min(x.sample),to=max(x.sample),length.
      out=2000)

  h_initial     <- h_CV
  ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h=h
      _initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  while(max(ghat.initial,na.rm=TRUE)<p[l]){
    h_initial     <- h_initial*(.95)
    ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h
       =h_initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  }
```

```
roots_vec_initialCV<-grid_search(h=h_initial,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
copt_hat_CV<-optimize(f=AR.hat,lower=0,upper=100,xj=roots_vec_
    initialCV,p=p[l],nd=n,varY=varY_hat,f_h0_bin=h0_bin,df_h0_bin=
    h1_binning,grid=grid_points,counts=grid_cl,h1_binning_g=h_LP,h
    2_binning_g=h_LP)$minimum
h_ED_withLP_CV<-copt_hat_CV*(n^(-1/5))
h_ED_vec1_CV[m,l]<-h_ED_withLP_CV

roots_vec_ED_CV<-grid_search(h=h_ED_withLP_CV,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

if(!is.null(roots_vec_ED_CV)){
  HDR_ED_CV<-myHDR(roots_vector = roots_vec_ED_CV,d=d,p=p[l],h_
      estimate = h_ED_withLP_CV,grid=grid_points,cl=grid_cl,dlY=
      grid_dlY)
  HDR_ED_CV<-HDR_ED_CV[order(sapply(HDR_ED_CV,'[[',1))]
  error_ED1_CV[m,l]<-error_function(HDR_estimate =HDR_ED_CV,HDR=
      true_HDR[[l]],min=minimo,max=maximo)
} else {
  error_ED1_CV[m,l]<-999
}

#ROOTS LP
roots_vec_LP<-grid_searchLP(ghatx=ghat_LP$x,ghatLP=ghat_LP$y,p=p[
    l],xsample=x.sample,delta=d)

if(!is.null(roots_vec_LP)){
  HDR_LP<-myHDRLP(roots_vector=roots_vec_LP,d=d,p=p[l],ghatx=ghat
      _LP$x, ghatLP=ghat_LP$y,x.sample=x.sample)
  HDR_LP<-HDR_LP[order(sapply(HDR_LP,'[[',1))]
  error_LP1[m,l]<-error_function(HDR_estimate =HDR_LP,HDR=true_
```

```
          HDR[[l]],min=minimo,max=maximo)
    } else {
      error_LP1[m,l]<-999
    }
  }

  print(m)
  m<-m+1

}
```

Code related to the simulations from example three

```
########        TRUE FUNCTIONS      ########

g<-function(x){
  d<-sqrt(x*(1-x))*sin(2.1*pi/(x+.05))
  return(d)
}

dg_function <- function(x){

  argument     <-  2*pi/(x+0.50)
  temp_result <-  .5*(x*(1-x))^(-1/2)*(1-2*x)*sin(argument)-2*pi*cos(
     argument)*((x*(1-x))^(1/2))/((x+.50)^2)
  return(temp_result)
}

d2g_function <- function(x){
  argument        <-  2*pi/(x+0.50)
  dfinterior_1    <- (-2*pi)/(x+.5)^2
  dfinterior_2    <- .5*(1-2*x)/(sqrt(x*(1-x))*((x+.5)^2))-2*sqrt(x*(1
     -x))/(x+.5)^3
```

```
  result1         <- (-1/4)*(1-2*x)^2*sin(argument)*(x*(1-x))^(-3/2)
  result2         <- (1/2)*(x*(1-x))^(-1/2)*(-2*sin(argument)+(1-2*x)*
     cos(argument)*dfinterior_1)
  result3         <- 2*pi*sin(argument)*dfinterior_1*(sqrt(x*(1-x))/(x
     +.5)^2)-2*pi*cos(argument)*dfinterior_2
  temp_result     <- result1+result2+result3
  return(temp_result)
}

F<-function(x,min,max){
  temp<-(x-min)/(max-min)
  return(temp)
}
#######   THE SIMULATIONS PART    #######

set.seed(635017)
n          <-  1000
M          <-  250
p          <-  c(.18,.25,.30,.40)
minimo     <-.1
varY       <-.1
maximo     <-1
true_hdr   <- function(pp){
  roots      <-true_roots(p=pp,min=minimo,max=maximo,delta=.9/1000)
  hroots     <-length(roots)/2
  true_HDR   <-list()
  for(i in 1:hroots){

    true_HDR<-c(true_HDR,list(c(roots[2*i-1],roots[2*i])))
  }
  return(true_HDR)
}
```

```
true_HDR  <- lapply(p,true_hdr)

g(unlist(true_HDR))


error_ED1_CV        <- matrix(,nrow=M,ncol=length(p))
error_LP1           <- matrix(,nrow=M,ncol=length(p))
error_LSCV1         <- matrix(,nrow=M,ncol=length(p))

h_LSCV_vec1         <-  rep(0,M)
h_LP_vec1           <-  rep(0,M)
h_ED_vec1_CV        <-  matrix(,nrow=M,ncol=length(p))
h_initial_vec1      <-  matrix(,nrow=M,ncol=length(p))
initial_roots       <-  list()

Y.matrix                <-  matrix(,nrow=M,ncol=n)
X.matrix                <-  matrix(,nrow=M,ncol=n)

for(m in 1:M){
  x   <- X.sample(n=n,min=minimo,max=maximo)
  g.x <- g(x)
  y   <- g.x+rnorm(n,mean=0,sd=sqrt(varY))

  X.matrix[m,]  <- x
  Y.matrix[m,]  <- y
}

m<-1
while(m<=M){

  n           <-  1000
  x.sample    <-  X.matrix[m,]
```

```
Y.sample     <-  Y.matrix[m,]

grid_points      <-  x.sample
grid_cl          <-  rep(1,n)
grid_dlY         <-  Y.sample

ndiv         <-  1000
d            <-  (max(x.sample)-min(x.sample))/ndiv

#bandwidth for fhat
s.hat         <-  sigma.hat(x.sample)
h6_book_bin   <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
h4_book_bin   <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,grid=
    grid_points,counts = grid_cl,n=n)*n))^(1/7)
h0_bin        <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,grid=
    grid_points,counts = grid_cl,n=n)*n))^(1/5)

#bandwidth for dfhat
h8_book_binning          <-   (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
    ^(1/11)
h6_book_deriv_binning    <-   (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_
    book_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/9)
h1_binning               <-   (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_
    book_deriv_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/7)

h_LP                   <- dpill(x=x.sample,y=Y.sample)

h_LP_vec1[m]    <- h_LP
ghat_LP         <- locpoly(x=x.sample,y=Y.sample,bandwidth = h_LP)

h_CV            <- npregbw(formula=Y.sample~x.sample,bwmethod="cv.ls
    ")$bw
```

```
h_LSCV_vec1[m] <- h_CV

Y_bar          <- sum(Y.sample)/n
varY_hat       <- sum((Y.sample-Y_bar)^2)/(n-1)
for(l in 1:length(p)){
  ####################
  #ROOTS CV
  roots_vec_cv<-grid_search(h=h_CV,p=p[l],xsample=x.sample,delta=d,
     grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  if(!is.null(roots_vec_cv)){
    HDR_CV<-myHDR(roots_vector = roots_vec_cv,d=d,p=p[l],h_estimate
        = h_CV,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    HDR_CV<-HDR_CV[order(sapply(HDR_CV,'[[',1))]
    error_LSCV1[m,l]<-error_function(HDR_estimate = HDR_CV,HDR=true
       _HDR[[l]],min=minimo,max=maximo)
  }else{error_LSCV1[m,l]<-999}

  #INITIAL ROOTS LSCV
  seqplot        <-  seq(from=min(x.sample),to=max(x.sample),length.
     out=2000)

  h_initial     <- h_CV
  ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h=h
     _initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  while(max(ghat.initial,na.rm=TRUE)<p[l]){
    h_initial     <- h_initial*(.95)
    ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h
       =h_initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  }

  roots_vec_initialCV<-grid_search(h=h_initial,p=p[l],xsample=x.
     sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
```

```
copt_hat_CV<-optimize(f=AR.hat,lower=0,upper=100,xj=roots_vec_
    initialCV,p=p[l],nd=n,varY=varY_hat,f_h0_bin=h0_bin,df_h0_bin=
    h1_binning,grid=grid_points,counts=grid_cl,h1_binning_g=h_LP,h
    2_binning_g=h_LP)$minimum
h_ED_withLP_CV<-copt_hat_CV*(n^(-1/5))
h_ED_vec1_CV[m,l]<-h_ED_withLP_CV

roots_vec_ED_CV<-grid_search(h=h_ED_withLP_CV,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

if(!is.null(roots_vec_ED_CV)){
  HDR_ED_CV<-myHDR(roots_vector = roots_vec_ED_CV,d=d,p=p[l],h_
      estimate = h_ED_withLP_CV,grid=grid_points,cl=grid_cl,dlY=
      grid_dlY)
  HDR_ED_CV<-HDR_ED_CV[order(sapply(HDR_ED_CV,'[[',1))]
  error_ED1_CV[m,l]<-error_function(HDR_estimate =HDR_ED_CV,HDR=
      true_HDR[[l]],min=minimo,max=maximo)
} else {
  error_ED1_CV[m,l]<-999
}

#ROOTS LP
roots_vec_LP<-grid_searchLP(ghatx=ghat_LP$x,ghatLP=ghat_LP$y,p=p[
    l],xsample=x.sample,delta=d)

if(!is.null(roots_vec_LP)){
  HDR_LP<-myHDRLP(roots_vector=roots_vec_LP,d=d,p=p[l],ghatx=ghat
      _LP$x, ghatLP=ghat_LP$y,x.sample=x.sample)
  HDR_LP<-HDR_LP[order(sapply(HDR_LP,'[[',1))]
  error_LP1[m,l]<-error_function(HDR_estimate =HDR_LP,HDR=true_
      HDR[[l]],min=minimo,max=maximo)
} else {
```

```
      error_LP1[m,l]<-999
    }
  }

  print(m)
  m<-m+1

}

set.seed(635017)
n          <-  100000
M          <-  250
p          <-  c(.18,.25,.30,.40)
minimo     <-.1
varY       <-.1
maximo     <-1
true_hdr   <- function(pp){
  roots      <-true_roots(p=pp,min=minimo,max=maximo,delta=.9/1000)
  hroots     <-length(roots)/2
  true_HDR   <-list()
  for(i in 1:hroots){

    true_HDR<-c(true_HDR,list(c(roots[2*i-1],roots[2*i])))
  }
  return(true_HDR)
}
true_HDR   <- lapply(p,true_hdr)

g(unlist(true_HDR))


error_ED1_CV         <- matrix(,nrow=M,ncol=length(p))
```

```
error_LP1            <- matrix(,nrow=M,ncol=length(p))
error_LSCV1          <- matrix(,nrow=M,ncol=length(p))

h_LSCV_vec1          <-  rep(0,M)
h_LP_vec1            <-  rep(0,M)
h_ED_vec1_CV         <-  matrix(,nrow=M,ncol=length(p))
h_initial_vec1       <-  matrix(,nrow=M,ncol=length(p))
initial_roots        <-  list()

Y.matrix                 <-  matrix(,nrow=M,ncol=n)
X.matrix                 <-  matrix(,nrow=M,ncol=n)

for(m in 1:M){
  x    <- X.sample(n=n,min=minimo,max=maximo)
  g.x <- g(x)
  y    <- g.x+rnorm(n,mean=0,sd=sqrt(varY))

  X.matrix[m,]  <- x
  Y.matrix[m,]  <- y
}

m<-1
while(m<=M){

  n            <-  100000
  x.sample     <-  X.matrix[m,]
  Y.sample     <-  Y.matrix[m,]

  nbins            <-  3000

  binning_object  <-  linear_binning_fast(xsample=x.sample,Y.sample=Y
     .sample,nbins=nbins)
```

```
grid_points      <-  binning_object$grid_points
grid_cl          <-  binning_object$cl
grid_dlY         <-  binning_object$dl
grid_dlY2        <-  binning_object$dl_2

ndiv         <-  1000
d            <-  (max(x.sample)-min(x.sample))/ndiv

#bandwidth for fhat
s.hat         <-  sigma.hat(x.sample)
h6_book_bin   <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
h4_book_bin   <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,grid=
   grid_points,counts = grid_cl,n=n)*n))^(1/7)
h0_bin        <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,grid=
   grid_points,counts = grid_cl,n=n)*n))^(1/5)

#bandwidth for dfhat
h8_book_binning          <-   (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
   ^(1/11)
h6_book_deriv_binning    <-   (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_
   book_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/9)
h1_binning               <-   (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_
   book_deriv_binning,grid=grid_points,counts=grid_cl,n=n)))^(1/7)

h_LP                   <- dpill(x=x.sample,y=Y.sample)

h_LP_vec1[m]    <- h_LP
ghat_LP         <- locpoly(x=x.sample,y=Y.sample,bandwidth = h_LP)

h_CV            <- optimize(my_CV,grid=grid_points,counts=grid_cl,dY
   =grid_dlY,dY_2=grid_dlY2,x.sample=x.sample,Y.sample=Y.sample,
```

```
    lower=.00001,upper=1)$minimum
h_LSCV_vec1[m] <- h_CV

Y_bar          <- sum(Y.sample)/n
varY_hat       <- sum((Y.sample-Y_bar)^2)/(n-1)
for(l in 1:length(p)){
  ####################
  #ROOTS CV
  roots_vec_cv<-grid_search(h=h_CV,p=p[l],xsample=x.sample,delta=d,
     grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  if(!is.null(roots_vec_cv)){
    HDR_CV<-myHDR(roots_vector = roots_vec_cv,d=d,p=p[l],h_estimate
        = h_CV,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    HDR_CV<-HDR_CV[order(sapply(HDR_CV,'[[',1))]
    error_LSCV1[m,l]<-error_function(HDR_estimate = HDR_CV,HDR=true
       _HDR[[l]],min=minimo,max=maximo)
  }else{error_LSCV1[m,l]<-999}

