New Procedure: `chorussell`

Alexander Torgovitsky*

August 23, 2020

This is a simple procedure that works off of the bootstrapped distribution of the output of `estbounds`. The drawback is that its validity depends on some assumptions that may not be satisfied in some applications.

- Let $\hat{\theta}_{\text{lb}}$, $\hat{\theta}_{\text{ub}}$ denote the estimated lower and upper bounds from `estbounds` using the original data.
- Let $\{\hat{\theta}^b_{\text{lb}}\}_{b=1}^B$ and $\{\hat{\theta}^b_{\text{ub}}\}_{b=1}^B$ denote the estimated lower and upper bounds from `estbounds` using the bootstrapped samples $b = 1, \ldots, B$. (Note that $\hat{\theta}^b_{\text{lb}}, \hat{\theta}^b_{\text{ub}}$ use the same bootstrap sample b.)
- Let $\Delta \equiv \hat{\theta}_{\text{ub}} - \hat{\theta}_{\text{lb}}$ denote the length of the estimated identified set.
- To construct a level $1-\alpha$ confidence region, solve the following problem:

$$\begin{aligned}
&\min_{c_{\text{lb}}, c_{\text{ub}}} \ (c_{\text{lb}} + c_{\text{ub}}) \\
&\text{s.t.} \quad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left[\sqrt{n}\left(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}}\right) \leq c_{\text{lb}} \quad \text{and} \quad -c_{\text{ub}} \leq \sqrt{n}\left(\hat{\theta}^b_{\text{ub}} - \hat{\theta}_{\text{ub}} + \Delta\right)\right] \geq 1-\alpha \\
&\qquad \frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left[\sqrt{n}\left(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}} - \Delta\right) \leq c_{\text{lb}} \quad \text{and} \quad -c_{\text{ub}} \leq \sqrt{n}\left(\hat{\theta}^b_{\text{ub}} - \hat{\theta}_{\text{ub}}\right)\right] \geq 1-\alpha
\end{aligned}$$

  Call the solution $c_{\text{lb}}(\alpha), c_{\text{ub}}(\alpha)$. Then $[\hat{\theta}_{\text{lb}} - \frac{c_{\text{lb}}(\alpha)}{\sqrt{n}}, \hat{\theta}_{\text{ub}} + \frac{c_{\text{ub}}(\alpha)}{\sqrt{n}}]$ is a $1-\alpha$ confidence interval.
- This is not a convex problem (unless I am missing something), so not something that we can use Gurobi to solve. On the other hand, it should be possible to solve by "brute

*Department of Economics, University of Chicago.

force", since the indicator functions will be constant for many values of $c_{\text{lb}}, c_{\text{ub}}$. That is, the only possible solutions for c_{lb} should be in $\{\sqrt{n}(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}})\}_{b=1}^B \cup \{\sqrt{n}(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}} - \Delta))\}_{b=1}^B$ and similarly for $\hat{\theta}_{\text{ub}}$. So, in principle, we could try all possible $(2B)^2$ combinations for c_{lb} and c_{ub}, determine which ones satisfy the constraints, and then among those look for the smallest value of $c_{\text{lb}} + c_{\text{ub}}$, then call those the solution $c_{\text{lb}}(\alpha), c_{\text{ub}}(\alpha)$.

I would try this brute force approach first, since it is easiest to program.

- Here is a suggestion for a simple refinement of the brute force approach. Note that

$$\mathbb{1}\left[\sqrt{n}\left(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}}\right) \le c_{\text{lb}}\right] \ge \mathbb{1}\left[\sqrt{n}\left(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}}\right) \le c_{\text{lb}} \quad \text{and} \quad -c_{\text{ub}} \le \sqrt{n}\left(\hat{\theta}^b_{\text{ub}} - \hat{\theta}_{\text{ub}} + \Delta\right)\right].$$

  Thus, if

$$\frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left[\sqrt{n}\left(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}}\right) \le c_{\text{lb}}\right] < 1 - \alpha, \tag{1}$$

  then it cannot be the case that the first constraint in the program is satisfied. So, start by removing all values c_{lb} that satisfy (1). Analogously, for the upper bound we can get rid of those values c_{ub} that satisfy

$$\frac{1}{B}\sum_{b=1}^{B} \mathbb{1}\left[-c_{\text{ub}} \le \sqrt{n}\left(\hat{\theta}^b_{\text{ub}} - \hat{\theta}_{\text{ub}}\right)\right] < 1 - \alpha.$$

  Both of these procedures are quick, since they are just a matter of one-dimensional sorting. Once we have removed these c_{lb} and c_{ub} values that cannot individually be solutions of the optimization problem, then we do brute force on the combination of all possible values that remain. This should be a much smaller number of values.

  Check this against the full brute force approach to make sure that it obtains the same answer.

- Unlike the other inference procedures we have in `lpinfer`, this one directly builds confidence intervals, rather than testing a single point. This creates a bit of a conflict with our design philosophy, which has been to write functions that test a single point, then construct confidence intervals by inverting those tests using bisection. Here, we want to do the opposite, which means we use the duality between confidence intervals and testing in the opposite direction.

To keep the user interface stable, I would still have `chorussell` conduct a test and return a p–value. This can be done through bisection, with the duality relationship that a level α test rejects $t \in [\theta_{\text{lb}}, \theta_{\text{ub}}]$ if and only if t is not contained in a level $1 - \alpha$ confidence region. Thus, suppose that we have a bracket $\alpha_0 < \alpha_1$ such that there is rejection at α_1 but not at α_0. Then the p–value must lie between the two. So build a confidence region at level $(\alpha_1 + \alpha_0)/2$ and try the midpoint, then adjust the bracket accordingly depending on whether there is rejection here or not. Note that the major computational work of the `chorussell` procedure is going to be computing $\{\hat{\theta}^b_{\text{lb}}, \hat{\theta}^b_{\text{ub}}\}_{b=1}^B$, and this does not need to be repeated when changing the level of the confidence region.

In contrast, is the user wants a confidence interval with `chorussell`, then the above procedure can be done directly, instead of doing our current bisection routine.

If you want to be fancy, you could think of an abstraction "do a bisection" that can serve as a single function for both cases.