`lpcde`: Estimation and Inference for Local Polynomial Conditional Density Estimators

Matias D. Cattaneo* Rajita Chandak† Michael Jansson‡ Xinwei Ma§

March 7, 2025

Abstract

This paper discusses the `R` package `lpcde`, which stands for local polynomial conditional density estimation. It implements the kernel-based local polynomial smoothing methods introduced in Cattaneo et al. (2024a) for statistical estimation and inference of conditional distributions, densities, and derivatives thereof. The package offers mean square error optimal bandwidth selection and associated point estimators, as well as uncertainty quantification based on robust bias correction both pointwise (e.g., confidence intervals) and uniformly (e.g., confidence bands) over evaluation points. The methods implemented are boundary adaptive whenever the data is compactly supported. The package also implements regularized conditional density estimation methods, ensuring the resulting density estimate is non-negative and integrates to one. We contrast the functionalities of `lpcde` with existing open-source packages for conditional density estimation, and showcase its main features using simulated and real datasets. An abbreviated version of this article is published in Cattaneo et al. (2025).

1 Introduction

Conditional cumulative distribution functions (CDFs), conditional probability density functions (PDFs), and derivatives thereof, are important parameters of interest in statistics, econometrics, and other data science disciplines. This article discusses the main methodological features of the `R` package `lpcde` for estimation of and inference on conditional CDFs, conditional PDFs, and derivatives thereof, employing the kernel-based local polynomial smoothing approach introduced in Cattaneo et al. (2024a, CCJM hereafter).

Wand and Jones (1995), Fan and Gijbels (1996), Simonoff (2012), and Scott (2015) give textbook introductions to kernel-based density and local polynomial estimation and inference methods. The core idea underlying the estimator introduced in CCJM is to use kernel-based local polynomial smoothing methods to construct an automatically boundary adaptive estimator for CDFs, PDFs, and derivatives thereof. The estimation approach consists of two steps. The first step estimates the conditional distribution function using standard local polynomial regression methods, and the second step applies local polynomial smoothing to the (non-smooth) local polynomial conditional CDF estimate from the first step to obtain a smooth estimate of the CDF, PDF, and derivatives thereof.

*Department of Operations Research and Financial Engineering, Princeton University.
†Institute of Mathematics, École Polytechnique Fédérale de Lausanne
‡Department of Economics, UC Berkeley.
§Department of Economics, UC San Diego.

For the case of PDF estimation, classical estimation approaches typically employ ratios of unconditional kernel density estimators, the derivative of kernel-based non-linear distribution function regression estimators, or local polynomial estimators based on some preliminary density-like approximation. See, for example, Fan et al. (1996), Hall et al. (1999), De Gooijer and Zerom (2003), Hall et al. (2004), and references therein. These approaches are not boundary adaptive unless specific modifications (e.g., boundary corrected kernels) are introduced. CCJM's estimator is conceptually different and is boundary adaptive for a possibly unknown compact support of the data. Furthermore, the estimator has a simple closed form representation, which leads to easy and fast implementation. Unlike some other boundary adaptive procedures, it does not require pre-processing of data, and thus avoids the challenges of hyper-parameter tuning: only one bandwidth parameter needs to be selected for implementation.

Building on the theoretical and methodological work reported in CCJM, the package `lpcde` offers data-driven (pointwise and uniform) estimation and inference methods for conditional CDFs, conditional PDFs, and derivatives thereof, which are automatically valid at interior, near-boundary, and boundary points on the support of both the variable of interest and the conditioning variables. For point estimation, the package offers mean squared error optimal bandwidth selection and associated point estimators. For inference, the package offers valid confidence intervals and confidence bands based on robust bias-correction techniques (Calonico et al., 2018, 2022). Finally, these statistical procedures can be easily used for visualization and graphical presentation of smooth empirical CDFs, conditional PDFs, and derivative thereof. We give an overview of the main methods implemented in the package below, along with a discussion of more specific implementation issues. We also showcase the performance of the package with simulated data.

The package `lpcde` includes two main functions.

- `lpcde()`: This function implements the estimator of interest over a grid of evaluation points on the support of the variable of interest and at a pre-specified conditioning value. The function takes three main inputs: data, a bandwidth, and polynomial orders. When the bandwidth is not specified by the user, the function employs the companion function `lpbwcde()` for automatic, data-driven bandwidth selection. When the polynomial orders are not specified by the user, the function employs the next polynomial order relative to the parameter of interest. For example, for CDF estimation, the polynomial orders are set to $\mathtt{p} = \mathtt{q} = 1$, while for PDF estimation they are set to $\mathtt{p} = 2$ and $\mathtt{q} = 1$, where $\mathtt{p}$ denotes the polynomial order for the variable of interest, and $\mathtt{q}$ denotes the polynomial order for the conditioning variables. The Epanechnikov kernel is used by default. However, there are alternative kernel options that can be provided by the user, if desired. This function implements pointwise and uniform inference via robust bias-correction methods, employing the same grid of points used for point estimation.

- `lpbwcde()`: This function implements pointwise and integrated mean square error (IMSE) optimal bandwidth selection for the kernel-based local polynomial smoothing methods introduced in CCJM. The resulting bandwidth selection procedure leads to an IMSE-rate optimal point estimator whenever the difference of polynomial order and derivative order of interest is odd (see below for further details). This bandwidth choice is also valid, and in some cases optimal from a distributional approximation perspective, when coupled with robust bias-correction methods for statistical inference.

The methods `coef()`, `confint()`, `vcov()`, `print()`, `plot()` and `summary()` are supported for objects returned by the `lpcde` function, while the methods `coef()`, `print()` and `summary()` are supported for objects returned by the `lpbwcde` function. The `plot()` function builds on the

`ggplot2` (Wickham, 2016) package in `R` and can be used for illustrations of conditional CDFs, conditional PDFs or higher order derivatives and their pointwise or uniform confidence bands for a given value of the conditioning variable(s).

The package `lpcde` contributes to a rather small set of open source software packages for estimation and inference about conditional CDF, PDF, and derivatives thereof. More specifically, we identified two `R` packages, `hdrcde` (Hyndman et al., 2021), `haldensify` (Hejazi et al., 2022), and `np` (Hayfield and Racine, 2008), and one `Python` package, `cde` (Rothfuss et al., 2019), which provide related methodology. There are no open source `Stata` packages that implement conditional CDF, PDF, and derivative thereof estimation. Table 1 summarizes some of the main differences between those packages and `lpcde`. As is noted in the Table 1, `lpcde` is the only package available across multiple programming languages that provides both pointwise and uniform confidence interval construction that is asymptotically valid, in addition to producing mean square and uniform optimal boundary adaptive point estimates, with the option of ensuring proper conditional density estimates that are non-negative and integrate to one. These features are unique contributions of the package to the `R` toolkit and, more broadly, the open source statistical software community.

Table 1: Comparison of open source software packages for conditional density estimation.

Package	Programming language	CDF / Derivative estimation	Regularized density	Valid at boundary	Standard error	Valid inference	Confidence bands	Bandwidth selection
`hdrcde`	`R`	×	×	×	×	×	×	✓
`np`	`R`	×	×	×	✓	×	×	✓
`haldensify`	`R`	×	×	×	✓	×	×	✓
`cde`	`Python`	×	×	×	×	×	×	✓
`lpcde`	`R`	✓	✓	✓	✓	✓	✓	✓

Notes: (i) all packages provide conditional PDF point estimation; (ii) bandwidth selection is done via cross-validation in `hdrcde` and `np`, and using plug-in mean squared error approximations in `lpcde`.

In addition, Cattaneo et al. (2020, 2022, 2024b) develop complementary methods for local polynomial kernel based regression estimation and inference for *unconditional* densities and higher-order derivatives. These methods and companion statistical software (`lpdensity`) cannot be used to conduct estimation and inference for *conditional* distributions, densities, and derivative thereof.