  #INITIAL ROOTS LSCV
  seqplot        <-  seq(from=min(x.sample),to=max(x.sample),length.
     out=2000)

  h_initial     <- h_CV
  ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h=h
     _initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  while(max(ghat.initial,na.rm=TRUE)<p[l]){
    h_initial     <- h_initial*(.95)
    ghat.initial <- NWkernel_regression_gaussian_binned(x=seqplot,h
       =h_initial,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  }

  roots_vec_initialCV<-grid_search(h=h_initial,p=p[l],xsample=x.
```

```
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
copt_hat_CV<-optimize(f=AR.hat,lower=0,upper=100,xj=roots_vec_
    initialCV,p=p[l],nd=n,varY=varY_hat,f_h0_bin=h0_bin,df_h0_bin=
    h1_binning,grid=grid_points,counts=grid_cl,h1_binning_g=h_LP,h
    2_binning_g=h_LP)$minimum
h_ED_withLP_CV<-copt_hat_CV*(n^(-1/5))
h_ED_vec1_CV[m,l]<-h_ED_withLP_CV

roots_vec_ED_CV<-grid_search(h=h_ED_withLP_CV,p=p[l],xsample=x.
    sample,delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

if(!is.null(roots_vec_ED_CV)){
  HDR_ED_CV<-myHDR(roots_vector = roots_vec_ED_CV,d=d,p=p[l],h_
      estimate = h_ED_withLP_CV,grid=grid_points,cl=grid_cl,dlY=
      grid_dlY)
  HDR_ED_CV<-HDR_ED_CV[order(sapply(HDR_ED_CV,'[[',1))]
  error_ED1_CV[m,l]<-error_function(HDR_estimate =HDR_ED_CV,HDR=
      true_HDR[[l]],min=minimo,max=maximo)
} else {
  error_ED1_CV[m,l]<-999
}

#ROOTS LP
roots_vec_LP<-grid_searchLP(ghatx=ghat_LP$x,ghatLP=ghat_LP$y,p=p[
    l],xsample=x.sample,delta=d)

if(!is.null(roots_vec_LP)){
  HDR_LP<-myHDRLP(roots_vector=roots_vec_LP,d=d,p=p[l],ghatx=ghat
      _LP$x, ghatLP=ghat_LP$y,x.sample=x.sample)
  HDR_LP<-HDR_LP[order(sapply(HDR_LP,'[[',1))]
  error_LP1[m,l]<-error_function(HDR_estimate =HDR_LP,HDR=true_
      HDR[[l]],min=minimo,max=maximo)
```

```
    } else {
      error_LP1[m,l]<-999
    }
  }

  print(m)
  m<-m+1

}
```

B.2 Example code for DCS study

```
library(hdrcde)
library(np)
library(rootSolve)



data    <- read.csv("data.csv", header=TRUE)
sheep <- data[, 1:8]
sheep   <- sheep[, c(1,2,4)]

sheep[1:10,]

Xmatrix             <- sheep[,c(1,2)]
colnames(Xmatrix)   <- c("pressure","time")
head(Xmatrix)
Yvector             <- sheep[,3]
PCA_sheep           <- prcomp(Xmatrix,scale=TRUE)
str(PCA_sheep)

PCA_sheep$center  #mean of variables
PCA_sheep$scale   #s.d. of variables
```

```
PCA_sheep$rotation  <-  -PCA_sheep$rotation
PCA_sheep$x         <-  -PCA_sheep$x         #pca scores
PCA_sheep$sdev    #s.d. of each principal component

PCA_data              <-   cbind(PCA_sheep$x[,1],Yvector)
colnames(PCA_data)    <-  c("PCA","Y")
head(PCA_data)
PCA_data              <-  as.data.frame(PCA_data)


########     FUNCTION TO OBTAIN h_LP         ########

hLP_function      <-   function(x.sample,Y.sample,h0_bin,grid_points,
grid_cl,grid_dlY){
  glm_polynomial   <- glm(Y.sample ~ x.sample + I(x.sample^2)
  + I(x.sample^3),family=binomial)

  coefficients_polyn  <- coef(glm_polynomial)
  first_integrand     <- function(x){

  matrix_powers  <-  sapply(x,function(i)c(i^0,i^1,i^2,i^3))
  inv_variances  <-  apply(matrix_powers,MARGIN=2,
  FUN=function(x_polynomial)(1+exp(-coefficients_polyn%*%x_polynomial
     ))^2/
  exp(-coefficients_polyn%*%x_polynomial))
  fhat           <-   Kernel_density_binning(x=x,n=length(x.sample),
  h=h0_bin,grid=grid_points,counts=grid_cl)
    argument       <-  inv_variances*fhat
    return(argument)
  }

  integral_numerator  <-  integrate(first_integrand,lower=min(x.
```

```
    sample),
  upper=max(x.sample),subdivisions = 200)$value
  integral_numerator  <-  integral_numerator*(1/(2*sqrt(pi)))

  second_integrand    <-  function(x){
  d2_eta              <- 2*coefficients_polyn[3]+6*coefficients_polyn
    [4]*x
  fhat                <- Kernel_density_binning(x=x,n=length(x.sample
    ),
  h=h0_bin,grid=grid_points,counts=grid_cl)
  fhat2               <- fhat^2
  d2_eta2             <- d2_eta^2
  argument2           <- d2_eta2*fhat2
  return(argument2)
  }

  integral_denominator  <-  integrate(second_integrand,lower=min(x.
    sample),
  upper=max(x.sample),subdivisions=200)$value

  c_AMISE_LP      <-  (integral_numerator/integral_denominator)^(1/5)
  h_AMISE_LP      <-   c_AMISE_LP*(length(x.sample)^(-1/5))
  return(h_AMISE_LP)

}
#######      FUNCTIONS TO OBTAIN g'_hat g''_hat       #######

g_derivatives_function       <-    function(x,h0_LP,x.sample,Y.sample
   ){
  function_one   <-  function(beta,x){
    b0           <-  beta[1]
    b1           <-  beta[2]
```

```
  p_function    <-  1/(1+exp(-b0-b1*(x.sample-x)))
  argument      <-  (Y.sample*log(p_function)+(1-Y.sample)*log(1-p_
     function))
  *(1/h0_LP)*unlist(lapply((x.sample-x)/h0_LP,dnorm))
  argument      <-  sum(argument)
  return(argument)
}
eta_hat<-optim(c(0.5,0.5),function_one,x=x,control=list(fnscale=-1)
   )$par[1]

function_two    <-  function(beta){
  b0            <-  beta[1]
  b1            <-  beta[2]
  b2            <-  beta[3]
  p_function    <-  1/(1+exp(-b0-b1*(x.sample-x)-b2*(x.sample-x)^2))
  argument      <-  (Y.sample*log(p_function)+(1-Y.sample)*log(1-p_
     function))
  *(1/h0_LP)*unlist(lapply((x.sample-x)/h0_LP,dnorm))
  argument      <-  sum(argument)
  return(argument)
}
deriv_eta_hat<-optim(c(0.5,0.5,0.5),function_two,
                     control=list(fnscale=-1))$par[2]

function_three    <-  function(beta){
  b0            <-  beta[1]
  b1            <-  beta[2]
  b2            <-  beta[3]
  b3            <-  beta[4]
  p_function    <-  1/(1+exp(-b0-b1*(x.sample-x)-b2*(x.sample-x)^2
  -b3*(x.sample-x)^3))
  argument      <-  (Y.sample*log(p_function)+(1-Y.sample)
```

```
    *log(1-p_function))*(1/h0_LP)*unlist(lapply((x.sample-x)/h0_LP,
       dnorm))
    argument      <-  sum(argument)
    return(argument)
  }
  deriv2_eta_hat<-  optim(c(0,0,0,0),function_three,
                   control=list(fnscale=-1))$par[3]*2

  deriv_g_hat  <- (exp(-eta_hat)/((1+exp(-eta_hat))^2))*deriv_eta_hat
  deriv2_g_hat <- (exp(-eta_hat)/((1+exp(-eta_hat))^2))
          *((1-2/(1+exp(-eta_hat)))*(deriv_eta_hat^2)+deriv2_eta_hat)

  return(derivatives=list(d1_g=deriv_g_hat,d2_g=deriv2_g_hat))
}


######   ASYMPTOTIC RISK   ######
AR.hat<-function(c,xj,p,nd,varY,h0_bin,dfh0_binning,h0_LP,
x.sample,Y.sample,grid_points,grid_cl){
  #we assume that q(x)=f(x)
  #We assume that the kernel K is the standard normal distribution.

  sigmaK <- 1
  v_20   <- 1/(2*sqrt(pi))

  fhat    <- Kernel_density_binning(x=xj,n=nd,h=h0_bin,
  grid=grid_points,counts=grid_cl)
  dfhat   <- dfhat_binning(x=xj,grid=grid_points,
  counts=grid_cl,n=nd,h=dfh0_binning)

 derivatives_g <- unlist(lapply(xj,function(xx)
 g_derivatives_function(x=xx,h0_LP=h0_LP,x.sample=x.sample,Y.sample=Y
```

```
    .sample)))
  index             <- length(derivatives_g)/2
  dghat_xj          <- derivatives_g[2*c(1:index)-1]
  d2ghat_xj         <- derivatives_g[2*c(1:index)]

 A_xj <- -(dghat_xj*fhat)/((varY*fhat*v_20)^(1/2))
 B2j.hat<- -sigmaK*(.5*fhat*d2ghat_xj+dghat_xj*dfhat)/((varY*fhat*v_2
    0)^(1/2))
 B1j.hat <- 1/abs(A_xj)

 nT_vector <-    fhat*B1j.hat*((2*dnorm(x=c^(5/2)*B2j.hat)/c^(1/2))
 +c^(2)*B2j.hat*(2*pnorm(q=c^(5/2)*B2j.hat)-1))
  T_tilde      <-    nd^(-2/5)*sum(nT_vector)

  return(T_tilde)
}
########           ANALYSIS OF DATA           ########
LS_list      <-  list()
p            <-  c(0.20,0.50,0.80)
hopt_vector  <-  rep(0,length(p))
n            <-  nrow(PCA_data)
x.sample     <-  PCA_data$PCA
Y.sample     <-  PCA_data$Y

plot(x.sample,Y.sample,type="p",main="DCS␣in␣sheep",xlab="PCA␣score",
   ylab="mortality")

grid_points      <-  x.sample
grid_cl          <-  rep(1,n)
grid_dlY         <-  Y.sample

ndiv         <-  1000
```

```
d           <-  (max(x.sample)-min(x.sample))/ndiv

s.hat        <-  sigma.hat(x.sample)
h6_book_bin  <-  (6*15/(sqrt(2*pi)*psi_NS_8(s.hat)*n))^(1/9)
h4_book_bin  <-  (-6/(sqrt(2*pi)*psi6_binning(h_6=h6_book_bin,
grid=grid_points,counts = grid_cl,n=n)*n))^(1/7)
h0_bin       <-  (1/(2*sqrt(pi)*psi4_binning(h_4=h4_book_bin,
grid=grid_points,counts = grid_cl,n=n)*n))^(1/5)
#bandwidth for dfhat
 h8_book_binning          <-   (-210/(psi_NS_10(s.hat)*sqrt(2*pi)*n))
    ^(1/11)
 h6_book_deriv_binning <- (30/(sqrt(2*pi)*n*psi8_binning(h_8=h8_book_
    binning,
grid=grid_points,counts=grid_cl,n=n)))^(1/9)
 h1_binning  <- (-3/(4*sqrt(pi)*n*psi6_binning(h_6=h6_book_deriv_
    binning,
grid=grid_points,counts=grid_cl,n=n)))^(1/7)

 data            <- as.data.frame(cbind(x.sample,Y.sample))

 h_CV     <- npregbw(formula=data$Y.sample~data$x.sample,bwmethod="cv
    .aic")$bw
 h_LP     <-  hLP_function(x.sample=x.sample,Y.sample=Y.sample
 ,h0_bin=h0_bin,grid_points = grid_points,grid_cl = grid_cl,
 grid_dlY = grid_dlY)



 #########     first level   ###############

  l<-1
  #INITIAL ROOTS LSCV
```

```
  seqplot       <-  seq(from=min(x.sample),to=max(x.sample),length.
     out=2000)
  h_initial1    <-  h_CV
  ghat.initial  <- NWkernel_regression_gaussian_binned(x=seqplot,h=h_
     initial1,
  grid=grid_points,cl=grid_cl,dlY=grid_dlY)

  roots_vec_initialMISE<-grid_search(h=h_initial1,p=p[l],xsample=x.
     sample,
  delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  roots_vec_initialMISE[20]<-roots_vec_initialMISE[20]+0.014
  var_xj   <-  p[l]*(1-p[l])
  copt_hat_CV      <-  optimize(f=AR.hat,lower=0,upper=100,$minimum,
  xj=roots_vec_initialMISE,p=p[l],nd=n,varY=var_xj,h0_bin=h0_bin,
  dfh0_binning=h1_binning,h0_LP=h_LP,x.sample=x.sample,Y.sample=Y.
     sample,
  grid_points=grid_points,grid_cl=grid_cl)
  h_opt              <-  copt_hat_CV*(n^(-1/5))
  hopt_vector[l]     <-  h_opt

  roots_vec_LS   <-    grid_search(h=h_opt,p=p[l],xsample=x.sample,
     delta=d,
  grid=grid_points,cl=grid_cl,dlY=grid_dlY)

  if(!is.null(roots_vec_LS)){
    LS_opt   <-    myHDR(roots_vector = roots_vec_LS,d=d,p=p[l],
    h_estimate = h_opt,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    LS_opt   <-    LS_opt[order(sapply(LS_opt,'[[',1))]
  }

  LS_list[[l]]  <- LS_opt
##########       second level       #########
```

```
l<-2
roots_vec_initialMISE<-grid_search(h=h_initial1,p=0.50001,xsample=x
    .sample,
delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
roots_vec_initialMISE[15]<-roots_vec_initialMISE[15]+0.0135
var_xj    <-  0.50001*(1-0.50001)
  copt_hat_CV       <-  optimize(f=AR.hat,lower=0,upper=100,
  xj=roots_vec_initialMISE,p=0.50001,nd=n,varY=var_xj,h0_bin=h0_bin
      ,
  dfh0_binning=h1_binning,h0_LP=h_LP,x.sample=x.sample,
  Y.sample=Y.sample,grid_points=grid_points,grid_cl=grid_cl)$
      minimum
h_opt               <-  copt_hat_CV*(n^(-1/5))
hopt_vector[l]      <-  h_opt

roots_vec_LS    <-    grid_search(h=h_opt,p=0.50001,xsample=x.sample
    ,
delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

if(!is.null(roots_vec_LS)){
  LS_opt    <-    myHDR(roots_vector = roots_vec_LS,d=d,p=p[l],h_
      estimate = h_opt,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  LS_opt    <-    LS_opt[order(sapply(LS_opt,'[[',1))]
}

