Replication Code for “Inference on Linear Conditional Moment Inequalities” by Isaiah Andrews, Jonathan Roth, and Ariel Pakes

Summary

This repository contains Matlab code for replicating the simulations in the paper “Inference on Linear Conditional Moment Inequalities” by Andrews, Roth, and Pakes (henceforth ARP).

For empirical researchers interested in implementing the methods proposed in ARP, we recommend instead using the Matlab package available at **https://github.com/jonathandroth/LinearMomentInequalities/.**

System and Software Requirements

Running the code requires the Matlab software, as well as a CVXGEN license (see below for details). The code was last run in Matlab 2021a on a 2022 MacStudio computer. (The code should run on Windows or Linux machines as well.)

Running the code

The simulation code can be run after taking the following steps:

1. Unzip the replication code. In Matlab, set the working directory to the location of the folder Code/Simulate_Data/.

2. Open the file Code/Simulate_Data/set_working_dir.m and set the *working_dir* variable in line 3 to the location of the Code/Simulate_Data/ directory, e.g. “~/Dropbox/ARP-Replication/Code/Simulate_Data”.

3. Set-up CVXGEN (see below for details).

4. Set simulation parameters (optional).
    a. By default, the code uses the parameters for the main simulation specifications (500 simulations).
    b. If the parameter *onLaptop* is set to 1 on line 6 of the file Code/Simulate_Data/lp_set_parameters.m, then the code uses only 20 simulated datasets (and a different grid for test inversion) as used for the timing comparisons in Table 3 of ARP.

c. The number of simulations used can also be changed directly on line 10 or 13 (depending on whether onLaptop =1).

5. Run the file Code/Simulate_Data/run_all.m.
    a. Note that all of the scripts in Part 2 of the run_all.m script can be run in parallel. We highly recommend parallelization (e.g., running these scripts on a server) for users attempting to replicate the full 500 simulations (see Expected Runtime and Parallelization section below).

6. The Latex file Latex_for_Tables_and_Figures/ARP-Tables-and-Figures.tex can then be compiled to produce the tables in the paper from the output produced in the previous step.

CVXGEN Set-up

The simulation code uses the package provided by Kaido, Molinari, Stoye, and Thirkettle (2019, henceforth KMST) for calculating AS and KMS confidence intervals. The KMST code uses custom optimization code generated by CVXGEN, which requires a license to be used. We thus cannot publicly share the custom code generated by CVXGEN. However, CVXGEN licenses are free for academic researchers and can be requested at the CVXGEN website.

We are happy to share the CVXGEN code used in the ARP simulations with any researcher who has the requisite license.

Alternatively, researchers with a CVXGEN license can generate the code as follows. On the CVXGEN website, enter the following code in the “problem” field:

```
dimensions
  dim_p = 4
  J1 = 14
  J2 = 0
  S = 0
end


parameters
  A (J1 + 2*J2 + 2*dim_p + 2 + S, dim_p)
  b (J1 + 2*J2 + 2*dim_p + 2 + S, 1)
end


variables
  x ( dim_p , 1 )
```

```
end


minimize
  0
subject to
  A*x<=b
end
```

Click the “matlab” tab and follow the instructions to download a MEX solver for Matlab. Save the resulting file as “four_parameters_14_moments.mexmaci64” (the extension will be mexw64 on Windows or mexa64 on Linux). Place the file in Code/Simulate_Data/KMSCode/KMSPortable_V3/CVXGEN.

Repeat the instructions above to generate the files “two_parameters_6_moments”, “two_parameters_6_moments”, “four_parameters_38_moments”, where the parameter dim_p above is changed to match the number of parameters and the parameter J1 is changed to match to the number of moments.

Expected Runtime and Parallelization

Based on the runtimes in Table 2, we expect that running 1 simulation for all specifications would take about 30 core-hours. Thus, running all 500 simulations on a single-core computer would take over a year.

Fortunately, the runtime can be improved via parallelization in two ways. First, the code uses a parfor loop to loop over simulated datasets in parallel, and thus implements parallelization on computers with multiple cores. Second, all of the scripts run in the section labeled “Part 2” of the Code/Simulate_Data/run_all.m script can be run in parallel (e.g. on a computing server). Note that the code run on a server needs to have access to the results produced in Part 1.

We also note that researchers interested in reproducing a smaller number of simulations can do so by changing the number of simulations as discussed in step 4 above.