Honest inference in Regression Discontinuity Designs

Michal Kolesár

February 03, 2020

Contents

1 Introduction

The package `RDHonest` implements confidence intervals for the regression discontinuity parameter considered in Armstrong and Kolesár [2018], Armstrong and Kolesár [2020], and Kolesár and Rothe [2018]. In this vignette, we demonstrate the implementation of these confidence intervals using datasets from Lee [2008], Oreopoulos [2006], and Battistin et al. [2009], which are included in the package as a data frame `lee08`, `cghs`, and `rcp`. The datasets from Lalive [2008] and Ludwig and Miller [2007] that are used in Armstrong and Kolesár [2020], and Kolesár and Rothe [2018] are also included in the package as data frames `rebp` and `headst`.

2 Sharp RD

2.1 Model

In the sharp regression discontinuity model, we observe units $i = 1, \ldots, n$, with the outcome y_i for the ith unit given by

$$y_i = f(x_i) + u_i,$$

where $f(x_i)$ is the expectation of y_i conditional on the running variable x_i and u_i is the regression error. A unit is treated if and only if the running variable x_i lies above a known cutoff c_0. The parameter of interest is given by the jump of f at the cutoff,

$$\beta = \lim_{x \downarrow c_0} f(x) - \lim_{x \uparrow c_0} f(x).$$

Let $\sigma^2(x_i)$ denote the conditional variance of u_i.

In the Lee dataset, the running variable corresponds to the margin of victory of a Democratic candidate in a US House election, and the treatment corresponds to winning the election. Therefore, the cutoff is zero. The outcome of interest is the Democratic vote share in the following election.

The Oreopoulos dataset consists of a subsample of British workers, and it exploits a change in minimum school leaving age in the UK from 14 to 15, which occurred in 1947. The running variable is the year in which the individual turned 14, with the cutoff equal to 1947 so that the "treatment" is being subject to a higher minimum school-leaving age. The outcome is log earnings in 1998.

Some of the functions in the package require the data to be transformed into a custom `RDData` format. This can be accomplished with the `RDData` function:

```
library("RDHonest")
## Assumes first column in the data frame corresponds to
## outcome, and second to running variable
dl <- RDData(lee08, cutoff = 0)
## Transform earnings to log earnings
do <- RDData(data.frame(logearn = log(cghs$earnings), year14 = cghs$yearat14),
    cutoff = 1947)
```

2.2 Plots

The package provides a function `plot_RDscatter` to plot the raw data. To remove some noise, the function plots averages over `avg` number of observations. The function takes an `RDData` object as an argument

```
## plot 25-bin averages in for observations 50 at most
## points away from the cutoff.  See Figure 1
plot_RDscatter(dl, avg = 25, window = 50, xlab = "Margin of victory",
    ylab = "Vote share in next election")
```

The running variable in the Oreopoulos dataset is discrete. It is therefore natural to plot the average outcome by each value of the running variable, which is achieved using by setting `avg=Inf`. The option

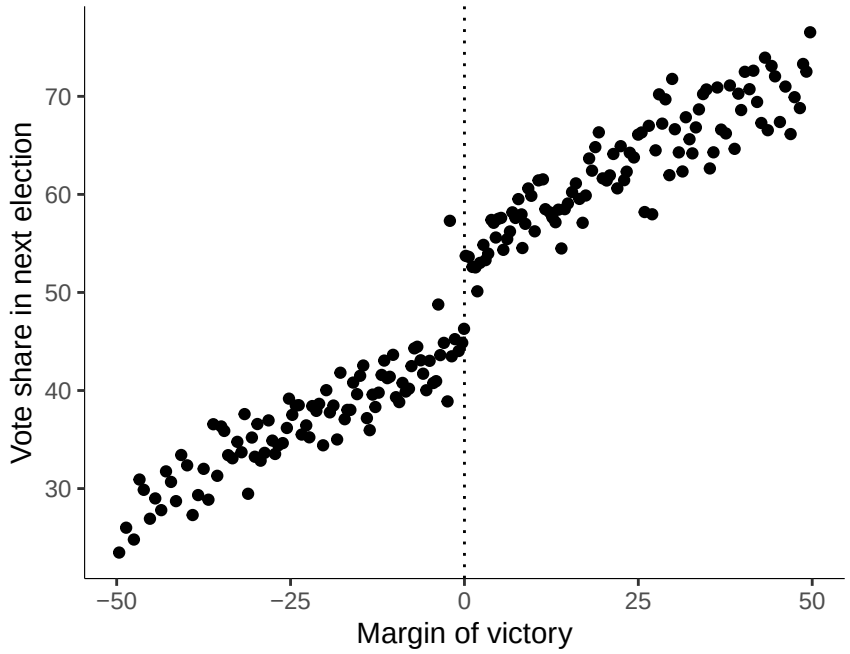

Figure 1: Lee (2008) data

`dotsize="count"` makes the size of the points proportional to the number of observations that the point averages over.

```
## see Figure 2
f2 <- plot_RDscatter(do, avg = Inf, xlab = "Year aged 14",
    ylab = "Log earnings", propdotsize = TRUE)
## Adjust size of dots if they are too big
f2 + ggplot2::scale_size_area(max_size = 4)
```

