Computaton Appendix for "Uniform Inference on Quantile Effects under Sharp Regression Discontinuity Designs"

Zhongjun Qu and Jungmo Yoon

November 7, 2017

This appendix is structured as follows. Section 1 explains how to use the main functions to select the bandwidth, estimate the QTE, test hypotheses, and construct uniform confidence bands. Section 2 outlines the structure of the replication file for the simulation results reported in the paper. Section 3 explains the empirical replication. Users can replace the current data file with theirs and conduct analysis parallel to that in the paper.

The R file '`qte_rdd.R`' contains five functions for analyzing quantile treatment effects under sharp RD designs; '`qte_rdd_funcs.R`' contains some supporting functions. These functions are used in '`qte_rdd_sim.R`' to reproduce the simulation results and in '`qte_rdd_app.R`' for the empirical application.

This version: November 7, 2017. The code will be continually updated; see the authors' website for the most recent version.

1 Five R functions

This section explains how to use the following five functions.

i. `rdd.bandwidth()`-- implements the five bandwidth selectors.

ii. `rdd.rqpro()`-- estimates conditional quantiles.

iii. `rdd.qte()`-- estimates the QTE and obtains the uniform and pointwise confidence bands.

iv. `Score()`-- performs the Score test.

v. `Wald()`-- performs the (robust) Wald tests.

Henceforth, suppose that the outcomes, covariate values, and treatment indicators are saved in objects `y`, `x`, and `d`. Put all R scripts in one directory. The first thing to do is to read the functions. In `R`, type:

```
> source("qte_rdd.R")
```

Bandwidth: `rdd.bandwidth()`. To compute the bandwidth at the median, use:

```
> rdd.bandwidth(x,y,d,x.eval=0,tt=c(0.2,0.8),m=9,method=c(1,2,3,4,5),val=(2:10/20),
kr=3).
```

The first three entries `x,y` and `d` specify the variables to use. `x.eval` is the cutoff point, which is set to be 0 in this example. `tt` defines the quantile range $\mathcal{T}$, which is set to $[0.2, 0.8]$. The next argument `m` is the number of quantiles to be estimated. In this example, we set `m`=9. One can get a finer approximation to the conditional quantile process by letting `m=19 or 29`, which can be done without restrictions.

The argument `method` specifies the bandwidth selectors. If `method=1`, users will get $h^{cv}_{n,0.5}$. Likewise, setting `method=2,3,4 or 5` will lead to $h^{cvi}_{n,0.5}$, $h^{int}_{n,0.5}$, $h^{bdy}_{n,0.5}$, or $h^{ik}_{n,0.5}$ respectively. One can ask for multiple bandwidths. For example, setting `method=c(1,4,5)` (or `method=c(2,3)`) will produce all boundary (or interior) point bandwidths. One can also get all five bandwidths, as the above command line shows. The argument `val` specifies the candidate values for the cross validation bandwidth. It is necessary to specify `val` regardless of the choice of `method`. It is because the MSE optimal bandwidths ($h^{int}_{n,0.5}$, $h^{bdy}_{n,0.5}$ and $h^{ik}_{n,0.5}$) require estimates of conditional densities and $h^{cv}_{n,0.5}$ is used as a pilot bandwidth to do so. The last argument `kr` specifies which kernel function to use. By letting `kr=3`, we use the Epanechnikov kernel, but other options are available. See comments in the function `rdd.bandwidth()`.

There are two additional arguments that the user can specify: `band` and `br`. The option `band` specifies the bandwidth values to estimate the second order derivatives at the median. (The values are needed for computing the MSE-optimal bandwidths.) If the user does not specify `band`, then the following default values will be used: one half of the length of the support of `x`. So, for example, for $h^{bdy}_{n,0.5}$ the default bandwidth values are $0.5 * (\max(x) - x_0)$ and $0.5 * (x_0 - \min(x))$. If the user chooses to specify `band`, then the values need to be entered in the following order. The first element `band[1]` is for $h^{int}_{n,0.5}$, the next two elements `band[2:3]` are for $h^{bdy}_{n,0.5}$, and finally `band[4:5]` are for $h^{ik}_{n,0.5}$. The last two cases each contains two values, for $Q''(\tau|x_0^+)$ and $Q''(\tau|x_0^-)$ respectively. The option `br` is needed only for $h^{int}_{n,0.5}$. If `br=1`, then when estimating the MSE-optimal bandwidth with a local cubic regression, the intercepts on two sides of the cutoff are allowed to be different. If `br=0`, then the intercepts are restricted to be equal. The default is `br=0`.

For example, a command that includes all the options mentioned above will look like

```
> rdd.bandwidth(x,y,d,x.eval=0,tt=c(0.2,0.8),m=9,method=c(1,2,3,4,5),val=(2:10/20),
kr=3,band=c(1,0.5,0.5,0.5,0.5),br=1).
```

If the user saves the outcome of `rdd.bandwidth()` in an object `H`, the median bandwidth of the selected methods can be found in `H$hcv`, `H$hcvi`, `H$hint`, `H$hbdy`, or `H$hik`.

Estimating conditional quantiles: `rdd.rqpro()`. Once selected a bandwidth, the user can proceed to estimate $Q(\tau|x_0)$, $Q(\tau|x_0^+)$, or $Q(\tau|x_0^-)$. To estimate $Q(\tau|x_0)$, run:

```
> rdd.rqpro(x,y,tt=c(0.2,0.8),m=9,x.eval=0,bandw=0.3,method=1,kr=3)
```

To estimate $Q(\tau|x_0^+)$, run:

```
> rdd.rqpro(x[d==1],y[d==1],tt=c(0.2,0.8),m=9,x.eval=0,bandw=0.3,method=1,kr=3)
```

Most options have the same meanings as before. We explain the new ones. `bandw` indicates what bandwidth to use for estimation. Just the median bandwidth would suffice. If one sets `bandw=0.3` as above, the function will use 0.3 as the value of the bandwidth. `method` indicates which estimation method to use: `method=1` implements the First Procedure in Section 3 of the paper, `method=2` uses the Second Procedure, and `method=3` uses a naive quantile-by-quantile estimation procedure without enforcing monotonicity.