LS_list[[l]]  <- c(roots_vec_LS,max(x.sample))

##########          third level       #########

l<-3
p<-c(0.20,0.50,0.80)
roots_vec_initialMISE<-grid_search(h=h_initial1,p=p[l],xsample=x.
```

```
    sample,
  delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
  roots_vec_initialMISE[7]<-roots_vec_initialMISE[7]+0.013
  var_xj   <-  p[l]*(1-p[l])
  copt_hat_CV      <-  optimize(f=AR.hat,lower=0,upper=100,
    xj=roots_vec_initialMISE,p=p[l],nd=n,varY=var_xj,h0_bin=h0_bin,
    dfh0_binning=h1_binning,h0_LP=h_LP,x.sample=x.sample,
    Y.sample=Y.sample,grid_points=grid_points,grid_cl=grid_cl)$
      minimum
  h_opt              <-  copt_hat_CV*(n^(-1/5))
  hopt_vector[l]     <-  h_opt

  roots_vec_LS   <-    grid_search(h=h_opt,p=p[l],xsample=x.sample,
  delta=d,grid=grid_points,cl=grid_cl,dlY=grid_dlY)

  if(!is.null(roots_vec_LS)){
    LS_opt   <-    myHDR(roots_vector = roots_vec_LS,d=d,p=p[l],
    h_estimate = h_opt,grid=grid_points,cl=grid_cl,dlY=grid_dlY)
    LS_opt   <-    LS_opt[order(sapply(LS_opt,'[[',1))]
  }

  LS_list[[l]]  <- LS_opt

#########          PLOTS           #########
f_support        <-      seq(from=-2,to=2,length.out=300)

plot(x.sample,Y.sample,type="p",main="",xlab="PCA score",ylab="
   mortality")

lines(f_support,NWkernel_regression_gaussian_binned(x=f_support,
h=hopt_vector[1],grid=grid_points,cl=grid_cl,dlY=grid_dlY))
rect(xleft=LS_list[[1]][[1]][1],ybottom = 0.025, xright=LS_list[[1
```

```
    ]][[1]][2],
ytop=0.05, density = 20, angle = 45, col = NA, border = NULL, lty = 1
    ,lwd=1.5)
rect(xleft=LS_list[[1]][[2]][1],ybottom = 0.025, xright=LS_list[[1
    ]][[2]][2],
ytop=0.05, density = 20, angle = 45, col = NA, border = NULL, lty = 1
    ,lwd=1.5)
rect(xleft=LS_list[[1]][[3]][1],ybottom = 0.025, xright=LS_list[[1
    ]][[3]][2],
ytop=0.05, density = 20, angle = 45, col = NA, border = NULL, lty = 1
    ,lwd=1.5)

plot(x.sample,Y.sample,type="p",main="",xlab="PCA␣score",ylab="
    mortality")

lines(f_support,NWkernel_regression_gaussian_binned(x=f_support,
h=hopt_vector[2],grid=grid_points,cl=grid_cl,dlY=grid_dlY))
rect(xleft=LS_list[[2]][1],ybottom = 0.025, xright=LS_list[[2]][2],
ytop=0.05, density = 20, angle = 45, col = NA, border = NULL, lty = 1
    ,lwd=1.5)

plot(x.sample,Y.sample,type="p",main="",xlab="PCA␣score",ylab="
    mortality")

lines(f_support,NWkernel_regression_gaussian_binned(x=f_support,
h=hopt_vector[3],grid=grid_points,cl=grid_cl,dlY=grid_dlY))
rect(xleft=LS_list[[3]][[1]][1],ybottom = 0.025, xright=LS_list[[3
    ]][[1]][2],
ytop=0.05, density = 20, angle = 45, col = NA, border = NULL, lty = 1
    ,lwd=1.5)
```

B.3 Example code for log-concave highest density region simulations

Common functions

The function *mixlcd* computes the EM algorithm for the log-concave mixture model. The code for this function and the code to select the appropriate number of components in the log-concave mixture model appear in Boonpatcharanon [2019].

```
library(logcondens)
library(mclust)
library(hdrcde)
rm(list=ls())

#-----------------------------------------

##### modified function logConDens #####
mod_logConDens<- function (x, xgrid = NULL, smoothed = FALSE, print =
    FALSE,w=w)
{
  res1 <- activeSetLogCon(x, xgrid = xgrid, print = print,w=w)
  if (identical(smoothed, FALSE)) {
    res <- c(res1)
  }
  if (identical(smoothed, TRUE)) {
    if (identical(xs, NULL)) {
      r <- diff(range(x))
      xs <- seq(min(x) - 0.1 * r, max(x) + 0.1 * r, length = 500)
    }
    smo <- evaluateLogConDens(xs, res1, which = 4:5, gam = gam,
                              print = print)
    f.smoothed <- smo[, "smooth.density"]
    F.smoothed <- smo[, "smooth.CDF"]
```

```
    mode <- xs[f.smoothed == max(f.smoothed)]
    res2 <- list(f.smoothed = f.smoothed, F.smoothed = F.smoothed,
                 gam = gam, xs = xs, mode = mode)
    res <- c(res1, res2)
  }
  res$smoothed <- smoothed
  class(res) <- "dlc"
  return(res)
}

####### function mixlcd ######

mixlcd    <-  function(x, k, plot=FALSE, print=TRUE){

  prec      <-   1e-08
  prec1     <-   1e-08

  # starting values
  mc_start  <-  hc(modelName="V", x)
  class                 <-      c(hclass(mc_start, k))
  props         <-      table(class)

  if(min(props)<2){
    mc_start    <-      hc(modelName="E", x)
    class               <-      c(hclass(mc_start, k))
  }
  props                 <-      as.vector(table(class)/length(class))

  q                     <-      matrix(0, nrow=n, ncol=k)
  for(i in 1:k){
    q[,i]       <-      dnorm(x, mean=mean(x[class==i]), sd=sd(x[
       class==i]))
```

```
}

fit                     <-      as.vector(props %*% t(q))

likold          <-  -100000000

### Start repeating loop ###

cchange <- c()

for (j in 1:500) {
  w                     <-      t(t(q)*props)
  fit           <-      as.vector(props %*% t(q))
  w                     <-      w/fit

  q                     <-      matrix(0, nrow=n, ncol=k)
  knots <-  rep(0,k)
  cphi  <-  c()

  cxx   <-  c()
  countx <- c()

  hphi <- rep(0,k)
  hphi_abs <- rep(0,k)
  hphi_kn <- rep(0,k)
  hphi_lo_noabs <- rep(0,k)
  hphi_lo <- rep(0,k)

  dlc_kobjects <- vector("list", length = k)
  for(i in 1:k){

    whichx              <-      w[,i]/sum(w[,i])>prec1/n
```

```
countx <- c(countx,length(x[whichx]))

mle       <-  mod_logConDens(x=x[whichx], xgrid = NULL,
   smoothed = FALSE, print = FALSE,w=w[whichx,i])

dlc_kobjects[[i]]<-mle

q[whichx,i]  <-  exp(mle$phi)
knots[i]  <-      length(mle$knots)

kn<-mle$knots
phi <-mle$phi[which(mle$IsKnot==1)]
cphi  <- c(cphi,phi)

xx  <- x[which(mle$IsKnot==1)]
cxx <- c(cxx,xx)


dx <- diff(xx)
################### use Jfunction in r
m <- length(phi)
j201 <- J20(phi[1],phi[2])

if (m > 2){
  j20d1 <- J20((phi[2:(m-1)]),(phi[3:m]))
  j20d2 <- J20((phi[2:(m-1)]),(phi[1:(m-2)]))
}

j20m <- J20(phi[m],phi[(m-1)])
j11 <- J11((phi[1:(m-1)]),(phi[2:m]))
########################################
```

```
###### build diagonal
a1 <- dx[1]*j201

if(m > 2){
  a2 <- dx[2:(m-1)]*j20d1
  a3 <- dx[1:(m-2)]*j20d2
}

a4 <- dx[m-1]*j20m
if (m > 2){
  diago <- c(a1,a2+a3,a4)
}
if (m <= 2){
  diago <- c(a1,a4)
}

###################################
dx<-c(0,dx)
j11 <- c(0,j11)
b1 <- dx*j11


knmat <- matrix(rep(0,m*m),m,m)
for (ar in 1:m){
  for (ac in 1:m){
    if (ac == ar){
      knmat[ar,ac] <- diago[ac]
    }
    if (ac == ar+1){
      knmat[ar,ac] <- b1[ac]
    }
```

```
    if(ac != ar && ac!= ar+1) {knmat[ar,ac] <- 0}
  }
}

ss <-lower.tri(knmat)
knmat[ss] <- t(knmat)[ss]
as.matrix(knmat)
dkn <- diago
ddkn <- prod(diago)

########## Hessian for "Locations" #######
 ######## Diagonal of the Matrix ######

####### use Jfunction in r
j01.lo <- J10(phi[2:m],phi[1:(m-1)])
j11.lo <- J11(phi[1:(m-1)],phi[2:m])
j10.lo <- J10(phi[1:(m-1)],phi[2:m])
dxx <- dx[-1]


q20 <- rep(0,m)

q20[2:m] <- (dxx^3)*(j01.lo - j11.lo)
q20[1] <- 1

q11 <-  (dxx^3)*j11.lo
q11 <- c(0,q11)

knmat.lo <- matrix(rep(0,m*m),m,m)
for (rr in 1:m){
  for (cc in 1:m){
    if(cc == rr){
```

```
      knmat.lo[rr,cc] <- q20[cc]
    }
    if (cc == rr+1){
      knmat.lo[rr,cc] <- q11[cc]
    }
    if(cc != rr && cc!= rr+1) {knmat.lo[rr,cc] <- 0}
  }
}

ss.lo <-lower.tri(knmat.lo)
knmat.lo[ss.lo] <- t(knmat.lo)[ss.lo]
as.matrix(knmat.lo)
dlo <- q20
ddlo <- prod(q20)

######## Hessian for "Lower Partials" ######
####### Diagonal of the Matrix #######

################### use Jfunction in r
j10.par1 <- -J10(phi[1],phi[2])
if (m > 2){
  j10.par  <- J10(phi[2:(m-1)],phi[1:(m-2)])-J10(phi[2:(m-1)],
     phi[3:m])
}
j10.parm <- J10(phi[m],phi[m-1])
j11.upar <- J10(phi[1:(m-1)],phi[2:m])
j11.lpar <- -J10(phi[2:m],phi[1:(m-1)])

if(m > 2){
  diago.lo <- c(j10.par1,j10.par,j10.parm)
}
if(m <=2){
```

```
  diago.lo <- c(j10.par1,j10.parm)
}

knmat.lpar <- matrix(rep(0,m*m),m,m)
for (rrr in 1:m){
  for (ccc in 1:m){
    if (ccc == rrr){
      knmat.lpar[rrr,ccc] <- diago.lo[ccc]
    }
    if (ccc == rrr+1){
      knmat.lpar[rrr,ccc] <- j11.upar[rrr]
    }
    if (ccc == rrr-1){
      knmat.lpar[rrr,ccc] <- j11.lpar[ccc]
    }
  }
}
as.matrix(knmat.lpar)
dlpar <- diago.lo
ddlpar <- prod(diago.lo)

######### Hessian for "Upper Partials" ########
knmat.upar <- t(knmat.lpar)
as.matrix(knmat.upar)
############ Full Hessian Matrix #############

mat1 <- cbind(knmat,knmat.upar)
mat2 <- cbind(knmat.lpar,knmat.lo)

hessian <- rbind(mat1,mat2)
hessian_kn <- knmat
```

```
    hphi[i] <- det(hessian)
    hphi_abs[i] <- abs(det(hessian))
    hphi_kn[i] <- abs(det(hessian_kn))
    hphi_lo_noabs[i] <- det(knmat.lo)
    hphi_lo[i] <- abs(det(knmat.lo))


  } ## end loop k

  props       <-      apply(w, 2, sum)/sum(w)
  fit                 <-      as.vector(props %*% t(q))


  temp                <-      t(log(t(t(q)*props)))*props
  liknew              <-      sum(temp[temp>-Inf])

  change      <-      abs((liknew-likold)/likold)
  cchange <- c(cchange,change)

  if(print==TRUE){
    print(iter)
    print(k)
    print(j)
    print(change)
  }

  if (change < prec) break
  likold      <-      liknew

} ## end loop j

######### build FI matrix ########
```

```
  # End repeating loop ##
  ll             <-       sum(log(fit))

  ######### Hessian for "Knots" ########
  ######## Diagonal of Hessian Matrix #####



  return(list(dlc_objects=dlc_kobjects,cchange=cchange, props=props,
     ll=ll, knots=knots,cphi=cphi , countx=countx, fit=fit, phi=phi,
     hphi_abs=hphi_abs, hphi_kn=hphi_kn, hphi_lo = hphi_lo, hphi_lo_
     noabs=hphi_lo_noabs,hphi=hphi, j=j, cxx=cxx))


}


##########SIMULATE FROM THE ESTIMATED LCMM##########
rLCMM        <- function(n, LCMM){

  n_rand <- n
  k      <- length(LCMM$props)

  simulate_grouping <- sample(c(1:k),size=n_rand,replace=TRUE,prob=
     LCMM$props)
  u_vector          <- runif(n_rand,min=0,max=1)
  xrand_obs         <- NULL
  for(i in 1:k){
    which_group_i <- which(simulate_grouping==i)
    xrand_group_i <- quantilesLogConDens(u_vector[which_group_i],LCMM
       $dlc_objects[[i]])[,"quantile"]
```

```
    xrand_obs      <- c(xrand_obs,xrand_group_i)
  }

  return(xrand_obs)
}

#########EVALUATE LCMM AT REAL VALUES##########

evalLCMM      <- function(x_tilda,LCMM){

  k     <- length(LCMM$props)
  fhat  <- rep(0,length(x_tilda))
  for (i in 1:k){
    f_temp <- LCMM$props[i]*evaluateLogConDens(x_tilda, LCMM$dlc_
        objects[[i]])[,"density"]
    fhat   <- fhat+f_temp
  }
  return(fhat)
}
###### ESTIMATE f_alpha ACCORDING TO HYNDMAN QUARTILE ALGORITHM ####

falpha_hat  <- function(alpha,fhat_xtilda){
  m                <- length(fhat_xtilda)
  alpha_hat        <- floor(alpha*m)
  fhat_xtilda_sort <- sort(fhat_xtilda)
  falpha           <- fhat_xtilda_sort[alpha_hat]
  return(falpha)
}