2.3 *Inference based on local polynomial estimates*

The function `RDHonest` constructs one- and two-sided confidence intervals (CIs) around local linear and local quadratic estimators using either a user-supplied bandwidth (which is allowed to differ on either side of the cutoff), or bandwidth that is optimized for a given performance criterion. The sense of honesty is that, if the regression errors are normally distributed with known variance, the CIs are guaranteed to achieve correct coverage *in finite samples*, and achieve correct coverage asymptotically uniformly over the parameter space otherwise. Furthermore, because the CIs explicitly take into account the possible bias of the estimators, the asymptotic approximation doesn't rely on the bandwidth to shrink to zero at a particular rate.

To describe the form of the CIs, let $\hat{\beta}_{h_+,h_-}$ denote a a local polynomial estimator with bandwidth equal to h_+ above the cutoff and equal to h_- below the cutoff. Let $\beta_{h_+,h_-}(f)$ denote its expectation conditional on the covariates when the regression function equals f. Then the bias of the estimator is given by $\beta_{h_+,h_-}(f) - \beta$. Let

$$B(\hat{\beta}_{h_+,h_-}) = \sup_{f \in \mathcal{F}} |\beta_{h_+,h_-}(f) - \beta|$$

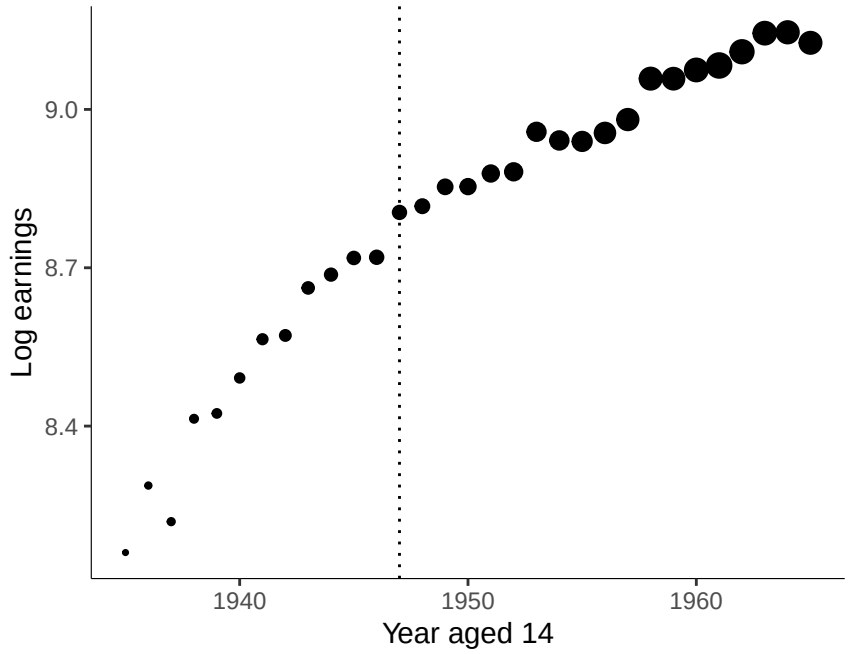

Figure 2: Oreopoulos (2006) data

denote the worst-case bias over the parameter space $\mathcal{F}$. Then the lower limit of a one-sided CI is given by

$$\hat{\beta}_{h_+,h_-} - B(\hat{\beta}_{h_+,h_-}) - z_{1-\alpha}\widehat{se}(\hat{\beta}_{h_+,h_-}),$$

where $z_{1-\alpha}$ is the $1-\alpha$ quantile of a standard normal distribution, and $\widehat{se}(\hat{\beta}_{h_+,h_-})$ is the standard error (an estimate of the standard deviation of the estimator). Subtracting the worst-case bias in addition to the usual critical value times standard error ensures correct coverage at all points in the parameter space.

A two-sided CI is given by

$$\hat{\beta}_{h_+,h_-} \pm cv_{1-\alpha}(B(\hat{\beta}_{h_+,h_-})/\widehat{se}(\hat{\beta}_{h_+,h_-})) \times \widehat{se}(\hat{\beta}_{h_+,h_-}),$$

where the critical value function $cv_{1-\alpha}(b)$ corresponds to the $1-\alpha$ quantile of the $|N(b,1)|$ distribution. To see why using this critical value ensures honesty, decompose the t-statistic as

$$\frac{\hat{\beta}_{h_+,h_-} - \beta}{\widehat{se}(\hat{\beta}_{h_+,h_-})} = \frac{\hat{\beta}_{h_+,h_-} - \beta_{h_+,h_-}(f)}{\widehat{se}(\hat{\beta}_{h_+,h_-})} + \frac{\beta_{h_+,h_-}(f) - \beta}{\widehat{se}(\hat{\beta}_{h_+,h_-})}$$

