The Stata Journal (2017)
17, Number 1, pp. 56–72

Commands for testing conditional moment inequalities and equalities

Donald W. K. Andrews
Yale University
New Haven, CT
donald.andrews@yale.edu

Wooyoung Kim
University of Wisconsin–Madison
Madison, WI
wkim68@wisc.edu

Xiaoxia Shi
University of Wisconsin–Madison
Madison, WI
xshi@ssc.wisc.edu

Abstract. In this article, we present two commands (`cmi_test` and `cmi_interval`) to implement the testing and inference methods for conditional moment inequality or equality models proposed in Andrews and Shi (2013, *Econometrica* 81: 609–666). The `cmi_test` command tests the validity of a finite number of conditional moment equalities or inequalities. This test returns the value of the test statistic, the critical values at significance levels 1%, 5%, and 10%, and the *p*-value. The `cmi_interval` command returns the confidence interval for a one-dimensional parameter defined by intersection bounds. We obtain this confidence interval by inverting `cmi_test`. All procedures implemented are uniformly asymptotically valid under appropriate conditions (specified in Andrews and Shi [2013]).

Keywords: st0467, cmi_test, cmi_interval, conditional moment inequalities and equalities, confidence interval, uniformly asymptotically valid test

1 Introduction

This article provides a brief introduction to conditional moment inequality or equality testing and describes the new `cmi_test` and `cmi_interval` commands. The `cmi_test` command implements the testing procedure proposed in Andrews and Shi (2013) for general moment inequality models, with a finite number of conditional moment restrictions and a finite-dimensional parameter. The `cmi_interval` command returns confidence intervals for a one-dimensional parameter bounded above or below by a finite number of conditional moments by inverting the testing procedure proposed in Andrews and Shi (2013).

The package we describe is not intended for computing confidence intervals for θ, unless the setting is the one associated with `cmi_interval`. Computing confidence intervals in a general setting requires numerically sketching out the set of θ values for which `cmi_test` returns an acceptance. Simple grid-search algorithms for this task become exponentially more costly as the dimension of θ increases. Other commonly used statistical software packages offer more efficient algorithms, but we are not aware of their implementation in Stata.

Null hypotheses arise frequently in econometrics in the form of conditional moment inequalities or equalities, for example, when testing the sign of the conditional average treatment effect, when certain incomplete models lead to conditional moment inequality or equality restrictions on parameters, and when testing the fundamental assumptions for the local average treatment-effect estimator (see Mourifié and Wan [2014]). Andrews and Shi (2013) first transform the conditional moment inequalities or equalities into a large number of unconditional moment inequalities or equalities, then construct a test statistic based on these unconditional moment inequalities or equalities. The resulting test is uniformly asymptotically valid and consistent against all fixed alternatives. Chernozhukov, Lee, and Rosen (2013) and Lee, Song, and Whang (2013) propose two main alternatives to the Andrews and Shi (2013) test, both based on nonparametric estimators of the conditional moment inequalities or equalities.[1] All three tests are consistent and do not dominate one another in terms of power. In practice, one may choose a test based on computational feasibility or implement more than one test for more robust conclusions.

The commands we describe offer a rich set of options to allow the user to fine tune the procedure. However, in most applications, the default options—as recommended in Andrews and Shi (2013)—work well without much user input.

We use the following notation throughout this article: $\lfloor a \rfloor$ denotes the largest integer less than or equal to a, and $\lceil a \rceil$ denotes the smallest integer larger than or equal to a.

2 Framework

2.1 Parameter inference based on conditional moment inequalities and equalities

Consider an independent and identically distributed sample $\{\mathbf{W}_i\}_{i=1}^n$. Let $\mathbf{X}_i$ be a vector of instrumental variables, which is a subvector of $\mathbf{W}_i$. Conditional moment inequality and equality models are

$$\begin{aligned} &E\{m_j(\mathbf{W}_i,\theta_0)|\mathbf{X}_i\} \geq 0 \text{ for } j=1,\ldots,p \\ &E\{m_j(\mathbf{W}_i,\theta_0)|\mathbf{X}_i\} = 0 \text{ for } j=p+1,\ldots,k, \text{ almost surely} \end{aligned} \tag{1}$$

p and k are two nonnegative integers so that $k \geq p$ and $\mathbf{m}(\cdot,\theta_0) := \{m_1(\cdot,\theta_0),\ldots, m_k(\cdot,\theta_0)\}'$ is a vector of moment functions of the observables that are known up to the parameter θ_0. The set $\Theta \subseteq R^{d_\theta}$ denotes the parameter space for θ_0. The moment functions need not depend on some elements of $\mathbf{W}_i$, which makes those elements excluded variables. The conditional moment inequality model arises in many modeling contexts. We give an example later, and more examples are in Andrews and Shi (2013).

In a conditional moment inequality model, the parameter θ_0 may or may not be point identified. Thus a consistent point estimator for θ_0 may or may not exist, and typical t

1. Commands for the procedure in Chernozhukov, Lee, and Rosen (2013) are introduced in Chernozhukov et al. (2015).

test-based confidence intervals do not apply. However, one can still test hypotheses on the parameter, such as

$$H_0\colon \theta_0 = \theta \tag{2}$$

for a given value θ. Andrews and Shi (2013) propose—and the `cmi_test` command implements—a test of the above hypothesis. Testing this hypothesis amounts to testing (1), with θ_0 replaced by θ. The test is

$$\phi_n(\theta) = 1\{T_n(\theta) > c_n(\theta, 1-\alpha)\} \tag{3}$$

where $T_n(\theta)$ is a test statistic, $c_n(\theta, 1-\alpha)$ is a simulated critical value, and α is the nominal level of the test.