###### FUNCTIONS TO COMPUTE THE HDR_hat GIVEN f_alpha USING
    GRIDSEARCH #######
```

```
#We compute the true HDR by doing bisection method a grid search on f
myBFfzero<-function (f, a, b, num = 1000, eps = 1e-05)
{
  h = abs(b - a)/num
  i = 0
  j = 0
  a1 = b1 = 0
  while (i <= num) {
    a1 = a + i * h
    b1 = a1 + h
    if (f(a1) == 0) {
      root<-a1
      f.root<-f(a1)
    }
    else if (f(b1) == 0) {
      root<-b1
      f.root<-f(b1)
    }
    else if (f(a1) * f(b1) < 0) {
      repeat {
        if (abs(b1 - a1) < eps)
          break
        x <- (a1 + b1)/2
        if (f(a1) * f(x) < 0)
          b1 <- x
        else a1 <- x
      }
      #print(j + 1)
      j = j + 1
      root<-(a1 + b1)/2
      f.root<-f((a1 + b1)/2)
```

```
    }
    i = i + 1
  }
  if (j == 0)
    print("finding root is fail")
  else return(list(root=root,f.root=f.root))
}


####  GET ALL THE SLOPES AND INTERCEPTS RELEVANT TO EACH COMPONENT OF
    THE LCMM  #####

components_function  <- function(dlc_object){
  phi_knots_j <- dlc_object$phi[dlc_object$IsKnot==1]
  knots_j     <- dlc_object$knots
  slopes_j    <- diff(phi_knots_j)/diff(knots_j)
  intercepts_j<- rep(0,length(knots_j)-1)

  for(i in 1:(length(knots_j)-1)){
    intercepts_j[i] <- (knots_j[i+1]*phi_knots_j[i]-knots_j[i]*phi_
       knots_j[i+1])/(knots_j[i+1]-knots_j[i])
  }

  slopes_j[length(knots_j)]     <- 0
  intercepts_j[length(knots_j)] <- 0

  components_table <- cbind(slopes_j,intercepts_j,knots_j)
  return(components_table)

}
```

```
######  GET THE FIRST DERIVATIVE OF fhat  #######

#Note that a direction needs to be specified. This direction is only
   relevant to knot points since f'(b) can be calculated in [a,b) or
   [b,c)

derivative_LCMM  <- function(x,direction="left",LCMM){
  components_table <- lapply(LCMM$dlc_objects,components_function)
  derivative_j <- c()
  j            <- 1
  k            <- length(LCMM$props)
  while(j<=k){
    if (x<components_table[[j]][1,"knots_j"] || x>components_table[[j
       ]][nrow(components_table[[j]]),"knots_j"]){
      derivative_j[j]<-0
      j<-j+1
    }else{
      num_interval   <- sum(x>=components_table[[j]][,"knots_j"])
      if(x %in% components_table[[j]][,"knots_j"]){
        if(direction=="left"){
          num_interval <- num_interval-1
        }else{
          num_interval <- num_interval
        }
      }
      if(num_interval==0){
        derivative_j[j] <- 0
      }else{
        derivative_j[j]<-LCMM$props[j]*components_table[[j]][num_
           interval,"slopes_j"]*evaluateLogConDens(x,LCMM$dlc_objects
           [[j]])[,3]
      }
```

```
      j<-j+1
    }
  }

  dfhat <- sum(derivative_j)
  return(dfhat)
}


#######  FUNCTION THAT RETURNS HOW MANY COMPONENTS ARE ACTIVE IN [a,b
   )  #######

components_at_interval  <- function(a,b,LCMM){
  components_table <- lapply(LCMM$dlc_objects,components_function)
  components <- 0
  j          <- 1
  k          <- length(LCMM$props)
  x          <-(a+b)/2
  while(j<=k){
    if (x<components_table[[j]][1,"knots_j"] || x>components_table[[j
       ]][nrow(components_table[[j]]),"knots_j"]){
      components<-components+0
      j<-j+1
    }else{
      components<-components+1
      j<-j+1
    }
  }
  return(components)
}

######  NEWTON RAPHSON ALGORITHM TO FIND ROOTS  ########
```

```
#Summary: Newton Raphson is an iterative algorithm that requires a
    starting point x0
#Then x1=x0-f(x0)/f'(x0)
#We will extend fhat for interval[a,b), find the roots, and make sure
     the roots are within [a,b) interval.
#If the root is outside of [a,b), eliminate that root.

NewtonRaphson_LCMM  <-  function(x0,p,LCMM){

  components_table    <- lapply(LCMM$dlc_objects,components_function)
  location_components <- c()
  j<-1
  k<-length(LCMM$props)

  while(j<=k){
    num_intervalj <- sum(x0>=components_table[[j]][,"knots_j"])
    if(num_intervalj==0){
      location_j<-NA
    }else{
      if(num_intervalj==length(components_table[[j]][,"knots_j"]) &&
         x!=max(components_table[[j]][,"knots_j"])){
        location_j<-NA
      }else{
        location_j<-num_intervalj
      }
    }
    location_components <- c(location_components,location_j)
    j<-j+1
  }

  active_j            <- c(1:k)[!is.na(location_components)]
```

```
slope_loc          <- location_components[!is.na(location_
   components)]
active_proportions <- LCMM$props[!is.na(location_components)]

f_extended   <- function(x){
  fj  <-  c()
  for(i in 1:length(active_j)){
    aj <- components_table[[active_j[i]]][slope_loc[i],"slopes_j"]
    bj <- components_table[[active_j[i]]][slope_loc[i],"intercepts_
       j"]

    component<- active_proportions[i]*exp(aj*x+bj)
    fj[i]    <- component
  }
  f <- sum(fj)-p
  return(f)
}

df_extended  <- function(x){
  dfj  <- c()
  for(i in 1:length(active_j)){
    aj              <- components_table[[active_j[i]]][slope_loc[i
       ],"slopes_j"]
    bj              <- components_table[[active_j[i]]][slope_loc[i
       ],"intercepts_j"]
    derivative_temp <- aj*active_proportions[i]*exp(aj*x+bj)
    dfj[i]          <- derivative_temp
  }
  df <- sum(dfj)
  return(df)
}
eps <- 1
```

```
  while(eps>.0000000001){
    x1 <- x0-f_extended(x0)/df_extended(x0)
    eps<- abs(x1-x0)
    x0 <- x1
  }
  return(x0)
}


######  NEWTON ALGORITHM TO FIND MINIMUM  #########
#Summary: Newton Method is an iterative algorithm that requires a
   starting point x0
#Then x1=x0-f'(x0)/f''(x0)
#This function only works on intervals [a,b) where fhat(a)*fhat(b)<0.
    Otherwise, fhat at [a,b) is a monotone function and this won't
   work at all.
#We will extend fhat for interval[a,b). Call this function f_extended
   . Then we will find the minimum of f_extended.
#The minimum is within [a,b)

Newton_LCMM           <-  function(x0,LCMM){

  components_table    <- lapply(LCMM$dlc_objects,components_function)
  location_components <- c()
  k<-length(LCMM$props)
  j<-1

  while(j<=k){
    num_intervalj <- sum(x0>=components_table[[j]][,"knots_j"])
    if(num_intervalj==0){
      location_j<-NA
    }else{
```

```
    #if(num_intervalj==length(components_table[[j]][,"knots_j"]) &&
        x0!=max(components_table[[j]][,"knots_j"])){
    if(num_intervalj==length(components_table[[j]][,"knots_j"]) ){
      location_j<-NA
    }else{
      location_j<-num_intervalj
    }
  }
  location_components <- c(location_components,location_j)
  j<-j+1
}

active_j           <- c(1:k)[!is.na(location_components)]
slope_loc          <- location_components[!is.na(location_
   components)]
active_proportions <- LCMM$props[!is.na(location_components)]


df_extended  <- function(x){
  dfj  <- c()
  for(i in 1:length(active_j)){
    aj              <- components_table[[active_j[i]]][slope_loc[i
       ],"slopes_j"]
    bj              <- components_table[[active_j[i]]][slope_loc[i
       ],"intercepts_j"]
    derivative_temp <- aj*active_proportions[i]*exp(aj*x+bj)
    dfj[i]          <- derivative_temp
  }
  df <- sum(dfj)
  return(df)
}
```

```
  d2f_extended   <- function(x){
    d2fj  <-  c()
    for(i in 1:length(active_j)){
      aj <- components_table[[active_j[i]]][slope_loc[i],"slopes_j"]
      bj <- components_table[[active_j[i]]][slope_loc[i],"intercepts_
         j"]

      derivative_temporal <- aj^2*active_proportions[i]*exp(aj*x+bj)
      d2fj[i]    <- derivative_temporal
    }
    d2f <- sum(d2fj)
    return(d2f)
  }
  eps <- 1
  while(eps>.000000001){
    x1 <- x0-df_extended(x0)/d2f_extended(x0)
    eps<- abs(x1-x0)
    x0 <- x1
  }
  return(x0)
}


########  FIND EXACT ROOT WHEN ONLY ONE COMPONENT IS PRESENT IN [a,b)
     #########

exact_root       <- function(knot_a,p,LCMM){

  components_table <- lapply(LCMM$dlc_objects,components_function)
  k                <- length(LCMM$props)
  j                <- 1
```

```
  while(j<=k){

    location <- sum(knot_a>=components_table[[j]][,"knots_j"])
    if(location==length(components_table[[j]][,"knots_j"])||location
       ==0){
      j<- j+1
    }else{
      aj        <- components_table[[j]][location,"slopes_j"]
      bj        <- components_table[[j]][location,"intercepts_j"]
      root      <- (1/aj)*(log(p/LCMM$props[j])-bj)
      j         <- j+1
    }
  }
  return(root)
}


##### FUNCTION TO FIND A ROOT IN INTERVAL [a,b) & FUNCTION TO FIND
   ALL ROOTS in fhat ######

findroot_interval   <-   function(a,b,p,LCMM){

  if(evalLCMM(a,LCMM)<p && evalLCMM(b,LCMM)<p){
    root  <-  NA
  }else{
    if(!(evalLCMM(a,LCMM)>p && evalLCMM(b,LCMM)>p)){
      if(components_at_interval(a=a,b=b,LCMM=LCMM)==1){
        root  <- exact_root(knot_a=a,p=p,LCMM=LCMM)
      }else{
        X0    <- mean(c(a,b))
        root  <- NewtonRaphson_LCMM(x0=X0,p=p,LCMM=LCMM)
```

```
    while(root<a||root>b){
      ss    <-(b-a)/10
      delta <- sample(x=c(ss,-ss),size=1,prob = c(.5,.5))
      X0    <- X0+delta
      root  <- NewtonRaphson_LCMM(x0=X0,p=p,LCMM=LCMM)
    }
  }
}else{
  if(derivative_LCMM(x=a,direction = "right",LCMM=LCMM)*
     derivative_LCMM(x=b,LCMM=LCMM)>=0){
    root  <- NA
  }else{
    x.minimum <- Newton_LCMM(x0=mean(c(a,b)),LCMM)
    minimum <- evalLCMM(x_tilda=x.minimum,LCMM=LCMM)
    if(minimum>p){
      root <- NA
    }else{
      x01   <- mean(c(a,x.minimum))
      x02   <- mean(c(x.minimum,b))
      root1 <- NewtonRaphson_LCMM(x0=x01,p=p,LCMM=LCMM)
      root2 <- NewtonRaphson_LCMM(x0=x02,p=p,LCMM=LCMM)

      while(root1<a||root1>b){
        ss     <-(x.minimum-a)/10
        delta  <- sample(x=c(ss,-ss),size=1,prob = c(.5,.5))
        x01    <- x01+delta
        root1  <- NewtonRaphson_LCMM(x0=x01,p=p,LCMM=LCMM)
      }
      while(root2<a||root2>b){
        ss     <-(b-x.minimum)/10
        delta  <- sample(x=c(ss,-ss),size=1,prob = c(.5,.5))
        x02    <- x02+delta
```

```
            root1  <- NewtonRaphson_LCMM(x0=x02,p=p,LCMM=LCMM)
          }
          root  <- c(root1,root2)
        }
      }
    }
  }
  return(root)
}


function_roots  <- function(superknots,p,LCMM){
  roots_atinterval <- unlist(lapply(1:(length(superknots)-1),function
     (i)findroot_interval(a=superknots[i],b=superknots[i+1],p=p,LCMM=
     LCMM)))
  roots_vector     <- roots_atinterval[!is.na(roots_atinterval)]
  return(roots_vector)
}

#######  TRANSFORM VECTOR OF ROOTS TO THE ACTUAL HDR ########

myHDR<-function(roots_vector,d,p,LCMM,x.sample=x){

  r<-length(roots_vector)
  HDR<-list()
  roots_vector<-sort(roots_vector)
  R<-list()
  i<-1
  left_boundary<-evalLCMM(x_tilda=roots_vector[1]-d,LCMM = LCMM)-p
  right_boundary<-evalLCMM(x_tilda=roots_vector[r]+d,LCMM = LCMM)-p
  if(left_boundary>0){
    R[[i]]<-c(min(x.sample),roots_vector[1])
```

```
    i<-i+1
  }
  if(right_boundary>0){
    R[[i]]<-c(roots_vector[r],max(x.sample))
    i<-i+1
  }
  l<-1
  while(l<length(roots_vector)){
    r1<-roots_vector[l]
    r2<-roots_vector[l+1]
    rm<-(r1+r2)/2
    f.rm<-evalLCMM(x_tilda=rm,LCMM = LCMM)-p
    if(f.rm>0){
      R[[i]]<-c(r1,r2)
      i<-i+1
    }
    l<-l+1
  }
  return(R)
}

