cemmap
centre for microdata methods and practice

Implementing intersection bounds in Stata

Victor Chernozhukov
Wooyoung Kim
Sokbae Lee
Adam Rosen

The Institute for Fiscal Studies
Department of Economics, UCL

cemmap working paper CWP38/13

E·S·R·C ECONOMIC & SOCIAL RESEARCH COUNCIL
An ESRC Research Centre

IMPLEMENTING INTERSECTION BOUNDS IN STATA

VICTOR CHERNOZHUKOV, WOOYOUNG KIM, SOKBAE LEE, AND ADAM M. ROSEN

ABSTRACT. We present the `clrbound`, `clr2bound`, `clr3bound`, and `clrtest` commands for estimation and inference on intersection bounds as developed by Chernozhukov et al. (2013). The commands `clrbound`, `clr2bound`, and `clr3bound` provide bound estimates that can be used directly for estimation or to construct asymptotically valid confidence sets. The command `clrbound` provides bound estimates for one-sided lower or upper intersection bounds on a parameter, while `clr2bound` and `clr3bound` provide two-sided bound estimates based on both lower and upper intersection bounds. `clr2bound` uses Bonferroni's inequality to construct two-sided bounds, whereas `clr3bound` inverts a hypothesis test. The former can be used to perform asymptotically valid inference on the identified set or the parameter, while the latter can be used to provide asymptotically valid and generally tighter confidence intervals for the parameter. `clrtest` performs an intersection bound test of the hypothesis that a collection of lower intersection bounds is no greater than zero. Inversion of this test can be used to construct confidence sets based on conditional moment inequalities as described in Chernozhukov et al. (2013). The commands include parametric, series, and local linear estimation procedures, and can be installed from within Stata by typing "`ssc install clrbound`".

KEY WORDS: clr2bound, clrbound, clrtest, clr3bound, bound analysis, conditional moments, partial identification, infinite dimensional constraints, adaptive moment selection.

1. INTRODUCTION

In this paper, we present the `clrbound`, `clr2bound`, `clr3bound`, and `clrtest` commands for estimation and inference on intersection bounds as developed by Chernozhukov et al. (2013). These commands, summarized in Table 1, enable one to perform hypothesis tests and construct set estimates and confidence sets for parameters restricted by intersection bounds. The procedures employ parametric, series, and local linear estimators.

Date: 15 August 2013.
We thank Cathy Redmond, Koohyun Kwon, Jungsik Hyun and Hyunmin Park for capable research assistance. Financial support from the UK Economic and Social Research Council through a grant (RES-589-28-0001) to the ESRC Centre for Microdata Methods and Practice (CeMMAP) and through the funding of the "Programme Evaluation for Policy Analysis" node of the UK National Centre for Research Methods, and from the European Research Council (ERC) grant ERC-2009-StG-240910-ROMETA is gratefully acknowledged.

Command	Description
`clrtest`	Test the hypothesis that the maximum of lower intersection bounds is nonpositive.
`clrbound`	Compute a one-sided bound estimate.
`clr2bound`	Compute two-sided bound estimates using Bonferroni's inequality.
`clr3bound`	Compute two-sided bound estimates by inverting `clrtest`.

TABLE 1. Intersection Bound Commands. Bound estimates can be used to construct asymptotically valid confidence intervals for parameters restricted by intersection bounds.

In Section 2 we recall the underlying framework of the intersection bounds setup from Chernozhukov et al. (2013). In Section 3 we describe the details of how our Stata program conducts hypothesis tests and constructs bound estimates. In Section 4, we explain how to install our Stata module. In Sections 5, 6, 7, and 8 we describe the `clr2bound`, `clrbound`, `clrtest` and `clr3bound` commands, respectively. We explain how each command is used, what each command does, the available command options, and saved results. In Section 9 we illustrate the use of all four of these commands using data from the National Longitudinal Survey of Youth of 1979 (NLSY79), as in Carneiro and Lee (2009). Specifically, we use these commands to estimate and perform inference on returns to education using monotone treatment response and monotone instrumental variable bounds developed by Manski and Pepper (2000).

2. FRAMEWORK

In this paper, we consider intersection bounds of the following form:

$$\max_{j\in\mathcal{J}_l}\sup_{x_j^l\in\mathcal{X}_j^l}\theta_j^l(x_j^l)\le\theta^*\le\min_{j\in\mathcal{J}_u}\inf_{x_j^u\in\mathcal{X}_j^u}\theta_j^u(x_j^u), \tag{1}$$

where θ^* is the parameter of interest, $\{\theta_j^l(\cdot):j\in\mathcal{J}_l\}$ are lower bounding functions, $\{\theta_j^u(\cdot):j\in\mathcal{J}_u\}$ are upper bounding functions, $\mathcal{X}_j^l$ and $\mathcal{X}_j^u$ are predetermined sets, and $\mathcal{J}_l$ and $\mathcal{J}_u$ are index sets with a finite number of positive integers. The interval of all values that lie within the bounds in (1) is the identified set, which we denote Θ_I:

$$\Theta_I\equiv\Big[\max_{j\in\mathcal{J}_l}\sup_{x_j^l\in\mathcal{X}_j^l}\theta_j^l(x_j^l),\min_{j\in\mathcal{J}_u}\inf_{x_j^u\in\mathcal{X}_j^u}\theta_j^u(x_j^u)\Big]. \tag{2}$$

We focus on the case where the bounding functions are conditional expectation functions such that

$$\theta_j^k(\cdot):=E[Y_j^k|X_j^k=\cdot],\ k=l,u,$$

where Y_j^k and X_j^k are the dependent variable and explanatory variables for each j and k, respectively. We allow for the possibility that X_j^k are different or the same across j and k.

The estimation problem of Chernozhukov et al. (2013) is to obtain estimators $\widehat{\theta}_{n0}(p)$ that provide bias-corrected estimates or the endpoints of confidence intervals depending on the chosen value of p, e.g. $p = 1/2$ or $p = 1 - \alpha$. Implementation details can be found in Chernozhukov et al. (2013), who focus on the upper bound for θ^*. However, as explained there, the estimation procedure can be easily adapted for the lower bound for θ^*. The command `clrbound` presented below gives estimators for these one-sided intersection bounds.

