Queen's Economics Department Working Paper No. 1406

Fast and Wild:
Bootstrap Inference in Stata using boottest

David Roodman, Open Philanthropy Project

James G. MacKinnon, Queen's University

Morten Ørregaard Nielsen, Queen's University and CREATES

Matthew D. Webb, Carleton University

Department of Economics
Queen's University
94 University Avenue
Kingston, Ontario, Canada
K7L 3N6

6-2018

Fast and Wild: Bootstrap Inference in Stata Using `boottest`

David Roodman
Open Philanthropy Project
San Francisco, CA, United States
david.roodman@openphilanthropy.org

James G. MacKinnon
Queen's University
Kingston, Ontario, Canada
jgm@econ.queensu.ca

Morten Ørregaard Nielsen
Queen's University
Kingston, Ontario, Canada
and CREATES, Aarhus University, Denmark
mon@econ.queensu.ca

Matthew D. Webb
Carleton University
Ottawa, Ontario, Canada
matt.webb@carleton.ca

June 21, 2018

Abstract

The wild bootstrap was originally developed for regression models with heteroskedasticity of unknown form. Over the past thirty years, it has been extended to models estimated by instrumental variables and maximum likelihood, and to ones where the error terms are (perhaps multi-way) clustered. Like bootstrap methods in general, the wild bootstrap is especially useful when conventional inference methods are unreliable because large-sample assumptions do not hold. For example, there may be few clusters, few treated clusters, or weak instruments. The Stata package `boottest` can perform a wide variety of wild bootstrap tests, often at remarkable speed. It can also invert these tests to construct confidence sets. As a post-estimation command, `boottest` works after linear estimation commands including `regress`, `cnsreg`, `ivregress`, `ivreg2`, `areg`, and `reghdfe`, as well as many estimation commands based on maximum likelihood. Although it is designed to perform the wild cluster bootstrap, `boottest` can also perform the ordinary (non-clustered) version. Wrappers offer classical Wald, score/LM, and Anderson-Rubin tests, optionally with (multi-way) clustering. We review the main ideas of the wild cluster bootstrap, offer tips for use, explain why it is particularly amenable to computational optimization, state the syntax of `boottest`, `artest`, `scoretest`, and `waldtest`, and present several empirical examples for illustration.

Keywords: boottest, artest, waldtest, scoretest, Anderson-Rubin test, Wald test, wild bootstrap, wild cluster bootstrap, score bootstrap, multi-way clustering, few treated clusters

JEL Codes: C15, C21, C23, C25, C36.

1 Introduction

It is common in social science research to assume that the error terms in regression models are correlated within clusters. These clusters might be, for example, jurisdictions, villages, firm types, classrooms, schools, or time periods. This is why many Stata estimation commands offer a `cluster` option to implement a variance matrix robust to both intra-cluster correlation and heteroskedasticity of unknown form. Inference based on the standard errors produced by this option can work well when large-sample theory provides a good guide to the finite-sample properties of the cluster-robust

variance matrix estimator, or CRVE. This will typically be the case if the number of clusters is large, if the clusters are reasonably homogeneous, in the sense that they are similar in size and have similar variance matrices, and—for regressions estimating treatment effects—if the number of treated clusters is not too small. But inference can sometimes be misleading when these conditions do not hold.

One way to improve inference when large-sample theory provides a poor guide is to use the bootstrap. The idea is to generate a large number of bootstrap samples that mimic the distribution from which the actual sample was obtained. Each of them is then used to compute a bootstrap test statistic, using the same test procedure as for the original sample. The bootstrap P value is then calculated as the proportion of the bootstrap statistics that are more extreme than the actual one from the original sample. See, among many others, Davison and Hinkley (1997) and MacKinnon (2009) for a general introduction.

In general, bootstrap inference will be more reliable the more closely the bootstrap data-generating process (DGP) matches the (unknown) true DGP; see Davidson and MacKinnon (1999). For many regression models, the wild bootstrap (Liu, 1988; Wu, 1986) frequently does a good job of matching the true DGP. This method was adapted to models with clustered errors by Cameron, Gelbach, and Miller (2008), and we discuss the wild cluster bootstrap in detail in Section 3. Simulation evidence suggests that bootstrap tests based on the wild and wild cluster bootstraps often perform well; see, for example, Davidson and MacKinnon (2010), MacKinnon (2013), and MacKinnon and Webb (2017a).

A less well-known strength of the wild bootstrap is that, in many important cases, its simple and linear mathematical form lends itself to extremely fast implementation. As we will explain in Section 5, the combined algorithm for generating a large number of bootstrap replications and computing a test statistic for each of them can often be condensed into a few matrix formulas. For the linear regression model with clustered errors, viewing the process in this way opens the door to fast implementation of the wild cluster bootstrap.

The post-estimation command `boottest` implements several versions of the wild cluster bootstrap, which include the ordinary (non-clustered) wild bootstrap as a special case. It supports:

- inference after OLS estimation with `regress`;
- inference after restricted OLS estimation with `cnsreg`;
- inference after the wild restricted efficient bootstrap of Davidson and MacKinnon (2010) for two-stage least squares, limited information maximum likelihood, and other instrumental variables (IV) methods, as with `ivregress` and `ivreg2` (Baum, Schaffer, and Stillman, 2007);
- inference after maximum likelihood (ML) estimation via the "score bootstrap" (Kline and Santos, 2012) with such ML-based commands as `probit`, `logit`, `glm`, `sem`, `gsem`, and the user-written `cmp` (Roodman, 2011);
- multi-way clustering, even after estimation commands that do not support it;
- models with one-way fixed effects, estimated with `areg`, `reghdfe` (Correia, 2016), `xtreg`, `xtivreg`, or `xtivreg2` with the `fe` option;
- inversion of tests of one-dimensional hypotheses to derive confidence sets;
- plotting of the corresponding confidence functions.

Section 2 of this paper describes the linear regression model with one-way clustering and explains how to compute cluster-robust variance matrices. Section 3 describes the key ideas of the wild cluster bootstrap and then introduces several variations and extensions. Section 4 discusses multi-way clustering and associated bootstrap methods. Section 5 details techniques implemented in `boottest` to speed up execution of the wild cluster bootstrap, especially when the number of clusters is small relative to the number of observations. Section 6 discusses extensions to instrumental variables estimation and to maximum likelihood estimation. Section 7 describes the syntax of the `boottest` package. Finally, Section 8 illustrates how to use the program in the context of some empirical examples.

2 Regression models with one-way clustering

Consider the linear regression model

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{u}, \tag{1}$$

where $\boldsymbol{y}$ and $\boldsymbol{u}$ are $N \times 1$ vectors of observations and error terms, respectively, $\boldsymbol{X}$ is an $N \times k$ matrix of covariates, and $\boldsymbol{\beta}$ is a $k \times 1$ parameter vector. The observations are grouped into G clusters, within each of which the error terms may be correlated. Model (1) can be rewritten as

$$\boldsymbol{y}_g = \boldsymbol{X}_g\boldsymbol{\beta} + \boldsymbol{u}_g, \quad g = 1, \ldots, G, \tag{2}$$

where the vectors $\boldsymbol{y}_g$ and $\boldsymbol{u}_g$ and the matrix $\boldsymbol{X}_g$ contain the rows of $\boldsymbol{y}$, $\boldsymbol{u}$, and $\boldsymbol{X}$ that correspond to the g^{th} cluster. Each of these objects has N_g rows, where N_g is the number of observations in cluster g.

Conditional on $\boldsymbol{X}$, the vector of error terms $\boldsymbol{u}$ has mean zero and variance matrix $\boldsymbol{\Omega} = \mathrm{E}(\boldsymbol{u}\boldsymbol{u}' \,|\, \boldsymbol{X})$, which is subject to the key assumption of no cross-cluster correlation:

$$\mathrm{E}(\boldsymbol{u}_{g'}\boldsymbol{u}_g' \,|\, \boldsymbol{X}) = \boldsymbol{0} \quad \text{if} \quad \mathbf{g}' \neq \mathbf{g}.$$

Thus $\boldsymbol{\Omega}$ is block-diagonal with blocks $\boldsymbol{\Omega}_g = \mathrm{E}(\boldsymbol{u}_g\boldsymbol{u}_g' \,|\, \boldsymbol{X})$. We make no assumptions about the entries in these blocks, except that each of the $\boldsymbol{\Omega}_g$ is a valid variance matrix (positive definite and finite).

The ordinary least squares (OLS) estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}, \tag{3}$$

and the vector of estimation residuals is

$$\hat{\boldsymbol{u}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}. \tag{4}$$

The conditional variance matrix of the estimator $\hat{\boldsymbol{\beta}}$ is

$$\mathrm{Var}\,(\hat{\boldsymbol{\beta}} \,|\, \boldsymbol{X}) = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{\Omega}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}. \tag{5}$$

This expression cannot be computed, because it depends on $\boldsymbol{\Omega}$, which by assumption is not known. The feasible analog of (5) is the cluster-robust variance estimator (CRVE) of (Liang and Zeger, 1986). It replaces $\boldsymbol{\Omega}$ by the estimator $\hat{\boldsymbol{\Omega}}$, which has the same block-diagonal form as $\boldsymbol{\Omega}$, but with $\hat{\boldsymbol{\Omega}}_g = \hat{\boldsymbol{u}}_g\hat{\boldsymbol{u}}_g'$ for $g = 1, \ldots, G$. This leads to the CRVE

$$\hat{\boldsymbol{V}} = m(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\hat{\boldsymbol{\Omega}}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}, \quad \hat{\boldsymbol{\Omega}} = \mathrm{blockdiag}(\hat{\boldsymbol{\Omega}}_1, \ldots, \hat{\boldsymbol{\Omega}}_G), \tag{6}$$

where the scalar finite-sample adjustment factor, m, will be defined below. The center factor in (6) is often more conveniently computed as

$$\boldsymbol{X}'\hat{\boldsymbol{\Omega}}\boldsymbol{X} = \sum_{g=1}^{G} \boldsymbol{X}_g'\hat{\boldsymbol{u}}_g\hat{\boldsymbol{u}}_g'\boldsymbol{X}_g. \tag{7}$$

It should be clear that $\hat{\boldsymbol{\Omega}}$ is in no sense a consistent estimator of $\boldsymbol{\Omega}$, because both are $N \times N$ matrices. However, the "sandwich" estimator $\hat{\boldsymbol{V}}$ defined in (6) is consistent, in the sense that the inverse of the true variance matrix defined in (5) times $\hat{\boldsymbol{V}}$ converges to an identity matrix as $G \to \infty$. In the context of clustered errors, this consistency result was shown by Carter, Schnepel, and Steigerwald (2017) under some additional restrictions on the intra-cluster dependence structure and by Djogbenou, MacKinnon, and Nielsen (2018) for general one-way clustering.

When the error terms in the regression model (1) are potentially heteroskedastic, but uncorrelated, we have the "robust" case in informal Stata parlance. This is a special case of the clustered model (2) in which each cluster contains just one observation. For this model, the CRVE is simply the heteroskedasticity-robust variance estimator of Eicker (1963) and White (1980).

We now write $\hat{\boldsymbol{\Omega}}$ in a way that will facilitate our discussion of fast computation in Section 5. We define $\boldsymbol{S}$ as the $G \times N$ matrix which by left-multiplication sums the columns of a data matrix cluster-wise. For example, $\boldsymbol{SX}$ is the $G \times k$ matrix of cluster-wise sums of the columns of $\boldsymbol{X}$. Note that $\boldsymbol{S}'\boldsymbol{S}$ is the $N \times N$ indicator matrix whose $(i,j)^{\text{th}}$ entry is 1 or 0 according to whether observations i and j are in the same cluster. Left-multiplying by $\boldsymbol{S}'$ expands a matrix with one row per cluster to a matrix with one row per observation, by duplicating entries within each cluster.

We also adopt notation used for the family of colon-prefixed operators in Stata's matrix programming language, Mata. For example, if $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same dimensions, then $\boldsymbol{A} :* \boldsymbol{B}$ is the Hadamard (elementwise) product. But if $\boldsymbol{v}$ is a column vector while $\boldsymbol{B}$ is a matrix, and if the two have the same number of rows, then $\boldsymbol{v} :* \boldsymbol{B}$ is the columnwise Hadamard product of $\boldsymbol{v}$ with the columns of $\boldsymbol{B}$. As in Mata, we give $:*$ lower precedence than ordinary matrix multiplication. For later use, we note that, if $\boldsymbol{v}$ is a column vector, then the $:*$ operator obeys the identities

$$(\boldsymbol{v} :* \boldsymbol{A})' = \boldsymbol{A}' :* \boldsymbol{v}' = \boldsymbol{v}' :* \boldsymbol{A}', \tag{8}$$

$$\boldsymbol{v} :* \boldsymbol{AB} = (\boldsymbol{v} :* \boldsymbol{A})\boldsymbol{B}, \tag{9}$$

$$\boldsymbol{AB} :* \boldsymbol{v}' = \boldsymbol{A}(\boldsymbol{B} :* \boldsymbol{v}'), \tag{10}$$

$$\boldsymbol{A}(\boldsymbol{v} :* \boldsymbol{B}) = (\boldsymbol{A} :* \boldsymbol{v}')\boldsymbol{B}. \tag{11}$$

In terms of these constructs, the CRVE in (6) is

$$\hat{\boldsymbol{V}} = m(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\hat{\boldsymbol{\Omega}}\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}, \quad \hat{\boldsymbol{\Omega}} = \hat{\boldsymbol{u}} :* \boldsymbol{S}'\boldsymbol{S} :* \hat{\boldsymbol{u}}'. \tag{12}$$

By Stata convention, the small-sample correction m in (6) and (12) is

$$m = \frac{G}{G-1} \times \frac{N-1}{N-k}. \tag{13}$$

This choice has the intuitive property that, in the limiting case of $G = N$, i.e. the "robust" case, the expression reduces to the classical $N/(N-k)$ correction factor.

The CRVE in (12) allows for the possibility that individual observations within each cluster are not independent of each other. In other words, for purposes of estimation, the effective sample size may be less than the formal sample size N. In the extreme case of complete within-group

dependence, the cluster becomes the effective unit of observation, and the effective sample size becomes G.

The large-sample theory of the CRVE (Carter, Schnepel, and Steigerwald, 2017; Djogbenou, MacKinnon, and Nielsen, 2018) is based on the assumption that $G \to \infty$ rather than $N \to \infty$. This appears to be why Stata has long used the $t(G-1)$ distribution to obtain P values for cluster-robust t-statistics, a practice supported by the theory in Bester, Conley, and Hansen (2011). However this theory can be misleading when G is small, or the clusters differ greatly in size or in the structure of their variance matrices, or (in the treatment case) if there are few treated clusters. In some cases, it can result in severe overrejection. Simulation results that show the severity of this overrejection may be found in MacKinnon and Webb (2017b, 2018) and Djogbenou, MacKinnon, and Nielsen (2018), among others. In extreme cases, tests at the .05 level can reject well over 60% of the time.

The "effective number of clusters" developed in Carter, Schnepel, and Steigerwald (2017) often provides a useful diagnostic, which can be computed using the package `clusteff`; see Lee and Steigerwald (2017). Inference based on the $t(G-1)$ distribution is likely to be unreliable when G^*, the effective number of clusters, is much smaller than G, especially when G^* is less than about 20. In such cases, the wild cluster bootstrap and the $t(G-1)$ distribution may yield inferences that differ sharply. However, there is no reason to restrict the use of the wild cluster bootstrap to such cases. Because `boottest` is so fast, we recommend using it all the time, at least for final results.

3 The wild cluster bootstrap

Bootstrap methods for hypothesis testing involve generating many bootstrap samples that resemble the actual one, computing the test statistic for each of them, and then deciding how extreme the original test statistic is by comparing it with the distribution of the bootstrap test statistics. This sort of bootstrapping often works well. In some cases, it provides an *asymptotic refinement*, which means that, as the sample size increases, the bootstrap distribution approaches the actual distribution faster than does the asymptotic distribution that is normally relied upon (such as the t or chi-squared). This type of theoretical result holds for test statistics that are *asymptotically pivotal*, such as most t-statistics.

In this section, we discuss the wild cluster bootstrap (WCB), which was proposed in Cameron, Gelbach, and Miller (2008) and the validity of which was proved in Djogbenou, MacKinnon, and Nielsen (2018). It is a generalization of the ordinary wild bootstrap (WB), which was developed in Liu (1988) for the non-clustered, heteroskedastic case, following a suggestion in Wu (1986) and commentary thereon by Beran (1986). Härdle and Mammen (1993) appears to have been the first paper to call the method "wild," observing that it "... can be thought of as attempting to reconstruct the distribution of each residual through the use of one single observation."

3.1 The algorithm

The bootstrap method most deeply embedded in Stata—via the `bootstrap` prefix command—is the *nonparametric* or *pairs* bootstrap. In the estimation context set out in the previous section, this bootstrap would typically operate by resampling $(\boldsymbol{y}_g, \boldsymbol{X}_g)$ pairs at the cluster level (Field and Welsh, 2007). This is asymptotically valid, but some of the bootstrap samples may not resemble the actual sample, especially if the cluster sizes vary a lot. In a difference-in-differences context with few treated clusters, the treatment dummy could even equal zero for all observations in some bootstrap samples. For these reasons, the pairs cluster bootstrap can work poorly in some cases; see Bertrand, Duflo, and Mullainathan (2004) and MacKinnon and Webb (2017a).

In the wild cluster bootstrap, all the bootstrap samples have the same covariates $\boldsymbol{X}$. If the bootstrap samples are denoted by an asterisk and indexed by b, only the bootstrap error vector $\boldsymbol{u}^{*b}$, and hence also the bootstrap dependent variable $\boldsymbol{y}^{*b}$, differs across them. In particular, the vectors $\boldsymbol{y}^{*b}$ are generated, cluster by cluster, as

$$\boldsymbol{y}_g^{*b} = \boldsymbol{X}_g\ddot{\boldsymbol{\beta}} + \boldsymbol{u}_g^{*b}, \quad \boldsymbol{u}_g^{*b} = v_g^{*b}\ddot{\boldsymbol{u}}_g, \tag{14}$$

where $\ddot{\boldsymbol{\beta}}$ is an estimate of $\boldsymbol{\beta}$, and $\ddot{\boldsymbol{u}}_g$ is the vector of residuals for cluster g computed using $\ddot{\boldsymbol{\beta}}$. The scalar v_g^{*b} is an auxiliary random variable with mean 0 and variance 1, sometimes referred to as a "wild weight." It is often drawn from the Rademacher distribution, meaning that it takes the values -1 and $+1$ with equal probability. The choice of $\ddot{\boldsymbol{\beta}}$, and hence also $\ddot{\boldsymbol{u}}_g$, will be discussed just below. Using the matrix $\boldsymbol{S}$ defined above, we can rewrite the wild bootstrap DGP as

$$\boldsymbol{y}^{*b} = \boldsymbol{X}\ddot{\boldsymbol{\beta}} + \boldsymbol{u}^{*b}, \quad \boldsymbol{u}^{*b} = \ddot{\boldsymbol{u}} :* \boldsymbol{S}'\boldsymbol{v}^{*b}, \tag{15}$$

where $\boldsymbol{v}^{*b}$ is a $G \times 1$ vector with g^{th} element v_g^{*b}.

Because the v_g^{*b} are independent of $\boldsymbol{X}$ and the $\ddot{\boldsymbol{u}}_g$ and have mean 0 and variance 1, multiplying the $\ddot{\boldsymbol{u}}_g$ by v_g^{*b} as in (14) preserves the first and second moments of the $\ddot{\boldsymbol{u}}_g$. Specifically, let E^* denote expectation conditional on the data, $\boldsymbol{y}$ and $\boldsymbol{X}$, so that only the v_g^{*b} are treated as random. Then it is not difficult to see that

$$\begin{aligned} \mathrm{E}^*(\boldsymbol{u}_g^{*b}) &= \boldsymbol{0}, \\ \mathrm{E}^*(\boldsymbol{u}_{g'}^{*b}\boldsymbol{u}_g^{*b\prime}) &= \boldsymbol{0} \text{ when } \mathbf{g}' \neq \mathbf{g}, \text{ and} \\ \mathrm{E}^*(\boldsymbol{u}_g^{*b}\boldsymbol{u}_g^{*b\prime}) &= \ddot{\boldsymbol{u}}_g\ddot{\boldsymbol{u}}_g'. \end{aligned}$$

Up to this point, we have used $\ddot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{u}}$ to denote vectors of parameter estimates and residuals, without specifying precisely how they are computed. One possibility is that $\ddot{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{u}} = \hat{\boldsymbol{u}}$, where $\hat{\boldsymbol{\beta}}$ is the vector of OLS estimates for the model (1) and $\hat{\boldsymbol{u}}$ is the associated vector of residuals. Another possibility is that $\ddot{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{u}} = \tilde{\boldsymbol{u}}$, where $\tilde{\boldsymbol{\beta}}$ denotes least squares estimates of (1) subject to whatever restriction(s) are to be tested and $\tilde{\boldsymbol{u}}$ denotes the associated vector of residuals. For example, if we wish to test whether $\beta_j = 0$, where β_j is the j^{th} element of $\boldsymbol{\beta}$, we would set $\tilde{\beta}_j = 0$ and obtain the remaining elements of $\tilde{\boldsymbol{\beta}}$ by regressing $\boldsymbol{y}$ on every column of $\boldsymbol{X}$ except the j^{th}.

These two choices lead to two variants of the wild cluster bootstrap, which we call WCU (for wild cluster unrestricted) and WCR (for wild cluster restricted). Cameron, Gelbach, and Miller (2008) referred to WCR as the wild cluster bootstrap with the null imposed. Section 3.2 discusses why it is generally better to use the WCR variant. To explain how both work, we continue to use the generic notation $\ddot{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{u}}$.

For concreteness, we suppose that our objective is to test the hypothesis that $\beta_j = 0$. Then the WCU and WCR algorithms are as follows.

1. Regress $\boldsymbol{y}$ on $\boldsymbol{X}$ to obtain $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{u}}$, and $\hat{\boldsymbol{V}}$ as given in (3), (4), and (12).

2. Calculate the usual cluster-robust t-statistic for the hypothesis that $\beta_j = 0$,

$$t_j = \frac{\hat{\beta}_j}{\sqrt{\hat{V}_{jj}}}, \tag{16}$$

   where $\hat{V}_{jj}$ is the j^{th} diagonal element of $\hat{\boldsymbol{V}}$.

