The Stata Journal (2017)
17, Number 2, pp. 372–404

rdrobust: Software for regression-discontinuity designs

Sebastian Calonico
University of Miami
Miami, FL
scalonico@bus.miami.edu

Matias D. Cattaneo
University of Michigan
Ann Arbor, MI
cattaneo@umich.edu

Max H. Farrell
University of Chicago
Chicago, IL
max.farrell@chicagobooth.edu

Rocío Titiunik
University of Michigan
Ann Arbor, MI
titiunik@umich.edu

Abstract. We describe a major upgrade to the Stata (and R) `rdrobust` package, which provides a wide array of estimation, inference, and falsification methods for the analysis and interpretation of regression-discontinuity designs. The main new features of this upgraded version are as follows: i) covariate-adjusted bandwidth selection, point estimation, and robust bias-corrected inference, ii) cluster–robust bandwidth selection, point estimation, and robust bias-corrected inference, iii) weighted global polynomial fits and pointwise confidence bands in regression-discontinuity plots, and iv) several new bandwidth selection methods, including different bandwidths for control and treatment groups, coverage error-rate optimal bandwidths, and optimal bandwidths for fuzzy designs. In addition, the upgraded package has superior performance because of several numerical and implementation improvements. We also discuss issues of backward compatibility and provide a companion R package with the same syntax and capabilities.

Keywords: st0366_1, rdrobust, rdbwselect, rdplot, regression discontinuity

1 Introduction

The regression-discontinuity (RD) design is widely used in applied work. It is one of the most credible quasi-experimental research designs for identification, estimation, and inference of treatment effects (local to the cutoff). RD designs are also easy to present, interpret, and falsify, which are features that have contributed to their popularity among practitioners and policy makers alike. See Imbens and Lemieux (2008) and Lee and Lemieux (2010) for early reviews; Cattaneo, Titiunik, and Vazquez-Bare for a practical introduction to RD designs with a comparison between leading empirical methods; and Cattaneo and Escanciano (2017) for an edited volume with a recent overview of the literature.

In this article, we describe a major upgrade to the Stata and R software package `rdrobust` (Calonico, Cattaneo, and Titiunik 2014a, 2015b), which provides a wide array of estimation, inference, and falsification methods for the analysis and interpretation of

RD designs. These major upgrades are implemented following the technical and methodological results discussed in Calonico, Cattaneo, and Farrell (Forthcoming, 2016a) and Calonico et al. (2016b, CCFT hereafter) and its supplemental appendix. To avoid repetition, in this article we focus exclusively on the new functionalities incorporated into the package; for a description of all previously available features, refer to the previously published software articles. The main new features of the upgraded `rdrobust` package are the following (organized by underlying command or function):

1. `rdrobust`. This command now allows for covariate-adjusted point estimation and covariate-adjusted robust bias-corrected inference. In addition, this command now allows for different heteroskedasticity-robust (heteroskedasticity-consistent k class or HCk class) and cluster–robust variance estimation methods. When mean squared error (MSE)–optimal bandwidths are used, the resulting point estimator for the RD treatment effect is MSE optimal. When coverage error-rate (CER) optimal bandwidths are used, the resulting confidence intervals (CIs) for the RD treatment effect are CER–optimal.

2. `rdbwselect`. This command now offers data-driven bandwidth selection for either one common bandwidth or two distinct bandwidths on either side of the cutoff, selected to be either MSE optimal or CER optimal. In addition, MSE- and CER-optimal bandwidth choices for fuzzy RD designs are now also available. Furthermore, new regularization methods are also provided. Finally, the new implementations allow for covariate-adjusted methods, as well as for different heteroskedasticity-robust (HCk class) and cluster–robust variance estimation methods.

3. `rdplot`. This command now allows for kernel weighting and possibly different bandwidths on either side of the cutoff, which permits plotting treatment effects using RD plots. In addition, this command now allows for CIs for each bin to assess the (local) variability of the partitioning fit.

Also, all three commands now allow for optional user-defined frequency weights. Furthermore, the upgraded `rdrobust` package has new and improved numerical implementations, which now permit feasible executions with large sample sizes.

First, we tested the default implementation of the old 2014 against the new 2016 `rdrobust` command, which includes data-driven bandwidth selection via `rdbwselect` and uses nearest neighbor (NN)–based variance estimators. We used 100 replications of simulation model 1 in Calonico, Cattaneo, and Titiunik (2014b) and CCFT. The average computation time is reported in table 1 for six different sample sizes: $n = 500, 1000, 5000, 10000, 50000, 100000$. The 2016 package exhibits remarkable improvements in execution time, especially for larger sample sizes (because the old version of the software is coded in a way that does not scale well with n). For example, for $n = 50000$, the average execution time is 95.651 seconds with the 2014 version but only 1.148 seconds with the 2016 version. Thus, for this sample size, the upgraded `rdrobust` command runs 83.32 times faster than its predecessor. Importantly, the default implementation is fully backward compatible (see section 6 for details), and therefore the

execution time improvements are exclusively attributable to the new way of implementing the package.

Second, the new package was tested using 30+ million observations in Stata/SE 14 with the following results: full execution, including data-driven bandwidth selection, took roughly 5 minutes if the default options were used and roughly 16 minutes if a cluster–robust option was used in addition. (We thank Quentin Brummet at the U.S. Census for carrying out these numerical tests.) In sum, we found that the new version of the package exhibits substantial speed improvements relative to its predecessor.

Table 1. Speed comparisons between 2014 and 2016 `rdrobust` versions

Sample size (n)	Old `rdrobust` (2014) (average time in seconds)	New `rdrobust` (2016) (average time in seconds)	Time improvement (old/new)
500	0.216	0.154	1.40
1,000	0.270	0.181	1.49
5,000	1.531	0.257	5.96
10,000	4.952	0.375	13.21
50,000	95.651	1.148	83.32
100,000	385.106	2.104	183.04

Notes: i) Computed using 100 replications of simulation model 1 in Calonico, Cattaneo, and Titiunik (2014b) and CCFT using an Intel Xeon CPU E5-2620 v2 @ 2.1GHz, 32Gb RAM and Stata 14.2. ii) Time is measured in seconds. iii) Time improvement is computed as the ratio of the old `rdrobust` (2014) average speed in seconds relative to the new `rdrobust` (2016) average speed in seconds.

The 2016 version of the `rdrobust` package offers a comprehensive set of tools for a systematic and objective analysis of RD designs in empirical work. For related Stata and R commands implementing manipulation testing based on discontinuity in density using local polynomial techniques, see Cattaneo, Jansson, and Ma (2017) and references therein. For Stata and R commands implementing inference procedures based on a local randomization assumption, see Cattaneo, Titiunik, and Vazquez-Bare (Forthcoming) and references therein.

In the remaining sections, we provide methodological, practical, and empirical introductions to the new functionalities of the `rdrobust` package. In section 2, we briefly review the main methodological concepts underlying the methods implemented. We then provide the full syntax and a brief explanation of the functionalities of each of the three upgraded Stata commands in sections 3, 4, and 5, explicitly highlighting what is new relative to the previous version. In section 6, we discuss issues of backward compatibility. In section 7, we present an empirical illustration of the new methods and functionalities available in the upgraded `rdrobust` package, using the same dataset previously used in Calonico, Cattaneo, and Titiunik (2014a, 2015b) to facilitate the comparison. Finally, we conclude the article in section 8. We also provide a companion R package with the same functionality and syntax.

The latest version of this software, as well as other related software for RD designs, can be found at https://sites.google.com/site/rdpackages/.

2 Overview of new methods

Here we provide a brief account of the main new features included in the upgraded version of the `rdrobust` package. All technical and methodological results are discussed in Calonico, Cattaneo, and Farrell (2016a) and CCFT and their supplemental appendices. As mentioned above, we assume that the reader is familiar with the previous versions of the `rdrobust` package, their companion software articles, and the underlying technical papers. Therefore, in this section, we focus exclusively on what is new.

In addition to several numerical and implementation upgrades, four main new features are available in the package: i) inclusion of covariates, ii) new heteroskedasticity-consistent (HC) and cluster–robust variance estimators, iii) new bandwidth selection methods, and iv) kernel weighting and CIs in RD plots. We discuss these new features in the next four subsections.

For clarity, we focus on sharp RD designs exclusively. The software does cover all other RD designs, but because all the new features are conceptually identical for any RD design, we do not spell out the details for fuzzy and kink RD designs beyond giving a few generic examples at the end of section 7. See Card et al. (2015) for identification results in kink RD designs, see the help files for specific implementation details, and see CCFT for technical and methodological results when using additional covariates or allowing for clustering.

2.1 Using additional covariates

The first main change to the software is that covariates may be included in the estimation. To make this precise, we first describe the set up. The observed data are assumed to be a random sample $(Y_i, T_i, X_i, \mathbf{Z}_i')'$, $i = 1, 2, \ldots, n$, from a large population. The score, index, or running variable is X_i, and treatment status is determined as $T_i = \mathbb{1}(X_i \geq \overline{x})$ for the known cutoff $\overline{x}$. Using the potential-outcomes framework, the observed outcome is $Y_i = Y_i(0) \times (1 - T_i) + Y_i(1) \times T_i$, where $Y_i(0)$ and $Y_i(1)$ denote the potential outcomes for each unit under control and treatment, respectively. The d-dimensional vector $\mathbf{Z}_i$ denotes a collection of "preintervention" covariates that could be continuous, discrete, or mixed.

