The Stata Journal (2019)
19, Number 1, pp. 210–245 DOI: 10.1177/1536867X19830919

Power calculations for regression-discontinuity designs

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

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

Gonzalo Vazquez-Bare
University of California, Santa Barbara
Santa Barbara, CA
gvazquez@econ.ucsb.edu

Abstract. In this article, we introduce two commands, `rdpow` and `rdsampsi`, that conduct power calculations and survey sample selection when using local polynomial estimation and inference methods in regression-discontinuity designs. `rdpow` conducts power calculations using modern robust bias-corrected local polynomial inference procedures and allows for new hypothetical sample sizes and bandwidth selections, among other features. `rdsampsi` uses power calculations to compute the minimum sample size required to achieve a desired level of power, given estimated or user-supplied bandwidths, biases, and variances. Together, these commands are useful when devising new experiments or surveys in regression-discontinuity designs, which will later be analyzed using modern local polynomial techniques for estimation, inference, and falsification. Because our commands use the community-contributed (and R) package `rdrobust` for the underlying bandwidths, biases, and variances estimation, all the options currently available in `rdrobust` can also be used for power calculations and sample-size selection, including preintervention covariate adjustment, clustered sampling, and many bandwidth selectors. Finally, we also provide companion R functions with the same syntax and capabilities.

Keywords: st0554, rdpow, rdsampsi, regression-discontinuity designs, power calculations, local polynomial methods

1 Introduction

Power calculations are used when designing experiments and field work in various disciplines in social, behavioral, and medical sciences. Recently, many empirical researchers have designed and implemented surveys using regression-discontinuity (RD) designs. See Imbens and Lemieux (2008); Lee and Lemieux (2010); Calonico, Cattaneo, and Titiunik (2015a); Cattaneo and Escanciano (2017); Cattaneo, Titiunik, and Vazquez-Bare (2017); Cattaneo, Idrobo, and Titiunik (Forthcoming a,b); and references therein for introductions to RD designs. In this article, we introduce two commands (and companion R functions) specifically developed to conduct power calculations and survey sample selection when using RD local polynomial estimation and inference methods.

There are two main approaches to interpret and analyze RD designs in modern applied work. The first approach is based on continuity or smoothness assumptions and typically uses local polynomial methods (for example, Calonico, Cattaneo, and Titiunik [2014b]; Calonico, Cattaneo, and Farrell [2018a,b,c]; Calonico et al. [Forthcoming]; and references therein). The second approach is based on a local (to the RD cutoff) independence assumption and hence uses ideas and methods from the classical literature on the analysis of experiments (Cattaneo, Frandsen, and Titiunik [2015]; Cattaneo, Titiunik, and Vazquez-Bare [2017]; Sekhon and Titiunik [2017]; and references therein). Both methods can be used for estimation, inference, and falsification in empirical work using RD designs.

In this article, we focus on local polynomial methods and introduce the commands `rdpow` and `rdsampsi`, which allow for power calculations and sample selection based on large-sample approximations under continuity or smoothness assumptions. `rdpow` conducts power calculations using modern robust bias-corrected local polynomial inference procedures and allows for new hypothetical sample sizes and bandwidth selections, among other features. The companion command `rdsampsi` uses power calculations to compute the minimum sample size required to achieve a desired level of power. Both commands also offer graphical presentation of the main results. Together, these commands are particularly useful when devising new experiments or surveys in RD design settings, assuming they will later be analyzed using modern local polynomial techniques for estimation, inference, and falsification.

The underlying main calculations (bandwidth selection, bias estimation, and variance estimation) in both commands rely on the package `rdrobust` (available in both Stata and R; see Calonico, Cattaneo, and Titiunik [2014a,b] and Calonico et al. [2017] for more details). This implies that all the options and features available in `rdrobust` can be used when using our power calculation and sample selection implementations. In particular, our commands allow for clustered sampling, preintervention covariate adjustment, and different bandwidth selectors, among many other possibilities and features.

Our two commands complement the recently introduced Stata and R commands for RD designs. See Calonico, Cattaneo, and Titiunik (2014a, 2015b) and Calonico et al. (2017) for an introduction to the commands `rdrobust`, `rdbwselect`, and `rdplot` for graphical presentation, estimation, and inference in RD designs using nonparametric local polynomial techniques. See Cattaneo, Titiunik, and Vazquez-Bare (2016) for an introduction to the commands `rdrandinf`, `rdwinselect`, `rdsensitivity`, and `rdrbounds` for implementing randomization-based inference methods for RD designs under a local randomization assumption. See Cattaneo, Jansson, and Ma (2018b) for an introduction to `rddensity` and `rdbwdensity` for manipulation testing using the methods developed in Cattaneo, Jansson, and Ma (2018a).

The rest of the article is organized as follows: Section 2 briefly reviews the main conceptual and methodological issues underlying power calculations for RD designs when using local polynomial inference techniques. Sections 3 and 4 provide details on the syntax of the commands `rdpow` and `rdsampsi`, respectively. Section 5 gives a detailed empirical illustration, and section 6 concludes.

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

2 Overview of methods

We briefly describe the main formulas and methods used in the commands `rdpow` and `rdsampsi` to conduct power calculations and sample selection in RD designs when using local polynomial methods for estimation, inference, and falsification testing. As clarified below, the main formulas and methods require estimating several unknown quantities such as bandwidths, biases, and variances, which are all estimated using the package `rdrobust` (whenever not provided directly by the user). We do not discuss these estimators here but rather refer the reader to Calonico, Cattaneo, and Titiunik (2014a, 2015b), Calonico et al. (2017), and the references therein for further details.

Analyzing the statistical power of widely used RD inference procedures is important for at least two distinct reasons. We briefly mention both now, but we will further discuss them below in our methodology summary (this section) and in our numerical illustration (section 5).

1. *Ex-post (or observational) RD analysis.* The researcher already has the final data for analysis, and the goal is to assess the statistical power underlying the testing procedures implemented in `rdrobust`. Specifically, `rdpow` will estimate the statistical power of the robust bias-corrected inference methods implemented using `rdrobust` for a given hypothesized RD treatment effect (denoted τ_{A} below). In this case, the sample size is fixed, and the goal is to understand the statistical power that different inference procedures have. For example, the researcher can compare the statistical power between using local linear and local quadratic methods.

2. *Ex-ante (or experimental) RD design.* The researcher is designing a new survey for an RD design, and the final data are not available yet. In some circumstances, for instance, the RD design might be a preferable alternative to a classical random assignment design, especially when such design is unfeasible for some reason (ethical or otherwise). For example, the RD design allows for a normal program operation—as opposed to purely randomized designs—because treatment assignments for the study population are determined by rules developed by program staff or policymakers rather than randomly assigned. In this sense, the treatment can be targeted to those who normally receive them (for evaluations of existing interventions) or to those who are likely to benefit most from them (for evaluations of new interventions). In this case, `rdsampsi` implicitly uses power calculations. The main goal of `rdsampsi` is to compute the minimal sample sizes needed to achieve a desired level of power when the final dataset will be analyzed using the testing procedures implemented in `rdrobust`. Typically, larger sample sizes are required in RD designs to construct inference procedures with the same level of statistical power as in classical randomized experiments.

The methods developed and implemented in this article are useful for both ex-post analysis and ex-ante design. In the remainder of this section, we first review classical design of experiments in the context of randomized control trials and then present the corresponding methodology for RD designs.

2.1 Review: Experimental designs

We first review standard approaches for power calculations and sample selection in simple experiments or randomized control trials. Suppose $\{(Y_i, T_i) : 1 \leq i \leq n\}$ is a random sample from a large population, with T_i denoting treatment status and $Y_i = (1-T_i)Y_i(0)+T_iY_i(1)$, with $Y_i(0)$ and $Y_i(1)$ denoting the underlying potential outcomes without and with treatment, respectively. Here, by assumption, T_i is statistically independent of $[Y_i(0), Y_i(1)]$.

We assume the population parameter of interest is the average treatment effect (ATE) $\tau_{\text{ATE}} = \mathbb{E}[Y_i(1) - Y_i(0)]$. To fix ideas, we consider the standard difference-in-means point estimator and corresponding t test statistic. The point estimator compares the difference in means between treated and control units,

$$\widehat{\tau}_{\text{ATE}} = \overline{Y}_1 - \overline{Y}_0, \qquad \overline{Y}_1 = \frac{1}{N_1}\sum_{i=1}^{n} T_iY_i, \qquad \overline{Y}_0 = \frac{1}{N_0}\sum_{i=1}^{n}(1-T_i)Y_i$$

where N_0 and N_1 are the sample sizes of the control and treatment groups, respectively. If we employ a central limit theorem under appropriate regularity conditions, and if τ is the true ATE (for example, $\tau = \tau_{\text{ATE}}$ under the true model), then

$$t_{\text{ATE}}(\tau) = \frac{\widehat{\tau}_{\text{ATE}} - \tau}{\sqrt{\mathsf{V}(\widehat{\tau}_{\text{ATE}})}} \to_d \mathcal{N}(0,1), \qquad \mathsf{V}(\widehat{\tau}_{\text{ATE}}) = \frac{\sigma_1^2}{N_1} + \frac{\sigma_0^2}{N_0}$$

where $\sigma_0^2 = \mathbb{V}[Y_i(0)]$ and $\sigma_1^2 = \mathbb{V}[Y_i(1)]$ are the population variances of the control and treatment groups, respectively. In practice, the two unknown variances are estimated, for instance, by

$$\widehat{\sigma}_0^2 = \frac{1}{N_0 - 1}\sum_{i=1}^{n}(1-T_i)\left(Y_i - \overline{Y}_0\right)^2 \quad \text{and} \quad \widehat{\sigma}_1^2 = \frac{1}{N_1 - 1}\sum_{i=1}^{n} T_i\left(Y_i - \overline{Y}_1\right)^2$$

respectively. Consider the two-sided hypothesis testing problem:

$$\mathsf{H}_0 : \tau = 0 \qquad \text{versus} \qquad \mathsf{H}_{\mathsf{A}} : \tau = \tau_{\mathsf{A}} \neq 0$$

