The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–17

Stata Commands for Testing Conditional Moment Inequalities/Equalities

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

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

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

Abstract. In this paper, we present two commands – **cmi_test** and **cmi_interval** – to implement the testing and inference methods for conditional moment inequalities/equalities models proposed in Andrews and Shi (2013). The command **cmi_test** tests the validity of a finite number of conditional moment equalities and/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 command **cmi_interval** returns the confidence interval for a one-dimensional parameter defined by intersection bounds. This confidence interval is obtained by inverting **cmi_test**. All procedures implemented are uniformly asymptotically valid under appropriate conditions (specified in Andrews and Shi (2013)).

Keywords: **cmi_test**, **cmi_interval**, conditional moment inequalities/equalities, confidence interval, uniformly asymptotically valid test

1 Introduction

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

The Stata package described in this paper is not intended as a tool to compute 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 at which **cmi_test** returns an acceptence. Simple grid-search algorithms for this task become exponentially more costly as the dimension of θ increases. More efficient algorithms are available in other commonly used statistical softwares, but we are not aware of an implementation of them in Stata.

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

The Stata commands in this paper offer a rich set of options to allow the user to fine tune the procedure. However, in most applications, the default options, which are the recommended options in Andrews and Shi (2013), work well and the user need not make many choices.

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

2 Framework

2.1 Parameter Inference Based on Conditional Moment Inequalities/Equalities

Consider an independent and identically distributed (i.i.d.) sample $\{W_i\}_{i=1}^n$. Let X_i be a vector of instrumental variables, which is a subvector of W_i. A conditional moment inequality/equality (CMI) model is of the form

$$\begin{aligned} &E[m_j(W_i, \theta_0)|X_i] \geq 0 \text{ for } j = 1, \ldots, p, \\ &E[m_j(W_i, \theta_0)|X_i] = 0 \text{ for } j = p+1, \ldots, k, \text{ almost surely,} \end{aligned} \tag{1}$$

where p and k are two non-negative integers such that $k \geq p$ and $m(\cdot, \theta_0) := m_1(\cdot, \theta_0), \ldots, m_k(\cdot, \theta_0))'$ 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 W_i, which makes those elements excluded variables. The CMI model arises in many modeling contexts. An example is given later in this paper, and more are given in Andrews and Shi (2013).

In a CMI model, the parameter θ_0 may or may not be point identified. Thus, a con-

1. Stata commands for the procedure in Chernozhukov, Lee and Rosen (2013) are introduced in Chernozhukov, Kim, Lee and Rosen (2015).

sistent point estimator for θ_0 may or may not exist, and typical t-test based confidence intervals do not apply. However, one can still test hypotheses on the parameter such as

$$H_0 : \theta_0 = \theta, \tag{2}$$

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

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

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.

The test then can be inverted to construct a confidence set for θ_0. The confidence set is defined as

$$\mathrm{CS}_n(1-\alpha) = \{\theta \in \Theta : \phi_n(\theta) = 0\}. \tag{4}$$

The standard way to compute such a confidence set 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

$$\begin{aligned} &E[m_j(W_i)|X_i] \geq 0 \text{ for } j = 1, \ldots, p, \\ &E[m_j(W_i)|X_i] = 0 \text{ for } j = p+1, \ldots, k, \text{ almost surely,} \end{aligned} \tag{5}$$

which does not depend on a parameter θ, where $m(\cdot) := (m_1(\cdot), \ldots, m_k(\cdot))'$ is a vector of known functions of the observables and W_i, 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 (5) is the same as testing (2) in the model in (1). One just replaces $m(\cdot, \theta)$ with $m(\cdot)$ and, in consequence, the test in (3) does not depend on θ.