#########FUNCTIONS TO COMPUTE \mu(HDR_hat\Delta HDR)########


error_function<-function(HDR_hat,true_HDR){
  n_hdr<-length(HDR_hat)
  counter<-n_hdr
  loss_temp<-0
  integral_regions_type1<-list()

  i_temp<-NULL
  for(i in 1:n_hdr){
```

```
  if(HDR_hat[[i]][2]<=true_HDR[[1]][1]){
    integral_regions_type1<-c(integral_regions_type1,HDR_hat[i])
    loss_temp<-loss_temp+F(HDR_hat[[i]][2])-F(HDR_hat[[i]][1])
    i_temp<-c(i_temp,i)
    counter<-counter-1
  }
}
if(!is.null(i_temp)){HDR_hat<-HDR_hat[-c(i_temp)]}
n_hdr<-length(HDR_hat)
i_temp<-NULL
if(counter>0){
  for(i in 1:n_hdr){
    if(HDR_hat[[i]][1]>=true_HDR[[1]][2]){
      integral_regions_type1<-c(integral_regions_type1,HDR_hat[i])
      loss_temp<-loss_temp+F(HDR_hat[[i]][2])-F(HDR_hat[[i]][1])
      i_temp<-c(i_temp,i)
      counter<-counter-1
    }
  }
  if(!is.null(i_temp)){HDR_hat<-HDR_hat[-c(i_temp)]}
}
i_temp<-NULL
if(counter>0){
  n_hdr<-length(HDR_hat)
  if(HDR_hat[[1]][1]<true_HDR[[1]][1]){
    integral_regions_type1<-c(integral_regions_type1,list(c(HDR_hat
        [[1]][1],true_HDR[[1]][1])))
    loss_temp<-loss_temp+F(true_HDR[[1]][1])-F(HDR_hat[[1]][1])
    HDR_hat[[1]][1]<-true_HDR[[1]][1]
  }
  if(HDR_hat[[n_hdr]][2]>true_HDR[[1]][2]){
```

```
      integral_regions_type1<-c(integral_regions_type1,list(c(true_
          HDR[[1]][2],HDR_hat[[n_hdr]][2])))
      loss_temp<-loss_temp+F(HDR_hat[[n_hdr]][2])-F(true_HDR[[1]][2])
      HDR_hat[[n_hdr]][2]<-true_HDR[[1]][2]
    }
    loss_temp<-loss_temp+F(true_HDR[[1]][2])-F(true_HDR[[1]][1])
    for(i in 1:n_hdr){
      loss_temp<-loss_temp-(F(HDR_hat[[i]][2])-F(HDR_hat[[i]][1]))
    }
  }
  return(loss_temp)
}
multipleIntervals_error<-function(HDR,HDR_estimate){
  R<-length(HDR)
  Rhat<-length(HDR_estimate)
  counter<-Rhat
  error<-0
  r<-1
  temp.hat<-NULL
  while(r<=R){
    indicator1<-c(1:Rhat)[unlist(lapply(HDR_estimate,function(x)x[1]<
        HDR[[r]][2] & x[2]<HDR[[r]][2]))]
    indicator2<-c(1:Rhat)[unlist(lapply(HDR_estimate,function(x)x[1]<
        HDR[[r]][2] & x[2]>=HDR[[r]][2]))]
    if(length(indicator1)==0 & length(indicator2)==0){
      error<-error+F(HDR[[r]][2])-F(HDR[[r]][1])
      r<-r+1
    }else{
      if(length(indicator1)>0){
        temp.hat<-HDR_estimate[(indicator1)]
        HDR_estimate<-HDR_estimate[-indicator1]
        Rhat<-length(HDR_estimate)
```

```
        counter<-counter-length(indicator1)
      }
      indicator2<-c(1:Rhat)[unlist(lapply(HDR_estimate,function(x)x[1
         ]<HDR[[r]][2] & x[2]>=HDR[[r]][2]))]
      if(length(indicator2)>0){
        last_interval<-HDR_estimate[indicator2]
        last_interval[[1]][2]<-HDR[[r]][2]
        temp.hat<-c(temp.hat,last_interval)
        HDR_estimate[[indicator2]][1]<-HDR[[r]][2]
        #counter<-counter-length(indicator2)
      }
      temp.error<-error_function(HDR_hat = temp.hat,true_HDR = HDR[r
         ])
      error<-error+temp.error
      r<-r+1
      temp.hat<-NULL
    }
  }
  if(counter>0){
    sum_error<-sum(unlist(lapply(HDR_estimate[(Rhat-counter+1):Rhat],
       function(x)F(x[2])-F(x[1]))))
    error<-error+sum_error
  }
  return(error)
}
```

Code related to the simulations from density four in Marron and Wand [1992b]

```
###### TRUE FUNCTIONS  ########
fx<-function(x){
  ff<-(2/3)*dnorm(x,mean=0,sd=1)+(1/3)*dnorm(x,mean=0,sd=1/10)
```

```
  return(ff)
}

F<-function(xx){
  pp   <-  c(2/3,1/3)
  s    <- c(0,0)
  r    <- c(1,1/100)
  cdf  <- pp[1]*pnorm(xx,mean=s[1],sd=sqrt(r[1]))+pp[2]*pnorm(xx,mean=
     s[2],sd=sqrt(r[2]))
  return(cdf)
}

#########  FUNCTIONS TO OBTAIN TRUE HDR  #########

true_falpha   <-  function(alpha,n){

  pp   <-  c(2/3,1/3)
  s <- c(0,0)
  r <- c(1,1/100)

  y1   <-  rnorm(n, mean=s[1], sd=sqrt(r[1]))
  y2   <-  rnorm(n, mean=s[2], sd=sqrt(r[2]))

  u    <-  sample(1:2, n, replace=TRUE, prob=pp)
  xx   <-  y1*(u==1)+y2*(u==2)

  f_xx<- fx(xx)
  f_xx<- sort(f_xx)

  location <- floor(n*alpha)
  f_alpha  <- f_xx[location]
  return(f_alpha)
```

```
}

true_hdr<-function(p){
  f_shift<-function(x){
    ff<-(2/3)*dnorm(x,mean=0,sd=1)+(1/3)*dnorm(x,mean=0,sd=1/10)-p
    return(ff)
  }

  r1<-myBFfzero(f_shift,-4,0,eps=.0000000001)$root
  r2<-myBFfzero(f_shift,0,4,eps=.0000000001)$root
  hdr_true<-c(r1,r2)
  return(hdr_true)
}

########    SIMULATIONS    ######

set.seed(89630)
alpha             <-  c(.20,.50,.80)
falpha            <-  true_falpha(alpha,10000000)
true_HDR          <-  lapply(falpha,true_hdr)
M                 <-  250
error_logconcave <- matrix(, nrow = M, ncol = length(alpha))
error_kernel      <- matrix(, nrow = M, ncol = length(alpha))
k_vector          <- c()

m<- 1

while(m<=M){

  ccres<-c()
  cresult<-c()
  mincha <- c()
```

```
maxcha <- c()
iter<-1

n      <-  c(1000)
p      <-  c(2/3,1/3)

s      <- c(0,0)
r      <- c(1,1/100)

kk <- 4


##  generate samples  ##

y1  <-  rnorm(n, mean=s[1], sd=sqrt(r[1]))
y2  <-  rnorm(n, mean=s[2], sd=sqrt(r[2]))

u  <-  sample(1:2, n, replace=TRUE, prob=p)
x  <-  y1*(u==1)+y2*(u==2)
x      <-       sort(x)

bic3  <-  c()
nbic35 <- c()

#### loop for number of components
cres<-c()
cfit<- c()
cumk <- 0

for (k in 1:kk){

  dfcumk <- k-cumk[length(cumk)]
```

```
if(dfcumk !=1) next

mix <-  tryCatch(mixlcd(x, k=k, plot=FALSE, print=FALSE),error=
   function(e) NULL)

if(is.null(mix$cphi) ==TRUE) next

cumk <- c(cumk,k)


ll <- mix$ll
mkn <- sum(mix$knots)

fish <- sum(log(mix$hphi_abs))

prop <- log(mix$props)  #log(pi_j)

kp <- mix$knots


bic3[k]    <-  -2*ll + (2*mkn+k-1)*log(n)


nbic35[k] <- -2*ll  + (2*mkn+k-1)*log(n) - sum(prop) + (fish)


res<-c(n,iter,k,mix$j,ll,mkn,sum(prop),fish,bic3[k],nbic35[k])

cres<-rbind(cres,res)

if(k == 2){
```

```
    mincha[iter] <- min(mix$cchange)
    maxcha[iter] <- max(mix$cchange)
  }


} ## loop k

if(dfcumk ==1){
  ####################################################

  minbic3<-which.min(bic3)
  minnbic35<-which.min(nbic35)

  #######  Other techniques compare to BIC ######

  ## 1. GMM using BIC criterion
  test  <-  Mclust(x, modelNames="V", G=1:kk)
  mcl<-test$G
  Gbic <- test$BIC[1:kk,]
  loglik <- test$loglik

  cres <- cbind(cres,Gbic,loglik)
  ccres<-rbind(ccres,cres)

  result<-c(n,iter,mcl,minbic3,minnbic35)

  cresult<-rbind(cresult,result)
  iter<-iter+1
}

###  RESULTS  ###
```

```
colnames(ccres)<-c("n","iter","k","j","ll","#knots","log(prop)","
    fisher","bic3","nbic35","Gbic","ll_GMM")

colnames(cresult)<-c("n","iter","GMM","bic3","nbic35")

k_hat <- cresult[1,"nbic35"]

k_vector[m] <- k_hat
LCMM<-mixlcd(x=x, k=k_hat, plot=FALSE, print=FALSE)

superknots<- c()
for(i in 1:k_hat){
  knotsj  <- LCMM$dlc_objects[[i]]$knots
  superknots<- c(superknots,knotsj)
}
superknots<-sort(superknots)


###############     CODE FOR VECTOR alpha    ###########

for(l in 1:length(alpha)){
  #############compute the estimate for f_alpha
  xtilda        <- rLCMM(30000,LCMM)
  fhat_xtilda   <- evalLCMM(x_tilda=xtilda,LCMM)
  f_alpha_hat   <- falpha_hat(alpha[l],fhat_xtilda)

  #Kernel HDR

  HDR_object      <- hdr(x,prob=(1-alpha[l])*100)
  no.intervals    <-length(HDR_object$hdr)/2
  HDR_kernel      <-list()
  for(i in 1:no.intervals){
```

```
      HDR_kernel <- c(HDR_kernel,list(c(HDR_object$hdr[2*i-1],HDR_
          object$hdr[2*i])))
    }


    #find the roots for fhat
    #transform a vector of roots into proper HDR regions
    vector_withroots<-function_roots(superknots,p=f_alpha_hat,LCMM=
        LCMM)
    HDR_logconcave  <-myHDR(roots_vector=vector_withroots,d=.0001,p=f
        _alpha_hat,LCMM=LCMM)


    error_logconcave[m,l]   <- multipleIntervals_error(HDR=true_HDR[l
        ],HDR_estimate=HDR_logconcave)
    error_kernel[m,l]       <- multipleIntervals_error(HDR=true_HDR[l
        ],HDR_estimate=HDR_kernel)

  }
  print(m)
  m <- m+1
}
```

Code related to the simulations from density six in Marron and Wand [1992b]

```
####   TRUE FUNCTIONS   ####

fx<-function(x){
  ff<-(1/2)*dnorm(x,mean=-1,sd=2/3)+(1/2)*dnorm(x,mean=1,sd=2/3)
  return(ff)
}
```

```
F<-function(xx){
  pp   <-  c(1/2,1/2)
  s    <- c(-1,1)
  r    <- c(4/9,4/9)
  cdf  <- pp[1]*pnorm(xx,mean=s[1],sd=sqrt(r[1]))+pp[2]*pnorm(xx,mean=
     s[2],sd=sqrt(r[2]))
  return(cdf)
}
#######  FUNCTIONS TO OBTAIN TRUE HDR  #######

true_falpha   <-  function(alpha,n){

  pp   <-  c(1/2,1/2)
  s <- c(-1,1)
  r <- c(4/9,4/9)

  y1   <-  rnorm(n, mean=s[1], sd=sqrt(r[1]))
  y2   <-  rnorm(n, mean=s[2], sd=sqrt(r[2]))

  u    <-  sample(1:2, n, replace=TRUE, prob=pp)
  xx   <-  y1*(u==1)+y2*(u==2)

  f_xx<- fx(xx)
  f_xx<- sort(f_xx)

  location <- floor(n*alpha)
  f_alpha  <- f_xx[location]
  return(f_alpha)
}

true_hdr<-function(p){
```

```
  f_shift<-function(x){
    ff<-(1/2)*dnorm(x,mean=-1,sd=2/3)+(1/2)*dnorm(x,mean=1,sd=2/3)-p
    return(ff)
  }

  r1<-myBFfzero(f_shift,-2,-1,eps=.0000000001)$root
  r2<-myBFfzero(f_shift,-1,0,eps=.0000000001)$root
  r3<-myBFfzero(f_shift,0,1,eps=.0000000001)$root
  r4<-myBFfzero(f_shift,1,2,eps=.0000000001)$root
  hdr_true<-c(list(c(r1,r2)),list(c(r3,r4)))
  return(hdr_true)
}

#######  SIMULATIONS   ########

set.seed(69785)
alpha            <-  c(.20,.50,.80)
falpha           <-  true_falpha(alpha,10000000)
true_HDR         <-  lapply(falpha,true_hdr)
M                <-  250
error_logconcave <- matrix(, nrow = M, ncol = length(alpha))
error_kernel     <- matrix(, nrow = M, ncol = length(alpha))
k_vector         <- c()

m<- 1

while(m<=M){

  ccres<-c()
  cresult<-c()
  mincha <- c()
  maxcha <- c()
```

```
iter<-1

n      <-  c(5000)
p      <-  c(1/2,1/2)

s      <- c(-1,1)
r      <- c(4/9,4/9)

#number of components
kk <- 4


##### Initialize parameters    ######

y1  <-  rnorm(n, mean=s[1], sd=sqrt(r[1]))
y2  <-  rnorm(n, mean=s[2], sd=sqrt(r[2]))

u  <-  sample(1:2, n, replace=TRUE, prob=p)
x  <-  y1*(u==1)+y2*(u==2)
x      <-       sort(x)

bic3  <-  c()
nbic35 <- c()