3. For the WCU bootstrap, set $\ddot{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{u}} = \hat{\boldsymbol{u}}$. For the WCR bootstrap, regress $\boldsymbol{y}$ on $\boldsymbol{X}$ subject to the restriction that $\beta_j = 0$ to obtain $\tilde{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{u}}$, and set $\ddot{\boldsymbol{\beta}} = \tilde{\boldsymbol{\beta}}$ and $\ddot{\boldsymbol{u}} = \tilde{\boldsymbol{u}}$.

4. For each of B bootstrap replications:

   (a) Using (15), generate a new set of bootstrap error terms $\boldsymbol{u}^{*b}$ and dependent variables $\boldsymbol{y}^{*b}$.

   (b) By analogy with step 1, regress $\boldsymbol{y}^{*b}$ on $\boldsymbol{X}$ to obtain $\hat{\boldsymbol{\beta}}^{*b}$ and $\hat{\boldsymbol{u}}^{*b}$, and use the latter in (12) to compute the bootstrap CRVE $\hat{\boldsymbol{V}}^{*b}$.

   (c) Calculate the bootstrap t-statistic for the null hypothesis that $\beta_j^{*b} = \ddot{\beta}_j$ as

$$t_j^{*b} = \frac{\hat{\beta}_j^{*b} - \ddot{\beta}_j}{\sqrt{\hat{V}_{jj}^{*b}}}, \tag{17}$$

   where $\hat{V}_{jj}^{*b}$ is the j^{th} diagonal element of $\hat{\boldsymbol{V}}^{*b}$. For the WCR bootstrap, the numerator of (17) can also be written as $\hat{\beta}_j^{*b}$, because $\ddot{\beta}_j = 0$.

5. For a one-tailed test, use the distribution of all the t_j^{*b} to compute the *lower-tail* or *upper-tail* P value,

$$\hat{P}_{\text{L}}^* = \frac{1}{B}\sum_{b=1}^{B} \mathbb{I}(t_j < t_j^{*b}) \quad \text{or} \quad \hat{P}_{\text{U}}^* = \frac{1}{B}\sum_{b=1}^{B} \mathbb{I}(t_j > t_j^{*b}), \tag{18}$$

   where $\mathbb{I}(\cdot)$ is the indicator function. For a two-tailed test, compute either the *symmetric* or the *equal-tail* P value,

$$\hat{P}_{\text{S}}^* = \frac{1}{B}\sum_{b=1}^{B} \mathbb{I}(|t_j| < |t_j^{*b}|) \quad \text{or} \quad \hat{P}_{\text{ET}}^* = 2\min(\hat{P}_{\text{L}}^*, \hat{P}_{\text{U}}^*). \tag{19}$$

   The former is appropriate if the distribution of t_j is symmetric around a mean of zero. In that case, if B is not very small, $\hat{P}_{\text{S}}^*$ and $\hat{P}_{\text{ET}}^*$ will be similar. If the symmetry assumption is violated, which it will be when $\hat{\beta}_j$ is biased, it is better to use $\hat{P}_{\text{ET}}^*$. Asymmetry becomes a greater concern when we extend the WCB to instrumental variables estimation in Section 6.1, because IV estimators are biased towards the corresponding OLS estimators.

We make a few observations on this algorithm:

- Whether the WCR or WCU bootstrap is used in step 3, the t-statistic calculated in step 4b tests a hypothesis, $\boldsymbol{\beta} = \ddot{\boldsymbol{\beta}}$, that is true *by construction* in the bootstrap samples, because $\ddot{\boldsymbol{\beta}}$ is used to construct the samples in step 4a. Since the bootstrap distribution is generated by testing a null hypothesis on samples from a DGP for which the null is correct, the resulting bootstrap distribution should mimic the distribution of the sample test statistic, t_j, when the null hypothesis of interest, $\beta_j = 0$, is also correct.

- Because the small-sample correction factor m defined in (13) and used in the CRVE (12) affects both t_j and the t_j^{*b} proportionally, the choice of m does not affect any of the bootstrap P values.

- The application of this algorithm does not produce standard errors, and `boottest` does not attempt to compute them. Instead, inference is based on P values and confidence sets; the latter are discussed below in Section 3.5. One could compute the standard deviation of the

bootstrap distribution of $\hat{\boldsymbol{\beta}}^{*b}$ and then use it for inference in several ways. However, this approach relies heavily on the asymptotic normality of $\hat{\boldsymbol{\beta}}$ in a case where large-sample theory may not apply. Higher-order asymptotic theory for the bootstrap (Davison and Hinkley, 1997, Chapter 5) predicts that this approach should not perform as well as the algorithm discussed above, and Monte Carlo simulations in (Cameron, Gelbach, and Miller, 2008) confirm this prediction.

- Given a significance level α, it is desirable to choose B such that $\alpha(B+1)$ is an integer (Davidson and MacKinnon, 2004, pp. 163–164). To see why, suppose that we violate this condition by setting $B = 20$ and $\alpha = 0.05$ when we are performing a one-sided test using $\hat{P}^*_{\mathrm{U}}$ in (18). We could form a sorted list containing the actual test statistic t_j and the bootstrap statistics t^{*b}_j; this list would have 21 members. If t_j and the t^{*b}_j obey the same distribution (which must be true asymptotically under the null, and we assume to be approximately true in finite samples), then t_j is equally likely to take any rank in the sorted list. As a result, $\hat{P}^*_{\mathrm{U}}$ can take on just 21 possible values, with equal probability: 0.0, 0.05, 0.10, ..., 1.0. Only the first of these would result in $\hat{P}^*_{\mathrm{U}} < 0.05$. Thus the probability of rejecting the null would be 1/21, and not the desired 1/20. But if instead we set $B = 19$, $\hat{P}^*_{\mathrm{U}}$ could take on 20 values, and the probability of rejecting the null would be exactly 1/20. For nearly the same reason, if we are using the equal-tail P value defined in (19), we should choose $\alpha(B+1)/2$ to be an integer.

- The argument in the previous paragraph implicitly assumes that t_j is equal to any of the t^{*b}_j with probability zero. However, there are only 2^G unique sets of draws for the Rademacher distribution. One of these has $v^{*b}_g = 1$ for all G, so that the corresponding bootstrap sample is the same as the actual sample. Therefore, each of the t^{*b}_j equals t_j with probability 2^{-G}.[1] When this happens, it is not entirely clear how to compute a bootstrap P value; see Davison and Hinkley (1997, Chapter 4) for a discussion of this issue. The most conservative approach is to change the strict inequalities in (18) and (19) into non-strict ones, which would cause the P value to be larger whenever t_j equaled any of the t^{*b}_j. `boottest` does not currently do so. To avoid the problem of having t_j equal any of the t^{*b}_j with non-trivial probability, it is better to use another auxiliary distribution instead of the Rademacher when G is small; see Section 3.4.

3.2 Imposing the null on the bootstrap data generating process

The bootstrap algorithm defined above lets the wild bootstrap DGP (15) either impose the restriction being tested (WCR) or not (WCU). Usually, it is better to impose the restriction. For any bootstrap DGP like (15) that depends on estimated parameters, those parameters are estimated more efficiently when restricted estimates are used (Davidson and MacKinnon, 1999). Intuitively, since inference involves estimating the probabilities of obtaining certain results under the assumption that the null is true, inference is improved by using bootstrap datasets in which the null in fact holds. Simulation evidence on this issue is presented in, among many others, Davidson and MacKinnon (1999) and Djogbenou, MacKinnon, and Nielsen (2018).

For this reason, `boottest` uses the restricted estimates $\tilde{\boldsymbol{\beta}}$ and restricted residuals $\tilde{\boldsymbol{u}}$ by default.[2] Nevertheless, `boottest` does allow the use of unrestricted estimates. This can be useful in two

[1] For symmetric tests, one of the unique sets of Rademacher draws has $v^{*b}_g = -1$ for all g, leading to $t^{*b}_j = -t_j$. Therefore, each of the $|t^{*b}_j|$ equals $|t_j|$ with probability 2^{1-G}.

[2] For the mechanics of restricted OLS in Stata, see [P] `makecns` and [R] `cnsreg`.

situations. First, WCU makes it possible to invert a hypothesis test to calculate confidence intervals for all parameters using just one set of bootstrap samples, whereas WCR requires constructing many sets of bootstrap samples to do so; see Davidson and MacKinnon (2004, Section 5.3) and Section 3.5. Thus, if the computational cost of WCR is prohibitive (although it rarely is with `boottest`), WCU is a practical alternative. Second, even when WCR *is* computationally manageable, WCU offers a robustness check. Often WCR P values modestly exceed WCU P values. When they are not close, this may indicate that the test results are unreliable (MacKinnon and Webb, 2017b); we will demonstrate such a case in Section 8.3.

3.3 General and multiple linear restrictions

The algorithm given above for wild cluster bootstrap inference is readily generalized to hypotheses involving any number of linear restrictions on $\boldsymbol{\beta}$. We can express a set of q such restrictions as

$$\mathrm{H}_0 : \boldsymbol{R}\boldsymbol{\beta} = \boldsymbol{r}, \tag{20}$$

where $\boldsymbol{R}$ and $\boldsymbol{r}$ are fixed matrices of dimensions $q \times k$ and $q \times 1$, respectively.

When $q = 1$, the t-statistic in (16) is replaced by

$$t = \frac{\boldsymbol{R}\hat{\boldsymbol{\beta}} - \boldsymbol{r}}{\sqrt{\boldsymbol{R}\hat{\boldsymbol{V}}\boldsymbol{R}'}}, \tag{21}$$

with $\boldsymbol{r}$ a scalar, and the bootstrap t-statistic in (17) is replaced by

$$t^{*b} = \frac{\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}})}{\sqrt{\boldsymbol{R}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}'}}. \tag{22}$$

For the WCR bootstrap, the numerator of expression (22) reduces to $\boldsymbol{R}\hat{\boldsymbol{\beta}}^{*b} - \boldsymbol{r}$ because $\boldsymbol{R}\ddot{\boldsymbol{\beta}} = \boldsymbol{R}\tilde{\boldsymbol{\beta}} = \boldsymbol{r}$. For the WCU bootstrap, it reduces to $\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \hat{\boldsymbol{\beta}})$. In both cases, the hypothesis being tested on the bootstrap samples is true in the bootstrap DGP.

When $q > 1$ in (20), the t-statistic (21) is replaced by the Wald statistic

$$W = (\boldsymbol{R}\hat{\boldsymbol{\beta}} - \boldsymbol{r})'(\boldsymbol{R}\hat{\boldsymbol{V}}\boldsymbol{R}')^{-1}(\boldsymbol{R}\hat{\boldsymbol{\beta}} - \boldsymbol{r}), \tag{23}$$

and the bootstrap t-statistic (22) is replaced by the bootstrap Wald statistic

$$W^{*b} = (\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}}))'(\boldsymbol{R}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}')^{-1}(\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}})). \tag{24}$$

The vector that appears twice in the quadratic form (24) is the numerator of (22), and the discussion that follows that equation applies here too. We will sometimes, by a slight abuse of terminology, call this vector the Wald numerator and the matrix that is inverted in (24) the Wald denominator. Because $W \geq 0$, only one type of bootstrap P value, namely, $\hat{P}^*_{\mathrm{U}}$, is relevant when $q > 1$. Thus the P value for the bootstrap Wald test is simply the fraction of the W^{*b} that exceed W.

3.4 The distribution of the auxiliary random variable

After equation (14), we noted that the auxiliary random variable v_g^{*b} that multiplies the residual vectors $\ddot{\boldsymbol{u}}_g$ in the bootstrap DGP—the "wild weight"—is usually drawn from the Rademacher distribution, taking the values -1 and $+1$ with equal probability. Under this distribution, $\mathrm{E}^*(v_g^{*b}) = 0$ and $\mathrm{E}^*((v_g^{*b})^2) = \mathrm{E}((v_g^{*b})^4) = 1$, so multiplying the residuals by v_g^{*b} ensures that the first, second,

and fourth moments of the residuals are preserved by the errors in the bootstrap samples. In fact, the Rademacher distribution is the only distribution that preserves first, second, and fourth moments. However, because its third moment is 0, it imposes symmetry on the distribution of the bootstrap error terms. Even if the elements of $\ddot{\boldsymbol{u}}_g$ are skewed, the elements of $\boldsymbol{u}_g^{*b}$ will not be. To that extent, the bootstrap errors become unrepresentative of the original sample errors. Unfortunately, there does not exist any auxiliary distribution that preserves all of the first four moments (MacKinnon, 2015, pp. 24–25).

Other distributions that have been proposed include:

1. The Mammen (1993) two-point distribution, which takes the value $1 - \phi$ with probability $\phi/\sqrt{5}$ and the value ϕ otherwise, where $\phi = (1 + \sqrt{5})/2$ is the golden ratio. One can confirm that $\mathrm{E}^*(v_g^{*b}) = 0$, and that $\mathrm{E}^*((v_g^{*b})^2) = \mathrm{E}^*((v_g^{*b})^3) = 1$. This means that the Mammen distribution preserves the first three moments of the residuals. Because its fourth moment is 2, it magnifies the kurtosis of the bootstrap errors. However, it does have the smallest kurtosis among all distributions with the same first three moments.

2. The Webb (2014) six-point distribution. As noted at the end of Section 3.1, the Rademacher distribution can yield only 2^G distinct bootstrap samples. This is true for all two-point distributions. The six-point distribution reduces, but does not totally eliminate, this problem by assigning probability 1/6 to each of 6 points, namely, $\pm\sqrt{1/2}$, $\pm\sqrt{2/2} = \pm 1$, and $\pm\sqrt{3/2}$. Its first three moments are 0, 1, and 0, like the Rademacher, so that it preserves first and second moments, and also imposes symmetry on the bootstrap errors. Its fourth moment is 7/6, which enlarges kurtosis only slightly.

3. The standard normal distribution, for which the first four moments are 0, 1, 0, and 3. This choice allows for an infinite number of possible bootstrap samples. It preserves the first two moments, but it imposes symmetry, and the large fourth moment greatly magnifies kurtosis.

4. The gamma distribution with shape parameter 4 and scale parameter 1/2, as suggested by Liu (1988). Like the Mammen distribution, this has third moment equal to 1. However, its fourth moment of 9/2 greatly enlarges kurtosis.

Simulation studies suggest that wild bootstrap tests based on the Rademacher distribution perform better than ones based on other auxiliary distributions; see, among others, Davidson, Monticini, and Peel (2007); Davidson and Flachaire (2008); Finlay and Magnusson (2016). However, the Webb six-point distribution is preferred to the Rademacher when G is less than 10 or perhaps 12. `boottest` offers all of these distributions and defaults to the Rademacher; see the `weighttype` option in Section 7.

3.5 Inverting a test to construct a confidence set

Any test of a hypothesis of the form $\boldsymbol{R}\boldsymbol{\beta} = \boldsymbol{r}$ implies a confidence set, which at level $1 - \alpha$ consists of all values of $\boldsymbol{r}$ for which $P > \alpha$. It is easiest to see this in the case of a single linear restriction of the form $\beta_j = \beta_{j0}$, where β_j is the j^{th} element of $\boldsymbol{\beta}$. The confidence set then consists of all values of β_{j0} for which the bootstrap P value for the test of $\beta_j = \beta_{j0}$ is equal to or greater than α.

Viewing a test as a mapping from values of β_{j0} to P values, the confidence set is the inverse image of the interval $[\alpha, 1]$. For bootstrap tests based on the algorithm of Section 3.1, this mapping is given by one of the bootstrap P values in step 5. When the bootstrap test that is inverted is based on a t-statistic, as all the tests we discuss are, the resulting interval is often called a

studentized bootstrap confidence interval. These intervals may be equal-tailed, symmetric, or one-sided, according to what type of bootstrap P value is used to construct them.

As discussed in Section 3.2, it is generally preferable to impose the null hypothesis when performing a bootstrap test. This is equally true when constructing a confidence interval. However, WCU does have a computational advantage over WCR in the latter case. For hypotheses of the form $\beta_j = \beta_{j0}$, inverting a bootstrap test means finding values of β_{j0} such that the associated bootstrap P values are equal to α. With WCU, finding these values is straightforward because, by definition, the WCU bootstrap DGP does not depend on the null hypothesis. Therefore, as we vary β_{j0}, the bootstrap samples do not change, and hence only one set of bootstrap samples needs to be constructed. Determining the bounds of the confidence set merely requires solving (16) for β_{j0} and plugging in the values for t_j that correspond to appropriate quantiles of the bootstrap distribution. For example, an equal-tailed studentized WCU bootstrap confidence interval is obtained by plugging in the $\alpha/2$ and $1-\alpha/2$ quantiles of the distribution of the t_j^{*b}, while a symmetric interval is obtained by plugging in the $1-\alpha$ quantile of the distribution of the $|t_j^{*b}|$.

For the WCR bootstrap, by contrast, the bootstrap samples depend on the null, and so they must be recomputed for each trial value of β_{j0}. In this case, it is essential for the convergence of the algorithm that the *same* set of v_g^{*b} values be used in each iteration. An iterative search in which each step requires a simulation could be computationally impractical. Fortunately, as we discuss in Section 5, the WCR bootstrap can be made to run extremely fast in many applications. Moreover, the search for bounds can be implemented so as to minimize re-computation by pre-computing all quantities in the WCR algorithm that do not depend on β_{j0}. Since `boottest` embodies this strategy, it is typically able to invert WCR bootstrap tests quickly. That said, it is often worthwhile to compute a WCU-based confidence interval too, since it provides a robustness check.

By default, `boottest` begins the search for confidence interval bounds by picking two trial values for β_{j0}, low and high, at which $P < \alpha$. `boottest` then calculates the P value at 25 evenly spaced points between these extreme bounds. From these 25 results, it selects those adjacent pairs of points between which the P value crosses α, and then finds the crossover points via an iterative search. In instrumental variables applications (Section 6.1), when identification is weak, the confidence set constructed in this way may consist of more than one disjoint segment, and these segments may be unbounded; see Section 8.4.

We have focused on the most common case, in which just one coefficient is restricted. However, `boottest` can invert any linear restriction of the form $\boldsymbol{R}\boldsymbol{\beta} = r$, in which $\boldsymbol{R}$ is now a $1 \times k$ vector, and r is a scalar. For example, if the restriction is that $\beta_2 = \beta_3$, $\boldsymbol{R}$ is a row vector with 1 as its second element, -1 as its third element, and every other element equal to 0. Under the null hypothesis, $r = 0$. In this case, the confidence set contains all values of r for which the bootstrap P values are equal to or greater than α.

4 Multi-way clustering

In Section 2, we assumed that error terms for observations in different clusters are uncorrelated. In some settings, this assumption is unrealistic. In modeling panel data at the firm level, for example, one might expect correlations both within firms over time and across firms at a given time. The variance and covariance patterns are typically unknown in both dimensions. Cameron, Gelbach, and Miller (2011) (CGM) and Thompson (2011) separately proposed a method for constructing cluster-robust variance matrices for the coefficients of linear regression models when the error terms

are clustered in two or more dimensions.[3] The theoretical properties of the multi-way CRVE were analyzed by MacKinnon, Nielsen, and Webb (2017), who also proposed and studied bootstrap methods for multi-way clustered data.

In the next subsection, we define the multi-way CRVE and discuss some practical considerations in computing it. Then, in Section 4.2, we discuss how to combine multi-way clustering with the wild bootstrap.

4.1 Computing the multi-way CRVE

`boottest` allows multi-way clustering along an arbitrary number of dimensions. For ease of exposition, however, we consider the two-way case. We index the G clusters in the first dimension by g and the H clusters in the second dimension by h. We attach subscripts G and H to objects previously defined for one-way clustering, such as $\boldsymbol{S}$ and $\boldsymbol{\Omega}$, to indicate which clustering dimension they are associated with. Similarly, the subscript "GH" indicates clustering by the intersection of the two primary clustering dimensions. The compound subscript "G, H" refers to the two-way clustered matrices proposed in CGM and Thompson (2011) and defined below.

The two-way CRVE extends the one-way CRVE in an intuitive manner. Recall from (6) that the one-way CRVE is built around the matrix $\hat{\boldsymbol{\Omega}}$, which has i, j entry equal to $\hat{u}_i\hat{u}_j$ if observations i and j are in the same cluster and 0 otherwise. The two-way CRVE is obtained by augmenting that definition: $\hat{\boldsymbol{\Omega}}_{G,H}$ is the matrix with i, j entry equal to $\hat{u}_i\hat{u}_j$ if observations i and j are in the same G-cluster or the same H-cluster, and 0 otherwise.

As a practical matter, we might try to compute $\hat{\boldsymbol{\Omega}}_{G,H}$ as $\hat{\boldsymbol{\Omega}}_G + \hat{\boldsymbol{\Omega}}_H$. But if observations i and j were in the same G-cluster *and* the same H-cluster, this would double count: entry i, j would be $2\hat{u}_i\hat{u}_j$. To eliminate this double counting, we instead compute

$$\hat{\boldsymbol{\Omega}}_{G,H} = \hat{\boldsymbol{\Omega}}_G + \hat{\boldsymbol{\Omega}}_H - \hat{\boldsymbol{\Omega}}_{GH}. \tag{25}$$

Replacing $\hat{\boldsymbol{\Omega}}$ in equation (6) by $\hat{\boldsymbol{\Omega}}_{G,H}$ gives us the two-way CRVE,

$$\hat{\boldsymbol{V}}_{G,H} = \hat{\boldsymbol{V}}_G + \hat{\boldsymbol{V}}_H - \hat{\boldsymbol{V}}_{GH}, \tag{26}$$

where we have used (25) and the linearity of (6) in $\hat{\boldsymbol{\Omega}}$.

By analogy with the standard Stata small-sample correction for the one-way case given in (13), CGM suggested redefining $\hat{\boldsymbol{V}}_{G,H}$ as follows:

$$\hat{\boldsymbol{V}}_{G,H} = m_G\hat{\boldsymbol{V}}_G + m_H\hat{\boldsymbol{V}}_H - m_{GH}\hat{\boldsymbol{V}}_{GH}, \tag{27}$$

where the three scalar m factors are computed as in (13). The number of clusters used in computing m_{GH} is the number of non-empty intersections between G and H clusters, which may be less than the product $G \cdot H$. CGM's `cgmreg` program uses these factors, as does the program `cgmwildboot` by Judson Caskey, which is derived from `cgmreg`. However, another popular implementation of multi-way clustering, `ivreg2` (Baum, Schaffer, and Stillman, 2007), takes the largest of m_G, m_H, and m_{GH} and uses it in place of all three. (The largest is always m_G or m_H.) This choice carries over to programs that work as wrappers around `ivreg2`, including `weakiv` (Finlay, Magnusson, and Schaffer, 2016), `xtivreg2` (Schaffer, 2010), and `reghdfe` (Correia, 2016).