Now, we briefly describe a conditional average treatment effect example of the testing problem in (5). 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 observed only if $D = 1$, and $Y(0)$ is the untreated outcome observable only if $D = 0$. Let X be a vector of covariates. Suppose that D is randomly assigned, with each individual receiving treatment with a known probability p. Then, the average conditional treatment effect

2. The testing command **cmi_test** cam be combined 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). This algorithm can be implemented using a MATLAB toolbox called "DACE" which is publicly available. So far, we are not aware whether this algorithm can be used for Stata command. For details of "DACE", see http://www2.imm.dtu.dk/projects/dace/.

given X can be expressed as follows:

$$E(Y(1) - Y(0)|X) = E\left(\frac{DY}{p} - \frac{(1-D)Y}{1-p} \middle| X \right). \tag{6}$$

Suppose that the researcher would like to test if the average treatement effect is negative for individuals at all X values. In the framework above, this problem can be written as testing the null hypothesis

$$H_0 : E[m_j(W)|X] \geq 0, \tag{7}$$

where $j = 1, W = (Y, D, X)$, and $m_1(W) = -\frac{DY}{p} - \frac{(1-D)Y}{1-p}$.

2.2 Confidence Intervals Based on Intersection Bounds

The command **cmi_interval** computes the confidence set 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 that the confidence set of θ_0 is an interval. The command **cmi_interval** combines a one-dimensional grid-search algorithm with **cmi_test** in order to compute this interval. Specifically, the command applies when the CMI model takes the following form:

$$\begin{aligned} &E[\rho_{u,j}(W_i) - \theta_0|X_i] \geq 0 \text{ for } j = 1, \ldots, k_u, \\ &E[\theta_0 - \rho_{\ell,j}(W_i)|X_i] \geq 0 \text{ for } j = 1, \ldots, k_\ell, \end{aligned} \tag{8}$$

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 $E[\rho_{u,1}(W_i)|X_i], \ldots, E[\rho_{u,k_u}(W_i)|X_i]$, and the lower bounds are $E[\rho_{\ell,1}(W_i)|X_i], \ldots, E[\rho_{\ell,k_\ell}(W_i)|X_i]$. It is easy to see that (8) is a special case of (1) with $p = k_u + k_\ell$, $k = p$, and

$$m_j(W_i, \theta_0) = \begin{cases} \rho_{u,j}(W_i) - \theta_0 & \text{for } j = 1, \ldots, k_u \\ \theta_0 - \rho_{\ell,j-k_u}(W_i) & \text{for } j = k_u + 1, \ldots, k_u + k_\ell. \end{cases} \tag{9}$$

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

We use a censored data example similar to that in Andrews and Shi (2014) to illustrate the model in (8). Let D be a binary variable indicating data censorship, and X be a covariate vector. Let Y^* be a variable subject to censoring, that is, it is only observed when $D = 1$. Let θ_0 denote the conditional cumulative distribution function (cdf) of Y^* given X evaluated at a certain point y_0. Then, θ_0 is bounded by the inequalities in (8) with $k_u = k_\ell = 1$,

$$\rho_{u,1}(W) = 1\{Y \leq y_0, D = 1\} + 1\{D = 0\}, \text{ and} \tag{10}$$

$$\rho_{\ell,1}(W) = 1\{Y_i \leq y_0, D_i = 1\}. \tag{11}$$

We illustrate the implementation of both Stata commands using this example in Section 7 below.

3 Detailed Procedures

This section describe the detailed procedures that the commands implement. These procedures are from Andrews and Shi (2013). 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 (8).

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), the procedure can be applied similarly to the hypothesis in (5) as well. 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 X_i. The ones we employ are countable hypercubes on standardized X_i. We define the standardized X_i variables first. The standardized X_i, denoted X_i^o, equals

$$X_i^o = \Phi(\widehat{\Sigma}_{X,n}^{-1/2}(X_i - \bar{X}_n)),$$

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

The instrumental functions are of the form

$$g_{a,r}(X_i^o) = 1\{X_i^o \in \times_{u=1}^{d_x}((a_u - 1)/(2r), a_u/(2r)]\}, \tag{12}$$

