`lspartition`: Partitioning-Based Least Squares Regression*

Matias D. Cattaneo† Max H. Farrell‡ Yingjie Feng§

December 3, 2018

Abstract

Nonparametric series least squares regression is an important tool in empirical work. Commonly used examples include B-splines, wavelets, and piecewise polynomials (also known as regressograms, and related to regression trees). This article discusses the underlying methodologies and numerical features of the R software package `lspartition`, which implements the theoretical and methodological results in Cattaneo and Farrell (2013) and Cattaneo, Farrell and Feng (2018) for partitioning-based least squares (series) regression estimation and inference. First, it provides pointwise and uniform (over the support of the conditioning variables) point estimators and robust bias-corrected inference procedures for the regression function and derivatives thereof. Second, asymptotically valid plug-in and bootstrap confidence bands are provided. Third, the package also includes data-driven methods for selecting the optimal number of partitioning knots in a integrated mean squared error sense for tensor-product partitioning schemes. Finally, an extension to two-sample analysis is given, which can be use in treatment-control comparisons and related problems.

Keywords: series nonparametrics, partitioning least squares, tuning parameter selection, robust bias-corrected inference, confidence bands, splines, wavelets, piecewise polynomials, generalized regressogram, R.

*We thank Sebastian Calonico, David Drukker, and Xinwei Ma for thoughtful comments and suggestions on our software implementation and related articles. Cattaneo gratefully acknowledges financial support from the National Science Foundation (SES-1459931).

†Department of Economics and Department of Statistics, University of Michigan.

‡Booth School of Business, University of Chicago.

§Department of Economics, University of Michigan.

1 Introduction

Nonparametric series regression estimation is an important and commonly used method for researchers in statistics, economics and other disciplines. This includes the popular class of partitioning-based least squares methods (splines, compact-supported wavelets, piecewise polynomials or generalized regressograms, regression trees, etc.). These methods first partition the support of covariates, and then construct set of local basis functions on top of the partition scheme to approximate the unknown regression function or derivatives thereof. For textbook reviews on classical and modern nonparametric regression methodology see, among others, Fan and Gijbels (1996), Györfi et al. (2002), Wasserman (2006), and Ruppert et al. (2009). For a review on partitioning-based approximations in nonparametrics and machine learning see Breiman et al. (1984), Hastie et al. (2009), Zhang and Singer (2010), and references therein.

This article gives a detailed discussion of the software package `lspartition` available in R, which implements partitioning-based least squares regression estimation and inference methods developed in Cattaneo and Farrell (2013) and Cattaneo, Farrell and Feng (2018, CFF hereafter). The package has several features specifically developed to help practitioners overcome many common difficulties in implementing least squares partitioning-based least squares regressions. First, for partitioning schemes formed as tensor products of evenly-spaced or quantile-spaced univariate partitions ("tensor-product partitions"), the package offers several integrated mean squared error (IMSE) optimal data-driven selectors for the total number of knots to use in practice. These tuning parameter selectors are developed for splines, compact-supported wavelets, and piecewise polynomials.

Second, given a partitioning scheme, the package offers pointwise and uniform (over the conditioning variables) point estimation and inference methods for the regression and derivatives thereof. Importantly, our implementations employ pre-asymptotic quantities, an approach that has been shown to offer demonstrably improvements in finite samples (Calonico, Cattaneo and Farrell, 2018b). Furthermore, the variance estimators are robust

to conditional heteroskedasticity of unknown form. In addition, for (pointwise or uniform) inference, several robust bias-correction methods are implemented, which allow for employing partitioning schemes that are IMSE-optimal from a point estimation perspective. These methods have also been shown to enjoy other theoretical and practical advantages in the context of kernel-based density and regression estimation (Calonico, Cattaneo and Farrell, 2018a,b). All these results are implemented for splines, compact-supported wavelets, and piecewise polynomials.

Third, the package `lspartition` offers two data-driven valid confidence band estimators for the entire regression function and derivatives thereof. These confidence bands are constructed by approximating the distribution of the t-statistic processes, using either a plug-in approach or a bootstrap approach. Importantly, the confidence bands constructions do not employ (asymptotic) extreme value theory, but instead use strong approximation/coupling results (CFF), which perform better in finite samples.

Lastly, the package also offers a convenient command to implement estimation and inference for linear combinations of regression estimators of different groups with all features mentioned above. As a direct application, it can be used to analyze conditional treatment effects in random control trials in particular, or for two-sample comparisons more generally.

More specifically, the package `lspartition` includes the following four functions:

- `lsprobust()`: This function implements estimation and inference procedures for partitioning-based least squares regression. It takes the partitioning scheme as given, and construct point and variance estimation, bias correction estimators, conventional and robust bias-corrected confidence intervals, and simulation-based conventional and robust bias-corrected uniform inference procedures. Three approximation basis are provided: splines, Cohen-Daubechies-Vial wavelets, and piecewise polynomials (generalized regressogram, regression trees). When partitioning scheme is not specified, the companion function `lspkselect()` is used to select the number of tensor-product partitioning knots in a fully data-driven fashion.

- `lspkselect()`: This function implements a data-driven procedure to select the number of knots for partitioning-based least squares regression. This function allows for evenly-spaced and quantile-spaced knot placements, and computes the corresponding IMSE-optimal choices. Two selectors are provided: rule-of-thumb (ROT) and direct plug-in (DPI) rule.

- `lsplincom()`: This function implements estimation and robust inference procedures for linear combinations of regression estimators for multiple groups based on `lsprobust()`. This function takes a user-specified linear combination, and implements point and variance estimation, bias correction, conventional and robust confidence intervals and sampling-based robust bias-corrected confidence bands.

- `lsprobust.plot()`: This function builds on `ggplot2`, and is used as a wrapper for plotting results. It plots regression function curves, robust bias-corrected confidence intervals, and uniform confidence bands, among other possibilities.

The outline of this paper is as follows. Section 2 describes the basic setup for partitioning-based least squares regression and the main methods implemented in the package `lspartition`. Section 3 discusses IMSE selection of number of knots and gives implementation details on the data-driven procedures for selecting these tuning parameters. Section 4 overviews three robust bias correction strategies and corresponding uniform inference procedures, and discusses their implementation. The last section concludes. We purposely avoid technicalities as much as possible, but all omitted details can be found in CFF and the references therein.

2 Setup

We assume that $\{(y_i, \mathbf{x}_i') : 1 \leq i \leq n\}$ is an observed random sample with y_i the scalar outcome of interest and $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ a vector of covariates. The estimation and inference results do not restrict the shape of th support $\mathcal{X}$ nor the partitioning used, but for tuning

parameter selection we do assume a rectangular support, i.e., $\mathcal{X} = \times_{\ell=1}^{d} \mathcal{X}_\ell$ where $\mathcal{X}_\ell = [\underline{x}_\ell, \bar{x}_\ell]$ is the support of ℓ-th covariate, $\ell = 1, \ldots, d$. Without loss of generality, we normalize it to $[0,1]^d$ throughout the paper. The object of interest is $\mu(\mathbf{x}) = \mathbb{E}[y_i | \mathbf{x}_i = \mathbf{x}]$ or its derivatives. Given a d-tuple $\mathbf{q} = (q_1, \ldots, q_d)' \in \mathbb{Z}_+^d$, we denote $\partial^{\mathbf{q}}\mu(\mathbf{x}) = \partial^{[\mathbf{q}]}\mu(\mathbf{x}) / \partial x_1^{q_1} \cdots \partial x_d^{q_d}$ where $[\mathbf{q}] = \sum_{j=1}^{d} q_j$. Estimation and inference are based on a partitioning-based series regression. Specifically, let $\Delta = \{\delta_l \subset \mathcal{X} : 1 \leq l \leq \bar{\kappa}\}$ be a collection of $\bar{\kappa}$ disjoint open sets and the closure of their union is $\mathcal{X}$.

Given the partition Δ of $\mathcal{X}$, the partitioning-based least squares estimator is constructed using an approximation basis consisting of K functions, each of which is order m, and denoted by $\mathbf{p}(\mathbf{x}) = (p_1(\mathbf{x}), \ldots, p_K(\mathbf{x}))'$. Then, the final estimator of $\partial^{\mathbf{q}}\mu(\mathbf{x})$ is

$$\widehat{\partial^{\mathbf{q}}\mu}(\mathbf{x}) = \partial^{\mathbf{q}}\mathbf{p}(\mathbf{x})'\widehat{\boldsymbol{\beta}}, \qquad \widehat{\boldsymbol{\beta}} = \underset{\mathbf{b} \in \mathbb{R}^K}{\arg\min} \sum_{i=1}^{n} \left(y_i - \mathbf{p}(\mathbf{x}_i)'\mathbf{b}\right)^2, \tag{1}$$

where $[\mathbf{q}] < m$. When $\mathbf{q} = \mathbf{0}$, we write $\widehat{\mu}(\cdot) = \widehat{\partial^{\mathbf{0}}\mu}(\cdot)$ for simplicity.

The package `lspartition` allows for three possible bases: B-splines, Daubechies wavelets, and piecewise polynomials (c.f., regressogram and regression trees). The approximation power of such bases usually relies on two user-specified parameters: the partition Δ and the order m. For the first two bases, the order m of the basis can be any positive integer, and any derivative of μ up to total order $(m-1)$ can be estimated employing such a basis. For Daubechies wavelets, the current version allows for $m \leq 4$ (i.e., up to cubic wavelets), and $\mathbf{q} = (0, \ldots, 0)$. The package takes $m = 2$ (linear basis) as default.