The parameter of interest is the standard RD treatment effect at the cutoff:

$$\tau = \tau(\overline{x}) = \mathbb{E}\left\{Y_i(1) - Y_i(0) | X_i = \overline{x}\right\}$$

The goal is to estimate τ via local polynomial methods at the cutoff $\overline{x}$. Previously, the `rdrobust` package would use only the outcome Y_i and score X_i to estimate the RD treatment effect. Now, the additional covariates $\mathbf{Z}_i$ may also be included, which can increase the efficiency of the estimator. The covariate-adjusted RD estimator of τ implemented in `rdrobust` is defined as

$$\widetilde{\tau}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h) - \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$$

where $\widetilde{\boldsymbol{\beta}}_{Y+,p}(h)$ and $\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$ are defined through

$$\widetilde{\boldsymbol{\theta}}_{Y,p}(h) = \underset{\boldsymbol{\beta}_-,\boldsymbol{\beta}_+,\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\{ Y_i - \mathbf{r}_{-,p}(X_i-\overline{x})'\boldsymbol{\beta}_- - \mathbf{r}_{+,p}(X_i-\overline{x})'\boldsymbol{\beta}_+ - \mathbf{Z}_i'\boldsymbol{\gamma} \right\}^2 K_h(X_i-\overline{x})$$

with $\widetilde{\boldsymbol{\theta}}_{Y,p}(h) = \{\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)', \widetilde{\boldsymbol{\beta}}_{Y+,p}(h)', \widetilde{\boldsymbol{\gamma}}_{Y,p}(h)'\}'$; $\boldsymbol{\beta}_-, \boldsymbol{\beta}_+ \in \mathbb{R}^{p+1}$ and $\boldsymbol{\gamma} \in \mathbb{R}^d$; $\mathbf{r}_{-,p}(x) = \mathbb{1}(x<0)(1, x, \ldots, x^p)'$; $\mathbf{r}_{+,p}(x) = \mathbb{1}(x \geq 0)(1, x, \ldots, x^p)'$; $\mathbf{e}_0$, the $(p+1)$ vector, with a 1 in the first position and 0s in the rest; and $K_h(u) = K(u/h)/h$ for a kernel function $K(\cdot)$ and a positive bandwidth sequence h. The kernel and bandwidth serve to localize the regression fit near the cutoff, and the most popular choices are i) the uniform kernel, giving equal weighting to observations $X_i \in [\overline{x}-h, \overline{x}+h]$, and ii) the triangular kernel that assigns linear down-weighting to the same observations. The preferred choice of polynomial order is $p = 1$, which gives the standard local linear RD point estimator.

The new version of the package allows for weighted least-squares estimation and inference. To be more precise, if unit-specific weights w_i are provided by the user, then the above fitting (and all underlying estimation and inference procedures) is done with $w_i \times K_h(X_i - \overline{x})$ in place of the simple kernel weights $K_h(X_i - \overline{x})$.

As formalized in CCFT, the covariates are introduced in a joint least-squares fit to minimize the underlying assumptions required for the covariate-adjusted RD estimator $\widetilde{\tau}(h)$ to remain consistent for the standard RD treatment effect τ. This requires one to assume that some features of the marginal distributions of $\mathbf{Z}_i$ above and below the cutoff are equal. This idea matches what typically is understood as covariates being "pretreatment" in the context of randomized experiments. In the case of sharp and fuzzy RD designs, it is sufficient to assume that the potential covariates under treatment and control have equal conditional expectation at the cutoff. Indeed, this is often conceived and presented as a falsification or placebo test in RD empirical studies. This simple requirement of balanced covariates at the cutoff, or zero RD treatment effect on covariates, ensures that $\widetilde{\tau}(h) \to_{\mathbb{P}} \tau$. In the case of sharp and fuzzy kink RD designs, it is required to assume that the first derivative of the regression functions under treatment and control are equal at the cutoff. These conditions can be tested empirically using the package `rdrobust`, for example, by taking the covariates as outcome variables.

The precise form of $\widetilde{\tau}(h)$ warrants several comments. Most notably, the estimation combines units from both sides of the cutoff $\overline{x}$, whereas, typically, the two sides are estimated separately. The most salient consequence is that the coefficient on the covariates, $\widetilde{\boldsymbol{\gamma}}_{Y,p}(h)$, is common to both treatment and comparison groups. This turns out to be important for consistency of $\widetilde{\tau}(h)$, the covariate-adjusted RD estimator of τ, as mentioned above. Furthermore, this mimics standard widespread practice for experimental treatment-effects estimation with covariate adjustment, where additional covariates are typically included in an additive-separable, linear-in-parameters way. Finally, when the covariates are excluded, the standard RD treatment effect previously implemented in `rdrobust` is recovered. See section 6 for more discussion on backward compatibility.

In the upgraded version of `rdrobust`, we continue to use the same kernel function for units below and above the cutoff, but we now allow for possibly different bandwidths on either side. This feature is discussed in more detail below in the context of data-driven bandwidth selection. The presentation of $\widetilde{\tau}(h)$ assumes an equal bandwidth applied to both sides only to simplify the exposition. The supplemental appendix of CCFT contains all the details, including the possibility of using different bandwidths on either side of the cutoff.

For inference based on $\widetilde{\tau}(h)$, we continue to use robust nonparametric bias-correction methods, following the recent results in Calonico, Cattaneo, and Titiunik (2014b) and Calonico, Cattaneo, and Farrell (Forthcoming, 2016a). Although technically more cumbersome, because of the inclusion of additional covariates and joint RD estimation, the core ideas underlying robust bias-correction inference do not change much with the inclusion of $\mathbf{Z}_i$, and they have already been discussed from an implementation perspective in Calonico, Cattaneo, and Titiunik (2014a, 2015b).

Thus, we do not reproduce the details here and instead offer a quick outline of their main features for future reference and discussion: i) the misspecification bias of $\widetilde{\tau}(h)$ now depends on the curvature of $\mathbb{E}\{Y_i(t)|X_i = x\}$, as before, and also on the curvature of the conditional expectations of the (potential) additional covariates given the score included in the estimation; ii) this leading bias takes the form $h^{p+1}\mathcal{B}$, where $\mathcal{B}$ is different depending on whether additional covariates are included; iii) the bias-corrected estimator is then $\widetilde{\tau}^{\mathsf{bc}}(h,b) := \widetilde{\tau}(h) - h^{p+1}\widetilde{\mathcal{B}}(b)$, where $\widetilde{\mathcal{B}}(b)$ denotes an estimator of $\mathcal{B}$ constructed using a possibly different preliminary bandwidth sequence b; and iv) the variance of $\widetilde{\tau}^{\mathsf{bc}}(h,b)$ is denoted by $\mathcal{V}^{\mathsf{bc}}(h,b)$, which captures both the variability of the RD point estimator and the variability of bias correction.

Based on the above, and under standard regularity conditions, CCFT show that valid asymptotic inference for τ can be conducted using the usual robust bias-corrected t statistic $\sqrt{nh}\{\widetilde{\tau}^{\mathsf{bc}}(h,b) - \tau\}/\sqrt{\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)}$, where $\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)$ denotes a consistent variance estimator of $\mathcal{V}^{\mathsf{bc}}(h,b)$. Using this result, for example, we obtain a $100\times(1-\alpha)\%$ robust bias-corrected covariate-adjusted CI for the treatment effect τ,

$$\left[\widetilde{\tau}^{\mathsf{bc}}(h,b) - \frac{\Phi_{1-\alpha/2}}{\sqrt{nh}} \times \sqrt{\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)} \quad , \quad \widetilde{\tau}^{\mathsf{bc}}(h,b) + \frac{\Phi_{1-\alpha/2}}{\sqrt{nh}} \times \sqrt{\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)}\right]$$

where Φ_α denotes the α percentile of the standard normal distribution.

In the next two subsections, we discuss the choice of variance estimator, $\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)$, which now allows for different heteroskedasticity-robust and cluster–robust methods, and the choice of bandwidths, which now allows for several data-driven plug-in methods. Recall that we focus exclusively on the new features of the `rdrobust` package. See section 6 for more discussion on backward compatibility.

2.2 HCk and cluster–robust variance estimation

The variance estimator used in `rdrobust` is designed to capture the variability of the initial estimator, $\widetilde{\tau}(h)$, and the bias correction, $h^{p+1}\widetilde{\mathcal{B}}(b)$, and as such will depend on both bandwidths. The particular form of the variance, $\mathcal{V}^{\mathsf{bc}}(h,b)$, is derived with a fixed-n (preasymptotic) approach, conditional on the score observations $X_1, X_2, \ldots, X_n$. This approach, together with the fact that both sources of variability are captured, yields asymptotic refinements and increased robustness to bandwidth choice of the associated inference procedures. Importantly, in the present context, $\mathcal{V}^{\mathsf{bc}}(h,b)$ depends on the additional covariates and hence is necessarily different from prior work.

The only unknown elements of $\mathcal{V}^{\mathsf{bc}}(h,b)$ are the variances of the outcome and the covariates, and their covariance, conditional on $X_1, X_2, \ldots, X_n$ (the latter two being new here). That is, to form an estimator $\widetilde{\mathcal{V}}^{\mathsf{bc}}(h,b)$, we must estimate, for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, d$,

$$\sigma_{YZ_k-,i} = \mathbb{C}\mathrm{ov}\left[Y_i(0), Z_{ki}(0) | X_i\right], \qquad \sigma_{YZ_k+,i} = \mathbb{C}\mathrm{ov}\left[Y_i(1), Z_{ki}(1) | X_i\right]$$

The feasible variance estimators are then constructed by replacing these unknown objects with estimators thereof, according to one of the following two options:

1. Nearest-neighbor method. This method uses ideas in Müller and Stadtmuller (1987) and Abadie and Imbens (2008). We replace $\sigma_{YZ_k-,i}$ and $\sigma_{YZ_k+,i}$ by the corresponding sample covariance estimator based on the J NNs to unit i, among units belonging to the same group (that is, below or above the cutoff). Specifically, neighbors are determined using the Euclidean distance based on the score variable X_i, and J denotes a (fixed) number of neighbors chosen. Up to the change of variable being used (that is, Y_i or Z_{ki} for $k = 1, 2, \ldots, d$), the procedure is exactly the same as the one used in the previous version of the `rdrobust` package.