If one wishes to perform inference on the identified set in such circumstances, then one can use the intersection of upper and lower one-sided intervals each based on $\tilde{p} = (1 + p)/2$ as an asymptotic level-p confidence set for Θ_I, which is valid by Bonferroni's inequality. The command `clr2bound` below provides this type of confidence interval.

Such confidence intervals are, however, conservative for inference on θ^*, generally providing higher asymptotic coverage for any $\theta \in \Theta_I$. As an alternative, one can construct a sharper confidence interval for θ^* by inverting a test. Specifically, we may consider testing the null hypothesis that a given value, say θ_{null}, is in the identified set by transforming the full set of lower and upper bounds into a collection of only one-sided bounds, and then constructing a confidence interval through the inversion of this test. We offer the command `clr3bound` for this purpose.

3. Implementation

In this section, we describe the details of our implementation for estimation of one-sided bounds. We focus on the lower intersection bounds and drop the l superscript to simplify notation.

Let J denote the number of inequalities concerned. Suppose that we have observations $\{(Y_{ji}, X_{ji}) : i = 1, \ldots, n, j = 1, \ldots, J\}$, where n is the sample size. For each $j = 1, \ldots, J$, let $\mathbf{y}_j$ denote the $n \times 1$ vector whose ith element is Y_{ji} and $\mathbf{X}_j$ the $n \times d_j$ matrix whose ith row is X_{ji}', where d_j is the dimension of X_{ji}. We allow multidimensional $\mathbf{X}_j$ only for parametric estimation. We set $d_j = 1$ for series and local linear estimation.

To evaluate the supremum in (1) numerically, we set a dense set of grid points for each $j = 1, \ldots, J$, say $\{\mathbf{x}_1, \ldots, \mathbf{x}_J\}$, where $\mathbf{x}_j = (x_{j1}', \ldots, x_{jM_j}')'$ for some sufficiently large numbers M_j, where each x_{jm} is a $d_j \times 1$ vector. Also, let $\mathbf{\Psi}_j$ denote the $M_j \times d_j$ matrix whose mth row is x_{jm}', where $m = 1, \ldots, M_j$ and $j = 1, \ldots, J$. Note that the number of grid points can be different for different inequalities.

3.1. **Parametric Estimation.** Define

$$\mathbf{X} := \begin{pmatrix} \mathbf{X}_1 & \cdots & \mathbf{0} \\ & \vdots & \\ \mathbf{0} & \cdots & \mathbf{X}_J \end{pmatrix}, \ \mathbf{y} := \begin{pmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_J \end{pmatrix}, \text{ and } \ \mathbf{\Psi} := \begin{pmatrix} \mathbf{\Psi}_1 & \cdots & \mathbf{0} \\ & \vdots & \\ \mathbf{0} & \cdots & \mathbf{\Psi}_J \end{pmatrix}.$$

Let $\boldsymbol{\theta}_j(\mathbf{x}_j) \equiv (\theta_j(x_{j1}), \cdots, \theta_j(x_{jM_j}))'$ and $\boldsymbol{\theta} \equiv (\boldsymbol{\theta}_1(\mathbf{x}_1)', \ldots, \boldsymbol{\theta}_J(\mathbf{x}_J)')'$. Then the estimator of $\boldsymbol{\theta}$ is $\widehat{\boldsymbol{\theta}} \equiv \mathbf{\Psi}\widehat{\boldsymbol{\beta}}$, where $\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$. Also, the heteroskedasticity-robust standard error of $\widehat{\boldsymbol{\theta}}$, say $\widehat{\mathbf{s}}$, can be computed as

$$\widehat{\mathbf{s}} \equiv \sqrt{\text{diag}_{\text{vec}}(\mathbf{V})},$$

where

$$\mathbf{\Omega} = [\text{diag}(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})]^2, \ \mathbf{V} = \mathbf{\Psi}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{\Omega}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{\Psi}',$$

$\text{diag}(\mathbf{a})$ is the diagonal matrix whose diagonal terms are elements of the vector $\mathbf{a}$, and $\text{diag}_{\text{vec}}(\mathbf{A})$ is the vector whose elements are diagonal elements of the matrix $\mathbf{A}$.

To compute the critical value, say $k(p)$, define

$$\widehat{\mathbf{\Sigma}} := [\text{diag}(\widehat{\mathbf{s}})]^{-1}\mathbf{V}[\text{diag}(\widehat{\mathbf{s}})]^{-1}.$$

Let $\text{chol}(\mathbf{A})$ denote the Cholesky decomposition of the matrix $\mathbf{A}$ such that $\mathbf{A} = \text{chol}(\mathbf{A})\text{chol}(\mathbf{A})'$. Simulate pseudo random numbers from $N(0,1)$ and construct a $\dim(\widehat{\mathbf{\Sigma}}) \times R$-dimensional matrix, say $\mathbf{Z}_R$. Then the critical value is selected as

$$k(p) = \text{the } p\text{th quantile of } \max_{\text{col.}}[\text{chol}(\widehat{\mathbf{\Sigma}})\mathbf{Z}_R], \tag{3}$$

where $\max_{\text{col.}}(\mathbf{B})$ is a set of maximum values in each column of the matrix $\mathbf{B}$. Then our bias-corrected estimator $\widehat{\theta}_{n0}(p)$ for $\max_{j\in\mathcal{J}_l}\sup_{x_j^l\in\mathcal{X}_j^l}\theta_j^l(x_j^l)$ is

$$\widehat{\theta}_{n0}(p) = \max_{\text{col.}}[\mathbf{\Psi}\widehat{\boldsymbol{\beta}} - k(p)\widehat{\mathbf{s}}]. \tag{4}$$

The critical value in (4) is obtained under the least favorable case. To improve the estimator, we carry out the following adaptive inequality selection (AIS) procedure:

(Step 1) Set $\tilde{\gamma}_n \equiv 1 - .1/\log n$. Let ψ_k' denote the kth row of $\mathbf{\Psi}$, where $k = 1, \ldots, \sum_{j=1}^J M_j$. Keep each row ψ_k' of $\mathbf{\Psi}$ if and only if

$$\psi_k'\widehat{\boldsymbol{\beta}} \geq \widehat{\theta}_{n0}(\tilde{\gamma}_n) - 2k(\tilde{\gamma}_n)\widehat{s}_k,$$

where $\widehat{s}_k$ is the kth element of $\widehat{\mathbf{s}}$.