The remainder of this article is organized as follows. Section 2 describes the derivation of our estimator along with details on how the bandwidth, covariance matrix and confidence intervals can be constructed. This section also highlights some key computational considerations when implementing the estimator. Section 3 discusses how the various functions and features of the package can be implemented in practice through examples of code snippets with a toy dataset. Section 4 illustrates the performance of the estimator through Monte Carlo simulation excercises and compares the performance of `lpcde` against the alternative packages identified in Table 1 on a real dataset. Finally, we conclude in Section 5. An abbreviated version of this article is published in Cattaneo et al. (2025), and additional information about the `R` package `lpcde`, including replication files and datasets, can be found at `https://nppackages.github.io/lpcde/`.

2 Methodology

We give an overview of the methodology implemented in `lpcde`; technical details in full generality can be found in CCJM. We start by considering a random sample $(Y_1, \mathbf{X}_1^\top), \ldots, (Y_n, \mathbf{X}_n^\top)$ from the continuously distributed random vector $(Y, \mathbf{X}^\top) \in \mathcal{Y} \times \mathcal{X}$. We assume $\mathcal{Y} \subseteq \mathbb{R}$ is a 1-dimensional

and $\mathcal{X} \subseteq \mathbb{R}^d$ is a d-dimensional possibly, but not necessarily, compactly supported set. The goal is to estimate and conduct inference on the conditional CDF, PDF, and derivatives thereof, of $Y|\mathbf{X}$. Prior to setting up our estimator, the following section establishes all necessary notation.

2.1 Notation

Notation introduced in this section will be used through the remainder of the text. Our parameter of interest is

$$F^{(\mu,\boldsymbol{\nu})}(y|\mathbf{x}) = \frac{\partial^{\mu+|\boldsymbol{\nu}|}}{\partial y^{\mu} \partial \mathbf{x}^{\boldsymbol{\nu}}} F(y|\mathbf{x}), \qquad F(y|\mathbf{x}) = \mathbb{P}[Y \leq y | \mathbf{X} = \mathbf{x}],$$

where $\mu \in \mathbb{N}_0$ denotes the derivative order with respect to the variable of interest Y and, employing multi-index notation, $\boldsymbol{\nu} \in \mathbb{N}_0^d$ denotes the multi-index for the corresponding derivatives of interest with respect to the conditioning variables $\mathbf{X}$. For example,

- $F(y|\mathbf{x}) = F^{(0,\mathbf{0})}(y|\mathbf{x})$ is the conditional CDF of $Y|\mathbf{X}$;
- $f(y|\mathbf{x}) = F^{(1,\mathbf{0})}(y|\mathbf{x})$ is the conditional PDF of $Y|\mathbf{X}$;
- $f^{(1,0)}(y|\mathbf{x}) = F^{(2,\mathbf{0})}(y|\mathbf{x})$ is the derivative (with respect to y) of conditional PDF of $Y|\mathbf{X}$.

To simplify the exposition, we abstract from derivative estimation with respect to the conditioning variables in $\mathbf{X}$, and therefore set $\boldsymbol{\nu} = \mathbf{0}$ for the rest of this article. Consequently, we denote

$$F^{(\mu)}(y|\mathbf{x}) = F^{(\mu,\mathbf{0})}(y|\mathbf{x}) \text{ for } \mu \in \mathbb{N}_0.$$

See CCJM for theoretical and methodological results concerning $|\boldsymbol{\nu}| > 0$, all of which are also implemented in the `R` package `lpcde`, thereby allowing for estimation of derivatives with respect to $\mathbf{X}$ of the conditional CDF of $Y|\mathbf{X}$.

The following notation is used in constructing and analysing our estimator:

- $\mathbf{e}_\ell$ is the conformable $(\ell+1)$-th unit vector.
- $|\mathcal{A}|$ denotes the cardinality of a set $\mathcal{A}$.
- $\mathbf{q}(\mathbf{u})$: $\mathfrak{q}$-th order polynomial expansion for some $\mathfrak{q} \in \mathbb{N}$. It is a $(\mathfrak{q}_d+1)$-dimensional vector collecting the ordered elements $\mathbf{u}^{\boldsymbol{\nu}}/\boldsymbol{\nu}!$ for $0 \leq |\boldsymbol{\nu}| \leq \mathfrak{q}$, where, employing multi-index notation, $\mathbf{u}^{\boldsymbol{\nu}} = u_1^{\nu_1} u_2^{\nu_2} \cdots u_d^{\nu_d}$, $\boldsymbol{\nu}! = \boldsymbol{\nu}_1! \boldsymbol{\nu}_2! \cdots \boldsymbol{\nu}_d!$, $|\boldsymbol{\nu}| = \nu_1 + \nu_2 + \cdots + \nu_d$, and $\mathfrak{q}_d = (d+\mathfrak{q})!/(\mathfrak{q}! d!) - 1$.
- $\mathbf{p}(u)$: $\mathfrak{p}$-th order polynomial expansion for some $\mathfrak{p} \in \mathbb{N}$. It is a $(\mathfrak{p}+1)$-dimensional vector collecting the ordered elements $u^{\mu}/\mu!$ for $0 \leq \mu \leq \mathfrak{p}$.
- $K_h(x;u) = K((x-u)/h)/h$, where $K(\cdot)$ is a kernel function and h is a bandwidth.
- $L_h(\mathbf{x};\mathbf{u}) = K_h(x_1-u_1)K_h(x_2-u_2)\cdots K_h(x_d-u_d)$.
- $\widehat{\mathbf{S}}_y = \frac{1}{n}\sum_{i=1}^n K_h(y_i;y)\mathbf{p}\Big(\frac{y_i-y}{h}\Big)\mathbf{p}\Big(\frac{y_i-y}{h}\Big)^\top$.
- $\widehat{\mathbf{S}}_{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^n L_h(\mathbf{x}_i;\mathbf{x})\mathbf{q}\Big(\frac{\mathbf{x}_i-\mathbf{x}}{h}\Big)\mathbf{q}\Big(\frac{\mathbf{x}_i-\mathbf{x}}{h}\Big)^\top$.
- $\widehat{\mathbf{R}}_{y,\mathbf{x}} = \frac{1}{n^2 h}\sum_{j=1}^n\sum_{i=1}^n K_h(y_j;y)\mathbf{p}\Big(\frac{y_j-y}{h}\Big)L_h(\mathbf{x}_i;\mathbf{x})\mathbf{q}\Big(\frac{\mathbf{x}_i-\mathbf{x}}{h}\Big)^\top \mathbb{1}(y_i \leq y_j)$.

2.2 General estimation idea

The construction of the conditional CDF, PDF and derivatives thereof involves two steps. First, the conditional distribution function $F(y|\mathbf{x})$ is estimated by standard local polynomial methods:

$$\widehat{F}_{\mathtt{q}}(y|x) = \mathbf{e}_0^\top \widehat{\boldsymbol{\gamma}}_{\mathtt{q}}(y|\mathbf{x}), \qquad \widehat{\boldsymbol{\gamma}}_{\mathtt{q}}(y|\mathbf{x}) = \underset{\mathbf{c}\in\mathbb{R}^{\mathtt{q}_d}}{\operatorname{argmin}} \sum_{i=1}^{n} \Big(\mathbb{1}(y_i \le y) - \mathbf{q}(\mathbf{x}_i - \mathbf{x})^\top \mathbf{c}\Big)^2 L_h(\mathbf{x}_i - \mathbf{x}), \tag{1}$$

Note here that the estimator $\widehat{F}_{\mathtt{q}}(y|x)$ of $F(y|\mathbf{x})$ is not smooth as a function of y and therefore cannot be used to construct an estimator of the conditional PDF and higher-order derivatives with respect to y.