Then, the associated (asymptotic) two-sided α-level power function based on the t test statistic and large-sample distribution theory given above is

$$\beta_{\text{ATE}}(\tau) = 1 - \Phi\left(\frac{\tau}{\sqrt{\mathsf{V}(\widehat{\tau}_{\text{ATE}})}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau}{\sqrt{\mathsf{V}(\widehat{\tau}_{\text{ATE}})}} - z_{1-\alpha/2}\right)$$

where $\Phi(\cdot)$ denotes the cumulative distribution function of the standard normal distribution and z_a denotes its a quantile; that is, $z_a = \Phi^{-1}(a)$. In practice, $\mathsf{V}(\widehat{\tau}_{\text{ATE}})$ is replaced by the consistent estimator

$$\widehat{\mathsf{V}}(\widehat{\tau}_{\text{ATE}}) = \frac{\widehat{\sigma}_1^2}{N_1} + \frac{\widehat{\sigma}_0^2}{N_0}$$

Given the parameter of interest (τ_{ATE}) and test statistic [$t_{\text{ATE}}(\tau)$], one can use the large-sample Gaussian approximation to determine i) the total sample size $n = N_0 + N_1$ needed to achieve a predetermined level of power against a given alternative hypothesis and ii) the optimal assignment of relative sample sizes N_0 and N_1. We briefly discuss these choices next because the same logic will be used further below for RD designs. For simplicity and without loss of generality, we set $\tau_{\text{ATE}} = 0$.

First, we determine the optimal relative sample size between control and treatment groups by minimizing the asymptotic variance of the estimator, given an overall choice of sample size $n = N_0 + N_1$. To be specific, set $\varrho = N_1/n$, and observe that minimizing $\mathsf{V}(\widehat{\tau}_{\text{ATE}})$ with respect to ϱ gives the optimal choice:

$$\varrho = \frac{\sigma_1}{\sigma_0 + \sigma_1}$$

For example, if $\sigma_0 = \sigma_1$, then $\varrho = 1/2$; thus, equal sample sizes for control and treatment units are chosen. More generally, this approach leads to a relatively larger sample size for the group with a relatively larger variability in the outcome variable. This optimal choice of sample-size assignment between control and treatment groups is easily estimable in practice: $\widehat{\varrho} = \widehat{\sigma}_1/(\widehat{\sigma}_0 + \widehat{\sigma}_1)$.

Second, given the choice of ϱ, we determine the total sample size

$$n = N_0 + N_1, \qquad N_0 = (1-\varrho)n, \qquad N_1 = \varrho n$$

using the (asymptotic) power function. Specifically, given a choice of alternative $\tau_{\mathsf{A}} \neq 0$ (usually determined as a fraction of σ_0) and the desired power $\beta \in [0,1]$ (usually set at $\beta = 80\%$), the overall sample size n is chosen as the unique integer value n solving the equation $\beta = \beta_{\text{ATE}}(\tau_{\mathsf{A}})$; that is,

$$n \text{ solves: } \beta = 1 - \Phi\left(\frac{\tau_{\mathsf{A}}}{\sqrt{\frac{\sigma_1^2}{\varrho n} + \frac{\sigma_0^2}{(1-\varrho)n}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau_{\mathsf{A}}}{\sqrt{\frac{\sigma_1^2}{\varrho n} + \frac{\sigma_0^2}{(1-\varrho)n}}} - z_{1-\alpha/2}\right)$$

where the only unknown is n after ϱ, σ_0^2, and σ_1^2 are replaced by their corresponding estimators $\widehat{\varrho}$, $\widehat{\sigma}_0^2$, and $\widehat{\sigma}_1^2$, as commonly done in practice.

The approach above is used in empirical work to determine the total sample size, n, and the relative proportion of sample sizes of control and treatment units, ϱ. In the following sections, we build on these ideas and apply them to the context of RD designs using local polynomial methods.

2.2 Regression discontinuity designs

For the remainder of this article, we study power calculation and sample selection in RD designs. We assume that $\{(Y_i, T_i, X_i) : 1 \leq i \leq n\}$ is a random sample from a large population, where for each unit i in the sample, $Y_i = (1 - T_i)Y_i(0) + T_iY_i(1)$ is the outcome variable, with $Y_i(0)$ and $Y_i(1)$ denoting the underlying potential outcomes without and with treatment; $T_i = \mathbb{1}(X_i < c)T_i(0) + \mathbb{1}(X_i \geq c)T_i(1)$ is the treatment status, with $T_i(0)$ and $T_i(1)$ denoting the underlying potential treatment status; X_i denotes the so-called running variable, score, or index; and c denotes the RD cutoff. In RD designs, treatment assignment for each unit is determined based on whether the score X_i exceeds the known cutoff c. We cover all possible RD cases simultaneously; therefore, we do not assume perfect compliance. That is, we do not force $T_i = \mathbb{1}(X_i \geq c)$ as in sharp RD designs.

Our presentation is sufficiently high level enough that we avoid unnecessary repetition and overwhelming notation required to discuss each RD setting in detail, so we refer the reader to Calonico et al. (2017) and the references therein for specific details. To be more concrete, we denote a generic RD parameter of interest by τ_ν, where $\nu = 0, 1, \ldots, p$ traces out the specific cases: i) $\nu = 0$ corresponds to sharp RD cases (for example, the average sharp RD treatment effect at the cutoff $\tau_0 = \mathbb{E}[Y_i(1) - Y_i(0)|X_i = c]$, or the fuzzy RD estimand $\varsigma_0 = \mathbb{E}[Y_i(1) - Y_i(0)|X_i = c, \text{compliers}]$); and ii) $\nu = 1$ corresponds to kink RD cases (for example, $\tau_1 = \partial\mathbb{E}[Y_i(1) - Y_i(0)|X_i = c]/\partial c$ corresponds to the average sharp kink RD treatment effect at the cutoff, and ς_1 is the corresponding fuzzy–kink RD estimand).

To describe how to do power calculations and sample selection in RD designs, and continuing with our high level of generality, we let $\widehat{\tau}_\nu$ denote a generic (pth order) local polynomial estimator constructed with bandwidths h_- (for left estimation) and h_+ (for right estimation) of the corresponding RD treatment effect τ_ν. Then, under regularity conditions and bandwidth sequence restrictions, the following distributional approximation holds whenever τ is the population RD treatment effect (for example, $\tau = \tau_\nu$ under the true model):

$$\frac{\widehat{\tau}_\nu - \tau - \mathsf{B}}{\sqrt{\mathsf{V}}} \to_d \mathcal{N}(0, 1) \tag{1}$$

with

$$\mathsf{B} = h_+^{1+p-\nu}\mathsf{B}_+ - h_-^{1+p-\nu}\mathsf{B}_-, \qquad \mathsf{V} = \frac{1}{nh_+^{1+2\nu}}\mathsf{V}_+ + \frac{1}{nh_-^{1+2\nu}}\mathsf{V}_-$$

B_- and B_+ denote the left and right (to the RD cutoff) misspecification biases, and V_- and V_+ denote the left and right (to the RD cutoff) asymptotic variance of the RD estimator. Depending on the estimand and estimator under consideration, the exact form of these quantities is different. Nevertheless, the generic result above applies to all main RD cases at this level of generality, including cases where the estimand of interest is the fuzzy, kink, or fuzzy–kink RD parameter; cases where preintervention covariate adjustment is considered; and cases where clustered sampling is present. For

more details, see Calonico, Cattaneo, and Titiunik (2014b) as well as Calonico et al. (Forthcoming) and references therein.

We use the generic large-sample Gaussian distributional approximation in (1) to construct asymptotic power functions and to select sample sizes for survey design based on them. Without loss of generality, we assume $\tau_\nu = 0$ in the rest of this article.

Power calculations with misspecification bias

In this section, we consider an approach to power calculation and sample selection in RD designs that explicitly ignores misspecification bias. This approach is not the default in the `rdpow` and `rdsampsi` commands, but it can be recovered using some of the options provided (more details are given below).

Misspecification bias arises when the local polynomial approximation used to estimate the RD treatment effect is not good enough near the cutoff because it occurs when a mean squared error (MSE) optimal or other "large" bandwidth is used. As in the case of experimental designs discussed previously, the distributional result for RD designs (1) gives a generic asymptotic power function for the corresponding (nominal) α-level two-sided hypothesis test about τ_ν of the form

$$\beta_\nu(\tau) = 1 - \Phi\left(\frac{\tau + \mathsf{B}}{\sqrt{\mathsf{V}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau + \mathsf{B}}{\sqrt{\mathsf{V}}} - z_{1-\alpha/2}\right)$$

In practice, the unknown biases, variances, and bandwidths are estimated using nonparametric methods and accounting for the specific sampling assumptions and the specific choice of estimand and estimator. These estimates are all already available in the package `rdrobust`. In general, we denote the bias estimators by $\widehat{\mathsf{B}}_+$ and $\widehat{\mathsf{B}}_-$, the variance estimators by $\widehat{\mathsf{V}}_+$ and $\widehat{\mathsf{V}}_-$, and the bandwidth estimators by $\widehat{h}_+$ and $\widehat{h}_-$ (in many applications, a common bandwidth is selected, in which case $\widehat{h} = \widehat{h}_+ = \widehat{h}_-$).