(Step 2) Replace $\mathbf{\Psi}$ with the kept rows of $\mathbf{\Psi}$ in Step 1. Then recompute $\mathbf{V}$ and $\widehat{\mathbf{\Sigma}}$ to update the critical value in (3), and obtain the final estimator $\widehat{\theta}_{n0}(p)$ in (4) with the updated critical value.

3.2. **Series Estimation.** The implementation of series estimation is similar to parametric estimation. For each $j = 1, \ldots, J$, let $p_{nj}(x) \equiv (p_{n,1}(x), \ldots, p_{n,\kappa_j}(x))'$ denote the κ_j-dimensional vector of approximating functions by cubic B-splines. Here, the number of series terms κ_j can be different from one inequality to another. Let $\widetilde{\mathbf{X}}_j$ denote the $n \times \kappa_j$ matrix whose ith row is $p_{nj}(X_{ji})'$ and $\widetilde{\mathbf{\Psi}}_j$ the $M_j \times \kappa_j$ matrix whose mth row is $p_{nj}(x_{jm})'$. Then the same procedure as described in Section 3.1 can be carried out, substituting $\widetilde{\mathbf{X}}_j$ and $\widetilde{\mathbf{\Psi}}_j$ for $\mathbf{X}_j$ and $\mathbf{\Psi}_j$, respectively.

3.3. **Local Linear Estimation.** For any vector $\mathbf{v}$, let $\hat{\boldsymbol{\rho}}_j(\mathbf{v})$ denote the vector whose kth element is the local linear regression estimate of $\mathbf{y}_j$ on $\mathbf{X}_j$ at the kth element of $\mathbf{v}$. In detail, the kth element of $\hat{\boldsymbol{\rho}}_j(\mathbf{v})$, say $\hat{\rho}_j(v_k)$, is defined as follows:

$$\hat{\rho}_j(v_k) \equiv \mathbf{e}_1'(\boldsymbol{X}_{v_k}'\boldsymbol{W}_j\boldsymbol{X}_{v_k})^{-1}\boldsymbol{X}_{v_k}'\boldsymbol{W}_j\boldsymbol{y}_j,$$

where $\mathbf{e}_1 \equiv (1,0)'$,

$$\boldsymbol{X}_{v_k} \equiv \begin{pmatrix} 1 & (X_{j1} - v_k) \\ \vdots & \vdots \\ 1 & (X_{jn} - v_k) \end{pmatrix}, \ \boldsymbol{W}_j \equiv \text{diag}\left(K\left(\frac{X_{j1} - v_k}{h_j}\right), \cdots, K\left(\frac{X_{jn} - v_k}{h_j}\right)\right),$$

$K(\cdot)$ is a kernel function, and h_j is the bandwidth for inequality j. In our implementation, we used the following kernel function:

$$K(s) = \frac{15}{16}(1 - s^2)^2 1(\mid s \mid \leq 1).$$

Then the estimator of $\boldsymbol{\theta} \equiv (\boldsymbol{\theta}_1(\mathbf{x}_1)', \ldots, \boldsymbol{\theta}_J(\mathbf{x}_J)')'$ is $\hat{\boldsymbol{\theta}} \equiv (\hat{\boldsymbol{\rho}}_1(\boldsymbol{\psi}_1)', \ldots, \hat{\boldsymbol{\rho}}_J(\boldsymbol{\psi}_J))'$, where $\boldsymbol{\psi}_j$ denotes the $M_j \times 1$ vector whose mth element is x_{jm}.

Now let $\hat{s}_j$ denote the $M_j \times 1$ vector whose mth element is $\sqrt{\overline{g^2}_{jm}(\mathbf{y}_j, \mathbf{X}_j)/nh_j}$, where

$$\overline{g^2}_{jm}(\mathbf{y}_j, \mathbf{X}_j) = n^{-1}\sum_{i=1}^{n} \hat{g}_{ji}(Y_{ji}, X_{ji}, x_{jm})^2,$$

$$\hat{g}_{ji}(Y_{ji}, X_{ji}, x_{jm}) = \frac{Y_{ji} - \hat{\rho}_j(X_{ji})}{\sqrt{h_j}\hat{f}_j(x_{jm})} K\left(\frac{x_{jm} - X_{ji}}{h_j}\right),$$

$\hat{f}_j(x_{jm})$ is the kernel estimate of the density of the covariate for the jth inequality, evaluated at x_{jm}. Then, $\hat{\boldsymbol{s}}$ can be computed as $\hat{\boldsymbol{s}} = (\hat{s}_1', \ldots, \hat{s}_J')'$.

To compute the critical value, $k(p)$, let $\mathbf{\Phi}_j$ denote the $M_j \times n$ matrix whose mth row is $(\hat{g}_{j1}(Y_{j1}, X_{j1}, x_{jm}), \ldots, \hat{g}_{jn}(Y_{jn}, X_{jn}, x_{jm}))/\sqrt{nh_j\overline{g^2}_{jm}(\mathbf{y}_j, \mathbf{X}_j)}$. Define

$$\mathbf{\Phi} \equiv \begin{pmatrix} \mathbf{\Phi}_1 \\ \vdots \\ \mathbf{\Phi}_J \end{pmatrix}.$$

We simulate pseudo random numbers from the $N(0,1)$ distribution and construct a $n \times R$ matrix, $\mathbf{Z_R}$. Then, the critical value is selected as

$$k(p) = \text{the } p\text{th quantile of } \max_{\text{col.}}[\mathbf{\Phi Z_R}]. \tag{5}$$

The calculation of the bias-corrected estimator $\widehat{\theta}_{n0}(p)$ is almost the same as that of parametric estimation. That is,

$$\widehat{\theta}_{n0}(p) = \max_{\text{col.}}[\hat{\boldsymbol{\theta}} - k(p)\widehat{\mathbf{s}}]. \tag{6}$$

However, the AIS procedure is slightly different since we do not use $\mathbf{\Psi}$ in local linear estimation.