`boottest` uses CGM's choices for the m factors in (27). One reason is that this helps to address a practical issue that arises in computing the multi-way CRVE. Computing the matrix $\hat{\boldsymbol{V}}_{G,H}$ in (27)

[3] CGM provided the ado package `cgmreg` to implement this method in Stata; see http://faculty.econ.ucdavis.edu/faculty/dlmiller/statafiles.

involves subtraction, so in finite samples the result is not guaranteed to be positive definite. When it is not, the Wald statistic can be negative, or the denominator of the t-statistic can be the square root of a negative number. Multiplying the negative term in (27) by a smaller factor, i.e. by m_{GH} instead of the larger of m_G and m_H, increases the chance that $\hat{\boldsymbol{V}}_{G,H}$ is positive definite.

Nevertheless, the possibility that $\hat{\boldsymbol{V}}_{G,H}$ is not positive definite still needs to be confronted. A fuller solution proposed by CGM starts by taking the eigendecomposition $\hat{\boldsymbol{V}}_{G,H} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}'$. Then it censors any negative eigenvalues in $\boldsymbol{\Lambda}$ to zero, giving $\boldsymbol{\Lambda}^+$, and builds a positive semi-definite variance matrix estimate:

$$\hat{\boldsymbol{V}}^+_{G,H} = \boldsymbol{U}\boldsymbol{\Lambda}^+\boldsymbol{U}'. \tag{28}$$

This solution, however, has some disadvantages. First, there appears to be no strong theoretical justification for (28). Second, $\hat{\boldsymbol{V}}^+_{G,H}$ is only positive semi-definite, not positive definite. So it does not guarantee that Wald and t-statistics will have the right properties. Third, replacing $\hat{\boldsymbol{V}}_{G,H}$ by $\hat{\boldsymbol{V}}^+_{G,H}$ may often be unnecessary, because the relevant quantity $\boldsymbol{R}\hat{\boldsymbol{V}}_{G,H}\boldsymbol{R}'$ in the denominator of the test statistic can be positive definite—which is what is needed—even when $\hat{\boldsymbol{V}}_{G,H}$ is not. Modifying $\hat{\boldsymbol{V}}_{G,H}$ in that case may unnecessarily change the finite-sample distribution of the test statistics. Fourth, as a byproduct of computational choices that dramatically increase speed, `boottest` never actually computes $\hat{\boldsymbol{V}}_{G,H}$; see Section 5. Of course, it would have to do that in order to compute the eigendecomposition, and consequently the computational burden of doing so would be substantial.

For these reasons, `boottest` does not modify $\hat{\boldsymbol{V}}_{G,H}$ in the manner of CGM. If this results in a test statistic that is invalid, the package simply reports the test to be infeasible.

4.2 Wild-bootstrapping the multi-way CRVE

The wild bootstrap and the multi-way CRVE appear to have been combined first in the Stata package `cgmwildboot` by Judson Caskey. Bringing them together raises several practical issues, in addition to those discussed in the context of the one-way wild cluster bootstrap.

The asymptotic properties of the multi-way CRVE are analyzed in MacKinnon, Nielsen, and Webb (2017). That paper also presents simulations which suggest that CRVE-based inference is unreliable in certain cases, including when the number of clusters in any dimension is small and/or the cluster sizes are very heterogeneous. It also discusses several methods for combining the wild and wild cluster bootstraps with the multi-way CRVE, proves their asymptotic validity, and shows by simulation that they can lead to greatly improved inferences.

The first practical issue that arises in wild-bootstrapping the multi-way CRVE is what to do when some of the bootstrap variance matrices, $\hat{\boldsymbol{V}}^{*b}$, are not positive definite. In their simulations, MacKinnon, Nielsen, and Webb (2017) apply the CGM correction (28) to these, too. In contrast, `boottest` merely omits instances where the test statistic is degenerate from the bootstrap distribution and decrements the value of B accordingly.[4] Like the CGM approach, deleting infeasible statistics from the bootstrap distribution is atheoretical. However, reassuringly, re-running the Monte Carlo experiments of MacKinnon, Nielsen, and Webb (2017) with the eigendecomposition procedure disabled suggests that the change has little effect in the cases examined in those experiments.[5]

The second practical issue is that, in contrast with the one-way case, the choice of small-sample correction factors now affects results. `boottest` applies CGM's proposed values for m_G, m_H, and

[4] For computational reasons, it is easier simply to omit these "degenerate" bootstrap samples than to replace them with new ones. However, it means that $\alpha(B+1)$ will probably not be an integer whenever any bootstrap samples have to be omitted.

[5] Stata code and results for the modified simulations are posted at `http://davidroodman.com/david/MNW2017.zip`.

m_{GH} in (27) also to each bootstrap sample for the reasons discussed above. Since each component of (27) is scaled by a different factor, the scaling affects the actual and bootstrap CRVEs differently. Although the impact is likely minor in most cases, it might not be when at least one of G and H is very small. An alternative would be to set all three factors to $\max\{m_G, m_H\}$, as `ivreg2` does (or would, if it were bootstrapped). If this were done for both the actual and bootstrap CRVEs, it would be equivalent to using no small-sample correction at all.

The third issue is that the elegant symmetry of the multi-way CRVE formula does not carry over naturally to the wild bootstrap. The wild cluster bootstrap is designed to preserve the pattern of correlations within each cluster for one-way clustering, but it cannot preserve the correlations in two or more dimensions at once. Therefore, we must now distinguish between the *error clustering(s)* and the *bootstrap clustering*. In bootstrapping, should we draw and apply one "wild weight" for each G-cluster, or for each H-cluster, or perhaps for each GH-cluster? The implementation in `boottest` supports all these choices and defaults to the last of them; see the `bootcluster()` option in Section 7.[6] Monte Carlo experiments in MacKinnon, Nielsen, and Webb (2017) suggest that wild bootstrap tests typically perform best when the bootstrap applies clustering in the dimension with fewest clusters.

However, even this strategy fails in at least one case, namely, in treatment models with few treated clusters, with or without a difference-in-differences structure. Here, the WCR bootstrap can dramatically underreject and the WCU bootstrap dramatically overreject. In this context, MacKinnon and Webb (2018) proposed turning to a *subcluster* bootstrap, in which the bootstrap error terms are clustered more finely than the CRVE. The subcluster bootstrap of course includes the ordinary wild bootstrap as a limiting case. We demonstrate the potential of this approach in Section 8.3.

5 Fast execution of the wild cluster bootstrap for OLS

It is easy to implement the one-way wild cluster bootstrap in Stata's ado programming language. However, this is computationally extremely inefficient. This section explains how to speed up the wild cluster bootstrap in the Stata environment. The efficiency gains make the wild cluster bootstrap feasible for datasets with millions of observations, even with a million bootstrap replications, and even when running the bootstrap test repeatedly in order to invert it and construct confidence sets. The main proviso is that the number of clusters for the bootstrap DGP should not be too large.

Some of the speed gain comes about by moving from Stata's ado programming language to its compiled Mata language. However, impressive gains are also obtained by carefully examining the matrix algebra and taking advantage of linearity and the associativity of matrix multiplication to reorder operations. The wild cluster bootstrap turns out to be especially amenable to such tricks.

To illustrate the computational methods employed by `boottest`, we use Stata's "nlsw88" dataset, which is an extract from the National Longitudinal Surveys of Young Women and Mature Women. We fit a linear regression model to `wage` (hourly wage) with covariates `tenure` (years in current job), `ttl_exp` (total work experience), `collgrad` (a dummy for college graduates), and a constant. There are 2217 observations, clustered into 12 industries. We test the null hypothesis that $\beta_{\texttt{tenure}}$, the coefficient on `tenure`, is equal to 0, against the alternative $\beta_{\texttt{tenure}} \neq 0$.

We first implement the WCR bootstrap test of this hypothesis in the ado language, as in CGM's

[6] The default of bootstrap clustering at the GH level was chosen for symmetry and because it works even when G or H is extremely small, not because it is generally the best choice.

"bs_example.do"[7] and `cgmwildboot`. To prepare the ground, the following code sets the random number seed (to ensure exact replicability) and loads the dataset. It then takes two steps to simplify subsequent programming: recoding the industry identifier to run sequentially from 1 to 12, in the new variable `clustid`, and then sorting the data by this variable:

```
set seed 293867483
webuse nlsw88, clear
drop if industry==. | tenure==.
egen clustid = group(industry)
sort clustid, stable
```

The next code block applies the WCR to test $\beta_{\texttt{tenure}} = 0$. The code imposes the null on the bootstrap DGP, takes Rademacher draws, runs $B = 999$ replications, and computes the symmetric, two-tailed P value, $\hat{P}^*_{\text{S}}$:

```
program define wild1
  syntax, b(integer) // get passed parameter, number of bootstrap replications

  quietly {
    regress wage tenure ttl_exp collgrad, cluster(industry) // full model
    scalar t = abs(_b[tenure] / _se[tenure]) // test statistic
    local G = e(N_clust) // number of clusters

    regress wage        ttl_exp collgrad // base for DGP: model with null imposed
    predict XB // restricted fit
    predict u, resid // restricted-fit residuals

    local exceedances 0
    forvalues i=1/`b' { // for each bootstrap replication...
      gen byte v = cond(runiform()<.5,1,-1) in 1/`G' // Rademacher draws -> first G rows of v
      gen ystar = XB + u * v[clustid] // bootstrap outcome
      regress ystar tenure ttl_exp collgrad, cluster(clustid) // replication regression
      drop v ystar

      if abs(_b[tenure] / _se[tenure]) > t {
        local `++exceedances' // count replication t statistics exceeding full-sample t
      }
    }
  }
  display _n "p value for _b[tenure]=0: " `exceedances'/`b' // symmetric, two-tailed p value
end
```

One subtlety in the code warrants explanation. In the line that generates `ystar`, the expression "`v[clustid]`" exploits the ado language's ability to treat a variable as a vector. For each observation, the expression obtains the value of `clustid`, between 1 and 12, and looks up the corresponding entry in `v`, whose first 12 rows hold the Rademacher draws for a given replication, one for each industry cluster.

Having prepared the data and code, we run the latter on the former:

```
. wild1, b(999)
p value for _b[tenure]=0: .2952953
```

When running Stata 15.1 on a single core of an Intel i7-8650 in a Lenovo laptop, this bootstrap test takes 7.19 seconds.

Next, we translate the algorithm rather literally into Mata. In contrast to the ado version, the Mata version must perform OLS and compute the CRVE itself, according to (3), (7), and (12). We will not explain every line; readers can consult the relevant Mata documentation:

```
mata set matastrict off
mata set matalnum off
mata set mataoptimize on
```

[7] Available at http://faculty.econ.ucdavis.edu/faculty/dlmiller/statafiles.

```
void wild2(real scalar B) { // takes one argument, the number of replications
  Y       = st_data(., "wage                                ")
  X       = st_data(., "     tenure ttl_exp collgrad        ")
  Xr      = st_data(., "            ttl_exp collgrad        ") // X for restricted model
  clustid = st_data(., "                                clustid")
  cons = J(rows(X),1,1); X = X, cons; Xr = Xr, cons
  info = panelsetup(clustid, 1) // Records start and stop obs numbers for each cluster
  k = cols(X); G = rows(info)

  R0 = 1,0,0,0 // R * beta = coefficient on tenure

  betahat = invsym(cross(Xr,Xr)) * cross(Xr,Y) // restricted OLS
  XB = Xr * betahat // restricted fit
  uhat = Y - XB // restricted fit residuals

  t = J(B+1,1,.) // 1st entry will be real t stat; remaining will be replication t stats
  for (i=1; i<=B+1; i++) {
    if (i>1) { // except for first iteration, construct a bootstrap Y
      v = 2*(runiform(G,1) :< .5) :- 1 // Rademacher draws
      Y = XB + uhat :* v[clustid] // bootstrap dependent variable
    }

    invXX = invsym(cross(X,X))
    betastarhat = invXX * cross(X,Y) // bootstrap regression fit
    ustarhat = Y - X * betastarhat // bootstrap regression residuals

    Gammahat = J(k,k,0) // will accumulate CRVE
    for (g=1; g<=G; g++) { // loop over clusters to compute CRVE
      u_g = panelsubmatrix(ustarhat, g, info)
      X_g = panelsubmatrix(X       , g, info)
      uX = cross(u_g,X_g)
      Gammahat = Gammahat + cross(uX,uX)
    }
    t[i] = abs(R0 * betastarhat) / sqrt(R0 * invXX * Gammahat * invXX * R0') // t stat

    printf("p value for _b[tenure]=0: %f", mean(t[1] :< t[|2\\.|]))
}
```

A trick introduced here is to run the replication loop $B+1$ times in order to compute the actual and bootstrap test statistics with the same code. In the first (extra) iteration, we set every element of $\boldsymbol{v}^*$ to 1, which forces $\boldsymbol{y}^{*1} = \boldsymbol{y}$.

We run this Mata version of the code with $B = 9999$ replications:

```
. wild2(9999)
p value for _b[tenure]=0: .289128913
```

Despite the tenfold increase in the number of replications, the run time falls by more than half, to 3.0 seconds. This shows how much more efficient it can be to use Mata compared with ado-based bootstrapping. To be fair, the slowness of the ado implementation is not pure inefficiency. For example, `regress` computes the matrix $(\boldsymbol{X}'\boldsymbol{X})^{-1}$ in order to estimate the parameter covariance matrix, but it does not estimate the parameters themselves via $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$ as in the Mata code. Rather, it solves $\boldsymbol{X}'\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{y}$, which is slower but more numerically stable when $\boldsymbol{X}$ is ill-conditioned (Gould, 2010). However, much of the work that `regress` does in our context is redundant, including a check for collinearity of the regressors in every bootstrap replication.

The code above is not fully optimized for Mata. We will gain further efficiency through careful analysis of the mathematical recipe. Our strategy includes these steps:

1. Consolidate the algorithm into a series of equations expressing the computations.
2. Combine steps by substituting earlier equations into later ones.
3. Use the linearity and associativity of matrix multiplication to reorder calculations.

Two main techniques fall under the heading of item 3:

a. Multiplying by thin, dimension-reducing matrices first. For example, if $\boldsymbol{A}$ is 1×100 and $\boldsymbol{B}$ and $\boldsymbol{C}$ are 100×100, computing $(\boldsymbol{AB})\boldsymbol{C}$ takes 20,000 scalar multiplications (and nearly as many additions) while $\boldsymbol{A}(\boldsymbol{BC})$ takes 1,010,000. More generally, since the final output of each replication is a scalar test statistic, constructing large, for example $N \times N$ or $N \times B$, matrices during the calculations is typically inefficient, because these matrices will eventually be multiplied by thin matrices to produce smaller ones. Reordering calculations can allow dimension-reducing multiplications to occur early enough so that the large matrices are never needed.

b. "Vectorizing" loops via built-in matrix operations. Mata programs are pre-compiled into universal byte code, and then, at run time, translated into environment-specific machine language. The two-step process produces code that runs more slowly than code produced in a single step by an environment-specific C or Fortran compiler. But Mata's built-in operators, such as matrix multiplication, and some of its functions, *are* implemented in C or Fortran and are fully pre-compiled, making them very fast. In Mata, for example, matrix multiplication is hundreds of times faster when using the built-in implementation than when running explicit `for` loops of scalar mathematical operations. Similarly, left-multiplication by $\boldsymbol{S}$, defined in Section 2, can be performed with the fast panelsum() function included in Stata since version 13.0.[8]

We next demonstrate how to perform the above steps in the context of the wild cluster bootstrap. First, we show that when $q = 1$ (leaving treatment of the case with $q \geq 1$ to Appendix A), each wild-bootstrap Wald statistic can be written as

$$\boldsymbol{v}^{*b\prime}\mathbf{a}(m\boldsymbol{v}^{*b\prime}\boldsymbol{B}\boldsymbol{B}'\boldsymbol{v}^{*b})^{-1}\mathbf{a}'\boldsymbol{v}^{*b},$$

where $\boldsymbol{v}^{*b}$ is the G-vector of wild weights, $\mathbf{a}$ is also $G \times 1$, and $\boldsymbol{B}$ is $G \times G$. The small matrices $\boldsymbol{a}$ and $\boldsymbol{B}$ are fixed across replications, and so only need to be computed once. With care, they can be built without creating large intermediate matrices. It is then convenient to compute all the bootstrap Wald numerators and denominators at once via $\mathbf{a}'\boldsymbol{v}^*$ and $\boldsymbol{B}'\boldsymbol{v}^*$, where $\boldsymbol{v}^*$ is the $G \times B$ matrix with b^{th} column equal to $\boldsymbol{v}^{*b}$.

The following equations consolidate the computations required to perform the b^{th} bootstrap replication (all of which were presented earlier). The equations start after the OLS regression of (1), which yields the residuals $\ddot{\boldsymbol{u}}$ and estimates $\ddot{\boldsymbol{\beta}}$:

$$\boldsymbol{u}^{*b} = \ddot{\boldsymbol{u}} :* \boldsymbol{S}'\boldsymbol{v}^{*b}, \qquad \text{(bootstrap errors)} \qquad (29)$$

$$\boldsymbol{y}^{*b} = \boldsymbol{X}\ddot{\boldsymbol{\beta}} + \boldsymbol{u}^{*b}, \qquad \text{(bootstrap sample)} \qquad (30)$$

$$\hat{\boldsymbol{\beta}}^{*b} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}^{*b}, \qquad \text{(bootstrap OLS fit)} \qquad (31)$$

$$\hat{\boldsymbol{u}}^{*b} = \boldsymbol{y}^{*b} - \boldsymbol{X}\hat{\boldsymbol{\beta}}^{*b}, \qquad \text{(bootstrap residuals)} \qquad (32)$$

$$\hat{\boldsymbol{V}}^{*b} = m(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'(\hat{\boldsymbol{u}}^{*b} :* \boldsymbol{S}'\boldsymbol{S} :* (\hat{\boldsymbol{u}}^{*b})')\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}, \qquad \text{(bootstrap CRVE)} \qquad (33)$$

$$W^{*b} = (\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}})'\boldsymbol{R}'(\boldsymbol{R}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}')^{-1}\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}}). \qquad \text{(bootstrap Wald statistic)} \qquad (34)$$

[8] This approach, which `boottest` currently takes, has the disadvantage that it requires the data to be sorted by cluster, which can be a costly operation in very large datasets. One could instead compute groupwise column sums without sorting, via a hash table.

Focusing first on the Wald numerator in (34), $\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}})$, observe that

$$\hat{\boldsymbol{\beta}}^{*b} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}^{*b} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'(\boldsymbol{X}\ddot{\boldsymbol{\beta}} + \boldsymbol{u}^{*b}) = \ddot{\boldsymbol{\beta}} + (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{u}^{*b}. \tag{35}$$

As a result, the numerator can be computed as

$$\begin{aligned}
\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}}) &= \boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{u}^{*b} && \text{(by (35))} \\
&= \boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'(\ddot{\boldsymbol{u}} :* \boldsymbol{S}'\boldsymbol{v}^{*b}) && \text{(by (29))} \\
&= (\boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' :* \ddot{\boldsymbol{u}}')\boldsymbol{S}'\boldsymbol{v}^{*b} && \text{(by (11))} \\
&= (\boldsymbol{S}(\ddot{\boldsymbol{u}} :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}'))'\boldsymbol{v}^{*b}. && \text{(by (8))}
\end{aligned} \tag{36}$$

These rearrangements flip the role of $\boldsymbol{S}$ in a way that reduces computation. The second and third lines left-multiply the vector $\boldsymbol{v}^{*b}$ by $\boldsymbol{S}'$, to expand it from height G to height N, then multiply the result by another large matrix. In contrast, the last line left-multiplies the tall matrix $\ddot{\boldsymbol{u}} :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}'$ by $\boldsymbol{S}$ in order to collapse it from length N to length G; and it does so *before* involving the G-vector $\boldsymbol{v}^{*b}$, the entries of which therefore need not be duplicated across observations within clusters.

Next, we vectorize expression (36) so as to compute the Wald numerators for all the bootstrap samples in a single algebraic calculation:

$$\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*} :- \ddot{\boldsymbol{\beta}}) = (\boldsymbol{S}(\ddot{\boldsymbol{u}} :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}'))'\boldsymbol{v}^{*}, \tag{37}$$

where $\hat{\boldsymbol{\beta}}^{*}$ is the $k \times B$ matrix whose columns are formed by the $\hat{\boldsymbol{\beta}}^{*b}$ vectors, and $\boldsymbol{v}^{*}$ is as defined earlier. Within the factor $\ddot{\boldsymbol{u}} :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}'$, multiplication should proceed from right to left, because $\boldsymbol{R}'$ is thin (it is $k \times q$, where here $q = 1$). Left-multiplication by the sparse matrix $\boldsymbol{S}$ is performed by using the Mata `panelsum()` function.

Equation (37) illustrates why the run time of the wild cluster bootstrap need not grow with NB, as one might expect it to. The calculations within the outer parentheses, which produce a G-vector, have to be performed only once, and their computational cost depends on N, but not B. The situation is reversed in the next step: The cost of multiplying this pre-computed G-vector by the $G \times B$ matrix $\boldsymbol{v}^{*}$ grows with GB, which of course depends on B, but not on N. Thus, provided G is not too large, it can be feasible to make B as high as 99,999 or even 999,999, regardless of sample size.

We turn now to the Wald denominator in (34). Using the identity (11) and substituting the formula for $\hat{\boldsymbol{V}}^{*b}$ in (33) into the Wald denominator in (34), we obtain

$$\begin{aligned}
\boldsymbol{R}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}' &= m\boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'(\hat{\boldsymbol{u}}^{*b} :* \boldsymbol{S}'\boldsymbol{S} :* (\hat{\boldsymbol{u}}^{*b})')\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' \\
&= m(\boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' :* (\hat{\boldsymbol{u}}^{*b})')\boldsymbol{S}'\boldsymbol{S}(\hat{\boldsymbol{u}}^{*b} :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}') \\
&= m\boldsymbol{J}^{*b'}\boldsymbol{J}^{*b},
\end{aligned} \tag{38}$$

where the $G \times q$ matrix $\boldsymbol{J}^{*b}$ is

$$\boldsymbol{J}^{*b} = \boldsymbol{S}(\hat{\boldsymbol{u}}^{*b} :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}'). \tag{39}$$

Again, matrix multiplications should be done from right to left.