The choice of m is usually fixed in practice, and hence the main tuning parameter for this class of nonparametric regression estimators is the partition Δ. CFF considers general quasi-uniform partitioning scheme which allows δ_l to be polyhydral. In practice, however, the most popular choice is tensor product of intervals. Specifically, $\Delta_\ell = \{\underline{x}_\ell = t_{\ell,0} < t_{\ell,1} < \cdots < t_{\ell,\kappa_{\ell-1}} < t_{\ell,\kappa_\ell} = \bar{x}_\ell\}$ partitions $\mathcal{X}_\ell$ into κ_ℓ subintervals, and the complete partition $\Delta = \otimes_{\ell=1}^{d} \Delta_\ell$ where $\otimes$ denotes tensor product. Generally, a finer partition leads to a more

precise approximation to the unknown function. The cells forming the partition Δ have to be quasi-uniform, which excludes too irregular partitioning schemes: this condition requires that the volumes of all cells in Δ should not differ too much, and thus the unknown function can be approximated "similarly" across all cells. See CFF for all technical details.

Given a tensor-product partition Δ, the tuning parameter reduces to the *number* and *position* of partitioning knots used in each dimension of $\mathbf{x}_i$. Three types of knot configuration are supported by the package `lspartition`: user-specified, evenly-spaced, and quantile-spaced partitions. In the first case, the user has complete freedom to choose both the number and positions of knots for each dimension. In the latter two cases, the knot placement scheme is pre-specified, and hence only the number of subintervals for each dimension needs to be chosen. We denote the number of knots in each of the d dimensions of the regressor $\mathbf{x}_i$ by $\boldsymbol{\kappa} = (\kappa_1, \ldots, \kappa_d) \in \mathbb{Z}_+^d$, which can be either specified by users or selected by data-driven procedures (see Section 3 below). Moreover, for wavelet bases, motivated by the standard multi-resolution analysis, we also provide an option `J` for the regression command `lsprobust()`, which indicates the resolutional level of the basis. It relates to the number of partitioning knots in the following manner: the resolutional level s for dimension ℓ is equivalent to evenly partitioning $\mathcal{X}_\ell$ into 2^s subintervals. For more technical details, see Cohen, Daubechies and Vial (1993), Chui (2016), and references therein.

It is popular practice to choose the tuning parameter by minimizing some loss function (e.g., as in cross validation or mean square minimization). The package `lsprobust` minimizes the IMSE of the point estimator to select the number of knots forming a tensor-product partition. As discussed in the next section, when the same number of knots are used for each dimension, i.e., $\boldsymbol{\kappa} = (\kappa, \ldots, \kappa)$ for some $\kappa \in \mathbb{Z}_+$, we typically have

$$\mathsf{IMSE}[\widehat{\partial^{\mathbf{q}}\mu}(\mathbf{x})|\mathbf{X}] \asymp \kappa^{-2(m-[\mathbf{q}])} + \frac{\kappa^{d+2[\mathbf{q}]}}{n},$$

where $\asymp$ denotes that two quantities are of the same order in large samples, and $\mathbf{X} =$

$(\mathbf{x}_1, \ldots, \mathbf{x}_n)$. This IMSE approximation leads to the IMSE-optimal choice $\kappa_{\texttt{IMSE}} \asymp n^{\frac{1}{2m+d}}$ of number of knots forming the partition.

The estimator $\widehat{\partial^{\mathbf{q}}\mu}(\mathbf{x})$ is IMSE-optimal from a point estimation perspective when implemented using the choice $\kappa_{\texttt{IMSE}}$ to form Δ, but conventional inference methods based on this resulting point estimator will be invalid. More precisely, the ratio of bias to standard error in the t-statistic is non-negligible, requiring either ad-hoc undersmoothing or some form of bias correction. In addition to the (uncorrected) point estimate in (1), the package `lspartition` gives three bias correction options discussed in CFF for valid (pointwise and uniform) inference. All these strategies resort to a higher-order basis $\tilde{\mathbf{p}}(\mathbf{x}) = (\tilde{p}_1(\mathbf{x}), \ldots, \tilde{p}_{\tilde{K}}(\mathbf{x}))'$, the order of which is $\tilde{m} > m$. The partition $\tilde{\Delta}$ where $\tilde{\mathbf{p}}(\mathbf{x})$ is built on may be different from Δ, but we assume both are quasi-uniform. The maximum mesh sizes of Δ and $\tilde{\Delta}$, denoted by h and $\tilde{h}$ respectively, satisfy $h/\tilde{h} \to \rho_0 \in (0, \infty)$. In asymptotic analysis, we take limit $h \to 0$ as $n \to \infty$. The details of bias-corrected inference are discussed and illustrated in Section 4 below.

2.1 Package and Data

Throughout this article, we review in more detail the main aspects of the methods implemented in the package `lspartition`, and we also give an empirical illustration of their basic functionalities and numerical performance using a real dataset. The package is available in R and can be installed as follows:

```
> install.packages("lspartition", dependencies = TRUE)
> library(lspartition)
```

We employ a bike rental dataset from Capital Bikeshare, which is available at https://www.kaggle.com/c/bike-sharing-demand/data/. The data corresponds to the first 19 days of every month during the years 2011 and 2012. Three variables are considered: the total number of rentals per day (`count`), the "feels-like" temperature measured in Celsius degrees for each day (`atemp`), and binary variable indicating whether the day is a work day

or not (`workingday`). The following is a summary of the data:

```
> data <- read.csv("bikeSharing.csv", header = TRUE)
> summary(data)
     count            atemp          workingday
 Min.   :  1.0   Min.   : 0.76   Min.   :0.0000
 1st Qu.: 42.0   1st Qu.:16.66   1st Qu.:0.0000
 Median :145.0   Median :24.24   Median :1.0000
 Mean   :191.6   Mean   :23.66   Mean   :0.6809
 3rd Qu.:284.0   3rd Qu.:31.06   3rd Qu.:1.0000
 Max.   :977.0   Max.   :45.45   Max.   :1.0000
```

To illustrate empirically the main methods implemented in the package `lspartition`, we investigate the nonparametric relationship between temperature and number of rentals, separately for working and other days. Specifically, the outcome variable is the total number of rentals (`count`), the regression variable is the "feels-like" temperature (`atemp`), and the group variable is binary indicator of whether it is a work day or not (`workingday`):

```
> y <- data$count
> x <- data$atemp
> g <- data$workingday
```

For simplicity, we use linear splines (`m=2`) for point estimation and quadratic splines (`m.bc=3`) for bias correction in the empirical illustration, which is also the default of the package.

3 Partitioning Scheme Selection

We briefly review the IMSE expansion for the (uncorrected) partitioning-based regression estimator $\widehat{\partial^{\mathtt{q}}\mu}(\mathbf{x})$ and illustrate the IMSE-optimal partitioning selection employing this result. To differentiate the point estimator from various bias-corrected estimators that will feature below, we add a subscript "0" to the estimator in (1) and related quantities from this point on.

3.1 Bias and variance

The estimation and inference performance of $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$ depends on its bias and variance. The bias expansion for the estimator is:

$$\mathbb{E}[\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})|\mathbf{X}] - \partial^{\mathbf{q}}\mu(\mathbf{x}) = \widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x})'\mathbb{E}_n[\boldsymbol{\Pi}_0(\mathbf{x}_i)\mu(\mathbf{x}_i)] - \partial^{\mathbf{q}}\mu(\mathbf{x}) \tag{2}$$

$$\approx \mathscr{B}_{m,\mathbf{q}}(\mathbf{x}) - \widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x})'\mathbb{E}_n[\boldsymbol{\Pi}_0(\mathbf{x}_i)\mathscr{B}_{m,\mathbf{0}}(\mathbf{x}_i)], \tag{3}$$

where $\mathscr{B}_{m,\mathbf{q}}(\cdot)$ is the leading approximation error in terms of L_∞-norm, and the second term can be viewed as the accompanying error from the linear projection of $\mathscr{B}_{m,\mathbf{0}}(\cdot)$ onto the space spanned by the basis functions used ($\boldsymbol{\Pi}_0(\mathbf{x}_i) = \mathbf{p}(\mathbf{x}_i)$). Throughout the paper, $\approx$ denotes that the approximation holds for large sample in probability and $\mathbb{E}_n[\cdot]$ denotes the sample average over $1 \leq i \leq n$. The definition of $\widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}$ is more cumbersome, and can be found in CFF, where further details on notation are also given. The leading approximation error takes the following form:

$$\mathscr{B}_{m,\mathbf{q}}(\mathbf{x}) = -\sum_{\mathbf{u}\in\Lambda_m} \Big(\partial^{\mathbf{u}}\mu(\mathbf{x})\Big) h_{\mathbf{x}}^{m-[\mathbf{q}]} B_{\mathbf{u},\mathbf{q}}(\mathbf{x})$$

where Λ_m is a multi-index set with $[\mathbf{u}] = m$ for all $\mathbf{u} \in \Lambda_m$, $h_{\mathbf{x}}$ is the mesh size of the cell containing $\mathbf{x}$, and $B_{\mathbf{u},\mathbf{q}}(\mathbf{x})$ is a known function depending on the selected approximation basis. For splines, wavelets, and piecewise polynomials, the detailed definition of Λ_m as well as $B_{\mathbf{u},\mathbf{q}}(\cdot)$ are given in Supplemental Appendix of CFF.