Therefore, in practice all the unknown quantities can be estimated, and hence, the power function can be computed. However, an interesting implication of the presence of misspecification biases implies that the estimated power function will generally exhibit size distortions (under the null hypothesis $\tau = \tau_\nu = 0$). That is, we have

$$\widehat{\beta}_\nu(0) = 1 - \Phi\left(\frac{\widehat{\mathsf{B}}}{\sqrt{\widehat{\mathsf{V}}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\widehat{\mathsf{B}}}{\sqrt{\widehat{\mathsf{V}}}} - z_{1-\alpha/2}\right) > \alpha$$

where

$$\widehat{\mathsf{B}} = \widehat{h}_+^{1+p-\nu}\widehat{\mathsf{B}}_+ - \widehat{h}_-^{1+p-\nu}\widehat{\mathsf{B}}_-, \qquad \widehat{\mathsf{V}} = \frac{1}{n\widehat{h}_+^{1+2\nu}}\widehat{\mathsf{V}}_+ + \frac{1}{n\widehat{h}_-^{1+2\nu}}\widehat{\mathsf{V}}_-$$

Hence, conventional inference with misspecification bias will show higher power but at the expense of size distortion, which depends on the magnitude of the bias relative to the standard error. We discuss this point further in the empirical illustration given below.

Power calculations with robust bias correction

Calonico, Cattaneo, and Titiunik (2014b) and Calonico et al. (Forthcoming) develop robust bias-correction inference methods for RD designs, which allow for MSE optimal bandwidth selection and provide higher-order refinements. See also Calonico, Cattaneo, and Farrell (2018a,b,c) for related results. The idea is to first use an estimator of the bias to correct the point estimator and then adjust the variance estimator to account for the additional variability introduced by the bias estimate.

In the robust bias-correction approach, under regularity conditions, bandwidth sequence restrictions, and the assumption that τ is the population RD treatment effect, the following distributional approximation holds:

$$\frac{\widehat{\tau}_\nu^{\mathsf{bc}} - \tau}{\sqrt{\widehat{\mathsf{V}}^{\mathsf{bc}}}} \to_d \mathcal{N}(0,1), \qquad \widehat{\tau}_\nu^{\mathsf{bc}} = \widehat{\tau}_\nu - \widehat{\mathsf{B}}, \qquad \widehat{\mathsf{V}}^{\mathsf{bc}} = \frac{1}{n\widehat{h}_+^{1+2\nu}}\widehat{\mathsf{V}}_+^{\mathsf{bc}} + \frac{1}{n\widehat{h}_-^{1+2\nu}}\widehat{\mathsf{V}}_-^{\mathsf{bc}}$$

$\widehat{\mathsf{V}}_-^{\mathsf{bc}}$ and $\widehat{\mathsf{V}}_+^{\mathsf{bc}}$ are, respectively, consistent estimators of the asymptotic variances $\mathsf{V}_-^{\mathsf{bc}} \neq \mathsf{V}_-$ and $\mathsf{V}_+^{\mathsf{bc}} \neq \mathsf{V}_+$. Crucially, the variances $\mathsf{V}_-^{\mathsf{bc}}$ and $\mathsf{V}_+^{\mathsf{bc}}$ account for the effect of estimating the misspecification errors B_- and B_+, leading to hypothesis tests and confidence interval estimators with demonstrably superior properties.

Therefore, in the robust bias-corrected framework, the generic (estimated) asymptotic power function for the α-level two-sided hypothesis test about τ takes the form

$$\widehat{\beta}_\nu^{\mathsf{bc}}(\tau) = 1 - \Phi\left(\frac{\tau}{\sqrt{\widehat{\mathsf{V}}^{\mathsf{bc}}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau}{\sqrt{\widehat{\mathsf{V}}^{\mathsf{bc}}}} - z_{1-\alpha/2}\right)$$

This power function is the default in `rdpow` and `rdsampsi`. We discuss some specifics underlying these commands in the next two subsections.

rdpow: User-chosen bandwidths and sample sizes

The command `rdpow` computes an approximation to the power function using either the misspecification approach (with estimated size distortion) or the robust bias-correction approach. By default, `rdpow` uses the latter approach implemented via `rdrobust` and setting all options to its data-driven choices whenever possible. Specifically, by default, `rdpow` computes power for the robust bias-corrected statistics using all default data-driven choices for bandwidth, bias, variance, and even alternative hypothesis τ_{A}, which is set to half of a standard deviation of the outcome variable for the control group in the estimated region $[c - \widehat{h}_-, c)$, where $\widehat{h}_- = \widehat{h}_+ = \widehat{h}$ is (by default) the common MSE optimal bandwidth for $p = 1$ according to `rdrobust`'s default options.

However, in practice, researchers may want to set different biases, variances, bandwidths, and effective sample sizes. `rdpow` allows user-specified choices of these quantities. For example, in empirical applications, biases and variances may be estimated using a data source, but then the researcher may want to consider a different bandwidth, sample size, or both when planning a new survey or field work. That is, the follow-up survey

may not use all the same data points available for estimation of biases and variances when computing power, but perhaps focus on observations closer or further away from the cutoff, depending on the specific field work planned.

To account for the above possibilities, `rdpow` allows for user-chosen biases and variances, which enter the power function directly in place of the estimated quantities $\widehat{\mathsf{V}}_-^{\mathsf{bc}}$ and $\widehat{\mathsf{V}}_+^{\mathsf{bc}}$ (for robust bias-corrected methods) and $\widehat{\mathsf{B}}_-$, $\widehat{\mathsf{B}}_+$, $\widehat{\mathsf{V}}_-$, and $\widehat{\mathsf{V}}_+$ (for misspecified methods). Furthermore, `rdpow` allows user-chosen bandwidths and sample sizes entering the power function, in which case the command uses the following adjusted power function:

$$\widetilde{\beta}_\nu^{\mathsf{bc}}(\tau) = 1 - \Phi\left(\frac{\tau}{\sqrt{\widetilde{\mathsf{V}}^{\mathsf{bc}}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau}{\sqrt{\widetilde{\mathsf{V}}^{\mathsf{bc}}}} - z_{1-\alpha/2}\right)$$

$$\widetilde{\mathsf{V}}^{\mathsf{bc}} = \frac{\widehat{\mathsf{V}}_+^{\mathsf{bc}}}{mh_+^{1+2\nu}} + \frac{\widehat{\mathsf{V}}_-^{\mathsf{bc}}}{mh_-^{1+2\nu}}, \qquad m = \frac{N_+}{N_{h_+}} \times M_+ + \frac{N_-}{N_{h_-}} \times M_-$$

$$N_- = \sum_{i=1}^{n} \mathbb{1}(X_i < c), \quad N_+ = \sum_{i=1}^{n} \mathbb{1}(c \leq X_i)$$

$$N_{h_-} = \sum_{i=1}^{n} \mathbb{1}(c - h_- \leq X_i < c), \quad N_{h_+} = \sum_{i=1}^{n} \mathbb{1}(c \leq X_i \leq c + h_+)$$

M_- and M_+ denote the new (postulated by the user) sample sizes in the neighborhoods $[c-h_-, c)$ and $[c, c+h_+]$, respectively, for control and treatment. h_- and h_+ denote the new bandwidths chosen below and above the cutoff, respectively. If M_- and M_+ are not specified by the user, then `rdpow` sets $M_- = N_{h_-}$ and $M_+ = N_{h_+}$, implying that $m = n$. Similarly, if the user does not specify the new bandwidths h_- and h_+, then `rdpow` sets $h_- = \widehat{h}_-$ and $h_+ = \widehat{h}_+$; that is, `rdpow` uses the automatic bandwidth chosen in a data-driven way to estimate the bias and variances entering the power function.

Finally, for power calculations with misspecification bias, the adjusted power function takes the form

$$\widetilde{\beta}_\nu(\tau) = 1 - \Phi\left(\frac{\tau + \widetilde{\mathsf{B}}}{\sqrt{\widetilde{\mathsf{V}}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau + \widetilde{\mathsf{B}}}{\sqrt{\widetilde{\mathsf{V}}}} - z_{1-\alpha/2}\right)$$

$$\widetilde{\mathsf{B}} = h_+^{1+p-\nu}\widehat{\mathsf{B}}_+ - h_-^{1+p-\nu}\widehat{\mathsf{B}}_-, \qquad \widetilde{\mathsf{V}} = \frac{\widehat{\mathsf{V}}_+}{mh_+^{1+2\nu}} + \frac{\widehat{\mathsf{V}}_-}{mh_-^{1+2\nu}}$$

where m is constructed exactly like robust bias-corrected inference.

rdsampsi: Sample-size selection

The command `rdsampsi` uses the estimated, possibly adjusted, RD power functions presented above to compute the minimal sample size required to achieve a prespecified level of power. It also computes the optimal relative sample sizes for control and treatment

groups, following the same logic discussed previously in the context of experimental designs. All the results are mapped to effective sample sizes within the neighborhood around the cutoff determined by the user-supplied bandwidth or bandwidths or, if not specified, then within the neighborhood around the cutoff determined by the estimated bandwidth or bandwidths using `rdrobust`.