2. HCk plug-in residuals method. This method applies ideas from least-squares estimation and inference; see MacKinnon (2013) and Cameron and Miller (2015) for review on variance estimation in this context. In this case, we replace $\sigma_{YZ_k-,i}$ and $\sigma_{YZ_k+,i}$ by, respectively,

$$\mathbb{1}(X_i < \overline{x})\omega_{-,i}\left\{Y_i - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Y-,q}(h)\right\}\left\{Z_{ki} - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Z_k-,q}(h)\right\}$$
$$\mathbb{1}(X_i \geq \overline{x})\omega_{+,i}\left\{Y_i - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Y+,q}(h)\right\}\left\{Z_{ki} - \mathbf{r}_q(X_i - \overline{x})'\widehat{\boldsymbol{\beta}}_{Z_k+,q}(h)\right\}$$

   for $k = 1, 2, \ldots, d$, which are plug-in residuals obtained from running local polynomial regressions using either the main outcome variable or the additional covariates as the dependent variable. The weights $\omega_{-,i}$ and $\omega_{+,i}$ denote a possible finite-sample adjustment for HCk variance estimators, and $q > p$ denotes the polynomial order used for bias correction.

Precise use of these estimators, and the relevant formulas, are discussed in detail in the supplemental appendix of CCFT. See also Bartalotti and Brummet (2017) for a

discussion on cluster–robust inference in sharp RD designs. Cluster–robust versions of variance estimators are also implemented, following the same logic described above but also accounting for the (one-way) cluster structure of the data.

The different variance estimators are used to Studentize statistics, form CIs, and implement data-driven bandwidth selectors, allowing for both conditional heteroskedasticity (NN, HC0, HC1, HC2, HC3) and clustering (NN or plug-in residuals with the usual degrees-of-freedom adjustment). To summarize, the upgraded `rdrobust` package includes a total of 10 distinct variance estimators (NN or plug-in residuals with either HC0, HC1, HC2, or HC3, and NN-adjusted or degrees-of-freedom–adjusted clustering). As a comparison, the previous version of `rdrobust` had only two (NN or plug-in residuals using HC0 weighting). As explained above, when additional covariates are not included, the upgraded implementations may be used for standard RD inference involving only the outcome and the score variables. See section 6 for more discussion on backward compatibility.

Finally, an important practical issue regarding the NN variance estimators arises when the running variable X_i is not continuously distributed. In some applications, X_i may exhibit mass points, making the number of eligible equidistant near neighbors strictly larger than the number specified in `rdrobust` or `rdbwselect` (recall that the default is three neighbors for each observation). In the previous version of these commands, ties were broken at random to select the exact number of neighbors prespecified by the user (or set by default). In the upgraded version of these commands, the NN variance estimators use all equidistant neighbors, even if the total number exceeds the one selected. This new approach is fully replicable and also leads to a more efficient variance estimator.

2.3 Bandwidth selection

The `rdbwselect` command has also been upgraded and now offers several new data-driven bandwidth selection methods. The three main upgrades are i) homogenized and improved MSE-optimal bandwidth choices for sharp RD designs, ii) new MSE-optimal bandwidth choices for fuzzy RD designs, and iii) new CER-optimal bandwidth choices for sharp and fuzzy RD designs. As a comparison, the previous version of `rdbwselect` implemented only three main types of MSE-optimal bandwidth choices for sharp RD designs (`ik`, `cct`, and `cv`). The new version implements over 10 different choices for sharp RD designs (and also implements the corresponding choices for fuzzy RD designs). In addition, we continue to offer regularization methods for all choices implemented, following Imbens and Kalyanaraman (2012), but implemented as discussed in Calonico, Cattaneo, and Titiunik (2014a,b, 2015b) and the corresponding supplemental appendices.

In this section, we heuristically explain some of the bandwidth choices offered in the upgraded version of the `rdbwselect` command. See Cattaneo and Vazquez-Bare (2016) for a more comprehensive discussion and comparison between bandwidth selection procedures. This command now allows for different bandwidths on either side of the cutoff

(previously only one common bandwidth was allowed), in addition to distinguishing between sharp and fuzzy RD designs and between MSE-optimal and CER-optimal choices. We outline only the case of sharp RD designs to avoid cumbersome notation. For technical and methodological details, see CCFT and its supplemental appendix.

Regarding the new bandwidth selectors implemented, note that a typical asymptotic MSE expansion gives

$$\text{MSE}\left\{\widehat{\theta}(h)\right\} \approx h^{2p+2}B + \frac{1}{nh}V$$

for some estimator $\widehat{\theta}(h)$, where B and V denote the squared bias and the variance of the estimator, respectively. Different estimators will have different bias and variance terms, which also depend on whether additional covariates are included. For example, in sharp RD designs, we consider four main alternative MSE expansions determined by the choice of estimator:

1. RD estimator $\widehat{\theta}(h) = \widetilde{\tau}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h) - \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$
2. Left-hand-side estimator $\widehat{\theta}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$
3. Right-hand-side estimator $\widehat{\theta}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h)$
4. Sum of the one-sided estimators $\widehat{\theta}(h) = \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y+,p}(h) + \mathbf{e}_0'\widetilde{\boldsymbol{\beta}}_{Y-,p}(h)$

We construct these MSE expansions for both the standard case without covariates and the covariate-adjusted case. This gives a set of alternative MSE-optimal bandwidth choices: $h_{\texttt{mse,rd}}$, $h_{\texttt{mse,l}}$, $h_{\texttt{mse,r}}$, and $h_{\texttt{mse,sum}}$, with or without additional covariates. The first three are directly applicable to RD designs, while the fourth is mostly useful for regularization purposes.

Assuming the denominator is not 0, any of these MSE-optimal bandwidth choices, with or without additional covariates, takes the following form:

$$h_{\texttt{mse},j} = \left\{\frac{V_j/n}{2(1+p)B_j}\right\}^{\frac{1}{3+2p}} \qquad j \in \{\texttt{rd},\texttt{l},\texttt{r},\texttt{sum}\}$$

where the constants are specific to the option chosen. Given a choice of MSE-optimal bandwidth, preliminary estimates of the leading asymptotic constants are straightforward to construct, though they depend on whether additional covariates have been included as well as whether heteroskedasticity or clustering is assumed. This leads to the MSE-optimal plug-in bandwidth selectors:

$$\widehat{h}_{\texttt{mse},j} = \left\{\frac{\widehat{V}_j/n}{2(1+p)\widehat{B}_j}\right\}^{\frac{1}{3+2p}} \qquad j \in \{\texttt{rd},\texttt{l},\texttt{r},\texttt{sum}\}$$

where the exact form of the specific preliminary estimates, and some of their asymptotic properties, are discussed in the supplemental appendix of CCFT. The upgraded `rdbwselect` command implements all these alternatives.

In addition, following the recent work in Calonico, Cattaneo, Farrell (Forthcoming, 2016a), we implement CER-optimal bandwidth choices. These alternative plug-in bandwidth selectors take the form

$$\widehat{h}_{\texttt{cer},j} = n^{-\frac{p}{(3+p)(3+2p)}} \times \widehat{h}_{\texttt{mse},j} \qquad\qquad j \in \{\texttt{rd}, \texttt{l}, \texttt{r}, \texttt{sum}\}$$

These bandwidth choices minimize the CER of the robust bias-corrected CI implemented by `rdrobust` and therefore may be preferable in practice for inference purposes.

To summarize, the major upgrade of the `rdbwselect` command offers now eight distinct MSE-optimal bandwidth choices ($\widehat{h}_{\texttt{mse,rd}}$, $\widehat{h}_{\texttt{mse,l}}$, $\widehat{h}_{\texttt{mse,r}}$, and $\widehat{h}_{\texttt{mse,sum}}$, with and without regularization) and eight distinct CER-optimal bandwidth choices ($\widehat{h}_{\texttt{cer,rd}}$, $\widehat{h}_{\texttt{cer,l}}$, $\widehat{h}_{\texttt{cer,r}}$, and $\widehat{h}_{\texttt{cer,sum}}$, with and without regularization). Furthermore, we also provide the following two additional bandwidth selectors, which have better rate properties:

- Possibly different bandwidths on either side: $\widehat{h}_{\texttt{comb,l}} = \text{median}\{\widehat{h}_{\texttt{mse,l}}, \widehat{h}_{\texttt{mse,rd}}, \widehat{h}_{\texttt{mse,sum}}\}$ and $\widehat{h}_{\texttt{comb,r}} = \text{median}\{\widehat{h}_{\texttt{mse,r}}, \widehat{h}_{\texttt{mse,rd}}, \widehat{h}_{\texttt{mse,sum}}\}$, and similarly for the CER-optimal version.
- Equal bandwidth on both sides: $\widehat{h}_{\texttt{comb}} = \min\{\widehat{h}_{\texttt{mse,rd}}, \widehat{h}_{\texttt{mse,sum}}\}$, and similarly for the CER-optimal version.

Importantly, note that the `ik`, `cct`, and `cv` bandwidth choices have been deprecated and are no longer supported as part of the `rdrobust` package. The bandwidth choice $\widehat{h}_{\texttt{mse,rd}}$ is an upgraded version of both the `ik` and the `cct` implementations of the MSE-optimal bandwidth selectors discussed in Imbens and Kalyanaraman (2012) and Calonico, Cattaneo, and Titiunik (2014b), respectively. See section 6 for more discussion on backward compatibility. Lastly, the `cv` (cross-validation) bandwidth selection method was removed because it appears to be considerably less popular than plug-in bandwidth selection methods in empirical work, and at present it is not theoretically justified nor easily portable to the new settings considered in the upgraded version of `rdrobust` (for example, inclusion of covariates or clustering).