The second term in (3) sometimes is of smaller order when the basis functions used are carefully constructed so that $\mathscr{B}_{m,\mathbf{0}}(\cdot)$ is orthogonal to $\boldsymbol{\Pi}_0(\cdot)$ in an L_2 sense with respect to the Lebesgue measure. For B-splines, orthogonal wavelets, and piecewise polynomials, this "orthogonalilty" is discussed further in Appendix A of CFF. In finite samples, however, both terms in (3) may be important. Therefore, the package `lspartition` allows users to choose whether the projection of the leading error is used in partitioning scheme selection, as well

as estimation and inference discussed below.

The conditional variance is straightforward from least-squares algebra, and takes the following familiar form:

$$\mathbb{V}\left[\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})|\mathbf{X}\right] = \frac{1}{n}\widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x})'\bar{\boldsymbol{\Sigma}}_0\widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x}), \qquad \bar{\boldsymbol{\Sigma}}_0 = \mathbb{E}_n\left[\boldsymbol{\Pi}_0(\mathbf{x}_i)\boldsymbol{\Pi}_0(\mathbf{x}_i)'\sigma^2(\mathbf{x}_i)\right] \tag{4}$$

where $\sigma^2(\mathbf{x}_i) = \mathbb{V}[y_i|\mathbf{x}_i]$ is the conditional variance. In practice, the conditional variance function is replaced by some proper "estimator", leading for example to various well-known Heteroskedasticity Consistent (HC) variance estimators for linear least squares fitting and variants thereof in nonparametric settings; see Long and Ervin (2000) for a review in the context of least-squares regression and Calonico, Cattaneo and Farrell (2018c) for an implementation in kernel-based nonparametric regression. The package `lspartition` allows for four popular HC variance estimators (`hc0`, `hc1`, `hc2`, `hc3`).

3.2 Integrated Mean Squared Error

Using the bias and variance representation above, and given a weighting function $w(\mathbf{x})$, we have the following (conditional) IMSE expansion:

$$\texttt{IMSE}[\widehat{\partial^{\mathbf{q}}\mu}(\mathbf{x})|\mathbf{X}] \approx \frac{1}{n}\mathscr{V}_{\boldsymbol{\kappa},\mathbf{q}} + \mathscr{B}_{\boldsymbol{\kappa},\mathbf{q}}, \qquad \mathscr{V}_{\boldsymbol{\kappa},\mathbf{q}} \asymp h^{-d-2[\mathbf{q}]}, \qquad \mathscr{B}_{\boldsymbol{\kappa},\mathbf{q}} \asymp h^{2(m-[\mathbf{q}])},$$

where the n-varying quantities $\mathscr{V}_{\boldsymbol{\kappa},\mathbf{q}}$ and $\mathscr{B}_{\boldsymbol{\kappa},\mathbf{q}}$ correspond to a fixed-n approximation to the variance and squared bias, respectively. Given this fact, it can be shown that the IMSE-optimal number of cells $\bar{\kappa}_{\texttt{IMSE}} = \prod_{\ell=1}^{d}\kappa_\ell \asymp n^{\frac{d}{2m+d}}$, or equivalently the IMSE-optimal mesh size $h_{\texttt{IMSE}} \asymp n^{-\frac{1}{2m+d}}$.

Under more regularity conditions on the partition and basis used, CFF derives explicit leading constants in this expansion. Specifically, let Δ be a tensor-product partition with κ subintervals used for every dimension. Then the tuning parameter reduces to a scalar κ. Assume that the partitioning knots $\{0 = t_{\ell,0} < t_{\ell,1} < \cdots < t_{\ell,\kappa-1} < t_{\ell,\kappa} = 1\}$ are generated

as quantiles of marginal distributions $G_\ell(\cdot)$ with continuous densities $g_\ell(\cdot)$, $\ell = 1, \ldots, d$, that is,

$$t_{\ell,l} = G_\ell^{-1}\left(\frac{l}{\kappa}\right), \quad l = 0, 1, \ldots, \kappa, \quad \ell = 1, \ldots, d.$$

where $G_\ell^{-1}(v) = \inf\{x \in \mathbb{R} : G_\ell(x) \geq v\}$. Then, the IMSE-optimal choice for $\mathbf{q} = \mathbf{0}$ reduces to

$$\kappa_{\mathtt{IMSE},\mathbf{0}} = \left\lceil \left(\frac{2m \sum_{\mathbf{u}_1,\mathbf{u}_2 \in \Lambda_m} \mathscr{B}_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{0}}}{d\mathscr{V}_{\mathbf{0}}} \right)^{\frac{1}{2m+d}} n^{\frac{1}{2m+d}} \right\rceil$$

where $\lceil x \rceil$ is a ceiling operator that outputs the smallest integer that is no less than x. Precise definitions of the quantities in the above formula are omitted to save space (see CFF for details).

Two popular choices of partitioning scheme are: evenly-spaced partitions (`ktype="uni"`), which sets $G_\ell(\cdot)$ to be the uniform distribution over the support of the data, and quantile-spaced partitions (`ktype="qua"`), which sets $G_\ell(\cdot)$ to be the inverse of the empirical distribution function of each covariate. The package `lspartition` implements both partitioning schemes, and for each case offers two IMSE-optimal tuning parameter selection procedures: rule of thumb (`imse-rot`) and direct plug-in (`imse-dpi`) choices. We close this section with a brief description of the implementation details and an illustration using real data.

3.3 Rule-of-thumb choice

The rule-of-thumb choice is based on the special case in which $\mathbf{q} = \mathbf{0}$ and knots are evenly spaced. The implementation steps are summarized as follows.

- **Preliminary regression**. Implement a preliminary regression using a global polynomial of degree $(m + 4)$, and denote this estimate of $\mu(\cdot)$ by $\hat{\mu}_{\mathtt{pre}}(\cdot)$.

- **Bias constant**: Let the weighting function $w(\mathbf{x})$ be the density function of $\mathbf{x}_i$. Use the preliminary regression coefficients to obtain an estimate of the mth derivatives of $\mu(\cdot)$, i.e., $\widehat{\partial^{\mathbf{u}}\mu}(\cdot) = \partial^{\mathbf{u}}\widehat{\mu}_{\mathtt{pre}}(\cdot)$, for each $\mathbf{u} \in \Lambda_m$. Then an estimate of the bias constant

is

$$\widehat{\mathscr{B}}_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{0}} = \eta_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{0}} \times \frac{1}{n}\sum_{i=1}^{n} \widehat{\partial^{\mathbf{u}_1}\mu}(\mathbf{x}_i)\widehat{\partial^{\mathbf{u}_2}\mu}(\mathbf{x}_i).$$

where $\eta_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{0}}$ is some known constant depending on the basis.

- **Variance constant**. Implement another regression of y_i^2 on $\mathbf{x}_i$ using global polynomials of degree $(m+4)$, leading to an estimate $\widehat{\mathbb{E}}[y_i^2|\mathbf{x}_i = \mathbf{x}]$. Combining it with $\hat{\mu}_{\texttt{pre}}(\cdot)$, we obtain an estimate of the conditional variance function, denoted by $\hat{\sigma}^2(\cdot)$, since $\sigma^2(\mathbf{x}) = \mathbb{E}[y_i^2|\mathbf{x}_i = \mathbf{x}] - (\mathbb{E}[y_i|\mathbf{x}_i = \mathbf{x}])^2$. Then an estimate of the variance constant is

$$\widehat{\mathscr{V}_{\mathbf{0}}} = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \hat{\sigma}^2(\mathbf{x}_i) & \text{for splines and wavelets,} \\ \binom{d+m-1}{m-1} \times \frac{1}{n}\sum_{i=1}^{n} \hat{\sigma}^2(\mathbf{x}_i) & \text{for piecewise polynomials.} \end{cases}$$

- **Rule-of-thumb** $\hat{\kappa}_{\texttt{rot}}$. Using the above results, a simple rule-of-thumb choice of κ is

$$\hat{\kappa}_{\texttt{rot}} = \left\lceil \left(\frac{2(m-[\mathbf{q}])\sum_{\mathbf{u}_1,\mathbf{u}_2\in\Lambda_m} \widehat{\mathscr{B}}_{\mathbf{u}_1,\mathbf{u}_2,\mathbf{0}}}{(d+2[\mathbf{q}])\widehat{\mathscr{V}_{\mathbf{0}}}} \right)^{\frac{1}{2m+d}} n^{\frac{1}{2m+d}} \right\rceil.$$

  While this choice of κ is obtained under strong parametric assumptions, it still exhibits the correct convergence rate ($\hat{\kappa}_{\texttt{rot}} \asymp n^{\frac{1}{2m+d}}$).

The command `lspkselect()` implements the rule-of-thumb selection (`kselect="imse-rot"`). For example, we focus on a subsample of bike rentals during working days (`g==1`), and then the selected number of knots are reported in the following:

```
> summary(lspkselect(y, x, kselect = "imse-rot", subset = (g ==
+ 1)))
Call: lspkselect

Sample size (n)                              =    7412
Basis function (method)                      =    B-spline
Order of basis point estimation (m)          =    2
Order of derivative (deriv)                  =    (0)
Order of basis bias correction (m.bc)        =    3
Knot placement (ktype)                       =    Uniform
Knot method  (kselect)                       =    imse-rot
```

```
=======================
          IMSE-ROT
       k      k.bc
=======================
       8         8
=======================
```

In this example, the IMSE rule-of-thumb estimator for the number of knots of an evenly-spaced partition is $\hat{\kappa}_{\texttt{rot}} = 8$. Note that the number of knots for bias correction (`k.bc`) is the same as `k` in this procedure.