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 only consider $r = 1, 2, \ldots, r_{1,n}$ for a positive integer $r_{1,n}$. The command **cmi_test** uses $\lfloor n^{d_x/2}/2 \rfloor$ 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 $m_j(W_i, \theta)$ on θ throughout. We have

$$\bar{m}_{n,j}(g_{a,r}) = n^{-1}\sum_{i=1}^n m_j(W_i) g_{a,r}(X_i^o). \tag{13}$$

We also compute the sample variance, $\widehat{\sigma}_{n,j}^2(g_{a,r})$, of $m_j(W_i)g_{a,r}(X_i^o)$. 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 $m_j(W_i)$ for the purpose of regularizing $\widehat{\sigma}_{n,j}^2(g_{a,r})$. The regularized variance

$$\bar{\sigma}_{n,j}^2(g_{a,r}) = \widehat{\sigma}_{n,j}^2(g_{a,r}) + \varepsilon\widehat{\sigma}_{n,j}^2 \tag{14}$$

is used in the test statistic. The regularization parameter ε equals 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 the sample moments. After constructing the test statistic, we proceed to 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 in next three subsections.

By default, **cmi_test** uses summation (Sum) to aggregate over j for each (a,r), and uses Cramer-von Mises (CvM)-type aggregation over (a,r), which yields the following test statistic:

$$T_n = n\sum_{r=1}^{r_{1,n}} \frac{\sum_{a\in\{1,\ldots,2r\}^{d_x}} \left(\sum_{j=1}^{p} \left[\frac{\bar{m}_{n,j}(g_{a,r})}{\bar{\sigma}_{n,j}(g_{a,r})} \right]_-^2 + \sum_{j=p+1}^{k} \left(\frac{\bar{m}_{n,j}(g_{a,r})}{\bar{\sigma}_{n,j}(g_{a,r})} \right)^2 \right)}{(r^2+100)(2r)^{d_x}}, \tag{15}$$

where the negative part function $[x]_- = \max\{0,-x\}$. By specifying the options **sfunc** and **ks**, **cmi_test** allows the user to choose from the CvM-Max statistic, the Kolmogorov-Smirnov (KS)-Sum statistic, or the Kolmogorov-Max statistic.[3] Choosing Max instead of Sum replaces the expression in the large brackets in the numerator by

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

Choosing KS instead of CvM replaces the $\sum_{r=1}^{r_{1,n}} \frac{\sum_{a\in\{1,\ldots,2r\}^{d_x}}}{(r^2+100)(2r)^{d_x}}$ in (15) by

$$\max_{(a,r):a\in\{1,\ldots,2r\}^{d_x}, r=1,\ldots,r_{1,n}}. \tag{16}$$

Asymptotic Critical Values

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

$$n^{-1/2}(\nu_{n,j}(g_{a,r}) + \varphi_{n,j}(g_{a,r})), \tag{17}$$

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

3. The commands do not incorporate the quasi-likelihood ratio (QLR) statistic discussed in Andrews and Shi (2013) because the QLR statistic requires repeatedly carrying out a quadratic optimization operation a large number of times and we are not aware of a fast quadratic optimization routine in Stata.

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

$$(m_j(W_i)g_{a,r}(X_i^o))_{j=1,\ldots,k,a\in\{1,\ldots,2r\}^{d_r},r=1,\ldots,r_{1,n}}. \tag{18}$$

A large number of draws are taken, and each draw is used to compute a draw of the statistic T_n^{Asy}. Then the command 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 be 5001 and the seed of the random number generator is set to 1000. These can be changed by specifying the options **rep** and **seed**.