(Step 1) Set $\tilde{\gamma}_n \equiv 1 - .1/\log n$. Keep the mth row of each $\mathbf{\Phi}_j$, $j = 1, \ldots, J$, if and only if

$$\hat{\rho}_j(x_{jm}) \geq \widehat{\theta}_{n0}(\tilde{\gamma}_n) - 2k(\tilde{\gamma}_n)\widehat{s}_{jm},$$

where $\widehat{s}_{jm}$ is the mth element of $\widehat{\mathbf{s}}_j$.

(Step 2) For $j = 1, \ldots, J$, replace $\mathbf{\Phi}_j$ with the kept rows of $\mathbf{\Phi}_j$ in Step 1. Then recompute the critical value in (5), and obtain the final estimator $\widehat{\theta}_{n0}(p)$ with the updated critical value.

4. Installation of the clrbound package

All Stata commands below are available at the Statistical Software Components (SSC) archive. Our Stata module called `clrbound` (Chernozhukov et al. (2013))[1] can be installed from within Stata by typing "`ssc install clrbound`". All of our commands require the package `moremata` (Jann (2005))[2] , which can also be installed by typing "`ssc install moremata, replace`" in the Stata command window.

5. The clr2bound command

5.1. Syntax.

The syntax of `clr2bound` is as follows:

`clr2bound` ((*lowerdepvar1 indepvars1 range1*) (*lowerdepvar2 indepvars2 range2*) ... (*lowerdepvarN indepvarsN rangeN*)) ((*upperdepvarN+1 indepvarsN+1 rangeN+1*) (*upperdepvarN+2 indepvarsN+2 rangeN+2*) ... (*upperdepvarN+M indepvarsN+M rangeN+M*)) [*if*] [*in*] [, `method("series"|"local") notest null(`*real* `) level(`*numlist* `) noais minsmooth(#) maxsmooth(#) noundersmooth bandwidth(`*numlist* `) rnd(#) norseed seed(#)`]

[1] http://econpapers.repec.org/software/bocbocode/s457674.htm.
[2] http://econpapers.repec.org/software/bocbocode/s455001.htm.

5.2. **Description.** `clr2bound` estimates two-sided intersection bounds on a parameter. The variables called *lowerdepvar1*∼*lowerdepvarN* are the dependent variables (Y_j^l's) for the lower bounding functions and the *upperdepvarN+1*∼*upperdepvarN+M* are the dependent variables (Y_j^u's) for the upper bounding functions, respectively. The variables *indepvars1*∼*lowerdepvarN+M* in the syntax refer to explanatory variables for the corresponding dependent variables. Recall that `clr2bound` allows for multidimensional *indepvars* for parametric estimation, but only for a one dimensional independent variable for series and local linear estimation.

The variables *range1*∼*rangeN+M* are sets of grid points over which the bounding function is estimated. The number of observations for the *range* is not necessary the same as the number of observations for the *depvar* and *indepvars*. The latter is the sample size, whereas the former is the number of grid points to evaluate the maximum or minimum values of the bounding functions.

It should be noted that the parentheses must be used properly. Variables for lower bounds and upper bounds must be put in additional parentheses separately. For example, if there are two variable sets, (*ldepvar1 indepvars1 range1*) and (*ldepvar2 indepvars2 range2*), for the lower bounds estimation and one variable set, (*udepvar1 indepvars3 range3*), for the upper bounds estimation, the right syntax for two-sided intersection bounds estimation is ((*ldepvar1 indepvars1 range1*)(*ldepvar2 indepvars2 range2*))((*udepvar1 indepvars3 range3*)).

5.3. **Options.** `method(`*string*`)` specifies the method of estimation. By default, `clr2bound` will conduct parametric estimation. Specifying `method("series")`, `clr2bound` will conduct series estimation with cubic B-splines. Specifying `method("local")`, will result in local linear estimation.

`notest` determines whether `clr2bound` conducts a test or not. `clr2bound` provides a test for the null hypothesis that the specified value is in the intersection bounds at the confidence levels specified in the `level` option below. By default, `clr2bound` conducts the test. Specifying this option causes `clr2bound` to output Bonferroni bounds only.

`null(`*real*`)` specifies the value for θ^* under the null hypothesis of the test we described above. The default value is `null(0)`.

`level(`*numlist*`)` specifies confidence levels. *numlist* has to be filled with real numbers between 0 and 1. In particular, if this option is specified as `level(0.5)`, the result is the half-median-unbiased estimator of the parameter of the interest. The default is `level(0.5 0.9 0.95 0.99)`.

`noais` determines whether the adaptive inequality selection would be applied or not. The adaptive inequality selection (AIS) helps to get sharper bounds by using the problem-dependent cutoff to drop irrelevant grid points of the *range*. The default is to use AIS.

`minsmooth(#)` and `maxsmooth(#)` specify the minimum and maximum possible numbers of approximating functions considered in the cross validation procedure

for B-splines. Specifically, the number of approximating functions $\hat{K}_{cv}$ is set to the minimizer of the leave-one-out least squares cross validation score within this range. For example, if a user inputs `minsmooth(5)` and `maxsmooth(9)`, $\hat{K}_{cv}$ is chosen from the set {5,6,7,8,9}. The procedure calculates this number separately for each inequality. The default is `minsmooth(5)` and `maxsmooth(20)`. If under-smoothing is performed, the number of approximating functions K ultimately used will be given by the largest integer smaller than $\hat{K}_{cv}$ times the under-smoothing factor $n^{-1/5} \times n^{2/7}$, see option `noundersmooth` below. This option is only available for series estimation.