3.4 Direct plug-in choice

Assuming that the weighting function $w(\mathbf{x})$ is equal to the density function of $\mathbf{x}_i$, the package `lspartition` implements a direct-plug-in (DPI) procedure summarized by the following steps.

- **Preliminary choice of** κ: Implement the rule-of-thumb procedure to obtain $\hat{\kappa}_{\texttt{rot}}$.

- **Preliminary regression**. Given the user-specified basis (splines, piecewise polynomials, or wavelets), knot placement scheme ("uniform" or "quantile"), and rule-of-thumb choice $\hat{\kappa}_{\texttt{rot}}$, implement a partitiong-based series regression of order $(m+1)$ to obtain derivative estimates for every $\mathbf{u} \in \Lambda_m$. Denote this preliminary estimate by $\widehat{\partial^{\mathbf{u}}\mu}_{\texttt{pre}}(\cdot)$.

- **Bias constant**. Construct an estimate $\widehat{\mathscr{B}}_{m,\mathbf{q}}(\cdot)$ of the leading error $\mathscr{B}_{m,\mathbf{q}}(\cdot)$ by replacing $\partial^{\mathbf{u}}\mu(\cdot)$ by $\widehat{\partial^{\mathbf{u}}\mu}_{\texttt{pre}}(\cdot)$. $\widehat{\mathscr{B}}_{m,\mathbf{0}}(\cdot)$ can be obtained similarly. Then, use the pre-asymptotic version of the conditional bias to estimate the bias constant:

$$\widehat{\mathscr{B}}_{\kappa,\mathbf{q}} = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{\mathscr{B}}_{m,\mathbf{q}}(\mathbf{x}_i) - \widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x})'\mathbb{E}_n[\boldsymbol{\Pi}_0(\mathbf{x}_i)\mathscr{B}_{m,\mathbf{0}}(\mathbf{x}_i)]\right)^2.$$

  As mentioned before, the second term in the conditional bias is of smaller order under some additional assumptions on the bases considered in the package `lspartition`. We employ this property to simplify the estimate of bias constant for wavelets. For splines

and piecewise polynomials, however, users may specify whether the projection of the leading error is taken into account in the selection procedure.

- **Variance constant**. Implement a partitioning-based series regression of order m using $\hat{\kappa}_{\texttt{rot}}$, and then use the pre-asymptotic version of the conditional variance to obtain an estimate of variance constant. Specifically, we have

$$\widehat{\mathscr{V}}_{\kappa,\mathbf{q}} = \frac{1}{n}\sum_{i=1}^{n} \widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x}_i)'\widehat{\boldsymbol{\Sigma}}_0\widehat{\boldsymbol{\gamma}}_{\mathbf{q},0}(\mathbf{x}_i), \quad \widehat{\boldsymbol{\Sigma}}_0 = \mathbb{E}_n[\boldsymbol{\Pi}_0(\mathbf{x}_i)\boldsymbol{\Pi}_0(\mathbf{x}_i)' w_i \widehat{\epsilon}_i^2]$$

  where $\widehat{\epsilon}_i$'s are regression residuals, $\widehat{\boldsymbol{\Sigma}}_0$ is an estimate of $\boldsymbol{\Sigma}_0 = \mathbb{E}[\boldsymbol{\Pi}_0(\mathbf{x}_i)\boldsymbol{\Pi}_0(\mathbf{x}_i)'\sigma^2(\mathbf{x}_i)]$, and w_i corresponds to an additional weighting scheme used to construct different HC variance estimators (Long and Ervin, 2000; Calonico, Cattaneo and Farrell, 2018c).

- **Direct plug-in** κ. Collecting all these results, a direct plug-in choice of κ is

$$\hat{\kappa}_{\texttt{dpi}} = \left\lceil \left(\frac{2(m-[\mathbf{q}])\hat{\kappa}_{\texttt{rot}}^{2(m-[\mathbf{q}])}\widehat{\mathscr{B}}_{\kappa,\mathbf{q}}}{(d+2[\mathbf{q}])\hat{\kappa}_{\texttt{rot}}^{-(d+2[\mathbf{q}])}\widehat{\mathscr{V}}_{\kappa,\mathbf{q}}} \right)^{\frac{1}{2m+d}} n^{\frac{1}{2m+d}} \right\rceil.$$

The following command shows results from using the direct plug-in procedure with real data:

```
> summary(lspkselect(y, x, kselect = "imse-dpi", subset = (g ==
+ 1)))
Call: lspkselect

Sample size (n)                              =    7412
Basis function (method)                      =    B-spline
Order of basis point estimation (m)          =    2
Order of derivative (deriv)                  =    (0)
Order of basis bias correction (m.bc)        =    3
Knot placement (ktype)                       =    Uniform
Knot method  (kselect)                       =    imse-dpi

=======================
         IMSE-DPI
       k     k.bc
=======================
      10        8
=======================
```

The direct plug-in procedure gives more partitioning knots than the rule-of-thumb, leading to a finer partition. For point estimation, $\hat{\kappa}_{\texttt{dpi}} = 10$ knots are estimated, while for bias correction purpose, it selects $\hat{\kappa}_{\texttt{dpi}} = 8$ knots to estimate derivatives in the leading bias. Alternatively, a selection procedure based on quantile-spaced knot placement is as follows:

```
> summary(lspkselect(y, x, kselect = "imse-dpi", ktype = "qua",
+ subset = (g == 1)))
```

The output is omitted to conserve space.

4 Estimation and inference

This section reviews and illustrates the estimation and inference procedures implemented.

4.1 Point estimation and bias correction

The IMSE-optimal choice of κ balances the variance and squared bias, leading to $\kappa_{\texttt{IMSE}} \asymp n^{\frac{1}{2m+d}}$. As a result, for $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$ with $\kappa = \kappa_{\texttt{IMSE}}$, the ratio of its bias to standard error is non-negligible, rendering conventional inference based on t-statistics invalid.

Rather than ad-hoc undersmoothing, the package `lspartition` implements three bias-corrected estimators and offers robust bias-corrected inference (Calonico et al., 2018b; Cattaneo et al., 2018). These approaches allow researchers to combine an optimal point estimate $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$ based on the IMSE-optimal $h_{\texttt{IMSE}}$ with inference based on the same tuning parameter and partitioning scheme choices.

Because the only unknown objects in the bias representation (2) are $\mu(\mathbf{x}_i)$ and $\partial^{\mathbf{q}}\mu(\mathbf{x})$, the bias correction strategies are as follows.

- **Higher-order-basis bias correction**. Use $\tilde{\mathbf{p}}(\mathbf{x})$ to construct a high-order least squares estimator $\widehat{\partial^{\mathbf{q}}\mu}_1(\mathbf{x})$, numbered as approach 1, which takes the exactly same form as $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$ but has less bias. One simply substitutes y_i and $\widehat{\partial^{\mathbf{q}}\mu}_1(\mathbf{x})$ for $\mu(\mathbf{x}_i)$ and $\partial^{\mathbf{q}}\mu(\mathbf{x})$ in (2) respectively, and then subtract this estimated bias from $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$.

The resulting "bias-corrected" estimator is equivalent to $\widehat{\partial^{\mathbf{q}}\mu}_1(\mathbf{x})$. In the package `lspartition` this option is denoted by `bc="bc1"`.

- **Least-squares bias correction**. Construct $\widehat{\partial^{\mathbf{q}}\mu}_1(\mathbf{x})$ and substitute it for $\partial^{\mathbf{q}}\mu(\mathbf{x})$ in (2), but $\mu(\mathbf{x}_i)$ is replaced by $\hat{\mu}_1(\mathbf{x}_i)$ rather than y_i. The least squares bias-corrected estimator $\widehat{\partial^{\mathbf{q}}\mu}_2(\mathbf{x})$, numbered as approach 2, is obtained by subtracting this estimated bias from $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$. The supplemental appendix to CFF discusses in detail how this approach relates to "higher-order basis bias correction" and when they are equivalent. In the package `lspartition` this option is denoted by `bc="bc2"`.

- **Plug-in bias correction**. The conditional bias of $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$ given by (3) includes the unknown object $\partial^{\mathbf{u}}\mu(\mathbf{x})$, for each $\mathbf{u} \in \Lambda_m$, which motivates the third correction strategy. Use $\tilde{\mathbf{p}}(\mathbf{x})$ to construct $\widehat{\partial^{\mathbf{u}}\mu}_1(\mathbf{x})$ for every $\mathbf{u} \in \Lambda_m$. Substitute $\widehat{\partial^{\mathbf{u}}\mu}_1(\mathbf{x})$ and $\widehat{\partial^{\mathbf{u}}\mu}_1(\mathbf{x}_i)$ for $\partial^{\mathbf{u}}\mu(\mathbf{x})$ and $\partial^{\mathbf{u}}\mu(\mathbf{x}_i)$ in $\mathscr{B}_{m,\mathbf{q}}(\mathbf{x})$ and $\mathscr{B}_{m,\mathbf{0}}(\mathbf{x}_i)$ respectively. Subtracting this estimated bias from $\widehat{\partial^{\mathbf{q}}\mu}_0(\mathbf{x})$ leads to a plug-in bias corrected estimator $\widehat{\partial^{\mathbf{q}}\mu}_3(\mathbf{x})$, numbered as approach 3. In the package `lspartition` this option is denoted by `bc="bc3"`.