We can then invert the test to construct a confidence set (CS) for θ_0. The CS is defined as

$$\text{CS}_n(1-\alpha) = \{\theta \in \Theta : \phi_n(\theta) = 0\}$$

The standard way to compute this CS is to consider many grid points in Θ, compute $\phi_n(\theta)$ at each grid point, and collect the values for which $\phi_n(\theta) = 0$.[2]

In some cases, it is of interest to test a null hypothesis of the form

$$\begin{aligned} &E\{m_j(\mathbf{W}_i)|\mathbf{X}_i\} \geq 0 \text{ for } j = 1, \ldots, p \\ &E\{m_j(\mathbf{W}_i)|\mathbf{X}_i\} = 0 \text{ for } j = p+1, \ldots, k, \text{ almost surely} \end{aligned} \tag{4}$$

which does not depend on a parameter θ, where $\mathbf{m}(\cdot) := \{m_1(\cdot), \ldots, m_k(\cdot)\}'$ is a vector of known functions of the observables and $\mathbf{W}_i$, $\mathbf{X}_i$, k, and p are as above. For example, this arises when one is interested in the sign of a conditional average treatment effect or the shape of a dose–response function, as discussed in examples 2.1 and 2.2 in Lee, Song, and Whang (2013). Testing the hypothesis in (4) is the same as testing (2) in the model in (1). One just replaces $m(\cdot, \theta)$ with $m(\cdot)$; consequently, the test in (3) does not depend on θ.

Now, we briefly describe a conditional average treatment-effect example of the testing problem in (4). Let D be a binary treatment variable, which equals 1 if treated and 0 if untreated. Let Y be the outcome variable. In the potential-outcome notation, $Y = DY(1) + (1-D)Y(0)$, where $Y(1)$ is the treated outcome observable only if $D = 1$ and $Y(0)$ is the untreated outcome observable only if $D = 0$. Let $\mathbf{X}$ be a vector of covariates. Suppose that D is randomly assigned, with each individual receiving treatment with a known probability p. We can then express the average conditional treatment effect, given $\mathbf{X}$, as follows:

$$E\{Y(1) - Y(0)|\mathbf{X}\} = E\left\{ \frac{DY}{p} - \frac{(1-D)Y}{1-p} \middle| \mathbf{X} \right\}$$

2. You can combine `cmi_test` with any grid-search algorithm to complete this task. Usually, this grid search is computationally costly when the dimension of the parameter space is large. One way to circumvent the computational burden is applying a response surface algorithm for global optimization introduced by Kaido, Molinari, and Stoye (2016). You can implement this algorithm using a MATLAB toolbox called "DACE", which is publicly available. So far, we are not aware of whether this algorithm can be used with Stata commands. For details of "DACE", see http://www2.imm.dtu.dk/projects/dace/.

Suppose the researcher wants to test whether the average treatment effect is negative for individuals at all $\mathbf{X}$ values. In the framework above, this problem can be written as testing the null hypothesis

$$H_0\colon E\{m_j(\mathbf{W})|\mathbf{X}\} \geq 0$$

where $j = 1, \mathbf{W} = (Y, D, \mathbf{X})$, and $m_1(\mathbf{W}) = -\{(DY)/p\} + [\{(1-D)Y\}/(1-p)]$.

2.2 Confidence intervals based on intersection bounds

The `cmi_interval` command computes the CS for the special, but popular case that the parameter θ_0 is one dimensional, and the moment inequalities provide intersection bounds for this parameter, so the CS of θ_0 is an interval. The `cmi_interval` command combines a one-dimensional grid-search algorithm with `cmi_test` to compute this interval. Specifically, the command applies when the conditional moment inequality model is

$$\begin{aligned} E\{\rho_{u,j}(\mathbf{W}_i) - \theta_0|\mathbf{X}_i\} &\geq 0 \text{ for } j = 1, \ldots, k_u \\ E\{\theta_0 - \rho_{\ell,j}(\mathbf{W}_i)|\mathbf{X}_i\} &\geq 0 \text{ for } j = 1, \ldots, k_\ell \end{aligned} \tag{5}$$

where θ_0 is a real-valued parameter and $\rho_{u,j}(\cdot)$ and $\rho_{\ell,j}(\cdot)$ are known functions of the observables. The upper bounds for θ_0 are $E\{\rho_{u,1}(\mathbf{W}_i)|\mathbf{X}_i\}, \ldots, E\{\rho_{u,k_u}(\mathbf{W}_i)|\mathbf{X}_i\}$, and the lower bounds are $E\{\rho_{\ell,1}(\mathbf{W}_i)|\mathbf{X}_i\}, \ldots, E\{\rho_{\ell,k_\ell}(\mathbf{W}_i)|\mathbf{X}_i\}$. It is easy to see that (5) is a special case of (1), with $p = k_u + k_\ell$, $k = p$, and

$$m_j(\mathbf{W}_i, \theta_0) = \begin{cases} \rho_{u,j}(\mathbf{W}_i) - \theta_0 & \text{for } j = 1, \ldots, k_u \\ \theta_0 - \rho_{\ell,j-k_u}(\mathbf{W}_i) & \text{for } j = k_u + 1, \ldots, k_u + k_\ell \end{cases}$$

The `cmi_interval` command allows one or more upper bounds, $\rho_{u,j}(\mathbf{W}_i)$, to be identical to some lower bounds, $\rho_{\ell,j'}(\mathbf{W}_i)$.

We use a censored data example like that in Andrews and Shi (2014) to illustrate the model in (5). Let D be a binary variable indicating data censorship and $\mathbf{X}$ be a covariate vector. Let Y^* be a variable subject to censoring; that is, we observe it only when $D = 1$. Let θ_0 denote the conditional cumulative distribution function (c.d.f.) of Y^* given $\mathbf{X}$ evaluated at a certain point, y_0. Then, θ_0 is bounded by the inequalities in (5) with $k_u = k_\ell = 1$, and

$$\rho_{u,1}(\mathbf{W}) = 1\{Y \leq y_0, D = 1\} + 1\{D = 0\} \tag{6}$$

$$\rho_{\ell,1}(\mathbf{W}) = 1\{Y_i \leq y_0, D_i = 1\} \tag{7}$$

We illustrate the implementation of both commands using this example in section 6 below.

3 Detailed procedures

In this section, we describe the detailed procedures from Andrews and Shi (2013) that the commands implement. Section 3.1 summarizes the steps in section 9 of Andrews and

Shi (2013), and section 3.2 describes the algorithm to compute the confidence interval for the intersection bound model in (5).