`bandwidth(#)` specifies the value of the bandwidth used in the local linear estimation. By default, `clr2bound` calculates a bandwidth for each inequality. With undersmoothing, we use the rule of thumb bandwidth $h = \hat{h}_{ROT} \times \hat{s}_v \times n^{1/5} \times n^{-2/7}$ where $\hat{s}_v$ is the square root of the sample variance of V, and $\hat{h}_{ROT}$ is the rule-of-thumb bandwidth for estimation of $\theta(v)$ with Studentized V. See Chernozhukov et al. (2013) for the exact form of $\hat{h}_{ROT}$. When the `bandwidth(#)` is specified, `clr2bound` uses the given bandwidth as the global bandwidth for every inequality. This option is only available for local linear estimation.

`noundersmooth` determines whether under-smoothing is carried out, with the default being to under-smooth. In series estimation, under-smoothing is implemented by first computing $\hat{K}_{cv}$ as the minimizer of the leave-one-out least squares cross validation score. We then set the number of approximating functions to K, given by the largest integer which is smaller than or equal to $\hat{K} := \hat{K}_{cv} \times n^{-1/5} \times n^{2/7}$. The `noundersmooth` option simply uses $\hat{K}_{cv}$. For local linear estimation under-smoothing is done by setting the bandwidth to $h = \hat{h}_{ROT} \times \hat{s}_v \times n^{1/5} \times n^{-2/7}$, where $\hat{h}_{ROT}$, is the rule-of-thumb bandwidth used in Chernozhukov et al. (2013). The `noundersmooth` option instead uses $\hat{h}_{ROT} \times \hat{s}_v$. This option is only available for series and local linear estimation.

`rnd(#)` specifies the number of columns of the random matrix generated from the standard normal distribution. This matrix is used for computation of critical values. For example, if the number is 10000 and the level is 0.95, we choose the 0.95 quantile from 10000 randomly generated elements. The default is `rnd(10000)`.

`norseed` determines whether the seed number for the simulation used in the calculation would be reset. If a user wants to use this command for simulations such as Monte Carlo method, he can prevent the command from resetting the seed number every lap by using this option. The default is to reset the seed number.

`seed(#)` specifies the seed number for the random number generation described above. To prevent the estimation result from changing one particular values to another randomly, `clr2bound` always conducts `set seed #` initially. The default is `seed(0)`.

5.4. **Saved results.** In the following, "l.b.e." stands for lower bound estimation, "u.b.e." for upper bound estimation, and "ineq" means an inequality. (i) denotes the i-th inequality. (lev) means the confidence level's decimal part. For example, when the confidence level is 97.5% or 0.975, (lev) is 975. The number of elements in (lev) is equal to the number of confidence levels specified by the `level` option. Some results are only available for series or local linear estimation.

`clr2bound` saves the following in `e()`. Note that for this and all other commands, in the saved AIS results 1 is used to denote values that were kept in the index set, and 0 values that were dropped.

Scalars

`e(N)`	#of observations	`e(null)`	The null hypothesis
`e(l_ineq)`	#of ineq's in l.b.e.	`e(u_ineq)`	#of ineq's in r.b.e.
`e(l_grid(i))`	#of grid points in (i) of l.b.e.	`e(u_grid(i))`	#of grid points in (i) of r.b.e.
`e(l_nf_x(i))`	#of approx. functions for l.b.e. at x(i)	`e(u_nf_x(i))`	#of approx. functions for u.b.e. at x(i)
`e(l_bdwh(i))`	bandwidth for (i) of l.b.e	`e(l_bdwh(i))`	bandwidth for (i) of u.b.e.
`e(lbd(lev))`	est. results of l.b.e.	`e(ubd(lev))`	est. results of u.b.e.
`e(lcl(lev))`	critical value of l.b.e.	`e(ucl(lev))`	critical value of l.b.e.
`e(t_det(lev))`	`1` : in the bound `0` : not	`e(t_cvl(lev))`	critical value of test
`e(t_bd(lev))`	est. results of test	`e(t_nf_x(i))`	#of approx. functions in test

Macros

`e(cmd)`	"`clr2bound`"	`e(title)`	"CLR Intersection Bounds (*method*)"
`e(ldepvar)`	dep. var. in l.b.e.	`e(udepvar)`	dep. var. in r.b.e.
`e(level)`	confidence levels	`e(smoothing)`	"(NOT) Undersmoothed"
`e(l_indep(i))`	indep. var. in (i) of l.b.e.	`e(u_indep(i))`	indep. var. in (i) of u.b.e.
`e(l_range(i))`	range in (i) of l.b.e.	`e(u_range(i))`	range in (i) of u.b.e.

Matrices

`e(l_omega)`	$\hat{\Omega}_n$ for l.b.e.	`e(u_omega)`	$\hat{\Omega}_n$ for u.b.e.
`e(l_theta(i))`	$\hat{\theta}_n(v)$ for each v in l.b.e.	`e(u_theta(i))`	$\hat{\theta}_n(v)$ for each v in u.b.e.
`e(l_se(i))`	$s_n(v)$ for each v in l.b.e.	`e(u_se(i))`	$s_n(v)$ for each v in u.b.e.
`e(l_ais(i))`	AIS result for each v in l.b.e.	`e(u_ais(i))`	AIS result for each v in u.b.e.
`e(t_omega)`	$\hat{\Omega}_n$ for test	`e(t_theta(i))`	$\hat{\theta}_n(v)$ for each v in test
`e(t_se(i))`	$s_n(v)$ for each v in test	`e(t_ais(i))`	AIS result for each v in test

See Chernozhukov et al. (2013) for details on $\hat{\theta}_n(v)$, $s_n(v)$, and $\hat{\Omega}_n$.

