Replication Files for "Inference on Conditional Quantile Processes in Partially Linear Models with Applications to the Impact of Unemployment Benefits"

Zhongjun Qu — Boston University
Jungmo Yoon — Hanyang University
Pierre Perron — Boston University

December 17, 2021

The enclosed R files replicate the results in Qu, Yoon, and Perron (2021), "Inference on Conditional Quantile Processes in Partially Linear Models with Applications to the Impact of Unemployment Benefits." This note explains how to use them.

The R files are organized by folders. For example, the folder 'Table 1' contains scripts that reproduce the results in Table 1 of the paper. Similarly, the folder 'Simulation' contains scripts that reproduce the simulation results. The scripts in different folders run independently. The directory 'General' contains R functions that support the replication scripts in these folders. It must be added to the R path for other scripts to work.

Our empirical application uses Nekoei and Weber's (2017) data. Because the original data - administrative social security record data from the Austrian Social Security Database - are proprietary, they can not be posted on a public website. Therefore, besides providing a program that reproduces the empirical results provided there is access to the complete data, we also provide a shorter program that reproduces a subset of the results using an excerpt of the data that Nekoei and Weber made available on the AER website, at http://doi.org/10.3886/E113058V1. Aside from this data access issue, all results reported in the main paper and the online appendix can be replicated using the programs in this folder.

1 Overview

The replication package includes the following R files.

(1) `run_simulation.R` (in Simulation folder): replicates the simulation results.

(2) `run_application.R` and `run_application_sub.R` (in Empirical Application folder): replications empirical results.

(3) `qte_rdd_cov.R` (in General folder): contains functions to do (1) and (2).

(4) `qte_rdd_cov_fn.R` (in General folder): contains supporting functions for (3).

(5) `run_table1.R`: (in Table 1 folder): replicates Table 1.

(6) `qrp_rmpi.R` (in Parallel folder): implements parallel computation. It is used to calculate the computing time for the global partially linear model.

(7) `run_rmpi.txt` (in Parallel folder): contains batch commands to run (6).

(1) replicates simulation results for Models 1–3 in the paper. In each cases, it produces conditional quantile estimate, its MSE and Bias, the bias estimate, several uniform confidence bands, and bandwidths for conditional quantile estimates. It requires R functions in (3) and (4). So put (1), (3) and (4) in the same directory and run (1). The script includes numerous comments to assist users along the way.

(2) replicates the empirical results. `run_application.R` assumes that one has access to Nekoei and Weber's (2017) sample. The dataset is large. So choose one model (out of nine) and one dependent variable (out of three) at a time. The script includes many comments to guide users. `run_application_sub.R` uses an excerpt of the data that Nekoei and Weber made available on the AER website (Data_public.dta). To run this program, the user should download the data and place it in the same folder. The output is Figure 1 in the paper. These programs require functions in (3) and (4).

(3) includes three main R functions: `nrq()`, `nrq.band()`, and `nrq.bandwidth()`. While. `nrq()` estimates a conditional quantile process, `nrq.band()` obtains uniform confidence bands for it. And `nrq.bandwidth()` selects bandwidths for estimation and inference. All three functions have an option, `rdd`. When it is set to 1, they can be used to obtain a QTE, its uniform bands, and bandwidths. Section 2 in this note provides more details.

(3) also includes three functions: `nrq.het()`, `nrq.band.het()`, and `nrq.bandwidth.het()`. They are designed specifically for RDD with covariates. They produce heterogeneous QTEs for groups defined by covariates.

(4) includes several supporting functions used within the main functions in (3). It also includes functions needed for simulation exercises.

(5) runs the simulation reported in Table 1.

(6) is for parallel computation. Install a R package `Rmpi` in advance. When the sample size is big (e.g., bigger than 50,000), the global partially linear model can take time. This script shows how to use parallel computation to carry out such a computing intensive job.

(7) This is a batch file to run (6) in a cluster. Open a terminal and type

`> qsub run_rmpi.txt`.

A user needs to specify the number of nodes (computers) and the number of cores (processors) within a node. Contact your system administrator and find out resources available to you. In the batch file, we used 20 nodes and 28 cores per node.

2 Details on R functions in `qte_rdd_cov.R`.

i. `nrq()`: estimates a conditional quantile process and QTE.

ii. `nrq.band()`: obtains two kinds of uniform confidence bands. The first is based on the asymptotic linear representation of the estimator and the second is based on resampling. Each comes with two types bands; a conventional uniform band and a robust uniform band that takes into account additional uncertainty from bias term estimation.

iii. `nrq.bandwidth()`: implements two bandwidth selection methods; the cross-validation bandwidth and the MSE optimal bandwidth.

Save the outcome variable Y in an object `y` and independent variables (X, Z) in `x`. Let `dx` and `dz` denote the dimensions of X and Z.[1] The first `dx` columns in `x` contain X and the rest of columns in `x` contain Z. Let `x0` and `z0` denote the evaluation point of a conditional quantile process $Q(\tau|x, z)$. The quantile indexes to estimate, $\mathcal{T}$, is denoted by `tt`. For

[1] The paper uses notations d and p for the dimensions.

example, when $\mathcal{T} = [0.1, 0.9]$, one may set `tt = seq(0.1,0.9,by=0.05)`. It will generate an evenly spaced point grid with increment 0.05.