By a central limit theorem, the first term on the right-hand side will by distributed standard normal, irrespective of the bias. The second term is bounded in absolute value by $B(\hat{\beta}_{h_+,h_-})/\widehat{se}(\hat{\beta}_{h_+,h_-})$, so that, in large samples, the $1-\alpha$ quantile of the absolute value of the t-statistic will be bounded by $cv_{1-\alpha}(B(\hat{\beta}_{h_+,h_-})/\widehat{se}(\hat{\beta}_{h_+,h_-}))$. This approach gives tighter CIs than simply adding and subtracting $B(\hat{\beta}_{h_+,h_-})$ from the point estimate, in addition to adding and subtracting $z_{1-\alpha}\widehat{se}(\hat{\beta}_{h_+,h_-})$

The function `CVb` gives these critical values:

```
## Usual critical value
CVb(0, alpha = 0.05)  # returns a list
#>   bias alpha       cv TeXDescription
#> 1    0  0.05 1.959964 $\\alpha=0.05$
CVb(1/2, alpha = 0.05)$cv  # extract critical value
#> [1] 2.181477
## Tabulate critical values for different significance
## levels when bias-sd ratio equals 1/4
knitr::kable(CVb(1/4, alpha = c(0.01, 0.05, 0.1)), caption = "Critical values")
```

Table 1: Critical values

bias	alpha	cv	TeXDescription
0.25	0.01	2.652241	$\alpha = 0.01$
0.25	0.05	2.019713	$\alpha = 0.05$
0.25	0.10	1.695581	$\alpha = 0.1$

The field `TeXDescription` is useful for plotting, or for exporting to LaTeX, as in the table above.

Parameter space

To implement the honest CIs, one needs to specify the parameter space $\mathcal{F}$. The function `RDHonest` computes honest CIs when the parameter space $\mathcal{F}$ corresponds to a second-order Taylor or second-order Hölder smoothness class, which capture two different types of smoothness restrictions. The second-order Taylor class assumes that f lies in the the class of functions

$$\mathcal{F}_{\text{Taylor}}(M) = \{f_+ - f_- : f_+ \in \mathcal{F}_T(M; [c_0, \infty)),\ f_- \in \mathcal{F}_T(M; (-\infty, c_0))\},$$

where $\mathcal{F}_T(M; \mathcal{X})$ consists of functions f such that the approximation error from second-order Taylor expansion of $f(x)$ about c_0 is bounded by $M|x|^2/2$, uniformly over $\mathcal{X}$:

$$\mathcal{F}_T(M; \mathcal{X}) = \left\{f :\ |f(x) - f(c_0) - f'(c_0)x| \leq M|x|^2/2 \text{ all } x \in \mathcal{X}\right\}.$$

The class $\mathcal{F}_T(M; \mathcal{X})$ formalizes the idea that the second derivative of f at zero should be bounded by M. See Section 2 in Armstrong and Kolesár [2018] (note the constant C in that paper equals $C = M/2$ here). This class is doesn't impose smoothness away from boundary, which may be undesirable in some empirical applications. The Hölder class addresses this problem by bounding the second derivative globally. In particular, it assumes that f lies in the class of functions

$$\mathcal{F}_{\text{Hölder}}(M) = \{f_+ - f_- : f_+ \in \mathcal{F}_H(M; [c_0 f_- \in \mathcal{F}_H(M; (-\infty, c_0))\},$$

where

$$\mathcal{F}_H(M; \mathcal{X}) = \{f : |f'(x) - f'(y)| \leq M|x - y| \ \ x, y \in \mathcal{X}\}.$$

The smoothness class is specified using the option `sclass`. CIs around a local linear estimator with bandwidth that equals to 10 on either side of the cutoff when the parameter space is given by a Taylor and Hölder smoothness class, respectively, with $M = 0.1$:

```
RDHonest(voteshare ~ margin, data = lee08, kern = "uniform",
    M = 0.1, h = 10, sclass = "T")
#> Call:
#> RDHonest(formula = voteshare ~ margin, data = lee08, M = 0.1,
#>     kern = "uniform", h = 10, sclass = "T")
#>
#>
#> Inference by se.method:
#>    Estimate Maximum Bias Std. Error
#> nn 6.056774     3.782238   1.190527
#>
#> Confidence intervals:
#> nn    (0.3162933, 11.79725), (0.3162933, Inf), (-Inf, 11.79725)
#>
#> Bandwidth: 10
#> Number of effective observations: 292.3246
RDHonest(voteshare ~ margin, data = lee08, kern = "uniform",
    M = 0.1, h = 10, sclass = "H")
#> Call:
#> RDHonest(formula = voteshare ~ margin, data = lee08, M = 0.1,
#>     kern = "uniform", h = 10, sclass = "H")
#>
#>
#> Inference by se.method:
#>    Estimate Maximum Bias Std. Error
#> nn 6.056774     1.723768   1.190527
#>
#> Confidence intervals:
#> nn    ( 2.37473, 9.738817), (2.374763, Inf), (-Inf, 9.738784)
#>
#> Bandwidth: 10
#> Number of effective observations: 292.3246
```

The confidence intervals use the nearest-neighbor method to estimate the standard error by default (this can be changed using the option `se.method`, see help file for `RDHonest`). The package reports two-sided as well one-sided CIs (with lower as well as upper limit) by default.

