RDDtools: an overview

Matthieu Stigler
Affiliation IHEID

Abstract

RDDtools is a R package for sharp regression discontinuity design (RDD). It offers various estimators, tests and graphical procedures following the guidelines of Imbens and Lemieux (2008) and Lee and Lemieux (2010). This note illustrate how to use the package, using the well-known dataset of Lee (2008).

NOTE THAT this is a preliminary note, on a preliminary package still under development. Changes of the function names, arguments and output are to be expected, as well as possible mistakes and inconsistencies. Please report any mistakes or suggestion to Matthieu.Stigler@iheid.ch

Keywords: Regression discontinuity design, non-parametric analysis, **RDDtools**, R.

Contents

1. Introduction

1.1. Introduction to RDD

1.2. Introduction to RDDtools

The R package **RDDtools** aims at offering a complete a toolbox for regression discontinuity design, following the step-by-step recommendations of Imbens and Lemieux (2008) and Lee

and Lemieux (2010). Summarising the approaches advocated in the two papers, a RDD analysis comprises of following steps:

1. Graphical representation of the data
2. Estimation
3. Validity tests

We add to this list a step that is too often forgotten, yet can be very burdensome: data preparation. Hence, this list is extended with the fundamental step 0, which involves preparing the data in the right way.

RDDtools offers an object-oriented way to analysis, building on the R mechanism of S3 methods and classes. Concretely, this implies that the user has to specify the input data only once, and that most of the functions can be called directly on the new object of class `RDDdata`.

2. Step 0: data input

As first step of the analysis, the user has to specify the input data into the `RDDdata` function, which takes following arguments:

y The outcome variable

x The forcing variable

cutpoint The cutpoint/threshold (note only one cutpoint can be given)

z Eventual covariates

The RDDdata function returns an object of class `RDDdata`, as well as of the usual R class `data.frame`.

To illustrate this, we show how to use this with the benchmark dataset of Lee (2008), adding randomly generated covariates for the sake of illustration. The dataset is shipped with the package, and is available under the name *Lee2008*. Using the R `head` function, we look at the first rows of the dataset:

```
library(RDDtools)

## KernSmooth 2.23 loaded
## Copyright M. P. Wand 1997-2009
##
## RDDtools 0.1 (rev Wednesday 2013-04-24).  PLEASE NOTE THIS is currently only
a development version.
## Run vignette('RDDtools') for the documentation

data(Lee2008)
head(Lee2008)
```

```
##         x      y
## 1  0.1049 0.5810
## 2  0.1393 0.4611
## 3 -0.0736 0.5434
## 4  0.0868 0.5846
## 5  0.3994 0.5803
## 6  0.1681 0.6244
```

The data is already clean, so the only step required is to fit it into the RDDdata function, adding however the information on the cutpoint. For illustration purpose, we add also some random covariates as a matrix Z:

```
n_Lee <- nrow(Lee2008)
Z <- data.frame(z1 = rnorm(n_Lee), z2 = rnorm(n_Lee, mean = 20, sd = 2), z3 = sample(lette
    size = n_Lee, replace = TRUE))
Lee2008_rdd <- RDDdata(y = Lee2008$y, x = Lee2008$x, z = Z, cutpoint = 0)
```

We now have an object `Lee2008_rdd` of class `RDDdata` (and `data.frame`). It has a specific `summary` method, which gives a few summary informations about the dataset:

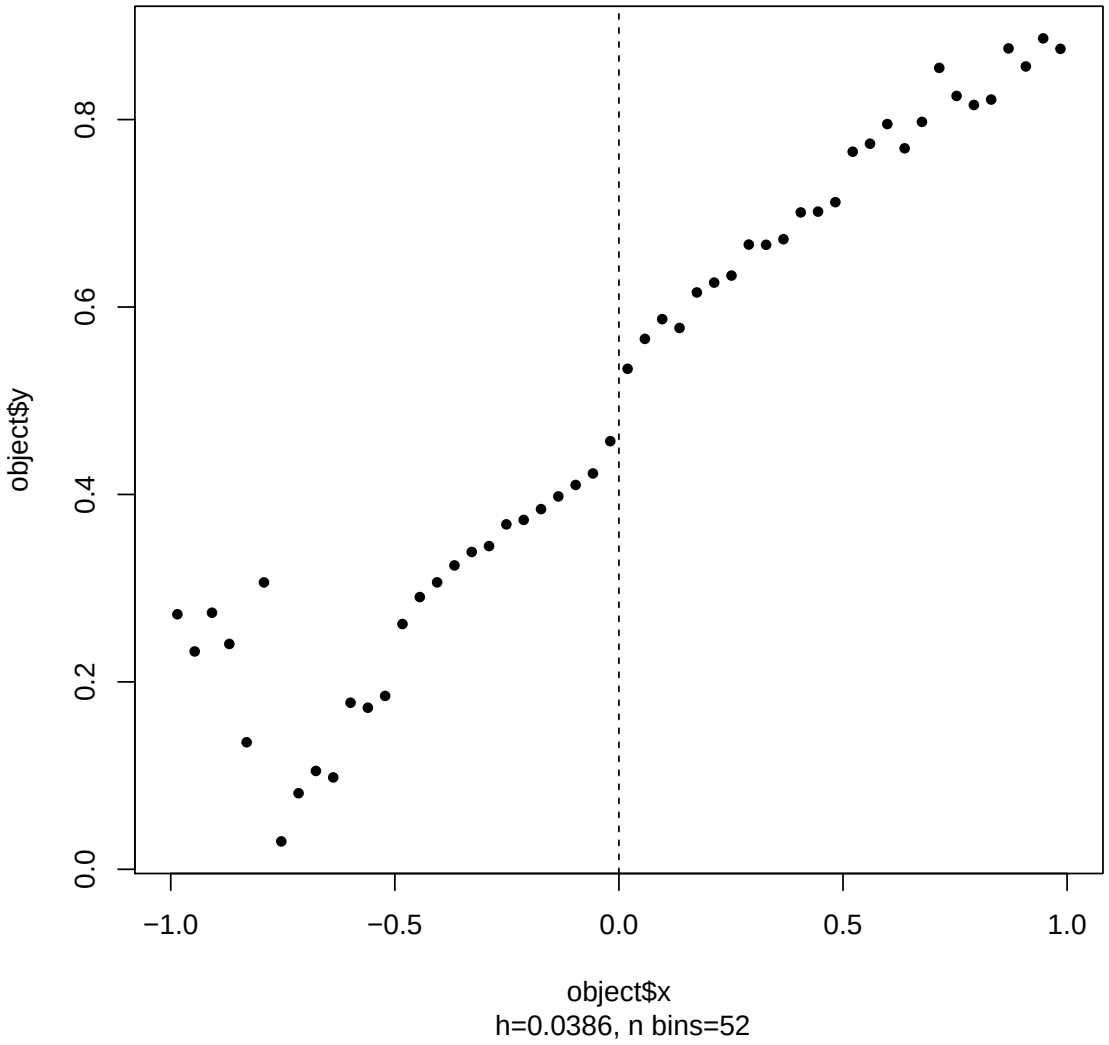
```
summary(Lee2008_rdd)

## ### RDDdata object ###
##
## Cutpoint: 0
## Sample size:
##  -Full : 6558
##  -Left : 2740
##  -Right: 3818
## Covariates: yes
```

Another function for `RDDdata` objects is the `plot()` function, discussed in the next section.