The optimal (uncorrected) point estimator and three bias-corrected estimators can be written in a unified form:

$$\widehat{\partial^{\mathbf{q}}\mu}_j(\mathbf{x}) = \widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\mathbf{x})'\mathbb{E}_n[\boldsymbol{\Pi}_j(\mathbf{x}_i)y_i], \quad j = 0, 1, 2, 3.$$

These estimators only differ in $\widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\cdot)$ and $\boldsymbol{\Pi}_j(\cdot)$. See CFF for exact formulas.

4.2 Conventional and robust confidence intervals

Pointwise inference relies on a Gaussian approximation for the t-statistics:

$$\widehat{T}_j(\mathbf{x}) = \frac{\widehat{\partial^{\mathbf{q}}\mu}_j(\mathbf{x}) - \partial^{\mathbf{q}}\mu(\mathbf{x})}{\sqrt{\widehat{\Omega}_j(\mathbf{x})/n}} \rightsquigarrow \mathsf{N}(0,1), \quad j = 0, 1, 2, 3,$$

where $\widehat{\Omega}_j(\mathbf{x})/n = \widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\mathbf{x})'\widehat{\boldsymbol{\Sigma}}_j\widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\mathbf{x})/n$ is an estimator of the conditional variance of $\widehat{\partial^{\mathbf{q}}\mu}_j(\cdot)$, and $\rightsquigarrow$ denotes convergence in distribution. Nominal $100(1-\alpha)$-percent symmetric confi-

dence intervals are

$$I_j(\mathbf{x}) = \left[\widehat{\partial^{\mathbf{q}}\mu}_j(\mathbf{x}) - \Phi_{1-\alpha/2}\sqrt{\widehat{\Omega}_j(\mathbf{x})/n}, \quad \widehat{\partial^{\mathbf{q}}\mu}_j(\mathbf{x}) - \Phi_{\alpha/2}\sqrt{\widehat{\Omega}_j(\mathbf{x})/n}\right] \tag{5}$$

where $\Phi_u = \Phi^{-1}(u)$ with $\Phi(u)$ denoting the standard normal cumulative distribution function.

For $j = 0$ (uncorrected t-statistic), correct coverage of the confidence interval relies on undersmoothing ($\kappa \gg \kappa_{\texttt{IMSE}}$), which renders the bias negligible in comparison with the standard error asymptotically. Though straightforward in theory, it is not easy to implement in practice and some ad hoc procedures are often needed.

In comparison, given the IMSE-optimal tuning parameter, all three bias-corrected estimators ($j = 1, 2, 3$) have smaller-order bias, and thus the corresponding confidence intervals based on these estimators will have asymptotically correct coverage. Importantly, the Studentization quantity $\widehat{\Omega}_j(\mathbf{x})/n$ also captures the variability of estimated bias and its covariance with the uncorrected estimator. For $j = 0, 1, 2, 3$, $\widehat{\mathbf{\Sigma}}_j(\mathbf{x}) = \mathbb{E}_n[\mathbf{\Pi}_j(\mathbf{x}_i)\mathbf{\Pi}_j(\mathbf{x}_i)' w_i \widehat{\epsilon}^2_{i,j}]$ is a consistent estimator of $\mathbf{\Sigma}_j = \mathbb{E}[\mathbf{\Pi}_j(\mathbf{x}_i)\mathbf{\Pi}_j(\mathbf{x}_i)\sigma^2(\mathbf{x}_i)]$ where $\widehat{\epsilon}_{i,j} = y_i - \widehat{\mu}_j(\mathbf{x}_i)$, and w_i's are the weights corresponding to the HC variance estimation choice (e.g., `hc0`, `hc1`, etc).

We now illustrate the command `lsprobust()` using the data on number of bike rentals. We directly employ the previous results of knot selection based on the direct plug-in procedure. Specifically, we set `nknot=10` for point estimation. For higher-order bias correction (`bc="bc1"`), the same number of knots is used to correct bias by default, while for plug-in bias correction (`bc="bc3"`), we use 8 knots (`bnknot=8`) to estimate higher-order derivatives in the leading bias. It is also possible to leave these options unspecified, in which case the command `lsprobust()` automatically implements the knot selection procedure using the command `lspkselect()`.

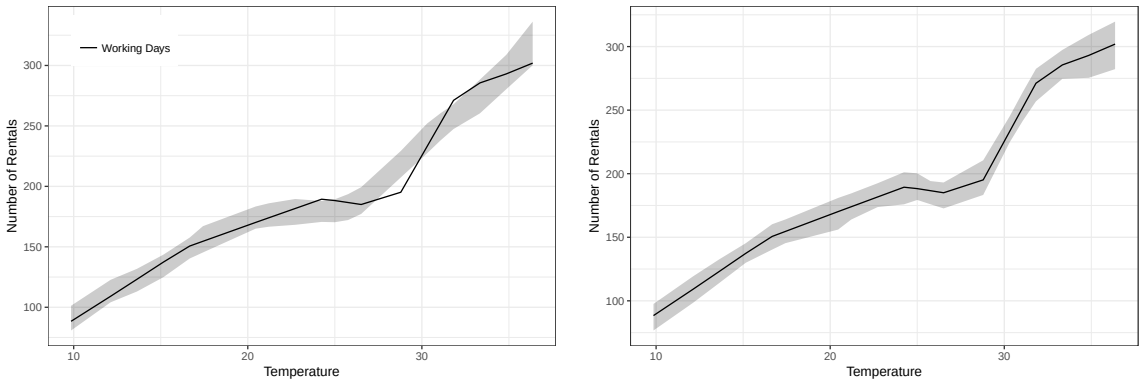
```
> est_workday_bc1 <- lsprobust(y, x, neval = 20, bc = "bc1", nknot = 10,
+ subset = (g == 1))
> est_workday_bc3 <- lsprobust(y, x, neval = 20, bc = "bc3", nknot = 10,
+ bnknot = 8, subset = (g == 1))
```

```
> summary(est_workday_bc1)
Call: lprobust

Sample size (n)                              =    7412
Num. covariates (d)                          =    1
Basis function (method)                      =    B-spline
Order of basis point estimation (m)          =    2
Order of derivative (deriv)                  =    (0)
Order of basis bias correction (m.bc)        =    3
Smoothness point estimation (smooth)         =    0
Smoothness bias correction (bsmooth)         =    1
Knot placement (ktype)                       =    Uniform
Knots method (kselect)                       =    User-specified
Uniform inference method (uni.method)        =    NA
Num. knots point estimation (nknot)          =    (10)
Num. knots bias correction (bnknot)          =    (10)

=============================================================================
      Eval               Point      Std.        Robust B.C.
       X1          n      Est.     Error       [ 95% C.I. ]
=============================================================================
1      9.850    7412    88.394     4.625    [80.800 , 101.279]
2     12.120    7412   109.093     5.693   [104.104 , 122.712]
3     13.635    7412   123.166     4.529   [113.038 , 131.821]
4     15.150    7412   137.337     3.801   [124.877 , 143.508]
5     16.665    7412   150.655     5.390   [140.337 , 157.678]
-----------------------------------------------------------------------------
6     17.425    7412   154.575     4.367   [145.334 , 167.094]
7     20.455    7412   170.197     6.688   [165.017 , 183.378]
8     21.210    7412   174.023     5.171   [166.605 , 186.009]
9     22.725    7412   181.699     4.261   [168.202 , 189.567]
10    24.240    7412   189.376     6.803   [170.587 , 187.916]
-----------------------------------------------------------------------------
11    25.000    7412   188.266     5.385   [170.342 , 189.321]
12    25.760    7412   186.631     4.423   [172.129 , 193.565]
13    26.515    7412   185.006     4.783   [177.201 , 199.335]
14    28.790    7412   195.144     7.244   [207.383 , 229.301]
15    30.305    7412   233.154     4.794   [227.434 , 252.045]
-----------------------------------------------------------------------------
16    31.060    7412   252.097     5.495   [237.854 , 260.383]
17    31.820    7412   271.165     7.222   [247.439 , 267.953]
18    33.335    7412   285.564     5.656   [260.395 , 288.333]
19    34.850    7412   293.036     7.603   [280.376 , 308.831]
20    36.365    7412   301.954    10.633   [300.155 , 336.275]
-----------------------------------------------------------------------------
=============================================================================
```

The above table summarizes the results for pointwise estimation and inference, including point estimates, conventional standard errors, and robust confidence intervals based on higher-order bias correction for 20 quantile-spaced evaluation points. We can use the com-

panion plotting command `lsprobust.plot()` to visualize the results:

```
> lsprobust.plot(est_workday_bc1, xlabel = "Temperature", ylabel = "Number
    of Rentals",
+ legendGroups = "Working Days") + theme(legend.position = c(0.15,
+ 0.9))
> ggsave("output/pointwise1.pdf")
> lsprobust.plot(est_workday_bc3, xlabel = "Temperature", ylabel = "Number
    of Rentals") +
+ theme(legend.position = "none")
> ggsave("output/pointwise2.pdf")
```



(a) Higher-order bias correction (b) Plug-in bias correction

Figure 1: Point estimation and robust confidence intervals

The result is displayed in Figure 1. It can be seen that as the temperature gets higher, the number of rentals increases as expected. We plot both the robust confidence intervals based on high-order bias correction (Figure 1a) and plug-in bias correction (Figure 1b). Since the higher-order bias correction is equivalent to quadratic spline fitting, the resulting confidence interval has a smoother shape.