When $q = 1$, the $\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}'$ term in (39) is a column vector, and we can vectorize (39) over all replications as

$$\boldsymbol{J}^{*} = \boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \hat{\boldsymbol{u}}^{*}), \tag{40}$$

$$(\boldsymbol{R}\hat{\boldsymbol{V}}\boldsymbol{R}')^{*} = m \cdot \text{colsum}(\boldsymbol{J}^{*} :* \boldsymbol{J}^{*}), \tag{41}$$

in which $\hat{\boldsymbol{u}}^*$ is the $N \times B$ matrix with typical column $\hat{\boldsymbol{u}}^{*b}$, $\boldsymbol{J}^*$ is the $G \times B$ matrix with typical column $\boldsymbol{J}^{*b}$, and, with some abuse of notation, $(\boldsymbol{R}\hat{\boldsymbol{V}}\boldsymbol{R}')^*$ is the $1 \times B$ vector of all the bootstrap Wald denominators.

This formulation is still not computationally efficient, however. In an intermediate step, it constructs the $N \times B$ matrix $\hat{\boldsymbol{u}}^*$, which can be impractical in large datasets. We therefore substitute the formula for $\hat{\boldsymbol{u}}^*$ into (40) and reorder certain operations. Note first, using (32), (35), and (30), that

$$\hat{\boldsymbol{u}}^{*b} = \boldsymbol{y}^{*b} - \boldsymbol{X}(\ddot{\boldsymbol{\beta}} + (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{u}^{*b}) = \boldsymbol{u}^{*b} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{u}^{*b} = \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}^{*b}, \tag{42}$$

where the orthogonal projection matrix $\boldsymbol{M}_{\boldsymbol{X}} = \mathbf{I} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ yields residuals from a regression on $\boldsymbol{X}$. Vectorizing over replications, (42) is $\hat{\boldsymbol{u}}^* = \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}^*$. Substituting this into (40), we find that

$$\begin{aligned}
\boldsymbol{J}^* &= \boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}^*) \\
&= \boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \boldsymbol{M}_{\boldsymbol{X}}(\ddot{\boldsymbol{u}} :* \boldsymbol{S}'\boldsymbol{v}^*)) && \text{(by (29))} \\
&= \boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \boldsymbol{M}_{\boldsymbol{X}})(\ddot{\boldsymbol{u}} :* \boldsymbol{S}'\boldsymbol{v}^*) && \text{(by (9))} \\
&= (\boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \boldsymbol{M}_{\boldsymbol{X}} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}')\boldsymbol{v}^*. && \text{(by (11))}
\end{aligned} \tag{43}$$

The last line, like equation (37) for the numerator, postpones multiplication by the $G \times B$ matrix $\boldsymbol{v}^*$ until everything else has been calculated.

As it stands, however, expression (43) is still computationally expensive, because $\boldsymbol{M}_{\boldsymbol{X}}$ is an $N \times N$ matrix. However, this problem is surmounted in the same way. We replace $\boldsymbol{M}_{\boldsymbol{X}}$ by $\mathbf{I} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ and rearrange one last time to find that

$$\begin{aligned}
\boldsymbol{J}^* &= \Big(\boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* (\mathbf{I} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}') :* \ddot{\boldsymbol{u}}')\boldsymbol{S}'\Big)\boldsymbol{v}^* \\
&= \Big(\boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \mathbf{I} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}' - \boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}' :* \ddot{\boldsymbol{u}}')\boldsymbol{S}'\Big)\boldsymbol{v}^*.
\end{aligned} \tag{44}$$

The first term can be simplified through the identities

$$\boldsymbol{S}(\boldsymbol{a} :* \mathbf{I} :* \boldsymbol{b}')\boldsymbol{S}' = \boldsymbol{S}\,\mathrm{diag}(\boldsymbol{a} :* \boldsymbol{b})\boldsymbol{S}' = \mathrm{diag}(\boldsymbol{S}(\boldsymbol{a} :* \boldsymbol{b})),$$

for $\boldsymbol{a}$ and $\boldsymbol{b}$ suitably conformable column vectors. The second term can be rearranged using the identities (9) and (10). We finally obtain

$$\boldsymbol{J}^* = \Big(\mathrm{diag}(\boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \ddot{\boldsymbol{u}})) - \boldsymbol{S}(\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{R}' :* \boldsymbol{X})(\boldsymbol{X}'\boldsymbol{X})^{-1}(\boldsymbol{S}(\ddot{\boldsymbol{u}} :* \boldsymbol{X}))'\Big)\boldsymbol{v}^*. \tag{45}$$

This formulation avoids the construction of large intermediate matrices and postpones multiplication by the $G \times B$ matrix $\boldsymbol{v}^*$ until the final step.[9] Equations (45), (41), and (37) therefore constitute an efficient algorithm for simultaneously computing the numerators and denominators for all the bootstrap test statistics.

The following Mata function applies these equations to the earlier example:

```
void wild3(real scalar B) {
  Y         = st_data(., "wage                                  ")
  X         = st_data(., "     tenure ttl_exp collgrad          ")
  Xr        = st_data(., "            ttl_exp collgrad          ")
  clustid   = st_data(., "                                clustid")
```

[9] In fact, `boottest`, unlike the sample Mata code presented in this section, does not construct the sparse, diagonal matrix indicated in (45), because that becomes inefficient as G increases. Rather, it computes the vector that forms the diagonal of that matrix and then subtracts it from the diagonal of the next term in (45), using an explicit `for` loop.

```
  cons = J(rows(X),1,1); X = X, cons; Xr = Xr, cons
  info = panelsetup(clustid, 1)
  N_G = rows(info) // number of clusters

  R0 = 1,0,0,0 // R0 * beta picks out coefficient on tenure

  uhat = Y - Xr * invsym(cross(Xr,Xr)) * cross(Xr,Y) // residuals from restricted model

  invXX = invsym(cross(X,X))
  v = 2*(runiform(N_G,B+1) :< .5) :- 1 // Rademacher weights for *all* replications
  v[,1] = J(N_G,1,1) // insert 1s in first column to reproduce full-sample regression

  XinvXXR = X * (invXX * R0')
  SXinvXXRu = panelsum(XinvXXR, uhat, info)
  numer = cross(SXinvXXRu, v) // all numerators
  J = (diag(SXinvXXRu) - panelsum(X, XinvXXR, info) * invXX * panelsum(X, uhat, info)') * v
  t = abs(numer) :/ sqrt(colsum(J :* J)) // all t stats

  printf("p value for _b[tenure]=0: %f", rowsum( t[1] :< t ) / B)
}
```

This code calls `panelsum()` to left-multiply by $\boldsymbol{S}$. In doing so, it takes advantage of the function's optional middle argument, which is a weight vector. Thus $\boldsymbol{S}(\ddot{\boldsymbol{u}} :* \boldsymbol{X})$ in (45) becomes not `panelsum(X:*uhat, info)`, but `panelsum(X, uhat, info)`, which is equivalent but slightly faster.

Returning to our example, we run the new function:

```
. mata wild3(999999)
p value for _b[tenure]=0: .290660291
```

The number of replications has gone up by an additional factor of one hundred, but the run time has dropped from 3.0 to 0.69 seconds. Compared to the original (ado) implementation, the final implementation is 10,000 times faster per replication.

Appendix A generalizes this discussion to include observation weights, one-way fixed effects, multi-way clustering, subcluster bootstrapping, inversion of the test to form confidence sets, and higher-dimensional hypotheses ($q > 1$). It shows, for example, how to avoid construction of large matrices even when the model contains fixed effects for a a grouping that is not congruent with the error clustering.

Having demonstrated how to speed up the wild cluster bootstrap dramatically, we end with a word of caution. The method developed here can backfire when a key design assumption—that there are relatively few clusters in the bootstrap DGP—is violated. For example, the $G \times B$ wild weight matrix $\boldsymbol{v}^*$ can become untenably large if G is of the same order of magnitude as N. In many contexts, if G is large, the bootstrap will be unnecessary, because large-sample theory will apply and tests based on standard distributions such as the t distribution will be reliable. However, that is not always the case. For example, in multi-way clustering, the last term on the right-hand side of (26) can contain a large number of clusters. Another case where it may be desirable to use a large value of G in the bootstrap DGP is for estimation of treatment effects, where, as discussed in Section 4.2, the subcluster wild bootstrap may be attractive.

`boottest` contains specially optimized code for the extreme "robust" case, in which $G = N$, but that does not help if the number of clusters is large yet smaller than N, as in the examples in the previous paragraph. Importantly, if the memory demands exceed available RAM, performance will degrade sharply as virtual memory is cached to disk. This can be prevented by using the `matsize()` option, which limits the memory demands, but not the actual size, of the $G \times B$ matrix $\boldsymbol{v}^*$; see Section 7.

6 Extensions to IV, GMM, and maximum likelihood

The tests that `boottest` implements are not limited to models estimated by OLS. In this section, we briefly discuss `boottest` implementations of the wild bootstrap in the context of other estimation methods.

6.1 The wild restricted efficient bootstrap for IV estimation

Davidson and MacKinnon (2010) (DM) proposed an extension of the wild bootstrap to linear regression models estimated with instrumental variables. We consider the model

$$\boldsymbol{y}_1 = \boldsymbol{Y}_2\boldsymbol{\gamma} + \boldsymbol{X}_1\boldsymbol{\beta} + \boldsymbol{u}_1, \tag{46}$$
$$\boldsymbol{Y}_2 = \boldsymbol{X}_1\boldsymbol{\Pi}_1 + \boldsymbol{X}_2\boldsymbol{\Pi}_2 + \boldsymbol{U}_2, \tag{47}$$

where $\boldsymbol{y}_1$, $\boldsymbol{Y}_2$, $\boldsymbol{X}_1$, and $\boldsymbol{X}_2$ are the observation vectors or matrices for the dependent variable, the endogenous (or instrumented) variables, the included exogenous variables, and the excluded exogenous variables (instruments), respectively. Equation (46) is the structural equation for $\boldsymbol{y}_1$, and equation (47) is the reduced-form equation for $\boldsymbol{Y}_2$. The matrices $\boldsymbol{Y}_2$, $\boldsymbol{\Pi}_1$, $\boldsymbol{\Pi}_2$, and $\boldsymbol{U}_2$ all have one column for each endogenous variable on the right-hand side of (46).

DM allowed for correlation between corresponding elements of $\boldsymbol{u}_1$ and $\boldsymbol{U}_2$, and for heteroskedasticity, but not for within-cluster dependence. We will instead consider the clustered case, to which DM's "wild restricted efficient" (WRE) bootstrap naturally extends (Finlay and Magnusson, 2016). Once again, we make no assumption about the within-cluster error variances and covariances. We merely assume that, if observations i and j are in different clusters, then $\mathrm{E}(u_{1i}u_{1j})$, $\mathrm{E}(u_{1i}\boldsymbol{u}_{2j})$, and $\mathrm{E}(\boldsymbol{u}_{2i}\boldsymbol{u}_{2j}')$ are identically zero, where u_{1i} is the i^{th} element of $\boldsymbol{u}_1$ and $\boldsymbol{u}_{2j}$ is the column vector made from the j^{th} row of $\boldsymbol{U}_2$.

Like the WCR bootstrap, the WRE bootstrap begins by fitting the model subject to the null hypothesis, which in DM is that $\boldsymbol{\gamma} = \boldsymbol{0}$. Imposing this restriction on (46) makes OLS estimation appropriate. DM suggested a technique for estimating the reduced form (47) which yields more efficient estimates than simply regressing $\boldsymbol{Y}_2$ on $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. The residual vector from the OLS fit of (46) is added as an extra regressor in (47). This explains the word "efficient" in the name of the procedure.

As implemented in `boottest`, the WRE bootstrap admits null restrictions other than $\boldsymbol{\gamma} = \boldsymbol{0}$. It can test any linear hypothesis involving any elements of $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. To achieve this generalization, `boottest` fits the entire system using maximum likelihood, subject to $\boldsymbol{R\delta} = \boldsymbol{r}$, where $\boldsymbol{\delta} = [\boldsymbol{\gamma}' \;\; \boldsymbol{\beta}']'$.[10] Because there is only one structural equation, the ML estimates of $\boldsymbol{\delta}$ are actually limited-information maximum likelihood (LIML) estimates. Appendix B derives the restricted LIML estimator used by `boottest`. Like classical LIML, this estimator can be computed analytically, without iterative searching. Since the estimates of $\boldsymbol{\Pi}_1$ and $\boldsymbol{\Pi}_2$ are also ML estimates, they are asymptotically equivalent to the efficient ones used in DM's procedure.

Having obtained estimates of all coefficients under the null hypothesis—that is, $\tilde{\boldsymbol{\gamma}}$, $\tilde{\boldsymbol{\beta}}$, $\tilde{\boldsymbol{\Pi}}_1$, and $\tilde{\boldsymbol{\Pi}}_2$—the WRE bootstrap works in the same way as the WCR bootstrap. It next computes the restricted-fit residuals,

$$\tilde{\boldsymbol{u}}_1 = \boldsymbol{y}_1 - (\boldsymbol{Y}_2\tilde{\boldsymbol{\gamma}} + \boldsymbol{X}_1\tilde{\boldsymbol{\beta}}),$$
$$\tilde{\boldsymbol{U}}_2 = \boldsymbol{Y}_2 - (\boldsymbol{X}_1\tilde{\boldsymbol{\Pi}}_1 + \boldsymbol{X}_2\tilde{\boldsymbol{\Pi}}_2).$$

[10] `boottest` can also use the unrestricted fit, as in the WCU bootstrap DGP after OLS, but simulation results in DM suggest that this is a very bad idea.

To generate each WRE bootstrap sample, these residuals are then multiplied by the wild weight vector $\boldsymbol{v}^{*b}$, which, in the standard one-way-clustered case, has one element per cluster. This preserves the conditional variances of the error terms, their correlations within clusters, and their correlations across equations. For the b^{th} bootstrap sample, the simulated values for all endogenous variables are then constructed as follows:

$$\boldsymbol{y}_1^{*b} = \boldsymbol{Y}_2^{*b}\tilde{\boldsymbol{\gamma}} + \boldsymbol{X}_1\tilde{\boldsymbol{\beta}} + \boldsymbol{u}_1^{*b}, \qquad \boldsymbol{u}_1^{*b} = \tilde{\boldsymbol{u}}_1 :* \boldsymbol{S}'\boldsymbol{v}^{*b}, \tag{48}$$

$$\boldsymbol{Y}_2^{*b} = \boldsymbol{X}_1\tilde{\boldsymbol{\Pi}}_1 + \boldsymbol{X}_2\tilde{\boldsymbol{\Pi}}_2 + \boldsymbol{U}_2^{*b}, \boldsymbol{U}_2^{*b} = \tilde{\boldsymbol{U}}_2 :* \boldsymbol{S}'\boldsymbol{v}^{*b}, \tag{49}$$

where it should be noted that $\boldsymbol{Y}_2^{*b}$ is actually generated before $\boldsymbol{y}_1^{*b}$.

After the bootstrap datasets have been constructed, various estimators can be applied. `boottest` currently supports two-stage least squares (2SLS), LIML, k-class, Fuller LIML, and generalized method of moments (GMM) estimators. Wald tests may then be performed for hypotheses about the parameters. The WRE also extends naturally to multi-way clustering and the subcluster bootstrap. `boottest` supports all these possibilities, except that, for GMM estimation, it does not recompute the moment-weighting matrix for each bootstrap replication.

Unfortunately, some of the techniques introduced in Section 5 to speed up execution do not work for IV estimators. The problem is that the bootstrap estimators are nonlinear in $\boldsymbol{v}^{*b}$. For example, if $\boldsymbol{Z}^{*b} = [\boldsymbol{Y}_2^{*b} \;\; \boldsymbol{X}_1]$ is the full collection of right-hand-side bootstrap data in the $\boldsymbol{y}_1^{*b}$ equation and $\boldsymbol{X} = [\boldsymbol{X}_1 \;\; \boldsymbol{X}_2]$ is the same for the $\boldsymbol{Y}_2$ equation, then the 2SLS estimator is given by

$$\hat{\boldsymbol{\delta}}^{*b} = \left((\boldsymbol{Z}^{*b})'\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{Z}^{*b}\right)^{-1}(\boldsymbol{Z}^{*b})'\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y}_1^{*b},$$

where $\boldsymbol{P}_{\boldsymbol{X}} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$. This estimator is nonlinear in $\boldsymbol{Z}^{*b}$, which contains $\boldsymbol{Y}_2^{*b}$, which is linear in $\boldsymbol{v}^{*b}$. The resulting overall nonlinearity prevents much of the reordering of operations that is possible for OLS.

This computational limitation does not apply to the Anderson-Rubin (AR) test, which is also available in `boottest`. To perform an AR test of the hypothesis that $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$, we subtract $\boldsymbol{Y}_2\boldsymbol{\gamma}_0$ from both sides of (46) and substitute for $\boldsymbol{Y}_2$ from (47):

$$\begin{aligned}\boldsymbol{y}_1 - \boldsymbol{Y}_2\boldsymbol{\gamma}_0 &= \boldsymbol{Y}_2(\boldsymbol{\gamma} - \boldsymbol{\gamma}_0) + \boldsymbol{X}_1\boldsymbol{\beta} + \boldsymbol{u}_1 \\ &= (\boldsymbol{X}_1\boldsymbol{\Pi}_1 + \boldsymbol{X}_2\boldsymbol{\Pi}_2 + \boldsymbol{U}_2)(\boldsymbol{\gamma} - \boldsymbol{\gamma}_0) + \boldsymbol{X}_1\boldsymbol{\beta} + \boldsymbol{u}_1 \\ &= \boldsymbol{X}_1(\boldsymbol{\Pi}_1(\boldsymbol{\gamma} - \boldsymbol{\gamma}_0) + \boldsymbol{\beta}) + \boldsymbol{X}_2\boldsymbol{\Pi}_2(\boldsymbol{\gamma} - \boldsymbol{\gamma}_0) + \boldsymbol{U}_2(\boldsymbol{\gamma} - \boldsymbol{\gamma}_0) + \boldsymbol{u}_1. \end{aligned} \tag{50}$$

If the instruments $\boldsymbol{X}_2$ are valid, it must be valid to apply OLS to (50), that is, to regress $\boldsymbol{y}_1 - \boldsymbol{y}_2\gamma_0$ on $\boldsymbol{X}_2$ while controlling for $\boldsymbol{X}_1$. If the hypothesis $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$ is correct *and* the $\boldsymbol{X}_2$ are valid instruments, then the coefficients on $\boldsymbol{X}_2$ in this regression model are all 0. The AR test is computed as the Wald test that the coefficients on $\boldsymbol{X}_2$ are all 0. It is then interpreted as a joint test of the hypothesis $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$ and of the overidentifying restrictions that the instruments are valid. Because the test does not involve *estimating* any coefficients on instrumented variables, it is robust to instrument weakness (Baum, Schaffer, and Stillman, 2007).

Note that the AR test is exact under classical assumptions because it tests rather than assumes the key condition of instrument validity. For this reason, it may seem odd to bootstrap the AR test. However, we have allowed equations (46) and (47) to have error terms that are heteroskedastic and/or correlated within clusters. Since the AR test is not exact under these conditions, bootstrapping it should generally improve its finite-sample properties.

When the WRE bootstrap is used to estimate the distribution of an AR test statistic, it varies only the left-hand side of (50) as it constructs bootstrap samples. In particular, using (48) and

(49), the bootstrap values of the left-hand side of (50) for cluster g are

$$\boldsymbol{y}_{1g}^{*b} - \boldsymbol{Y}_{2g}^{*b}\boldsymbol{\gamma}_0 = \tilde{\boldsymbol{y}}_1 - \tilde{\boldsymbol{Y}}_2\boldsymbol{\gamma}_0 + (\tilde{\boldsymbol{u}}_{1g} - \tilde{\boldsymbol{U}}_{2g}\boldsymbol{\gamma}_0)v_g^{*b}.$$

Therefore, the WRE bootstrap of an AR test can be implemented as a WCR bootstrap test of the hypothesis that the coefficients on $\boldsymbol{X}_2$ are all zero in regression (50).

We stress that the AR test has a different null hypothesis than the other tests we consider. It tests not only that the coefficients on $\boldsymbol{Y}_2$ take certain values, but also that the instruments are valid, in the sense that the overidentifying restrictions are satisfied. In consequence, its results need to be interpreted with great care, especially if the test is inverted in order to construct a confidence set for $\boldsymbol{\gamma}$. If the test tends to reject the null for most trial values of $\boldsymbol{\gamma}$, this may simply indicate that the instrument validity assumption should be rejected rather than that $\boldsymbol{\gamma}$ is being estimated with high precision; see Davidson and MacKinnon (2014).

6.2 The score bootstrap

The "score bootstrap" adapts the wild bootstrap to the class of *extremum estimators*, which includes maximum likelihood (ML) and generalized method of moments (GMM). The notion of a residual, which is central to the wild bootstrap, does not carry over directly to extremum estimators in general. Instead of working with residuals, the score bootstrap works with the contributions to the scores, which are the derivatives of the objective function with respect to the parameters. Hu and Zidek (1995) and Hu and Kalbfleisch (2000) showed how to apply the pairs bootstrap to scores. Kline and Santos (2012) applied the wild bootstrap instead, producing what they dub the score bootstrap.

Consider the probit model for binary data, $y_i = \mathbb{I}(\Phi(\boldsymbol{x}_i\boldsymbol{\beta}) > 0)$, where $\Phi(\cdot)$ is the cumulative standard normal distribution function. It would make no sense to apply the wild bootstrap to such a model, because the "residuals" would equal either $-\Phi(\boldsymbol{x}_i\hat{\boldsymbol{\beta}})$ or $1-\Phi(\boldsymbol{x}_i\hat{\boldsymbol{\beta}})$, depending on whether the dependent variable equaled 0 or 1.

Although there is no reason to use the score bootstrap in the context of the linear regression model in Section 2, it is illuminating to see how it would work. If, for estimation purposes, the error terms are assumed to be normally and independently distributed with variance σ^2, then the log-likelihood contribution of observation i is

$$\ell_i(\boldsymbol{\beta}, \sigma) = -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(y_i - \boldsymbol{x}_i\boldsymbol{\beta})^2. \tag{51}$$

The vector of derivatives of (51) with respect to $\boldsymbol{\beta}$ is $(y_i - \boldsymbol{x}_i\boldsymbol{\beta})\boldsymbol{x}_i/\sigma^2$. Evaluating this at $\ddot{\boldsymbol{\beta}}$ and summing over all observations yields the score vector

$$\boldsymbol{s} = \frac{1}{\sigma^2}\boldsymbol{X}'\ddot{\boldsymbol{u}} = \sum_{g=1}^{G}\boldsymbol{s}_g, \quad \boldsymbol{s}_g = \frac{1}{\sigma^2}\boldsymbol{X}_g'\ddot{\boldsymbol{u}}_g,$$

where $\boldsymbol{s}_g$ is the score subvector corresponding to the g^{th} cluster. Similarly, the negative of the Hessian matrix, the matrix of second derivatives of the log-likelihood for the entire sample, is $-\boldsymbol{H} = \boldsymbol{X}'\boldsymbol{X}/\sigma^2$.