2.4 RD plots

RD plots are commonly used in RD empirical work, and their main methodological features are discussed in great detail in Calonico, Cattaneo, and Titiunik (2015a). These plots can be easily constructed using the `rdplot` command. The upgraded version of this command now includes two main new features.

- Kernel-weighted polynomial fits with possibly different bandwidths. The `rdplot` command now allows for (global or restricted) weighted polynomial fits using any of the kernel weighting schemes available for estimation and inference in RD designs: uniform, triangular, or Epanechnikov. Furthermore, following the upgrades to `rdrobust` and `rdbwselect`, the `rdplot` command now also allows for possibly different bandwidths on either side of the cutoff when computing the global

polynomial fits. These new options permit the exact graphical presentation of RD point estimation and inference by restricting the support of the running variable to the neighborhood around the cutoff determined by the bandwidth used. See section 5 for syntax details and section 7 for an empirical illustration of this new feature.

- Confidence intervals for local binned fits. The `rdplot` command now allows the user to plot and report CIs for local means within each bin. Specifically, for each bin $j = 1, 2, \ldots, J_n$, the `rdplot` command computes the following CI:

$$\text{CI}_j = \left[\overline{X}_j - \mathcal{T}_{1-\alpha/2} \times \sqrt{S_j^2/N_j} \ , \ \overline{X}_j + \mathcal{T}_{1-\alpha/2} \times \sqrt{S_j^2/N_j} \right]$$

where $\overline{X}_j$ denotes the sample mean in bin j, S_j^2 denotes the sample variance in bin j, N_j denotes the sample size in bin j, and $\mathcal{T}_\alpha$ denotes the $\alpha \in (0,1)$ quantile of the Student's t distribution with $N_j - 1$ degrees of freedom. This formula is justified by preasymptotic inference and as a nonparametric inference procedure, up to smoothing bias, following the results in Cattaneo and Farrell (2013) for partitioning regression methods.

3 The rdrobust command

This section describes the full syntax of the upgraded `rdrobust` command. Whenever possible, we retain the same syntax as in the previous version of this command (Calonico, Cattaneo, and Titiunik 2014a, 2015b).

3.1 Syntax

`rdrobust` *depvar runvar* [*if*] [*in*] [`,` `c(`*cutoff*`)` `p(`*pvalue*`)` `q(`*qvalue*`)`
`deriv(`*dvalue*`)` `fuzzy(`*fuzzyvar* [`sharpbw`]`)` `covs(`*covars*`)` `kernel(`*kernelfn*`)`
`weights(`*weightsvar*`)` `h(`*hvalueL hvalueR*`)` `b(`*bvalueL bvalueR*`)` `rho(`*rhovalue*`)`
`scalepar(`*scaleparvalue*`)` `bwselect(`*bwmethod*`)` `scaleregul(`*scaleregulvalue*`)`
`vce(`*vcemethod*`)` `level(`*level*`)` `all`]

where *depvar* is the dependent variable and *runvar* is the running variable (also known as the score or forcing variable).

Only new options or options that have changed in this version are discussed in section 3.2.

3.2 Options

`fuzzy(`*fuzzyvar* [`sharpbw`]`)` specifies the treatment status variable used to implement fuzzy RD estimation (or fuzzy kink RD if `deriv(1)` is also specified). The default is

sharp RD design. If the `sharpbw` option is set, the fuzzy RD estimation is performed using a bandwidth selection procedure for the sharp RD model. This option is automatically selected if there is perfect compliance at either side of the threshold.

`covs(`*covars*`)` specifies additional covariates to be used for estimation and inference.

`weights(`*weightsvar*`)` specifies the variable used for optional weighting of the estimation procedure. The unit-specific weights multiply the kernel function.

`h(`*hvalueL hvalueR*`)` specifies the main bandwidth, h, to be used on the left and on the right of the cutoff, respectively. If only one value is specified, then this value is used on both sides. If not specified, the bandwidth(s) h is computed by the companion command `rdbwselect`.

`b(`*bvalueL bvalueR*`)` specifies the bias bandwidth, b, to be used on the left and on the right of the cutoff, respectively. If only one value is specified, then this value is used on both sides. If not specified, the bandwidth(s) b is computed by the companion command `rdbwselect`.

`bwselect(`*bwmethod*`)` specifies the bandwidth selection procedure to be used. By default, it computes both h and b, unless ρ is specified, in which case it computes only the h and sets $b = h/\rho$. Implementation and numerical details are given in CCFT. *bwmethod* may be one of the following:

`mserd` specifies one common MSE-optimal bandwidth selector for the RD treatment-effect estimator. `mserd` is the default.

`msetwo` specifies two different MSE-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`msesum` specifies one common MSE-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`msecomb1` specifies min(`mserd`, `msesum`).

`msecomb2` specifies median(`msetwo`, `mserd`, `msesum`) for each side of the cutoff separately.

`cerrd` specifies one common CER-optimal bandwidth selector for the RD treatment-effect estimator.

`certwo` specifies two different CER-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`cersum` specifies one common CER-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`cercomb1` specifies min(`cerrd`, `cersum`).

`cercomb2` specifies median(`certwo`, `cerrd`, `cersum`) for each side of the cutoff separately.

`vce(`*vcemethod*`)` specifies the procedure used to compute the variance–covariance matrix estimator. Implementation and numerical details are given in CCFT. *vcemethod* may be one of the following:

`nn` [*nnmatch*] specifies a heteroskedasticity-robust NN variance estimator with *nnmatch* indicating the minimum number of neighbors to be used. The default is `vce(nn 3)`.

`hc0` specifies a heteroskedasticity-robust HC0 plug-in residuals variance estimator.

`hc1` specifies a heteroskedasticity-robust HC1 plug-in residuals variance estimator.

`hc2` specifies a heteroskedasticity-robust HC2 plug-in residuals variance estimator.

`hc3` specifies a heteroskedasticity-robust HC3 plug-in residuals variance estimator.

`nncluster` *clustervar* [*nnmatch*] specifies a cluster–robust NN variance estimator with *clustervar* indicating the cluster ID variable and *nnmatch* indicating the minimum number of neighbors to be used.

`cluster` *clustervar* specifies a cluster–robust plug-in residuals variance estimator with *clustervar* indicating the cluster ID variable.

3.3 Options removed or deprecated

The following options were removed from the upgraded `rdrobust` command: `delta()`, `cvgrid_min()`, `cvgrid_max()`, `cvgrid_length()`, `cvplot`, and `matches()`.

4 The rdbwselect command

This section describes the full syntax of the upgraded `rdbwselect` command. Whenever possible, we retain the same syntax as in the previous version of this command (Calonico, Cattaneo, and Titiunik 2014a, 2015b).

4.1 Syntax

`rdbwselect` *depvar runvar* [*if*] [*in*] [`, c(`*cutoff*`) p(`*pvalue*`) q(`*qvalue*`)`
`deriv(`*dvalue*`) fuzzy(`*fuzzyvar* [`sharpbw`]`) covs(`*covars*`) kernel(`*kernelfn*`)`
`weights(`*weightsvar*`) bwselect(`*bwmethod*`) scaleregul(`*scaleregulvalue*`)`
`vce(`*vcemethod*`) all`]

where *depvar* is the dependent variable and *runvar* is the running variable (also known as the score or forcing variable).

Only new options or options that have changed in this version are discussed in section 4.2.

4.2 Options

`fuzzy(`*fuzzyvar* [`sharpbw`]`)` specifies the treatment status variable used to implement fuzzy RD estimation (or fuzzy kink RD if `deriv(1)` is also specified). The default is sharp RD design. If the `sharpbw` option is set, the fuzzy RD estimation is performed using a bandwidth selection procedure for the sharp RD model. This option is automatically selected if there is perfect compliance at either side of the threshold.

`covs(`*covars*`)` specifies additional covariates to be used for estimation and inference.

`weights(`*weightsvar*`)` specifies the variable used for optional weighting of the estimation procedure. The unit-specific weights multiply the kernel function.

`bwselect(`*bwmethod*`)` specifies the bandwidth selection procedure to be used. Implementation and numerical details are given in CCFT. *bwmethod* may be one of the following:

`mserd` specifies one common MSE-optimal bandwidth selector for the RD treatment-effect estimator. This is the default.

`msetwo` specifies two different MSE-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`msesum` specifies one common MSE-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`msecomb1` specifies min(`mserd`, `msesum`).

`msecomb2` specifies median(`msetwo`, `mserd`, `msesum`) for each side of the cutoff separately.

`cerrd` specifies one common CER-optimal bandwidth selector for the RD treatment-effect estimator.

`certwo` specifies two different CER-optimal bandwidth selectors (below and above the cutoff) for the RD treatment-effect estimator.

`cersum` specifies one common CER-optimal bandwidth selector for the sum of regression estimates (as opposed to the difference thereof).

`cercomb1` specifies min(`cerrd`, `cersum`).

`cercomb2` specifies median(`certwo`, `cerrd`, `cersum`) for each side of the cutoff separately.

`vce(`*vcemethod*`)` specifies the procedure used to compute the variance–covariance matrix estimator. Implementation and numerical details are given in CCFT. *vcemethod* may be one of the following:

`nn` [*nnmatch*] specifies a heteroskedasticity-robust NN variance estimator with *nnmatch* indicating the minimum number of neighbors to be used. The default is `vce(nn 3)`.

`hc0` specifies a heteroskedasticity-robust HC0 plug-in residuals variance estimator.

`hc1` specifies a heteroskedasticity-robust HC1 plug-in residuals variance estimator.

`hc2` specifies a heteroskedasticity-robust HC2 plug-in residuals variance estimator.

`hc3` specifies a heteroskedasticity-robust HC3 plug-in residuals variance estimator.

`nncluster` *clustervar* [*nnmatch*] specifies a cluster–robust NN variance estimator with *clustervar* indicating the cluster ID variable and *nnmatch* indicating the minimum number of neighbors to be used.