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

$$\varphi_{n,j}(g_{a,r}) = \begin{cases} \widehat{\sigma}_{n,j}B_n & \text{if } \kappa_n^{-1}n^{1/2}\bar{m}_{n,j}(g_{a,r})/\bar{\sigma}_{n,j}(g_{a,r}) > 1 \\ 0 & \text{otherwise,} \end{cases} \tag{19}$$

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}) that is defined in the same way as T_n is defined except with $\bar{m}_{n,j}(g_{a,r})/\bar{\sigma}_{n,j}(g_{a,r})$ replaced by

$$\frac{n^{-1/2}(\nu_{n,j}^{\text{Boot}}(g_{a,r}) + \varphi_{n,j}(g_{a,r}))}{\bar{\sigma}_{n,j}^{\text{Boot}}(g_{a,r})}, \tag{20}$$

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

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

$$n^{1/2}\left[\bar{m}_{n,j}^*(g_{a,r}) - \bar{m}_{n,j}(g_{a,r})\right], \tag{21}$$

where $\bar{m}_{n,j}^*(g_{a,r})$ is computed using the same procedure as that for $\bar{m}_{n,j}(g_{a,r})$ except with $\{W_i^*\}$ (and its subvector $\{X_i^*\}$ replaced by $\{W_i\}$ (and its subvector $\{X_i\}$). The draw of $\bar{\sigma}_{n,j}^{\text{Boot}}(g_{a,r})$ is computed using the same procedure as that for $\bar{\sigma}_{n,j}(g_{a,r})$ except with $\{W_i^*\}$ (and its subvector $\{X_i^*\}$) replacing $\{W_i\}$ (and its subvector $\{X_i\}$). These are then used to compute one draw of T_n^{Boot}. By repeating the process, a large number

of T_n^{Boot} draws are taken, and $c_n(1-\alpha)$ is defined to be the $1-\alpha$ empirical quantile of these draws. By default, the number of draws is set to be 5001 and the seed of the random number generator is set to 1000. These can be changed by specifying the options **rep** and **seed**.

3.2 Confidence Interval Construction for Intersection Bound Models

In this section, we describe the algorithm used to construct a confidence interval for the one-dimensional parameter in the model in (8).

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 following algorithm. The algorithm consists 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:

$$\begin{aligned}\hat{\theta}_{lb,pre} &= \min_{1\leq j\leq k_l} \min_{i\leq n} \rho_{l,j}(W_i) \\ \hat{\theta}_{ub,pre} &= \max_{1\leq j\leq k_u} \max_{i\leq n} \rho_{u,j}(W_i).\end{aligned}$$

In addition, we define two auxiliary statistics:

$$\begin{aligned}\hat{\theta}_{lb,bound} &= \max_{1\leq j\leq k_l} \max_{i\leq n} \rho_{l,j}(W_i) \\ \hat{\theta}_{ub,bound} &= \min_{1\leq j\leq k_u} \min_{i\leq n} \rho_{u,j}(W_i).\end{aligned}$$

Note that $\hat{\theta}_{lb,pre}$ ($\hat{\theta}_{ub,pre}$) is a preliminary conservative lower (upper) bound for the confidence interval. Meanwhile, $\hat{\theta}_{lb,bound}$ ($\hat{\theta}_{ub,bound}$) 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 $\lfloor\hat{\theta}_{lb,pre}\rfloor$ and $\lceil\hat{\theta}_{lb,bound}\rceil$ is 1, skip the current step, let $\hat{\theta}_{lb,0} = \lfloor\hat{\theta}_{lb,pre}\rfloor$ and move to the next step. Otherwise, consider grid points on $[\lfloor\hat{\theta}_{lb,pre}\rfloor, \lceil\hat{\theta}_{lb,bound}\rceil]$ with distance between adjacent grid points being $d_0 = \lfloor\max((\lceil\hat{\theta}_{lb,bound}\rceil - \lfloor\hat{\theta}_{lb,pre}\rfloor)/20, 1)\rfloor$. Apply **cmi_test** for θ_0 being each of these grid points. Record the largest grid point rejected by the test as $\hat{\theta}_{lb,0}$ and consider grid points on $[\hat{\theta}_{lb,0}, \hat{\theta}_{lb,0} + d_0]$ 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 $\hat{\theta}_{lb,0}$.

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

...