We first discuss selecting the relative sample sizes of control and treatment groups within a given neighborhood around the cutoff c. Recall that `rdrobust` provides data-driven point estimators of the robust bias-corrected variances at the cutoff for control and treatment units: $\widehat{\mathsf{V}}_{-}^{\mathsf{bc}}$ and $\widehat{\mathsf{V}}_{+}^{\mathsf{bc}}$. Thus, the relative (effective) control and treatment sample sizes can be chosen in exactly the same way as previously discussed for experimental designs: the estimated optimal proportion of effective treated units in RD designs is

$$\widehat{\varrho}_{\nu} = \frac{\sqrt{\widehat{\mathsf{V}}_{+}^{\mathsf{bc}}}}{\sqrt{\widehat{\mathsf{V}}_{-}^{\mathsf{bc}}} + \sqrt{\widehat{\mathsf{V}}_{+}^{\mathsf{bc}}}}$$

Suppose that M is the effective sample; that is, M is the number of selected observations in the neighborhood $[c - h_{-}, c + h_{+}]$, then we recommend sampling (rounding to the closest integer) the following number of observations on either side of the cutoff:

- $M_1 = \widehat{\varrho}_{\nu} M$ treatment units (that is, those with $X_i \geq c$)
- $M_0 = (1 - \widehat{\varrho}_{\nu}) M$ control units (that is, those with $X_i < c$)

Finally, we discuss how to select the total number of effective observations $M = M_0 + M_1$. As with experimental designs, the total number of observations can be determined by preselecting a desired level of power β for a given alternative hypothesis τ_{A}, using the power functions presented above and the already determined factor of proportionality $\widehat{\varrho}_{\nu}$. Specifically, the optimal effective sample size M in the neighborhood $[c - h_{-}, c + h_{+}]$ is

$$M \text{ solves} : \beta = 1 - \Phi\left(\frac{\tau_{\mathsf{A}}}{\sqrt{\breve{\mathsf{V}}^{\,\mathsf{bc}}}} + z_{1-\alpha/2}\right) + \Phi\left(\frac{\tau_{\mathsf{A}}}{\sqrt{\breve{\mathsf{V}}^{\,\mathsf{bc}}}} - z_{1-\alpha/2}\right)$$

where

$$\breve{\mathsf{V}}^{\,\mathsf{bc}} = \frac{1}{M} \times \frac{1}{\frac{N_{+}}{N_{h_{+}}} \times \widehat{\varrho}_{\nu} + \frac{N_{-}}{N_{h_{-}}} \times (1 - \widehat{\varrho}_{\nu})} \times \left(\frac{\widehat{\mathsf{V}}_{+}^{\mathsf{bc}}}{h_{+}^{1+2\nu}} + \frac{\widehat{\mathsf{V}}_{-}^{\mathsf{bc}}}{h_{-}^{1+2\nu}}\right)$$

How to survey or sample in RD designs

For internal validity in RD designs, it is always best to first sample observations that are the closest to the cutoff c in terms of their running variable X_i because of the same assumptions underlying identification, estimation, and inference methods based on continuity or smoothness of the unknown conditional expectations. In this framework,

the parameter of interest is defined at the cutoff c, so having observations as close as possible to $X_i = c$ is most useful.

Therefore, once M_0 and M_1 are determined, the actual sampling or field work is straightforward:

1. Order control ($X_i < c$) and treatment ($X_i \geq c$) observations in terms of their distance to the cutoff ($|X_i - c|$).
2. Begin sampling with the closest observation to the cutoff in each group, and continue sampling in order according to their distance to the cutoff, within each group, until the desired M_0 and M_1 sample sizes are reached.

Clustered sampling

All the results above apply directly to independent and identically distributed sampling and can be extended to clustered sampling. Estimation of unknown biases and variances, both with misspecification and with robust bias correction, is readily available via `rdrobust`. Therefore, power calculations and sample-size selection at the cluster level are also readily available in our commands `rdpow` and `rdsampsi`. We do not discuss details here to avoid repetition and overwhelming notation.

3 The rdpow command

In this section, we describe the syntax of the command `rdpow`, which performs power calculations using robust bias-corrected local polynomial techniques for inference. This command is useful for ex-post RD analysis and an input for ex-ante RD analysis via the companion command `rdsampsi`. Options highlighted in gray are passed directly to `rdrobust`, while the options in black are specific to this command.

3.1 Syntax

`rdpow` *depvar runvar* [*if*] [*in*] [`, c(#) tau(#) alpha(#)`
`nsamples(# # # #) sampsi(# #) samph(# #) all plot`
`graph_range(# #) graph_step(#) graph_options(`*graph_opt*`) bias(# #)`
`variance(# #) covs(`*covars*`) deriv(`*dvalue*`) p(#) q(#) h(`*hvalueL hvalueR*`)`
`b(`*bvalueL bvalueR*`) rho(#) fuzzy(`*fuzzyvar* [`sharpbw`]`) kernel(`*kernelfn*`)`
`bwselect(`*bwmethod*`) vce(`*vcemethod*`) scalepar(#) scaleregul(#)`]

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

3.2 Options

`c(#)` specifies the RD cutoff. The default is `c(0)`.

`tau(#)` specifies the treatment effect under the alternative at which the power function is evaluated. The default is half the standard deviation of the outcome for the untreated group.

`alpha(#)` specifies the significance level for the power function. The default is `alpha(0.05)`.

`nsamples(# # # #)` sets the total sample size to the left, sample size to the left inside the bandwidth, total sample size to the right, and sample size to the right of the cutoff inside the bandwidth to calculate the variance when the running variable is not specified. By default, the values are calculated using the running variable.

`sampsi(# #)` sets the sample size at each side of the cutoff for power calculation. The first number is the sample size to the left of the cutoff, and the second number is the sample size to the right. The default values are the sample sizes inside the chosen bandwidth.

`samph(# #)` sets the bandwidths at each side of the cutoff for power calculation. The first number is the bandwidth to the left of the cutoff, and the second number is the bandwidth to the right. The default values are the bandwidths used by `rdrobust`.

`all` displays the power using the conventional variance estimator and the robust bias-corrected one.

`plot` plots the power function using the robust bias-corrected standard errors from `rdrobust`. If `all` is specified, the conventional power function is also plotted.

`graph_range(# #)` specifies the range of the plot when the `plot` option is used. The default range is $[-1.5 \times \tau; 1.5 \times \tau]$.

`graph_step(#)` specifies the step increment of the plot when the `plot` option is used. The default is `graph_step(0.2*`*range*`)`.

`graph_options(`*graph_opt*`)` specifies the graph options (title, axes titles, etc.) to be passed to the plot when the `plot` option is used.

`bias(# #)` allows the user to set bias to the left and right of the cutoff. By default, the biases are estimated using `rdrobust`.

`variance(# #)` allows the user to set variance to the left and right of the cutoff. By default, the variances are estimated using `rdrobust`.

The following options are passed to `rdrobust`:

`covs(`*covars*`)` specifies the list of covariates to be used for covariate adjustment.

`deriv(`*dvalue*`)` specifies the order of the derivative of the regression functions to be estimated. The default is `deriv(0)` (sharp RD or fuzzy RD if `fuzzy()` is also specified). Setting `deriv(1)` results in estimation of a kink RD design (up to scale) or fuzzy–kink RD if `fuzzy()` is also specified.

`p(`#`)` specifies the order of the local polynomial to be used to construct the point estimator. The default is `p(1)` (local linear regression).

`q(`#`)` specifies the order of the local polynomial to be used to construct the bias correction. The default is `q(2)` (local quadratic regression).

`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. By default, the bandwidth or bandwidths h are 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. By default, the bandwidth or bandwidths b are computed by the companion command `rdbwselect`.

`rho(`#`)` specifies the value of ρ so that the bias bandwidth, b, equals $b = h/\rho$. The default is `rho(1)` if h is specified but b is not.

`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.

`kernel(`*kernelfn*`)` specifies the kernel function used to construct the local polynomial estimators. *kernelfn* may be `triangular`, `uniform`, or `epanechnikov`. The default is `kernel(triangular)`.

`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 h and sets $b = h/\rho$. Implementation and numerical details are given in references mentioned previously. *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 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 the median(`msetwo`, `mserd`, `msesum`) for each side of the cutoff separately.

`cerrd` specifies one common coverage error-rate (CER)-optimal bandwidth selector for the RD treatment-effect estimator.

`certwo` specifies two 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 the 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 references mentioned previously. *vcemethod* may be one of the following:

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

`hc0` specifies a heteroskedasticity-robust plugin residuals variance estimator.

`hc1` specifies a heteroskedasticity-robust plugin residuals variance estimator.

`hc2` specifies a heteroskedasticity-robust plugin residuals variance estimator.

`hc3` specifies a heteroskedasticity-robust plugin residuals variance estimator.

`nncluster` *clustervar* [*nnmatch*] specifies a cluster–robust nearest-neighbor 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 plugin residuals variance estimator with degrees-of-freedom weights and *clustervar* indicating the cluster ID variable.

`scalepar(`#`)` specifies the scaling factor for the RD parameter of interest. This option is useful when the population parameter of interest involves a known multiplicative factor (for example, sharp kink RD). The default is `scalepar(1)` (no scaling).

`scaleregul(`#`)` specifies the scaling factor for the regularization term added to the denominator of the bandwidth selectors. Setting `scaleregulvalue(0)` removes the regularization term from the bandwidth selectors. The default is `scaleregul(1)`.