`cluster` *clustervar* specifies a cluster–robust plug-in residuals variance estimator with *clustervar* indicating the cluster ID variable.

4.3 Options removed or deprecated

The following options were removed from the upgraded `rdbwselect` command: `delta()`, `cvgrid_min()`, `cvgrid_max()`, `cvgrid_length()`, `cvplot`, and `matches()`.

5 The rdplot command

This section describes the full syntax of the upgraded `rdplot` command. Whenever possible, we retain the same syntax as in the previous version of this command (Calonico, Cattaneo, and Titiunik 2014a, 2015b).

5.1 Syntax

`rdplot` *depvar runvar* [*if*] [*in*] [`,` `c(`*cutoff*`)` `p(`*pvalue*`)` `kernel(`*kernelfn*`)`
`weights(`*weightsvar*`)` `h(`*hvalueL hvalueR*`)` `nbins(`*nbinsvalueL nbinsvalueR*`)`
`binselect(`*binmethod*`)` `scale(`*scalevalueL scalevalueR*`)` `ci(`*cilevel*`)` `shade`
`support(`*supportvalueL supportvalueR*`)` `genvars` `graph_options(`*gphopts*`)` `hide`]

where *depvar* is the dependent variable and *runvar* is the running variable (also known as the score or forcing variable).

Only new options or options that have changed in this version are discussed in section 5.2.

5.2 Options

`kernel(`*kernelfn*`)` specifies the kernel function used to construct the global polynomial estimators. *kernelfn* may be `triangular`, `uniform`, or `epanechnikov`. The default is `kernel(uniform)` (that is, equal or no weighting to all observations on the support of the kernel).

`weights(`*weightsvar*`)` specifies the variable used for optional weighting of the estimation procedure. The unit-specific weights multiply the kernel function.

`h(`*hvalueL hvalueR*`)` specifies the main bandwidth, h, to be used on the left and on the right of the cutoff, respectively. If only one value is specified, then this value is used on both sides. If two bandwidths are specified, the first bandwidth is used for the data below the cutoff and the second bandwidth is used for the data above the cutoff. If not specified, it is chosen to span the full support of the data.

`nbins(`*nbinsvalueL nbinsvalueR*`)` specifies the number of bins used to the left of the cutoff (denoted J_-) and the number of bins used to the right of the cutoff (denoted J_+), respectively. If only one value is specified, then this value is used on both sides. If not specified, J_- and J_+ are estimated using the `binselect()` option.

`scale(`*scalevalueL scalevalueR*`)` specifies a multiplicative factor to be used with the optimal number of bins selected. Specifically, for the control and treated units, the number of bins used will be $\langle scalevalueL \times \widehat{J}_{-,n} \rangle$ and $\langle scalevalueR \times \widehat{J}_{+,n} \rangle$, respectively. If only one value is specified, then this value is used on both sides. The default is `scale(1 1)`.

`ci(`*cilevel*`)` specifies the optional graphical option to display CIs of *cilevel* coverage for each bin.

`shade` specifies the optional graphical option to replace CIs with shaded areas.

`support(`*supportvalueL supportvalueR*`)` specifies an optional extended support of the running variable to be used in the construction of the bins. The default is the sample range.

`genvars` generates the following new variables that store results:

`rdplot_id` stores a unique bin ID for each observation. Negative natural numbers are assigned to observations to the left of the cutoff, and positive natural numbers are assigned to observations to the right of the cutoff.

`rdplot_N` stores the number of observations in the corresponding bin for each observation.

`rdplot_min_bin` stores the lower end value of the bin for each observation.

`rdplot_max_bin` stores the upper end value of the bin for each observation.

`rdplot_mean_bin` stores the middle point of the corresponding bin for each observation.

`rdplot_mean_x` stores the sample mean of the running variable within the corresponding bin for each observation.

`rdplot_mean_y` stores the sample mean of the outcome variable within the corresponding bin for each observation.

`rdplot_se_y` stores the standard deviation of the mean of the outcome variable within the corresponding bin for each observation.

`rdplot_ci_l` stores the lower end value of the confidence interval for the sample mean of the outcome variable within the corresponding bin for each observation.

`rdplot_ci_r` stores the upper end value of the confidence interval for the sample mean of the outcome variable within the corresponding bin for each observation.

`rdplot_hat_y` stores the predicted value of the outcome variable given by the global polynomial estimator.

5.3 Options removed or deprecated

The following options were removed from the upgraded `rdplot` command: `numbinl()`, `numbinr()`, `scalel()`, `scaler()`, `generate()`, `lowerend()`, and `upperend()`.

6 Backward compatibility

Here we discuss backward compatibility with the previous version of `rdrobust` for Stata and the R software package. We organize the presentation in terms of the three main functions.

- `rdrobust`. For a given choice of bandwidth(s), this command is backward compatible by default when additional covariates are not included. Point estimators are identical in all cases, but variance estimators may slightly change in some cases because of internal numerical and implementation upgrades. The previous version of this command included only two variance estimators: NN and plug-in residuals without weighting (HC0). The upgraded version of this command continues to offer these two choices, but the residuals are computed in a slightly different way to improve speed and to give further compatibility with linear least-squares regression-based methods. This new way of computing residuals may generate numerical changes in the following cases:

  1. Nearest-neighbor variance estimation. This variance estimator (which is the default) will be identical to the previous version of the `rdrobust` command in the absence of ties in the running variable X_i but will change slightly when ties occur. If the running variable X_i is truly continuously distributed, there should be no ties, and hence the default option of `rdrobust` is fully backward compatible. When ties occur, the new optimized NN procedure will result in a much faster but slightly different variance estimator.
  2. Plug-in residuals variance estimation. The upgraded version of `rdrobust` includes a new variance estimator based on plug-in residuals, which is not backward compatible. This new version computes all residuals at the cutoff point and is therefore much faster. This approach also mimics exactly linear least-squares methods. In contrast, the previous version of this command computed plug-in residuals using nonparametric methods and hence evaluated the predicted values at different values near the cutoff.

- `rdbwselect`. This command is not backward compatible. Given the many new data-driven bandwidth options, and the several numerical and implementation upgrades, we were forced to redo the command completely. To allow for backward compatibility, we do provide its previous version (called `rdbwselect_2014`) as part of the upgraded `rdrobust` package. This previous (deprecated) version may be used to obtain data-driven bandwidths, which then can be inputted manually to `rdrobust` to obtain backward compatible RD estimates and inference procedures.

  As mentioned above, now two distinct approaches are available for bandwidth selection in fuzzy RD design settings:

  1. Select the bandwidth(s) focusing only on the sharp RD intention-to-treat estimator entering the numerator of the fuzzy RD treatment-effect estimator.
  2. Select the bandwidth(s) focusing only on the actual fuzzy RD treatment-effect estimator, that is, the ratio of reduced-form RD estimators.

  In the previous version of `rdbwselect`, only the first approach was available, but now both alternatives are implemented in the upgraded version. In fuzzy RD contexts, `rdbwselect` will use the second approach by default when two-sided imperfect compliance is present but will otherwise use the first approach. To force `rdbwselect` to use the first approach even when two-sided imperfect compliance is present, use the additional option `sharpbw` within the `fuzzy()` option when specifying the fuzzy variable. Because `rdrobust` selects automatic data-driven bandwidths using `rdbwselect`, the above remarks and options apply to the upgraded `rdrobust` command as well.

- `rdplot`. This command is fully backward compatible. All upgrades are included in addition to the features previously available in this command. Notice that the options for this command have been reorganized to improve consistency and homogeneity with `rdrobust` and `rdbwselect`.

7 Illustration of new methods

We illustrate our commands using the same dataset already used in Calonico, Cattaneo, and Titiunik (2014a, 2015b), where the previous version of the `rdrobust`, `rdbwselect`, and `rdplot` commands are introduced and discussed. While this facilitates the discussion and comparison, in this section, we focus almost exclusively on the new features available in the new version of the package. Whenever possible, we briefly compare and highlight any substantive changes in implementation.

`rdrobust_senate.dta` contains the outcome variable (Y_i), running variable (X_i), and four additional covariates ($\mathbf{Z}_i$) constructed by Cattaneo, Frandsen, Titiunik (2015). The illustration focuses on party advantages in U.S. Senate elections for the period 1914–2010, using a sharp RD design with the unit of analysis being the state at a given election. In this section, we focus on the running variable used to analyze the RD effect of the Democratic party winning a U.S. Senate seat on the vote share obtained in the following election for that same seat.

First, we load the database and present summary statistics:

```
. use rdrobust_senate.dta, clear
. sum vote margin class termshouse termssenate population, sep(2)
```

Variable	Obs	Mean	Std. Dev.	Min	Max
vote	1,297	52.66627	18.12219	0	100
margin	1,390	7.171159	34.32488	-100	100
class	1,390	2.023022	.8231983	1	3
termshouse	1,108	1.436823	2.357133	0	16
termssenate	1,108	4.555957	3.720294	1	20
population	1,390	3827919	4436950	78000	3.73e+07

The database also includes two other variables, `state` and `year`, which record the state and year of each election. The running variable is `margin`, which ranges from -100 to 100 and records the Democratic party's margin of victory in the statewide election for a given U.S. Senate seat, defined as the vote share of the Democratic party minus the vote share of its strongest opponent. The outcome variable is `vote`, which ranges from 0 to 100 and records the Democratic vote share in the following election for the same seat (that is, six years later). The cutoff is normalized to $\overline{x} = 0$. The additional covariates are `class`, `termshouse`, `termssenate`, and `population`. The variable `class` identifies the electoral class each Senate seat belongs to (this indicates which of the possible three electoral cycles each seat is in); the variables `termshouse` and `termssenate` capture the experience of the Democratic candidate by recording the cumulative number of terms previously served in U.S. House and Senate, respectively; and the variable `population` records the population of the Senate seat's state.