4.3 Uniform inference

To implement uniform inference, CFF establishes Gaussian approximations for the *whole* t-statistic processes, and proposes several sampling-based approximating processes which are feasible and easy to implement in practice. To be concrete, there exists a Gaussian process

$Z_j(\cdot)$ such that

$$\widehat{T}_j(\cdot) \approx_d Z_j(\cdot), \quad Z_j(\cdot) = \frac{\boldsymbol{\gamma}_{\mathbf{q},j}(\cdot)'\boldsymbol{\Sigma}_j^{1/2}}{\sqrt{\Omega_j(\cdot)}}\mathsf{N}_{K_j}, \quad K_j = \dim(\boldsymbol{\Pi}_j(\cdot))$$

where $\boldsymbol{\gamma}_{\mathbf{q},j}(\cdot)$ and $\Omega_j(\cdot)$ are population counterparts of $\widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\cdot)$ and $\widehat{\Omega}_j(\cdot)$, N_{K_j} is a K_j-dimensional standard normal vector, and K_j is a constant depending on the particular bias correction method. Specifically, $K_0 = K$, $K_1 = \tilde{K}$ and $K_j = K + \tilde{K}$ for $j = 2, 3$. The notation $\approx_d$ means that the two processes are approximately equal in distribution in the following sense: in a sufficiently rich probability space, we have identical copies of $\widehat{T}_j(\cdot)$ and $Z_j(\cdot)$ whose difference converges in probability to zero.

The Gaussian stochastic process $Z_j(\cdot)$ is not feasible in practice because it involves unknown population quantities. Thus, the package `lspartition` offers two options for implementation: *plug-in* or *bootstrap*.

- **Plug-in**. Replace all unknowns in $Z_j(\cdot)$ by their consistent estimates:

$$\widehat{Z}_j(\cdot) = \frac{\widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\cdot)'\widehat{\boldsymbol{\Sigma}}_j^{1/2}}{\sqrt{\widehat{\Omega}_j(\cdot)}}\mathsf{N}_{K_j}.$$

  CFF shows that $\widehat{Z}_j(\cdot)$ delivers valid distributional approximation to $\widehat{T}_j(\cdot)$. In practice one may obtain many simulated realizations of $\widehat{Z}_j(\cdot)$ by sampling from the K_j-dimensional standard normal distribution *conditional on the data*. In the package `lspartition` the option is `uni.method="pl"`.

- **Bootstrap**. Construct a bootstrapped version of approximating process (conditional on the data):

$$\widehat{z}_j^*(\cdot) = \frac{\widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\cdot)'\mathbb{E}_n[\boldsymbol{\Pi}_j(\mathbf{x}_i)\widehat{\epsilon}_{i,j}^*]}{\sqrt{\widehat{\Omega}_j^*(\cdot)/n}}$$

  where $\widehat{\Omega}_j^*(\cdot) = \widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\cdot)'\mathbb{E}_n[\boldsymbol{\Pi}_j(\mathbf{x}_i)\boldsymbol{\Pi}_j(\mathbf{x}_i)'(\widehat{\epsilon}_{i,j}^*)^2]\widehat{\boldsymbol{\gamma}}_{\mathbf{q},j}(\cdot)$, $\widehat{\epsilon}_{i,j}^* = \omega_i\widehat{\epsilon}_{i,j}$ and $\{\omega_i\}_{i=1}^n$ is an i.i.d sequence of bounded mean zero variables with unit variance. Similarly, it is shown

in CFF that this bootstrapped process is also approximated by $Z_j(\cdot)$ conditional on the data. Thus one can implement bootstrapping by sampling from the distribution of ω_i given the data. In the package `lspartition`, ω_i's are taken to be Rademacher variables, and the corresponding option is `uni.method="wb"`.

Importantly, these strong approximations apply to the whole t-statistic processes, and thus can be used to implement general inference procedures based on transformations of $\widehat{T}_j(\cdot)$. The main regression command `lsprobust()` in the package `lspartition` may output the necessary quantities for such potential analysis when they are requested by users. Specifically, when `uni.out = TRUE`, the command returns additional objects:

- `t.num.pl, t.num.wb1, t.num.wb2`: The numerators of approximating processes except the "simulated components", which is evaluated at a set of pre-specified grid points $\mathcal{K}$. Suppose that $\mathcal{K}$ contains L grid points. Then for the plug-in method, the numerator, stored in `t.num.pl`, is the $L \times K_j$ matrix $\left\{\widehat{\gamma}_{\mathbf{q},j}(\mathbf{x})'\widehat{\mathbf{\Sigma}}_j^{1/2}/\sqrt{n} : \mathbf{x} \in \mathcal{K}\right\}$. For wild bootstrap, the numerator is separated to `t.num.wb1` and `t.num.wb2`, which are $\left\{n\widehat{\gamma}_{\mathbf{q},j}(\mathbf{x})' : \mathbf{x} \in \mathcal{K}\right\}$ and $(\mathbf{\Pi}_j(\mathbf{x}_1), \ldots, \mathbf{\Pi}(\mathbf{x}_n))'$ respectively.

- `t.denom`: The denominator of approximating processes, i.e., $\left\{\sqrt{\widehat{\Omega}_j(\mathbf{x})/n} : \mathbf{x} \in \mathcal{K}\right\}$, stored in a vector of length n.

- `res`: Residuals from the specified bias corrected regression, which is needed for the boostrap-based approximation.

For example, the following command requests the necessary quantities for uniform inference based on the plug-in method:

```
> est_workday_bc1 <- lsprobust(y, x, bc = "bc1", nknot = 4, uni.method = "
   pl",
+ uni.ngrid = 100, uni.out = T, subset = (data$workingday ==
+
+ 1))
> est_workday_bc1$uni.output$t.num.pl[1:5, ]
         [,1]         [,2]       [,3]        [,4]       [,5]        [,6]
   [,7]
```

```
[1,] 30.54858 -4.92253389 2.3112103 -1.4695793 0.7785777 -0.4509633
   0.12138911
[2,] 27.10438 -3.55298238 1.7462531 -1.1619001 0.6197846 -0.3543685
   0.09049096
[3,] 23.85586 -2.28463492 1.2358244 -0.8801707 0.4739864 -0.2658479
   0.06231217
[4,] 20.80303 -1.11749152 0.7799241 -0.6243913 0.3411832 -0.1854014
   0.03685274
[5,] 17.94588 -0.05155218 0.3785522 -0.3945616 0.2213750 -0.1130291
   0.01411266
```

We list the first 5 rows of the numerator matrix. Each row corresponds to a grid point. Since we use a linear spline for point estimation and set `nknot=4`, the higher-order bias correction is equivalent to a quadratic spline fitting. Thus the numerator matrix has 7 columns which is simply the length of quadratic spline basis.

As a special application, these results can be used to construct uniform confidence bands, which builds on the suprema of $|\widehat{T}_j(\cdot)|$. The command `lsprobust()` computes the critical value to construct confidence bands. Specifically, it generates many simulated realizations of $\widehat{Z}_j(\cdot)$ or $\widehat{z}^*_j(\cdot)$ using the methods described above, and then obtains an estimated $100(1-\alpha)$-quantile of $\sup_{\mathbf{x}\in\mathcal{X}}|\widehat{Z}_j(\mathbf{x})|$ or $\sup_{\mathbf{x}\in\mathcal{X}}|\widehat{z}^*_j(\mathbf{x})|$ given the data, denoted by $q_j(1-\alpha)$. Then, $(1-\alpha)$ confidence band for $\partial^{\mathbf{q}}\mu(\mathbf{x})$ is given by

$$\widehat{\partial^{\mathbf{q}}\mu}_j(\mathbf{x}) \pm q_j(1-\alpha)\sqrt{\widehat{\Omega}_j(\mathbf{x})/n}.$$

For example, the following command requests a critical value used to construct confidence bands:

```
> est_workday_bc1 <- lsprobust(y, x, neval = 20, bc = "bc1", uni.method =
   "pl",
+ nknot = 10, subset = (g == 1), band = T)
> est_workday_bc1$sup.cval
     95%
2.992905
```

Once the critical value is available, the command `lsprobust.plot()` is able to visualize confidence bands:

```
> lsprobust.plot(est_workday_bc1, CS = "all", xlabel = "Temperature",
+ ylabel = "Number of Rentals", legendGroups = "Working Days") +
+ theme(legend.position = c(0.15, 0.9))
> ggsave("output/uniform1.pdf")
```

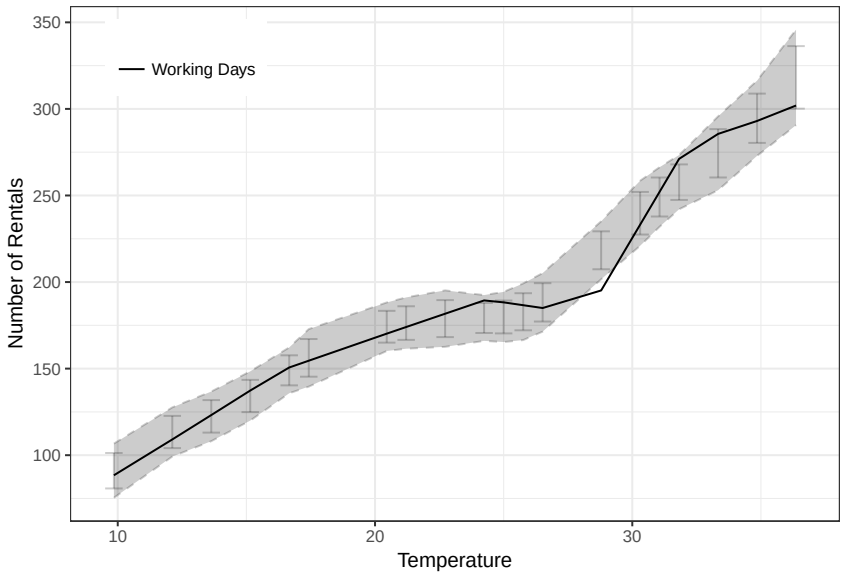


Figure 2: Point estimation and robust inference: plug-in method, higher-order BC

The result is displayed in Figure 2. Since we set `CS="all"`, the command simultaneously plots pointwise confidence intervals (error bars) and uniform confidence band (shaded regions).