3.1 Basic testing procedure

Test statistics

Now we describe the testing procedure that `cmi_test` implements. Although we focus on testing the hypothesis in (2) for the model in (1), we can also apply the procedure to the hypothesis in (4). Following Andrews and Shi (2013), we transform the conditional moment restrictions in (1) into unconditional moment restrictions before using them to construct the test statistic. The instrumental functions are functions of the instrumental variables, $\mathbf{X}_i$. The ones we use are countable hypercubes on standardized $\mathbf{X}_i$. We define the standardized $\mathbf{X}_i$ variables first. The standardized $\mathbf{X}_i$, denoted $\mathbf{X}_i^o$, is

$$\mathbf{X}_i^o = \Phi\left\{\widehat{\Sigma}_{X,n}^{-1/2}\left(\mathbf{X}_i - \overline{\mathbf{X}}_n\right)\right\}$$

where $\overline{\mathbf{X}}_n = n^{-1}\sum_{i=1}^n \mathbf{X}_i \in R^{d_x}$, $\widehat{\Sigma}_{X,n} = n^{-1}\sum_{i=1}^n(\mathbf{X}_i - \overline{\mathbf{X}}_n)(\mathbf{X}_i - \overline{\mathbf{X}}_n)'$, and $\Phi(x) = \{\Phi(x_1), \ldots, \Phi(x_{d_x})\}'$. The function $\Phi(\cdot)$ denotes the standard normal c.d.f. and $x = (x_1, \ldots, x_{d_x})'$.

The instrumental functions are

$$g_{a,r}(\mathbf{X}_i^o) = 1\{\mathbf{X}_i^o \in \times_{u=1}^{d_x}[(a_u - 1)/(2r), a_u/(2r)]\} \tag{8}$$

where $a = (a_1, \ldots, a_{d_x})' \in \{1, 2, \ldots, 2r\}^{d_x}$ and $r = 1, 2, 3, \ldots$. In the implementation, we consider only $r = 1, 2, \ldots, r_{1,n}$ for a positive integer $r_{1,n}$. The `cmi_test` command uses $\lfloor n^{1/(2d_x)}/2 \rfloor$ as $r_{1,n}$ by default and allows the user to opt for a different positive integer.

Next, we compute the sample average of the unconditional moment functions for each $j = 1, \ldots, k$ and each (a, r) described above. For notational simplicity, in the discussion below, we suppress the possible dependence of $m_j(\mathbf{W}_i, \theta)$ on θ throughout. We have

$$\overline{m}_{n,j}(g_{a,r}) = n^{-1}\sum_{i=1}^n m_j(\mathbf{W}_i) g_{a,r}(\mathbf{X}_i^o)$$

We also compute the sample variance, $\widehat{\sigma}_{n,j}^2(g_{a,r})$, of $m_j(\mathbf{W}_i) g_{a,r}(\mathbf{X}_i^o)$. Because $\widehat{\sigma}_{n,j}^2(g_{a,r})$ could be zero for some (a, r), we also compute the variance, $\widehat{\sigma}_{n,j}^2$, of the conditional moment function, $m_j(\mathbf{W}_i)$, to regularize $\widehat{\sigma}_{n,j}^2(g_{a,r})$. We use the regularized variance

$$\overline{\sigma}_{n,j}^2(g_{a,r}) = \widehat{\sigma}_{n,j}^2(g_{a,r}) + \varepsilon\widehat{\sigma}_{n,j}^2$$

in the test statistic. The regularization parameter ε is 0.05 in `cmi_test` by default, and the user is allowed to set it to a different small positive number by specifying the `epsilon()` option. We then construct the test statistic that combines the information in all of these sample moments. After constructing the test statistic, we construct the critical value $c_n(1-\alpha)$. There are two versions of the critical value. One is based on the asymptotic approximation, and the other is based on the bootstrap. The command implements the former by default. The bootstrap version can be activated by selecting the `boot` option. The test statistic and critical values are described next.

By default, `cmi_test` uses summation (sum) to aggregate over j for each (a,r) and uses Cramér–von Mises-type aggregation over (a,r), which yields the following test statistic,

$$T_n = n \sum_{r=1}^{r_{1,n}} \frac{\sum_{a\in\{1,\ldots,2r\}^{d_x}} \left(\sum_{j=1}^{p} \left[\frac{\overline{m}_{n,j}(g_{a,r})}{\overline{\sigma}_{n,j}(g_{a,r})} \right]_-^2 + \sum_{j=p+1}^{k} \left(\frac{\overline{m}_{n,j}(g_{a,r})}{\overline{\sigma}_{n,j}(g_{a,r})} \right)^2 \right)}{(r^2+100)(2r)^{d_x}} \tag{9}$$

where the negative part function $[x]_- = \max\{0,-x\}$. By specifying the `sfunc()` and `ks` options, `cmi_test` allows the user to choose from the Cramér–von Mises max statistic, the Kolmogorov–Smirnov sum statistic, or the Kolmogorov max statistic.[3] Choosing max instead of sum replaces the expression in the large brackets in the numerator with

$$\max\left\{ \max_{j=1,2,\ldots,p} \left[\frac{\overline{m}_{n,j}(g_{a,r})}{\overline{\sigma}_{n,j}(g_{a,r})} \right]_-^2, \max_{j=p+1,p+2,\ldots,k} \left(\frac{\overline{m}_{n,j}(g_{a,r})}{\overline{\sigma}_{n,j}(g_{a,r})} \right)^2 \right\}$$

Choosing the Kolmogorov–Smirnov sum statistic instead of the Cramér–von Mises max statistic replaces $\sum_{r=1}^{r_{1,n}}(\sum_{a\in\{1,\ldots,2r\}^{d_x}})/\{(r^2+100)(2r)^{d_x}\}$ in (9) with

$$\max_{(a,r):a\in\{1,\ldots,2r\}^{d_x},r=1,\ldots,r_{1,n}}$$

Asymptotic critical values

The asymptotic approximation version of the critical value is a simulated quantile of a statistic (denoted by T_n^{Asy}), defined the same as T_n, except with $\overline{m}_{n,j}(g_{a,r})$ replaced by

$$n^{-1/2}\{\nu_{n,j}(g_{a,r}) + \varphi_{n,j}(g_{a,r})\}$$

where $\{\nu_{n,j}(g_{a,r})\}_{j,a,r}$ is a Gaussian random vector that approximates the distribution of $(n^{1/2}\,[\overline{m}_{n,j}(g_{a,r}) - E\{m_j(\mathbf{W}_i)g_{a,r}(\mathbf{X}_i^o)\}])_{j,a,r}$ and $\varphi_{n,j}(g_{a,r})$ is the generalized moment selection (GMS) function that approximates $n^{1/2}E\{m_j(\mathbf{W}_i)g_{a,r}(\mathbf{X}_i^o)\}$ and selects the binding moment restrictions.