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

By default, the command iterates this algorithm up to the thousandth place (i.e. Step (3)). One can choose the number of iterations (i.e. 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 employ the upper bound and the lower bound moment inequalities. Then the algorithm 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 empty set. This implies that the model is rejected at the specified confidence level α (the default is 95%).

Let $\hat{\theta}_{lb,-1}$ and $\hat{\theta}_{ub,-1}$ be the lower and upper bounds of the crude interval just specified. Then we 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 $\hat{\theta}_{lb,0} = \hat{\theta}_{lb,-1}, \hat{\theta}_{ub,0} = \hat{\theta}_{ub,-1}$ and move to the next step. Otherwise, set $d_0 = \lfloor \max(\{\lceil \hat{\theta}_{ub,-1} \rceil - \lfloor \hat{\theta}_{lb,-1} \rfloor\}/20, 1) \rfloor$ and apply **cmi_test** using all of the inequalities for each of the evenly spaced grid points (including the endpoints) with spacing d_0 on $[\lfloor \hat{\theta}_{lb,-1} \rfloor, \lceil \hat{\theta}_{ub,-1} \rceil]$.

(Case 1) If there exists at least one grid point not rejected, let θ_{lb,d_0} and θ_{ub,d_0} denote the smallest and the largest non-rejected points, respectively.

(Case 2) If there is no point not rejected, find the grid point with the largest p-value (denoted by θ_{high,d_0}) and let $\theta_{lb,d_0} = \theta_{ub,d_0} = \theta_{high,d_0}$.

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 $[\theta_{lb,d_0} - d_0, \theta_{lb,d_0}]$ and also those on $[\theta_{ub,d_0}, \theta_{ub,d_0} + d_0]$. Apply **cmi_test** using all of the 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 $[\hat{\theta}_{lb,0}, \hat{\theta}_{ub,0}] = [\theta_{lb,d_J} - d_J, \theta_{ub,d_J} + d_J]$. This interval is the Andrews and Shi (2013) confidence interval accurate up to the integer level. If higher accuracy is desired, move on to the next step.

...

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

(Case 1) If there exists at least one point not rejected, let $\theta_{lb,j}$ and $\theta_{ub,j}$ 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 $\theta_{high,i+1}$) and let $\theta_{lb,i+1} = \theta_{ub,i+1} = \theta_{high,i+1}$.

Let $[\hat{\theta}_{lb,j}, \hat{\theta}_{ub,i+1}] = [\theta_{lb,i+1} - 10^{-i-1}, \theta_{ub,i+1} + 10^{-i-1}]$. This interval is the Andrews and Shi (2013) confidence interval with accuracy up to 10^{-i-1}. If higher accuracy is desired, 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 using the **deci** option.

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

4 Installation of the cmi_test Package

All Stata commands below are available at the Statistical Software Components (SSC) archive. Our Stata package, `cmitest`, can be installed from within Stata by typing `ssc install cmitest` or `ssc install cmitest, all`. The former installs the commands and the help files, while the latter installs those as well as the ancillary data file that allows the user to run the examples in the help files.

5 The cmi_test Command

5.1 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`]

5.2 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 (5)). To use this command, one first generates variables that equal $m_1(W_i, \theta), \ldots, m_k(W_i, \theta)$ for observations $i = 1, \ldots, n$ (or $m_1(W_i), \ldots, m_k(W_i)$ for observations $i = 1, \ldots, n$). The first p of them 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 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 (12) above by default. That choice is fine when the number of *indepvars* is three or less. When the dimension of *indepvars* is greater than 3, the number of cubes may be too large which causes long computation time. The command allows an alternative method for high dimensional independent variables. The user can select the **hd** option to opt for this method. This option implements the method described in the last paragraph of Section 9 of Andrews and Shi (2013).

5.3 Options

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

`hd` uses 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 this option is not used, the command computes the critical value based on a Gaussian asymptotic approximation.