7.1 RD plots with CIs

As mentioned above, the `rdplot` command is fully backward compatible. Hence, in this section, we illustrate only its new features. One of these new features is the inclusion of CIs for the binned sample mean (or partitioning) estimator. This additional option is useful in presenting and assessing the variability of the RD design.

```
. rdplot vote margin, binselect(es) ci(95)
>        graph_options(title("RD Plot: U.S. Senate Election Data")
>                      ytitle(Vote Share in Election at time t+2)
>                      xtitle(Vote Share in Election at time t)
>                      graphregion(color(white)))
RD Plot with evenly spaced number of bins using spacings estimators.
```

Cutoff c = 0	Left of c	Right of c		
Number of obs	595	702	Number of obs	= 1297
Eff. Number of obs	595	702	Kernel	= Uniform
Order poly. fit (p)	4	4		
BW poly. fit (h)	100.000	100.000		
Number of bins scale	1.000	1.000		

Outcome: vote. Running variable: margin.

	Left of c	Right of c
Bins selected	8	9
Average bin length	12.500	11.111
Median bin length	12.500	11.111
IMSE-optimal bins	8	9
Mimicking Var. bins	15	35
Rel. to IMSE-optimal:		
Implied scale	1.000	1.000
WIMSE var. weight	0.500	0.500
WIMSE bias weight	0.500	0.500

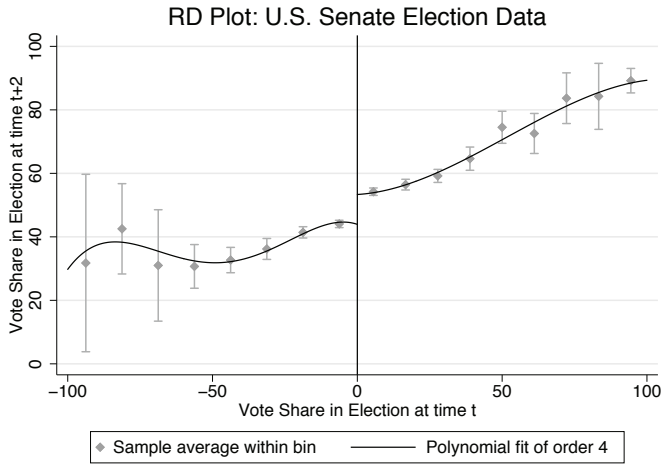

Figure 1. IMSE-optimal evenly spaced RD plot with CIs

Figure 1 provides CIs using the IMSE-optimal number of bins choice for evenly spaced bins on the support of the running variable. In theory, the CIs presented may exhibit a first-order smoothing bias, so in applications it may be useful to select a larger number

of bins when including these CIs. One way of doing so is to use the mimicking-variance number of bins choice (for example, using the `binselect(esmv)` option), which we do not present here to conserve space. Alternatively, the researcher can simply "undersmooth" (that is, choose a larger number of bins) manually either by scaling up the IMSE-optimal choice with the `scale()` option or by setting the number of bins directly with the `nbins()` option.

We illustrate the other new features of the upgraded `rdplot` command in the following subsections.

7.2 Default results and backward compatibility

The upgraded `rdrobust` command works in exactly the same way as before. For example, using only the outcome and running variables, we obtain the following results with its default options. (We will refer to these default results several times in the upcoming subsections.)

```
. rdrobust vote margin
Sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1297
                   |                                 BW type       =      mserd
     Number of obs |       595         702           Kernel        = Triangular
Eff. Number of obs |       359         322           VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.708      17.708
       BW bias (b) |    27.984      27.984
         rho (h/b) |     0.633       0.633
Outcome: vote. Running variable: margin.
-------------------------------------------------------------------------------
            Method |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------------+-----------------------------------------------------------
      Conventional |      7.416     1.4604   5.0782   0.000     4.55378     10.2783
            Robust |          -          -   4.3095   0.000     4.09441     10.9255
-------------------------------------------------------------------------------
```

The above results are not numerically identical to those reported in Calonico, Cattaneo, and Titiunik (2014a, 2015b). The differences are due to the choice of bandwidths because, as explained above, the new upgraded `rdbwselect` command is not backward compatible. Recall that, by default, the `rdrobust` command uses the companion command `rdbwselect` to select the bandwidths optimally whenever they are not specified by the user.

Nonetheless, the upgraded `rdrobust` command is backward compatible given the choice of bandwidths. To see this, we set the bandwidths manually to match those reported in Calonico, Cattaneo, and Titiunik (2014a, 2015b), which gives the following results:

```
. rdrobust vote margin, h(16.79369) b(27.43745)
Sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1297
-------------------+----------------------           BW type       =     Manual
     Number of obs |       595         702           Kernel        = Triangular
Eff. Number of obs |       343         310           VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    16.794      16.794
       BW bias (b) |    27.437      27.437
         rho (h/b) |     0.612       0.612
Outcome: vote. Running variable: margin.
-------------------------------------------------------------------------------
            Method |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------------+-----------------------------------------------------------
      Conventional |      7.4253     1.4954    4.9656  0.000     4.49446      10.3561
            Robust |           -          -    4.2675  0.000     4.06975      10.9833
-------------------------------------------------------------------------------
```

These results are equal to those reported in Calonico, Cattaneo, and Titiunik (2014a, 2015b).

7.3 Using RD plots to present treatment effects

We already showed how to incorporate local CIs in RD plots. Now we show how to use these plots to present the RD treatment effects visually, which is now possible using the new features of the `rdplot` command.

We use the default MSE-optimal RD estimate presented above, which is obtained via the command `rdrobust vote margin`. Recall that by default `rdrobust` uses a triangular kernel with a common bandwidth on both sides of the cutoff. To plot the point estimate, we can use the `rdplot` command after setting the `p()`, `kernel()`, and `h()` options appropriately.

```
. quietly rdrobust vote margin
. rdplot vote margin if -e(h_l)<= margin & margin <= e(h_r),
>        binselect(esmv) kernel(triangular) h(`e(h_l)´ `e(h_r)´) p(1)
>        graph_options(title("RD Plot: U.S. Senate Election Data")
>                      ytitle(Vote Share in Election at time t+2)
>                      xtitle(Vote Share in Election at time t)
>                      graphregion(color(white)))
RD Plot with evenly spaced mimicking variance number of bins using spacings
> estimators.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	359	322
Eff. Number of obs	359	322
Order poly. fit (p)	1	1
BW poly. fit (h)	17.708	17.708
Number of bins scale	1.000	1.000

Number of obs = 681
Kernel = Triangular

Outcome: vote. Running variable: margin.

	Left of c	Right of c
Bins selected	16	18
Average bin length	1.090	0.983
Median bin length	1.090	0.983
IMSE-optimal bins	7	7
Mimicking Var. bins	16	18
Rel. to IMSE-optimal:		
Implied scale	2.286	2.571
WIMSE var. weight	0.077	0.056
WIMSE bias weight	0.923	0.944

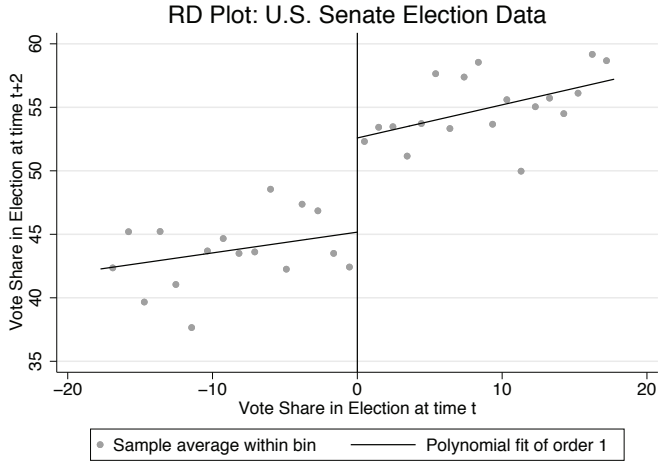

Figure 2. RD plot of treatment effect

Figure 2 is constructed using the upgraded `rdplot` command by restricting the support to the neighborhood around the cutoff defined by the choice of bandwidth h (in this example, equal on both sides). We then set the (global) fit in the RD plot to match the local polynomial point estimation conducted by `rdrobust` in that neighborhood; that is, we choose $p = 1$, $K(\cdot)$ to be the triangular kernel, and h to be the bandwidth used in the estimation, shown above. The resulting polynomial fit represents the RD point estimator exactly.

For graphical presentation purposes, we also selected in `rdplot` a mimicking-variance number of bins to exhibit the variability of the data within the window around the cutoff determined by the data-driven choice of bandwidth. In this example, the vertical distance between the two weighted linear polynomial fits is exactly 7.416, as reported in the previous section. By increasing the number of bins using the `nbins()` option, the RD plot can be used to exhibit the actual raw data instead of the average values of the outcome variable within each bin.

7.4 Robust bias-corrected inference with covariates and clustering

In this section, we illustrate the new features of the upgraded `rdrobust` command. We show how to incorporate covariates in estimation and inference and how to use cluster–robust variance estimators (with or without additional covariates).

First, we incorporate the covariates `class`, `termshouse`, and `termssenate`, keeping the neighborhood around the cutoff constant. That is, we use the same bandwidths obtained above via the default command: `rdrobust vote margin`.