Therefore, in a second step, a smoothed (with respect to y) estimator of the CDF and its derivatives is constructed also using local polynomial methods: for any $0 \le \mu \le \mathtt{p}$,

$$\begin{aligned}
&\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x}) = \mathbf{e}_\mu^\top \widehat{\boldsymbol{\beta}}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x}),\\
&\widehat{\boldsymbol{\beta}}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x}) = \underset{\mathbf{b}\in\mathcal{B}}{\operatorname{argmin}} \sum_{i=1}^{n} \Big(\widehat{F}_{\mathtt{q}}(y_i|\mathbf{x}) - \mathbf{p}(y_i - y)^\top \mathbf{b}\Big)^2 K_h(y_i - y),
\end{aligned} \tag{2}$$

where $\mathcal{B}$ is some general constraint set. There are different forms of search space that may be of interest to researchers based on the application for which the estimator is being used. For example, it may be necessary that the first element of $\mathbf{b}$, corresponding to the conditional PDF estimator, be nonnegative. In this setting, the set over which $\mathbf{b}$ is minimized can be defined as $\mathcal{B} = \{\mathbf{b} \in \mathbb{R}^{\mathtt{p}+1} : \mathbf{e}_1^\top \mathbf{b} \ge 0\}$. This case is studied further in Section 2.5. On the other hand, if $\mathcal{B} = \mathbb{R}^{\mathtt{p}+1}$, no constraints are imposed on $\mathbf{b}$. This is the case that we focus on in the following sections.

2.3 Point estimation

Solving Equations 1 and 2 (with $\mathcal{B} = \mathbb{R}^{\mathtt{p}+1}$) gives a simple closed form for the general estimator:

$$\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x}) = \mathbf{e}_\mu^\top \widehat{\mathbf{S}}_y^{-1} \widehat{\mathbf{R}}_{y,\mathbf{x}} \widehat{\mathbf{S}}_{\mathbf{x}}^{-1} \mathbf{e}_0, \tag{3}$$

A complete derivation of this closed-form solution is provided in the supplemental material of CCJM.

In the `R` package, for a choice of derivative μ with respect to y (and a choice of of derivative $\boldsymbol{\nu}$ with respect of $\mathbf{x}$, a choice of polynomial orders $(\mathtt{p},\mathtt{q})$, a choice of bandwidth h and kernel function $K(\cdot)$), the function `lpcde()` implements the estimator $\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x})$ over a grid of points on $\mathcal{Y}$ for a given conditioning evaluation point $\mathbf{x}$. By default, the function sets $(\mu,\boldsymbol{\nu}) = (1,0)$ (conditional PDF), $\mathtt{q} = 1$ (local linear nonsmooth conditional CDF estimation), $\mathtt{p} = 2$ (local quadratic smooth conditional CDF estimation), and $K(\cdot)$ is to chosen to be the Epanechnikov kernel. Generally speaking, it is recommended to choose the local polynomial order such that $\mathtt{p} - \mu$ and $\mathtt{q} - |\boldsymbol{\nu}|$ are both odd. Although the second-order Epanechnikov kernel is implemented by default, the function `lpcde()` can also be implemented with second-order uniform and triangular kernels by setting the variable `kernel_type` appropriately. The choice of the kernel does not affect the orders of the bias and the variance. Last but not least, the choice of bandwidth h is important: by default, whenever h is not supplied by the user, the function `lpcde()` relies on the companion function `lpbwcde()`, which implements data-driven bandwidth selection based on the minimization of the (approximate) mean squared error of the estimator $\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x})$.

In the remainder of this section we review some of the main statistical properties and inference techniques developed in CCJM and the computational considerations in implementing these methods in the package `lpcde`.

2.4 Bandwidth selection

Once we have the closed form of the point estimator, we can derive the leading bias and variance of the estimator. The leading bias and variance for odd values of $\mathfrak{p}-\mu$ and $\mathfrak{q}-|\boldsymbol{\nu}|$ take the following form:

$$\text{Bias}\left[\widehat{F}^{(\mu)}(y|\mathbf{x})\right] = h^{\mathfrak{q}+1}\sum_{|\mathbf{m}|=\mathfrak{q}+1} F^{(\mu,\mathbf{m})}(y|\mathbf{x})B^{(i)}_{\mathbf{m}}(\mathbf{x}) + h^{\mathfrak{p}+1-\mu}F^{(\mathfrak{p}+1)}B^{(ii)}_{\mathfrak{p}+1}(y), \tag{4}$$

$$\text{Var}\left[\widehat{F}^{(\mu)}(y|\mathbf{x})\right] = \frac{1}{nh^{d+2\mu+1}}F^{(1)}(y|\mathbf{x})V^{(\mu)}_{\mathfrak{p},\mathfrak{q}}(y,\mathbf{x}). \tag{5}$$

The quantities on the right hand side above implicitly depend on the kernel function. It is straightforward to show that both the bias and variance terms converge in probability to non-random, well-defined limits. Exact expressions and technical details for other cases can be found in the supplemental appendix of CCJM.

Equations 4 and 5 are valid for all evaluation points on the support of the data. As a result, the pointwise mean squared error (MSE) optimal bandwidth can be approximated as

$$h^{\text{MSE}}_{\mathfrak{p},\mathfrak{q}}(y,\mathbf{x}) = \underset{h>0}{\text{argmin}}\left[\text{Var}\left[\widehat{F}^{(\mu)}_{\mathfrak{p},\mathfrak{q}}(y|\mathbf{x})\right] + \text{Bias}\left[\widehat{F}^{(\mu)}_{\mathfrak{p},\mathfrak{q}}(y|\mathbf{x})\right]^2\right].$$

Under standard regularity conditions, $h^{\text{MSE}}(y,\mathbf{x})$ is MSE-optimal if $\mathfrak{p}-\mu$ and $\mathfrak{q}-|\boldsymbol{\nu}|$ are odd. Precise closed-form expressions for the MSE-optimal bandwidth can be found in the supplemental appendix of CCJM. In practice, the MSE-optimal bandwidth is estimated by plugging-in estimates of the unknown quantities in Equations 4 and 5, given some initial bandwidth choice and then direcetly solving for the optimal bandwidth.

The IMSE-optimal bandwidth is estimated similarly, with the main difference being that a set of grid points on the support of $\mathcal{Y}$ is used to approximate the integral. Detailed expressions are given in the supplemental material of CCJM. Bandwidth selection is implemented through the `lpbwcde()` function.

The number of grid points or specific locations of grid points (default is 19 equally-spaced points over the implied support) can be specified by the user as an input to both the `lpbwcde()` and `lpcde()` functions. For generating quantile-spaced grid points, the flag `grid_spacing` should be set to `'quantile'`. Users should be aware of possible issues with using equally-spaced grid points at low-density regions or near boundary points. If a small banwidth is coupled with grid points that have few data points that can be used for estimation, the resulting point estimates as well as standard error approximations may have numerical inaccuracies that cause instability in the output. We recommend either prior checking of effective sample sizes for the choice of bandwidth or choosing quantile-spaced grid points.

2.5 Constrained density estimation

As mentioned in Section 2.2, some applications may require that the conditional density estimate satisfy additional constraints. For example, it may be desirable to ensure that the PDF estimate be non-negative on the support and integrates to one. Fortunately, our two-step formulation of the estimator allows for the non-negativity constraint to be incorporated directly into the second step given in Equation 2:

$$\widehat{f}_{\mathtt{N}}(y|\mathbf{x}) = \mathbf{e}_1^\top\widehat{\boldsymbol{\beta}}_{\mathtt{N}}(y|\mathbf{x}), \qquad \widehat{\boldsymbol{\beta}}_{\mathtt{N}}(y|\mathbf{x}) = \underset{\mathbf{b}\in\mathbb{R}^{\mathfrak{p}+1}:\ \mathbf{e}_1^\top\mathbf{u}\geq 0}{\text{argmin}}\sum_{i=1}^{n}\left(\widehat{F}(y_i|\mathbf{x}) - \mathbf{p}(y_i-y)^\top\mathbf{b}\right)^2 K_h(y_i;y),$$

where we use the subscript "$\mathtt{N}$" to denote the non-negative estimator. The solution to this modified optimization problem leads to a simple closed form solution that can be written in terms of the unconstrained estimator $\widehat{f}(y|\mathbf{x})$:

$$\widehat{f}_{\mathtt{N}}(y|\mathbf{x}) = \max\left\{\widehat{f}(y|\mathbf{x})\ ,\ 0\right\}.$$