3. Step 1: Graphical representation

Once the dataset has been formatted with the RDDdata function, it can be used directly for simple illustration. Indeed, as recommended by Lee and Lemieux (2010), it is always good to show the raw data first, if ones wishes to convince that there is a discontinuity. This is simply done using the standard R plot() function, which has been customised for RDDdata objects. The function shows a scatter plot of the outcome variable against the forcing variable. Following Lee and Lemieux (2010), not all single datapoints are shown: instead, a "binned" scatterplot is shown, using non-overlapping averages:

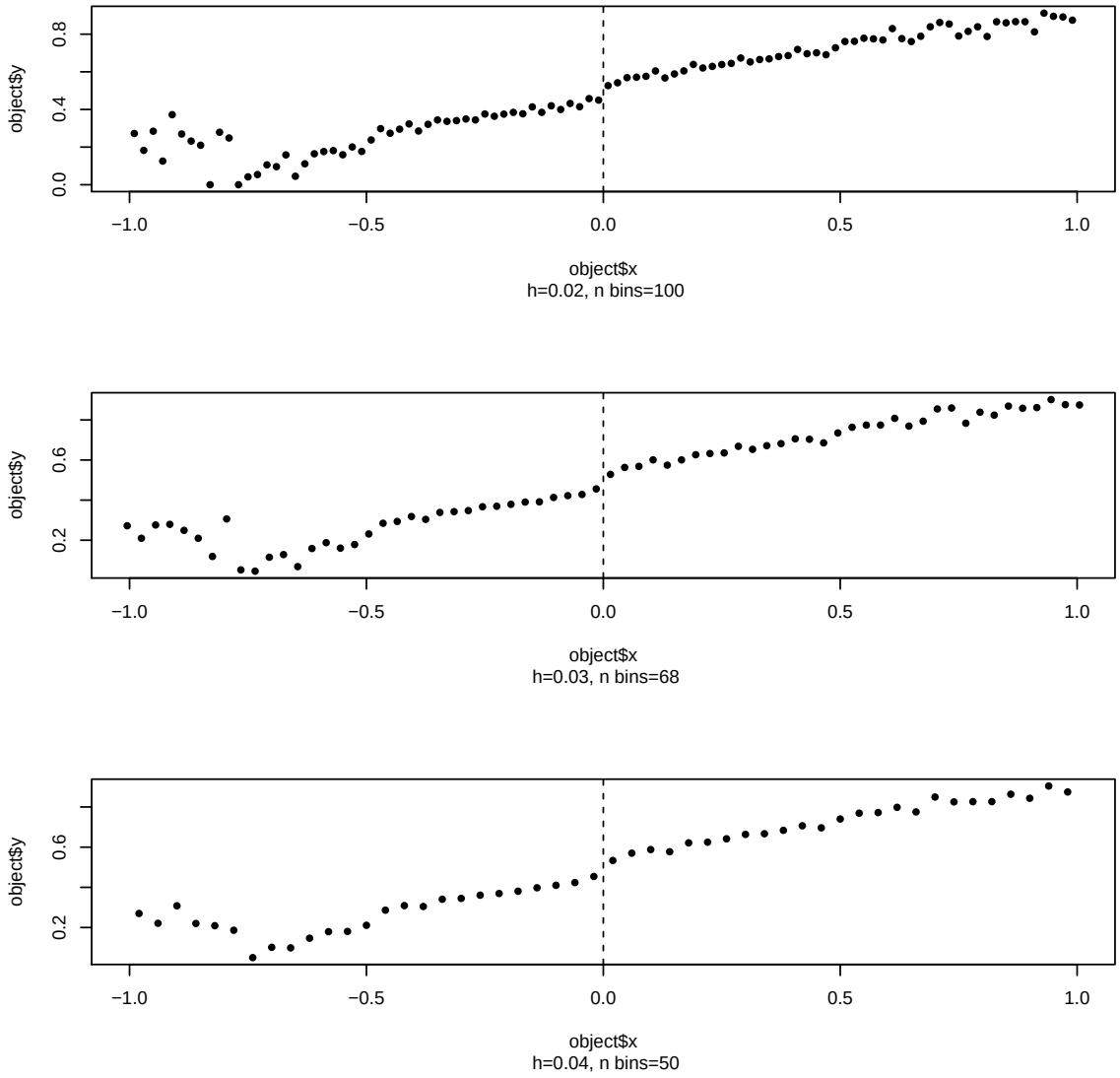
```
plot(Lee2008_rdd)

## Warning:  font width unknown for character 0x9
```



The bandwidth for the bins (also called binwidth) can be set by the user with the `h` argument. If this it is not provided by the user, the function uses by default the global bandwidth of Ruppert, Sheather, and Wand (1995), implemented in the `RDDbw_RSW()` function.

Another argument that might be useful for the user is the option `nplot`, which allows to plot multiple plots with different bandwidths:

```
plot(Lee2008_rdd, nplot = 3, h = c(0.02, 0.03, 0.04))

## Warning:  font width unknown for character 0x9
## Warning:  font width unknown for character 0x9
## Warning:  font width unknown for character 0x9
```



Note however that experience shows that showing multiple plots have the effect to shrink considerably the y axis, reducing the visual impression of discontinuity.

4. Step 2: Estimation

RDDtools offers currently two estimators:

- the simple parametric estimator: function `RDDreg_lm()`.
- the non-parametric local-linear estimator: function `RDDreg_np()`.

These two functions share some common arguments, which are:

RDDobject: the input data as obtained with the `RDDdata()` function

bw: the bandwidth.

covariates: this will allow to add covariates in the analysis. Note that it is presently NOT used.