Specifically, the command simultaneously draws the $k\sum_{r=1}^{r_{1,n}}(2r)^{d_x}$ dimensional vector $\{\nu_{n,j}(g_{a,r})\}_{j=1,\ldots,k,a\in\{1,\ldots,2r\}^{d_r},r=1,\ldots,r_{1,n}}$ from a multivariate normal distribution. The multivariate normal distribution has mean zero, and its variance–covariance matrix is the empirical variance–covariance matrix of

$$\{m_j(\mathbf{W}_i)g_{a,r}(\mathbf{X}_i^o)\}_{j=1,\ldots,k,a\in\{1,\ldots,2r\}^{d_r},r=1,\ldots,r_{1,n}}$$

3. The commands do not incorporate the quasilikelihood-ratio statistic discussed in Andrews and Shi (2013), because that statistic requires carrying out a quadratic optimization operation many times. We are not aware of a fast quadratic optimization routine in Stata.

Many draws are taken, and each is used to compute a draw of T_n^{Asy}. The command then computes the empirical $1-\alpha$ quantile of the sample of T_n^{Asy} values obtained. This quantile is $c_n(1-\alpha)$. By default, the number of draws is set to 5,001, and the seed of the random-number generator is set to 10,000. You can change these by specifying the `rep()` and `seed()` options.

The GMS function, $\varphi_{n,j}(g_{a,r})$, is

$$\varphi_{n,j}(g_{a,r}) = \begin{cases} \widehat{\sigma}_{n,j} B_n & \text{if } \kappa_n^{-1} n^{1/2} \overline{m}_{n,j}(g_{a,r})/\overline{\sigma}_{n,j}(g_{a,r}) > 1 \\ 0 & \text{otherwise} \end{cases}$$

where B_n and κ_n are two user-chosen tuning parameters that, in the asymptotic thought experiment, should satisfy $\kappa_n \to \infty$, $\kappa_n/n^{1/2} \to 0$, $B_n \to \infty$, and $B_n/\kappa_n \to 0$ as $n \to \infty$. By default, the command uses the recommended choices from Andrews and Shi (2013): $\kappa_n = \sqrt{0.3 \log n}$ and $B_n = \sqrt{0.4 \log n / \log \log n}$.

Bootstrap critical values

The bootstrap version of the critical value is a simulated quantile of a statistic (denoted by T_n^{Boot}) defined in the same way as T_n, except with $\overline{m}_{n,j}(g_{a,r})/\overline{\sigma}_{n,j}(g_{a,r})$ replaced by

$$\frac{n^{-1/2}\{\nu_{n,j}^{\text{Boot}}(g_{a,r}) + \varphi_{n,j}(g_{a,r})\}}{\overline{\sigma}_{n,j}^{\text{Boot}}(g_{a,r})}$$

$\{\nu_{n,j}^{\text{Boot}}(g_{a,r})\}_{j,a,r}$ is a bootstrap approximation of $(n^{1/2}[\overline{m}_{n,j}(g_{a,r}) - E\{m_j(\mathbf{W}_i) g_{a,r}(\mathbf{X}_i^o)\}])_{j,a,r}$, $\varphi_{n,j}(g_{a,r})$ is the GMS function described above, and $\overline{\sigma}_{n,j}^{\text{Boot}}(g_{a,r})$ is a bootstrap version of $\overline{\sigma}_{n,j}(g_{a,r})$.

Specifically, the command first randomly draws n observations with replacement from the sample $\{\mathbf{W}_i\}_{i=1}^n$. These n observations, denoted $\{\mathbf{W}_i^*\}_{i=1}^n$, form a bootstrap sample, which is used to compute one draw of $\nu_{n,j}^{\text{Boot}}(g_{a,r})$ and $\overline{\sigma}_{n,j}^{\text{Boot}}(g_{a,r})$. The draw of $\nu_{n,j}^{\text{Boot}}(g_{a,r})$ is

$$n^{1/2}\left\{\overline{m}_{n,j}^*(g_{a,r}) - \overline{m}_{n,j}(g_{a,r})\right\}$$

where we compute $\overline{m}_{n,j}^*(g_{a,r})$ using the same procedure as that for $\overline{m}_{n,j}(g_{a,r})$, except with $\{\mathbf{W}_i^*\}$ (and its subvector $\{\mathbf{X}_i^*\}$) replaced by $\{\mathbf{W}_i\}$ (and its subvector $\{\mathbf{X}_i\}$). We compute the draw of $\overline{\sigma}_{n,j}^{\text{Boot}}(g_{a,r})$ using the same procedure as that for $\overline{\sigma}_{n,j}(g_{a,r})$, except with $\{\mathbf{W}_i^*\}$ (and its subvector $\{\mathbf{X}_i^*\}$) replacing $\{\mathbf{W}_i\}$ (and its subvector $\{\mathbf{X}_i\}$). We then use these to compute one draw of T_n^{Boot}. By repeating the process, we take many T_n^{Boot} draws, with $c_n(1-\alpha)$ defined as the $1-\alpha$ empirical quantile of these draws. By default, the number of draws is set to 5,001, and the seed of the random-number generator is set to 10,000. You can change these by specifying the `rep()` and `seed()` options.

3.2 Confidence interval construction for intersection bound models

In this section, we describe the algorithm to construct a confidence interval for the one-dimensional parameter in the model in (5).