4 The rdsampsi command

This section describes the syntax of the command `rdsampsi`, which performs sample-size calculations using robust bias-corrected local polynomial techniques for inference in RD designs. This command is most useful for ex-ante RD design and relies on the companion command `rdpow` for power calculations. Options highlighted in gray are passed directly to `rdrobust`, while the options in black are specific to this command.

4.1 Syntax

`rdsampsi` *depvar runvar* [*if*] [*in*] [, `c(#) tau(#) alpha(#) beta(#)`
`nsamples(# # # #) samph(# #) all plot graph_range(# #)`
`graph_step(#) graph_options(`*graph_opt*`) bias(# #) variance(# #)`
`nratio(#) init_cond(#) covs(`*covars*`) deriv(`*dvalue*`) p(#) q(#)`
`h(`*hvalueL hvalueR*`) b(`*bvalueL bvalueR*`) rho(#) fuzzy(`*fuzzyvar* [`sharpbw`]`)`
`kernel(`*kernelfn*`) bwselect(`*bwmethod*`) vce(`*vcemethod*`) scalepar(#)`
`scaleregul(#)`]

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

4.2 Options

`c(#)` specifies the RD cutoff. The default is `c(0)`.

`tau(#)` specifies the treatment effect under the alternative at which the power function is evaluated. The default is half the standard deviation of the outcome for the untreated group.

`alpha(#)` specifies the significance level for the power function. The default is `alpha(0.05)`.

`beta(#)` specifies the desired power. The default is `beta(0.8)`.

`nsamples(# # # #)` sets the total sample size to the left, sample size to the left inside the bandwidth, total sample size to the right, and sample size to the right of the cutoff inside the bandwidth to calculate the variance when the running variable is not specified. By default, the values are calculated using the running variable.

`samph(# #)` sets the bandwidths at each side of the cutoff for power calculation. The first number is the bandwidth to the left of the cutoff, and the second number is the bandwidth to the right. The default values are the bandwidths used by `rdrobust`.

`all` displays the power using the conventional variance estimator and the robust bias-corrected one.

`plot` plots the power function using the robust bias-corrected standard errors from `rdrobust`. If `all` is specified, the conventional power function is also plotted.

`graph_range(# #)` specifies the range of the plot when the `plot` option is used. The default range is $[-1.5 \times \tau; 1.5 \times \tau]$.

`graph_step(#)` specifies the step increment of the plot when the `plot` option is used. The default is `graph_step(0.2*`*range*`)`.

`graph_options(`*graph_opt*`)` specifies the graph options (title, axes titles, etc.) to be passed to the plot when the `plot` option is used.

`bias(# #)` allows the user to set bias to the left and right of the cutoff. By default, the biases are estimated using `rdrobust`.

`variance(# #)` allows the user to set variance to the left and right of the cutoff. By default, the variances are estimated using `rdrobust`.

`nratio(#)` specifies the proportion of treated units in the window. The default is the ratio of the standard deviation of the treated to the sum of the standard deviations for treated and controls.

`init_cond(#)` sets the initial condition for the Newton–Raphson algorithm that finds the sample size. The default is the number of observations in the sample with nonmissing values of the outcome and running variable.

The following options are passed to `rdrobust`:

`covs(`*covars*`)` specifies the list of covariates to be used for covariate adjustment.

`deriv(`*dvalue*`)` specifies the order of the derivative of the regression functions to be estimated. The default is `deriv(0)` (sharp RD or fuzzy RD if `fuzzy()` is also specified). Setting `deriv(1)` results in estimation of a kink RD design (up to scale) or fuzzy–kink RD if `fuzzy()` is also specified.

`p(#)` specifies the order of the local polynomial to be used to construct the point estimator. The default is `p(1)` (local linear regression).

`q(#)` specifies the order of the local polynomial to be used to construct the bias correction. The default is `q(2)` (local quadratic regression).

`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. By default, the bandwidth or bandwidths h are computed by the companion command `rdbwselect`.

`b(`*bvalueL bvalueR*`)` specifies the values of 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. By default, the bandwidth or bandwidths b are computed by the companion command `rdbwselect`.

`rho(#)` specifies the value of ρ so that the bias bandwidth, b, equals $b = h/\rho$. The default is `rho(1)` if h is specified but b is not.

`fuzzy(`*fuzzyvar* [`sharpbw`]`)` is the treatment status variable used to implement fuzzy RD estimation (or fuzzy–kink RD if `deriv(1)` is also specified). The additional option `sharpbw` forces bandwidth selection for the numerator of the fuzzy RD design estimator (that is, the sharp RD intention-to-treat estimator).

`kernel(`*kernelfn*`)` specifies the kernel function used to construct the local polynomial estimators. *kernelfn* may be `triangular`, `uniform`, and `epanechnikov`. The default is `kernel(triangular)`.

`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 h and sets $b = h/\rho$. Implementation and numerical details are given in the references mentioned previously. *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 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 the 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 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 the 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 references mentioned previously. *vcemethod* may be one of the following:

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

`hc0` specifies a heteroskedasticity-robust plugin residuals variance estimator without weights.

`hc1` specifies a heteroskedasticity-robust plugin residuals variance estimator with `hc1` weights.

`hc2` specifies a heteroskedasticity-robust plugin residuals variance estimator with `hc2` weights.

`hc3` specifies a heteroskedasticity-robust plugin residuals variance estimator with `hc3` weights.

`nncluster` *clustervar* [*nnmatch*] specifies a cluster–robust nearest-neighbor 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 plugin residuals variance estimator with degrees-of-freedom weights and *clustervar* indicating the cluster ID variable.

`scalepar(#)` specifies the scaling factor for the RD parameter of interest. This option is useful when the population parameter of interest involves a known multiplicative factor (for example, sharp kink RD). The default is `scalepar(1)` (no scaling).

`scaleregul(#)` specifies the scaling factor for the regularization terms of bandwidth selectors. Setting `scaleregulvalue(0)` removes the regularization term from the bandwidth selectors. The default is `scaleregul(1)`.

5 Illustration of methods

We illustrate our commands using the dataset from Cattaneo, Frandsen, and Titiunik (2015), which has also been used to illustrate related RD methods (Calonico, Cattaneo, and Titiunik 2014a, 2015b; Cattaneo, Titiunik, and Vazquez-Bare 2017; Cattaneo, Jansson, and Ma 2018b).

The dataset `rdpower_senate.dta` contains information on 1,390 U.S. Senate elections between 1914 and 2010 and was used before to analyze the effect of the incumbent status of a political party on the probability of winning future elections. The running variable in this dataset is `demmv`, the Democratic margin of victory in a statewide Senate election at time t, defined as the difference in vote share between the Democratic party and its strongest opponent. A positive value of the running variable indicates that the Democratic party won the election, and the cutoff is therefore $\bar{r} = 0$.

We start by loading the dataset and providing some descriptive statistics:

```
. use rdpower_senate.dta
. summarize demmv demvoteshfor2 population dopen dmidterm
```

Variable	Obs	Mean	Std. Dev.	Min	Max
demmv	1,390	7.171159	34.32488	-100	100
demvoteshf~2	1,297	52.66627	18.12219	0	100
population	1,390	3827919	4436950	78000	3.73e+07
dopen	1,380	.2471014	.4314826	0	1
dmidterm	1,390	.5136691	.499993	0	1

The running variable ranges from -100 to 100 with an average of 7 percentage points. The outcome of interest is `demvoteshfor2`, the Democratic vote share in the following election for the same Senate seat—which, given the staggered nature of senate elections in the United States, occurs two elections later at $t + 2$.

5.1 Power calculations using rdpow

The most basic syntax to calculate power against an alternative hypothesis of $\tau = 5$ is the following:

```
. rdpow demvoteshfor2 demmv, tau(5)
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708
New sample	359	322

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.300	0.631	0.818

The output of `rdpow` is divided in three panels. The upper-left panel shows some descriptive statistics and parameters, including the number of observations at each side of the cutoff the number of observations inside the chosen window, the order of the local polynomials, used to estimate the effect and the bias, the corresponding bandwidths at each side of the cutoff, and the ratio between these two. The upper-right panel displays the number of observations in the sample and specifies the selected options for the command, namely, the bandwidth selector, kernel, variance estimation method, and value of the treatment effect under the alternative (τ). Finally, the bottom panel shows the size of the test (that is, the value of the power function at the null hypothesis of $\tau = 0$) and the power of the test against values under the alternative that range from 0.2τ to τ (hence, the power for the value selected by the user is given in the last column). Note that, by construction, the value in the first column will be equal to the significance level (although this may not be true when using conventional inference, as explained later). By default, this panel uses the robust bias-corrected estimator (see section 2 and references therein for details).

The output shows that the power against $\tau = 5$ is 0.818, which is slightly above the usual threshold of 0.8. In many contexts, empirical researchers include covariates hoping to reduce the variability of the estimator. This option can be added to the `rdpow` command as follows:

```
. rdpow demvoteshfor2 demmv, tau(5) covs(population dopen dmidterm)
```

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	358	317	VCE method =	NN
BW loc. poly. (h)	17.415	17.415	Derivative =	0
Order loc. poly. (p)	1	1	HA: tau =	5.000
Sampling BW	17.415	17.415		
New sample	358	317		

Outcome: demvoteshfor2. Running variable: demmv. Number of covariates: 3.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.296	0.624	0.812

However, in this case there seems to be no gain in power by including covariates. Note that when the `covs()` option is specified, the output displays the number of included covariates in the line just above the bottom panel.

The `plot` option allows the user to plot the power function. The `graph_range()` and `graph_step()` options can be used to change the labeling of the x axis. Additionally, the `graph_options()` option can be used to change the appearance of the graph. For instance, the following syntax yields the graph shown in figure 1:

```
. rdpow demvoteshfor2 demmv, tau(5) plot graph_range(-9 9) graph_step(2)
> graph_options(title(Power function)
>      xline(0, lcolor(black) lpattern(dash))
>      yline(.05, lpattern(shortdash) lcolor(black))
>      xtitle(tau) ytitle(power)
>      graphregion(fcolor(white)))
```

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
BW loc. poly. (h)	17.708	17.708	Derivative =	0
Order loc. poly. (p)	1	1	HA: tau =	5.000
Sampling BW	17.708	17.708		
New sample	359	322		

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.300	0.631	0.818

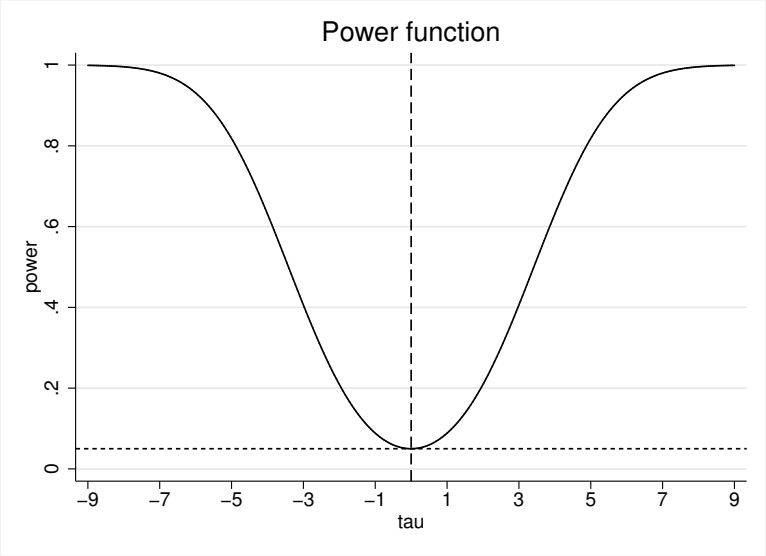


Figure 1. Robust bias-corrected power function

By default, `rdpow` uses `rdrobust` to select the bandwidths used to estimate the effect and the bias. The user can manually specify these bandwidths using the `h()` and `b()` options:

```
. rdpow demvoteshfor2 demmv, tau(5) h(16 18) b(18 20)
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	332	325
BW loc. poly. (h)	16.000	18.000
Order loc. poly. (p)	1	1
Sampling BW	16.000	18.000
New sample	332	325

Number of obs =	1297
BW type =	Manual
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.079	0.240	0.517	0.707

`rdpow` allows for most of the options from `rdrobust`. For instance, the following syntax specifies a uniform kernel and clusters the standard errors at the state level:

```
. rdpow demvoteshfor2 demmv, tau(5) kernel(uniform) vce(cluster state)
```

Cutoff c = 0	Left of c	Right of c		
Number of obs	595	702	Number of obs =	1297
Eff. Number of obs	295	258	BW type =	mserd
BW loc. poly. (h)	13.242	13.242	Kernel =	Uniform
Order loc. poly. (p)	1	1	VCE method =	Cluster
			Derivative =	0
Sampling BW	13.242	13.242	HA: tau =	5.000
New sample	49	48		
Number of clusters	50	50		

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.114	0.454	0.839	0.958

Standard errors clustered by state.

On the other hand, the following syntax specifies different CER-optimal bandwidths at each side of the cutoff, a heteroskedasticity-robust plugin residuals variance estimator with `hc3` weights, removes the regularization term from the bandwidth selectors, and sets equal bandwidths for the estimator and bias terms (see Calonico, Cattaneo, and Titiunik [2014a] and Calonico et al. [2017] for technical details):

```
. rdpow demvoteshfor2 demmv, tau(5) bwselect(certwo) vce(hc3) scaleregul(0) rho(1)
```

Cutoff c = 0	Left of c	Right of c		
Number of obs	595	702	Number of obs =	1297
Eff. Number of obs	393	401	BW type =	certwo
BW loc. poly. (h)	20.524	24.805	Kernel =	Triangular
Order loc. poly. (p)	1	1	VCE method =	HC3
			Derivative =	0
Sampling BW	20.524	24.805	HA: tau =	5.000
New sample	393	401		

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.082	0.259	0.556	0.747

Finally, the `all` option allows the user to compare the robust bias-correction and conventional approaches:

```
. rdpow demvoteshfor2 demmv, tau(5) all
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708
New sample	359	322

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000
Size dist =	0.0133

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.300	0.631	0.818
Conventional	0.063	0.176	0.537	0.868	0.964

In this case, we see that the conventional approach yields higher power but at the expense of ignoring the misspecification bias, which creates a size distortion of 0.0133.

Manually setting bias and variance

By default, `rdpow` uses `rdrobust` to estimate the bias and variance of the local polynomial estimator. However, the user can also manually specify the desired bias and variance used to calculate power. We will start by illustrating how to replicate the results from `rdrobust` but setting the bias and variance manually. The code to perform this calculation is as follows:

```
. quietly rdrobust demvoteshfor2 demmv
. local samph = e(h_l)
. local sampsi_l = e(N_h_l)
. local sampsi_r = e(N_h_r)
. local bias_l = e(bias_l)/e(h_l)
. local bias_r = e(bias_r)/e(h_r)
. matrix VL_RB = e(V_rb_l)
. matrix VR_RB = e(V_rb_r)
. local Vl_rb = e(N)*e(h_l)*VL_RB[1,1]
. local Vr_rb = e(N)*e(h_r)*VR_RB[1,1]
. rdpow demvoteshfor2 demmv, tau(5) bias(`bias_l´ `bias_r´)
> var(`Vl_rb´ `Vr_rb´) samph(`samph´) sampsi(`sampsi_l´ `sampsi_r´)
```

Cutoff c = 0	Left of c	Right of c		
Number of obs	595	702	Number of obs =	1297
Eff. Number of obs	359	322	BW type =	.
BW loc. poly. (h)	17.708	17.708	Kernel =	.
Order loc. poly. (p)	1	1	VCE method =	.
			Derivative =	0
Sampling BW	17.708	17.708	HA: tau =	5.000
New sample	359	322		

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.300	0.631	0.818

The first line simply runs `rdrobust` (omitting the output). The following three lines save the bandwidth and sample sizes inside the window. Next, we save the bias and variance to the left and right of the cutoff. Because these terms include the rates (see section 2), we must rescale the bias by dividing it by $h^{1+p-\nu}$ and the variance by $nh^{1+2\nu}$ (with $p = 1$ and $\nu = 0$ in this case). Once these magnitudes are obtained, we simply add them to the `rdpow` syntax using the `bias()` and `variance()` options. As we can see in the output above, the results are identical to those using `rdrobust` from within `rdpow`, with the difference that some values are missing from the output because `rdrobust` is actually not run.

The asymptotic variances of the local polynomial estimator at each side of the cutoff take the following form,

$$\mathsf{V}_+ \to_{\mathbb{P}} \frac{\sigma_+^2(c)}{f(c)} \times \mathcal{C}_+ \qquad \text{and} \qquad \mathsf{V}_- \to_{\mathbb{P}} \frac{\sigma_-^2(c)}{f(c)} \times \mathcal{C}_-$$

where $\sigma_+^2(c) = \lim_{x \downarrow c} \mathbb{V}\{Y_i(1)|X_i = x\}$, $\sigma_-^2(c) = \lim_{x \uparrow c} \mathbb{V}\{Y_i(0)|X_i = x\}$, $f(c)$ is the density of the running variable at the cutoff and $\mathcal{C}_-, \mathcal{C}_+$ are constants that capture the variability of the design (depending on the kernel and the order of the polynomial, among other things).

These terms may be hard to interpret in practice, so the user will rarely have a specific number for these magnitudes to specify as an option of `rdpow`. However, it is easy to answer the question of how the power decreases if the variance increases by, say, 20%. To do this, we simply take the variance estimates from `rdrobust` and multiply them by 1.2, as follows:

```
. quietly rdrobust demvoteshfor2 demmv
. matrix VL_RB = e(V_rb_l)
. matrix VR_RB = e(V_rb_r)
. local Vl_rb = e(N)*e(h_l)*VL_RB[1,1]*1.2
. local Vr_rb = e(N)*e(h_r)*VR_RB[1,1]*1.2
. rdpow demvoteshfor2 demmv, tau(5) var(`Vl_rb´ `Vr_rb´)
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708
New sample	359	322

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.082	0.258	0.554	0.745

We see that the power decreases from 0.818 to 0.745 after increasing the variance terms by 20%.