Instead of specifying a bandwidth, one can just specify the smoothness class and smoothness constant M, and the bandwidth will be chosen optimally for a given optimality criterion:

```
RDHonest(voteshare ~ margin, data = lee08, kern = "triangular",
    M = 0.1, opt.criterion = "MSE", sclass = "H")
#> Call:
#> RDHonest(formula = voteshare ~ margin, data = lee08, M = 0.1,
#>     kern = "triangular", opt.criterion = "MSE", sclass = "H")
#>
#>
#> Inference by se.method:
#>    Estimate Maximum Bias Std. Error
#> nn 5.936649    0.8322587   1.294421
#>
#> Confidence intervals:
#> nn    (2.954829, 8.918468), (2.975258, Inf), (-Inf,  8.89804)
#>
#> Bandwidth: 8.848511
#> Number of effective observations: 213.463
## Choose bws optimal for length of CI, allowing for
## different bws on either side of cutoff
RDHonest(voteshare ~ margin, data = lee08, kern = "triangular",
    M = 0.1, opt.criterion = "FLCI", sclass = "H", bw.equal = FALSE)
#> Call:
#> RDHonest(formula = voteshare ~ margin, data = lee08, M = 0.1,
#>     kern = "triangular", opt.criterion = "FLCI", bw.equal = FALSE,
#>     sclass = "H")
#>
#>
#> Inference by se.method:
#>    Estimate Maximum Bias Std. Error
#> nn 5.960239    0.8809659   1.276487
#>
#> Confidence intervals:
#> nn    (2.964701, 8.955777), (2.979639, Inf), (-Inf,  8.94084)
#>
#> Bandwidth below cutoff: 8.804116
#> Bandwidth above cutoff: 9.380408
#> Number of effective observations: 220.3345
```

It is also possible to compute the optimal bandwidths directly using the function `RDOptBW`

```
RDOptBW(voteshare ~ margin, data = lee08, kern = "triangular",
    M = 0.1, opt.criterion = "MSE", sclass = "H")
#> Call:
#> RDOptBW(formula = voteshare ~ margin, data = lee08, M = 0.1,
```

```
#>     kern = "triangular", opt.criterion = "MSE", sclass = "H")
#>
#>
#> Bandwidth: 8.848511
```

Inference when running variable is discrete

The confidence intervals described above can also be used when the running variable is discrete, with G support points: their construction makes no assumptions on the nature of the running variable (see Section 5.1 in Kolesár and Rothe [2018] for more detailed discussion).

As an example, consider the Oreopoulos [2006] data, in which the running variable is age in years:

```
## Replicate Table 2, column (10)
RDHonest(log(earnings) ~ yearat14, cutoff = 1947, data = cghs,
    kern = "uniform", M = 0.04, opt.criterion = "FLCI",
    sclass = "H")
#> Call:
#> RDHonest(formula = log(earnings) ~ yearat14, data = cghs, cutoff = 1947,
#>     M = 0.04, kern = "uniform", opt.criterion = "FLCI", sclass = "H")
#>
#>
#> Inference by se.method:
#>      Estimate Maximum Bias Std. Error
#> nn 0.07909463   0.04736585 0.06784089
#>
#> Confidence intervals:
#> nn    (-0.08061322, 0.2388025), (-0.07985957, Inf), (-Inf, 0.2380488)
#>
#> Bandwidth: 2
#> Number of effective observations: 2017.075
## Triangular kernel generally gives tigher CIs
RDHonest(log(earnings) ~ yearat14, cutoff = 1947, data = cghs,
    kern = "triangular", M = 0.04, opt.criterion = "FLCI",
    sclass = "H")
#> Call:
#> RDHonest(formula = log(earnings) ~ yearat14, data = cghs, cutoff = 1947,
#>     M = 0.04, kern = "triangular", opt.criterion = "FLCI", sclass = "H")
#>
#>
#> Inference by se.method:
#>      Estimate Maximum Bias Std. Error
#> nn 0.07327071   0.05947669 0.05638953
#>
```

```
#> Confidence intervals:
#> nn    (-0.07900588, 0.2255473), (-0.07895849, Inf), (-Inf, 0.2254999)
#>
#> Bandwidth: 3.202072
#> Number of effective observations: 2265.826
```

In addition, the package provides function `RDHonestBME` that calculates honest confidence intervals under the assumption that the specification bias at zero is no worse at the cutoff than away from the cutoff as in Section 5.2 in Kolesár and Rothe [2018].

```
## Replicate Table 2, column (6), run local linear
## regression (order=1) with a uniform kernel (other
## kernels are not yet implemented)
RDHonestBME(log(earnings) ~ yearat14, cutoff = 1947, data = cghs,
    h = 3, order = 1)
#> Call:
#> RDHonestBME(formula = log(earnings) ~ yearat14, data = cghs,
#>     cutoff = 1947, h = 3, order = 1)
#>
#>
#> Confidence intervals:
#> (-0.06965587, 0.2019889)
```