In order to incorporate the constraint that the conditional density estimator also integrates to one, a global constraint must be imposed on the estimator. In CCJM, we propose and study a modification of the $\widehat{f}_{\mathtt{N}}$ based on the Kullback-Leibler divergence,

$$\widehat{f}_{\mathtt{I}}(y|\mathbf{x}) = \underset{g\in\mathcal{G}}{\operatorname{argmin}}\, \mathrm{KL}\big(g \parallel \widehat{f}_{\mathtt{N}}(\cdot|\mathbf{x})\big), \qquad \text{where } \mathrm{KL}(g \parallel f) = \int_{\mathcal{Y}} g(y)\log\left(\frac{g(y)}{f(y)}\right)\mathrm{d}y,$$

where the subscript "$\mathtt{I}$" stands for "integrating to one" and $\mathcal{G} = \{g \geq 0 : \int_{\mathcal{Y}} g(y)\mathrm{d}y = 1,\ g(y) = 0 \text{ for } y \notin \mathcal{Y}\}$.

Fortunately, $\widehat{f}_{\mathtt{I}}$ can be written in closed form as

$$\widehat{f}_{\mathtt{I}}(y|\mathbf{x}) = \frac{\widehat{f}_{\mathtt{N}}(y|\mathbf{x})}{\int_{\mathcal{Y}} \widehat{f}_{\mathtt{N}}(u|\mathbf{x})\mathrm{d}u}$$

Uniform rates of convergence as well as distributional convergence of both constrained estimators can be established with slight modifications from the theory that was established for the unconstrained estimator. Crucially, this means we can construct robust bias-corrected uniform confidence bands for the constrained estimators as well. Further details regarding the convergence guarantees of the constrained estimators are provided in Section 4 of CCJM.

2.6 Distribution theory and robust bias-corrected inference

In order to conduct inference, we first construct a Wald-type test statistic that has the following distributional convergence

$$T_{\mathtt{p},\mathtt{q}}(y,\mathbf{x}) = \frac{\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x}) - F^{(\mu)}(y|\mathbf{x})}{\sqrt{\mathrm{Var}\left[\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x})\right]}} \rightsquigarrow \mathcal{N}(B,1),$$

where $\rightsquigarrow$ denotes weak (distributional) convergence as $h \to 0$ and $n \to \infty$, $\mathcal{N}$ denotes the Gaussian distribution, and B denotes the standardized asymptotic bias emerging whenever a too "large" bandwidth is employed (e.g., when the MSE-optimal or IMSE-optimal bandwidth is used). See CCJM for details.

As a result, standard confidence intervals with nominal $(1-\alpha)$ coverage takes the form:

$$\mathrm{CI}(y,\mathbf{x}) = \left[\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x}) \pm z_{1-\alpha/2}\sqrt{\widehat{\mathrm{Var}}\left[\widehat{F}^{(\mu)}_{\mathtt{p},\mathtt{q}}(y|\mathbf{x})\right]}\right],$$

where z_α is the α-th quantile of the standard normal distribution. However, for "large" bandwidths, this confidence interval would be invalid due to the asymptotic bias, B. In practice, undersmoothing is often used to address the asymptotic bias present. However, Calonico et al. (2018, 2022) show that undersmoothing is sub-optimal under the standard assumptions of the model. Instead, they propose a robust bias-correction (RBC) technique that has better higher-order approximations and asymptotically correct coverage probabilities. RBC requires bias-correction of the point estimator

and then adjusting the variance estimate appropriately to construct a bias-corrected Wald-type statistic.

For our estimator, we first correct for the first-order bias by using a point estimator that is generated by increasing the polynomial order for both variables, y and $\mathbf{x}$. To be specific, we use $\widehat{F}^{(\mu)}_{\mathfrak{p}+1,\mathfrak{q}+1}(y|\mathbf{x}; h^{\text{MSE}}_{\mathfrak{p},\mathfrak{q}})$ in place of $\widehat{F}^{(\mu)}_{p,q}(y|\mathbf{x}; h^{\text{MSE}}_{p,q})$. The bandwidth used is optimal for the point estimate with the lower order polynomials. The asymptotically valid confidence intervals now take the form

$$\text{CI}^{\text{RBC}}(y,\mathbf{x}) = \left[\widehat{F}^{(\mu)}_{\text{RBC}}(y|\mathbf{x}) \pm z_{1-\alpha/2}\sqrt{\widehat{\text{Var}}\left[\widehat{F}^{(\mu)}_{\text{RBC}}(y|\mathbf{x})\right]}\right],$$

where $\widehat{F}^{(\mu)}_{\text{RBC}}(y|\mathbf{x}) \equiv \widehat{F}^{(\mu)}_{\mathfrak{p}+1,\mathfrak{q}+1}(y|\mathbf{x}) = \widehat{F}^{(\mu)}_{\mathfrak{p},\mathfrak{q}}(y|\mathbf{x}) - \widehat{\text{Bias}}\left[\widehat{F}^{(\mu)}_{\mathfrak{p},\mathfrak{q}}(y|\mathbf{x})\right]$.

Additionally, uniform confidence bands can be constructed as

$$\text{CB}^{\text{RBC}}(\mathcal{M}) = \left\{\left[\widehat{F}^{(\mu)}_{\text{RBC}}(y|\mathbf{x}) \pm z_{\mathcal{M},1-\alpha/2}\sqrt{\widehat{\text{Var}}\left[\widehat{F}^{(\mu)}_{\text{RBC}}(y|\mathbf{x})\right]}\right], \quad y \in \mathcal{M}\right\},$$

where $\mathcal{M}$ is a collection of evaluation points on the support $\mathcal{Y}$ and $z_{\mathcal{M},\alpha}$ is the α-quantile over the collection of evaluation points for a normal distribution centered at 0 and with the same variance-covariance matrix as the estimator. The critical value $z_{\mathcal{M},1-\alpha/2}$, is defined by the upper α quantile of the supremum of the simulated Gaussian process on the grid $\mathcal{M}$:

$$z_{\mathcal{M},\alpha} = \inf\left\{u \geq 0 : \ \mathbb{P}\left[\sup_{y\in\mathcal{M}} |\widehat{\mathcal{Z}}^{(\mu)}(y|\mathbf{x})| \leq u \,\middle|\, \text{Data}\right] \geq 1-\alpha\right\},$$

where $\widehat{Z}^{(\mu)}(y|\mathbf{x}) \overset{a}{\sim} \mathcal{N}\Big(0, \widehat{\text{Cov}}\big[\widehat{F}^{(\mu)}_{\text{RBC}}(y|\mathbf{x})\big]\Big)$. The confidence band depends on the entire collection of evaluation points. In `lpcde`, $z_{\mathcal{M},1-\alpha/2}$, is estimated by using the maximum over the grid points as an approximation for the supremum over $\mathcal{M}$. See CCJM (and its supplemental) for technical details and regularity conditions.

The RBC method leads to confidence intervals/bands that are not centered at the density point estimates since different order polynomials are used for the point estimates and for inference. Thus, it may happen that the point estimates lies outside of the RBC confidence intervals/bands if the underlying distribution has high curvature at some evaluation point(s). One solution in this case is to increase the polynomial orders $\mathfrak{p}$ and $\mathfrak{q}$, or to use a smaller-than-optimal bandwidth.

2.7 Implementation of the covariance estimator

Implementing the variance estimator for both estimating the MSE-optimal bandwidth and constructing confidence intervals, requires careful consideration. It is particularly crucial to consider the computational cost of estimating the variance when employing uniform confidence bands, which requires the construction of the full $|\mathcal{M}|\times|\mathcal{M}|$ covariance matrix in order to approximate the critical value $z_{\mathcal{M},1-\alpha/2}$. The discussion in this section focuses only on the covariance matrix estimation for the conditional PDF, $\widehat{f}(y|\mathbf{x})$, purely for simplicity of presentation.