Now consider the wild bootstrap Wald statistic (24), which can also be written as

$$W^{*b} = (\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}})'\boldsymbol{R}'(\boldsymbol{R}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}')^{-1}\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}}). \tag{52}$$

As noted in (36), the numerator of this statistic can be rewritten as $\boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{u}^{*b}$. Therefore,

$$\begin{aligned}\boldsymbol{R}(\hat{\boldsymbol{\beta}}^{*b} - \ddot{\boldsymbol{\beta}}) &= \boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\sum_{g=1}^{G}\boldsymbol{X}_g'\boldsymbol{u}_g^{*b} \\ &= \boldsymbol{R}(\boldsymbol{X}'\boldsymbol{X})^{-1}\sum_{g=1}^{G}(\boldsymbol{X}_g'\ddot{\boldsymbol{u}}_g)v_g^{*b} = -\boldsymbol{R}\boldsymbol{H}^{-1}\sum_{g=1}^{G}\boldsymbol{s}_g v_g^{*b}. \end{aligned} \tag{53}$$

From the rightmost expression here, we see that the v_g^{*b} are generating the bootstrap variation in the Wald numerator by perturbing the score contributions, $\boldsymbol{s}_g$. Because of the perturbations, the full set of bootstrapped score contributions, $\boldsymbol{s}^{*b} = \boldsymbol{X}'\boldsymbol{u}^{*b}$, is not itself a score matrix, i.e., it is not a set of observation-level derivatives of any log-likelihood. Nonetheless, the premise of the score bootstrap is that this randomness generates information about the distribution of the score vector from the actual sample, and hence of test statistics based upon it.

In the OLS case, the score bootstrap combines the Wald numerator (53), which is the same as in the wild bootstrap, with a somewhat different estimate of its variance that avoids reference to residuals and takes scores as primary quantities. This variance estimate enters the denominator of the bootstrap Wald or score/Lagrange multiplier statistic. To show the difference, we write the wild bootstrap CRVE as

$$\hat{\boldsymbol{V}}^{*b} = m(\boldsymbol{X}'\boldsymbol{X})^{-1}\left(\sum_{g=1}^{G}\boldsymbol{X}_g'\hat{\boldsymbol{u}}_g^{*b}(\hat{\boldsymbol{u}}_g^{*b})'\boldsymbol{X}_g\right)(\boldsymbol{X}'\boldsymbol{X})^{-1};$$

see (7) and (33). The score bootstrap instead uses

$$\hat{\boldsymbol{V}}^{*b} = m(\boldsymbol{X}'\boldsymbol{X})^{-1}\left(\sum_{g=1}^{G}\boldsymbol{X}_g'\boldsymbol{u}_g^{*b}(\boldsymbol{u}_g^{*b})'\boldsymbol{X}_g\right)(\boldsymbol{X}'\boldsymbol{X})^{-1} = m\boldsymbol{H}^{-1}\left(\sum_{g=1}^{G}\boldsymbol{s}_g^{*b}(\boldsymbol{s}_g^{*b})'\right)\boldsymbol{H}^{-1}.$$

Thus the bootstrap residuals from re-estimation on each bootstrap sample are dropped in favor of the bootstrap errors. The latter, when multiplied by $\boldsymbol{X}$ in the formula, constitute the bootstrap scores. In Kline and Santos (2012), $\boldsymbol{s}^{*b}$ is demeaned columnwise before entering this variance estimate; see Appendix A.3.

As Kline and Santos (2012) showed, this formulation generalizes straightforwardly to tests based on any extremum estimator for which cluster-level contributions to the score and the Hessian are computable. This allows us to use the computational tricks of Section 5 to calculate many score bootstrap statistics quickly. When the null hypothesis is imposed, the actual test statistic that corresponds to (52) is a score or Lagrange multiplier statistic; see Wooldridge (2002, eqn. 12.68).

The score bootstrap is computationally attractive, but it does not always provide a good approximation, for two reasons. First, as mentioned just above, in the OLS case the denominator of the score bootstrap test statistic uses bootstrap errors instead of bootstrap residuals to calculate the variance estimate, but the test statistic that is calculated on the original data uses residuals. Hence, the distribution of the score bootstrap test statistics cannot account for the use of residuals instead of errors in the variance estimate. Second, in nonlinear models, because the scores and the Hessian for all the bootstrap samples are evaluated at $\ddot{\boldsymbol{\beta}}$, rather than at estimates that vary across bootstrap samples, the score bootstrap cannot fully capture the nonlinearity of the estimator.

A dramatic example of the second issue can be found in Roodman and Morduch (2014), which replicated the Pitt and Khandker (1998) evaluation of microcredit programs in Bangladesh. The latter relied on a nonlinear, multi-equation ML estimator, and Roodman and Morduch (2014)

showed that the maximum likelihood estimator was bimodal. The previously unreported second mode corresponded to negative rather than positive impact, and Wald tests performed at either mode returned large and mutually contradictory t-statistics. In a pairs bootstrap, the second mode was favored in a third of the replications. Because the score bootstrap estimates test statistic distributions from the scores and Hessian computed at a single estimate, $\ddot{\boldsymbol{\beta}}$, it is incapable in this context of accurately representing the distribution of the parameter estimates of primary concern. We suggest that the score bootstrap not be relied upon without evidence that it works well in the case of interest.

7 The `boottest` command

The `boottest` package performs wild bootstrap tests of linear hypotheses. It is compatible with Stata versions back to 11.0, but it runs faster in Stata versions 13.0 and later because they include the Mata `panelsum()` function. The syntax is modeled on that of Stata's built-in command for Wald testing, `test`. Like `test`, but unlike other Stata implementations of the wild bootstrap, `boottest` is a post-estimation command. It determines the context for inference from the current estimation results.

The following three commands implement the extended example in Section 5:

```
webuse nlsw88, clear
regress wage tenure ttl_exp collgrad, cluster(industry)
boottest tenure
```

Here, by default, `boottest` generates 999 wild cluster bootstrap samples using the Rademacher distribution, with the null hypothesis imposed. It reports the t-statistic from the Wald test and its bootstrapped P value (by default, symmetric). It then automatically inverts the test, as described in Section 3.5, and reports the bounds of the confidence set for the default level of confidence, which is normally 95%. Finally, it plots the "confidence curve" underlying this calculation, that is, the bootstrap P value for the hypothesis $\beta_{\texttt{tenure}} = c$ as a function of c. It marks the points where the curve crosses 0.05, which are the limits of the confidence set.

In general, `boottest` accepts any hypothesis expressed in conformity with the syntax of Stata's `constraint define`. The hypothesis is stated before the comma in a `boottest` command line, and options come after. In the hypothesis definition(s), a simple reference to a regressor implies "`= 0`". Similarly, the joint hypothesis $\beta_{\texttt{tenure}} = \beta_{\texttt{ttl\_exp}} = 0$ could be tested by:

```
boottest tenure ttl_exp
```

However, because the null hypothesis is now two-dimensional, `boottest` no longer attempts to determine a confidence set or plot a confidence curve. Expressing more complex linear hypotheses requires the equals sign:

```
boottest 2*tenure + 3*ttl_exp = 4
```

To jointly test several complex hypotheses, each must be enclosed in parentheses:

```
boottest (tenure) (ttl_exp = 2)
```

More idiosyncratically, `boottest` allows the user to test hypotheses independently rather than jointly, using curly braces. The following example has the same effect as running "`boottest tenure`" and "`boottest ttl_exp = 2`" separately, except that it makes available the Bonferroni and Sidak adjustments for multiple hypothesis testing (on which, see `test`):

```
boottest {tenure} {ttl_exp = 2}, madjust(bonferroni)
```

Parentheses may be nested within curly braces in order to test several joint hypotheses separately:

```
boottest {(tenure) (ttl_exp=1)} {(tenure) (ttl_exp=2)}, madj(sidak)
```

The `boottest` command can be run after application of the following estimators:

1. OLS, as performed by `regress`.
2. Restricted OLS, as performed by `cnsreg`.
3. Instrumental variables estimators—2SLS, LIML, Fuller's LIML, k-class, or GMM—as fit with `ivregress` or the `ivreg2` package of Baum, Schaffer, and Stillman (2007). In all cases, the WRE bootstrap is applied by default. `boottest` initializes the bootstrap DGP with LIML, and in particular restricted LIML unless the user includes the `nonull` option. It then applies the user's chosen estimator on the bootstrap datasets. After GMM estimation, `boottest` bootstraps only with the final moment-weighting matrix; it does not replicate the process for computing that matrix.[11] By default, Wald tests are performed. However, the Anderson-Rubin test is available; its default hypothesis is that all instrumented variables have zero coefficients and that the overidentifying restrictions are satisfied. After 2SLS and GMM, the score bootstrap is also available.
4. OLS and linear IV estimators with one set of "absorbed" fixed effects, as fit with Stata's `areg` command, its `xtreg` and `xtivreg` commands with the `fe` option, or the user-written `xtivreg2` (Schaffer, 2010) or `reghdfe` (Correia, 2016).
5. Maximum likelihood (ML) estimators, as fit with commands such as `probit`, `logit`, `glm`, `sem`, `gsem`, and the user-written `cmp` (Roodman, 2011). Here, `boottest` offers only the score bootstrap. In order to re-estimate nonlinear models while imposing the null, `boottest` must modify and reissue the original estimation command. This requires that the estimation package accept the standard options `constraints()`, `iterate()`, and `from()`. The package must also support generation of scores via `predict`. Many ML-based estimation procedures in Stata meet these criteria. Because of the computational burden that is typical of ML estimation, `boottest` does not attempt to construct confidence sets by inverting score bootstrap tests.

`boottest` detects and accommodates the choice of variance matrix type used in the estimation procedure, be it homoskedastic, heteroskedasticity-robust, cluster-robust, or multi-way cluster-robust. It does the same for small-sample corrections, such as with the `small` option of `ivregress`. It also lets users *override* those settings during inference, accepting its own `robust`, `cluster()`, and `small` options. Thus, for example, tests after `regress` can be multi-way clustered even though that command does not itself support multi-way clustering. In fact, the `boottest` package includes the wrappers `waldtest` and `scoretest` to expose such functionality without requiring any bootstrapping. The following code shows two examples:

```
regress wage tenure ttl_exp collgrad, cluster(industry)
waldtest tenure, cluster(industry age)

probit c_city tenure wage ttl_exp collgrad, cluster(industry)
scoretest tenure
```

A third wrapper, `artest`, offers the non-bootstrapped Anderson-Rubin test:

```
ivregress 2sls wage ttl_exp collgrad (tenure = union), cluster(industry)
artest, cluster(industry age)
```

[11] It would be better to recompute the GMM weighting matrix in each replication according to whatever algorithm the researcher has chosen, but that has not been implemented.

The `boottest` command accepts the following options, nearly all of which are inherited by the wrapper commands:

`nonull` suppresses the imposition of the null hypothesis on the bootstrap DGP.

`reps(#)` sets the number of replications, B. The default is 999. `reps(0)` requests a Wald test or, if `boottype(score)` is also specified and `nonull` is not, a score test. The wrappers `waldtest` and `scoretest` facilitate this usage.

`seed(#)` sets the initial state of the random number generator; see `set seed`.

`ptype(symmetric | equaltail | lower | upper)` chooses the P value type from those listed in (18) and (19). It applies only to one-dimensional hypotheses. The default is `symmetric`.

`level(#)` specifies the confidence level, in percent, for any confidence sets derived; see `level`. The default is usually 95. Setting it to 100 suppresses computation and plotting of the confidence set while still allowing plotting of the confidence curve.

`madjust(bonferroni | sidak)` requests the Bonferroni or Sidak adjustment for multiple hypothesis tests. The Bonferroni P value is $\min(1, nP)$ where P is the unadjusted P value and n is the number of hypotheses. The Sidak P value is $1-(1-P)^n$.

`weighttype(rademacher | mammen | webb | normal | gamma)` specifies the distribution for the wild weights v_g^{*b}, from among the choices in Section 3.4. The default is `rademacher`. Under the Rademacher, if the number of replications B exceeds the number of possible draw combinations, 2^G, then `boottest` will use each possible combination once (enumeration).

`noci` prevents the automatic computation of a confidence interval when it would otherwise be computed. This can save a lot of time when the null is imposed and when B or the number of bootstrapping clusters is large.

`nograph` prevents graphing of the confidence function but not the derivation of the confidence set.

`small` requests finite-sample corrections even after estimates that did not make them.

`robust` and `cluster(`*varlist*`)` have the traditional meanings and override settings used in estimation. To cluster by several variables, list them all in `cluster()`.

`bootcluster(`*varname*`)` specifies the bootstrap clustering variable(s). If more than one variable is specified, then bootstrapping is clustered by the *intersections* of clustering implied by the listed variables. The default is to cluster by the intersection of all the `cluster()` variables, although this is generally not recommended with multi-way clustering; see Section 4.2. Note that `cluster(A B)` and `bootcluster(A B)` mean rather different things. The first is two-way clustering of the errors by `A` and `B`. The second is one-way clustering of the bootstrap errors by the intersections of `A` and `B`.

`ar` requests the Anderson-Rubin test. It applies only to instrumental variables estimation. The null defaults to all coefficients on endogenous (instrumented) variables being zero. If the null is specified explicitly, then it must fix all coefficients on instrumented variables, and no others.

`boottype(wild | score)` specifies the bootstrap type. After ML estimation, `score` is the default and only option. Otherwise, the wild or wild restricted efficient bootstrap is the default.

`quietly`, when working with ML estimates, suppresses display of the initial re-estimation with the null imposed.

`gridmin(#)`, `gridmax(#)`, and `gridpoints(#)` override the default lower and upper bounds and the resolution of the grid search that begins the process of determining bounds for confidence sets. By default, `boottest` estimates the lower and upper bounds by working with the bootstrap distribution, and initially samples 25 grid points. Reducing the number can save time in computationally demanding problems, at the cost of some crudeness in the confidence plot and the risk, if there is reason to expect the confidence set to be disjoint, of missing one or more pieces.

`graphname(`*name*`[, replace])` names any confidence plots. When multiple independent hypotheses are being tested, *name* will be used as a stub, producing *name*_1, *name*_2, and so on.

`graphopt(`*string*`)` allows the user to pass formatting options to the [G] **graph** command in order to control the appearance of the confidence plot.

`svmat[(t | numer)]` requests that the bootstrapped test statistics be saved in return value `r(dist)`. This can be diagnostically useful, since it allows analysis of the simulated distribution. Section 8.3 below provides an example. If `svmat(numer)` is specified, over-riding the default of `svmat(t)`, only the test statistic numerators are returned. If the null hypothesis is that a coefficient is zero, then these numerators are the estimates of that coefficient in all the bootstrap replications.

`cmdline(string)` provides `boottest` with the command line just used to generate the estimates. This is typically needed only when performing the score bootstrap after estimation with `ml model`. In order to impose the null on an ML estimate, `boottest` needs to re-run the estimation subject to the constraints imposed by the null. To do that, it needs access to the command line that generated the results. Most Stata estimation commands save the command line in the `e(cmdline)` return macro. However, if one uses Stata's `ml model` command, perhaps with a custom likelihood evaluator, no `e(cmdline)` is saved. The `cmdline(string)` option provides a work-around, by allowing the user to pass the estimation command line manually.

`matsize(#)` limits the memory demand of the $G \times B$ matrix $\boldsymbol{v}^*$ to prevent caching of virtual memory to disk. The limit is specified in gigabytes; e.g., `matsize(8)` would limit the memory demand to 8GB. Note that this option does not limit the actual size of $\boldsymbol{v}^*$. Instead, it forces `boottest` to break the matrix into chunks no larger than the limit, and then create and destroy each chunk in turn.

8 Empirical examples

To illustrate the methods available through `boottest`, we present four empirical examples adapted from published research.

8.1 OLS with few clusters

Gruber and Poterba (1994) studied whether tax incentives prompt self-employed people to buy health insurance. They used a difference-in-differences design to study the impact of the passage of the Tax Reform Act of 1986 in the U.S. The act extended a pre-existing tax subsidy for employer-provided insurance to self-employed people as well. Thus the employed served as the comparison group for the self-employed. The dataset spans the eight years 1983–1990, with the post-treatment period beginning in 1988, when the law came into full effect.

CGM revisited Gruber and Poterba (1994) using an aggregated version of the dataset with just 16 observations, one for each combination of year and employment type. They regressed the insured fraction on dummies for employment type, the post-treatment period, and the product thereof, with the last of these being the treatment dummy:

```
use http://faculty.econ.ucdavis.edu/~dlmiller/statafiles/collapsed
regress hasinsurance selfemployed post post_self, cluster(year)
```

This command estimates $\beta_{\texttt{post\_self}}$, the parameter of interest, to be 0.055 with a standard error of 0.0074 when clustering by year. To illustrate the wild cluster bootstrap, CGM tested the hypothesis $\beta_{\texttt{post\_self}} = 0.04$. The associated t-statistic is $(0.055 - 0.04)/0.0074 = 2.02$, which, when evaluated against the standard normal and the $t(6)$ distribution yielded P values of 0.043 and 0.090,

respectively. They reported that a wild bootstrap with 999 replications, Mammen weights, and no imposition of the null returned $P = 0.070$. We obtain fairly similar results as follows:

```
. waldtest post_self = .04, noci

Wald test:
  post_self = .04

                                        t(7) =     2.0194
                                  Prob>|t| =     0.0832

. boottest post_self=.04, weight(mammen) nonull noci seed(2309487)

Warning: with 8 Clusters, the number of replications, 999, exceeds the universe of
Mammen draws, 2^8 = 256.
Consider Webb weights instead, using weight(webb).

Wild bootstrap, null not imposed, 999 replications, Wald test, bootstrapping by year,
Mammen weights:
  post_self=.04

                                        t(7) =     2.0194
                                  Prob>|t| =     0.0480
```

In the first test, following Stata convention, `waldtest` uses the $t(G-1)$ distribution, and here there are $G = 8$ clusters. For the bootstrap test, `boottest` produces an interesting warning: although CGM used 999 replications, with eight clusters, the two-point Mammen distribution can only produce 256 different bootstrap samples.

We improve on this example by imposing the null, as advised by CGM and in Section 3.2, drawing from the six-point Webb distribution, increasing the number of bootstrap samples to $B = 999,999$, and inverting the test to derive a 95% confidence interval for $\beta_{\texttt{post\_self}}$:

```
. boottest post_self=.04, reps(999999) weight(webb)
.........................
Wild bootstrap, null imposed, 999999 replications, Wald test, bootstrapping by year,
Webb weights:
  post_self=.04

                                        t(7) =     2.0194
                                  Prob>|t| =     0.0756

95% confidence set for null hypothesis expression: [.03851, .07106]
```

8.2 OLS with multi-way clustering

Michalopoulos and Papaioannou (2013) (MP) investigated the effect of pre-colonial ethnic institutions on comparative regional development in African countries. They proxied regional variation in economic activity by satellite images of light density taken at night. In Section 4 of MP, geographic pixels of size 0.125×0.125 decimal degrees are the unit of analysis. Their specification is

$$y_{p,i,c} = a_c + \gamma \text{IQL}_i + \lambda \text{PD}_{p,i,c} + \boldsymbol{Z}'_{p,i,c}\Psi + \boldsymbol{X}'_{i,c}\Phi + \zeta_{p,i,c},$$

where $y_{p,i,c}$ represents economic activity in pixel p in the historical homeland of ethnicity i in country c. Controls include country-level fixed effects (a_c), log population density ($\text{PD}_{p,i,c}$), pixel-level variables ($\boldsymbol{Z}_{p,i,c}$), and ethnicity-country-level variables ($\boldsymbol{X}_{i,c}$). The predictor of interest is IQL_i, which represents the degree of jurisdictional hierarchy, beyond the local level, for ethnic institutions.

In the regression we replicate—from MP Table V, Panel A, column 10—the dependent variable is the log of average pixel luminosity in 2007–2008. The $N = 66,173$ observations come from

$G = 49$ countries and $H = 94$ ethno-linguistic groups. The two clustering dimensions have 295 non-empty intersections. We exactly reproduce this regression using the MP data file[12] and CGM's multi-way clustering command, `cgmreg`[13].

```
use pixel-level-baseline-final, clear
global pix lnkm pixpetro pixdia pixwaterd pixcapdist pixmal pixsead pixsuit pixelev pixbdist
global geo lnwaterkm lnkm2split mean_elev mean_suit malariasuit petroleum diamondd
global poly capdistance1 seadist1 borderdist1
encode pixwbcode, gen(ccode) // make numerical country identifier
cgmreg lnl0708s i.ccode centr_tribe lnpd0 $pix $geo $poly, cluster(ccode pixcluster)
```

The coefficient estimate of interest, for `cntr_tribe`, is 0.156. The standard error, two-way clustered by ethno-linguistic group and country, is 0.048, yielding a t-statistic (reported in MP as a z-statistic) of 3.228 for the null hypothesis that $\gamma = 0$. This implies a 95% confidence interval of $[0.061, 0.251]$.

As discussed in Section 4.2, there are three natural choices for the level of bootstrap clustering in this case: by ethno-linguistic group, by country, or by the intersection of the two. Simulation results in MacKinnon, Nielsen, and Webb (2017) favor clustering in the dimension with the fewest clusters, which, in this case, is the $G = 49$ countries. For illustration, we perform tests for all three levels. Also for illustration, we move to Stata's fixed-effect linear regression command, `areg`. The latter cannot perform multi-way clustering, so we do not bother to specify clustering during estimation. However, we can use the `waldtest` wrapper to calculate the appropriate P value. Additionally, we simply add the `cluster()` option to the `boottest` command lines. We then use the `bootcluster()` option to control the level of bootstrap clustering:

```
areg lnl0708s centr_tribe lnpd0 $pix $geo $poly, absorb(ccode)
waldtest centr_tribe, cluster(ccode pixcluster)
set seed 2309487 // for exact reproducibility of results
boottest centr_tribe, reps(9999) clust(ccode pixcluster) bootclust(ccode)
boottest centr_tribe, reps(9999) clust(ccode pixcluster) bootclust(pixcluster)
boottest centr_tribe, reps(9999) clust(ccode pixcluster) bootclust(ccode pixcluster)
```

The three calls to `boottest` shown above return 95% confidence intervals of $[0.055, 0.249]$, $[0.045, 0.264]$, and $[0.054, 0.249]$, respectively. These differ only slightly from the original $[0.061, 0.251]$. This is not surprising, because even the coarsest clustering dimension here contains 49 clusters. The number of clusters is probably large enough that the performance of non-bootstrap tests and confidence intervals is close to their asymptotic behavior.