One-sided bound

If either k_u or k_ℓ is zero, the model gives a one-sided bound for the parameter. In this case, the command uses the following algorithm consisting of iterative steps, where step (-1) is the preparation step, and for $i \geq 0$, step (i) finds the confidence interval bounds for θ_0 up to the ith digit after the decimal point.

Step (-1). First, we set a preliminary lower (upper) bound of the confidence interval:

$$\begin{aligned}\widehat{\theta}_{\text{lb,pre}} &= \min_{1\leq j\leq k_l} \min_{i\leq n} \rho_{l,j}(\mathbf{W}_i)\\ \widehat{\theta}_{\text{ub,pre}} &= \max_{1\leq j\leq k_u} \max_{i\leq n} \rho_{u,j}(\mathbf{W}_i)\end{aligned}$$

In addition, we define two auxiliary statistics:

$$\begin{aligned}\widehat{\theta}_{\text{lb,bound}} &= \max_{1\leq j\leq k_l} \max_{i\leq n} \rho_{l,j}(\mathbf{W}_i)\\ \widehat{\theta}_{\text{ub,bound}} &= \min_{1\leq j\leq k_u} \min_{i\leq n} \rho_{u,j}(\mathbf{W}_i)\end{aligned}$$

Note that $\widehat{\theta}_{\text{lb,pre}}$ ($\widehat{\theta}_{\text{ub,pre}}$) is a preliminary conservative lower (upper) bound for the confidence interval. Meanwhile, $\widehat{\theta}_{\text{lb,bound}}$ ($\widehat{\theta}_{\text{ub,bound}}$) is trivially contained in the one-sided confidence interval and thus is greater (smaller) than the lower (upper) bound. The following steps take advantage of these conservative bounds.

We explain the method for deriving the lower bound here. The upper bound method is analogous.

Step (0). If the distance between $\lfloor\widehat{\theta}_{\text{lb,pre}}\rfloor$ and $\lceil\widehat{\theta}_{\text{lb,bound}}\rceil$ is 1, skip the current step, let $\widehat{\theta}_{\text{lb},0} = \lfloor\widehat{\theta}_{\text{lb,pre}}\rfloor$, and move to the next step. Otherwise, consider grid points on $[\lfloor\widehat{\theta}_{\text{lb,pre}}\rfloor, \lceil\widehat{\theta}_{\text{lb,bound}}\rceil]$ with distance between adjacent grid points being $d_0 = \lfloor\max\{(\lceil\widehat{\theta}_{\text{lb,bound}}\rceil - \lfloor\widehat{\theta}_{\text{lb,pre}}\rfloor)/20, 1\}\rfloor$. Apply `cmi_test` for θ_0 being each of these grid points. Record the largest grid point rejected by the test as $\widehat{\theta}_{\text{lb},0}$, and consider grid points on $[\widehat{\theta}_{\text{lb},0}, \widehat{\theta}_{\text{lb},0} + d_0]$ with the updated spacing between grids, $d_1 = \lfloor\max(d_0/2, 1)\rfloor$. Repeat until the distance equals 1. Record the smallest θ_0 value not rejected and subtract 1. Let the resulting number be $\widehat{\theta}_{\text{lb},0}$.

Step (1). Apply `cmi_test` for θ_0 being each of the points $\widehat{\theta}_{\text{lb},0}, \widehat{\theta}_{\text{lb},0}+0.1, \ldots, \widehat{\theta}_{\text{lb},0}+0.9$. Record the smallest point not rejected and subtract 0.1. Let the resulting number be $\widehat{\theta}_{\text{lb},1}$.

...

Step (i+1). Apply `cmi_test` for θ_0 being each of the points $\widehat{\theta}_{\text{lb},i}, \widehat{\theta}_{\text{lb},i} + 10^{-(i+1)}, \ldots, \widehat{\theta}_{\text{lb},i} + 9 \times 10^{-(i+1)}$. Record the smallest point not rejected and subtract $10^{-(i+1)}$. Let the resulting number be $\widehat{\theta}_{\text{lb},i+1}$.

By default, the command iterates this algorithm up to the thousandth place [that is, step (3)]. One can choose the number of iterations (that is, the accuracy of the confidence interval) by specifying the `deci()` option.

Two-sided bound

When $k_\ell > 0$ and $k_u > 0$, the model gives two-sided bounds for the parameter θ_0. In this case, we first obtain two one-sided confidence intervals, each of confidence level $1-\alpha/2$. The two one-sided confidence intervals separately use the upper-bound and the lower-bound moment inequalities. The algorithm then forms a preliminary two-sided confidence interval (of nominal level $1-\alpha$) by intersecting the two one-sided bounds. If the two one-sided bounds do not intersect, `cmi_interval` terminates and returns an empty set. This implies that the model is rejected at the specified confidence level α (the default is 95%).

Let $\widehat{\theta}_{\mathrm{lb},-1}$ and $\widehat{\theta}_{\mathrm{ub},-1}$ be the lower and upper bounds of the crude interval just specified. We then obtain the Andrews and Shi (2013) confidence interval by applying the following algorithm.

Step (0). Check the length of the crude interval. If it is less than 2, then skip step (0), let $\widehat{\theta}_{\mathrm{lb},0} = \widehat{\theta}_{\mathrm{lb},-1}, \widehat{\theta}_{\mathrm{ub},0} = \widehat{\theta}_{\mathrm{ub},-1}$, and move to the next step. Otherwise, set $d_0 = \lfloor \max\{(\lceil \widehat{\theta}_{\mathrm{ub},-1} \rceil - \lfloor \widehat{\theta}_{\mathrm{lb},-1} \rfloor)/20, 1\} \rfloor$, and apply `cmi_test` using all inequalities for each of the evenly spaced grid points (including the endpoints) with spacing d_0 on $[\lfloor \widehat{\theta}_{\mathrm{lb},-1} \rfloor, \lceil \widehat{\theta}_{\mathrm{ub},-1} \rceil]$.

Case 1: If at least one grid point is not rejected, let θ_{lb,d_0} and θ_{ub,d_0} denote the smallest and the largest nonrejected points, respectively.