The standard plug-in estimator (constructed by estimating the unknown quantities in Equation 5) that is proposed and studied in CCJM has a computational complexity of $O(|\mathcal{M}|^2 n^4)$. For large datasets and a fine grid of evaluation points, this is prohibitively slow to run in practice. As

a result the default covariance estimator used in `lpcde` implements a significantly faster jackknife covariance estimatior. The construction of the jackknife estimator relies on the fact that the closed-form of the estimator $\widehat{f}$ can be written as a V-statistic to which the Hoeffding decomposition can be applied. The covariance expression then decomposes to a sum of two independent functions that depend on the evaluation points and a small subset of the data in the neighborhood of the evaluation points, thus reducing the computational cost to only $O(|\mathcal{M}|^2(nh)^2)$.

Here we provide a sketch of how this estimator is constructed. The interested reader can find a complete derivation in Section 6 of the Supplementary Material of CCJM.

We start by first observing that the estimator $\hat{f}(y|\mathbf{x})$ is a V-statistic:

$$\begin{aligned}\hat{f}(y|\mathbf{x}) &= \frac{1}{n^2 h}\sum_{i,j}\mathbb{1}(y_i \le y_j)\mathbf{e}_1^\top \hat{\mathbf{S}}_y^{-1}\mathbf{p}\left(\frac{y_j - y}{h}\right)K_h\left(y_j; y\right)\mathbf{q}^\top\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)L_h\left(\mathbf{x}_i; \mathbf{x}\right)\hat{\mathbf{S}}_{\mathbf{x}}^{-1}\mathbf{e}_0 \\ &= \frac{1}{n^2}\sum_{i=1}^{n} a(y_i, y)b(\mathbf{x}_i, \mathbf{x}) + \frac{1}{n^2}\sum_{1\le i\ne j\le n}\mathbb{1}(y_i \le y_j)a(y_j, y)b(\mathbf{x}_i, \mathbf{x}), \end{aligned} \tag{6}$$

where

$$\begin{aligned} a(y_i, y) &= h^{-2}\mathbf{e}_1^\top \hat{\mathbf{S}}_y^{-1}\mathbf{p}\left(\frac{y_i - y}{h}\right)K\left(\frac{y_i - y}{h}\right), \\ b(\mathbf{x}_i, \mathbf{x}) &= h^{-d}\mathbf{e}_0^\top \hat{\mathbf{S}}_{\mathbf{x}}^{-1}\mathbf{Q}\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right)L\left(\frac{\mathbf{x}_i - \mathbf{x}}{h}\right). \end{aligned}$$

The scalar functions $a(\cdot)$ and $b(\cdot)$ are are non-zero only for data points that are within h distance of the evaluation point, a feature that the package `lpcde` leverages explicitly to improve numerical performance in applications. The second term in Equation 6 can now be symmetrized and treated as a U-statistic. Then, the Hoeffding decomposition can be applied and plugged back into Equation 6. This leads to a natural alternative jackknife covariance estimator, which is simple to write and computationally efficient:

$$\hat{\mathsf{C}}(y, \mathbf{x}, y', \mathbf{x}') = \frac{1}{n}\sum_{i=1}^{n}\hat{L}_{(i)}(y, \mathbf{x})\hat{L}_{(i)}(y', \mathbf{x}').$$

where

$$\hat{L}_{(i)}(y, \mathbf{x}) = \frac{2}{n-1}\sum_{j\ne i}\Big(u_{i,j} - \hat{f}(y|\mathbf{x})\Big).$$

and $u_{i,j} = \frac{1}{2}(\mathbb{1}(y_i \le y_j)a(y_j, y)b(\mathbf{x}_i, \mathbf{x}) + \mathbb{1}(y_j \le y_i)a(y_i, y)b(\mathbf{x}_i, \mathbf{x}))/2$. In particular, if the two evaluation points are equivalent, then we return the (approximately jackknife) variance estimator

$$\hat{\mathsf{V}}(y, \mathbf{x}) \equiv \hat{\mathsf{C}}(y, \mathbf{x}, y, \mathbf{x}) = \frac{1}{n-1}\sum_{i=1}^{n}\hat{L}_{(i)}^2(y, \mathbf{x}).$$

It can be easily verified that this jackknife covariance estimator is asymptotically equivalent to the theoretical variance expression in Equation 5.

3 Implementation

In this section we discuss how each of the functions provided in `lpcde` can be used with the aid of code snippets on a simulated dataset. We consider a bi-variate jointly normal data generating process with mean 0 and variance 1.

3.1 Density estimation

The function `lpcde()` provides information on point estimates, standard errors and confidence interval or bands for a given value of $\mathbf{x}$ over a range of grid points for y. If the grid points are not provided by the user, the function chooses nineteen equally-spaced grid points over the implied support of the data and, if no bandwidth is provided, computes the rule-of-thumb MSE bandwidth at each point.

The following example estimates the conditional density at $\mathbf{x} = 0$, with a fixed bandwidth of 1, using the default local polynomial approximation $\mathfrak{p} = 2,\ \mathfrak{q} = 1$. RBC confidence intervals over the grid are also computed, in this case using the default polynomial orders $\mathfrak{p} = 3,\ \mathfrak{q} = 2$.

```
R> set.seed(42)
R> n = 1000
R> x_data = as.matrix(stats::rnorm(n, mean = 0, sd = 1))
R> y_data = as.matrix(stats::rnorm(n, mean = x_data, sd = 1))
R> y_grid = seq(from = -2, to = 2, length.out = 10)
R> model1 = lpcde(y_data = y_data, x_data = x_data, y_grid = y_grid, x = 0,
+    bw = 1, rbc = TRUE)
R> summary(model1)
```

The function returns an object of type `lpcde`. Standard `R` methods, `coef()`, `confint()`, `vcov()`, `print()`, `plot()` and `summary()`, can be used on objects of type `lpcde` to understand the output.

Below we reproduce the output of running the `summary` command on `model1`. The first part of the summary output provides basic information about some of the options specified to the function. The second part provides relevant information for each point estimate generated in a table with 7 columns, (i) grid evaluation points, (ii) bandwidth used at each point, (iii) effective number of data points used to generate the point estimate, (iv) point estimate, (v) standard error, (vi) lower $(1-\alpha)$-confidence interval, and, (vii) upper $(1-\alpha)$-confidence interval.

```
Call: lpcde

Sample size                                              1000
Polynomial order for Y point estimation      (p=)        2
Polynomial order for X point estimation      (q=)        1
Density function estimated                   (mu=)       1
Order of derivative estimated for covariates (nu=)       0
Kernel function                                          epanechnikov
Bandwidth method

===============================================================================
                                    Point       Std.       Robust B.C.
Index     Grid      B.W.    Eff.n     Est.      Error      [ 95\% C.I. ]
===============================================================================
1      -2.0000    1.0000      132    0.0768    0.0126     0.0043 ,  0.1108
2      -1.5556    1.0000      211    0.1446    0.0089     0.0834 ,  0.1693
3      -1.1111    1.0000      304    0.2255    0.0064     0.2135 ,  0.2829
4      -0.6667    1.0000      370    0.2982    0.0051     0.2964 ,  0.3584
5      -0.2222    1.0000      411    0.3407    0.0047     0.3410 ,  0.3981
-------------------------------------------------------------------------------
```

```
6       0.2222    1.0000     409    0.3397    0.0044     0.3576 ,  0.4113
7       0.6667    1.0000     359    0.2958    0.0050     0.2874 ,  0.3447
8       1.1111    1.0000     279    0.2229    0.0064     0.1916 ,  0.2574
9       1.5556    1.0000     186    0.1390    0.0082     0.0818 ,  0.1656
10      2.0000    1.0000     117    0.0663    0.0112    -0.0291 ,  0.0886
--------------------------------------------------------------------------
==========================================================================
```