8.3 Difference-in-differences with few treated clusters

Conley and Taber (2011) (CT) observed that, in difference-in-differences estimation with few treated clusters, the cluster-robust CRVE can be severely biased. In response, CT proposed a particular application of randomization inference. MacKinnon and Webb (2017b) showed that the wild cluster bootstrap also fails in this setting: The WCU bootstrap overrejects, and the WCR bootstrap underrejects. Later, MacKinnon and Webb (2018) proposed the subcluster bootstrap mentioned in Section 4.2 as a way to reduce the extent of the problem. We illustrate these findings in the context of an empirical example studied in CT.

CT studied the impact on college enrollment of state-level merit scholarship programs using data from Current Population Surveys for all states and the District of Columbia, covering 1989–2000.[14] During the study period, ten states adopted such programs. CT regressed an individual-level dummy for eventual college enrollment on dummies for sex, race, state, year, as well as the

[12] Available at http://econometricsociety.org/content/supplement-pre-colonial-ethnic-institutions-and-contemporary-african-development-0.

[13] http://web.archive.org/20170406170058/http://gelbach.law.upenn.edu/~gelbach/ado/cgmreg.ado.

[14] Data are available at http://economics.uwo.ca/people/conley_docs/code_to_download.html.

treatment variable, which is a dummy for the availability of state-level scholarships. One of the CT specifications evaluates the average impact of all ten states' programs, and another focuses on the most famous program, Georgia's HOPE, by dropping the other nine treatment states from the sample. In both cases, standard errors are clustered by state. If G_1 denotes the number of clusters in which treatment occurs, these models have $N = 42{,}161$, $G = 51$, and $G_1 = 10$ for the merit scholarship regressions and $N = 34{,}902$, $G = 42$, and $G_1 = 1$ for the HOPE scholarship regressions. Thus the HOPE regressions illustrate the extreme case in which there is just one treated cluster.

We first consider the model with ten states. We apply six variants of the wild cluster bootstrap to test the hypothesis that the scholarship programs were not associated with college enrollment. Three impose the null hypothesis, and three do not. Within each triplet, the first variant clusters the bootstrap errors by state, the level at which they are clustered in the CRVE. The second variant clusters the bootstrap errors by state-year combination, and the third variant clusters them by individual. Clustering by individual means not clustering at all, so this variant is actually the ordinary wild bootstrap.

```
infile coll merit male black asian year state chst using regm.raw, clear
set seed 3541641
regress coll merit male black asian i.year i.state, cluster(state)
gen individual = _n  // unique ID for each observation

boottest merit, reps(9999) gridpoints(10) // defaults to bootcluster(state)
boottest merit, reps(9999) gridpoints(10) nonull
boottest merit, reps(9999) gridpoints(10)         bootcluster(state year)
boottest merit, reps(9999) gridpoints(10) nonull bootcluster(state year)
boottest merit, reps(9999) gridpoints(10)         bootcluster(individual)
boottest merit, reps(9999) gridpoints(10) nonull bootcluster(individual)
```

The last two commands are especially demanding: Memory usage temporarily rises by about 4.5GB. The primary cause is the construction of a wild weight matrix $\boldsymbol{v}^*$ with one row for each of the 42,161 observations ("clusters" for bootstrapping purposes) and one column for each of the 9,999 replications[15]. When the null hypothesis is imposed, the computational burden of constructing and multiplying the large matrices is compounded by the search for confidence interval bounds, which requires re-running the bootstrap for many trial values (see Section 3.5). The "`gridpoints(10)`" options save a bit of time by instructing `boottest` to evaluate 10 instead of the default 25 evenly spaced values in its initial search for confidence interval bounds. The cost of that choice is that the confidence curve plots, which we do not look at here, are rougher. On the hardware referenced in Section 5, the penultimate command takes about 9 minutes.[16]

We next run the same tests for Georgia's HOPE program alone. This time, we condense the `boottest` commands into a loop, and take advantage of `boottest`'s `svmat` option to save the t-statistics from each bootstrap sample. To give insight into the degenerate behavior of the wild cluster bootstrap in this context, we plot histograms for all six distributions, as follows:

```
regress coll merit male black asian i.year i.state ///
  if !inlist(state,34,57,59,61,64,71,72,85,88), cluster(state)
local gr 0
foreach bootcluster in state "state year" individual {
  foreach nonull in "" nonull {
    boottest merit, reps(9999) gridpoints(10) bootcl(`bootcluster')
      `nonull' svmat
    mat dist = r(dist)
    svmat dist
    histogram dist1, xline(`r(t)') title("`bootcluster' `nonull'") ///
    xtitle("") ytitle("") ylabel(,angle(horizontal)) name(gr`++gr', replace)
    drop dist1
```

[15] Actually, it has 10,000 columns, because an extra one is inserted as in the model code in Section 5.

[16] In Stata/MP, using 4 processors cuts the run time to 6.5 minutes.

Table 1: Estimates, P values, and 95% confidence intervals for scholarship programs

Treatment group		Ten states		Georgia only
Estimate		0.0337		0.0722
Cluster-robust s.e.		0.0127		0.0108
t-statistic		2.664		6.713
P values and confidence intervals:	P val.	Conf. int.	P val.	Conf. int.
$t(G-1)$	0.010	$[0.008, 0.059]$	0.000	$[0.050, 0.094]$
Bootstrap by state, restricted	0.020	$[0.007, 0.067]$	0.495	$[-2.787, 1.307]$
Bootstrap by state, unrestricted	0.025	$[0.005, 0.063]$	0.000	$[0.050, 0.095]$
Bootstrap by state-year, restricted	0.037	$[0.003, 0.066]$	0.291	$[-5.375, 4.700]$
Bootstrap by state-year, unrestricted	0.041	$[0.002, 0.066]$	0.094	$[-0.013, 0.157]$
Bootstrap by individual, restricted	0.031	$[0.004, 0.064]$	0.390	$[-0.099, 0.246]$
Bootstrap by individual, unrestricted	0.033	$[0.003, 0.064]$	0.392	$[-0.096, 0.240]$

```
  }
}
graph combine gr1 gr2 gr3 gr4 gr5 gr6, imargin(zero) xcommon ycommon colfirst
```

Table 1 presents parameter estimates, cluster-robust standard errors, and t-statistics for the ten-state and Georgia-specific regressions, along with six bootstrap P values and the associated 95% confidence intervals.[17] For the ten-state regression, the P values and confidence intervals are similar to one another, which is to be expected given the numbers of clusters (51) and treated clusters (10). In contrast, the Georgia-only P values and confidence intervals vary radically. As predicted by the theory in MacKinnon and Webb (2017b, 2018), the WCR bootstrap test strongly rejects and the WCU bootstrap test does not when the bootstrap errors are clustered by state. Clustering the bootstrap errors by state-year intersections instead of by state reduces, but does not eliminate, the disagreement. However, "clustering" the bootstrap errors by individual (that is, using the ordinary wild bootstrap) brings the restricted and unrestricted bootstrap tests into very good agreement, with almost identical P values and confidence intervals.

The simulated distributions for the six Georgia-specific bootstrap t-statistics are displayed in Figure 1. In all six plots, a vertical line marks the actual t-statistic of 6.713. In the two left-most panels, we see the degenerate behavior of the bootstrap distribution with clustering at the state level: It is bimodal when the null is imposed (WCR), and unimodal and narrow when it is not imposed (WCU). Bootstrapping with finer clustering produces much more plausible-looking distributions. They suggest that it would not be unusual to obtain a t-statistic as large as 6.713 even if the HOPE program had no impact.

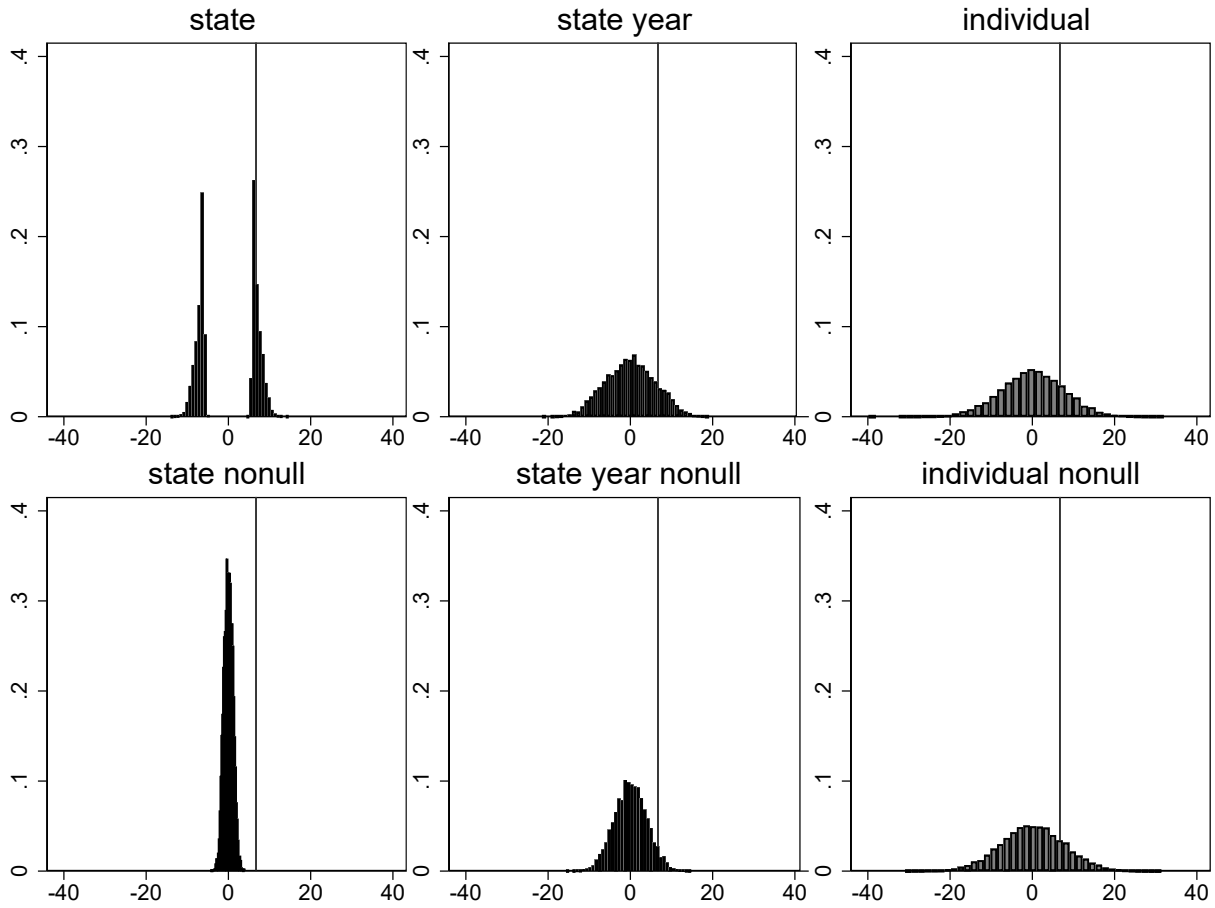
8.4 IV estimation

Levitt (1996) studied the short-term elasticity of the crime rate with respect to the incarceration rate in a U.S. state panel covering 1973–1993. To identify causal impacts, the study instrumented prisoners per capita with a set of dummies representing the somewhat arbitrarily timed progression of lawsuits over prison overcrowding. For example, if a court temporarily took control of a prison system, prison growth tended to slow at the onset of control and accelerate after control ended.

The definition of the overcrowding-lawsuit instrumental variables is complex, so we will omit most details here. Levitt (1996) divided the life cycle of an overcrowding lawsuit into six stages and

[17] The top rows of Table 1 match the results in column C of Tables 1 and 2 in CT.

Figure 1: Wild bootstrap distributions of t-statistic in Conley and Taber (2011) difference-in-differences evaluation of Georgia's HOPE program, varying the bootstrap clustering and whether the null is imposed



subdivided these into three substages. Starting from a dataset provided by Steven Levitt, which may not exactly match that used in the study, we computed and added these variables. We also pre-computed some per-capita variables and logarithms thereof.[18]

Separately for violent and property crime, Levitt regressed year-on-year log changes (`D.lViolentpop` and `D.lPropertypop` in our regressions) on the previous year's log change in prisoners per capita (`LD.lpris_totpop`). In the 2SLS regressions, the latter is instrumented with interactions of `stage` and `substage` as factor variables. State-level demographic and economic controls, all first-differenced, include log income per capita, unemployment, log police per capita, the metropolitan population fraction, the fraction that is black, and various age group fractions. Year and state dummies are included in the specification we copy here. Standard errors are robust to heteroskedasticity only.

This code produces our best replications of the original regressions, and runs some bootstrap tests:

```
use Levitt, clear
set seed 8723419
```

[18] Data and preparation code are available at http://davidroodman.com/david/Levitt.zip.

```
foreach crimevar in Violent Property {
  ivregress 2sls D.l`crimevar'pop ///
    (LD.lpris_totpop = ibnL.stage#i(1/3)L.substage) D.(lincomepop unemp lpolicepop ///
    metrop black a*pop) i.year i.state, robust

  boottest LD.lpris_totpop, clust(state year) bootclust(year) ptype(equaltail) ///
    gridmin(-2) gridmax(2) graphname(`crimevar', replace) ///
    graphopt(xtitle("") ytitle("") title(`crimevar' crime) ///
    ylabel(,angle(horizontal)))
}
graph combine Violent Property, imargin(tiny) xsize(5.25) ysize(2.5) iscale(*1.5)
```

From `ivregress`, we obtain elasticity estimates of −0.456 (standard error 0.170) for violent crime and −0.243 (0.106) for property crime, as opposed to −0.379 (0.180) and −0.261 (0.117) in Levitt (1996, Table VI, cols. 3 and 6). For a more extensive discussion of this study, see Roodman (2017).

The `boottest` command line above treats the specification in a more modern way. It two-way clusters the standard errors by state and year, and then bootstraps the distribution of the resulting t-statistic using the WRE (see Section 6.1). It applies the equal-tail P value, which is recommended for IV applications, in which bias toward the OLS estimate tends to make the confidence curve asymmetric (Davidson and MacKinnon, 2010). The bootstrap is clustered by the coarsest error-clustering dimension, which is the year. Surprisingly, this procedure produces 95% confidence sets that are disjoint and unbounded. The confidence set for violent crime is $(-\infty, 0.178] \cup [0.561, +\infty)$ and that for property crime is $(-\infty, -1.546] \cup [-0.681, 0.148] \cup [0.407, +\infty)$.

The confidence plots presented in Figure 2 show exactly how these confidence sets were obtained. The `boottest` command line above customizes their appearance using "`gridmin(2) gridmax(2)`" to set their horizontal bounds and `graphopt()` to pass options to the underlying `graph` command. For violent crime, only the interval between 0.178 and 0.561 does *not* belong to the 95% confidence set, because the bootstrap P values are less than .05 only in that interval. Similarly, for property crime, only the two intervals between −1.546 and −0.681 and between 0.148 and 0.407 do not belong to the 95% confidence set. These types of confidence sets may seem odd, but they can often arise when the instruments are weak; see Dufour (1997).

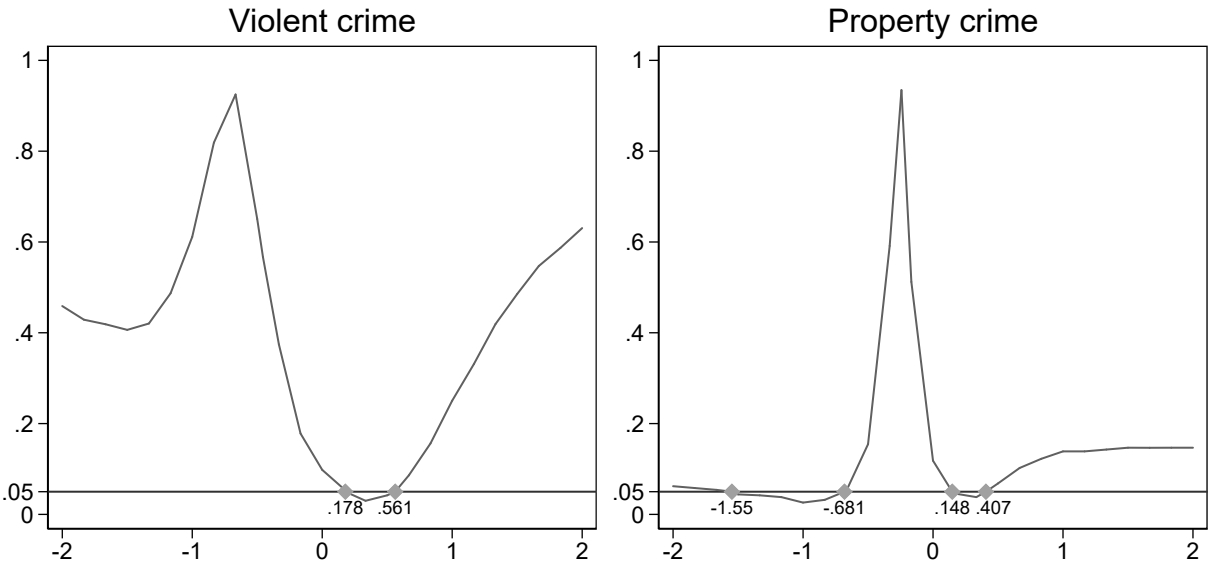
9 Conclusion

In this paper, we have discussed in some detail the Stata package `boottest`, which calculates several versions of the wild cluster bootstrap for tests of linear restrictions on linear regression models with heteroskedastic errors, one-way clustering, or multi-way clustering. The package also calculates the wild restricted efficient bootstrap (Davidson and MacKinnon, 2010) for models with one or more endogenous explanatory variables, with or without clustering, and the score bootstrap (Kline and Santos, 2012) for a variety of nonlinear models.

The mathematical structure of the wild cluster bootstrap, especially for models estimated by OLS, lends itself to efficient implementation, thus removing computational cost as a barrier to use. As we explained in Section 5, `boottest` takes advantage of this, and the code is remarkably fast, even when both the number of observations and the number of bootstrap replications are very large.

The empirical examples of Section 8 reinforce a well-known message from simulation studies (e.g., Cameron, Gelbach, and Miller, 2008; Davidson and MacKinnon, 2010; Finlay and Magnusson, 2016; Djogbenou, MacKinnon, and Nielsen, 2018): Results from wild bootstrap tests are often very similar to those from tests relying on large-sample theory, but not always. In particular, when the assumptions of asymptotic theory are far from being satisfied—for example, when there are few clusters, very unbalanced clusters, few treated clusters, or weak instruments—the bootstrap-based

Figure 2: Confidence curve for short-term elasticity of violent and property crime rate with respect to imprisonment rate, using a Wald test with the wild restricted efficient bootstrap, in the context of Levitt (1996) fixed-effects IV regressions



tests and the tests based on large-sample theory may result in very different inferences. In these cases, it is almost always preferable to rely on bootstrap-based tests, since they typically exhibit better finite-sample properties.

10 Acknowledgments

We thank Joshua Roxborough for research assistance. MacKinnon and Webb thank the Social Sciences and Humanities Research Council of Canada (SSHRC) for financial support. Nielsen thanks the Canada Research Chairs program, the SSHRC, and the Center for Research in Econometric Analysis of Time Series (CREATES, funded by the Danish National Research Foundation, DNRF78) for financial support.

References

Baum, C. F., M. E. Schaffer, and S. Stillman. 2007. Enhanced routines for instrumental variables/GMM estimation and testing. *Stata Journal* 7: 465–506.

Beran, R. 1986. Discussion: jackknife, bootstrap and other resampling methods in regression analysis. *Annals of Statistics* 14: 1295–1298.

Bertrand, M., E. Duflo, and S. Mullainathan. 2004. How much should we trust differences-in-differences estimates? *Quarterly Journal of Economics* 119: 249–275.

Bester, C. A., T. G. Conley, and C. B. Hansen. 2011. Inference with dependent data using cluster covariance estimators. *Journal of Econometrics* 165: 137–151.

Cameron, A. C., J. B. Gelbach, and D. L. Miller. 2008. Bootstrap-based improvements for inference with clustered errors. *Review of Economics and Statistics* 90: 414–427.

———. 2011. Robust inference with multiway clustering. *Journal of Business & Economic Statistics* 29: 238–249.

Carter, A. V., K. T. Schnepel, and D. G. Steigerwald. 2017. Asymptotic behavior of a t test robust to cluster heterogeneity. *Review of Economics and Statistics* 99: 698–709.

Conley, T. G., and C. R. Taber. 2011. Inference with "difference in differences" with a small number of policy changes. *Review of Economics and Statistics* 93: 113–125.

Correia, S. 2016. Linear models with high-dimensional fixed effects: an efficient and feasible estimator. Working paper, Duke University.

Davidson, J., A. Monticini, and D. Peel. 2007. Implementing the wild bootstrap using a two-point distribution. *Economics Letters* 96: 309–315.

Davidson, R., and E. Flachaire. 2008. The wild bootstrap, tamed at last. *Journal of Econometrics* 146: 162–169.

Davidson, R., and J. G. MacKinnon. 1993. *Estimation and Inference in Econometrics*. New York: Oxford University Press.

———. 1999. The size distortion of bootstrap tests. *Econometric Theory* 15: 361–376.

———. 2004. *Econometric Theory and Methods*. New York: Oxford University Press.

———. 2010. Wild bootstrap tests for IV regression. *Journal of Business & Economic Statistics* 28: 128–144.

———. 2014. Confidence sets based on inverting Anderson–Rubin tests. *Econometrics Journal* 17: S39–S58.

Davison, A. C., and D. V. Hinkley. 1997. *Bootstrap Methods and Their Application*. Cambridge, UK: Cambridge University Press.

Djogbenou, A. A., J. G. MacKinnon, and M. Ø. Nielsen. 2018. Asymptotic theory and wild bootstrap inference with clustered errors. QED Working Paper 1399, Queen's University, Department of Economics.

Dufour, J.-M. 1997. Some impossibility theorems in econometrics with applications to structural and dynamic models. *Econometrica* 65: 1365–1388.

Eicker, F. 1963. Asymptotic normality and consistency of the least squares estimators for families of linear regressions. *Annals of Mathematical Statistics* 34: 447–456.

Field, C. A., and A. H. Welsh. 2007. Bootstrapping clustered data. *Journal of the Royal Statistical Society Series B* 69: 369–390.

Finlay, K., L. Magnusson, and M. E. Schaffer. 2016. WEAKIV: Stata module to perform weak-instrument-robust tests and confidence intervals for instrumental-variable (IV) estimation of linear, probit and tobit models. Statistical Software Components, Boston College Department of Economics. URL https://ideas.repec.org/c/boc/bocode/s457684.html.