The bandwidth argument has a different behaviour in the parametric and non-parametric way: while the parametric estimation can be done without bandwidth, the non-parametric estimator is by definition based on a bandwidth. This means that the default behaviours are different: if no bandwidth is given for the parametric model, the model will be simply estimated withut bandwidth, that is covering the full sample on both sides of the cutpoint. On the other side, if no bandwidth is provided in the non-parametric case, a bandwidth will still be computed automatically using the method advocated by Imbens and Kalyanaraman (2012).

4.1. Parametric

The parametric estimator simply estimates a function over the whole sample (hence called *pooled regression* by Lee and Lemieux 2010):

$$Y = \alpha + \tau D + \beta(X - c) + \epsilon \tag{1}$$

where D is a dummy variable, indicating whether the observations are above (or equal to) the cutoff point, i.e. $D = I(X \geq c)$. The parameter of interest is τ, which represents the difference in intercepts $\alpha_r - \alpha_l$, i.e. the discontinuity. Note that equation 1 imposes the slope to be equal on both sides of the cutoff point. While such restriction should hold locally around the threshold (due to the assumption of random assignment around the cutoff point), the parametric regression is done by default using the whole sample, so the restriction is unlikely to hold. In this case, one should rather estimate:

$$Y = \alpha + \tau D + \beta_1(X - c) + \beta_2 D(X - c) + \epsilon \tag{2}$$

so that $\beta_1 = \beta_l$, and $\beta_2 = \beta_r - \beta_l$.

The two estimators are available with the `RDDreg_lm()` function, the choice between the specifications being made through the `slope=c("separate", "same")` argument:

separate: the default, estimates different slopes, i.e. equation 2.

same: Estimates a common slope, i.e. equation 1.

Note that the order of X has been set as 1 in both cases. If the function shows moderate non-linearity, this can be potentially captured by adding further power of X, leading to (for the separate slope equation:)

$$Y = \alpha + \tau D + \beta_1^1(X - c) + \beta_2^1 D(X - c) + \ldots + \beta_1^p(X - c)^p + \beta_2^p D(X - c)^p + \epsilon \tag{3}$$

The order of the polynomial can be adjusted with the `order` argument.

Finally, the estimator can be restricted to a (symmetric) window around the cutoff point, as is done usually in practice. This is done using the `bw` option.

In summary, the function `RDDreg_lm()` has three main options:

slope: Whether to use different slopes on each side of the cutoff (default) or not.

order: Order of the polynomial in X. Default to 1.

bw: Eventual window to estimate the data. Default to full data.

We show now the different applications, still using the Lee dataset:

```
reg_linear_1 <- RDDreg_lm(Lee2008_rdd)
```

We now estimate different versions, first restricting the slope to be the same, then changing the order, and finally using a smaller window:

```
reg_linear_2 <- RDDreg_lm(Lee2008_rdd, slope = "separate")
reg_linear_3 <- RDDreg_lm(Lee2008_rdd, order = 3)
reg_linear_4 <- RDDreg_lm(Lee2008_rdd, bw = 0.4)
```

Model's output is shown with the `print()` and `summary()` function: while the `print()` function just shows few informations and the LATE estimate, the `summary()` function shows the full output of the underlying regression model:

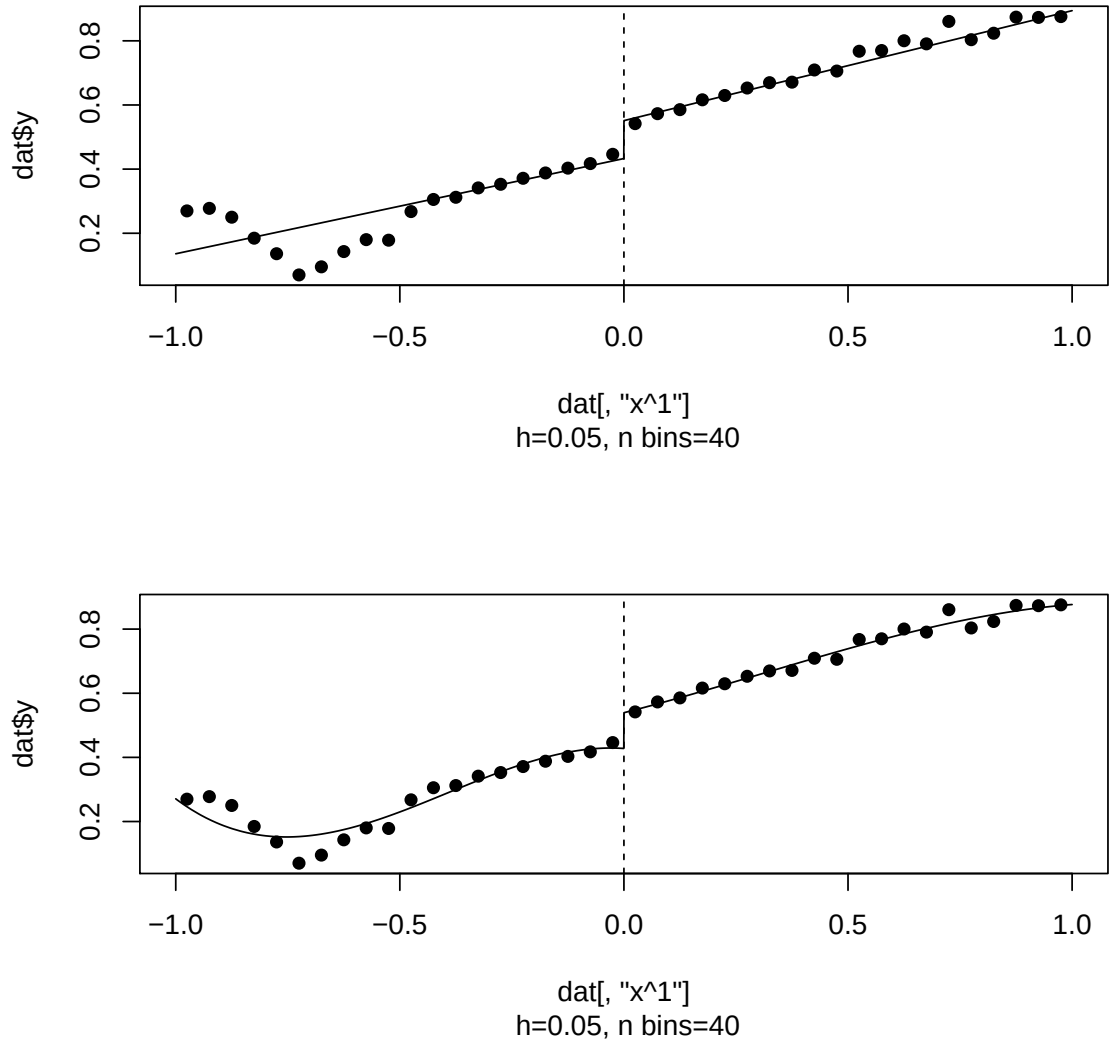
```
reg_linear_1

## ### RDD regression: parametric ###
##  Polynomial order:  1
##  Slopes:  separate
##  Number of obs: 6558 (left: 2740, right: 3818)
##
##  Coefficient:
##   Estimate Std. Error t value Pr(>|t|)
## D  0.11823    0.00568    20.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

summary(reg_linear_1)

##
## Call:
## lm(formula = y ~ ., data = dat_step1, subset = isIn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8941 -0.0619  0.0023  0.0713  0.8640
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.43295    0.00428  101.25  < 2e-16 ***
```

```
## D            0.11823    0.00568   20.82  < 2e-16 ***
## `x^1`        0.29691    0.01155   25.71  < 2e-16 ***
## `x^1_right`  0.04598    0.01350    3.41  0.00066 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.138 on 6554 degrees of freedom
## Multiple R-squared: 0.671,Adjusted R-squared: 0.671
## F-statistic: 4.45e+03 on 3 and 6554 DF,  p-value: <2e-16

reg_linear_2

## ### RDD regression: parametric ###
##  Polynomial order:  1
##  Slopes:  separate
##  Number of obs: 6558 (left: 2740, right: 3818)
##
##  Coefficient:
##   Estimate Std. Error t value Pr(>|t|)
## D  0.11823    0.00568    20.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

reg_linear_3

## ### RDD regression: parametric ###
##  Polynomial order:  3
##  Slopes:  separate
##  Number of obs: 6558 (left: 2740, right: 3818)
##
##  Coefficient:
##   Estimate Std. Error t value Pr(>|t|)
## D   0.1115     0.0107    10.5   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

reg_linear_4

## ### RDD regression: parametric ###
##  Polynomial order:  1
##  Slopes:  separate
##  Bandwidth:  0.4
##  Number of obs: 4169 (left: 2043, right: 2126)
##
##  Coefficient:
##   Estimate Std. Error t value Pr(>|t|)
```

```
## D  0.08863    0.00727    12.2   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, a `plot()` function adds the estimated curve to the binned plot. Here we show the difference between the model estimated with polynomial of order 1 and order 3:

```
par(mfrow = c(2, 1))
plot(reg_linear_1)

## Warning:  font width unknown for character 0x9

plot(reg_linear_3)

## Warning:  font width unknown for character 0x9
```




```
par(mfrow = c(1, 1))
```

4.2. Non-parametric

Although the parametric estimator is often used in practice, another estimator has important appeal, in this context where one is interested in estimating a regression just around a cutoff. In this case, non-parametric estimators such as the local-linear kernel regression of Fan and Gijbels (1992, 1996), which aim at estimating a regression locally at each point, have interesting features, as advocated by Porter (2003). A local linear regression amounts to do a simple weighted linear regression, where the weights are given by a kernel function. Formally, the

local-linear estimator (LLE) is given by its estimating equation:

$$\hat{\alpha}(c), \hat{\beta}(c), \hat{\tau}(c) = \underset{\alpha,\beta,\tau}{\arg\min} \sum_{i=1}^{n} (Y_i - \alpha - \tau D - \beta(X_i - c))^2 \mathcal{K}\left(\frac{X_i - c}{h}\right) \quad (4)$$

where $\mathcal{K}(\cdot)$ is a kernel function attributing weights to each point according to their distance to the point c. Note that the parameters α, β and τ are written as of function of c to emphasize the fact that these are *local* estimate, unlike in the parametric rate. The kernel used in RDDtools here is the triangular kernel (also called *edge* function sometimes): $K(x) = I(|x| \leq 1)(1 - |x|)$. This choice, which departs from the the suggestion of Lee and Lemieux (2010), is driven by the fact that the triangular kernel was shown to be optimal when one estimates a parameter at a boundary, which is precisely our case here (Cheng, Fan, and Marron 1997). Unlike the package **rdd**, we do not offer other kernels in **RDDtools**, since the kernel selected is optimal, and changing the kernel is found to have little impact compared to changing the bandwidths.

Note that using the LLE estimator reduces to do a weighted OLS (WOLS) at each point[1], which allows to use the usual regression function `lm()` in R, specifying the weights as given by the kernel. However, although this is a WOLS, the variance of the LLE is not the same as that of the WOLS, unless one is ready to assume that the bandwidth used is the true *bandwidth*[2]. However, most, if not all, papers in the literature do use the standard WOLS inference, eventually adjusted for heteroskedasticity. This is also done currently in the RDDtools package, although we intend to do this following the work of Calonico, Cattaneo, and Titiunik (2012).

Another question arises is the choice of the bandwidth, which is a crucial question since this choice has a huge impact on the estimation. Typically, decreasing the bandwidth will reduce the bias of the estimator, but increase its variance. One way of choosing the bandwidth is then to try to minimise the mean-squared error (MSE) of the estimator, which allows to trade-off bias and variance. This approach is pursued by Imbens and Kalyanaraman (2012), and is available in **RDDtools** with the function `RDDbw_IK()`. This function takes simply a RDDdata object as input, and returns the optimal value according to the MSE criterion.

As an illustration, we use now the non-parametric estimator for the Lee dataset, estimating first the bandwidth and then the discontinuity with `RDDreg_np()`:

```
bw_IK <- RDDbw_IK(Lee2008_rdd)
bw_IK

##  h_opt
## 0.2939

reg_nonpara <- RDDreg_np(RDDobject = Lee2008_rdd, bw = bw_IK)
```

The output, of class `RDDreg_np`, has the usual `print()`, `summary()` and `plot()` functions:

[1] See (Fan and Gijbels 1996, equ. 3.4, page 58).

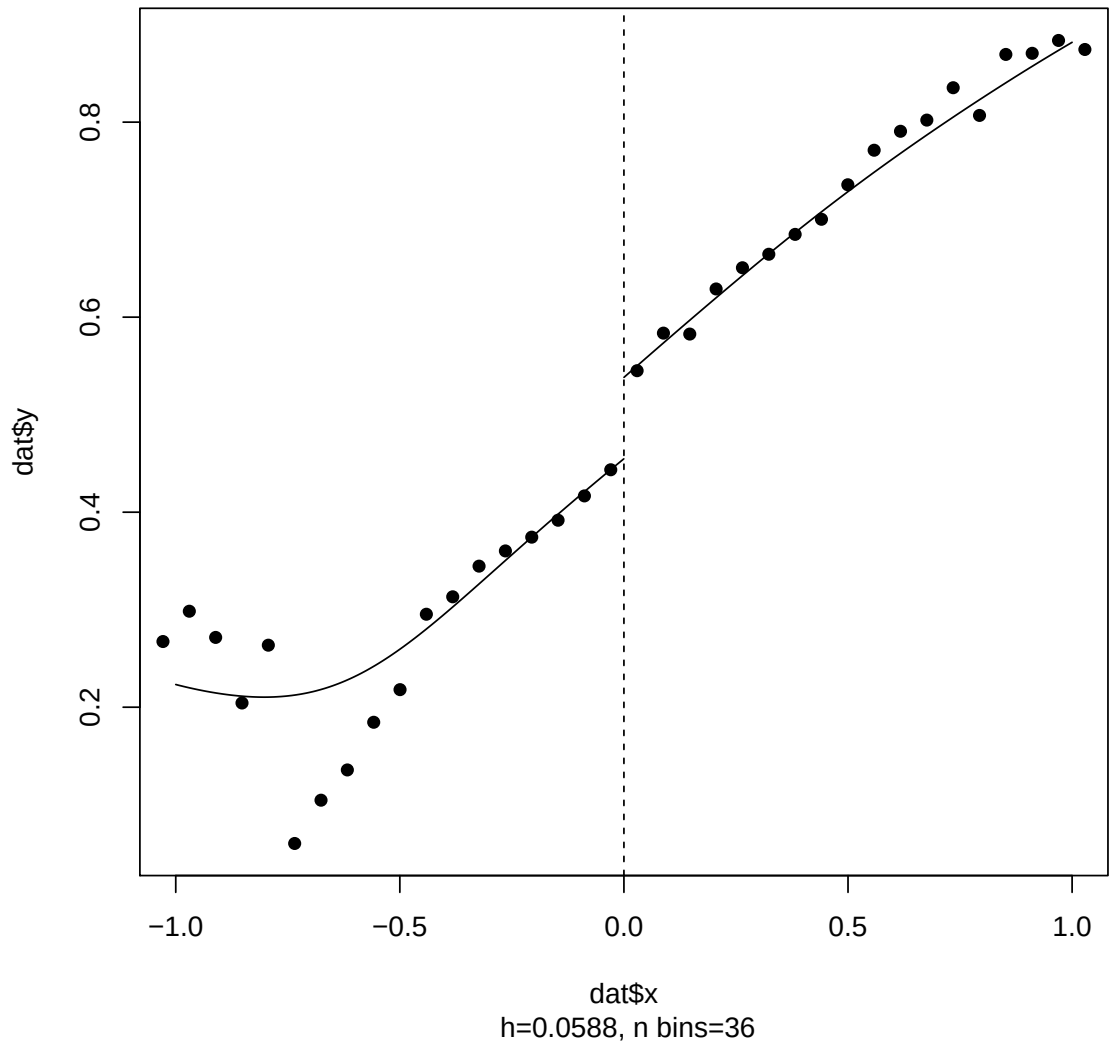
[2] A second option is use a smaller bandwidth, in which case standard inference can be applied. This has however the drawback of using a sub-optimal bandwidth, with a slower rate of convergence.

```
reg_nonpara

## ### RDD regression: nonparametric local linear###
##  Bandwidth:  0.2939
##  Number of obs: 3200 (left: 1594, right: 1606)
##
##  Coefficient:
##   Estimate Std. Error t value Pr(>|t|)
## D  0.07992    0.00682    11.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

summary(reg_nonpara)

##
## Call:
## lm(formula = y ~ ., data = dat_step1, weights = kernel_w)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5235 -0.0015  0.0000  0.0000  0.4605
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.45328    0.00491    92.3   <2e-16 ***
## x            0.39089    0.04204     9.3   <2e-16 ***
## D            0.07992    0.00682    11.7   <2e-16 ***
## x_right      0.05344    0.05916     0.9     0.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0816 on 3196 degrees of freedom
## Multiple R-squared: 0.356,Adjusted R-squared: 0.356
## F-statistic:  590 on 3 and 3196 DF,  p-value: <2e-16
```

The `plot()` function shows the point estimates[3] over a grid defined within the bandwidth range, i.e. the parameter $\alpha(x)$ from equation 4 such as $\alpha(x) \quad \forall \ [x - bw; x + bw]$. This should not be confused with the line drawn in the parametric plots, which show the curve $y = f(x) = \hat{\alpha} + \hat{\beta}(x - c) + \hat{\tau}D$.

[3]Note that the estimates are obtained with the `locpoly()` function from package **KernSmooth**. This has however the disadvantage that it is not the same kernel used as in the previously, since the locpoly function uses a gaussian kernel, while we use a triangular one. Since this is only for visual purpose, the difference should however not be perceptible. Furthermore, using the `locpoly()` function has the advantage that the algorithm is way faster, since the authors did implement a fast binned implementation, see Fan and Gijbels (1996, section 3.6).

```
plot(reg_nonpara)