By default, the function provides estimates according to the original formulation of the estimator $\widehat{f}(y|\mathbf{x})$. If a constrained density estimate that is non-negative and integrates to one ($\widehat{f}_{\mathtt{I}}$ as defined in Section 2.5) is desired, the flags `nonneg` and `normalize` can be turned on.

```
R> model_reg = lpcde(y_data = y_data, x_data = x_data, y_grid = y_grid,
+     x = 0, bw = 1, nonneg = TRUE, normalize = TRUE)
```

Figure 1 shows how the estimates differ when the additional constraints are imposed:

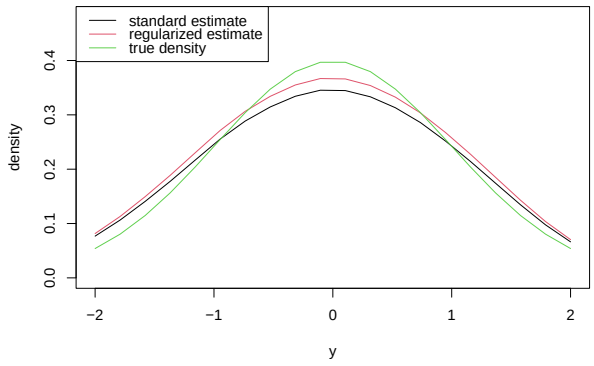


Figure 1: Comparing standard density estimate with normalized estimate.

3.2 Out-of-sample prediction

The `lpcde` function can be directly used for prediction on a new dataset. Below is a simple illustration of how a researcher may want to implement this.

Suppose we want to use some data to *train* the estimator and then we would like to *test* it on an unseen dataset. In this case, a common method is to randomly subset the data into training and testing. We assume the same simulation set up as in the previous section. The data is split with 95% for training and 5% for testing

```
R> sample = sample(c(TRUE, FALSE), nrow(y_data), replace=TRUE, prob=c(0.95,0.05))
R> y_train  = y_data[sample, ]
R> x_train = x_data[sample, ]
R> y_test   = y_data[!sample, ]
```

Now the `y_test` sample can be used directly as the grid of evaluation points for `lpcde`:

```
R> prediction_model = lpcde::lpcde(x_data=x_train, y_data=y_train,
+    y_grid=y_test, x=0.5, bw=0.5, cov_flag="off")
```

3.3 Covariance estimation

As noted in Section 2.7, estimating the full covariance matrix can be computationally intensive. In order to allow full flexibility in application of this functionality, an optional input `cov_flag` to the `lpcde` function can be used. This input can take on three different values:

(a) `"full"`: the function will compute the entire covariance matrix (and therefore allow confidence interval and band construction),

(b) `"diag"`: this will only compute the diagonal entries of the covariance matrix (i.e. the standard errors, only pointwise confidence intervals can be computed), and

(c) `"off"`: no entries of the covariance matrix are estimated. Inference tools will be unavailable.

3.4 Plotting

The `plot()` function uses the `ggplot2` package with objects of type `lpcde` to produce illustrations of point estimates and confidence intervals and/or bands. A simple plot of the conditional PDF with 95% confidence intervals can be generated by running the following code.

```
R> plot(model1, CIuniform = TRUE, rbc = TRUE, xlabel = "y")
```

This code snippet produces an image of the type shown in Figure 2.

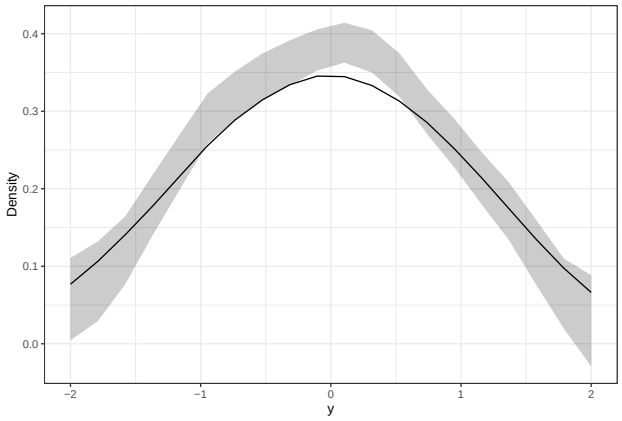


Figure 2: A simple density plot with robust 95% confidence bands.

By default the `plot()` function plots pointwise confidence intervals at 95% level with the point estimates. Additional options for confidence levels, bands and RBC inference are detailed in the package manual. Editing other visual aspects of the plots can be done by providing standard inputs to `ggplot2` functions.

3.5 Bandwidth selection

`lpbwcde()` implements the rule-of-thumb MSE- and IMSE- bandwidth selection by implementing the formulae provided in Section 2.3.

By default `lpbwcde()` computes the rule-of-thumb MSE optimal bandwidth for the conditional PDF with locally quadratic polynomial in y and locally linear polynomial in $\mathbf{x}$ and Epanechnikov

kernel on nineteen equally-spaced grid points on the implied support of $\mathcal{Y}$ determined by the observed data. The output of this function is similar to that of `lpcde()` and provides basic information for the data and options specified. The summary of objects returned by this function additionally provides a table with three columns: (i) `y_grid`: values of the grid points for which the bandwidth is estimated, (ii) `B.W.`: the estimated bandwidth corresponding to each grid point, and (iii) `Eff.n.`: the number of effective data points at each evaluation point given the estimated bandwidth. An example of standard bandwidth selection is provided in the following output.

```
R> model2 = lpbwcde(y_data = y_data, x_data = x_data, x = 0,
+    y_grid = y_grid)
R> summary(model2)

Call: lpbwcde

Sample size                                              1000
Polynomial order for Y point estimation      (p=)        2
Polynomial order for X point estimation      (q=)        1
Density function estimated                   (mu=)       1
Order of derivative estimated for covariates (nu=)       0
Kernel function                                          epanechnikov
Bandwidth method                                         mse-rot

==================================
Index     y_grid      B.W.   Eff.n
==================================
1      -2.0000    1.0250      76
2      -1.5556    1.1594     238
3      -1.1111    2.0298     808
4      -0.6667    1.2968     615
5      -0.2222    1.0609     560
----------------------------------
6       0.2222    1.0634     558
7       0.6667    1.3103     607
8       1.1111    1.9603     774
9       1.5556    1.1566     219
10      2.0000    1.0274      71
----------------------------------
==================================
```

The estimated bandwidth from this function can be used as bandwidth input to `lpcde()` directly by using the option of `bwselect` to specify bandwidth selection type instead of running `lpbwcde()` first.

4 Computational performance

In this section we demonstrate the performance of the `lpcde` package. We start with a simulated dataset analysis to showcase each of the inference features. Then, we compare the performance of our package with the existing conditional density estimators in `R` (as identified in Table 1) on the Iris dataset.

4.1 Simulations

In this section we illustrate the effectiveness of our estimator with a Monte Carlo study. For the sake of simplicity, we set $d = 1$ and assume that $\mathbf{x}$ and y are simulated by a joint normal distribution truncated on $[-1.5, 1.5]^2$. We simulate 100 data sets of 2000 independent samples each. The point estimates are generated at three distinct values that are characterized by their location (a) interior (0), (b) near-boundary (0.8), and (c) at-boundary (1.5) relative to the implied boundary of the data.

For each conditional value, we present the average bandwidth, average bias, standard deviation, 95% coverage, and width of the confidence intervals across the simulated datasets. We present these results for both the standard estimate (rows "WBC") which is generated with a quadratic polynomial ($\mathfrak{p} = 2$) with respect to the variable y, and linear polynomial ($\mathfrak{q} = 1$) with respect to the variable $\mathbf{x}$, as well as the robust bias-corrected estimates (rows "RBC") which uses cubic polynomial ($\mathfrak{p} = 3$) for y and quadratic polynomial ($\mathfrak{q} = 2$) for $\mathbf{x}$.