It is also possible to specify other bias correction approaches or to request uniform methods.

```
> est_workday_bc3 <- lsprobust(y, x, neval = 20, bc = "bc3", nknot = 10,
+ bnknot = 8, uni.method = "wb", subset = (g == 1), band = T)
> est_workday_bc3$sup.cval
     95%
3.009072
> lsprobust.plot(est_workday_bc3, CS = "all", xlabel = "Temperature",
+ ylabel = "Number of Rentals", legendGroups = "Working Days") +
+ theme(legend.position = c(0.15, 0.9))
> ggsave("output/uniform2.pdf")
```

The result is displayed in Figure 3.

4.4 Linear combination

The package `lspartition` also includes the command `lsplincom()`, which implements estimation and inference for a linear combination of regression functions for different subgroups. To be concrete, consider a random trial with G groups. Let $\mu(\mathbf{x}; g)$ be the condi-

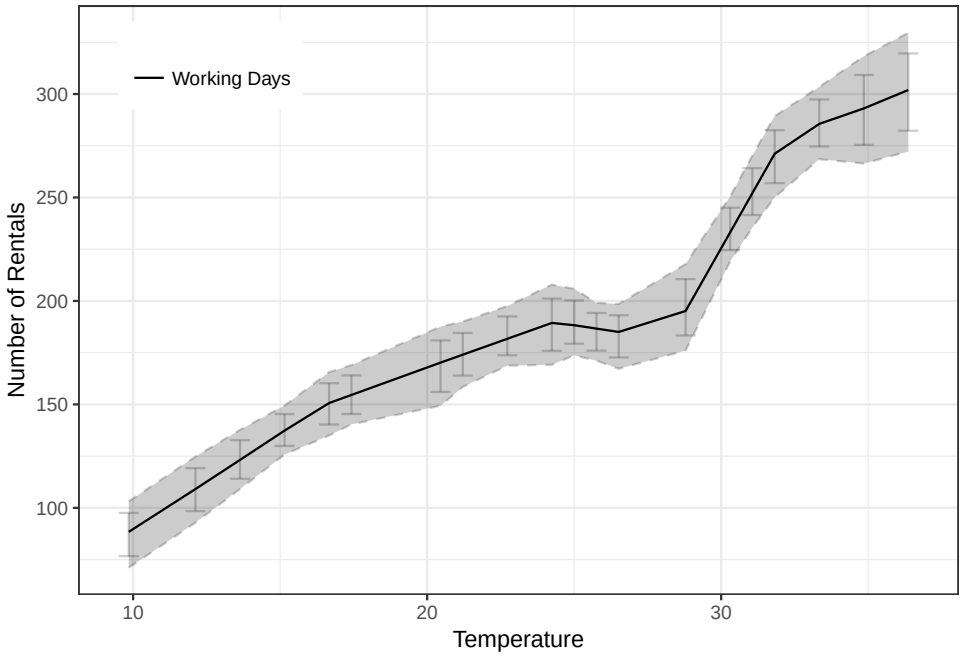


Figure 3: Point estimation and robust inference: bootstrap method, plug-in BC

tional expectation function (CEF) for group g, $g = 1, \ldots, G$. The parameter of interest is $\theta(\mathbf{x}) = \sum_{g=1}^{G} r_g \partial^{\mathbf{q}} \mu(\mathbf{x}; g)$, i.e., a linear combination of CEFs (or derivatives thereof) from different groups. As a special case, if $G = 2$, $\mathbf{q} = \mathbf{0}$ and $(r_1, r_2) = (-1, 1)$, $\theta(\mathbf{x})$ can be simply understood as the conditional average treatment effect.

To implement estimation and inference for $\theta(\mathbf{x})$, `lsplincom()` first calls `lsprobust()` to obtain a point estimate $\widehat{\partial^{\mathbf{q}} \mu}_0(\mathbf{x}; g)$ for each group as well as standard errors and other objects. The tuning parameter for each group can be selected by the data-driven procedures discussed in Section 3. Then the point estimate of $\theta(\mathbf{x})$ is

$$\widehat{\theta}_0(\mathbf{x}) = \sum_{g=1}^{G} r_g \widehat{\partial^{\mathbf{q}} \mu}_0(\mathbf{x}).$$

Regarding the confidence intervals, notice that under a random trial setting,

$$\mathbb{V}[\widehat{\theta}_j(\mathbf{x}) | \mathbf{X}] = \sum_{g=1}^{G} r_g^2 \mathbb{V}[\widehat{\partial^{\mathbf{q}} \mu}_j(\mathbf{x}; g) | \mathbf{X}].$$

Thus, the standard error of $\widehat{\theta}_j(\mathbf{x})$ can be obtained simply by taking a linear combination of standard errors for each $\widehat{\partial^{\mathbf{q}}\mu}_j(\mathbf{x};g)$. Robust confidence intervals can be similarly constructed as in (5).

`lsplincom()` also allows users to construct confidence bands for $\theta(\mathbf{x})$. Specifically, it requests `lsprobust()` to output the numerators (`t.num.pl` for "plug-in", or `t.num.wb1` and `t.num.wb2` for "bootstrap") and denominators (`t.denom`) of the feasible approximating processes $\widehat{Z}_j(\cdot)$ or $\widehat{z}^*(\cdot)$. To simplify notation, we use $\mathbf{U}_j(\cdot;g)$ and $\mathbf{v}_j(\cdot;g)$ to denote the numerator and denominator from group g using bias correction approach j, $g=1,\ldots,G$ and $j=1,2,3$. The approximating process for the t-statistic process based on $\widehat{\theta}_j(\mathbf{x})$ is

$$\widehat{Z}_{j,\theta}(\cdot)=\frac{\sum_{g=1}^{G} r_g \mathbf{U}_j(\cdot;g)\mathsf{N}_{g,K_{j,g}}}{\sqrt{\sum_{g=1}^{G} r_g^2 \mathbf{v}_{j,g}(\cdot)^2}}$$

where $\{\mathsf{N}_{g,K_{j,g}}\}_{g=1}^{G}$ is a collection of independent standard normal vectors, and $K_{j,g}$ indicates the dimension of $\mathsf{N}_{g,K_{j,g}}$. As discussed before, the dimension of these normal vectors depends on the particular bias correction approach and may vary across groups since the selected number of knots may be different across groups. The bootstrap approximating process $\widehat{z}^*_{j,\theta}(\cdot)$ can be constructed similarly.

Given these approximating processes, inference is implemented by sampling from G standard normal vectors ("plug-in" method) or G groups of Rademacher vectors given the data. Then critical values used to construct $100(1-\alpha)$ confidence bands for $\theta(\cdot)$ are estimated similarly by $100(1-\alpha)$ empirical quantiles of $\sup_{\mathbf{x}\in\mathcal{X}}|\widehat{Z}_{j,\theta}(\mathbf{x})|$ or $\sup_{\mathbf{x}\in\mathcal{X}}|\widehat{z}^*_{j,\theta}(\mathbf{x})|$.

To illustrate how the command `lsplincom()` works, we compare the number of rentals during working days and other time periods (weekends and holidays). We use linear splines combined with plug-in bias correction. To begin with, we first estimate the conditional mean function for each group using the command `lsprobust()`.

```
> est_workday <- lsprobust(y, x, neval = 20, bc = "bc3", nknot = 10,
+ subset = (g == 1))
> est_nworkday <- lsprobust(y, x, neval = 20, bc = "bc3", nknot = 9,
```

```
+ subset = (g == 0))
> lsprobust.plot(est_workday, est_nworkday, legendGroups = c("Working Days
    ",
+ "Others Days"), xlabel = "Temperature", ylabel = "Number of Rentals") +
+ theme(legend.position = c(0.15, 0.85))
```

The pointwise results for each group are displayed in Figure 4. The shaded regions represent confidence intervals. Clearly, when temperature is low, two regions are well separated, implying that people may rent bikes more during working days than weekends or holidays when the weather is cold.