```
. qui rdrobust vote margin
. local len = `e(ci_r_rb)´ - `e(ci_l_rb)´
. rdrobust vote margin, covs(class termshouse termssenate)
>          h(`e(h_l)´ `e(h_r)´) b(`e(b_l)´ `e(b_r)´)
Covariate-adjusted sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1108
-------------------+----------------------           BW type       =     Manual
     Number of obs |       491         617           Kernel        = Triangular
Eff. Number of obs |       309         280           VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.708      17.708
       BW bias (b) |    27.984      27.984
         rho (h/b) |     0.633       0.633
Outcome: vote. Running variable: margin.
-------------------------------------------------------------------------------
            Method |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------------+-----------------------------------------------------------
      Conventional |     6.8595     1.4165   4.8426   0.000     4.08322      9.63574
            Robust |          -          -   4.1911   0.000     3.75238       10.345
-------------------------------------------------------------------------------
Covariate-adjusted estimates. Additional covariates included: 3
. display "CI length change: "
> round(((`e(ci_r_rb)´-`e(ci_l_rb)´)/`len´-1)*100,.01) "%"
CI length change: -3.49%
```

The results above set the bandwidths manually (to be equal to the MSE-optimal bandwidths without covariates) and also compute the percentage change in the interval length of the robust bias-corrected CIs, because of the inclusion of the three additional covariates. Specifically, in this illustration, the CI length shrinks by 3.49%. The MSE-optimal point estimate (without covariates) of 7.416 changes to 6.860 when the additional covariates are included (though this change is statistically indistinguishable from 0 at conventional levels).

Second, we incorporate the same additional covariates but let `rdrobust` select the optimal bandwidths, which is done via `rdbwselect`. The data-driven bandwidths are chosen to be MSE optimal and equal on both sides of the cutoff by default.

```
. qui rdrobust vote margin
. local len = `e(ci_r_rb)´ - `e(ci_l_rb)´
. rdrobust vote margin, covs(class termshouse termssenate)
Covariate-adjusted sharp RD estimates using local polynomial regression.
```

Cutoff c = 0	Left of c	Right of c		
Number of obs	491	617	Number of obs	1108
Eff. Number of obs	313	283	BW type	mserd
Order est. (p)	1	1	Kernel	Triangular
Order bias (q)	2	2	VCE method	NN
BW est. (h)	17.987	17.987		
BW bias (b)	28.943	28.943		
rho (h/b)	0.621	0.621		

```
Outcome: vote. Running variable: margin.
```

Method	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
Conventional	6.8514	1.4081	4.8656	0.000	4.09148	9.61125
Robust	-	-	4.1999	0.000	3.72856	10.2537

```
Covariate-adjusted estimates. Additional covariates included: 3
. display "CI length change: "
> round(((`e(ci_r_rb)´-`e(ci_l_rb)´)/`len´-1)*100,.01) "%"
CI length change: -4.48%
```

In this illustration, the point estimators and the bandwidth choices change only slightly [from $\widetilde{\tau}(h) = 6.860$ and $h = 17.708$ to $\widetilde{\tau}(h) = 6.851$ and $h = 17.987$], and the interval length reduction increases because of the inclusion of the additional covariates in both point estimation and bandwidth selection (from 3.49% to 4.48%).

Third, we show that, as is well known in the literature, including covariates does not always lead to improved precision. For example, if the covariates are irrelevant, they can even increase the length of the CIs. In our illustration, the covariate `population` gives an example.

```
. qui rdrobust vote margin
. local len = `e(ci_r_rb)´ - `e(ci_l_rb)´
```

```
. rdrobust vote margin, covs(population)
Covariate-adjusted sharp RD estimates using local polynomial regression.

      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1297
                   |                                 BW type       =      mserd
     Number of obs |       595         702           Kernel        = Triangular
Eff. Number of obs |       359         320           VCE method    =         NN
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.585      17.585
       BW bias (b) |    27.857      27.857
         rho (h/b) |     0.631       0.631

Outcome: vote. Running variable: margin.

------------------------------------------------------------------------------
            Method |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------------+----------------------------------------------------------
      Conventional |     7.4376     1.4654   5.0754   0.000     4.56545      10.3097
            Robust |          -          -   4.3102   0.000     4.10714      10.9573
------------------------------------------------------------------------------

Covariate-adjusted estimates. Additional covariates included: 1

. display "CI length change: "
> round(((`e(ci_r_rb)´-`e(ci_l_rb)´)/`len´-1)*100,.01) "%"
CI length change: .28%
```

As seen above, conducting covariate-adjusted RD inference with `population` as the additional covariate slightly increases the length of the resulting CIs (by roughly a quarter of a percentage point).

Fourth, as discussed above, the covariates will not affect the consistency of the RD treatment-effect estimator if they are "balanced" in the appropriate sense. For the case of sharp RD designs, "balanced" means that they should have equal conditional expectations at the cutoff. For the case of kink RD designs, "balanced" means that they should have equal first derivatives of the conditional expectations at the cutoff. This can be tested empirically.

```
. local covs "class termshouse termssenate population"
. local num: list sizeof covs
. mat balance = J(`num´,2,.)
. local row = 1
. foreach z in `covs´ {
  2.     qui rdrobust `z´ margin
  3.         mat balance[`row´,1] = round(e(tau_cl),.001)
  4.         mat balance[`row´,2] = round(e(pv_rb),.001)
  5.         local ++row
  6. }
. mat rownames balance = `covs´
. mat colnames balance = "RD Effect" "Robust p-val"
. mat lis balance
balance[4,2]
             RD Effect  Robust p-val
      class         -.021          .897
 termshouse         -.173          .561
termssenate         -.192          .901
 population    -318455.26          .634
```

Based on the empirical results above, we find that all four additional covariates have an RD treatment effect indistinguishable from 0 at conventional significance levels. In other words, we cannot reject the null hypothesis of equal conditional expectations at the cutoff. Notice that we can also use RD plots to show covariate balance at the cutoff, but we do not present these additional results to conserve space.

Fifth, the upgraded `rdrobust` command also allows for cluster–robust variance estimation, as does the underlying upgraded `rdbwselect` command used to compute data-driven bandwidth selectors. This is illustrated using the Senate data as follows, where we cluster at the `state` level using NN methods to construct the estimated residuals (recall that by default three matches per observation are used).

```
. rdrobust vote margin, vce(nncluster state)

Sharp RD estimates using local polynomial regression.

      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1297
                   |                                 BW type       =      mserd
     Number of obs |       595         702           Kernel        = Triangular
Eff. Number of obs |       359         320           VCE method    =  NNcluster
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    17.509      17.509
       BW bias (b) |    27.032      27.032
         rho (h/b) |     0.648       0.648
Number of clusters |        50          50

Outcome: vote. Running variable: margin.

------------------------------------------------------------------------------
      Method |     Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
Conventional |     7.4221     1.5225    4.8750   0.000     4.43811      10.4061
      Robust |          -          -    4.2659   0.000     4.09109      11.0456
------------------------------------------------------------------------------

Std. Err. adjusted for clusters in state
```

In this case, the robust bias-corrected CIs change from $[4.094, 10.926]$, the (default) heteroskedasticity-robust interval reported previously, to the cluster–robust interval $[4.091, 11.046]$.

To end this section, we provide one final illustration using i) covariate-adjustment, ii) cluster–robust variance estimation, and iii) MSE-optimal bandwidth selection with (possibly) different bandwidths on either side of the cutoff.

```
. rdrobust vote margin, covs(class termshouse termssenate)
>                       bwselect(msetwo) vce(nncluster state)
Covariate-adjusted sharp RD estimates using local polynomial regression.
      Cutoff c = 0 | Left of c  Right of c           Number of obs =       1108
-------------------+----------------------           BW type       =     msetwo
     Number of obs |       491         617           Kernel        = Triangular
Eff. Number of obs |       274         310           VCE method    =  NNcluster
    Order est. (p) |         1           1
    Order bias (q) |         2           2
       BW est. (h) |    14.661      20.893
       BW bias (b) |    24.458      37.338
         rho (h/b) |     0.599       0.560
Number of clusters |        48          50
Outcome: vote. Running variable: margin.
-------------------------------------------------------------------------------
            Method |     Coef.   Std. Err.      z     P>|z|     [95% Conf. Interval]
-------------------+-----------------------------------------------------------
      Conventional |     6.8072     1.3696    4.9704   0.000     4.12297      9.49153
            Robust |          -          -    4.6153   0.000     4.13522      10.2398
-------------------------------------------------------------------------------
Covariate-adjusted estimates. Additional covariates included: 3
Std. Err. adjusted for clusters in state
```

In this final case, we observe that the two one-sided MSE-optimal bandwidths are actually quite distinct from their common bandwidth counterpart. To be clear, these bandwidth selectors also account for the additional covariates and the clustering structure of the matrix of variances and covariances, but the numerical results show that the two bandwidths are different (for example, $\widehat{h}_{\mathtt{l}} = 14.661$ and $\widehat{h}_{\mathtt{r}} = 20.893$). The point estimate is, nonetheless, quite stable [$\widehat{\tau}(\widehat{h}_{\mathtt{l}}, \widehat{h}_{\mathtt{r}}) = 6.807$]. Finally, the robust bias-corrected covariate-adjusted cluster–robust CIs are $[4.135, 10.240]$, quite similar to the cluster–robust version reported previously.

7.5 Data-driven bandwidth selectors

As already implicitly illustrated above, the upgraded `rdbwselect` command includes several new features, such as covariate-adjusted and cluster–robust bandwidth selection. Furthermore, although not used above explicitly to conserve space, several other data-driven bandwidth selectors are now available.

In this section, we present one empirical result exhibiting all the available data-driven bandwidth selectors, in the context of our empirical illustration. We do not include additional covariates or consider cluster–robust variance estimation only for simplicity, because the main goal is to discuss the different bandwidth selectors available.

```
. rdbwselect vote margin, all
Bandwidth estimators for sharp RD local polynomial regression.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Min of margin	-100.000	0.036
Max of margin	-0.079	100.000
Order est. (p)	1	1
Order bias (q)	2	2

Number of obs = 1297
Kernel = Triangular
VCE method = NN

Outcome: vote. Running variable: margin.