`Conditional quantiles and QTE`

The function `nrq()` can implement both a global partially linear (GPL) model and a local partially linear (LPL) model. The argument `step1` determines which model is to be estimated. The following command shows how to estimate a GPL model. The main output is the conditional quantile process estimate.

```
> nrq(y,x,dx,dz,x0,z0,tau=tt,h.med,h.med2,rdd=0,step1=1,se=0)
```

To estimate a LPL model, type

```
> nrq(y,x,dx,dz,x0,z0,tau=tt,h.med,h.med2,rdd=0,step1=0,se=0)
```

Additional arguments have the following meanings. `h.med` and `h.med2` are the two bandwidth values at the median, denoted by h_n and $b_{n,\tau}$ in the paper. The former is used in Step 1 of the GPL model. The latter is the main bandwidth for conditional quantiles and uniform bands. When `rdd=0`, the function estimates conditional quantiles at `x0` and `z0`. If a user is working on a RDD setup, set `rdd=1`. Then it will split the sample into two parts using `x0` as the cutoff value and obtains the (conditional) QTE at `z0`. For RDD, the independent variable X is the running variable, and currently it should be one dimensional. In other words, the option `rdd=1` works only when `dx=1`. If `se=1`, pointwise standard errors will be calculated. It is used only for the resampling based uniform band. In most cases, `se=0` is appropriate.

If a user saves outputs of `nrq()` in an object `A`, the conditional quantile estimate can be found in `A$Q.est`, and the bias corrected conditional quantile estimate is `A$Q.est.cor`. The function also produces other useful statistics. The estimate for the nonparametric part of the model $\hat{g}(x, \tau)$ can be found in `A$g.est`, the estimate for the linear part of the model $\hat{\beta}(\tau)$ is in `A$beta.est`, and the bias term estimate can be found in `A$bias.est`. When `rdd=1`, `A$Q.est` is the QTE estimate and `A$Q.est.cor` is the bias corrected QTE estimate.

`Uniform confidence bands`

For inference, one needs a confidence band and it is provided by `nrq.band()`. The following command shows how to obtain a uniform confidence band for a LPL model.

```
> nrq.band(y,x,dx,dz,x0,z0,tau=tt,h.med,h.med2,alpha,n.sim=1000,rdd=0,step1=0,res=1)
```

For a GPL model, set `step1=1`,

```
> nrq.band(y,x,dx,dz,x0,z0,tau=tt,h.med,h.med2,alpha,n.sim=1000,rdd=0,step1=1,res=1)
```

Most arguments have the same meanings. We explain the new ones. When `alpha=0.9`, it obtains a 90% uniform confidence band. `alpha` can take multiple values. When `alpha=c(0.9,0.95)`, two confidence bands with coverage probabilities 90% and 95% will be calculated. Uniform bands are calculated by simulation-based methods. The argument `n.sim` sets the number of simulation draws. From our experience, `n.sim=1000` will be reasonable in most cases.

The paper proposes two methods to obtain a uniform band. The first method is based on the asymptotic approximation and the second one is based on resampling. The default method in `nrq.band()` is the first one. If users are interested in the second uniform band, set `res=1`. This will produce both bands. When `res=0`, the function only produces the first band. The first uniform band is simple and fast but requires nuisance parameters estimation. The second uniform band does not depend on nuisance parameters such as conditional density function, but it is computationally intensive. When the sample size is big, setting `res=0` can save time.

If a user saves outputs of `nrq.band()` in an object `B`, the conventional and robust uniform bands based on the asymptotic approximation can be found in `B$uband` and `B$uband.robust`, respectively. If `res=1`, the conventional and robust uniform bands based on resampling can be found in `B$uband.R` and `B$uband.robust.R`, respectively. The function also reports conditional quantile estimates for the convenience of users. `B$Q.est` is the conditional quantiles estimate and `B$Q.est.cor` is the bias corrected conditional quantiles estimate. When `rdd=1`, outcomes such as `B$uband` and `B$uband.robust` refer to uniform bands for the QTE.

`Bandwidths`

To implement the bandwidth selector, type

```
> nrq.bandwidth(y,x,dx,dz,x0,z0,rdd=0,step1=1,bdy=1,order=2,cv=1,hp)
```

`nrq.bandwidth()` can provide two types of bandwidth: the cross-validation (CV) bandwidth and the (MSE) optimal bandwidth. When `cv=1`, the function produce both. When `cv=0`, the function only yield the optimal bandwidth. The optimal bandwidth requires a pilot bandwidth in order to estimate nuisance parameters in the asymptotic MSE expres-

sion. When `cv=1`, the CV bandwidth is used for the pilot bandwidth. When `cv=0`, the value provided by the argument `hp` will be used for the pilot bandwidth. So `hp` can be ignored if `cv=1`.