#### loop for number of components
cres<-c()
cfit<- c()
cumk <- 0

for (k in 1:kk){

  dfcumk <- k-cumk[length(cumk)]
```

```
if(dfcumk !=1) next

mix <-  tryCatch(mixlcd(x, k=k, plot=FALSE, print=FALSE),error=
   function(e) NULL)

if(is.null(mix$cphi) ==TRUE) next

cumk <- c(cumk,k)


ll <- mix$ll
mkn <- sum(mix$knots)

fish <- sum(log(mix$hphi_abs))

prop <- log(mix$props)  #log(pi_j)

kp <- mix$knots


bic3[k]    <-  -2*ll + (2*mkn+k-1)*log(n)


nbic35[k] <- -2*ll  + (2*mkn+k-1)*log(n) - sum(prop) + (fish)



res<-c(n,iter,k,mix$j,ll,mkn,sum(prop),fish,bic3[k],nbic35[k])

cres<-rbind(cres,res)
```

```
  if(k == 2){
    mincha[iter] <- min(mix$cchange)
    maxcha[iter] <- max(mix$cchange)
  }


} ## loop k

if(dfcumk ==1){


  minbic3<-which.min(bic3)
  minnbic35<-which.min(nbic35)


  ###### Other techniques compare to BIC #######

  ## 1. GMM using BIC criterion
  test  <-  Mclust(x, modelNames="V", G=1:kk)
  mcl<-test$G
  Gbic <- test$BIC[1:kk,]
  loglik <- test$loglik

  cres <- cbind(cres,Gbic,loglik)
  ccres<-rbind(ccres,cres)

  result<-c(n,iter,mcl,minbic3,minnbic35)

  cresult<-rbind(cresult,result)
  iter<-iter+1
}
```

```
colnames(ccres)<-c("n","iter","k","j","ll","#knots","log(prop)","
    fisher","bic3","nbic35","Gbic","ll_GMM")

colnames(cresult)<-c("n","iter","GMM","bic3","nbic35")

k_hat <- cresult[1,"nbic35"]

k_vector[m] <- k_hat
LCMM<-mixlcd(x=x, k=k_hat, plot=FALSE, print=FALSE)

superknots<- c()
for(i in 1:k_hat){
  knotsj  <- LCMM$dlc_objects[[i]]$knots
  superknots<- c(superknots,knotsj)
}
superknots<-sort(superknots)

########     CODE FOR VECTOR alpha    ########

for(l in 1:length(alpha)){
  #############compute the estimate for f_alpha
  xtilda        <- rLCMM(30000,LCMM)
  fhat_xtilda   <- evalLCMM(x_tilda=xtilda,LCMM)
  f_alpha_hat   <- falpha_hat(alpha[l],fhat_xtilda)

  #Kernel HDR

  HDR_object      <- hdr(x,prob=(1-alpha[l])*100)
  no.intervals    <-length(HDR_object$hdr)/2
  HDR_kernel      <-list()
  for(i in 1:no.intervals){
```

```
      HDR_kernel <- c(HDR_kernel,list(c(HDR_object$hdr[2*i-1],HDR_
          object$hdr[2*i])))
    }


    vector_withroots<-function_roots(superknots,p=f_alpha_hat,LCMM=
        LCMM)
    HDR_logconcave  <-myHDR(roots_vector=vector_withroots,d=.0001,p=f
        _alpha_hat,LCMM=LCMM)


    error_logconcave[m,l]   <- multipleIntervals_error(HDR=true_HDR[[
        l]],HDR_estimate=HDR_logconcave)
    error_kernel[m,l]       <- multipleIntervals_error(HDR=true_HDR[[
        l]],HDR_estimate=HDR_kernel)

  }
  print(m)
  m <- m+1
}
```

Code related to the simulations from a mixture of T distributions

```
####   TRUE FUNCTIONS  ####
fx<-function(x){
  ff<-(2/3)*dt(x,df=5)+(1/3)*dt(x,df=2)
  return(ff)
}


F<-function(xx){
  pp  <-  c(2/3,1/3)
  cdf <- pp[1]*pt(xx,df=5)+pp[2]*pt(xx,df=2)
```

```
  return(cdf)
}

######  FUNCTIONS TO OBTAIN TRUE HDR  ######

true_falpha   <-  function(alpha,n){

  pp  <-  c(2/3,1/3)


  y1  <-  rt(n=n,df=5)
  y2  <-  rt(n=n,df=2)

  u   <-  sample(1:2, n, replace=TRUE, prob=pp)
  xx  <-  y1*(u==1)+y2*(u==2)

  f_xx<- fx(xx)
  f_xx<- sort(f_xx)

  location <- floor(n*alpha)
  f_alpha  <- f_xx[location]
  return(f_alpha)
}

true_hdr<-function(p){
  f_shift<-function(x){
    ff<-(2/3)*dt(x,df=5)+(1/3)*dt(x,df=2)-p
    return(ff)
  }

  r1<-myBFfzero(f_shift,-4,0,eps=.0000000001)$root
  r2<-myBFfzero(f_shift,0,4,eps=.0000000001)$root
```

```
  hdr_true<-c(r1,r2)
  return(hdr_true)
}

#####    SIMULATIONS    #####

set.seed(89630)
alpha             <-  c(.20,.50,.80)
falpha            <-  true_falpha(alpha,10000000)
true_HDR          <-  lapply(falpha,true_hdr)
M                 <-  250
error_logconcave <- matrix(, nrow = M, ncol = length(alpha))
error_kernel      <- matrix(, nrow = M, ncol = length(alpha))
k_vector          <- c()

m<- 1

while(m<=M){

  ccres<-c()
  cresult<-c()
  mincha <- c()
  maxcha <- c()
  iter<-1

  n      <-  c(1000)
  p      <-  c(2/3,1/3)


  #number of components
  kk <- 4
```

```
y1  <-  rt(n=n,df=5)
y2  <-  rt(n=n,df=2)

u   <-  sample(1:2, n, replace=TRUE, prob=p)
x   <-  y1*(u==1)+y2*(u==2)
x       <-    sort(x)

bic3  <-  c()
nbic35 <- c()

#### loop for #of components
cres<-c()
cfit<- c()
cumk <- 0

for (k in 1:kk){

  dfcumk <- k-cumk[length(cumk)]

  if(dfcumk !=1) next

  mix <-  tryCatch(mixlcd(x, k=k, plot=FALSE, print=FALSE),error=
     function(e) NULL)

  if(is.null(mix$cphi) ==TRUE) next

  cumk <- c(cumk,k)


  ll <- mix$ll
```

```
    mkn <- sum(mix$knots)

    fish <- sum(log(mix$hphi_abs))

    prop <- log(mix$props)  #log(pi_j)

    kp <- mix$knots


    bic3[k]     <-  -2*ll + (2*mkn+k-1)*log(n)



    nbic35[k] <- -2*ll  + (2*mkn+k-1)*log(n) - sum(prop) + (fish)


    res<-c(n,iter,k,mix$j,ll,mkn,sum(prop),fish,bic3[k],nbic35[k])

    cres<-rbind(cres,res)

    if(k == 2){
      mincha[iter] <- min(mix$cchange)
      maxcha[iter] <- max(mix$cchange)
    }


} ## loop k

if(dfcumk ==1){


    minbic3<-which.min(bic3)
```

```
  minnbic35<-which.min(nbic35)


  ########## Other techniques compare to BIC #########

  ## 1. GMM using BIC criterion
  test  <-  Mclust(x, modelNames="V", G=1:kk)
  mcl<-test$G
  Gbic <- test$BIC[1:kk,]
  loglik <- test$loglik

  cres <- cbind(cres,Gbic,loglik)
  ccres<-rbind(ccres,cres)

  result<-c(n,iter,mcl,minbic3,minnbic35)

  cresult<-rbind(cresult,result)
  iter<-iter+1
}


colnames(ccres)<-c("n","iter","k","j","ll","#knots","log(prop)","
    fisher","bic3","nbic35","Gbic","ll_GMM")

colnames(cresult)<-c("n","iter","GMM","bic3","nbic35")

k_hat <- cresult[1,"nbic35"]

k_vector[m] <- k_hat
LCMM<-mixlcd(x=x, k=k_hat, plot=FALSE, print=FALSE)

superknots<- c()
```

```
for(i in 1:k_hat){
  knotsj  <- LCMM$dlc_objects[[i]]$knots
  superknots<- c(superknots,knotsj)
}
superknots<-sort(superknots)

#########     CODE FOR VECTOR alpha    ########
for(l in 1:length(alpha)){
  #############compute the estimate for f_alpha
  xtilda         <- rLCMM(30000,LCMM)
  fhat_xtilda    <- evalLCMM(x_tilda=xtilda,LCMM)
  f_alpha_hat    <- falpha_hat(alpha[l],fhat_xtilda)

  #kernel HDR

  HDR_object       <- hdr(x,prob=(1-alpha[l])*100)
  no.intervals     <-length(HDR_object$hdr)/2
  HDR_kernel       <-list()
  for(i in 1:no.intervals){

    HDR_kernel <- c(HDR_kernel,list(c(HDR_object$hdr[2*i-1],HDR_
       object$hdr[2*i])))
  }


  vector_withroots<-function_roots(superknots,p=f_alpha_hat,LCMM=
     LCMM)
  HDR_logconcave  <-myHDR(roots_vector=vector_withroots,d=.0001,p=f
     _alpha_hat,LCMM=LCMM)


  error_logconcave[m,l]    <- multipleIntervals_error(HDR=true_HDR[l
```

```
        ],HDR_estimate=HDR_logconcave)
    error_kernel[m,1]       <- multipleIntervals_error(HDR=true_HDR[1
        ],HDR_estimate=HDR_kernel)

  }
  print(m)
  m <- m+1
}
```

B.4 Example code for Melbourne's temperature data set

We present the code necessary for the analysis of the first two densities. The code for the remaining five follows.

```
library(logcondens)
library(mclust)  ## for GMM
library(hdrcde)
library(lubridate)

rm(list=ls())

#-----------------------------------------
set.seed(12345)


########         TEMPERATURES DATA          #######
########         from hdrcde package        #######
data<- maxtemp
head(data)
length(data)
plot(data)
str(data)
```

```
today_temp        <-  data[1:3649]
tomorrow_temp     <-  data[2:3650]

data              <-  as.data.frame(cbind(today_temp,tomorrow_temp))
colnames(data)    <-  c("today","tomorrow")
head(data)
summary(data$today)

data$group    <-  0
data$group[data$today>=5 & data$today<10]    <- 1
data$group[data$today>=10 & data$today<15]   <- 2
data$group[data$today>=15 & data$today<20]   <- 3
data$group[data$today>=20 & data$today<25]   <- 4
data$group[data$today>=25 & data$today<30]   <- 5
data$group[data$today>=30 & data$today<35]   <- 6
data$group[data$today>=35 & data$today<40]   <- 7
data$group[data$today>=40 & data$today<45]   <- 8

data$group[data$today>=5&data$today<10]

### EM algorithm for the LCMM ###
LC_MLE     <-  function(x, k, plot=FALSE, print=TRUE){

  prec      <-   1e-08
  prec1      <-  1e-08

  n        <-  length(x)
  init           <-      Mclust(x, G=k)
  class          <-      init$z

  cu        <-  outer(unique(x), x, `==`) %*% class
  x         <- unique(x)
```

```
n         <- length(x)

for( i in 1:ncol(cu)){
  cu[,i]  <- cu[,i]/sum(cu[,i])
}

props          <-       init$parameters$pro

mle_list  <- list()
for(i in 1:k){
  mle_list[[i]] <- mod_logConDens(x=x, xgrid = NULL, smoothed =
     FALSE, print = FALSE,w=cu[,i])
}

likold    <-  0
for(i in 1:k){
  likold   <-  likold+props[i]*mle_list[[i]]$L
}


cchange <- c()

#ITERATION PART FOR THE EM ALGORITHM

for (j in 1:500) {
  q                    <-       matrix(0, nrow=n, ncol=k)
  knots <-  rep(0,k)
  cphi  <-  c()
  cxx   <-  c()
  countx <- c()

  hphi <- rep(0,k)
```

```
hphi_abs <- rep(0,k)
hphi_kn <- rep(0,k)
hphi_lo_noabs <- rep(0,k)
hphi_lo <- rep(0,k)
print(j)

# E-STEP

denominator  <-  0
for(i in 1:k){
  denominator   <- denominator+ props[i]*exp(mle_list[[i]]$phi)
}
for(i in 1:k){
  cu[,i]  <-  props[i]*exp(mle_list[[i]]$phi)/denominator
}

props   <-  apply(cu,2,sum)/n


# M-STEP
mle_list  <- list()
dlc_kobjects <- vector("list", length = k)
liknew<-0
for(i in 1:k){
  mle <- mod_logConDens(x=x, xgrid = NULL, smoothed = FALSE,
     print = FALSE,w=cu[,i])
  mle_list[[i]]<-mle
  dlc_kobjects[[i]]<-mle

  q[,i]  <-  exp(mle$phi)
  knots[i]  <-      length(mle$knots)
```

```
kn<-mle$knots
phi <-mle$phi[which(mle$IsKnot==1)]
cphi  <- c(cphi,phi)


xx  <- x[which(mle$IsKnot==1)]
cxx <- c(cxx,xx)


dx <- diff(xx)
#################### use Jfunction in r
m <- length(phi)
j201 <- J20(phi[1],phi[2])

if (m > 2){
  j20d1 <- J20((phi[2:(m-1)]),(phi[3:m]))
  j20d2 <- J20((phi[2:(m-1)]),(phi[1:(m-2)]))
}

j20m <- J20(phi[m],phi[(m-1)])
j11 <- J11((phi[1:(m-1)]),(phi[2:m]))
##############################

###### build diagonal
a1 <- dx[1]*j201

if(m > 2){
  a2 <- dx[2:(m-1)]*j20d1
  a3 <- dx[1:(m-2)]*j20d2
}

a4 <- dx[m-1]*j20m
```

```
if (m > 2){
  diago <- c(a1,a2+a3,a4)
}
if (m <= 2){
  diago <- c(a1,a4)
}

############################
dx<-c(0,dx)
j11 <- c(0,j11)
b1 <- dx*j11


knmat <- matrix(rep(0,m*m),m,m)
for (ar in 1:m){
  for (ac in 1:m){
    if (ac == ar){
      knmat[ar,ac] <- diago[ac]
    }
    if (ac == ar+1){
      knmat[ar,ac] <- b1[ac]
    }
    if(ac != ar && ac!= ar+1) {knmat[ar,ac] <- 0}
  }
}

ss <-lower.tri(knmat)
knmat[ss] <- t(knmat)[ss]
as.matrix(knmat)
dkn <- diago
ddkn <- prod(diago)
```