## Warning:  font width unknown for character 0x9
```



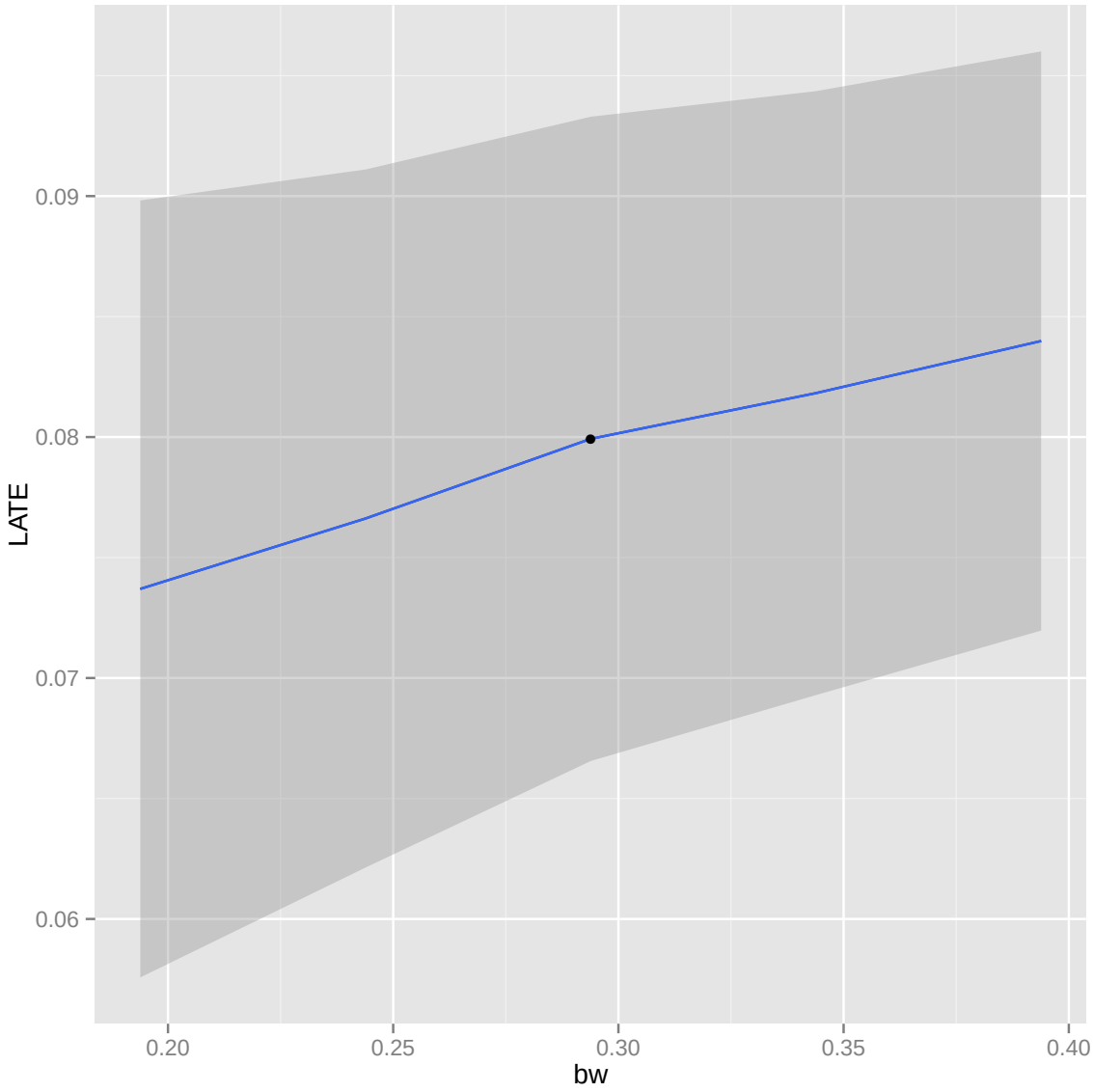
4.3. Assessing the sensibility of the estimator

Both the parametric and non-parametric estimators are dependent on the choice of extra-parameters such as the polynomial order, or the bandwidth. It is however known that this choice can have a big impact, especially in the case of the bandwidth choice for the non-parametric case. A simple way to assess the sensitivity of the results is to plot the value of the estimate against multiple bandwidths. This is the purpose of the function `plotSensi()`, which work both on `RDDreg_lm()` as well as `RDDreg_np()`. In the former case, the function will assess the sensitivity against the polynomial order (eventually the bandwidth if it was

specified), while in the latter case against the bandwidth.

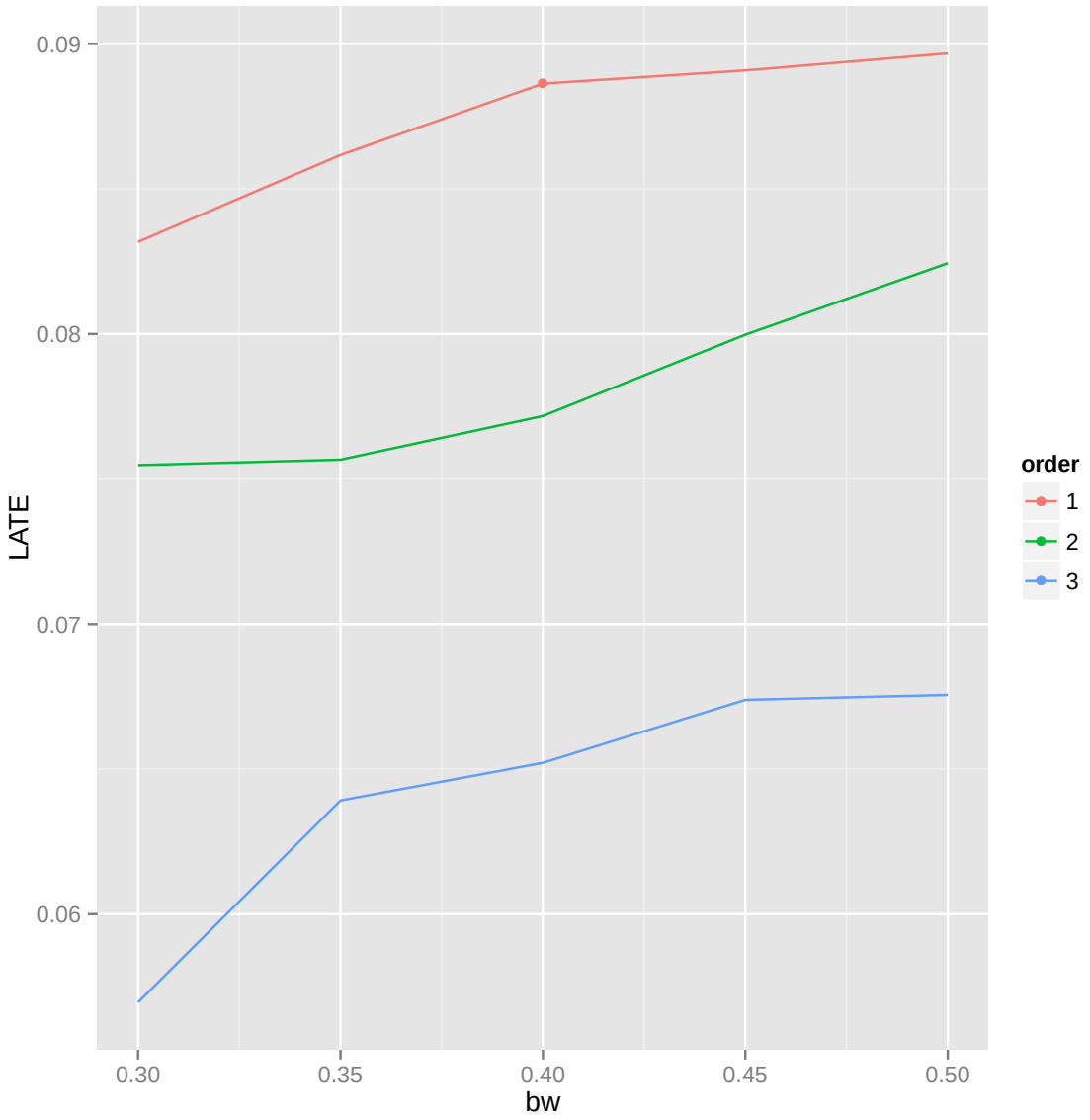
We illustrate this on the previous non-parametric estimator:

```
plotSensi(reg_nonpara, device = "ggplot")
```



and we illustrate it also on the parametric estimator where a bandwidth was specified:

```
plotSensi(reg_linear_4, device = "ggplot")
```



5. Step 3: Validity tests

Once the discontinuity estimated and its sensitivity against the bandwidth choice assessed, the last step in the analysis is to proceed to a few validity tests.

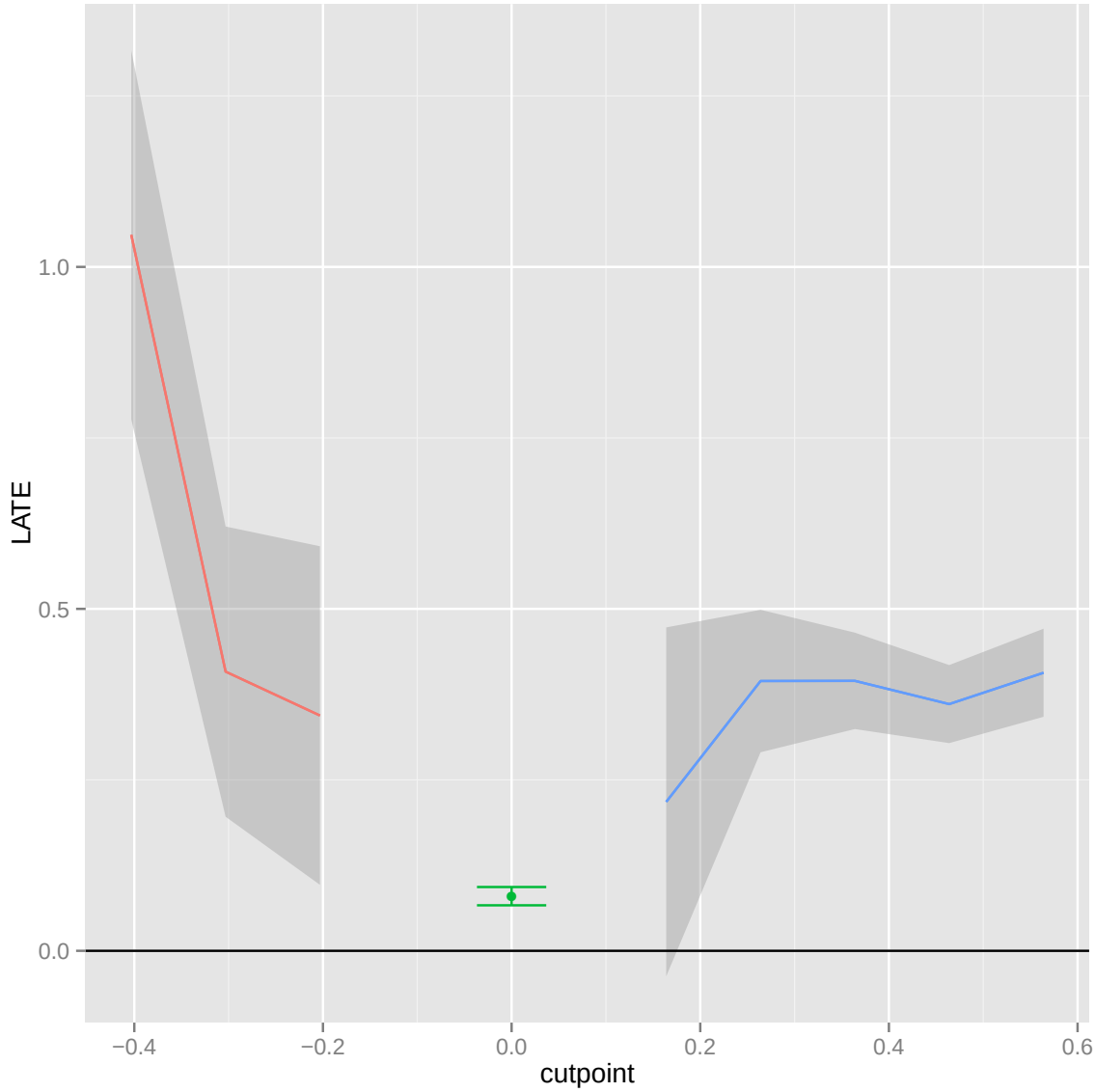
5.1. Placebo tests

A way to convince its readers that the discontinuity one has found is a true one is to show that it is not the a spurious result one could have obtained at a random cutoff. Hence, as advocated by Imbens and Lemieux (2008), one can run placebo tests, where one estimates a discontinuity but at a different point than the true cutoff. This is available through the `plotPlacebo()` function, which works on `RDDreg_lm` or `RDDreg_np` objects. An important question is on which point this should be tested. The fact is that the sample should not

contain the cutoff point (so that the presence of a discontinuity at its point does not impact the estimates at other points), and be far away from that cutoff (as well as from the min and max of the whole distribution) so that it contains a fair amount of points at both sides for estimation. The default is then to run for points on the left within the first and last quartiles of the left sample, and the same on the right.

We illustrate this on the non-parametric estimator:

```
plotPlacebo(reg_nonpara, device = "ggplot")
```



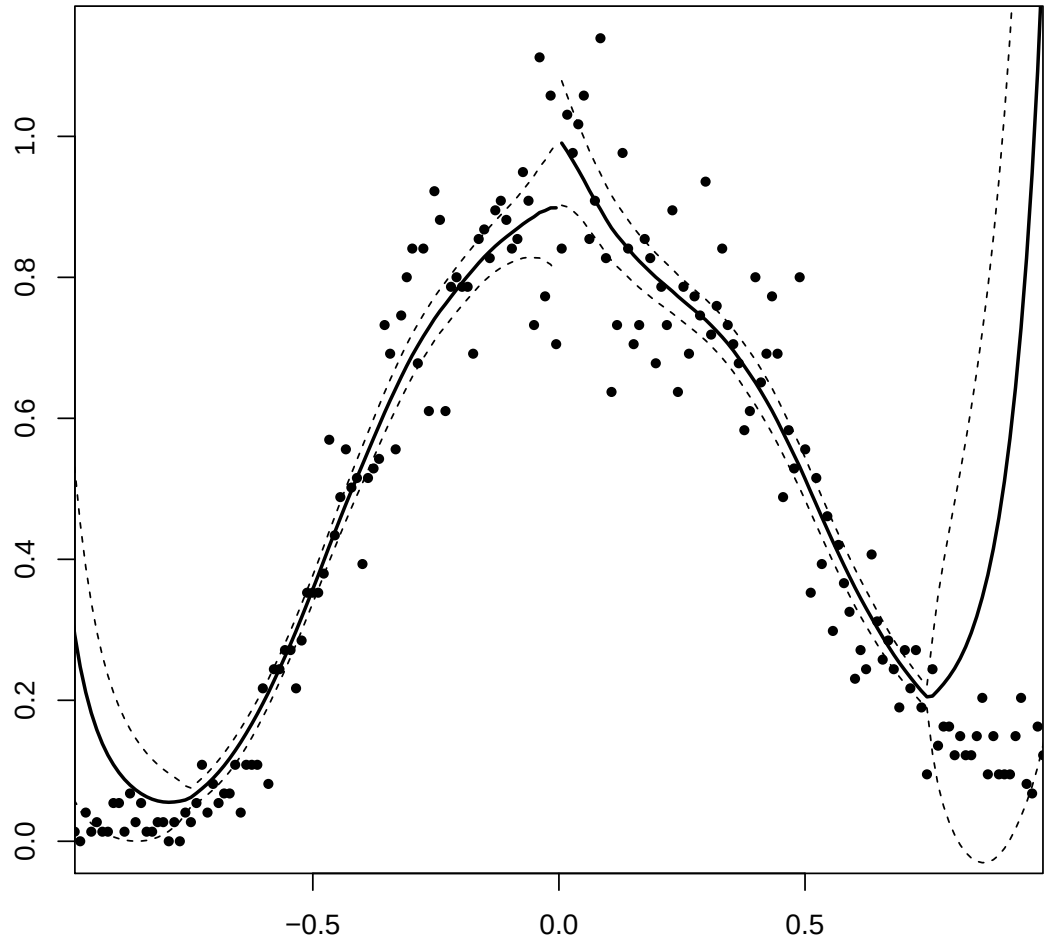
5.2. Forcing variable

One of the cases where the assumptions underlying the RDD analysis might be incorrect is when participants are allowed to manipulate the variable that lead to treatment, i.e. are able

to affect whether they are treated or not. This question is usually answered factually, looking at the context of the experiment. One can however also test whether the forcing variable itself shows a trace of manipulation, which would result into a discontinuity of its density, as suggested by McCrary (2008).

The test was implemented by D Dimmery in package **rdd**, and is simply wrapped by the function dens_test(), so that it works directly on a RDDdata object:

```
dens_test(Lee2008_rdd)
```




```
## [1] 0.1952
```

The test automatically returns a plot, showing the density estimates at the left and right of the cutoff, together with the confidence intervals of these estimates. One rejects the null hypothesis of no discontinuity if visually the confidence intervals do not overlap.

5.3. Baseline Covariates

Another crucial assumption in RDD is that treatment is randomly distributed around the cutoff, so that individuals around are similar. This can be easily tested, as is done in the Randomised Control Trial (RCT) case, by running test for balanced covariates. Two kinds of tests have been implemented, allowing to test equality in means (t-test) or in distribution (Kolmogorov-Smirnov). As this is a typical case of multiple testing, both functions offers the possibility to adjust the p-values with various procedures such as the Bonferoni, Holmes or the more recent Benjamini-Hochberg procedures.

We run here the equality in means test:

```
covarTest_mean(Lee2008_rdd)

##    mean of x mean of y Difference statistic p.value
## z1 0.005348  0.0358    0.03045    -1.226    0.2202
## z2 20.05     20        -0.04979   0.9974    0.3186
## z3 2.003     1.997     -0.005801  0.2826    0.7775
```

as well as the equality in distribution test:

```
covarTest_dis(Lee2008_rdd)

## Warning:  p-values will be approximate in the presence of ties

##    statistic p.value
## z1 0.01921   0.5985
## z2 0.02843   0.1518
## z3 0.0153    0.849
```

Since the covariates were generated randomly with a single parameter, we would expect that no equality test is rejected.

6. Conclusion

References

Calonico S, Cattaneo MD, Titiunik R (2012). "Robust Nonparametric Bias-Corrected Inference in the Regression Discontinuity Design." *Technical report*.

Cheng MY, Fan J, Marron JS (1997). "On Automatic Boundary Corrections." *Annals of Statistics*, **25**, 1691–1708.

Fan J, Gijbels I (1992). "Variable Bandwidth and Local Linear Regression Smoothers." *Annals of Statistics*, **20**, 2008–2036.

Fan J, Gijbels I (1996). *Local Polynomial Modeling and its Implications*. Boca Raton: Chapman and Hall/CRC, Monographs on Statistics and Applied Probability no. 66.

Imbens G, Kalyanaraman K (2012). "Optimal Bandwidth Choice for the Regression Discontinuity Estimator." *Review of Economic Studies*, **79**, 933–959.

Imbens GW, Lemieux T (2008). "Regression discontinuity designs: A guide to practice." *Journal of Econometrics*, **142**(2), 615–635. URL http://ideas.repec.org/a/eee/econom/v142y2008i2p615-635.html.

Lee DS (2008). "Randomized experiments from non-random selection in U.S. House elections." *Journal of Econometrics*, **142**, 675–697.

Lee DS, Lemieux T (2010). "Regression Discontinuity Designs in Economics." *Journal of Economic Literature*, **48(2)**, 281–355.

McCrary J (2008). "Manipulation of the Running Variable in the Regression Discontinuity Design: A Density Test." *Journal of Econometrics*, **142**, 698–714.

Porter J (2003). "Estimation in the Regression Discontinuity Model." *Technical report*, University of Wisconsin, Madison, Department of Economics.

Ruppert D, Sheather SJ, Wand MP (1995). "An effective bandwidth selector for local least squares regression." *Journal of the American Statistical Association*, **90**, 1257–1270.

Affiliation:

Matthieu Stigler
Centre for Finance and development
IHEID
Geneva
E-mail: Matthieu.Stigler@iheid.ch