6. The clrbound command

6.1. **Syntax.** The syntax of `clrbound` is as follows:

`clrbound ( `*depvar1 indepvars1 range1*` ) ( `*depvar2 indepvars2 range2*` ) ... ( `*depvarN indepvarsN rangeN*` ) [ `*if*` ] [ `*in*` ] [ , lower | upper method("series"|"local") level(`*numlist*` ) noais minsmooth(#) maxsmooth(#) nounderssmooth bandwidth(`*numlist*` ) rnd(#) norseed seed(#) ]`

6.2. **Description.** `clrbound` estimates the one-sided lower or upper intersection bounds of a parameter. The variables are defined similarly as for `clr2bound`.

6.3. **Options.** `lower` specifies whether the estimation is for the lower bound or the upper bound. By default, it will return the upper intersection bound. Specifying `lower`, `clrbound` will return the lower intersection bound.

Other options of the `clrbound` are the same as those of the `clr2bound`. However, the `clrbound` does not have `notest` and `null` options because it does not provide a testing procedure.

6.4. **Saved results.** In the following, we use the same abbreviations as in Section 5.4. The `clrbound` saves the following in `e()`:

Scalars

`e(N)`	#of observations	`e(n_ineq)`	#of inequality
`e(grid(i))`	#of grids points (i)	`e(nf_x(i))`	#of approx. functions in (i)
`e(bd(lev))`	results of estimation	`e(cl(lev))`	critical value
`e(bdwh(i))`	bandwidth for (i)		

Macros

`e(cmd)`	"`clrbound`"	`e(title)`	"CLR Intersection (upper/lower) Bounds (*method*)"
`e(depvar)`	dependent variables	`e(level)`	confidence levels
`e(smoothing)`	"(NOT) Undersmoothed"	`e(indep(i))`	indep. variables in (i)
`e(range(i))`	range in (i)		

Matrices

`e(omega)`	$\hat{\Omega}_n$	`e(theta(i))`	$\hat{\theta}_n(v)$ for each v
`e(se(i))`	$s_n(v)$ for each v	`e(ais(i))`	AIS result for each v

7. The clrtest command

7.1. **Syntax.** The syntax of `clrtest` is as follows:

`clrtest` (*depvar1 indepvars1 range1*) (*depvar2 indepvars2 range2*) ... (*depvarN indepvarsN rangeN*) [*if*] [*in*] [, `method("series"|"local")` `level(`*numlist* `)` `noais` `minsmooth(#)` `maxsmooth(#)` `noundersmooth` `bandwidth(`*numlist* `)` `rnd(#)` `norseed` `seed(#)`]

7.2. **Description.** `clrtest` offers a more comprehensive testing procedure than the `clr2bound` does. It returns the output telling whether the result of the lower intersection bound estimation deducted from the given *depvar*'s and confidence levels is smaller than 0 or not. For example, suppose that one wants to test the null hypothesis that 0.59 is in the 95% confidence interval for `yl` and `yu`. Then, we can make two inequalities, $yl\_test = yl - 0.59$ and $yu\_test = 0.59 - yu$. If the resulting estimator is larger than 0, the procedure rejects the null hypothesis. The variables are defined similarly as in the `clr2bound`.

7.3. **Options.** Since the options of the `clrtest` are the same as those of the `clrbound`, the explanation of options is omitted.

7.4. **Saved results.** Other saved results are the same as those of `clrbound` except the following:

Scalars

`e(det(lev))` rejected : 0, not rejected : 1

8. The clr3bound command

8.1. **Syntax.** The syntax of `clr3bound` is as follows:

`clr3bound` ((*lowerdepvar1 indepvars1 range1*) (*lowerdepvar2 indepvars2 range2*) ... (*lowerdepvarN indepvarsN rangeN*)) ((*upperdepvarN+1 indepvarsN+1 rangeN+1*) (*upperdepvarN+2 indepvarsN+2 rangeN+2*) ...(*upperdepvarN+M indepvarsN+M rangeN+M*)) [*if*] [*in*] [, `start(#) end(#) grid(#) method("series"|"local") level(#) noais minsmooth(#) maxsmooth(#) nounderssmooth bandwidth(#) rnd(#) norseed seed(#)`]

8.2. **Description.** `clr3bound` estimates the two-sided intersection bound of a parameter by carrying out pointwise tests using the `clrtest` command. Note that when one-sided intersection bounds are concerned, there is no need to implement pointwise tests. This is because in (1), we consider one-sided intersection bounds with θ^* additively separable with respect to the bounding functions. In this case, the command `clrbound` estimates tight bounds for θ^*.

Since this command is only relevant for two-sided intersection bounds, a user should input variables for both lower and upper bounds to calculate the bound. The variables are defined similarly as in the `clr2bound`. This command generally provides tighter bounds than those provided by the `clrtest` command over equi-spaced grids, which employ Bonferroni's inequality. Unlike the previous commands, `clr3bound` can only deal with one confidence level.

8.3. **Options.** `stepsize(#)` specifies the distance between two consecutive grid points. The procedure divides Bonferroni's bound into equi-distanced grids and implements the `clrtest` command for each grid point to determine a possible tighter bound. The default is `0.01`.

`level(#)` specifies the confidence level of the estimation. Different from previous commands, `clr3bound` can only deal with one confidence level. The default is `0.95`. Other options are exactly the same as those of `clr2bound`.

8.4. **Saved results.** `clr3bound` saves the following in `e()`:

Scalars

`e(N)`	#of observations	`e(step)`	step size
`e(level)`	confidence level		
`e(l_ineq)`	#of ineq's in l.b.e.	`e(u_ineq)`	#of ineq's in r.b.e.
`e(l_grid(i))`	#of grids points for l.b.e. at observation (i)	`e(u_grid(i))`	#of grids points for r.b.e. at observation (i)
`e(l_nf_x(i))`	#of approx. functions in (i) of l.b.e.	`e(u_nf_x(i))`	#of approx. functions in (i) of u.b.e.
`e(l_bdwh(i))`	bandwidth for (i) of l.b.e	`e(l_bdwh(i))`	bandwidth for (i) of u.b.e.
`e(lbd)`	est. results of l.b.e.	`e(ubd)`	est. results of u.b.e.
`e(lbd(lev))`	Bonferroni results of l.b.e.	`e(ubd(lev))`	Bonferroni results of u.b.e.
`e(lcl(lev))`	critical value of l.b.e.	`e(ucl(lev))`	critical value of l.b.e.