The remaining arguments have the following meanings. The option `order` determines how the CV bandwidth is calculated. When `order=1`, a local linear regression is used, when `order=2`, a local quadratic regression is used. The option `bdy` concerns the optimal bandwidth. When `bdy=1`, it uses a boundary value formula, and if `bdy=0`, it uses an interior value formula.

If a user saves outputs of `nrq.bandwidth()` in an object `C`, the outcomes `C$h.cv` and `C$h.opt` are the CV and optimal bandwidth, respectively. Under the RDD (when the option `rdd=1`), bandwidths for each side of the cutoff are calculated separately.

3 Functions for RDD with covariates

The three functions below (appearing in `qte_rdd_cov.R`) are specifically designed for RDD with covariates. The goal is to detect and estimate heterogeneity in QTEs across groups defined by covariates.

i. `nrq.het()`: estimates QTE for each group.

ii. `nrq.band.het()`: obtains uniform confidence bands for each group.

iii. `nrq.bandwidth.het()`: calculate bandwidths for each group.

The new functions closely follow the previous ones `nrq()`, `nrq.band()` and `nrq.bandwidth()`. But they are specialized to RDD with potentially a big sample. So they only implements a local (LPL) model and focuses on QTE. They provide outputs for each group defined by covariates.

`QTE`

The following command shows how to estimate QTE.

```
> nrq.het(y,x,d,dx,dz,x0,z0,tau,h.med,opt=1,case=1)
```

Here the argument `d` contains treatment variable d_i, and `x0` is the cutoff point. The function `nrq.het()` only accepts one dimensional running variable, so `dx` should be 1. The

running variable enters into the first column of `x`. The rest of columns in `x` contain covariates Z. When Z is a dummy covariate, set `case=1`, and when Z is a continuous covariate, set `case=2`. The argument `z0` determines groups whose QTEs are to be estimated.

For illustration, when the goal is to estimate QTEs by occupation, the covariate Z will be a dummy variable taking 1 for white-collar workers and 0 for blue-collar workers. Let X be the running variable. The argument `x` includes (X, Z), `dx` and `dz` should be 1, and `case=1`. Set `z0 = c(0,1)`, then a user will get outcomes for blue-collar workers and white-collar workers, respectively. Here the order of outcomes corresponds to the value in `z0`.

As a second example, suppose that the goal is to estimate QTEs for groups by the previous wage (before job separation) which is a continuous variable. Let Z be the previous wage arranged in percentile levels. The argument `x` includes (X, Z), `dx` and `dz` should be 1, and `case=2`. If a user is interested in QTEs for each quartile, set `z0 = c(0.25,0.5,0.75)`. Then outcomes for groups at three quartiles of previous wage level will be reported. The script 'run_application.R' has more examples and helpful comments.

Other arguments can be used as follows. `h.med` is the bandwidth at the median. Because the function only deals with a LPL model, one bandwidth is sufficient. When `opt=1`, it uses quantile specific bandwidth values, and when `opt=0`, it uses the same bandwidth across all quantile levels.

If a user saves outputs of `nrq.het()` in an object `D`, the QTE estimate can be found in `D$qte`. The conditional quantiles from the right side of x_0 can be found in `D$Qp.est`, while those from the left side of x_0 can be found in `D$Qm.est`.

`Uniform confidence bands for QTE`

To get uniform confidence bands for QTE, type

```
> nrq.band.het(y,x,d,dx,dz,x0,z0,tau,h.med,alpha=0.9,opt=1,case=1)
```

The function `nrq.band.het()` only implements the asymptotic approximation based band because it is suitable even for a big sample. All arguments have the same meanings as before.

If a user saves outputs of `nrq.band.het()` in an object `E`, the QTE estimate can be found in `E$qte`, the bias-corrected QTE estimate can be found in `E$qte.cor`. The conventional uniform band is in `E$uband`, while the robust uniform band is in `E$uband.robust`.

`Bandwidth`

The bandwidth can be obtained as follows.

```
> nrq.bandwidth.het(y,x,d,dx,dz,x0,z0,cv,pm.each,val,p.order,bdy,case=1)
```

When the argument `cv=1`, the function produces both CV and optimal bandwidths. When `cv=0`, only the optimal bandwidth will be provided. When `pm.each=1`, it calculates the CV bandwidth on each side of the cutoff. So two CV bandwidth values (from the right and left side of the cutoff) will be reported. In other words, it treats the cutoff as a boundary point. If `pm.each=0`, it treats the cutoff as an interior point and reports one bandwidth value. The CV bandwidth requires a list of candidate values to try. This candidate values enters into the argument `val`. When `p.order=1`, the CV bandwidth uses a local linear regression, when `p.order=2`, it uses a local quadratic regression. And `bdy=1` means the boundary value formula is used, and `bdy=0` means the interior value formula is used. The argument `case` can be used as before. Set `case=1` for a binary covariate, and `case=2` for a continuous covariate. This distinction is needed for the conditional density estimation.

If a user saves outputs of `nrq.bandwidth.het()` in an object `F`, all bandwidth estimates are available in `F$bandwidth`.