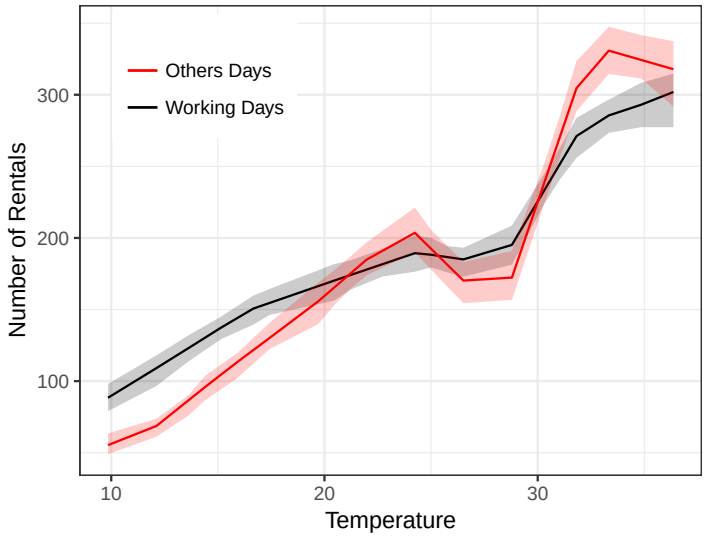


Figure 4: Point estimation and robust confidence intervals for two groups

Next, we employ the command `lsplincom()` to formally test this result. We specify `R=(-1, 1)`, denoting that -1 is the coefficient of the conditional mean function for the group `workingday==0` and 1 is the coefficient of the conditional mean function for the group `workingday==1`. In other words, we calculate the difference of two conditional mean functions.

```
> diff <- lsplincom(y, x, data$workingday, R = c(-1, 1), band = T,
+ cb.method = "pl")
> summary(diff)
Call: lprobust

Sample size (n)                              =    10886
```

```
Num. covariates (d)                              =    1
Num. groups (G)                                  =    2
Basis function (method)                          =    B-spline
Order of basis point estimation (m)              =    2
Order of derivative (deriv)                      =    (0)
Order of basis bias correction (m.bc)            =    3
Smoothness point estimation (smooth)             =    0
Smoothness bias correction (bsmooth)             =    1
Knot placement (ktype)                           =    Uniform
Knots method (kselect)                           =    imse-dpi
Confidence band method (cb.method)               =    Plug-in

=============================================================
      Eval        Point       Std.        Robust B.C.
       X1          Est.      Error       [ 95% C.I. ]
=============================================================
1      9.850     33.112      5.793     [20.350 , 44.302]
2     12.120     40.411      6.483     [27.451 , 52.223]
3     13.635     36.420      5.624     [28.185 , 51.371]
4     15.150     32.528      6.776     [14.400 , 41.242]
5     16.665     28.958      6.729     [14.461 , 40.635]
-------------------------------------------------------------
6     17.425     24.438      6.172     [10.845 , 36.681]
7     19.695     10.684      9.695     [-6.277 , 29.851]
8     21.210     -1.004      7.306    [-16.769 , 13.794]
9     21.970     -6.898      6.824    [-21.681 , 8.264]
10    23.485    -18.648     10.062    [-36.532 , 2.626]
-------------------------------------------------------------
11    25.000     -4.166      8.704    [-19.308 , 15.269]
12    25.760      5.341      7.543    [-10.787 , 22.180]
13    26.515     14.785      8.252     [-3.351 , 31.821]
14    28.790     22.860     11.688     [-0.699 , 43.017]
15    30.305     -5.310      8.108    [-21.270 , 13.813]
-------------------------------------------------------------
16    31.060    -19.349      9.098    [-37.540 , -0.418]
17    31.820    -33.480     11.622    [-58.659 , -13.827]
18    33.335    -45.258     10.195    [-66.161 , -25.796]
19    34.850    -31.297     10.370    [-55.345 , -12.040]
20    36.365    -15.889     15.561    [-48.047 , 11.056]
-------------------------------------------------------------
=============================================================
```

The pointwise results are summarized in the above table. Clearly, when temperature is low, the point estimate of the rental difference is significantly positive since the robust confidence intervals do not cover 0. In contrast, when the temperature is above 17.43, it is no longer significant. This implies that the difference in the number of rentals between working days and other periods is less pronounced when the weather is warm. Again, we can use the command `lsprobust.plot()` to plot point estimates, confidence intervals and

uniform band simultaneously:

```
> lsprobust.plot(diff, CS = "all", xlabel = "Temperature", ylabel = "
    Number of Rentals",
+ legendGroups = "Difference between Working and Other Days") +
+ theme(legend.position = c(0.3, 0.2))
> ggsave("output/diff2.pdf")
```

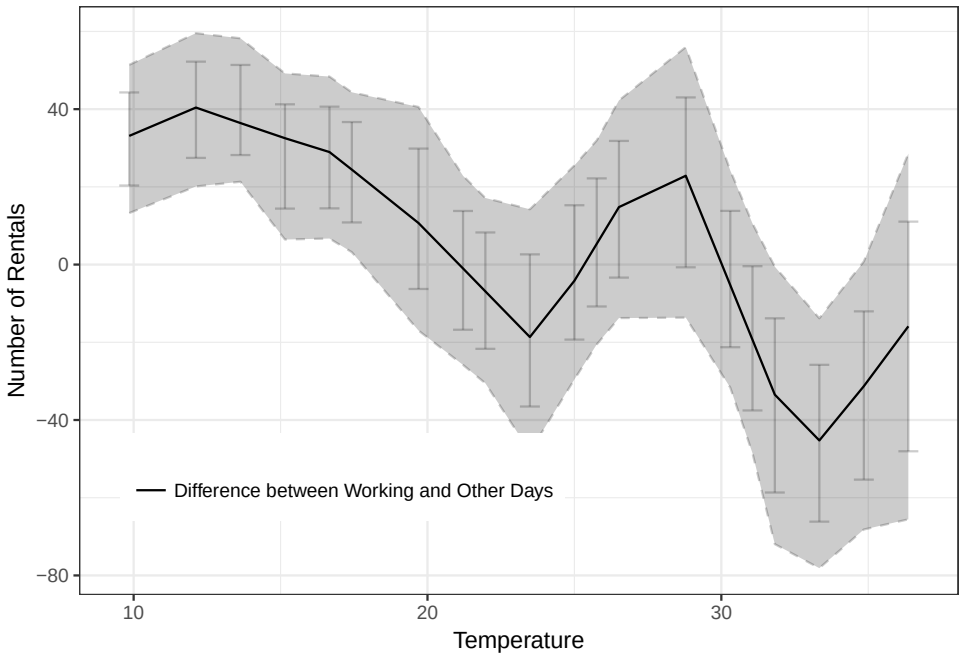


Figure 5: Point estimation and robust inference: rental difference

The requested confidence band for the difference is constructed based on the plug-in distributional approximation computed previously. It leads to an even stronger conclusion: the entire difference as a function of temperature is significantly positive uniformly over a range of low temperatures since the confidence band is above zero when the temperature is low.

5 Summary

We gave an introduction to the software package `lspartition`, which offers estimation and robust inference procedures (both pointwise and uniform) for partitioning-based least squares

regression. In particular, splines, piecewise polynomial and wavelets are implemented. The main underlying methodologies were illustrate empirically using a real dataset. Finally, installation details, scripts replicating the numerical results reported herein, links to software repositories, and other companion information, can be found in the package's website:

https://sites.google.com/site/lspartition/

References

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984), *Classification and regression trees*, CRC press.

Calonico, S., Cattaneo, M. D., and Farrell, M. H. (2018a), "Coverage Error Optimal Confidence Intervals," working paper, University of Michigan.

——— (2018b), "On the Effect of Bias Estimation on Coverage Accuracy in Nonparametric Inference," *Journal of the American Statistical Association*, 113, 767–779.

——— (2018c), "`nprobust`: Nonparametric Kernel-Based Estimation and Robust Bias-Corrected Inference," *Journal of Statistical Software*, forthcoming.

Cattaneo, M. D., and Farrell, M. H. (2013), "Optimal Convergence Rates, Bahadur Representation, and Asymptotic Normality of Partitioning Estimators," *Journal of Econometrics*, 174, 127–143.

Cattaneo, M. D., Farrell, M. H., and Feng, Y. (2018), "Large Sample Properties of Partitioning-Based Estimators," arXiv:1804.04916.

Chui, C. K. (2016), *An introduction to wavelets*, Elsevier.

Cohen, A., Daubechies, I., and Vial, P. (1993), "Wavelets on the interval and fast wavelet transforms," *Applied and Computational Harmonic Analysis*, 1, 54–81.

Fan, J., and Gijbels, I. (1996), *Local Polynomial Modelling and Its Applications*, New York: Chapman & Hall/CRC.

Györfi, L., Kohler, M., Krzyżak, A., and Walk, H. (2002), *A Distribution-Free Theory of Nonparametric Regression*, Springer-Verlag.

Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The elements of statistical learning*, Springer Series in Statistics, New York: Springer-Verlag.

Long, J. S., and Ervin, L. H. (2000), "Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model," *The American Statistician*, 54, 217–224.

Ruppert, D., Wand, M. P., and Carroll, R. (2009), *Semiparametric Regression*, New York: Cambridge University Press.

Wasserman, L. (2006), *All of Nonparametric Statistics*, Springer Science & Business Media.

Zhang, H., and Singer, B. H. (2010), *Recursive Partitioning and Applications*, Springer.