Macros

`e(cmd)`	"`clr3bound`"	`e(title)`	"CLR Intersection Bounds: inverting test bounds"
`e(ldepvar)`	dep. var. in l.b.e.	`e(udepvar)`	dep. var. in r.b.e.
`e(method)`	Estimation method	`e(smoothing)`	"(NOT) Undersmoothed"
`e(l_indep(i))`	indep. var. in (i) of l.b.e.	`e(u_indep(i))`	indep. var. in (i) of u.b.e.
`e(l_range(i))`	range in (i) of l.b.e.	`e(u_range(i))`	range in (i) of u.b.e.

Matrices

`e(l_omega)`	$\hat{\Omega}_n$ for l.b.e.	`e(u_omega)`	$\hat{\Omega}_n$ for u.b.e.
`e(l_theta(i))`	$\hat{\theta}_n(v)$ for each v in l.b.e.	`e(u_theta(i))`	$\hat{\theta}_n(v)$ for each v in u.b.e.
`e(l_se(i))`	$s_n(v)$ for each v in l.b.e.	`e(u_se(i))`	$s_n(v)$ for each v in u.b.e.
`e(l_ais(i))`	AIS result for each v in l.b.e.	`e(u_ais(i))`	AIS result for each v in u.b.e.

9. Examples

To illustrate the usage of `clrbound`, `clr2bound`, `clr3bound`, and `clrtest`, we use the data from the National Longitudinal Survey of Youth of 1979 (NLSY79), as in Carneiro and Lee (2009). The variable `lnwage` is hourly log wage, `eduyr` is years of schooling and `afqt` is the Armed Forces Qualifying Test score. As in Chernozhukov et al. (2013), we consider the MIV-MTR (monotone instrument variable - monotone treatment response) bounds of Manski and Pepper (2000). Specifically, let the parameter of interest be $\theta^* = \mathrm{P}[Y_i(t) > y | V_i = v]$ at $y = \log(16)$ (approximately the 70th percentile of hourly wages), $v = 0$ and $t = 13$ (college attendees, in other words those who have more years of schooling than high school graduates). Then the MIV-MTR upper bound is

$$\theta^* \leq \inf_{u \geq v} \mathrm{P}[1\{Y_i > y\} \cdot 1\{t \leq Z_i\} + 1\{t > Z_i\} | V_i = u],$$

where Y_i is the observed wages, Z_i is years of schooling, and V_i is the AFQT score. The lower bound is

$$\theta^* \geq \sup_{u \leq v} \mathrm{P}[1\{Y_i > y\} \cdot 1\{t \geq Z_i\} | V_i = u].$$

Note that, in this case, when choosing *range*, one has to be careful due to the MIV bound. In other words, when estimating the intersection bound, the *range* is different between the lower and upper bounds. A new variable which contains grid points larger(smaller) than 0 should be used for the upper(lower) bound estimation, since we want to know the value θ^* at $v = 0$.

We include *range* variables in our NLSY79 dataset, `vl_afqt` for lower intersection bounds, and `vu_afqt` for upper bounds which contain 101 grid points from -2 to 0 and 0 to 2, respectively. The commands we used for making the *range* variables are as follows:

```
. egen vl_afqt = fill("-2 -1.98")

. replace vl_afqt = . if vl_afqt > 0
(1943 real changes made, 1943 to missing)

. egen vu_afqt = fill("0 0.02")

. replace vu_afqt = . if vu_afqt > 2
(1943 real changes made, 1943 to missing)
```

9.1. **clr2bound.** First of all, one should create the dependent variables. The dataset provided for this paper contains these variables, `yl` for the lower bound and `yu` for the upper bound already. The commands for making these variables were as follows:

```
. gen yl = (lnwage > log(16)) * (eduyr >= 13)

. gen yu = (lnwage > log(16)) * (eduyr >= 13) + (eduyr < 13)
```

We compare three estimation methods (parametric, local linear, and series estimation). For the sake of illustration we also include the test result for whether or not 0.1 is in the two-sided intersection bounds using series estimation. The results are as follows:

```
. use NLSY, clear

. clr2bound ((yl afqt vl_afqt))((yu afqt vu_afqt)), notest

CLR Intersection Bounds (Parametric)                    Number of obs : 2044

< Lower Side >
Inequality #1 : yl (# of Grid Points : 101, Independent Variables : afqt )
< Upper Side >
Inequality #1 : yu (# of Grid Points : 101, Independent Variables : afqt )

AIS(adaptive inequality selection) is applied

   Bonferroni Bounds                 |                 Value
-------------------------------------+-----------------------------------------
50% two-sided confidence interval    |           [ 0.2139264, 0.5663448 ]
90% two-sided confidence interval    |           [ 0.2057242, 0.5879866 ]
95% two-sided confidence interval    |           [ 0.2033481, 0.5947234 ]
99% two-sided confidence interval    |           [ 0.1981462, 0.6064652 ]
-------------------------------------------------------------------------------

. clr2bound ((yl afqt vl_afqt))((yu afqt vu_afqt)), notest met("local")

CLR Intersection Bounds (Local Linear)                  Number of obs : 2044

< Lower Side >
Inequality #1 : yl (# of Grid Points : 101, Independent Variables : afqt )
< Upper Side >
```

```
Inequality #1 : yu (# of Grid Points : 101, Independent Variables : afqt )

AIS(adaptive inequality selection) is applied
Bandwidths are undersmoothed

   Bonferroni Bounds                 |                Value
-------------------------------------+-----------------------------------------
50% two-sided confidence interval    |           [ 0.1196608, 0.6405670 ]
90% two-sided confidence interval    |           [ 0.1052949, 0.6591008 ]
95% two-sided confidence interval    |           [ 0.1005279, 0.6657757 ]
99% two-sided confidence interval    |           [ 0.0884076, 0.6782955 ]
-------------------------------------------------------------------------------

. clr2bound ((yl afqt vl_afqt))((yu afqt vu_afqt)), met("series") null(0.1)

CLR Intersection Bounds (Series)                        Number of obs : 2044
Estimation Method : Cubic B-Spline (Undersmoothed)

< Lower Side >
-------------------------------------------------------------------------------
Inequality #1 : yl (# of Grid Points : 101, Independent Variables : afqt )
Numbers of Approximating Functions : 9
< Upper Side >
-------------------------------------------------------------------------------
Inequality #1 : yu (# of Grid Points : 101, Independent Variables : afqt )
Numbers of Approximating Functions : 9

AIS(adaptive inequality selection) is applied

   Bonferroni bounds                 |                Value
-------------------------------------+-----------------------------------------
50% two-sided confidence interval    |           [ 0.1026632, 0.6262088 ]
90% two-sided confidence interval    |           [ 0.0868428, 0.6457816 ]
95% two-sided confidence interval    |           [ 0.0819432, 0.6519276 ]
99% two-sided confidence interval    |           [ 0.0716845, 0.6651358 ]
-------------------------------------------------------------------------------

< Testing Result >                                    Null Hypothesis : .1
The value .1 is NOT in the 50 % confidence interval for two-sided bounds
The value .1 is in the 90 % confidence interval for two-sided bounds
The value .1 is in the 95 % confidence interval for two-sided bounds
The value .1 is in the 99 % confidence interval for two-sided bounds
```