`ks` uses the Kolmogorov-Smirnov-type statistic; the default is the Cramer-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 $\left(\frac{0.4 \log n}{\log \log n}\right)^{1/2}$

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

`seed(#)` sets the seed number 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. This option should be turned on when the command is used inside a Monte

Carlo simulation loop, so as not to interfere with the random number generation process set for the Monte Carlo simulation exercise.

5.4 Saved 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 kappa_n	`r(B)`	tuning parameter B_n
`r(epsilon)`	tuning parameter epsilon	`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)`	"Conditional Moment Inequalities Test"
`r(m_ineq)`	varlist for conditional moment inequalities, if any	`r(m_eq)`	varlist for conditional moment equalities, if any
`r(x)`	varlist for the instrumental variables		

6 The cmi_interval Command

6.1 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`]

6.2 Description

cmi_interval constructs the confidence interval for the parameter in model (8) by inverting **cmi_test**. The *upper bound vars* are $\rho_{u,1}(W_i), \ldots, \rho_{u,k_u}(W_i)$. The *lower bound vars* are $\rho_{\ell,1}(W_i), \ldots, \rho_{\ell,k_\ell}(W_i)$. The *indepvars* are the elements of X_i.

6.3 Options

cmi_interval accepts all the options that `cmi_test` does. There are two additional options 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 0.95.

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

6.4 Saved 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 kappa_n
`r(B)`	tuning parameter B_n	`r(epsilon)`	tuning parameter epsilon
`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		

Macros

`r(cmd)`	`cmi_test`	`r(title)`	"Conditional Moment Inequalities Interval"
`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 regressors		

7 Examples

We provide an example of estimating a conditional distribution with censored data, which is 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 National Longitudinal Survey of Youth 1997 (NLSY97). From that data set, 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 2054.

Let Y_i^* be the natural logarithm of the potential wage after college enrollment. The 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 cumulative distribution function Y_i^* evaluated at y_0. Then, θ_0 is bounded by the moment inequalities (8) with the bounding moment functions defined in (10) and (11). 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 sub-population (currently working, not self-employed male) whose expected hourly wage is lower than \$20 if they had enrolled in a college. We create two variables defined by $1\{Y_i \leq y_0, D_i = 1\}$ and $1\{Y_i \leq y_0, D_i = 1\} + 1\{D_i = 0\}$:

```
. local y0 = log(20)

. gen lbound = ( Y < `y0´ ) * D

. gen ubound = ( Y < `y0´ ) * D + 1 - D
```

7.1 cmi_test

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

$$H_0 : F_{Y^*}(\log(20)) = 0.5. \tag{22}$$

Then, the researcher creates two conditional moment inequalities ((10) and (11)) by using the following commands:

```
. local theta0 = 0.5

. gen CMI1 = `theta0´ - lbound

. gen 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%) : 0.2363
               (5%) : 0.1761
              (10%) : 0.1500
p-value             : 0.9872

. 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%) : 5.9032
               (5%) : 4.3052
```

```
              (10%) : 3.5024
p-value             : 0.9440

. 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%) : 0.2590
               (5%) : 0.1913
              (10%) : 0.1604
p-value             : 0.9952
```

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, then variables representing those equalities should be positioned in the second parenthesis of the syntax.

7.2 cmi_interval

Now we compute a confidence interval for θ_0. In this example, the variables `lbound` and `ubound` 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>
95% confidence interval is:
( 0.413 , 0.620 )

. 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>
95% confidence interval is:
( 0.413 , 0.619 )

. 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>
90% confidence interval is:
(  0.42 , inf )
```

The first case uses the default options and yields the 95% confidence interval: $(.413, .620)$. 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, `ubound` is omitted. This case illustrates how a one-sided confidence set can be constructed. Suppose that only the lower bounds for the parameter exist (i.e., $k_u = 0$ in (8)). Then, by emptying 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 the `deci(2)` option and thus yields results with accuracy up to the second digit.

8 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. *The 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. *Working Paper* .

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 LATE Assumptions. Technical report, University of Toronto.

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 at Madison.

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