	BW est. (h)		BW bias (b)	
Method	Left of c	Right of c	Left of c	Right of c
mserd	17.708	17.708	27.984	27.984
msetwo	16.154	18.009	27.096	29.205
msesum	18.326	18.326	31.280	31.280
msecomb1	17.708	17.708	27.984	27.984
msecomb2	17.708	18.009	27.984	29.205
cerrd	12.374	12.374	27.984	27.984
certwo	11.288	12.585	27.096	29.205
cersum	12.806	12.806	31.280	31.280
cercomb1	12.374	12.374	27.984	27.984
cercomb2	12.374	12.585	27.984	29.205

The output shows all the different bandwidth selectors available for estimation and inference in RD designs. These options are also available in the case of covariate-adjustment or cluster–robust variance estimation. The first group considers MSE-optimal bandwidths, while the second group considers CER-optimal bandwidths, both following the methodology discussed in previous sections.

Among these choices, the most useful ones are i) `mserd` for MSE-optimal point estimation using a common bandwidth on both sides of the cutoff, ii) `msetwo` for MSE-optimal point estimation using two distinct common bandwidths on either side of the cutoff, iii) `cerrd` for robust bias-corrected CIs with faster coverage error decay rates using a common bandwidth on both sides of the cutoff, and iv) `certwo` for robust bias-corrected CIs with faster coverage error decay rates using two distinct common bandwidths on either side of the cutoff. The other options are useful for regularization and sensitivity analysis purposes.

Finally, recall that the results above include regularization, as introduced in Imbens and Kalyanaraman (2012) but implemented as discussed in Calonico, Cattaneo, and Titiunik (2014a, 2014b, 2015b) and the corresponding supplemental appendix. Including this regularization term always leads to smaller bandwidths; it can be modified or removed with the `scaleregul()` option. In particular, it can be removed by simply adding `scaleregul(0)`.

7.6 Other examples and RD designs

The companion replication file (`rdrobust_illustration.do`) includes the syntax of all the examples discussed above, as well as the following additional examples:

1. `rdrobust vote margin, kernel(uniform) vce(cluster state)`
   Robust bias-corrected inference using a uniform kernel and clustering with plug-in residuals at `state` level. This is the command that produces the closest results to those obtained using the Stata built-in `regress` command with clustering to construct the RD point estimator. (Without clustering, the most similar results to those obtained with `regress` are obtained using the `vce(hc1)` option.)

2. `rdrobust vote margin, bwselect(certwo) vce(hc3)`
   Robust bias-corrected inference using the CER-optimal bandwidth choice, allowing for a different bandwidth on each side of the cutoff, and HC3 heteroskedasticity-robust variance estimation.

3. `rdrobust vote margin, h(12 15) b(18 20)`
   Robust bias-corrected inference with user-chosen bandwidths $(h_{\mathtt{l}}, h_{\mathtt{r}}) = (12, 15)$ and $(b_{\mathtt{l}}, b_{\mathtt{r}}) = (18, 20)$.

4. `rdrobust vote margin, covs(class) bwselect(cerrd) scaleregul(0) rho(1)`
   Robust bias-corrected inference using covariate adjustment with a single covariate (`class`), CER-optimal common bandwidth selector, no regularization, and $h = b$ (that is, $\rho = 1$).

5. `rdbwselect vote margin, kernel(uniform) vce(cluster state) all`
   All data-driven bandwidth selectors using uniform kernel and clustering with plug-in residuals at `state` level.

6. `rdbwselect vote margin, covs(class) bwselect(msetwo) vce(hc2) all`
   MSE-optimal bandwidth selectors on either side of the cutoff adjusting by covariate `class` and using HC2 heteroskedasticity-robust variance estimation.

Finally, we discuss how to implement other RD designs. Let `y` be the outcome variable, `t` the treatment status variable, `x` the running variable, `z` a "preintervention" covariate, and `cid` a cluster ID variable.

1. `rdrobust y x, deriv(1) covs(z) vce(nncluster cid)`
   Sharp kink RD with additional covariates and clustering.

2. `rdrobust y x, fuzzy(t) covs(z) vce(nncluster cid)`
   Fuzzy RD with additional covariates and clustering.

3. `rdrobust y x, fuzzy(t) deriv(1) covs(z) vce(nncluster cid)`
   Fuzzy kink RD with additional covariates and clustering.

8 Conclusion

In this article, we discussed a major upgrade to the `rdrobust` package for Stata and R, which provide general-purpose software for regression-discontinuity designs. The main new features of this upgraded version are i) covariate-adjusted bandwidth selection, point estimation, and robust bias-corrected inference, ii) clustered-consistent bandwidth selection, point estimation, and robust bias-corrected inference, iii) weighted global polynomial fits and pointwise confidence bands in RD plots, and iv) several new bandwidth selection methods, including different bandwidths for control and treatment groups, CER optimal bandwidths, and optimal bandwidth for fuzzy designs. We provided a detailed account of all technical and methodological results implemented in CCFT and its supplemental appendix.

A companion R package with the same functionality and syntax is also available.

9 Acknowledgments

We thank Quentin Brummet, David Drukker, Brian Jacob, Max Kapustin, Louis-Pierre Lepage, Xinwei Ma, Zhuan Pei, Vincent Pons, and Gonzalo Vazquez-Bare for useful comments and discussions that improved this manuscript as well as our implementations. We also thank an anonymous reviewer for useful comments on an early version of this paper. Cattaneo gratefully acknowledges financial support from the National Science Foundation through grants SES-1357561 and SES-1459931. Titiunik gratefully acknowledges financial support from the National Science Foundation through grant SES-1357561.

10 References

Abadie, A., and G. W. Imbens. 2008. Estimation of the conditional variance in paired experiments. *Annales d'Économie et de Statistique* 91/92: 175–187.

Bartalotti, O., and Q. Brummet. 2017. Regression discontinuity designs with clustered data. In *Advances in Econometrics: Vol. 38—Regression Discontinuity Designs: Theory and Applications*, ed. M. D. Cattaneo and J. C. Escanciano, 383–420. Bingley, UK: Emerald.

Calonico, S., M. D. Cattaneo, and M. H. Farrell. 2016a. Coverage error optimal confidence intervals for regression discontinuity designs. Working Paper, University of Michigan. http://www-personal.umich.edu/∼cattaneo/papers/Calonico-Cattaneo-Farrell_2016_wp.pdf.

———. Forthcoming. On the effect of bias estimation on coverage accuracy in nonparametric inference. *Journal of the American Statistical Association*.

Calonico, S., M. D. Cattaneo, M. H. Farrell, and R. Titiunik. 2016b. Regression discontinuity designs using covariates. Working Paper, University of Michi-

gan. http://www-personal.umich.edu/~cattaneo/papers/Calonico-Cattaneo-Farrell-Titiunik_2016_wp.pdf.

Calonico, S., M. D. Cattaneo, and R. Titiunik. 2014a. Robust data-driven inference in the regression-discontinuity design. *Stata Journal* 14: 909–946.

———. 2014b. Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica* 82: 2295–2326.

———. 2015a. Optimal data-driven regression discontinuity plots. *Journal of the American Statistical Association* 110: 1753–1769.

———. 2015b. rdrobust: An R package for robust nonparametric inference in regression-discontinuity designs. *R Journal* 7: 38–51.

Cameron, A. C., and D. L. Miller. 2015. A practitioner's guide to cluster–robust inference. *Journal of Human Resources* 50: 317–372.

Card, D., D. S. Lee, Z. Pei, and A. Weber. 2015. Inference on causal effects in a generalized regression kink design. *Econometrica* 83: 2453–2483.

Cattaneo, M. D., and J. C. Escanciano, eds. 2017. *Regression Discontinuity Designs: Theory and Applications (Advances in Econometrics, vol. 38)*. Bingley, UK: Emerald.

Cattaneo, M. D., and M. H. Farrell. 2013. Optimal convergence rates, Bahadur representation, and asymptotic normality of partitioning estimators. *Journal of Econometrics* 174: 127–143.

Cattaneo, M. D., B. R. Frandsen, and R. Titiunik. 2015. Randomization inference in the regression discontinuity design: An application to party advantages in the U.S. Senate. *Journal of Causal Inference* 3: 1–24.

Cattaneo, M. D., M. Jansson, and X. Ma. 2017. rddensity: Manipulation testing based on density discontinuity. https://sites.google.com/site/rdpackages/rddensity.

Cattaneo, M. D., R. Titiunik, and G. Vazquez-Bare. Forthcoming. Comparing inference approaches for RD designs: A reexamination of the effect of head start on child mortality. *Journal of Policy Analysis and Management*.

Cattaneo, M. D., and G. Vazquez-Bare. 2016. The choice of neighborhood in regression discontinuity designs. *Observational Studies* 2: 134–146.

Imbens, G. W., and K. Kalyanaraman. 2012. Optimal bandwidth choice for the regression discontinuity estimator. *Review of Economic Studies* 79: 933–959.

Imbens, G. W., and T. Lemieux. 2008. Regression discontinuity designs: A guide to practice. *Journal of Econometrics* 142: 615–635.

Lee, D. S., and T. Lemieux. 2010. Regression discontinuity designs in economics. *Journal of Economic Literature* 48: 281–355.

MacKinnon, J. G. 2013. Thirty years of heteroskedasticity-robust inference. In *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis: Essays in Honor of Halbert L. White Jr.*, ed. X. Chen and N. R. Swanson, 437–461. New York: Springer.

Müller, H.-G., and U. Stadtmuller. 1987. Estimation of heteroscedasticity in regression analysis. *Annals of Statistics* 15: 610–625.

About the authors

Sebastian Calonico is an assistant professor of economics at the University of Miami.

Matias D. Cattaneo is a professor of economics and a professor of statistics at the University of Michigan.

Max H. Farrell is an assistant professor of econometrics and statistics and is a John E. Jeuck faculty fellow at the University of Chicago Booth School of Business.

Rocío Titiunik is the James Orin Murfin associate professor of political science at the University of Michigan.