Finally, `rdpow` can be run without reference to the data. In practice, the researcher must input estimates of the biases, variances, sample sizes, and bandwidths at each side of the cutoff. The following code illustrates how to do this by using all the numbers calculated from `rdrobust`.

```
. quietly rdpow demvoteshfor2 demmv, tau(5)
. rdpow, tau(5) nsamples(r(N_l) r(N_h_l) r(N_r) r(N_h_r))
>                 bias(r(bias_l) r(bias_r))
>                 var(r(Vl_rb) r(Vr_rb)) sampsi(r(sampsi_l) r(sampsi_r))
>                 samph(r(samph_l) r(samph_r))
```

Cutoff c = 0	Left of c	Right of c
Number of obs	.	.
Eff. Number of obs	.	.
BW loc. poly. (h)	.	.
Order loc. poly. (p)	.	.
Sampling BW	17.708	17.708
New sample	359	322

Number of obs	.
BW type	.
Kernel	.
VCE method	.
Derivative	0
HA: tau	5.000

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.300	0.631	0.818

5.2 Comparing power across specifications

By changing the different options in `rdpow`, the user can compare the power given by different estimation or inference strategies, as described in section 2. The outputs below show the power for a given bandwidth (equal to 20) when using a linear and quadratic specification. The output shows that, for this given bandwidth, the linear specification has higher power.

```
. rdpow demvoteshfor2 demmv, tau(5) p(1) h(20) plot
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	389	346
BW loc. poly. (h)	20.000	20.000
Order loc. poly. (p)	1	1
Sampling BW	20.000	20.000
New sample	389	346

Number of obs	1297
BW type	Manual
Kernel	Triangular
VCE method	NN
Derivative	0
HA: tau	5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.080	0.248	0.534	0.724

```
. rdpow demvoteshfor2 demmv, tau(5) p(2) h(20) plot
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	389	346
BW loc. poly. (h)	20.000	20.000
Order loc. poly. (p)	2	2
Sampling BW	20.000	20.000
New sample	389	346

Number of obs =	1297
BW type =	Manual
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.067	0.162	0.339	0.488

The two outputs below illustrate the same idea but using optimal bandwidths for each specification. Using the optimal bandwidth increases power compared with the previous case, and again, the linear specification has higher ex-post power compared with the quadratic specification.

```
. rdpow demvoteshfor2 demmv, tau(5) p(1) plot
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708
New sample	359	322

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.300	0.631	0.818

```
. rdpow demvoteshfor2 demmv, tau(5) p(2) plot
```

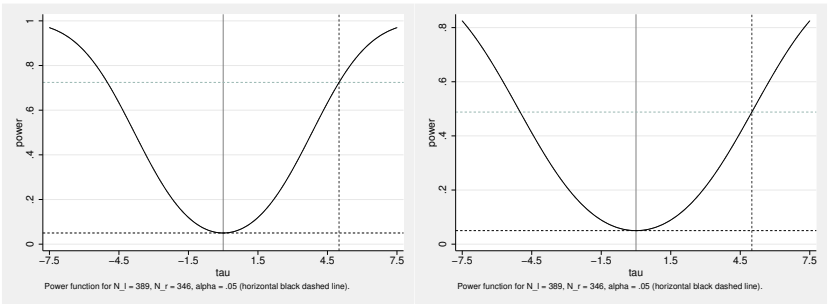
Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	409	370
BW loc. poly. (h)	22.210	22.210
Order loc. poly. (p)	2	2
Sampling BW	22.210	22.210
New sample	409	370

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000

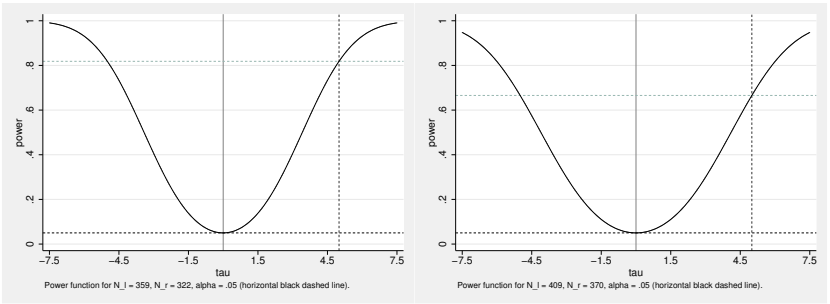
Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.077	0.223	0.480	0.666

These facts are illustrated in figure 2, which depicts the power functions for these cases.



(a) Linear, fixed bandwidth ($h = 20$) (b) Quadratic, fixed bandwidth ($h = 20$)



(c) Linear, MSE optimal bandwidth (d) Quadratic, MSE optimal bandwidth

Figure 2. Comparing power across specifications

5.3 Sample-size calculation using rdsampsi

The syntax of `rdsampsi` is similar to that of `rdrobust`. The most basic syntax to calculate the sample size to the left and right of the cutoff for an alternative of $\tau = 5$ is

```
. rdsampsi demvoteshfor2 demmv, tau(5)
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000
Power =	0.800

Outcome: demvoteshfor2. Running variable: demmv.

Chosen sample sizes	Sample size in window [c-h,c)	[c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	290	366	656	0.443

As with `rdpow`, the output of `rdsampsi` is divided in three panels. The upper-left panel displays the total number of observations at each side of the cutoff, the number of observations in the specified bandwidth (by default, the `rdrobust` bandwidth), the bandwidth chosen for `rdrobust`, and the sampling bandwidth chosen. The upper-right panel displays the total number of observations in the sample, some `rdrobust` options, the desired effect (τ), and the desired power. The main panel shows the number of observations inside the window at each side of the cutoff required to achieve the specified power and the proportion of units to the right of the cutoff inside the window. By default, this ratio is estimated using the variances of each group as explained in section 2. In this example, given the chosen parameters, we need 656 observations inside the window, 290 to the left, and 366 to the right of the cutoff.

The user can specify the desired power level, sampling bandwidths, and proportion of treated units, then plot the resulting power function (see figure 3). The syntax is the following:

```
. rdsampsi demvoteshfor2 demmv, tau(5) beta(.9) samph(18 19) nratio(.5) plot
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	18.000	19.000

Number of obs =	1297
BW type =	mserd
Kernel =	Triangular
VCE method =	NN
Derivative =	0
HA: tau =	5.000
Power =	0.900

Outcome: demvoteshfor2. Running variable: demmv.

Chosen sample sizes	Sample size in window [c-h,c)	[c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	431	431	862	0.500

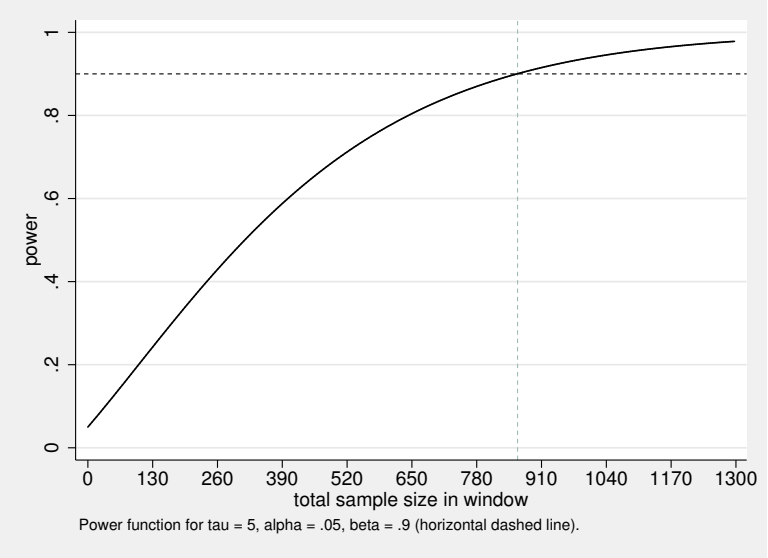


Figure 3. Robust bias-corrected power function

Note that the variances can be adjusted exactly as explained in the previous subsection for `rdpow`. As before, the `all` option adds the results based on the conventional approach to the output:

```
. rdsampsi demvoteshfor2 demmv, tau(5) all
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708

Number of obs	= 1297
BW type	= mserd
Kernel	= Triangular
VCE method	= NN
Derivative	= 0
HA: tau	= 5.000
Power	= 0.800
Size dist.	= 0.057

Outcome: demvoteshfor2. Running variable: demmv.

Chosen sample sizes	Sample size in window [c-h,c)	[c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	290	366	656	0.443
Conventional	173	208	381	0.453

The conventional approach yields a smaller sample size but generates a size distortion of 0.057.

A final check can be done by evaluating `rdpow` using the sample sizes obtained via `rdsampsi`, as follows:

```
. quietly rdsampsi demvoteshfor2 demmv, tau(5)
. rdpow demvoteshfor2 demmv, tau(5) sampsi(r(sampsi_h_l) r(sampsi_h_r))
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708
New sample	290	366

Number of obs	= 1297
BW type	= mserd
Kernel	= Triangular
VCE method	= NN
Derivative	= 0
HA: tau	= 5.000

Outcome: demvoteshfor2. Running variable: demmv.

Power against:	H0: tau= 0.000	0.2*tau = 1.000	0.5*tau = 2.500	0.8*tau = 4.000	tau = 5.000
Robust bias-corrected	0.050	0.088	0.296	0.625	0.813

As we can see from the above output, `rdsampsi` yields a slightly conservative sample size because of the rounding needed to obtain integer numbers. The obtained power is 0.813, which is close to the desired value of 0.8.

Like `rdpow`, `rdsampsi` can be used without data, as illustrated in the following code:

```
. quietly rdsampsi demvoteshfor2 demmv, tau(5)
. local init = r(init_cond)
. rdsampsi, tau(5) nsamples(r(N_l) r(N_h_l) r(N_r) r(N_h_r))
>                                bias(r(bias_l) r(bias_r))
>                                var(r(var_l) r(var_r))
>                                samph(r(samph_l) r(samph_r))
>                                init_cond(`init´)
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	.	.
Eff. Number of obs	.	.
BW loc. poly. (h)	.	.
Order loc. poly. (p)	.	.
Sampling BW	17.708	17.708

Number of obs	.
BW type	.
Kernel	.
VCE method	.
Derivative	0
HA: tau	5.000
Power	0.800

Chosen sample sizes	Sample size in window [c-h,c)	Sample size in window [c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	290	366	656	0.443

Finally, `rdsampsi` can be used to compare required sample sizes across specifications. Below, we compare required sample sizes for a local constant specification, which is the one used in a randomized control trial, and a local linear specification with a fixed bandwidth ($h = 20$) and the MSE optimal bandwidth.

```
. rdsampsi demvoteshfor2 demmv, tau(5) p(0) h(20) plot
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	389	346
BW loc. poly. (h)	20.000	20.000
Order loc. poly. (p)	0	0
Sampling BW	20.000	20.000

Number of obs	1297
BW type	Manual
Kernel	Triangular
VCE method	NN
Derivative	0
HA: tau	5.000
Power	0.800

Outcome: demvoteshfor2. Running variable: demmv.

Chosen sample sizes	Sample size in window [c-h,c)	Sample size in window [c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	204	239	443	0.460

```
. rdsampsi demvoteshfor2 demmv, tau(5) p(1) h(20) plot
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	389	346
BW loc. poly. (h)	20.000	20.000
Order loc. poly. (p)	1	1
Sampling BW	20.000	20.000

Number of obs	= 1297
BW type	= Manual
Kernel	= Triangular
VCE method	= NN
Derivative	= 0
HA: tau	= 5.000
Power	= 0.800

Outcome: demvoteshfor2. Running variable: demmv.