Case 2: If all points are rejected, find the grid point with the largest p-value (denoted by $\theta_{\mathrm{high},d_0}$) and let $\theta_{\mathrm{lb},d_0} = \theta_{\mathrm{ub},d_0} = \theta_{\mathrm{high},d_0}$.

For both cases, let $d_1 = \lfloor \max(d_0/2, 1) \rfloor$. Consider evenly spaced grid points (including endpoints) with spacing d_1 on $[\theta_{\mathrm{lb},d_0} - d_0, \theta_{\mathrm{lb},d_0}]$ and also those on $[\theta_{\mathrm{ub},d_0}, \theta_{\mathrm{ub},d_0} + d_0]$. Apply `cmi_test` using all inequalities for each of these grids. Repeat the checks in case 1 and case 2 above, and define θ_{lb,d_1} and θ_{ub,d_1} analogously to θ_{lb,d_0} and θ_{ub,d_0}, respectively. Iterate this step until $d_J = 1$, then let $[\widehat{\theta}_{\mathrm{lb},0}, \widehat{\theta}_{\mathrm{ub},0}] = [\theta_{\mathrm{lb},d_J} - d_J, \theta_{\mathrm{ub},d_J} + d_J]$. This interval is the Andrews and Shi (2013) confidence interval accurate up to the integer level. If you desire higher accuracy, move on to the next step.

. . .

Step (i+1). If $\widehat{\theta}_{\mathrm{ub},i} - \widehat{\theta}_{\mathrm{lb},i} \leq 2 \times 10^{-(i+1)}$, let $\widehat{\theta}_{\mathrm{lb},i+1} = \widehat{\theta}_{\mathrm{lb},i}$, $\widehat{\theta}_{\mathrm{ub},i+1} = \widehat{\theta}_{\mathrm{ub},i}$ and move to the next step. Otherwise, consider evenly spaced grid points with spacing $10^{-(i+1)}$ on the intervals $[\widehat{\theta}_{\mathrm{lb}}, \widehat{\theta}_{\mathrm{lb}} + 10^{-i}]$ and $[\widehat{\theta}_{\mathrm{ub}} - 10^{-i}, \widehat{\theta}_{\mathrm{ub}}]$ (including endpoints). Apply `cmi_test` for θ_0 being each of these grid points.

Case 1: If at least one point is not rejected, let $\theta_{\text{lb},j}$ and $\theta_{\text{ub},j}$ denote the smallest and the largest such point, respectively.

Case 2: If all points are rejected, find the point with the largest p-value (denoted by $\theta_{\text{high},i+1}$), and let $\theta_{\text{lb},i+1} = \theta_{\text{ub},i+1} = \theta_{\text{high},i+1}$.

Let $[\widehat{\theta}_{\text{lb},i+1}, \widehat{\theta}_{\text{ub},i+1}] = [\theta_{\text{lb},i+1} - 10^{-i-1}, \theta_{\text{ub},i+1} + 10^{-i-1}]$. This interval is the Andrews and Shi (2013) confidence interval with accuracy up to 10^{-i-1}. If you desire higher accuracy, move on to the next step.

Iterate until the desired accuracy is reached. `cmi_interval` iterates this algorithm up to the thousandth place by default. The user can set the accuracy level differently with `deci()`.

Remark. If the confidence interval is narrower than the smallest grid, for example, 10^{-k} (10^{-3} in the default setup), `cmi_interval` finds a grid point with the highest p-value, $\widehat{\theta}_p$, and returns $[\widehat{\theta}_p - 10^{-k}, \widehat{\theta}_p + 10^{-k}]$ as the confidence interval. One may adjust the last digit of the confidence interval with `deci()` or by rescaling $m_{u,j}(\mathbf{W}_i)$ and $m_{\ell,j}(\mathbf{W}_i)$ by multiplying all of them by an appropriate power of 10 to get a more accurate confidence interval.

4 The cmi_test command

4.1 Syntax

The syntax of `cmi_test` is as follows:

`cmi_test` ([*cmi_vars*]) ([*cme_vars*]) *indepvars* [*if*] [*in*] [`, rnum(`*#*`) hd boot`
`ks sfunc(`*#*`) epsilon(`*real*`) kap(`*real*`) bn(`*real*`) rep(`*#*`) seed(`*#*`) simul`]

4.2 Description

`cmi_test` implements the test described in section 3.1 for the hypothesis in (2) and the model in (1) [or the hypothesis in (4)]. To use this command, one first generates variables that equal $m_1(\mathbf{W}_i, \theta), \ldots, m_k(\mathbf{W}_i, \theta)$ for observations $i = 1, \ldots, n$ [or $m_1(\mathbf{W}_i), \ldots, m_k(\mathbf{W}_i)$ for observations $i = 1, \ldots, n$]. The first p are *cmi_vars*, and the next $k - p$ are *cme_vars*. The command allows *cmi_vars* or *cme_vars* to be empty. The variables in $\mathbf{X}_i$ are *indepvars*.

As described in section 3.1, `cmi_test` uses countable hypercubes as the collection of instrumental functions. They are constructed according to (8) above by default. That choice is fine when the number of *indepvars* is three or fewer. When the dimension of *indepvars* is greater than three, the number of cubes may be too large, resulting in long computation time. The command allows an alternative method for high-dimensional independent variables. The user can select the `hd` option to choose this method. This option implements the method described in the last paragraph of section 9 of Andrews and Shi (2013).

4.3 Options

`rnum(`*#*`)` sets a scalar indicating the minimum side-edge lengths. The default is the smallest integer greater than $n^{d_x/2}/2$, where d_x is the dimension of *indepvars*.

`hd` uses an alternative method for high-dimensional independent variables. This option is designed for three or more covariates; see the previous subsection for details.

`boot` lets the user turn on the bootstrap option. If the user does not specify this option, the command computes the critical value based on a Gaussian asymptotic approximation.

`ks` uses the Kolmogorov–Smirnov-type statistic. The default is the Cramér–von Mises-type statistic.