Let us describe the implementation of the variance estimator $\hat{V}(W)$ used to construct the CI as described in in Section 5.2 in Kolesár and Rothe [2018]. Suppose the point estimate is given by the first element of the regression of the outcome y_i on $m(x_i)$. For instance, local linear regression with uniform kernel and bandwidth h corresponds to $m(x) = I(|x| \leq h) \cdot (I(x > c_0), 1, x, x \cdot I(x > c_0))'$. Let $\theta = Q^{-1}E[m(x_i)y_i]$, where $Q = E[m(x_i)m(x_i)']$, denote the estimand for this regression (treating the bandwidth as fixed), and let $\delta(x) = f(x) - m(x)'\theta$ denote the specification error at x. The RD estimate is given by first element of the least squares estimator $\hat{\theta} = \hat{Q}^{-1}\sum_i m(x_i)y_i$, where $\hat{Q} = \sum_i m(x_i)m(x_i)'$.

Let $w(x_i)$ denote a vector of indicator (dummy) variables for all support points of x_i within distance h of the cutoff, so that $\mu(x_g)$, where x_g is the gth support point of x_i, is given by the gth element of the regression estimand $S^{-1}E[w(x_i)y_i]$, where $S = E[w(x_i)w(x_i)']$. Let $\hat{\mu} = \hat{S}^{-1}\sum_i w(x_i)y_i$, where $\hat{S} = \sum_i w(x_i)w(x_i)'$ denote the least squares estimator. Then an estimate of $(\delta(x_1), \ldots, \delta(x_G))'$ is given by $\hat{\delta}$, the vector with elements $\hat{\mu}_g - x_g\hat{\theta}$.

By standard regression results, the asymptotic distribution of $\hat{\theta}$ and $\hat{\mu}$ is given by

$$\sqrt{n}\begin{pmatrix}\hat{\theta}-\theta\\ \hat{\mu}-\mu\end{pmatrix} \xrightarrow{d} \mathcal{N}(0,\Omega),$$

where

$$\Omega = \begin{pmatrix} Q^{-1}E[(\epsilon_i^2 + \delta(x_i)^2)m(x_i)m(x_i)']Q^{-1} & Q^{-1}E[\epsilon_i^2 m(x_i)w(x_i)']S^{-1} \\ S^{-1}E[\epsilon_i^2 w(x_i)m(x_i)']Q^{-1} & S^{-1}E[\epsilon_i^2 w(x_i)w(x_i)']S^{-1} \end{pmatrix}.$$

Let $\hat{u}_i$ denote the regression residual from the regression of y_i on $m(x_i)$, and let $\hat{\epsilon}_i$ denote the regression residuals from the regression of y_i on $w(x_i)$. Then a consistent estimator of the asymptotic variance Ω is

given by

$$\hat{\Omega} = n \sum_i T_i T_i', \qquad T_i' = \left(\hat{u}_i m(x_i)' \hat{Q}^{-1} \quad \hat{e}_i w(x_i)' \hat{S}^{-1} \right).$$

Note that the upper left block and lower right block correspond simply to the Eicker-Huber-White estimators of the asymptotic variance of $\hat{\theta}$ and $\hat{\mu}$. By the delta method, a consistent estimator of the asymptotic variance of $(\hat{\delta}, \hat{\theta}_1)$ is given by

$$\hat{\Sigma} = \begin{pmatrix} -X & I \\ e_1' & 0 \end{pmatrix} \hat{\Omega} \begin{pmatrix} -X & I \\ e_1' & 0 \end{pmatrix}',$$

where X is a matrix with gth row equal to x_g', and e_1 is the first unit vector.

Recall that in the notation of Kolesár and Rothe [2018], $W = (g^-, g^+, s^-, s^+)$, and g^+ and g^- are such that $x_{g^-} < c_0 \leq x_{g^+}$, and $s^+, s^- \in \{-1, 1\}$. An upper limit for a right-sided CI for $\theta_1 + b(W)$ is then given by

$$\hat{\theta}_1 + s^+ \hat{\delta}(x_{g^+}) + s^- \hat{\delta}(x_{g^-}) + z_{1-\alpha} \hat{V}(W),$$

where $\hat{V}(W) = a(W)' \hat{\Sigma} a(W)$, and $a(W) \in \mathbb{R}^{G_h + 1}$ denotes a vector with the g_-th element equal to s^-, $(G_h^- + g_+)$th element equal to s^+, the last element equal to one, and the remaining elements equal to zero. The rest of the construction then follows the description in Section 5.2 in Kolesár and Rothe [2018].

2.4 *Data-driven choice of smoothness constant*

Without further restrictions, the smoothness constant M cannot be data-driven: to maintain honesty over the whole function class, a researcher must choose M a priori, rather than attempting to use a data-driven method. Therefore, one should, whenever possible, use problem-specific knowledge to decide what choice of M is reasonable a priori.