Chosen sample sizes	Sample size in window [c-h,c)	Sample size in window [c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	388	506	894	0.434

```
. rdsampsi demvoteshfor2 demmv, tau(5) p(0) plot
Calculating sample size...
Sample size obtained.
```

Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	143	131
BW loc. poly. (h)	5.906	5.906
Order loc. poly. (p)	0	0
Sampling BW	5.906	5.906

Number of obs	= 1297
BW type	= mserd
Kernel	= Triangular
VCE method	= NN
Derivative	= 0
HA: tau	= 5.000
Power	= 0.800

Outcome: demvoteshfor2. Running variable: demmv.

Chosen sample sizes	Sample size in window [c-h,c)	Sample size in window [c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	96	118	214	0.448

```
. rdsampsi demvoteshfor2 demmv, tau(5) p(1) plot
Calculating sample size...
Sample size obtained.
```

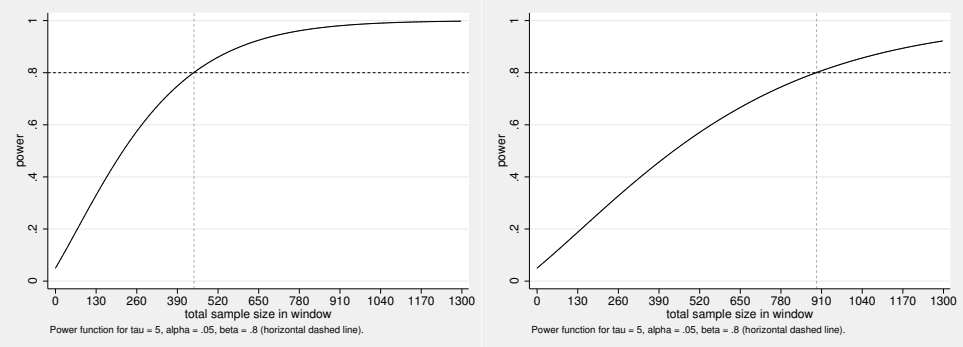
Cutoff c = 0	Left of c	Right of c
Number of obs	595	702
Eff. Number of obs	359	322
BW loc. poly. (h)	17.708	17.708
Order loc. poly. (p)	1	1
Sampling BW	17.708	17.708

Number of obs	= 1297
BW type	= mserd
Kernel	= Triangular
VCE method	= NN
Derivative	= 0
HA: tau	= 5.000
Power	= 0.800

Outcome: demvoteshfor2. Running variable: demmv.

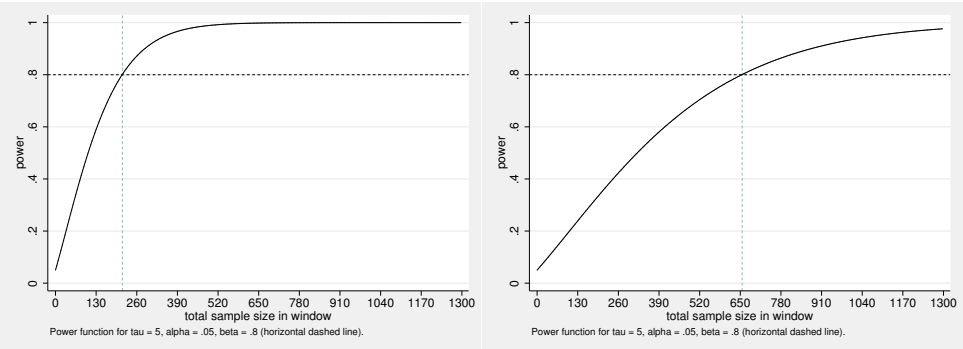
Chosen sample sizes	Sample size in window [c-h,c)	Sample size in window [c,c+h]	Total	Proportion [c,c+h]
Robust bias-corrected	290	366	656	0.443

These facts are illustrated in figure 4. We can see that, with both fixed and MSE optimal bandwidth choices, the local constant specification requires a smaller sample size (between half and a third of the required sample size for a local linear specification). The reason is that the linear specification introduces multicolinearity, which increases the variance of the local polynomial estimator. Goldberger (1972) pointed out this fact in the context of power calculations for classical linear regression settings. However, this feature should not be interpreted as implying that the constant specification is superior because, as discussed in section 2, a lower polynomial order implies a higher-order misspecification bias.



(a) Constant, fixed bandwidth ($h = 20$)

(b) Linear, fixed bandwidth ($h = 20$)



(c) Constant, MSE optimal bandwidth

(d) Linear, MSE optimal bandwidth

Figure 4. Comparing sample sizes across specifications

6 Conclusion

We introduced the command `rdpow` to perform power calculations and sample-size selection in RD designs using robust bias-corrected local polynomial inference techniques. These commands are particularly useful for planning new surveys and related field work

based on available RD data. We also provided a companion R function with the same syntax and capabilities.

7 Acknowledgments

This article and commands were motivated by impact evaluation work conducted at the Philippine Institute for Development Studies, Manila, Philippines, in the summers of 2014 and 2016, which were sponsored by the Asian Development Bank. We thank these institutions for their hospitality and support. We also thank an anonymous reviewer, Sebastian Calonico, David Drukker, Xinwei Ma, David Mckenzie, Aniceto Orbeta, and participants at short courses and workshops at the Abdul Latif Jameel Poverty Action Lab, the Asian Development Bank, the Inter-American Development Bank, and the Georgetown Center for Econometric Practice for useful questions, comments, and suggestions that improved this project. The authors gratefully acknowledge financial support from the National Science Foundation through grant SES-1357561.

8 References

Calonico, S., M. D. Cattaneo, and M. H. Farrell. 2018a. Coverage error optimal confidence intervals. ArXiv Working Paper No. arXiv:1808.01398. https://arxiv.org/abs/1808.01398.

———. 2018b. Optimal bandwidth choice for robust bias corrected inference in regression discontinuity designs. ArXiv Working Paper No. arXiv:1809.00236. https://arxiv.org/abs/1809.00236.

———. 2018c. On the effect of bias estimation on coverage accuracy in nonparametric inference. *Journal of the American Statistical Association* 113: 767–779.

Calonico, S., M. D. Cattaneo, M. H. Farrell, and R. Titiunik. 2017. rdrobust: Software for regression-discontinuity designs. *Stata Journal* 17: 372–404.

———. Forthcoming. Regression discontinuity designs using covariates. *Review of Economics and Statistics*.

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.

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

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., N. Idrobo, and R. Titiunik. Forthcoming a. *A Practical Introduction to Regression Discontinuity Designs: Volume I*. Cambridge: Cambridge University Press.

———. Forthcoming b. *A Practical Introduction to Regression Discontinuity Designs: Volume II*. Cambridge: Cambridge University Press.

Cattaneo, M. D., M. Jansson, and X. Ma. 2018a. Simple local polynomial density estimators. ArXiv Working Paper No. arXiv:1811.11512. https://arxiv.org/abs/1811.11512.

———. 2018b. Manipulation testing based on density discontinuity. *Stata Journal* 18: 234–261.

Cattaneo, M. D., R. Titiunik, and G. Vazquez-Bare. 2016. Inference in regression discontinuity designs under local randomization. *Stata Journal* 16: 331–367.

———. 2017. Comparing inference approaches for RD designs: A reexamination of the effect of head start on child mortality. *Journal of Policy Analysis and Management* 36: 643–681.

Goldberger, A. S. 1972. Selection bias in evaluating treatment effects: Some formal illustrations. Working paper, Institute for Research on Poverty Discussion Papers, University of Wisconsin–Madison.

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.

Sekhon, J., and R. Titiunik. 2017. On interpreting the regression discontinuity design as a local experiment. In *Advances in Econometrics: Vol. 38—Regression Discontinuity Designs: Theory and Applications*, ed. M. D. Cattaneo and J. C. Escanciano, 1–28. Bingley, UK: Emerald.

About the authors

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

Rocío Titiunik is the James Orin Murfin Associate Professor of Political Science at the University of Michigan.

Gonzalo Vazquez-Bare is a postdoctoral scholar and assistant professor of economics at the University of California at Santa Barbara.