Finlay, K., and L. M. Magnusson. 2016. Bootstrap methods for inference with cluster sample IV models. Working paper, University of Western Australia.

Gould, W. 2010. Mata matters: Stata in Mata. *Stata Journal* 10: 125–142.

Gruber, J., and J. Poterba. 1994. Tax incentives and the decision to purchase health insurance: evidence from the self-employed. *Quarterly Journal of Economics* 109: 701–733.

Härdle, W., and E. Mammen. 1993. Comparing nonparametric versus parametric regression fits. *Annals of Statistics* 21: 1926–1947.

Hu, F., and J. D. Kalbfleisch. 2000. The estimating function bootstrap. *Canadian Journal of Statistics* 28: 449–481.

Hu, F., and J. V. Zidek. 1995. A bootstrap based on the estimating equations of the linear model. *Biometrika* 82: 263–275.

Kline, P., and A. Santos. 2012. A score based approach to wild bootstrap inference. *Journal of Econometric Methods* 1: 23–41.

Lee, C. H., and D. G. Steigerwald. 2017. Inference for clustered data. Working paper, University of California, Santa Barbara.

Levitt, S. D. 1996. The effect of prison population size on crime rates: evidence from prison overcrowding litigation. *Quarterly Journal of Economics* 111: 319–351.

Liang, K.-Y., and S. L. Zeger. 1986. Longitudinal data analysis using generalized linear models. *Biometrika* 73: 13–22.

Liu, R. Y. 1988. Bootstrap procedures under some non-I.I.D. models. *Annals of Statistics* 16: 1696–1708.

MacKinnon, J. G. 2009. Bootstrap hypothesis testing. In *Handbook of Computational Econometrics*, ed. D. A. Belsley and E. J. Kontoghiorghes, 183–213. Wiley.

———. 2013. Thirty years of heteroskedasticity-robust inference. In *Recent Advances and Future Directions in Causality, Prediction, and Specification Analysis*, ed. X. Chen and N. R. Swanson, 437–461. Springer.

———. 2015. Wild cluster bootstrap confidence intervals. *L'Actualité Économique* 91: 11–33.

MacKinnon, J. G., M. Ø. Nielsen, and M. D. Webb. 2017. Bootstrap and asymptotic inference with multiway clustering. QED Working Paper 1386, Queen's University, Department of Economics.

MacKinnon, J. G., and M. D. Webb. 2017a. Pitfalls when estimating treatment effects with clustered data. *The Political Methodologist* 24: 20–31.

———. 2017b. Wild bootstrap inference for wildly different cluster sizes. *Journal of Applied Econometrics* 32: 233–254.

———. 2018. The wild bootstrap for few (treated) clusters. *Econometrics Journal* 21: to appear.

Mammen, E. 1993. Bootstrap and wild bootstrap for high dimensional linear models. *Annals of Statistics* 21: 255–285.

Michalopoulos, S., and E. Papaioannou. 2013. Pre-colonial ethnic institutions and contemporary African development. *Econometrica* 81: 113–152.

Pitt, M. M., and S. R. Khandker. 1998. The impact of group-based credit programs on poor households in Bangladesh: does the gender of participants matter? *Journal of Political Econonomy* 106: 958–996.

Roodman, D. 2011. Fitting fully observed recursive mixed-process models with cmp. *Stata Journal* 11: 159–206.

———. 2017. The impacts of incarceration on crime. Working paper, Open Philanthropy Project.

Roodman, D., and J. Morduch. 2014. The impact of microcredit on the poor in Bangladesh: revisiting the evidence. *Journal of Development Studies* 50: 583–604.

Schaffer, M. E. 2010. XTIVREG2: Stata module to perform extended IV/2SLS, GMM and AC/HAC, LIML and k-class regression for panel data models. Statistical Software Components, Boston College Department of Economics. URL https://ideas.repec.org/c/boc/bocode/s456501.html.

Thompson, S. B. 2011. Simple formulas for standard errors that cluster by both firm and time. *Journal of Financial Economics* 99: 1–10.

Webb, M. D. 2014. Reworking wild bootstrap based inference for clustered errors. QED Working Paper 1315, Queen's University, Department of Economics.

White, H. 1980. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica* 48: 817–838.

Wooldridge, J. M. 2002. *Econometric Analysis of Cross Section and Panel Data.* Cambridge, MA: MIT Press.

Wu, C. F. J. 1986. Jackknife, bootstrap and other resampling methods in regression analysis. *Annals of Statistics* 14: 1261–1295.

Appendices

A Methods and formulas in `boottest`

In this appendix, we present additional details of the discussion in Section 5 about the methods and formulas in `boottest`. For example, we extend the computational framework to multi-way clustering, discuss how to speed up the inversion of one-dimensional tests to form confidence intervals, and describe the modifications needed to perform the score bootstrap efficiently.

A.1 Generalizing the wild cluster bootstrap for OLS

Section 5 presented a method for performing the wild cluster bootstrap that is optimized for the case of few clusters. Here we generalize that exposition to incorporate:

- parameter constraints under the maintained hypothesis, if any, in addition to those under the null hypothesis;
- null hypotheses of dimension $q > 1$;
- observation weights;
- one-way fixed effects;
- multi-way clustering;
- subcluster bootstrapping.

We adopt the following definitions:

- The constraints under the null hypothesis are given by $\boldsymbol{R}\boldsymbol{\beta} = \boldsymbol{r}$, and there are $q \geq 1$ of them. Additional *a priori* constraints under the maintained hypothesis are given by $\boldsymbol{R}_1\boldsymbol{\beta} = \boldsymbol{r}_1$, and there are $q_1 \geq 0$ of them. For example, if applying `boottest` after `cnsreg`, the restrictions imposed by `cnsreg` are $\boldsymbol{R}_1\boldsymbol{\beta} = \boldsymbol{r}_1$. The latter are sometimes referred to as the "model constraints."
- $\boldsymbol{W}$ is an $N \times N$ diagonal observation weighting matrix. If there are no weights, $\boldsymbol{W} = \mathbf{I}$.
- N_{FE} is the number of fixed effects, if any. $\boldsymbol{D}$ is the $N \times N$ matrix, left-multiplication by which partials out the fixed-effect dummies; that is, $\boldsymbol{D}$ demeans data within fixed-effect groups. If there are no fixed effects, $\boldsymbol{D} = \mathbf{I}$.
- Because data may be clustered in several ways, c subscripts are affixed to objects whose definitions depend on the choice of a particular clustering dimension for the errors (in the same way that we used G, H, and GH specifically for two-way clustering in Section 4). These objects include $\boldsymbol{S}$, the matrix that by left-multiplication creates cluster-wise sums; $\hat{\boldsymbol{\Omega}}$, the "estimate" of the clustered covariance matrix of the error terms; and $\hat{\boldsymbol{V}}$, the cluster-robust covariance matrix estimate for the coefficients. N_c is the number of clusters in clustering dimension c.

- The clustering-specific finite-sample adjustment factors m_c are understood to be negated for clustering intersections of odd parity, such as the GH clustering under the Section 4 notation. Thus we condense the multi-way CRVE formula (27) into $\sum_c m_c \hat{\boldsymbol{V}}_c$, where the sum is taken over all the different error clusterings and combinations thereof.
- The subscript c^* indicates the clustering used in the bootstrap DGP. It may differ (e.g., in the subcluster bootstrap) from all of the error clusterings indicated with a plain c subscript. In the one-way wild cluster bootstrap, there is only one value for c, which also equals c^*.

If there is a model constraint $\boldsymbol{R}_1\boldsymbol{\beta} = \boldsymbol{r}_1$, the restriction can equivalently be stated in terms of an unconstrained parameter, $\boldsymbol{\tau}$, as

$$\boldsymbol{\beta} = \boldsymbol{T}_1\boldsymbol{\tau} + \boldsymbol{t}_1, \tag{54}$$
$$\boldsymbol{T}_1 = \boldsymbol{R}_{1\perp}', \tag{55}$$
$$\boldsymbol{t}_1 = \boldsymbol{R}_1'(\boldsymbol{R}_1\boldsymbol{R}_1')^{-1}\boldsymbol{r}_1, \tag{56}$$

where $\boldsymbol{R}_{1\perp}'$ is a $k \times (k-q)$ matrix whose columns are orthogonal to the columns of $\boldsymbol{R}_1'$.[19] The parameterization (54) is such that $\boldsymbol{\tau}$ exactly parameterizes the affine subspace of admissible values for $\boldsymbol{\beta}$ under the constraints $\boldsymbol{R}_1\boldsymbol{\beta} = \boldsymbol{r}_1$. If there is no model constraint, then we take $\boldsymbol{T}_1 = \mathbf{I}$ and $\boldsymbol{t}_1 = \mathbf{0}$.

To implement the restricted bootstrap, we need to estimate the model not only under the model constraints, but also under the null and model constraints jointly. To this end, we define

$$\boldsymbol{R}_0 = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R} \end{bmatrix} \quad \text{and} \quad \boldsymbol{r}_0 = \begin{bmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r} \end{bmatrix},$$

so that, under both sets of constraints, $\boldsymbol{R}_0\boldsymbol{\beta} = \boldsymbol{r}_0$. We obtain $\boldsymbol{T}_0$ and $\boldsymbol{t}_0$ by analogy with $\boldsymbol{T}_1$ and $\boldsymbol{t}_1$.

With this notation, we can define the constrained, weighted, fixed-effects linear regression estimators

$$\begin{aligned}
\tilde{\boldsymbol{\beta}} &= \boldsymbol{T}_0(\boldsymbol{T}_0'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{T}_0)^{-1}\boldsymbol{T}_0'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{t}_0) + \boldsymbol{t}_0, && \text{(null imposed)} \\
\hat{\boldsymbol{\beta}} &= \boldsymbol{T}_1(\boldsymbol{T}_1'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{T}_1)^{-1}\boldsymbol{T}_1'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{t}_1) + \boldsymbol{t}_1, && \text{(null not imposed)} \\
\hat{\boldsymbol{\beta}}^{*b} &= \boldsymbol{T}_1(\boldsymbol{T}_1'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{T}_1)^{-1}\boldsymbol{T}_1'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}(\boldsymbol{y}^{*b} - \boldsymbol{X}\boldsymbol{t}_1) + \boldsymbol{t}_1. && \text{(bootstrap estimate)}
\end{aligned} \tag{57}$$

Generalizing the estimation framework in these ways forces some changes to the numerical recipe in Section 5. We omit the details, providing just the results for the Wald numerators and denominators. The expression for the Wald numerators in equation (37) now becomes

$$\boldsymbol{R}(\hat{\boldsymbol{\beta}}^* :- \ddot{\boldsymbol{\beta}}) = (\boldsymbol{S}_{c^*}(\ddot{\boldsymbol{u}} :* \boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}'))'\boldsymbol{v}^*, \tag{58}$$

where, for conciseness, we have defined

$$\boldsymbol{A} = (\boldsymbol{T}_1'\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{T}_1)^{-1}. \tag{59}$$

Up to a constant of proportionality, $\boldsymbol{A}$ is the inverse Hessian. Note that $\boldsymbol{D}$ and $\boldsymbol{S}_{c^*}$ are never in fact constructed in order to compute the numerators. Rather, left-multiplication by each is carried out through more direct manipulation of the data. For $\boldsymbol{S}_{c^*}$, this is done by summing columns within bootstrapping clusters, and for $\boldsymbol{D}$ it is done by demeaning columns within fixed-effect groups.

[19] In practice, the matrix $\boldsymbol{R}_{1\perp}'$ is constructed from an eigendecomposition; see [P] **makecns** for details.

More complications arise when computing the Wald denominators. To see why, we again start by assuming that $q = 1$. The penultimate statement of the formula for $\boldsymbol{J}^*$, a key factor in the denominators—see (44)—generalizes to

$$\boldsymbol{J}_c^* = \boldsymbol{K}_c^* \boldsymbol{v}^*, \tag{60}$$

$$\begin{aligned}\boldsymbol{K}_c^* &= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{D} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' - \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{X}'\boldsymbol{D}'\boldsymbol{W} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' \\ &= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{D} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' - \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{D}\boldsymbol{X})\boldsymbol{A}(\boldsymbol{S}_{c^*}(\ddot{\boldsymbol{u}} :* \boldsymbol{W}\boldsymbol{D}\boldsymbol{X}))'. \end{aligned} \tag{61}$$

Again, we omit details. Notice that these quantities are indexed by c; for multi-way clustering, they must be computed for all the different error clustering and combinations thereof.[20] The results are then summed in a revised equation (41) for the full set of Wald denominators:

$$(\boldsymbol{R}\hat{\boldsymbol{V}}\boldsymbol{R}')^* = \text{colsum}\left(\textstyle\sum_c m_c \boldsymbol{J}_c^* :* \boldsymbol{J}_c^*\right) = \text{colsum}\left(\boldsymbol{v}^* :* \left(\textstyle\sum_c m_c \boldsymbol{K}_c^{*\prime}\boldsymbol{K}_c^*\right)\boldsymbol{v}^*\right). \tag{62}$$

The last rearrangement in (62) is novel: It aims once more to defer involvement of the matrix $\boldsymbol{v}^*$, this time by summing over clusterings first. The computational costs for the two alternatives in (62) depend on the dimensions of the matrices involved: $\boldsymbol{K}_c^*$ is $N_c \times N_{c^*}$, and $\hat{\boldsymbol{v}}^*$ is $N_{c^*} \times B$. `boottest` chooses between the two versions based on an estimate of their relative computational costs.

Note that (61) introduces a new complication, namely, an isolated instance of the large $N \times N$ matrix $\boldsymbol{D}$ in the first term. Because it is not positioned as left-multiplying against a data matrix, we cannot avoid constructing it merely by interpreting its presence as demeaning a data matrix within fixed-effect groups. In Section 5, we did not allow for fixed effects, so there $\boldsymbol{D} = \mathbf{I}$, which we avoided constructing by invoking the identity $\boldsymbol{a} :* \mathbf{I} :* \boldsymbol{b}' = \text{diag}(\boldsymbol{a} :* \boldsymbol{b})$. However, that approach will not work now.

As we have done several times earlier, we will avoid constructing the troublesome matrix by substituting a formula for it, then rearranging the larger expression. In this case, the formula is

$$\boldsymbol{D} = \mathbf{I} - \boldsymbol{F}\boldsymbol{F}'\overline{\boldsymbol{W}},$$

where $\boldsymbol{F}$ is a data matrix of the fixed-effect dummies and $\overline{\boldsymbol{W}}$ is a diagonal matrix holding the weight shares of each observation within its fixed-effect group (or, if observations are not weighted, the reciprocal of the number of observations in its group). If we substitute for $\boldsymbol{D}$ in the first term in (61) and expand it, using (9) and (10), we obtain:

$$\begin{aligned}&\boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{D} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' \\ &\quad= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \mathbf{I} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' - \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{F}\boldsymbol{F}'\overline{\boldsymbol{W}} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' \\ &\quad= \boldsymbol{S}_c\text{diag}(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \ddot{\boldsymbol{u}})\boldsymbol{S}_{c^*}' - \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{F})(\boldsymbol{F}'\overline{\boldsymbol{W}} :* \ddot{\boldsymbol{u}}')\boldsymbol{S}_{c^*}' \\ &\quad= \boldsymbol{S}_c\text{diag}\left(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \ddot{\boldsymbol{u}}\right)\boldsymbol{S}_{c^*}' - \boldsymbol{S}_c\left(\boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}' :* \boldsymbol{F})(\boldsymbol{S}_{c^*}(\overline{\boldsymbol{W}}\ddot{\boldsymbol{u}} :* \boldsymbol{F})\right)'. \end{aligned} \tag{63}$$

In general, if $\boldsymbol{a}$ is a column vector, then the $N_c \times N_{\text{FE}}$ matrix $\boldsymbol{S}_c(\boldsymbol{a} :* \boldsymbol{F})$, the type of expression found twice on the right-hand side of (63), is a so-called *crosstab*: Each entry (i, j) is the sum of those elements of $\boldsymbol{a}$ belonging to the i^{th} cluster in clustering c and the j^{th} fixed-effect group. We

[20] If a function such as `panelsum()` is used to create left-multiplication by the $\boldsymbol{S}_c$ matrices, then the various matrices being summed must in some iterations be re-sorted according to a different clustering, which is a potentially costly operation; `panelsum()` requires sorted data. If the clusterings are not too fine, this cost can be reduced substantially by preliminarily collapsing (cluster-wise summing) these matrices to the level of the intersection of all the different error clusterings.

symbolize this crosstab by $\mathrm{CT}_{c,\mathrm{FE}}(\boldsymbol{a})$. It can be computed directly without constructing the large, sparse matrix $\boldsymbol{F}$. The left-hand term of (63) is a crosstab too, of $\boldsymbol{WDXAR}' :* \ddot{\boldsymbol{u}}'$ with respect to the clusterings c and c^*, i.e. $\mathrm{CT}_{c,c^*}(\boldsymbol{WDXAR}' :* \ddot{\boldsymbol{u}}')$. Drawing these threads together, we rewrite (61) as

$$\begin{aligned}\boldsymbol{K}_c^* = \mathrm{CT}_{c,c^*}(\boldsymbol{WDXAR}' :* \ddot{\boldsymbol{u}}) - \mathrm{CT}_{c,\mathrm{FE}}(\boldsymbol{WDXAR}')\mathrm{CT}_{c^*,\mathrm{FE}}(\overline{\boldsymbol{W}}\ddot{\boldsymbol{u}})' \\ - \boldsymbol{S}_c(\boldsymbol{WDXAR}' :* \boldsymbol{DX})\boldsymbol{A}(\boldsymbol{S}_{c^*}(\ddot{\boldsymbol{u}} :* \boldsymbol{WDX}))'. \end{aligned} \tag{64}$$

Notice that, because the residuals $\ddot{\boldsymbol{u}}$ come from an estimator that controls for the fixed effects, their (weighted) sum within each fixed-effect group is zero. If every fixed-effect group is in turn either equal to or contained in a single cluster under clustering c^*, then $\mathrm{CT}_{c^*,\mathrm{FE}}(\overline{\boldsymbol{W}}\ddot{\boldsymbol{u}}) = \boldsymbol{0}$, and the second term in (64) drops out. That possibility includes the common case of the fixed-effect grouping and the (bootstrap) error clustering coinciding. Of course, the term never arises if there are no fixed effects.

Finally, we deal with the complications introduced by allowing $q > 1$. Now the $q \times k$ matrix $\boldsymbol{R}$ is no longer a row vector. This does not hamper the calculation of the bootstrap Wald numerators in (58), which is unchanged. However, the formulas for the denominators lose meaning. The breakdown occurs in (40), which requires, as explained just before that equation, that $\boldsymbol{R}'$ is a column vector.

In general, a bootstrap Wald denominator, $\boldsymbol{R}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}'$, is a $q \times q$ matrix with $(d_1, d_2)^{\text{th}}$ element given by $\boldsymbol{R}_{d_1}\hat{\boldsymbol{V}}^{*b}\boldsymbol{R}'_{d_2}$, where the d subscripts identify rows of $\boldsymbol{R}$ that express individual constraints. A natural way to generalize the denominator formulas to higher-dimensional hypotheses is to double-subscript the $\boldsymbol{J}$ and $\boldsymbol{K}$ matrices for both error clustering and null constraint row, as follows:

$$\boldsymbol{J}_{cd}^* = \boldsymbol{K}_{cd}^*\boldsymbol{v}^*, \tag{65}$$

$$\begin{aligned}\boldsymbol{K}_{cd}^* = \mathrm{CT}_{c,c^*}(\boldsymbol{WDXAR}'_d :* \ddot{\boldsymbol{u}}) - \mathrm{CT}_{c,\mathrm{FE}}(\boldsymbol{WDXAR}'_d)\mathrm{CT}_{c,\mathrm{FE}}(\overline{\boldsymbol{W}}\ddot{\boldsymbol{u}})' \\ - \boldsymbol{S}_c(\boldsymbol{WDXAR}'_d :* \boldsymbol{DX})\boldsymbol{A}(\boldsymbol{S}_{c^*}(\ddot{\boldsymbol{u}} :* \boldsymbol{WDX}))', \end{aligned} \tag{66}$$

$$(\boldsymbol{R}\hat{\boldsymbol{V}}\boldsymbol{R}')^*_{d_1,d_2} = \mathrm{colsum}\left(\boldsymbol{v}^* :* \left(\textstyle\sum_c m_c \boldsymbol{K}^{*\prime}_{c,d_1}\boldsymbol{K}^*_{c,d_2}\right)\boldsymbol{v}^*\right); \tag{67}$$

c.f. (60), (61), and (62). Equation (67) produces a $1 \times B$ row vector of the $(d_1, d_2)^{\text{th}}$ elements of all the bootstrap Wald denominators. Since these denominators are symmetric, the triplet of formulas must be applied for each of the $q(q+1)/2$ independent entries in such matrices. In `boottest`, the denominator for each replication b is then constructed via explicit loops that extract and arrange the b^{th} elements from the row vectors in (67).

Once again, we have arrived at a set of formulas that together compute all the wild bootstrap Wald numerators and denominators while minimizing explicit looping and avoiding construction of large intermediate matrices. This time, the formulas allow for *a priori* linear constraints on the model, higher-dimensional null hypotheses, observation weights, fixed effects, multi-way clustering, and subcluster bootstrapping.

A.2 Inverting a test when imposing the null hypothesis

As explained in Section 3.5, we can form confidence sets by inverting any bootstrap test. In that section, we focused on tests of the hypothesis that $\beta_j = \beta_{j0}$. More generally, however, we need to invert a test for the linear restriction $\boldsymbol{R\beta} = \boldsymbol{r}$. (In practice, `boottest` can only do so when $q = 1$ and $\boldsymbol{r}$ is therefore scalar.) When a bootstrap DGP satisfies the restrictions that are being

tested, the bootstrap distribution must be recomputed for every trial value of $\boldsymbol{r}$, and this can be computationally demanding. Fortunately, however, the formulas for the wild bootstrap Wald statistic under OLS are essentially linear in $\boldsymbol{r}$. This opens the door to substantial efficiency gains.

The linearity can be seen by tracing through how variation in $\boldsymbol{r}$ affects the various quantities in our numerical recipe.