`sfunc(`*#*`)` sets the function S to specify the form of the test statistic. `sfunc(1)` yields the modified method of moments or sum function, and `sfunc(3)` yields the max function. The default is `sfunc(1)`.

`epsilon(`*real*`)` sets the regularization parameter ε for the sample variances. The default is `epsilon(0.05)`.

`kap(`*real*`)` and `bn(`*real*`)` are two tuning parameters in the data-dependent GMS function $\varphi_n(g_{a,r})$. The default for the former is $(0.3 \log n)^{1/2}$ and for the latter is $\{(0.4 \log n)/(\log \log n)\}^{1/2}$.

`rep(`*#*`)` sets the number of repetitions for the critical value simulations. The default is `rep(5001)`.

`seed(`*#*`)` sets the random-number seed for the critical value simulations. The default is `seed(10000)`.

`simul` lets the user choose to leave the seed number for the critical value simulations unset. Use this option when the command is inside a Monte Carlo simulation loop to not interfere with the random-number generation process set for the Monte Carlo simulation exercise.

4.4 Stored results

`cmi_test` stores the following in `r()`:

Scalars

`r(N)`	number of observations
`r(stat)`	test statistic
`r(pval)`	p-value
`r(cv01)`	critical value for the 1% significance level
`r(cv05)`	critical value for the 5% significance level
`r(cv10)`	critical value for the 10% significance level
`r(kappa)`	tuning parameter κ_n
`r(B)`	tuning parameter B_n
`r(epsilon)`	tuning parameter ϵ
`r(rep_cv)`	repetitions for critical values
`r(a_obs)`	average number of observations in the smallest cubes
`r(r_n)`	index for minimum side-edge lengths
`r(ncube)`	number of cubes

Macros

`r(cmd)`	`cmi_test`
`r(title)`	title in output
`r(m_ineq)`	*varlist* for conditional moment inequalities, if any
`r(m_eq)`	*varlist* for conditional moment equalities, if any
`r(x)`	*varlist* for instrumental variables

5 The cmi_interval command

5.1 Syntax

The syntax of `cmi_interval` is as follows:

```
cmi_interval ([lower_bound_vars]) ([upper_bound_vars]) indepvars [if] [in]
   [, level(real) deci(#) rnum(#) hd boot ks sfunc(#) epsilon(real)
   kap(real) bn(real) rep(#) seed(#) simul]
```

5.2 Description

`cmi_interval` constructs the confidence interval for the parameter in (5) by inverting `cmi_test`. The *upper_bound_vars* are $\rho_{u,1}(\mathbf{W}_i),\ldots,\rho_{u,k_u}(\mathbf{W}_i)$. The *lower_bound_vars* are $\rho_{\ell,1}(\mathbf{W}_i),\ldots,\rho_{\ell,k_\ell}(\mathbf{W}_i)$. The *indepvars* are the elements of $\mathbf{X}_i$.

5.3 Options

`cmi_interval` accepts all the options that `cmi_test` does. Two additional options are available to `cmi_interval`, which are the following:

`level(`*real*`)` sets the confidence level $1-\alpha$, where $1-\alpha$ is the nominal confidence level. The default is `level(0.95)`.

`deci(#)` sets the accuracy of the confidence interval bounds as measured by the number of digits after the decimal point.

5.4 Stored results

`cmi_interval` stores the following in `r()`:

Scalars	
`r(N)`	number of observations
`r(lbound)`	estimated lower bound (if any)
`r(ubound)`	estimated upper bound (if any)
`r(level)`	confidence level
`r(ncube)`	number of cubes
`r(kappa)`	tuning parameter κ_n
`r(epsilon)`	tuning parameter ϵ
`r(rep_cv)`	repetitions for critical values
`r(a_obs)`	average number of observations in the smallest cubes
`r(r_n)`	index for minimum side-edge lengths
`r(B)`	tuning parameter B_n
Macros	
`r(cmd)`	`cmi_interval`
`r(title)`	title in output
`r(lbvar)`	*varlist* for conditional moment inequalities for the lower bound, if any
`r(ubvar)`	*varlist* for conditional moment inequalities for the upper bound, if any
`r(x)`	*varlist* for instrumental variables

6 Examples

In this section, we provide an example of estimating a conditional distribution with censored data, which was introduced earlier in section 2.1. We use the data for male employees who are not self-employed from the 15th round (year 2011) of the National Longitudinal Survey of Youth 1997. From that dataset, we take the log hourly dollar wage (`Y`), the dummy for college enrollment (`D`), the year of education of father (`X1`), and the year of education of mother (`X2`). The number of observations is 2,054.

Let Y_i^* be the natural logarithm of the potential wage after college enrollment. This variable is observed only for those who actually enrolled in a college. Suppose that the parameter of interest is $\theta_0 \equiv F_{Y^*}(y_0)$, that is, the c.d.f. Y_i^* evaluated at y_0. The parameter θ_0 is then bounded by (5), with the bounding moment functions defined in (6) and (7). See Andrews and Shi (2014) for details.

For the rest of the example, define $\theta_0 = F_{Y^*}\{\log(20)\}$. In other words, θ_0 is the percentage of the subpopulation (currently working, not self-employed male) whose expected hourly wage is lower than \$20 if he had enrolled in a college. We create two variables defined by $1\{Y_i \leq y_0, D_i = 1\}$ and $1\{Y_i \leq y_0, D_i = 1\} + 1\{D_i = 0\}$:

```
. use cmitest
. local y0 = log(20)
. generate lbound = (Y < `y0´) * D
. generate ubound = (Y < `y0´) * D + 1 - D
```

6.1 cmi_test

Suppose that the research question is whether 0.5 is the value of θ_0. That is, we would like to test

$$H_0 : F_{Y^*}\{\log(20)\} = 0.5 \tag{10}$$

Then, the researcher creates two conditional moment inequalities [(6) and (7)] by using the following commands:

```
. local theta0 = 0.5
. generate CMI1 = `theta0´ - lbound
. generate CMI2 = ubound - `theta0´
```