For cases in which this is difficult, the function `NPR_MROT.fit` implements the method considered in Section 3.4.1 in Armstrong and Kolesár [2020] based on a global polynomial approximation:

```
## Data-driven choice of M
M <- NPR_MROT.fit(dl)
RDHonest(voteshare ~ margin, data = lee08, kern = "uniform",
    M = M, sclass = "H", opt.criterion = "MSE")
#> Call:
#>
#> RDHonest(formula = voteshare ~ margin, data = lee08, M = M, kern = "uniform",
#>     opt.criterion = "MSE", sclass = "H")
#>
#>
#> Inference by se.method:
#>    Estimate Maximum Bias Std. Error
#> nn  4.79817    0.8912105   1.571406
#>
#> Confidence intervals:
#> nn    (1.282989, 8.313351), (1.322226, Inf), (-Inf, 8.274114)
```

```
#>
#> Bandwidth: 6.011996
#> Number of effective observations: 169.2906
```

See Armstrong and Kolesár [2020] for a discussion of the restrictions on the parameter space under which this method yields honest inference.

2.5 *Optimal inference*

For the second-order Taylor smoothness class, the function `RDHonest`, with `kernel="optimal"`, computes finite-sample optimal estimators and confidence intervals, as described in Section 2.2 in Armstrong and Kolesár [2018]. This typically yields tighter CIs. Comparing the lengths of two-sided CIs with optimally chosen bandwidths, using Silverman's rule of thumb to estimate the preliminary variance estimate used to compute optimal bandwidths:

```
2 * RDHonest(voteshare ~ margin, data = lee08, kern = "optimal",
    M = 0.1, opt.criterion = "FLCI", se.initial = "Silverman",
    se.method = "nn")$hl
#>       nn
#> 6.294084
2 * RDHonest(voteshare ~ margin, data = lee08, kern = "triangular",
    M = 0.1, opt.criterion = "FLCI", se.initial = "Silverman",
    se.method = "nn", sclass = "T")$hl
#>       nn
#> 6.648266
```

2.6 *Specification testing*

The package also implements lower-bound estimates for the smoothness constant M for the Taylor and Hölder smoothness class, as described in the supplements to Kolesár and Rothe [2018] and Armstrong and Kolesár [2018]

```
## Add variance estimate to the lee data so that the
## RDSmoothnessBound function doesn't have to compute
## them each time
dl <- NPRPrelimVar.fit(dl, se.initial = "nn")
### Only use three point-average for averages of a 100
### points closest to cutoff, and report results
### separately for points above and below cutoff
RDSmoothnessBound(dl, s = 100, separate = TRUE, multiple = FALSE,
    sclass = "T")
#>
#> Smoothness bound estimate using observations above cutoff:
#> Estimate: 0.1727232, Lower CI: [       0, Inf)
#>
```

```
#> Delta: 0.1833015, sd=0.1792815
#> E_n[f(x_1)]: 53.12906, I1=[0.01128614, 1.749364]
#> E_n[f(x_2)]: 53.14677, I2=[1.751874,  3.43698]
#> E_n[f(x_3)]: 56.17622, I3=[3.445935, 4.896765]
#>
#> Smoothness bound estimate using observations below cutoff:
#> Estimate: 0.3337537, Lower CI: [0.1388378, Inf)
#>
#> Delta: -0.3337537, sd=0.1184773
#> E_n[f(x_1)]: 44.95894, I1=[0.03081858, 1.953077]
#> E_n[f(x_2)]: 46.60759, I2=[2.002648, 3.754386]
#> E_n[f(x_3)]: 41.76886, I3=[3.785086, 5.494842]
### Pool estimates based on observations below and above
### cutoff, and use three-point averages over the entire
### support of the running variable
RDSmoothnessBound(dl, s = 100, separate = FALSE, multiple = TRUE,
    sclass = "H")
#>
#> Smoothness bound estimate:
#> Estimate: 0.2293755, Lower CI: [0.02500725, Inf)
#>
#> Delta: -2.15055, sd=0.7634111
#> E_n[f(x_1)]: 44.95894, I1=[0.03081858, 1.953077]
#> E_n[f(x_2)]: 46.60759, I2=[2.002648, 3.754386]
#> E_n[f(x_3)]: 41.76886, I3=[3.785086, 5.494842]
```

2.7 *Weighted regression*

In some cases, data is only observed as cell averages. For instance, suppose that instead of observing the original `cghs` data, we only observe averages for cells as follows:

```
d <- cghs
## Make 20 groups based on observation number
d$mod <- seq_along(d$yearat14)%%20
## Make cells defined as intersection of group and year
d$cell <- d$mod/100 + d$yearat14
## Data with cell averages
dd <- data.frame()
for (j in unique(d$cell)) {
    dd <- rbind(dd, data.frame(y = mean(log(d$earnings)[d$cell ==
        j]), x = mean(d$yearat14[d$cell == j]), weights = length(d$yearat14[d$cell ==
        j])))
}
```