Table 2 presents the results of this simulated study. The first four columns of the table present average pointwise MSE-optimal bandwidth used in estimation ($\widehat{h}_{\text{MSE}}$), bias, standard error (SE) and root mean squared-error (RMSE). The last four columns are the average pointwise confidence interval coverage and width (AW) of the confidence interval for the standard estimate and inference method ("WBC") and robust bias-corrected estimate and inference ("RBC").

					Coverage		AW	
Eval. point	$\widehat{h}_{\text{MSE}}$	Bias	SE	RMSE	WBC	RBC	WBC	RBC
$\mathbf{x} = 0$								
$y = 0$	0.48	0.01	0.02	0.03	70	92	0.07	0.22
$y = 0.8$	0.55	0.01	0.01	0.02	79	94	0.05	0.17
$y = 1.5$	0.78	0.01	0.01	0.02	56	95	0.03	0.08
$\mathbf{x} = 0.8$								
$y = 0$	0.65	0.01	0.01	0.02	78	94	0.05	0.16
$y = 0.8$	0.60	0.02	0.01	0.03	53	96	0.06	0.19
$y = 1.5$	0.68	0.01	0.01	0.02	75	94	0.05	0.15
$\mathbf{x} = 1.5$								
$y = 0$	1.00	0.02	0.01	0.02	49	92	0.04	0.12
$y = 0.8$	0.90	0.01	0.01	0.03	73	93	0.05	0.15
$y = 1.5$	0.90	0.04	0.02	0.05	29	95	0.06	0.17

Table 2: Pointwise results
WBC: without bias-correction, RBC: robust bias-corrected.

Note that robust bias-corrected inference produces accurate empirical coverage across all pointwise combinations. As such, we recommend users employ robust bias-corrected estimates for improved reliability of results.

Next, we test the bandwidth selection by simulating point estimation and coverage at varying bandwidth values. We choose the range of bandwidth values to be between 0.5 and 1.3 times the average MSE-optimal bandwidth ($\widehat{h}_{\text{MSE}}$). Table 3 presents the average bias, standard error (SE), root mean-squared error (RMSE), pointwise coverage rate (CR) and average width of confidence intervals (AW) for 100 simulations at the point $y = 0,\ \mathbf{x} = 0$.

$\times\widehat{h}_{\text{MSE}}$	$\widehat{h}$	Bias	SE	RMSE	WBC CR	RBC CR	WBC AW	RBC AW
0.5	0.24	0.00	0.13	0.15	100.00	100.00	0.53	1.76
0.6	0.29	0.01	0.08	0.09	100.00	100.00	0.30	1.01
0.7	0.34	0.01	0.05	0.06	97.00	100.00	0.19	0.64
0.8	0.38	0.01	0.03	0.05	89.00	100.00	0.13	0.43
0.9	0.43	0.01	0.02	0.04	80.00	98.00	0.09	0.30
1	0.48	0.01	0.02	0.03	70.00	92.00	0.07	0.22
1.1	0.53	0.02	0.01	0.03	57.00	85.00	0.05	0.17
1.2	0.58	0.02	0.01	0.03	40.00	76.00	0.04	0.13
1.3	0.62	0.02	0.01	0.03	28.00	72.00	0.03	0.10
1.4	0.67	0.03	0.01	0.03	17.00	68.00	0.03	0.08
1.5	0.72	0.03	0.01	0.03	7.00	64.00	0.02	0.07

Table 3: Bandwidth selection at interior point $(y = 0, \mathbf{x} = 0)$.
WBC: without bias-correction, RBC: robust bias-corrected.

4.2 Comparative analysis

We now turn to comparing the performance of `lpcde` against the other open source `R` packages available at the time of writing this article. To compare the performance of these packages, we consider the Iris dataset which is available as a default dataset in `R`. For this study, we estimate the distribution of the *Sepal length* feature conditional on the *Petal length* feature. The conditioning values are chosen based on the 25-th (1.6), 50-th (4.35) and 75-th (5.1) quantiles of the *Petal length*. Figure 3 shows a scatter plot of the data with vertical lines denoting the conditional values at which the *Sepal length* density will be estimated.

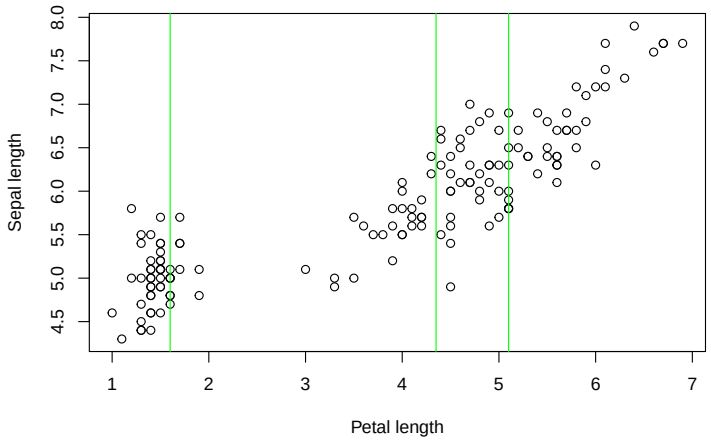
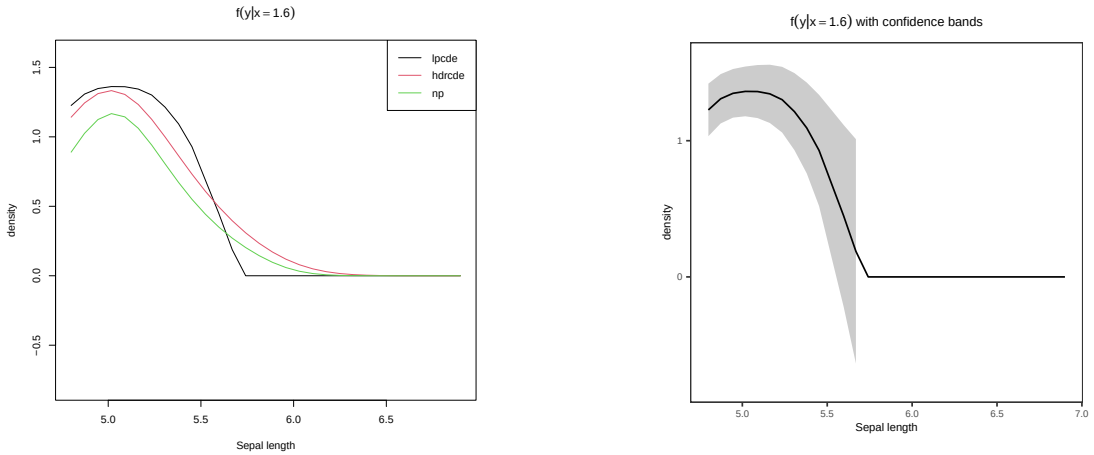


Figure 3: Scatter plot with conditioning values.



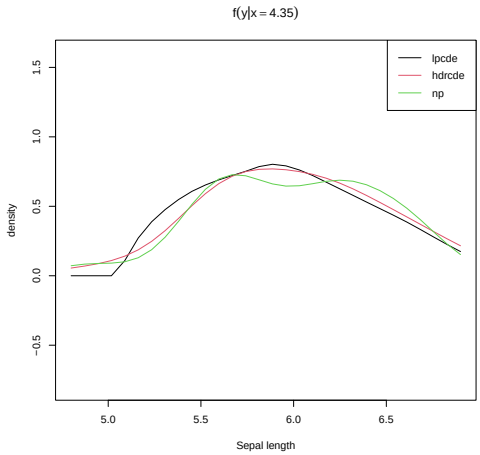
(a) Conditional density estimates.

(b) `lpcde` estimate with uniform confidence bands.

