Simulation Exercise for Issue #104

September 24, 2020

Suppose we set `alpha = .1` and `R = 10`. Running the `chorussell` procedure gives:

```
library(lpinfer)
set.seed(5)
dgp <- mixedlogit_dgp()
df <- mixedlogit_draw(dgp, n = 4000)
alpha <- .1

lpm <- lpmodel(A.obs = mixedlogit_Aobs(dgp),
               beta.obs = function(d) mixedlogit_betaobs(d, dgp),
               A.shp = rep(1, nrow(dgp$vdist)),
               beta.shp = 1,
               A.tgt = mixedlogit_Atgt_dfelast(dgp, w2eval = 1, eeval = -1))

set.seed(5)
r <- chorussell(data = df, lpmodel = lpm, ci = TRUE, R = 10, alpha = alpha)
print(r)
```

```
## 90%-confidence interval: [0.63716, 0.79291]
```

I am going to store the confidence interval as `ci`.

```
ci <- r$ci.df[1, 3:4]
```

Next, I construct the confidence interval by following the "six steps" as I did in the other document for the ease of reference.

Step 1. Get the point estimates of the bounds

The point estimates of the bounds and the length of the estimated identified set are obtained as follows:

```
# Point estimate of bounds
lb <- r$lb
ub <- r$ub

# Length of the identified set
delta <- ub - lb
print(delta)
```

```
## [1] 0.0005065283
```

Step 2. Get the bootstrap bounds

The bootstrap bounds are obtained by finding the estimated bounds on the bootstrap data. I am going to extract them from the `r` object above and store them as follows:

```
lb.bs <- r$lb.bs
ub.bs <- r$ub.bs
```

Again, the same problems exist:

```
print(max(ub.bs))
```

```
## [1] 0.6915303
```

```
print(min(lb.bs))
```

```
## [1] 0.4426166
```

```
sum(lb.bs <= r$lb)
```

```
## [1] 9
```

Step 3. Get the list of candidates

Again, I denote the set of candidates $\{\sqrt{n}(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}})\}_{b=1}^B$ and $\{\sqrt{n}(\hat{\theta}^b_{\text{lb}} - \hat{\theta}_{\text{lb}} - \Delta)\}_{b=1}^B$ as `lb.can1` and `lb.can2` respectively. `lb.can` should contain all the possible candidates for c_{lb}.

```
n <- nrow(df)
lb.can1 <- sqrt(n) * (lb.bs - lb)
lb.can2 <- sqrt(n) * (lb.bs - lb - delta)
lb.can <- c(lb.can1, lb.can2)
```

Similarly, I denote the set of candidates $\{-\sqrt{n}(\hat{\theta}^b_{\text{ub}} - \hat{\theta}_{\text{ub}})\}_{b=1}^B$ and $\{-\sqrt{n}(\hat{\theta}^b_{\text{ub}} - \hat{\theta}_{\text{ub}} + \Delta)\}_{b=1}^B$ as `-ub.can1` and `-ub.can2` respectively. `ub.can` should contain all the possible candidates for c_{ub}.

```
ub.can1 <- sqrt(n) * (ub.bs - ub)
ub.can2 <- sqrt(n) * (ub.bs - ub + delta)
ub.can <- -c(ub.can1, ub.can2)
```

Step 4 and 5. Solve the minimization problem

To better understand how the solution is obtained, I solve the minimization problem by looking at all possible solutions in the two-dimensional grid instead of using the refinement method. The answer is obtained as:

```
df.grid <- data.frame(matrix(vector(), nrow = 0, ncol = 6))
colnames(df.grid) <- c("lb", "ub", "len", "cons1", "cons2", "feasible")
# Check the candidates through building a two-dimensional grid
for (x in lb.can) {
  for (y in ub.can) {
    cons1 <- mean((lb.can1 <= x) * (-y <= ub.can2))
    cons2 <- mean((lb.can2 <= x) * (-y <= ub.can1))
    df.grid[nrow(df.grid) + 1, ] <- c(x,
                                      y,
                                      x + y,
                                      cons1,
                                      cons2,
                                      ((cons1 >= 1 - alpha) & (cons2 >= 1 - alpha)))
  }
}

# Choose the bounds that minimize the objective function
c.bd.grid <- filter(df.grid, feasible == 1) %>% slice(which.min(len))
```

Step 6. Construct the confidence interval

The 90%-confidence interval can be obtained by:

```r
bd <- c(lb - c.bd.grid$lb/sqrt(n), ub + c.bd.grid$ub/sqrt(n))
print(bd)
```

```
## [1] 0.6371614 0.7929088
```