The column `weights` gives the number of observations that each cell averages over. In this case, if we weight the observations using `weights`, we can recover the original estimates (and the same worst-case bias):

```
RDHonest(log(earnings) ~ yearat14, cutoff = 1947, h = 5,
    data = cghs, M = 0.1, se.method = c("EHW", "nn"))
#> Call:
#> RDHonest(formula = log(earnings) ~ yearat14, data = cghs, cutoff = 1947,
#>     M = 0.1, h = 5, se.method = c("EHW", "nn"))
#>
#>
#> Inference by se.method:
#>       Estimate Maximum Bias Std. Error
#> EHW 0.06210687    0.2717834 0.04297756
#> nn  0.06210687    0.2717834 0.04299237
#>
#> Confidence intervals:
#> EHW   (-0.2803683, 0.4045821), (-0.2803683, Inf), (-Inf, 0.4045821)
#> nn    (-0.2803927, 0.4046064), (-0.2803927, Inf), (-Inf, 0.4046064)
#>
#> Bandwidth: 5
#> Number of effective observations: 3153.642
RDHonest(y ~ x, cutoff = 1947, weights = weights, h = 5,
    data = dd, M = 0.1, se.method = c("EHW", "nn"))
#> Call:
#> RDHonest(formula = y ~ x, data = dd, weights = weights, cutoff = 1947,
#>     M = 0.1, h = 5, se.method = c("EHW", "nn"))
#>
#>
#> Inference by se.method:
#>       Estimate Maximum Bias Std. Error
#> EHW 0.06210687    0.2717834 0.04250739
#> nn  0.06210687    0.2717834 0.04362112
#>
#> Confidence intervals:
#> EHW   (-0.279595, 0.4038087), (-0.279595, Inf), (-Inf, 0.4038087)
#> nn    (-0.2814269, 0.4056406), (-0.2814269, Inf), (-Inf, 0.4056406)
#>
#> Bandwidth: 5
#> Number of effective observations: 43.05709
```

Note the variance estimates don't quite match, since the variance estimator is different, but the worst-case bias and the point estimate are identical.

3 Fuzzy RD

3.1 Model

In a fuzzy RD design, the treatment d_i is not entirely determined by whether the running variable x_i exceeds a cutoff. Instead, the cutoff induces a jump in the treatment probability. The resulting reduced-form and first-stage regressions are given by

$$y_i = f_1(x_i) + u_{i1}, \qquad d_i = f_2(d_i) + u_{i2},$$

See Section 3.3 in Armstrong and Kolesár [2020] for a more detailed description.

In the Battistin et al dataset, the treatment variable is an indicator for retirement, and the running variable is number of years since being eligible to retire. The cutoff is 0. (individuals exactly at the cutoff are dropped). Similarly to the `RDData` function, the `FRDData` function transforms the data into an appropriate format:

```
## Assumes first column in the data frame corresponds to
## outcome, second to the treatment variable, and third
## to the running variable Outcome here is log of
## non-durables consumption
dr <- FRDData(cbind(logf = log(rcp[, 6]), rcp[, c(3, 2)]),
    cutoff = 0)
```

3.2 Inference based on local polynomial estimates

The function `FRDHonest` constructs one- and two-sided confidence intervals (CIs) around local linear and local quadratic estimators using either a user-supplied bandwidth (which is allowed to differ on either side of the cutoff), or bandwidth that is optimized for a given performance criterion.

Parameter space and initial estimate

To implement the honest CIs, one needs to specify the parameter space $\mathcal{F}$ for f_1 and f_2. The function `FRDHonest` computes honest CIs when f_1 and f_2 both lie in a second-order Taylor or second-order Hölder smoothness class, $\mathcal{F}_T(M_1, M_2)$ and $\mathcal{F}_{\text{Hölder}}(M_1, M_2)$, where the smoothness constants M_1 and M_2 for the reduced form and the first stage are allowed to differ. Also, since the worst-case bias calculation requires an estimate of the treatment effect, for optimal bandwidth calculations, the user needs to supply an initial estimator of the treatment effect

```
## Initial estimate of treatment effect for optimal
## bandwidth calculations
r <- FRDHonest(log(cn) ~ retired | elig_year, data = rcp,
    kern = "triangular", M = c(0.001, 0.002), opt.criterion = "MSE",
    sclass = "H", T0 = 0)
## Use it to compute optimal bandwidth
FRDHonest(log(cn) ~ retired | elig_year, data = rcp, kern = "triangular",
    M = c(0.001, 0.002), opt.criterion = "MSE", sclass = "H",
```

```
    T0 = r$estimate)
#> Call:
#> FRDHonest(formula = log(cn) ~ retired | elig_year, data = rcp,
#>     M = c(0.001, 0.002), kern = "triangular", opt.criterion = "MSE",
#>     sclass = "H", T0 = r$estimate)
#>
#>
#> Inference by se.method:
#>       Estimate Maximum Bias Std. Error
#> nn -0.09062175    0.0437435  0.0716225
#>
#> Confidence intervals:
#> nn    (-0.253552, 0.07230845), (-0.2521738, Inf), (-Inf, 0.07093028)
#>
#> Bandwidth: 9.551737
#> Number of effective observations: 1485.949
```