The results show that the parametric bound is the narrowest. In local linear and series estimation, the output contains information about bandwidths and numbers of approximating functions, respectively. Notice that since we do not specify `level`, the procedure automatically gave four different confidence levels: 50%, 90%, 95%, and 99%, respectively. The testing result for series estimation is shown at the last part of the output.

9.2. **clrbound.** In this section, we show how estimation of the one-sided intersection bounds works. For illustration we generated another inequality for lower bounds, `yl2`:

```
. gen yl2 = -yu
```

We also added an additional independent variable, `afqt2`. This variable takes value of `afqt` squared. Accordingly, we add an additional range variable, `vl_afqt2`. We

use these variables only for the inequality #1, `yl`, to illustrate that the dimension of independent variables of one inequality may not be the same as that of other inequalities in the parametric approach. The result is as follows:

```
. clrbound (yl afqt afqt2 vl_afqt vl_afqt2)(yl2 afqt vu_afqt), lower

CLR Intersection Lower Bounds (Parametric)               Number of obs : 2044
Inequality #1 : yl (# of Grid Points : 101, Independent Variables : afqt afqt2 )
Inequality #2 : yl2 (# of Grid Points : 101, Independent Variables : afqt )

AIS(adaptive inequality selection) is applied

                                    |                Value
------------------------------------+-----------------------------------------
half-median-unbiased est.           |              0.1569847
90% one-sided confidence interval   |         [ 0.1440277, inf)
95% one-sided confidence interval   |         [ 0.1405620, inf)
99% one-sided confidence interval   |         [ 0.1336535, inf)
-------------------------------------------------------------------------------
```

9.3. **clrtest.** The result of testing the null hypothesis that 0.59 is in the 95% confidence interval of the parametric estimation is as follows:

```
. gen yl_test = yl - 0.59

. gen yu_test = 0.59 - yu

. clrtest (yl_test afqt vl_afqt)(yu_test afqt vu_afqt), level(0.95)

CLR Intersection Bounds (Test)                           Number of obs : 2044
Inequality #1 : yl_test (# of Grid Points : 101, Independent Variables : afqt )
Inequality #2 : yu_test (# of Grid Points : 101, Independent Variables : afqt )

AIS(adaptive inequality selection) is applied

< Testing Result >
The testing value is NOT in the 95 % confidence interval.
```

9.4. **clr3bound.** This command can obtain a tighter, or less conservative, confidence interval than the one given by `clr2bound`, which uses Bonferroni's inequality. The tighter bound is obtained as follows:

```
. clr3bound ((yl afqt vl_afqt)) ((yu afqt vu_afqt))

CLR Intersection Bounds: Test inversion bounds           Number of obs : 2044
Method : Parametric estimation                               Step size : .01
AIS(adaptive inequality selection) is applied

95% Bonferroni bounds:     (0.2030250 , 0.5945532)
95% Test inversion bounds: (0.2233481 , 0.5747234)
```

The last two lines of the result show that indeed the bound we obtain by using `clr3bound` is tighter than that which uses Bonferroni's inequality.

References

Carneiro, P. and S. Lee (2009). Estimating distributions of potential outcomes using local instrumental variables with an application to changes in college enrollment and wage inequality. *Journal of Econometrics 149*, 191–208.

Chernozhukov, V., W. Kim, S. Lee, and A. Rosen (2013). CLRBOUND: Stata module to perform estimation and inference on intersection bounds. Boston College Statistical Software Components S457674.

Chernozhukov, V., S. Lee, and A. Rosen (2013). Intersection bounds: Estimation and inference. *Econometrica 81*(2), 667–737.

Jann, B. (2005). MOREMATA: Stata module (mata) to provide various functions. Boston College Statistical Software Components S455001.

Manski, C. F. and J. V. Pepper (2000, July). Monotone instrumental variables: With an application to the returns to schooling. *Econometrica 68*(4), 997–1010.

Department of Economics, Massachusetts Institute of Technology, Cambridge, MA, USA.
E-mail address: vchern@mit.edu

Department of Economics, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 151-742, Republic of Korea.
E-mail address: burnfire83@gmail.com

Department of Economics, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul, 151-742, Republic of Korea, and Centre for Microdata Methods and Practice, Institute for Fiscal Studies, 7 Ridgmount Street, London, WC1E 7AE, UK.
E-mail address: sokbae@gmail.com

Department of Economics, University College London, Gower Street, London, WC1E 6BT, UK, and Centre for Microdata Methods and Practice, Institute for Fiscal Studies, 7 Ridgmount Street, London, WC1E 7AE, UK.
E-mail address: adam.rosen@ucl.ac.uk