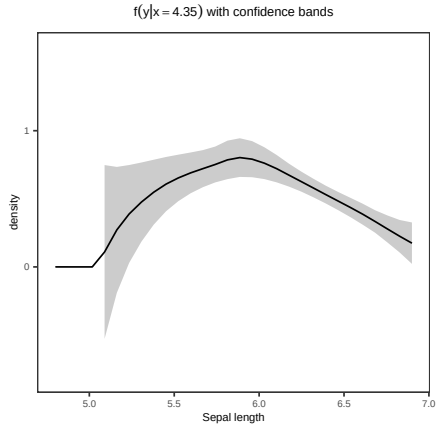
Figure 4: Conditional density estimates from each implementation conditioning at *Petal length* = 1.6.

From Figure 3, it is clear that the conditional expectation of the *Sepal length* shifts across the three evaluation points. Furthermore, at the conditional value of 1.6, there are very few data points in a reasonable neighbourhood that can be used to construct the estimates. We expect this to affect the standard error and resulting confidence intervals.

We now plot the conditional distributions of each of the three estimators. Since `lpcde` is the only package that provides confidence interval construction, we additionally plot the `lpcde` estimate with the pointwise confidence intervals. Note that the confidence intervals are only constructed for estimates that are generated with more that 15 data points as we believe standard errors on estimates generated with fewer data points will be unreliable. Figures 4, 5 and 6 show the estimated densities for condtioning at 1.6, 4.35 and 5.1, respectively. Figures 4a, 5a, 6a compare directly the estimates generated by the default implementations from each of the three packages (`hdrcde`, `np` and `lpcde`). Figures 4b, 5b, 6b illustrate the `lpcde` estimate with confidence intervals using the default plotting implementation provided in the package (pointwise, non bias-corrected confidence intervals).

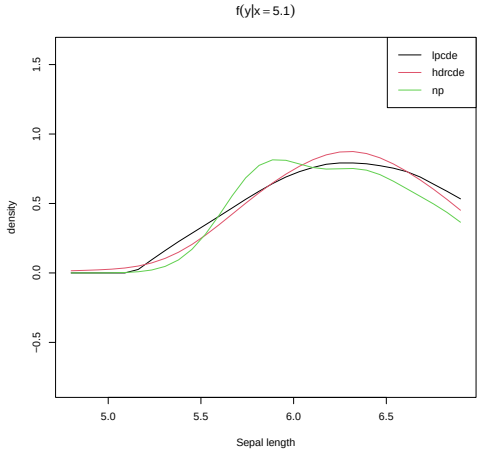


(a) Conditional density estimates.

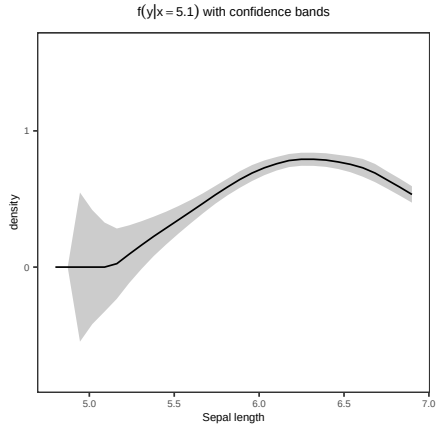


(b) `lpcde` estimate with uniform confidence bands.

Figure 5: Conditional density estimates from each implementation conditioning at *Petal length* = 4.35.



(a) Conditional density estimates.



(b) `lpcde` estimate with uniform confidence bands.

Figure 6: Conditional density estimates from each implementation conditioning at *Petal length* = 5.1.

From the plots in Figures 4a, 5a, 6a, the three estimators largely present the same expected trends of the density function. One observation that may be of interest is the that `np` estimate for $x = 4.35$ and $x = 5.1$ is slightly bi-modal, which is not reflected in the other two estimators and arguably is not present in the raw data (see Figure 3). Furthermore, the `np` estimator does not produce a valid density estimate in that the estimator does not integrate to 1 for any of conditioning values. On the other hand, `hdrcde` produces valid density estimates and is very similar to the estimates of `lpcde`. Given that `hdrcde` does not provide inference tools, we cannot compare the two packages further.

5 Conclusion

This article introduced the software package `lpcde`, which implements local polynomial kernel based regression estimation and inference for conditional densities and higher-order derivatives. This package is currently the only open source estimator that provides adaptive conditional density estimation with robust bias-correction and pointwise confidence interval and uniform confidence band construction, providing users with tools to better understand the reliability of their analysis. See Cattaneo et al. (2025) for an abbreviated published version of this article. Additional information and replication files can be found at https://nppackages.github.io/lpcde/.

Acknowledgments

The authors thank the reviewers of the *Journal of Open Source Software* (JOSS), who provided valuable feedback to improve our `R` package. Cattaneo gratefully acknowledges financial support from the National Science Foundation through grants SES-1947805, DMS-2210561, and SES-2241575, and from the National Institute of Health (R01 GM072611-16). Jansson gratefully acknowledges financial support from the National Science Foundation through grant SES-1947662.

References

Sebastian Calonico, Matias D. Cattaneo, and Max H. Farrell. On the effect of bias estimation on coverage accuracy in nonparametric inference. *Journal of the American Statistical Association*, 113(522):767–779, 2018.

Sebastian Calonico, Matias D. Cattaneo, and Max H. Farrell. Coverage error optimal confidence intervals for local polynomial regression. *Bernoulli*, 28(4):2998–3022, 2022.

Matias D. Cattaneo, Michael Jansson, and Xinwei Ma. Simple local polynomial density estimators. *Journal of the American Statistical Association*, 115(531):1449–1455, 2020.

Matias D. Cattaneo, Michael Jansson, and Xinwei Ma. `lpdensity`: Local polynomial density estimation and inference. *Journal of Statistical Software*, 101(2):1–25, 2022.

Matias D. Cattaneo, Rajita Chandak, Michael Jansson, and Xinwei Ma. Local polynomial conditional density estimators. *Bernoulli*, 30(4):3193–3223, 2024a.

Matias D. Cattaneo, Michael Jansson, and Xinwei Ma. Local regression distribution estimators. *Journal of Econometrics*, 240(2):105074, 2024b.

Matias D. Cattaneo, Rajita Chandak, Michael Jansson, and Xinwei Ma. `lpcde`: Estimation and inference for local polynomial conditional density estimators. *Journal of Open Source Software*, 10(107):7241, 2025. URL https://doi.org/10.21105/joss.07241.

Jan G De Gooijer and Dawit Zerom. On conditional density estimation. *Statistica Neerlandica*, 57 (2):159–176, 2003.

Jianqing Fan and Irene Gijbels. *Local Polynomial Modelling and Its Applications*. Chapman & Hall/CRC, 1996.

Jianqing Fan, Qiwei Yao, and Howell Tong. Estimation of conditional densities and sensitivity measures in nonlinear dynamical systems. *Biometrika*, 83(1):189–206, 1996.

Peter Hall, Rodney CL Wolff, and Qiwei Yao. Methods for estimating a conditional distribution function. *Journal of the American Statistical Association*, 94(445):154–163, 1999.

Peter Hall, Jeff Racine, and Qi Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004.

Tristen Hayfield and Jeffrey S. Racine. Nonparametric econometrics: The `np` package. *Journal of Statistical Software*, 27(5), 2008.

Nima S Hejazi, Mark J van der Laan, and David Benkeser. haldensify: Highly adaptive lasso conditional density estimation inr. *Journal of Open Source Software*, 7(77):4522, 2022.

Rob J Hyndman, Jochen Einbeck, and Matthew P Wand. *`hdrcde`: Highest Density Regions and Conditional Density Estimation*, 2021. `R` package version 3.4.

Jonas Rothfuss, Fabio Ferreira, Simon Walther, and Maxim Ulrich. Conditional density estimation with neural networks: Best practices and benchmarks. *arXiv:1903.00954*, 2019.

David W Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2015.

Jeffrey S Simonoff. *Smoothing Methods in Statistics*. Springer–Verlag, 2012.

M.P. Wand and M.C. Jones. *Kernel Smoothing*. Chapman & Hall/CRC, 1995.

Hadley Wickham. *`ggplot2`: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4.