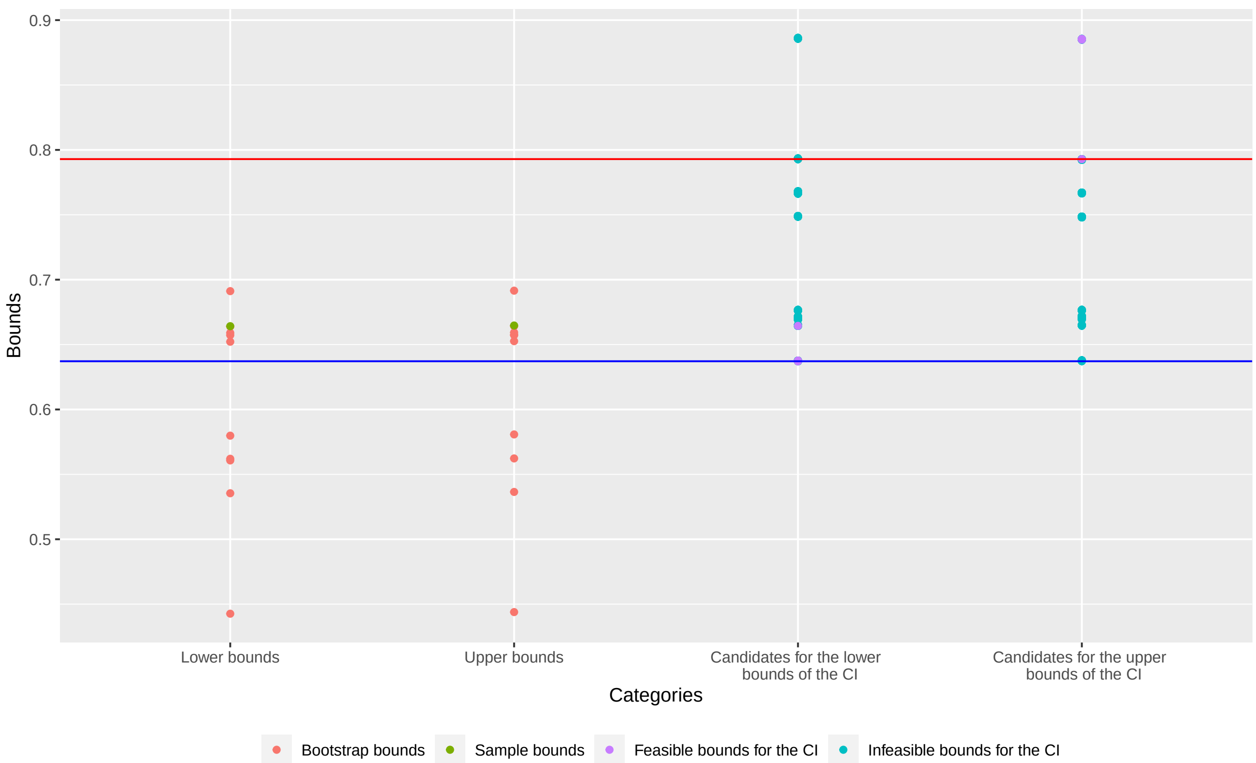
This matches with the output from the `chorussell` procedure of the `lpinfer` package (also confirming that the answers are the same in the brute force approach and in the refinement approach).

Some analysis

In this section, I visualize the bounds obtained from the sample and bootstrap data, as well as the upper and lower bounds of the confidence intervals obtained from the candidates `lb.can` and `ub.can`.

In the following plot, the blue line and the red line represent the lower and upper bound of the confidence interval obtained from the `chorussell` procedure. The points on the two left columns show the sample and bootstrap bounds obtained from the `estbounds` procedure, i.e. they represent `lb`, `ub`, `lb.bs` and `ub.bs`.

On the other hand, the third and fourth columns show the lower and upper bounds that are constructed from all possible values of c_{lb} and c_{ub} respectively. The purple dots refer to the ones that are feasible (i.e. they are the bounds constructed from the points $(c_{\text{lb}}, c_{\text{ub}})$ that satisfy the constraints of the minimization problem).



From the plot, the `chorussell` procedure is picking the smallest possible lower and upper bounds in this simulation exercise when minimizing $c_{\text{lb}} + c_{\text{ub}}$. This can also be seen in the following code:

```r
df.grid$ci.lb <- lb - df.grid$lb/sqrt(n)
df.grid$ci.ub <- ub + df.grid$ub/sqrt(n)
print(ci$lb == min(filter(df.grid, feasible == 1)$ci.lb))
```

```
## [1] TRUE
```

```r
print(ci$ub == min(filter(df.grid, feasible == 1)$ci.ub))
```

```
## [1] TRUE
```