```
######## Hessian for "Locations" ######
#### Diagonal of the Matrix ####

#### use Jfunction in r
j01.lo <- J10(phi[2:m],phi[1:(m-1)])
j11.lo <- J11(phi[1:(m-1)],phi[2:m])
j10.lo <- J10(phi[1:(m-1)],phi[2:m])
dxx <- dx[-1]



q20 <- rep(0,m)
q20[2:m] <- (dxx^3)*(j01.lo - j11.lo)
q20[1] <- 1

q11 <-  (dxx^3)*j11.lo
q11 <- c(0,q11)

knmat.lo <- matrix(rep(0,m*m),m,m)
for (rr in 1:m){
  for (cc in 1:m){
    if(cc == rr){
      knmat.lo[rr,cc] <- q20[cc]
    }
    if (cc == rr+1){
      knmat.lo[rr,cc] <- q11[cc]
    }
    if(cc != rr && cc!= rr+1) {knmat.lo[rr,cc] <- 0}
  }
}

ss.lo <-lower.tri(knmat.lo)
knmat.lo[ss.lo] <- t(knmat.lo)[ss.lo]
```

```
as.matrix(knmat.lo)
dlo <- q20
ddlo <- prod(q20)

###### Hessian for "Lower Partials" #####
##### Diagonal of the Matrix ####

####### use Jfunction in r
j10.par1 <- -J10(phi[1],phi[2])
if (m > 2){
  j10.par  <- J10(phi[2:(m-1)],phi[1:(m-2)])-J10(phi[2:(m-1)],
     phi[3:m])
}
j10.parm <- J10(phi[m],phi[m-1])
j11.upar <- J10(phi[1:(m-1)],phi[2:m])
j11.lpar <- -J10(phi[2:m],phi[1:(m-1)])

if(m > 2){
  diago.lo <- c(j10.par1,j10.par,j10.parm)
}
if(m <=2){
  diago.lo <- c(j10.par1,j10.parm)
}

knmat.lpar <- matrix(rep(0,m*m),m,m)
for (rrr in 1:m){
  for (ccc in 1:m){
    if (ccc == rrr){
      knmat.lpar[rrr,ccc] <- diago.lo[ccc]
    }
    if (ccc == rrr+1){
      knmat.lpar[rrr,ccc] <- j11.upar[rrr]
```

```
      }
      if (ccc == rrr-1){
        knmat.lpar[rrr,ccc] <- j11.lpar[ccc]
      }
    }
  }
  as.matrix(knmat.lpar)
  dlpar <- diago.lo
  ddlpar <- prod(diago.lo)
  ####### Hessian for "Upper Partials" ######
  knmat.upar <- t(knmat.lpar)
  as.matrix(knmat.upar)
  ############ Full Hessian Matrix #############

  mat1 <- cbind(knmat,knmat.upar)
  mat2 <- cbind(knmat.lpar,knmat.lo)

  hessian <- rbind(mat1,mat2)
  hessian_kn <- knmat

  hphi[i] <- det(hessian)
  hphi_abs[i] <- abs(det(hessian))
  hphi_kn[i] <- abs(det(hessian_kn))
  hphi_lo_noabs[i] <- det(knmat.lo)
  hphi_lo[i] <- abs(det(knmat.lo))

  liknew   <-  liknew+props[i]*mle_list[[i]]$L
}#end loop k

change      <-      abs((liknew-likold)/likold)
cchange <- c(cchange,change)
```

```
    if (change < prec) break
    likold      <-      liknew


    if(print==TRUE){

      print(k)
      print(j)
      print(change)
    }

  }
  ###### build FI matrix #######

  ##### End repeating loop ####

  ll  <-  0
  for(i in 1:k){
    ll   <- ll+ props[i]*exp(mle_list[[i]]$phi)
  }
  ll <- sum(log(ll))
  ####### Hessian for "Knots" ######
  ####### Diagonal of Hessian Matrix ######



  return(list(dlc_objects=dlc_kobjects,cchange=cchange, props=props,
     ll=ll, knots=knots,cphi=cphi , countx=countx, phi=phi, hphi_abs=
     hphi_abs, hphi_kn=hphi_kn, hphi_lo = hphi_lo, hphi_lo_noabs=hphi
     _lo_noabs,hphi=hphi, j=j, cxx=cxx))

}
```

```
####  Function for log-concave HDR  ##
LCHDR_function     <-   function(LCMM,alpha){

  k_hat    <-  length(LCMM$props)

  superknots<- c()

  for(i in 1:k_hat){
    knotsj  <- LCMM$dlc_objects[[i]]$knots
    superknots<- c(superknots,knotsj)
  }

  superknots<-sort(unique(superknots))

  ####     CODE FOR VECTOR alpha    ####

  HDR_kernel_list       <-  list()
  HDR_logconcave_list   <-  list()
  for(l in 1:length(alpha)){
    ## compute the estimate for f_alpha

    xtilda         <- rLCMM(30000,LCMM)
    fhat_xtilda    <- evalLCMM(x_tilda=xtilda,LCMM)
    f_alpha_hat    <- falpha_hat(alpha[l],fhat_xtilda)

    # Kernel HDR

    HDR_object       <- hdr(x,prob=(1-alpha[l])*100)
    no.intervals     <-length(HDR_object$hdr)/2
    HDR_kernel       <-list()
    for(i in 1:no.intervals){
```

```
      HDR_kernel <- c(HDR_kernel,list(c(HDR_object$hdr[2*i-1],HDR_
          object$hdr[2*i])))
    }
    HDR_kernel_list[[l]]  <-  HDR_kernel

    vector_withroots<-function_roots(superknots,p=f_alpha_hat,LCMM=
        LCMM)

    HDR_logconcave  <-myHDR(roots_vector=vector_withroots,d=.0001,p=f
        _alpha_hat,LCMM=LCMM)

    HDR_logconcave_list[[l]]  <-  HDR_logconcave

  }
  return(list(HDR_kernel_list=HDR_kernel_list,HDR_logconcave_list=HDR
      _logconcave_list))

}

alpha    <-  c(0.2,0.5,0.8)

#### FIRST DENSITY
x1                    <-       data[data$group==1,2]
n          <- length(x1)

ccres<-c()
cresult<-c()
mincha <- c()
maxcha <- c()
iter<-1
```

```
kk <- 4

x       <-      sort(x1)

bic3  <-  c()
nbic35 <- c()

### Select the number of components
cres<-c()
cfit<- c()
cumk <- 0

for (k in 1:kk){

  dfcumk <- k-cumk[length(cumk)]

  if(dfcumk !=1) next

  mix <-  tryCatch( LC_MLE(x, k=k, plot=FALSE, print=TRUE),error=
     function(e) NULL)


  if(is.null(mix$cphi) ==TRUE) next
  cumk <- c(cumk,k)



  ## ------------ criterion -----------
  ll <- mix$ll
  mkn <- sum(mix$knots)

  fish <- sum(log(mix$hphi_abs))
```

```
  prop <- log(mix$props)

  kp <- mix$knots


  bic3[k]     <-  -2*ll + (2*mkn+k-1)*log(n)

  ## new for present

  nbic35[k] <- -2*ll  + (2*mkn+k-1)*log(n) - sum(prop) + (fish)


  ## ---------- criterion -----------
  res<-c(n,iter,k,mix$j,ll,mkn,sum(prop),fish,bic3[k],nbic35[k])

  cres<-rbind(cres,res)

  if(k == 2){
    mincha[iter] <- min(mix$cchange)
    maxcha[iter] <- max(mix$cchange)
  }


} ## loop k


minbic3<-which.min(bic3)
minnbic35<-which.min(nbic35)


######### Other techniques compare to BIC #######
```

```
## 1. GMM using BIC criterion
kk<-max(cumk)
test  <-  Mclust(x, modelNames="V", G=1:kk)
mcl<-test$G
Gbic <- test$BIC[1:kk,]
loglik <- test$loglik

cres <- cbind(cres,Gbic,loglik)
ccres<-rbind(ccres,cres)

result<-c(n,iter,mcl,minbic3,minnbic35)

cresult<-rbind(cresult,result)


colnames(ccres)<-c("n","iter","k","j","ll","#knots","log(prop)","
   fisher","bic3","nbic35","Gbic","ll_GMM")

colnames(cresult)<-c("n","iter","GMM","bic3","nbic35")

ccres
cresult
k_hat1               <- cresult[1,"nbic35"]
LCMM1                <- LC_MLE(k=k_hat1,x=x1)
density_one_HDR     <- LCHDR_function(LCMM1,alpha)
density_one_HDR$HDR_kernel_list
density_one_HDR$HDR_logconcave_list

#### SECOND DENSITY
x2                     <-       data[data$group==2,2]
n            <- length(x2)
```

```
ccres<-c()
cresult<-c()
mincha <- c()
maxcha <- c()


kk <- 4

x       <-      sort(x2)

bic3  <-  c()
nbic35 <- c()

#### Select the number of components
cres<-c()
cfit<- c()
cumk <- 0

for (k in 1:kk){

  dfcumk <- k-cumk[length(cumk)]

  if(dfcumk !=1) next

  mix <-  tryCatch( LC_MLE(x, k=k, plot=FALSE, print=TRUE),error=
     function(e) NULL)

  if(is.null(mix$cphi) ==TRUE) next

  cumk <- c(cumk,k)
```

```
  ##---------- criterion ------------
  ll <- mix$ll
  mkn <- sum(mix$knots)

  fish <- sum(log(mix$hphi_abs))

  prop <- log(mix$props)

  kp <- mix$knots


  bic3[k]     <-  -2*ll + (2*mkn+k-1)*log(n)

  ## new for present

  nbic35[k] <- -2*ll  + (2*mkn+k-1)*log(n) - sum(prop) + (fish)


  ##--------- criterion ----------
  res<-c(n,iter,k,mix$j,ll,mkn,sum(prop),fish,bic3[k],nbic35[k])

  cres<-rbind(cres,res)

  if(k == 2){
    mincha[iter] <- min(mix$cchange)
    maxcha[iter] <- max(mix$cchange)
  }


} ## loop k
```

```
if(dfcumk ==1){

  minbic3<-which.min(bic3)
  minnbic35<-which.min(nbic35)

  ######### Other techniques compare to BIC ########

  ## 1. GMM using BIC criterion
  kk<-max(cumk)
  test  <-  Mclust(x, modelNames="V", G=1:kk)
  mcl<-test$G
  Gbic <- test$BIC[1:kk,]
  loglik <- test$loglik

  cres <- cbind(cres,Gbic,loglik)
  ccres<-rbind(ccres,cres)

  result<-c(n,iter,mcl,minbic3,minnbic35)

  cresult<-rbind(cresult,result)
}

colnames(ccres)<-c("n","iter","k","j","ll","#knots","log(prop)","
   fisher","bic3","nbic35","Gbic","ll_GMM")
colnames(cresult)<-c("n","iter","GMM","bic3","nbic35")

ccres
cresult
k_hat2              <- cresult[1,"nbic35"]
LCMM2               <- LC_MLE(k=k_hat2,x=x2)
density_two_HDR     <-   LCHDR_function(LCMM2,alpha)
density_two_HDR$HDR_kernel_list
```