`cmi_test` results are

```
. cmi_test (CMI1 CMI2) ( ) X1 X2
Conditional Moment Inequalities Test                      Number of obs : 2054
--------------------------------------------------------------------------------
<Variables>
Conditional Moment Inequalities : CMI1 CMI2
No Conditional Moment Equality
Instruments : X1 X2
--------------------------------------------------------------------------------
<Methods>
Countable Hyper Cubes
Asymptotic Critical Value
Cramer-von Mises-type statistic / Sum function
--------------------------------------------------------------------------------
<Results>
Test Statistic      : 0.0331
Critical Value (1%) : 0.2347
               (5%) : 0.1698
              (10%) : 0.1446
p-value             : 0.9882
. cmi_test (CMI1 CMI2) ( ) X1 X2, ks
Conditional Moment Inequalities Test                      Number of obs : 2054
--------------------------------------------------------------------------------
<Variables>
Conditional Moment Inequalities : CMI1 CMI2
No Conditional Moment Equality
Instruments : X1 X2
--------------------------------------------------------------------------------
<Methods>
Countable Hyper Cubes
Asymptotic Critical Value
Kolmogorov-Smirnov-type statistic / Sum function
--------------------------------------------------------------------------------
<Results>
Test Statistic      : 0.8413
Critical Value (1%) : 5.7807
               (5%) : 4.1274
              (10%) : 3.3612
p-value             : 0.9438
```

```
. cmi_test (CMI1 CMI2) ( ) X1 X2, sfunc(3) boot
Conditional Moment Inequalities Test                    Number of obs : 2054
--------------------------------------------------------------------------------
<Variables>
Conditional Moment Inequalities : CMI1 CMI2
No Conditional Moment Equality
Instruments : X1 X2
--------------------------------------------------------------------------------
<Methods>
Countable Hyper Cubes
Bootstrap Critical Value
Cramer-von Mises-type statistic / Max function
--------------------------------------------------------------------------------
<Results>
Test Statistic      : 0.0331
Critical Value (1%) : 0.2687
               (5%) : 0.1840
              (10%) : 0.1554
p-value             : 0.9960
```

The first result shows the `cmi_test` outcome with default options. The second result uses the Kolmogorov–Smirnov-type statistic. The last result uses the max function in the test statistic and uses the bootstrapped critical value. All three versions of the test yield high p-values, indicating that 0.5 is not rejected even at significance level 10%.

Note that the example given here is for an inequalities-only model. If a model contains conditional moment equalities, you should position variables representing those equalities in the second parenthesis of the syntax.

6.2 cmi_interval

Now we compute a confidence interval for θ_0. In this example, the `lbound` and `ubound` variables represent *lower_bound_vars* and *upper_bound_vars*, respectively. `cmi_interval` returns the following results:

```
. cmi_interval (lbound) (ubound) X1 X2
Conditional Moment Inequalities Interval                Number of obs : 2054
--------------------------------------------------------------------------------
<Variables>
Variables for the Lower Bound : lbound
Variables for the Upper Bound : ubound
Instruments : X1 X2
--------------------------------------------------------------------------------
<Methods>
Countable Hyper Cubes
Asymptotic Critical Value
Cramer-von Mises-type statistic / Sum function
--------------------------------------------------------------------------------
<Results>
.% confidence interval is:
( 0.413 , 0.621 )
```

```
. cmi_interval (lbound) (ubound) X1 X2, sfunc(3)
Conditional Moment Inequalities Interval              Number of obs : 2054
──────────────────────────────────────────────────────────────────────────────
<Variables>
Variables for the Lower Bound : lbound
Variables for the Upper Bound : ubound
Instruments : X1 X2
──────────────────────────────────────────────────────────────────────────────
<Methods>
Countable Hyper Cubes
Asymptotic Critical Value
Cramer-von Mises-type statistic / Max function
──────────────────────────────────────────────────────────────────────────────
<Results>
.% confidence interval is:
( 0.413 , 0.621 )
. cmi_interval (lbound) ( ) X1 X2, deci(2) level(0.9)
Conditional Moment Inequalities Interval              Number of obs : 2054
──────────────────────────────────────────────────────────────────────────────
<Variables>
Variables for the Lower Bound : lbound
Variables for the Upper Bound :
Instruments : X1 X2
──────────────────────────────────────────────────────────────────────────────
<Methods>
Countable Hyper Cubes
Asymptotic Critical Value
Cramer-von Mises-type statistic / Sum function
──────────────────────────────────────────────────────────────────────────────
<Results>
.% confidence interval is:
(  0.42 , inf )
```

The first case uses the default options and yields the 95% confidence interval: $[0.413, 0.621]$. The second case uses the max function for the test statistic and yields almost the same result as the first case.

In the third case, we omit `ubound`. This case illustrates how to construct a one-sided CS. Suppose only the lower bounds for the parameter exist [that is, $k_u = 0$ in (5)]. When we empty the second bracket of the syntax, the command gives a one-sided confidence interval. The third case also activates the `level(0.9)` option. Thus the resulting confidence level is 90%. It also activates `deci(2)`, yielding results with accuracy up to the second digit.

7 References

Andrews, D. W. K., and X. Shi. 2013. Inference based on conditional moment inequalities. *Econometrica* 81: 609–666.

———. 2014. Nonparametric inference based on conditional moment inequalities. *Journal of Econometrics* 179: 31–45.

Chernozhukov, V., W. Kim, S. Lee, and A. M. Rosen. 2015. Implementing intersection bounds in Stata. *Stata Journal* 15: 21–44.

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

Kaido, H., F. Molinari, and J. Stoye. 2016. Confidence intervals for projections of partially identified parameters. ArXiv Working Paper No. arXiv:1601.00934. http://arxiv.org/abs/1601.00934.

Lee, S., K. Song, and Y.-J. Whang. 2013. Testing functional inequalities. *Journal of Econometrics* 172: 14–32.

Mourifié, I., and Y. Wan. 2014. Testing local average treatment effect assumptions. Working Paper. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2429664.

About the authors

Donald W. K. Andrews is the T. C. Koopmans Professor of Economics and Statistics at Yale University.

Wooyoung Kim is a graduate student at the University of Wisconsin–Madison.

Xiaoxia Shi is an assistant professor at the University of Wisconsin–Madison.