In analogy with equation (56), $\boldsymbol{t}_0$ is linear in $\boldsymbol{r}_0$, and thus in its subvector $\boldsymbol{r}$. In turn, by equation (57), $\tilde{\boldsymbol{\beta}}$ is linear in $\boldsymbol{t}_0$. Thus the corresponding estimation residuals $\ddot{\boldsymbol{u}}$ are, too. The bootstrap numerators and the $\boldsymbol{K}^*_{cd}$ factors in the denominators are also linear in $\ddot{\boldsymbol{u}}$; see equations (58) and (66). As a result, we can express all these quantities in the form $\boldsymbol{P} + \boldsymbol{Q}\boldsymbol{r}$, where $\boldsymbol{P}$ and $\boldsymbol{Q}$ are the same for all values of $\boldsymbol{r}$, and so only need be computed once.

Currently, after OLS (or after IV estimation when performing the Anderson-Rubin test), `boottest` exploits much, but not all, of this linearity.

A.3 The score bootstrap

Adapting the methods presented here to the score bootstrap (Kline and Santos, 2012) brings simplifications and one complication. For exposition, we will first consider how to compute the score bootstrap after OLS, even though we recommend not doing this in practice; see Section 6.2.

Section 6.2 showed how, with reference to OLS, the move from the wild bootstrap to the score bootstrap can mostly be captured with a straightforward observation and a small algebraic change. The observation is that the inverse Hessian, which should be available for any appropriately differentiable extremum estimator, is what we defined in equation (59) as $\boldsymbol{A}$, up to a factor of proportionality that affects the numerator and denominator equally and hence is irrelevant. Thus the inverse Hessian should be used wherever $\boldsymbol{A}$ appears in equations (58), (66), and (67). The algebraic change is to replace $\hat{\boldsymbol{u}}^{*b}$ by $\boldsymbol{u}^{*b}$ in (33), the equation for the bootstrap CRVE.

Following Kline and Santos (2012), we recenter (demean) the bootstrapped score contributions, $\boldsymbol{s}^{*b} = \boldsymbol{X}'\boldsymbol{u}^{*b}$, for each replication. This requires some changes to the computational recipe given in Appendix A.1. For the numerators, we rearrange (58) to clarify where the scores and Hessian enter as primary inputs:

$$\begin{aligned}\boldsymbol{R}(\hat{\boldsymbol{\beta}}^* \coloneq \ddot{\boldsymbol{\beta}}) &= (\boldsymbol{S}_{c^*}(\ddot{\boldsymbol{u}} :* \boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}'))'\boldsymbol{v}^* \\ &= (\boldsymbol{S}_{c^*}(\boldsymbol{W}(\ddot{\boldsymbol{u}} :* \boldsymbol{D}\boldsymbol{X})\boldsymbol{A}\boldsymbol{R}'))'\boldsymbol{v}^* = (\boldsymbol{S}_{c^*}(\boldsymbol{W}\boldsymbol{s}^*\boldsymbol{H}^{-1}\boldsymbol{R}'))'\boldsymbol{v}^*. \end{aligned} \tag{68}$$

For the denominators, recall from (42) that $\hat{\boldsymbol{u}}^{*b} = \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}^{*b} = \boldsymbol{u}^{*b} - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u}^{*b}$. This algebraic expansion is the ultimate source of the third term of (66); see (44). Since the score bootstrap replaces $\hat{\boldsymbol{u}}^{*b} = \boldsymbol{u}^{*b} - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{u}^{*b}$ with $\boldsymbol{u}^{*b}$, the third term of (66) becomes zero. Next, (39) is the initial formula for $\boldsymbol{J}^*$, which is a major factor in the denominator. If we generalize it for the possibilities considered in this appendix, we get

$$\boldsymbol{J}^{*b}_{cd} = \boldsymbol{S}_c(\hat{\boldsymbol{u}}^{*b} :* \boldsymbol{W}\boldsymbol{D}\boldsymbol{X}\boldsymbol{A}\boldsymbol{R}'_d) = \boldsymbol{S}_c(\boldsymbol{W}(\boldsymbol{u}^{*b} :* \boldsymbol{D}\boldsymbol{X})\boldsymbol{A}\boldsymbol{R}'_d) = \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}'_d),$$

in which the bootstrapped scores are given by $\boldsymbol{s}^{*b} = \boldsymbol{u}^{*b} :* \boldsymbol{D}\boldsymbol{X}$.

The mathematical step of recentering $\boldsymbol{s}^{*b}$ by demeaning it column-wise may seem trivial. However, once more we can find some computational gains. Let $\boldsymbol{\iota}$ be the $N \times 1$ column of 1s and $\boldsymbol{M}_{\boldsymbol{\iota}} = \mathbf{I} - \boldsymbol{\iota}(\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{\iota})^{-1}\boldsymbol{\iota}'\boldsymbol{W}$ be the associated orthogonal projection matrix that demeans data columns

while allowing for observation weights, $\boldsymbol{W}$. Then

$$\begin{aligned}\text{recentered } \boldsymbol{J}_{cd}^{*b} &= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{M}_{\boldsymbol{\iota}}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') \\ &= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') - \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{\iota}(\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{\iota})^{-1}\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') \\ &= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') - (\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{\iota})^{-1}\boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{\iota}\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') \\ &= \boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') - (\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{\iota})^{-1}\boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{\iota})\,\text{colsum}(\boldsymbol{W}\boldsymbol{s}^{*b}\boldsymbol{A}\boldsymbol{R}_d') \\ &= \boldsymbol{J}_{cd}^{*b} - \overline{\boldsymbol{w}}_c\,\text{colsum}(\boldsymbol{J}_{cd}^{*b}),\end{aligned} \tag{69}$$

where we have implicitly defined $\overline{\boldsymbol{w}}_c = (\boldsymbol{\iota}'\boldsymbol{W}\boldsymbol{\iota})^{-1}\boldsymbol{S}_c(\boldsymbol{W}\boldsymbol{\iota})$. This is the $N_c \times 1$ column vector whose entries are each c-cluster's share of the weight total—or, if there are no weights, each cluster's share of the number of observations.

Using (69), we once more vectorize over bootstrap replications and defer involvement of the large matrix $\boldsymbol{v}^*$. Expanding with (65),

$$\begin{aligned}\text{recentered } \boldsymbol{J}_{cd}^{*} &= \boldsymbol{K}_{cd}^{*}\boldsymbol{v}^{*} - \overline{\boldsymbol{w}}_c\,\text{colsum}(\boldsymbol{K}_{cd}^{*}\boldsymbol{v}^{*}) \\ &= (\boldsymbol{K}_{cd}^{*} - \overline{\boldsymbol{w}}_c\,\text{colsum}(\boldsymbol{K}_{cd}^{*}))\boldsymbol{v}^{*} = (\text{recentered } \boldsymbol{K}_{cd}^{*})\boldsymbol{v}^{*}\end{aligned} \tag{70}$$

Counterintuitively, the mathematical work of recentering the bootstrapped scores can in effect be carried out *before* the bootstrap.

To recap, we can execute the score bootstrap by computing the numerators using (68), computing the matrices $\boldsymbol{K}_{cd}^{*}$ using (66), recentering them as in (70), and then plugging the results into the formula for the bootstrap denominators in (67).

B An analytical solution for restricted LIML

The wild restricted efficient (WRE) bootstrap of Davidson and MacKinnon (2010) brings the wild bootstrap to instrumental variables estimation. As presented in Section 6.1, the WRE begins by re-estimating the model of interest subject to the constraints of the null hypothesis using restricted limited-information maximum likelihood (LIML). This appendix defines restricted LIML and derives a variant of the usual analytical solution for LIML that accommodates such constraints while still avoiding iterative search.

We begin by collecting the regressors from the structural equation (46) in the matrix $\boldsymbol{Z} = [\boldsymbol{Y}_2 \;\; \boldsymbol{X}_1]$ with coefficient $\boldsymbol{\delta} = [\gamma' \;\; \beta']'$. Given a restriction $\boldsymbol{R}\boldsymbol{\delta} = \boldsymbol{r}$, we can find a matrix $\boldsymbol{T}$ and a column vector $\boldsymbol{t}$ such that

$$\boldsymbol{\delta} = \boldsymbol{T}\boldsymbol{\tau} + \boldsymbol{t}, \tag{71}$$

where $\boldsymbol{\tau}$ is the unconstrained parameter under the restriction; see (54)–(56). The main complication in restricted LIML is that in general the restrictions involve coefficients on both endogenous and exogenous regressors, which violates the usual partitioning between the two groups of variables that is typically used explicitly in the estimation procedure.

Substituting (71) into (46) and rearranging, we obtain the model

$$\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t} = \boldsymbol{Z}\boldsymbol{T}\boldsymbol{\tau} + \boldsymbol{u}_1, \tag{72}$$

$$\boldsymbol{Y}_2 = \boldsymbol{X}\boldsymbol{\Pi} + \boldsymbol{U}_2. \tag{73}$$

The regressand in the structural equation is $\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t}$ and the regressor matrix is $\boldsymbol{Z}\boldsymbol{T}$. We assume that $[\boldsymbol{Y}_2 \;\; \boldsymbol{X}]$ has full column rank. To ensure identification, we also need $\text{rank}(\boldsymbol{Z}\boldsymbol{T}) \leq \text{rank}(\boldsymbol{X})$,

for which a sufficient condition is $\text{rank}(\boldsymbol{Z}) \leq \text{rank}(\boldsymbol{X})$. We can consolidate (72) and (73) into the structural form,

$$\boldsymbol{Y\Gamma} = \boldsymbol{XB} + \boldsymbol{U}, \tag{74}$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1 - \boldsymbol{Zt}\ \ \boldsymbol{Y}_2]$, $\boldsymbol{U} = [\boldsymbol{u}_1\ \ \boldsymbol{U}_2]$ with mean $\boldsymbol{0}$ and contemporaneous covariance matrix $\boldsymbol{\Sigma}$,

$$\boldsymbol{\Gamma} = \begin{bmatrix} 1 & \boldsymbol{0} \\ -\boldsymbol{T_Y\tau} & \boldsymbol{I} \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \boldsymbol{T_X\tau} & \boldsymbol{0} \\ \boldsymbol{\Pi} & \end{bmatrix}, \tag{75}$$

and $\boldsymbol{T} = [\boldsymbol{T}_{\boldsymbol{Y}}'\ \ \boldsymbol{T}_{\boldsymbol{X}}']'$. Note that the unrestricted model—and unrestricted LIML estimator—is obtained by setting $\boldsymbol{T} = \boldsymbol{I}$ and $\boldsymbol{t} = \boldsymbol{0}$, which implies $\boldsymbol{T_Y\tau} = \boldsymbol{\gamma}$ and $\boldsymbol{T_X\tau} = \boldsymbol{\beta}$.

Restricted LIML is the application of ML estimation of the structural parameter vector $\boldsymbol{\tau}$, or equivalently $\boldsymbol{\delta}$ in view of (71), in the normal linear model (72) and (73). Note that this formulation includes as a special case the most common restriction, namely, that the coefficient vector $\boldsymbol{\gamma}$ on the endogenous regressors is zero. In the latter case, (73) remains in the estimation model even though there are, in effect, no endogenous variables to instrument.

We proceed to derive the LIML estimator in the model given by (72) and (73) or, equivalently, by (74). The quasi-log-likelihood function for a general simultaneous equations system as in (74), after concentrating out $\boldsymbol{\Sigma}$, is given by[21]

$$\ell_c(\boldsymbol{\Gamma}, \boldsymbol{B}) = \frac{N}{2}(l+1)(1+\log 2\pi) + \frac{N}{2}\log\det(\boldsymbol{\Gamma}) - \frac{N}{2}\log\det(N^{-1}\boldsymbol{U}'\boldsymbol{U}), \tag{76}$$

where N is the number of observations and l is the number of variables in $\boldsymbol{Y}_2$ (i.e., $l+1$ is the number of equations in the system). From here forward, $\boldsymbol{U} = \boldsymbol{Y\Gamma} - \boldsymbol{XB}$ should not be interpreted as the true error terms but rather as functions of the data and parameters. In the model (74), as can be seen from (75), $\det(\boldsymbol{\Gamma}) = 1$. As a result, maximizing (76) is equivalent to minimizing the sum of squares, $\det(\boldsymbol{U}'\boldsymbol{U})$.[22]

Both the structural and reduced-form residuals are needed for the WRE bootstrap. We group the parameters by equation, into $\boldsymbol{\tau}$ and $\boldsymbol{\Pi}$, rather than by regressor type as above (into $\boldsymbol{\Gamma}$ and $\boldsymbol{B}$). We first concentrate out $\boldsymbol{\Pi}$. For a vector $\boldsymbol{a}$ and a matrix $\boldsymbol{A}$ with the same number of rows, we have the well-known matrix identity

$$\det\begin{bmatrix} \boldsymbol{a}'\boldsymbol{a} & \boldsymbol{a}'\boldsymbol{A} \\ \boldsymbol{A}'\boldsymbol{a} & \boldsymbol{A}'\boldsymbol{A} \end{bmatrix} = (\boldsymbol{a}'\boldsymbol{a})\det(\boldsymbol{A}'\boldsymbol{A} - \boldsymbol{A}'\boldsymbol{a}(\boldsymbol{a}'\boldsymbol{a})^{-1}\boldsymbol{a}'\boldsymbol{A}) = (\boldsymbol{a}'\boldsymbol{a})\det(\boldsymbol{A}'\boldsymbol{M}_{\boldsymbol{a}}\boldsymbol{A}). \tag{77}$$

Applying this identity to $\det(\boldsymbol{U}'\boldsymbol{U})$ we find that

$$\begin{aligned} \det(\boldsymbol{U}'\boldsymbol{U}) &= (\boldsymbol{u}_1'\boldsymbol{u}_1)\det(\boldsymbol{U}_2'\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{U}_2) \\ &= (\boldsymbol{u}_1'\boldsymbol{u}_1)\det((\boldsymbol{Y}_2 - \boldsymbol{X\Pi})'\boldsymbol{M}_{\boldsymbol{u}_1}(\boldsymbol{Y}_2 - \boldsymbol{X\Pi})) \\ &= (\boldsymbol{u}_1'\boldsymbol{u}_1)\det((\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{Y}_2 - \boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{X\Pi})'(\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{Y}_2 - \boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{X\Pi})). \end{aligned} \tag{78}$$

The first factor in (78), $\boldsymbol{u}_1'\boldsymbol{u}_1$, does not depend on $\boldsymbol{\Pi}$. Since $\boldsymbol{\Pi}$ is unconstrained, even in restricted LIML, and the right-hand side of (73) contains the same regressors in each equation, the second factor in (78) can be minimized by equation-by-equation OLS regressions of $\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{Y}_2$ on $\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{X}$. In practice, using the Frisch-Waugh-Lovell Theorem, this is done by regressing $\boldsymbol{Y}_2$ on $\boldsymbol{X}$ and $\ddot{\boldsymbol{u}}_1$, the vector of LIML structural residuals, after the latter have been computed, possibly subject to

[21] See Davidson and MacKinnon (1993, Chapter 18) for a general reference.

[22] Were there only one equation in the system, this objective would reduce to the sum of the squared residuals. In this sense, LIML is the *one-stage* least squares IV estimator.

constraints on $\boldsymbol{\delta}$. The resulting estimates of $\boldsymbol{\Pi}$, say $\ddot{\boldsymbol{\Pi}}$, are then used to compute the reduced-form residuals as $\boldsymbol{Y}_2 - \boldsymbol{X}\ddot{\boldsymbol{\Pi}}$.

The minimized residuals from regressing the columns of $\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{Y}_2$ on $\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{X}$ are given by the expression $\boldsymbol{M}_{\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{X}}\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{Y}_2 = \boldsymbol{M}_{\boldsymbol{X},\boldsymbol{u}_1}\boldsymbol{Y}_2$, which again follows from the Frisch-Waugh-Lovell Theorem. Therefore,

$$\min_{\boldsymbol{\Pi}} \det(\boldsymbol{U}'\boldsymbol{U}) = (\boldsymbol{u}_1'\boldsymbol{u}_1)\det(\boldsymbol{Y}_2'\boldsymbol{M}_{\boldsymbol{u}_1}'\boldsymbol{M}_{\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{X}}\boldsymbol{M}_{\boldsymbol{u}_1}\boldsymbol{Y}_2) = (\boldsymbol{u}_1'\boldsymbol{u}_1)\det(\boldsymbol{U}_2'\boldsymbol{M}_{\boldsymbol{X},\boldsymbol{u}_1}\boldsymbol{U}_2). \tag{79}$$

Applying again (77) with $\boldsymbol{a} = \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}_1$ and $\boldsymbol{A} = \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{U}_2$, (79) becomes

$$\min_{\boldsymbol{\Pi}} \det(\boldsymbol{U}'\boldsymbol{U}) = \frac{\boldsymbol{u}_1'\boldsymbol{u}_1}{\boldsymbol{u}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}_1}\det(\boldsymbol{U}'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{U}) = \frac{\boldsymbol{u}_1'\boldsymbol{u}_1}{\boldsymbol{u}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}_1}\det(\boldsymbol{Y}'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{Y}), \tag{80}$$

where the last equality uses the facts that $\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{X} = \boldsymbol{0}$ and $\det(\boldsymbol{\Gamma}) = 1$. We assume that $\boldsymbol{u}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}_1 \neq 0$, i.e., that the dependent variable contains some variation not explainable by the exogenous variables. The second factor on the right-hand side of (80) does not depend on the parameters. We write the first factor as

$$\kappa = \frac{\boldsymbol{u}_1'\boldsymbol{u}_1}{\boldsymbol{u}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{u}_1} = \frac{\boldsymbol{\tau}_1'\boldsymbol{Z}_1'\boldsymbol{Z}_1\boldsymbol{\tau}_1}{\boldsymbol{\tau}_1'\boldsymbol{Z}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{Z}_1\boldsymbol{\tau}_1}, \tag{81}$$

where we have defined $\boldsymbol{\tau}_1 = [1 \;\; -\boldsymbol{\tau}']'$ and $\boldsymbol{Z}_1 = [\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t} \;\; \boldsymbol{Z}\boldsymbol{T}]$. Since κ is the ratio of the sum of squared residuals to the sum of squares of the same residuals after partialing out $\boldsymbol{X}$, it holds that $\kappa \geq 1$.

It remains to minimize κ in (81) with respect to the structural coefficients $\boldsymbol{\tau}$, or, equivalently, with respect to $\boldsymbol{\tau}_1$. After some algebra, the first-order condition $\partial\kappa/\partial\boldsymbol{\tau}_1 = \boldsymbol{0}$ gives

$$(\boldsymbol{Z}_1'\boldsymbol{Z}_1 - \kappa\boldsymbol{Z}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{Z}_1)\boldsymbol{\tau}_1 = \boldsymbol{0}. \tag{82}$$

As a result, the minimization of (81) reduces to an eigenvalue problem and has the well-known solution,

$$\kappa = 1/\lambda_{\max}((\boldsymbol{Z}_1'\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{Z}_1), \tag{83}$$

where $\lambda_{\max}(\cdot)$ is the function that returns the maximum eigenvalue of its argument. Note that we do not compute the right-hand side of (83) as the minimum eigenvalue of $(\boldsymbol{Z}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1'\boldsymbol{Z}_1$ because the inverse involved cannot be assumed to exist in general.[23]

Recalling that the first element of $\boldsymbol{\tau}_1$ is 1, we remove the first row in (82) and find

$$\boldsymbol{T}'\boldsymbol{Z}'(\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t} - \boldsymbol{Z}\boldsymbol{T}\boldsymbol{\tau}) - \kappa\boldsymbol{T}'\boldsymbol{Z}'\boldsymbol{M}_{\boldsymbol{X}}(\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t} - \boldsymbol{Z}\boldsymbol{T}\boldsymbol{\tau}) = \boldsymbol{0}.$$

Solving for $\boldsymbol{\tau}$, we finally obtain the LIML estimator of $\boldsymbol{\tau}$,

$$\hat{\boldsymbol{\tau}} = (\boldsymbol{T}'\boldsymbol{Z}'(\mathbf{I} - \kappa\boldsymbol{M}_{\boldsymbol{X}})\boldsymbol{Z}\boldsymbol{T})^{-1}\boldsymbol{T}'\boldsymbol{Z}'(\mathbf{I} - \kappa\boldsymbol{M}_{\boldsymbol{X}})(\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t}).$$

By (71), the restricted LIML estimator of $\boldsymbol{\delta}$ is

$$\hat{\boldsymbol{\delta}}_{\text{RLIML}} = \boldsymbol{T}\hat{\boldsymbol{\tau}} + \boldsymbol{t} = \boldsymbol{T}(\boldsymbol{T}'\boldsymbol{Z}'(\mathbf{I} - \kappa\boldsymbol{M}_{\boldsymbol{X}})\boldsymbol{Z}\boldsymbol{T})^{-1}\boldsymbol{T}'\boldsymbol{Z}'(\mathbf{I} - \kappa\boldsymbol{M}_{\boldsymbol{X}})(\boldsymbol{y}_1 - \boldsymbol{Z}\boldsymbol{t}) + \boldsymbol{t}, \tag{84}$$

which can alternatively be written in terms of $\boldsymbol{R}$ and $\boldsymbol{r}$ instead of $\boldsymbol{T}$ and $\boldsymbol{t}$ by using (55) and (56).

[23] In fact, $(\boldsymbol{Z}_1'\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1'\boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{Z}_1 = \mathbf{I} - (\boldsymbol{Z}_1'\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1'\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{Z}_1$. Thus, one can instead find the smallest eigenvalue of $(\boldsymbol{Z}_1'\boldsymbol{Z}_1)^{-1}\boldsymbol{Z}_1'\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{Z}_1$, subtract it from 1, and take the reciprocal. To avoid constructing large intermediate matrices, $\boldsymbol{Z}_1'\boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{Z}_1$ should be computed as $\boldsymbol{Z}_1'\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Z}_1$, with cross-products taken first.

Finally, by setting $\boldsymbol{T} = \mathbf{I}$ and $\boldsymbol{t} = \mathbf{0}$ in (84), we obtain the unrestricted LIML estimator of $\boldsymbol{\delta}$ from the formula for the restricted one as

$$\hat{\boldsymbol{\delta}}_{\text{LIML}} = \left(\boldsymbol{Z}'(\mathbf{I} - \kappa \boldsymbol{M}_{\boldsymbol{X}})\boldsymbol{Z}\right)^{-1} \boldsymbol{Z}'(\mathbf{I} - \kappa \boldsymbol{M}_{\boldsymbol{X}})\boldsymbol{y}_1. \tag{85}$$

Note that the solution for κ computed in (83) is the usual one. It has the same mathematical relationship to the minimization problem as in a more standard derivation of LIML, and consequently (85) is the standard LIML estimator. The relevant development here is that we derived the LIML estimator analytically without referencing the classification of structural regressors by endogeneity. This formulation makes it straightforward to allow arbitrary linear restrictions on the structural coefficients.