```
density_two_HDR$HDR_logconcave_list
```

Bibliography

H. A. Adler, R.J. Level Crossings for Random Fields. *The Annals of Probability*, 4(1):1 – 12, 1976. doi: 10.1214/aop/1176996176. URL `https://doi.org/10.1214/aop/1176996176`.

S. G. T. J. Adler, R.J. Excursion sets of three classes of stable random fields. *Advances in Applied Probability*, 42(2):293–318, 2010. ISSN 00018678. URL `http://www.jstor.org/stable/25683820`.

H. Akaike. Statistical predictor identification. *Annals of the Institute of Statistical Mathematics*, 22:203–217, 1970.

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

A. Baíllo. Total error in a plug-in estimator of level sets. *Universidad Carlos III, Departamento de Estadística y Econometría, Statistics and Econometrics Working Papers*, 01 2003.

A. Baíllo, A. Cuevas, and A. Justel. Set estimation and nonparametric detection. *Canadian Journal of Statistics*, 28(4):765–782, 2000. doi: https://doi.org/10.2307/3315915.

A. Baíllo, J. Cuesta-Albertos, and A. Cuevas. Convergence rates in nonparametric estimation of level sets. *Statistics and Probability Letters*, 53:27–35, 02 2001. doi: 10.1016/S0167-7152(01)00006-2.

F. Balabdaoui, K. Rufibach, and J. A. Wellner. Limit distribution theory for maximum

likelihood estimation of a log-concave density. *The Annals of Statistics*, 37(3):1299–1331, 06 2009. doi: 10.1214/08-AOS609.

P. Bhattacharya. Some aspects of change-point analysis. *Lecture Notes-Monograph Series*, 23:28–56, 1994. ISSN 07492170. URL http://www.jstor.org/stable/4355761.

L. Birgé. Estimation of unimodal densities without smoothness assumptions. *Ann. Statist.*, 25(3):970–981, 06 1997. doi: 10.1214/aos/1069362733. URL https://doi.org/10.1214/aos/1069362733.

S. Boonpatcharanon. *Semiparametric multivariate density estimation using copulas and shape-constraints*. PhD thesis, York University, May 2019.

E. J. Calabrese and L. A. Baldwin. Defining hormesis. *Human & Experimental Toxicology*, 21(2):91–97, 2002. doi: 10.1191/0960327102ht217oa. PMID: 12102503.

R. J. Carroll, J. Fan, I. Gijbels, and M. P. Wand. Generalized partially linear single-index models. *J. Amer. Statist. Assoc.*, 92(438):477–489, 1997. ISSN 0162-1459. doi: 10.2307/2965697.

L. Cavalier. Nonparametric estimation of regression level sets. *Statistics*, 29(2):131–160, 1997. doi: 10.1080/02331889708802579.

G. T. Chang and G. Walther. Clustering with mixtures of log-concave distributions. *Comput. Stat. Data Anal.*, 51:6242–6251, 2007.

Y. Chen and R. Samworth. Generalised additive and index models with shape constraints. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78, 04 2014. doi: 10.1111/rssb.12137.

J. Cooper and K. Hanson. Decompression sickness, 2021. URL https://www.ncbi.nlm.nih.gov/books/NBK537264/.

H. Cramér. *Mathematical Methods of Statistics*. Princeton Univ. Press, 1946. ISBN 9780691005478.

P. G. Craven and G. Wahba. Smoothing noisy data with spline functions. *Numerische Mathematik*, 31:377–403, 1975.

A. Cuevas, W. González-Manteiga, and A. Casal. Plug-in estimation of general level sets. *Australian and New Zealand Journal of Statistics*, 48:7 – 19, 03 2006. doi: 10.1111/j.1467-842X.2006.00421.x.

M. Cule and R. Samworth. Theoretical properties of the log-concave maximum likelihood estimator of a multidimensional density. *Electronic Journal of Statistics*, 4:254–270, 2010. doi: 10.1214/09-EJS505.

M. Cule, R. Gramacy, and R. Samworth. Logconcdead: An r package for maximum likelihood estimation of a multivariate log-concave density. *Journal of Statistical Software, Articles*, 29(2):1–20, 2009. ISSN 1548-7660. doi: 10.18637/jss.v029.i02.

M. Cule, R. Samworth, and M. Stewart. Maximum likelihood estimation of a multidimensional log-concave density. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 72(5):545–607, 2010. ISSN 13697412, 14679868.

L. Dümbgen and K. Rufibach. Maximum likelihood estimation of a log-concave density and its distribution function: Basic properties and uniform consistency. *Bernoulli*, 15(1):40–68, 02 2009. doi: 10.3150/08-BEJ141.

L. Dümbgen, R. Samworth, and D. Schuhmacher. Approximation by log-concave distributions, with applications to regression. *The Annals of Statistics*, 39(2):702–730, 04 2011. doi: 10.1214/10-AOS853.

C. R. Doss and J. A. Wellner. Global rates of convergence of the mles of log-concave and s-concave densities. *The Annals of Statistics*, 44(3):954–981, 06 2016. doi: 10.1214/15-AOS1394.

C. R. Doss and G. Weng. Bandwidth selection for kernel density estimators of multivariate level sets and highest density regions. *Electronic Journal of Statistics*, 12(2):4313–4376, 2018. doi: 10.1214/18-EJS1501.

L. Duembgen, A. Huesler, and K. Rufibach. Active set and em algorithms for log-concave densities based on complete and censored data. Technical report, Technical Report 61,IMSV, Univ. Bern. arXiv:0707.4643., 2007.

J. Fan. Design-adaptive nonparametric regression. *Journal of the American Statistical Association*, 87(420):998–1004, 1992. ISSN 01621459.

J. Fan, N. E. Heckman, and M. P. Wand. Local polynomial kernel regression for generalized linear models and quasi-likelihood functions. *Journal of the American Statistical Association*, 90(429):141–150, 1995. ISSN 01621459.

R. A. Fisher. Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 22(5):700–725, 1925. doi: 10.1017/S0305004100009580.

R. A. Fisher and E. J. Russell. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922. doi: 10.1098/rsta.1922.0009.

T. Gasser and J. Engel. The choice of weights in kernel regression estimation. *Biometrika*, 77(2):377–381, 06 1990. ISSN 0006-3444. doi: 10.1093/biomet/77.2.377.

G. Gayraud and J. Rousseau. Rates of convergence for a bayesian level set estimation. *Scandinavian Journal of Statistics*, 32:639–660, 12 2005. doi: 10.1111/j.1467-9469.2005.00448.x.

I. Gijbels and A. Goderniaux. Bandwidth selection for changepoint estimation in nonparametric regression. *Technometrics*, 46(1):76–86, 2004. doi: 10.1198/004017004000000130.

U. Grenander. On the theory of mortality measurement. *Scandinavian Actuarial Journal*, 1956(2):125–153, 1956. doi: 10.1080/03461238.1956.10414944.

P. Groeneboom, G. Jongbloed, and J. A. Wellner. Estimation of a convex function: Characterizations and asymptotic theory. *The Annals of Statistics*, 29(6):1653–1698, 12 2001. doi: 10.1214/aos/1015345958.

P. Hall and K. Kang. Bandwidth choice for nonparametric classification. *Annals of Statistics*, 33:284–306, 2005.

B. E. Hansen. Uniform convergence rates for kernel estimation with dependent data. *Econometric Theory*, 24(3):726–748, 2008. ISSN 0266-4666. doi: 10.1017/S0266466608080304.

W. Hardle and J. S. Marron. Optimal bandwidth selection in nonparametric regression function estimation. *The Annals of Statistics*, 13(4):1465–1481, 1985. ISSN 00905364.

E. Herrmann. Local bandwidth choice in kernel regression estimation. *Journal of Computational and Graphical Statistics*, 6(1):35–54, 1997. ISSN 10618600.

W. Härdle. Approximations to the mean integrated squared error with applications to optimal bandwidth selection for nonparametric regression function estimators. *Journal of Multivariate Analysis*, 18(1):150 – 168, 1986. ISSN 0047-259X. doi: https://doi.org/10.1016/0047-259X(86)90066-7.

W. Härdle and J. S. Marron. Asymptotic nonequivalence of some bandwidth selectors in nonparametric regression. *Biometrika*, 72(2):481–484, 1985. ISSN 00063444.

X. Huo and J.-C. Lu. A network flow approach in finding maximum likelihood estimate of high concentration regions. *Computational Statistics and Data Analysis*, 46:33–56, 05 2004. doi: 10.1016/S0167-9473(03)00134-8.

R. J. Hyndman. Computing and graphing highest density regions. *The American Statistician*, 50(2):120–126, 1996. ISSN 00031305.

H. Jankowski, X. Ji, and L. Stanberry. A random set approach to confidence regions with applications to the effective dose with combinations of agents. *Stat. Med.*, 33(24):4266–4278, 2014. ISSN 0277-6715. doi: 10.1002/sim.6226.

S. Karimzadeh, R. Bhopal, and H. Nguyen Tien. Review of infective dose, routes of transmission and outcome of covid-19 caused by the sars-cov-2: comparison with other respiratory viruses. *Epidemiology and Infection*, 149:e96, 2021. doi: 10.1017/S0950268821000790.

A. K. H. Kim and R. J. Samworth. Global rates of convergence in log-concave density estimation. *The Annals of Statistics*, 44(6):2756–2779, 12 2016. doi: 10.1214/16-AOS1480.

A. K. H. Kim, A. Guntuboyina, and R. J. Samworth. Adaptation in log-concave density estimation. *The Annals of Statistics*, 46(5):2279–2306, 10 2018. doi: 10.1214/17-AOS1619.

M. Kratz. Level crossings and other level functionals of stationary Gaussian processes. *Probability Surveys*, 3(none):230 – 288, 2006. doi: 10.1214/154957806000000087.

T. Laloë and R. Servien. Nonparametric estimation of regression level sets using kernel plug-in estimator. *Journal of the Korean Statistical Society*, 42(3):301 – 311, 2013. ISSN 1226-3192. doi: https://doi.org/10.1016/j.jkss.2012.10.001.

J. Li, C. Zhang, E. V. Nordheim, and C. E. Lehner. On the multivariate predictive distribution of multi-dimensional effective dose: a bayesian approach. *Journal of Statistical Computation and Simulation*, 78(5):429–442, 2008. doi: 10.1080/00949650601141712.

J. Li, C. Zhang, K. A. Doksum, and E. V. Nordheim. Simultaneous confidence intervals for semiparametric logistic regression and confidence regions for the multi-dimensional effective dose. *Statist. Sinica*, 20(2):637–659, 2010. ISSN 1017-0405.

V. Lushchak. Dissection of the hormetic curve: Analysis of components and mechanisms. *Dose-response : a publication of International Hormesis Society*, 12:466–79, 07 2014. doi: 10.2203/dose-response.13-051.Lushchak.

J. Marron and D. Nolan. Canonical kernels for density estimation. *Statistics & Probability Letters*, 7(3):195 – 199, 1988. ISSN 0167-7152. doi: https://doi.org/10.1016/0167-7152(88)90050-8.

J. S. Marron and M. P. Wand. Exact mean integrated squared error. *The Annals of Statistics*, 20(2):712–736, 06 1992a. doi: 10.1214/aos/1176348653.

J. S. Marron and M. P. Wand. Exact Mean Integrated Squared Error. *The Annals of Statistics*, 20(2):712 – 736, 1992b. doi: 10.1214/aos/1176348653.

M. Mason and W. Polonik. Asymptotic normality of plug-in level set estimates. *Annals of Applied Probability*, pages 1108–1142, 2009.

M. P. Mattson. Hormesis defined. *Ageing Research Reviews*, 7(1):1 – 7, 2008. ISSN 1568-1637. doi: https://doi.org/10.1016/j.arr.2007.08.007. Hormesis.

P. McCullagh and J. Nelder. *Generalized Linear Models, Second Edition*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1989. ISBN 9780412317606.

E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964. doi: 10.1137/1109020.

J. Navarro. A very simple proof of the multivariate Chebyshev's inequality. *Comm. Statist. Theory Methods*, 45(12):3458–3463, 2016. ISSN 0361-0926. doi: 10.1080/03610926.2013.873135.

J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972. doi: https://doi.org/10.2307/2344614.

J. K. Pal, M. Woodroofe, and M. Meyer. Estimating a polya frequency function. *Lecture Notes-Monograph Series*, 54:239–249, 2007. ISSN 07492170.

N. Picard and A. Bar-Hen. Estimation of the envelope of a point set with loose boundaries. *Applied Mathematics Letters*, 13(7):13–18, 2000. ISSN 0893-9659. doi: https://doi.org/10.1016/S0893-9659(00)00070-7.

N. Pya and S. Wood. Shape constrained additive models. *Statistics and Computing*, 2014, 02 2014. doi: 10.1007/s11222-013-9448-7.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL `http://www.R-project.org/`.

B. L. S. P. Rao. Estimation of a unimodal density. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 31(1):23–36, 1969. ISSN 0581572X.

C. R. Rao. A study of large sample test criteria through properties of efficient estimates: Part i: Tests for goodness of fit and contigency tables. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 23(1):25–40, 1961. ISSN 0581572X.

J. Rice. Bandwidth choice for nonparametric regression. *The Annals of Statistics*, 12(4): 1215–1230, 1984. ISSN 00905364.

K. Rufibach. Computing maximum likelihood estimators of a log-concave density function. *Journal of Statistical Computation and Simulation*, 77(7):561–574, 2007. doi: 10.1080/10629360600569097.

K. Rufibach and L. Dümbgen. *logcondens: Estimate a log-concave probability density from i.i.d. observations*, 2006. URL `https://CRAN.R-project.org/package=logcondens`. R package version 1.3.0.

D. Ruppert, S. J. Sheather, and M. P. Wand. An effective bandwidth selector for local least squares regression. *Journal of the American Statistical Association*, 90(432):1257–1270, 1995. ISSN 01621459.

R. J. Samworth and M. P. Wand. Asymptotics and optimal bandwidth selection for highest density region estimation. *Ann. Statist.*, 38(3):1767–1792, 2010. ISSN 0090-5364. doi: 10.1214/09-AOS766.

I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions: Part a.- on the problem of smoothing or graduation. a first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4(1):45–99, 1946. ISSN 0033569X, 15524485. URL `http://www.jstor.org/stable/43633538`.

C. Scott and M. Davenport. Regression level set estimation via cost-sensitive classification. *IEEE Transactions on Signal Processing*, 55(6):2752–2757, 2007. ISSN 1941-0476. doi: 10.1109/TSP.2007.893758.

C. Scott and R. Nowak. Minimax-optimal classification with dyadic decision trees. *IEEE Transactions on Information Theory*, 52(4):1335–1353, 2006. doi: 10.1109/TIT.2006.871056.

R. Shibata. An optimal selection of regression variables. *Biometrika*, 68(1):45–54, 1981. ISSN 00063444.

D. F. Signorini and M. C. Jones. Kernel estimators for univariate binary regression. *Journal of the American Statistical Association*, 99(465):119–126, 2004. doi: 10.1198/016214504000000115.

B. W. Silverman. On the estimation of a probability density function by the maximum penalized likelihood method. *The Annals of Statistics*, 10(3):795–810, 09 1982. doi: 10.1214/aos/1176345872.

B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall London ; New York, 1986. ISBN 0412246201.

M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*, pages 111–147, 1974.

S. Vogel and A. Schettler. A uniform concentration-of-measure inequality for multivariate kernel density estimators. 2013.

A. Wald. Note on the consistency of the maximum likelihood estimate. *Ann. Math. Statist.*, 20(4):595–601, 12 1949. doi: 10.1214/aoms/1177729952. URL `https://doi.org/10.1214/aoms/1177729952`.

G. Walther. Detecting the presence of mixing with multiscale maximum likelihood. *Journal of the American Statistical Association*, 97(458):508–513, 2002. doi: 10.1198/016214502760047032.

G. Walther. Inference and modeling with log-concave distributions. *Statist. Sci.*, 24(3): 319–327, 08 2009. doi: 10.1214/09-STS303. URL `https://doi.org/10.1214/09-STS303`.

M. P. Wand and M. C. Jones. Comparison of smoothing parameterizations in bivariate kernel density estimation. *Journal of the American Statistical Association*, 88(422):520–528, 1993. ISSN 01621459.

M. P. Wand and M. C. Jones. *Kernel Smoothing*. Number 60 in Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Chapman & Hall, Boca Raton, FL, U.S., December 1994. URL `http://oro.open.ac.uk/28198/`.

L. Wasserman. *All of Nonparametric Statistics (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387251456.

L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010. ISBN 1441923225.

G. S. Watson. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics*, 26(4): 359–372, 1964.

H. White. Maximum likelihood estimation of misspecified models. *Econometrica*, 50(1):1–25, 1982. ISSN 00129682, 14680262. URL `http://www.jstor.org/stable/1912526`.

R. M. Willett and R. D. Nowak. Minimax optimal level-set estimation. *IEEE Transactions on Image Processing*, 16(12):2965–2979, 2007. ISSN 1941-0042. doi: 10.1109/TIP.2007.910175.

S. N. Wood. Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, 99(467):673–686, 2004. ISSN 01621459.

D. E. Wright. A note on the construction of highest posterior density intervals. *Journal of the Royal Statistical Society Series C*, 35(1):49–53, 1986.

L. Y. Yang, Q. and Y. Zhang. Change point detection for nonparametric regression under strongly mixing process. *Statistical Papers*, 61(1):1465–1506, 2020. doi: 10.1007/s00362-020-01196-y.

K. Ziegler. On approximations to the bias of the Nadaraya-Watson regression estimator. *J. Nonparametr. Statist.*, 13(4):583–589, 2001. ISSN 1048-5252. doi: 10.1080/10485250108832866.