If the user saves the output of `rdd.rqpro()` in an object `A`, the quantile index used in estimation can be found in `A$taus` and the corresponding conditional quantile estimates can be found in `A$Q0`.

Estimating QTE: `rdd.qte()`. The QTE, $\delta(\tau)$, and its uniform and pointwise confidence bands can be estimated by

```
> rdd.qte(x,y,d,x.eval=0,alpha=0.9,tt=c(0.2,0.8),m=9,bandw=0.3,kr=3,bias=0,eql=0)
```

When `alpha=0.9`, one will get the 90% confidence band. This function has two new options, `bias` and `eql`. The former determines whether to use the robust confidence bands, and the latter indicates whether to use the equality constraint for the bias. So, for example, if `bias=1,eql=0`, one will get the robust uniform and pointwise bands using quantile-by-quantile bias estimation. If `bias=0,eql=0`, one will get the uniform and pointwise bands without bias correction.

If a user saves the outcome of the function in an object `B`, the QTE estimate, $\hat{\delta}(\tau)$, can be found in `B$qte`, and the uniform and pointwise bands can be found in `B$uci` and `B$pci`.

The score test: `Score()`. To implement the Score test for treatment significance, run:

```
> Score(x,y,x.eval=0,alpha=c(0.9,0.95),tt=c(0.2,0.8),m=9,bandw=0.3,kr=3)
```

The option `alpha` sets the desired confidence level $1-\alpha$. So, when `alpha=c(0.9,0.95)`, one will get critical values at the 10% and 5% levels. The other options are the same as before. If a user saves its outcome in an object `S`, the Score test statistic and critical values can be found in `S$test` and `S$crit`.

The Wald tests: `Wald()`. To implement the Wald tests, run:

```
> Wald(x,y,d,x.eval=0,alpha=c(0.9,0.95),tt=c(0.2,0.8),m=9,bandw=0.3,bandw2=0.3,
      kr=3,test.type=c(1,2,3),sign.opt=1,eql=0)
```

This function produces the following output: (i) the conventional Wald test without bias correction, (ii) the robust Wald test with bias correction.

If `eql=1`, one can get the robust Wald test with the equality constraint on the biases. The argument `bandw` specifies the value of $h_{n,0.5}$ and `bandw2` specifies the value of $b_{n,0.5}$. The option `test.type` determines which hypothesis to test. The values `1`,`2` and `3` indicate the treatment significance, homogeneity and unambiguity hypothesis, respectively. One can test a single hypothesis by setting `test.type=1` or test all three hypotheses by `test.type=c(1,2,3)`. The option `sign.opt` sets the sign of the treatment unambiguity hypothesis. If `sign.opt=1`, the effects are unambiguously positive under the null hypothesis, and if `sign.opt=2`, the effects are unambiguously negative under the null hypothesis.

If a user saves the outcome of the function in an object `W`, the test statistic and critical values of the conventional Wald test can be found in `W$wald.test` and `W$wald.crit`. The results for the robust Wald test can be found in `W$wald.robust.test` and `W$wald.robust.crit`.

2 Replicating the simulation results

The replication script is '`qte_rdd_sim.R`'. Running it over different models, sample sizes, bandwidth choices replicates results in the simulation section.

The script contains fairly detailed comments to provide step by step instructions.

3 Replicating empirical findings

The five functions we have discussed so far are optimized for small to medium sample sizes. From our experience, they work efficiently when the number of observations is 10,000 or less. In our application, the sample size, 457,615, turns out to be too big for some of these functions. Specifically, the four functions, `rdd.bandwidth()`, `rdd.qte()`, `Score()`, and `Wald()`, have routines that heavily rely on matrix operations. However, as the size of a matrix grows, these operations may slow down the computation or even stop it. To address this, we have developed alternative versions of the functions that are more suitable for larger sample sizes. For example, the following results hold regarding the distributions of the Wald tests (using notations in Section 5): $(nh_{n,\tau})^{-1}\sum_{i=1}^{n} d_i K_{i,\tau} z_{i,\tau} z_{i,\tau}' \to^p f_X(x_0) N^+(\tau)$ and $(nb_{n,\tau})^{-1}\sum_{i=1}^{n} d_i K_{i,\tau} \bar{z}_{i,\tau} \bar{z}_{i,\tau}' \to^p f_X(x_0)\bar{N}^+(\tau)$. The original functions calculate the left hand side expression by matrix multiplication, while the alternative functions use the right hand side expressions directly. In doing so, they use the fact that $N^+(\tau)$ or $\bar{N}^+(\tau)$ can be explicitly determined once the kernel function is fixed and the marginal density of X can be easily estimated.

These four new functions have an extension `.app` at the end of their names: `rdd.bandwidth.app()`, `rdd.qte.app()`, `Score.app()`, and `Wald.app()`. The new functions have the same arguments as the original ones, so can be used in the same way as before. But be aware that when applied to a small to medium sample sizes, they can be slower than the functions without `.app()`. This can make a significant difference if one uses the codes for simulation studies as in Section 8 of the current paper.

The replication script is '`qte_rdd_app.R`'. The script contains fairly detailed comments to provide step by step instructions.