It is also possible to compute the optimal bandwidths directly using the function `RDOptBW`

```
FRDOptBW(log(cn) ~ retired | elig_year, data = rcp, kern = "triangular",
    M = c(0.001, 0.002), opt.criterion = "MSE", sclass = "H",
    T0 = r$estimate)
#> Call:
#> FRDOptBW(formula = log(cn) ~ retired | elig_year, data = rcp,
#>     M = c(0.001, 0.002), kern = "triangular", opt.criterion = "MSE",
#>     sclass = "H", T0 = r$estimate)
#>
#>
#> Bandwidth: 9.551737
```

3.3 *Data-driven choice of smoothness constant*

Like in the sharp RD case, Without further restrictions, the smoothness constants M_1 and M_2 cannot be data-driven: to maintain honesty over the whole function class, a researcher must choose them a priori, rather than attempting to use a data-driven method. Therefore, one should, whenever possible, use problem-specific knowledge to decide what choices of M_1 and M_2 are reasonable a priori.

For cases in which this is difficult, the function `NPR_MROT.fit` implements the method considered in Section 3.4.1 in Armstrong and Kolesár [2020] based on a global polynomial approximation:

```
## Data-driven choice of M
M <- NPR_MROT.fit(dr)
print(M)
#>          M1          M2
#> 0.002849525 0.008178929
```

```
FRDHonest(log(cn) ~ retired | elig_year, data = rcp, kern = "triangular",
    M = M, opt.criterion = "MSE", sclass = "H", T0 = r$estimate)
#> Call:
#> FRDHonest(formula = log(cn) ~ retired | elig_year, data = rcp,
#>     M = M, kern = "triangular", opt.criterion = "MSE", sclass = "H",
#>     T0 = r$estimate)
#>
#>
#> Inference by se.method:
#>      Estimate Maximum Bias Std. Error
#> nn -0.1762096   0.08473374  0.1042236
#>
#> Confidence intervals:
#> nn    (-0.4329102, 0.08049089), (-0.4323759, Inf), (-Inf, 0.07995661)
#>
#> Bandwidth: 6.127526
#> Number of effective observations: 822.471
```

See Armstrong and Kolesár [2020] for a discussion of the restrictions on the parameter space under which this method yields honest inference.

4 Inference at a point

The package also contains functions `LPPHonest` and `LPPOptBW` for inference at a point, and optimal bandwidth selection for inference at a point. Suppose, for example, one was interested in the vote share for candidates with margin of victory equal to 20 points:

```
## Transform data, specify we're interested in inference
## at x0=20, and drop observations below cutoff
leep <- lee08[lee08$margin > 0, ]
## Data-driven choice of M
M <- NPR_MROT.fit(LPPData(leep, point = 20))
print(M)
#> [1] 0.0275703
LPPHonest(voteshare ~ margin, data = leep, point = 20, kern = "uniform",
    M = M, opt.criterion = "MSE", sclass = "H")
#> Call:
#> LPPHonest(formula = voteshare ~ margin, data = leep, point = 20,
#>     M = M, kern = "uniform", opt.criterion = "MSE", sclass = "H")
#>
#>
#> Inference by se.method:
#>    Estimate Maximum Bias Std. Error
#> nn 61.66394    0.2482525   0.468336
```

```
#>
#> Confidence intervals:
#> nn    (60.63086, 62.69703), (60.64535, Inf), (-Inf, 62.68254)
#>
#> Bandwidth: 7.311286
#> Number of effective observations: 737.7866
```

References

Timothy B. Armstrong and Michal Kolesár. Optimal inference in a class of regression models. *Econometrica*, 86(2):655–683, March 2018. doi: 10.3982/ECTA14434.

Timothy B. Armstrong and Michal Kolesár. Simple and honest confidence intervals in nonparametric regression. *Quantitative Economics*, 11(1):1–39, January 2020. doi: 10.3982/QE1199.

Erich Battistin, Agar Brugiavini, Enrico Rettore, and Guglielmo Weber. The retirement consumption puzzle: Evidence from a regression discontinuity approach. *American Economic Review*, 99(5):2209–2226, 2009. doi: 10.1257/aer.99.5.2209.

Michal Kolesár and Christoph Rothe. Inference in regression discontinuity designs with a discrete running variable. *American Economic Review*, 108(8):2277–2304, August 2018. doi: 10.1257/aer.20160945.

Rafael Lalive. How do extended benefits affect unemployment duration? a regression discontinuity approach. *Journal of Econometrics*, 142(2):785–806, February 2008. doi: 10.1016/j.jeconom.2007.05.013.

David S. Lee. Randomized experiments from non-random selection in U.S. House elections. *Journal of Econometrics*, 142(2):675–697, 2008. doi: 10.1016/j.jeconom.2007.05.004.

Jens Ludwig and Douglas L. Miller. Does head start improve children's life chances? evidence from a regression discontinuity design. *Quarterly Journal of Economics*, 122(1):159–208, February 2007. doi: 10.1162/qjec.122.1.159.

Philip Oreopoulos. Estimating average and local average treatment effects when compulsory education schooling laws really matter. *American Economic Review*, 96(1):152–175, 2006. doi: 10.1257/000282806776157641.