ECO539B - Problem Set 1

Filippo Palomba

15/03/2022

I benefited from discussion with Thomas Bearpark, Pier Paolo Creanza, Nicolas Hommel, Casey McQuillan, and Eric Qian. Errors are my own.

Question 1

a) First of all, the asymptotic distribution of the root $\widehat{R}_i - R_i$ is not going to be asymptotically normal because the support of R_i and $\widehat{R}_i$ is discrete and bounded between -19 and 19. Second, and most importantly, for some R_i the asymptotic distribution is not going to be even symmetric. This happens because $\text{supp}R_i = \{0.5, \dots, 9.5\}$, therefore swapping the quantiles of J_∞ might be problematic for the percentile bootstrap especially when R_i is close to the boundaries of the support. In other words, the finite sample distribution of the root $J_n(\cdot, F)$ is continuous in F.

If $\text{var}_n(\widehat{\theta}_n)$ converges to 0, then all θ_is will be ranked correctly almost surely, but still the distribution will be asymmetric at the boundaries inducing problems when flipping quantiles. Similarly, if $k_n \to \infty$ and $n \to \infty$ the problem will persist at the boundaries because of the asymmetric distribution. However, note that as k grows larger, the CDF of each R_i has smaller steps and becomes smoother. This will make the asymptotic distribution of $\widehat{R}_i - R_i$ more symmetric for interior points of R_i and the percentile bootstrap will perform better (for interior points) as k increases.

b) The algorithm to construct bootstrap confidence intervals is the following

1. Set the true $\{\theta_i\}_{i=1}^k$ according to Design 1 ($\theta_i = i$), Design 2 ($\theta_i = 10i$), or Design 3 ($\theta_i = i/10$) and compute accordingly
$$R_i(\theta_1, \dots, \theta_k) = \frac{1}{2} + \sum_{j=1}^{k} \mathbb{1}\{\theta_j < \theta_i\} + \frac{1}{2}\sum_{j=1}^{k} \mathbb{1}\{\theta_j = \theta_i\}.$$
2. Draw the fixed effect coefficients $\hat{\theta}_i \sim \texttt{N}(\theta_i, 1), i = 1, \dots, k$ and compute
$$\widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k) = \frac{1}{2} + \sum_{j \neq i} \mathbb{1}\left\{\hat{\theta}_j < \hat{\theta}_i\right\} + \frac{1}{2}\sum_{j \neq i} \mathbb{1}\left\{\hat{\theta}_j = \hat{\theta}_i\right\}$$
3. Draw N bootstrap draws $\{\tilde{\theta}_i^{(b)}\}_{b=1}^N$, where $\tilde{\theta}_i^{(b)} \sim \texttt{N}(\hat{\theta}_i, 1), i = 1, \dots, k, b = 1, \dots, N$
4. Compute the rank statistic for each bootstrap iteration
$$\widehat{\widehat{R}}_i^{(b)}(\tilde{\theta}_1^{(b)}, \dots \tilde{\theta}_k^{(b)}) = \frac{1}{2} + \sum_{j \neq i} \mathbb{1}\left\{\tilde{\theta}_j^{(b)} < \tilde{\theta}_i^{(b)}\right\} + \frac{1}{2}\sum_{j \neq i} \mathbb{1}\left\{\tilde{\theta}_j^{(b)} = \tilde{\theta}_i^{(b)}\right\}.$$

and construct the root statistic as

$$T^{(b)} = \widehat{\widehat{R}}_i^{(b)}(\tilde{\theta}_1^{(b)}, \dots \tilde{\theta}_k^{(b)}) - \widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k).$$

5. Compute the $\alpha/2$ and $1-\alpha/2$ quantiles of $\{T^{(b)}\}_{b=1}^N$, denoted $\mathfrak{c}_T(\alpha_2/2)$ and $\mathfrak{c}_T(1-\alpha_2/2)$, respectively
6. Construct confidence intervals for $\widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k), i = 1, \dots, k$

$$\widehat{\text{CI}}_{i,\texttt{pct}} = [\widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k) - \mathfrak{c}_T(1-\alpha_2/2); \widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k) - \mathfrak{c}_T(\alpha_2/2)],$$
$$\widehat{\text{CI}}_{i,\texttt{efr}} = [\widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k) + \mathfrak{c}_T(\alpha_2/2); \widehat{R}_i(\hat{\theta}_1, \dots \hat{\theta}_k) + \mathfrak{c}_T(1-\alpha_2/2)].$$

7. Repeat steps 1-7 M times and collect $\{R_i(\theta_1, \dots \theta_k), \widehat{\text{CI}}_{i,\texttt{pct}}^{(m)}, \widehat{\text{CI}}_{i,\texttt{efr}}^{(m)}\}_{m=1}^N$
8. Compute the asymptotic coverage of the two confidence intervals as

$$\texttt{CV}_i^{\texttt{pct}} = 100 \cdot \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}(R_i(\theta_1, \dots \theta_k) \in \widehat{\text{CI}}_{i,\texttt{pct}}^{(m)}), \quad \texttt{CV}_i^{\texttt{efr}} = 100 \cdot \frac{1}{M} \sum_{m=1}^{M} \mathbb{1}(R_i(\theta_1, \dots \theta_k) \in \widehat{\text{CI}}_{i,\texttt{efr}}^{(m)})$$

c) Table 1 shows the results of the MonteCarlo simulation to get the asymptotic coverage of the 95% confidence intervals obtained with the percentile bootstrap and with the Efron's percentile bootstrap.

```r
set.seed(8894)
k <-10
M <-1000
N <- 1000
results <- simulRun(k = k, M = M, N = N)
```

Table 1: Coverage probability of bootstrap confidence intervals.

	Design 1 - $\theta_i = i$		Design 2 - $\theta_i = 10i$		Design 3 - $\theta_i = i/10$	
	Percentile	Efron	Percentile	Efron	Percentile	Efron
R_1	95.60	99.40	100.00	100.00	67.60	73.90
R_2	74.20	99.40	100.00	100.00	57.70	88.40
R_3	89.40	99.40	100.00	100.00	54.50	93.40
R_4	90.00	99.70	100.00	100.00	54.20	96.30
R_5	91.60	99.60	100.00	100.00	51.30	97.10
R_6	94.20	99.50	100.00	100.00	50.10	97.40
R_7	89.20	99.30	100.00	100.00	51.10	96.10
R_8	93.10	99.30	100.00	100.00	54.50	91.80
R_9	72.60	98.80	100.00	100.00	56.70	88.40
R_{10}	96.00	99.10	100.00	100.00	68.70	76.40

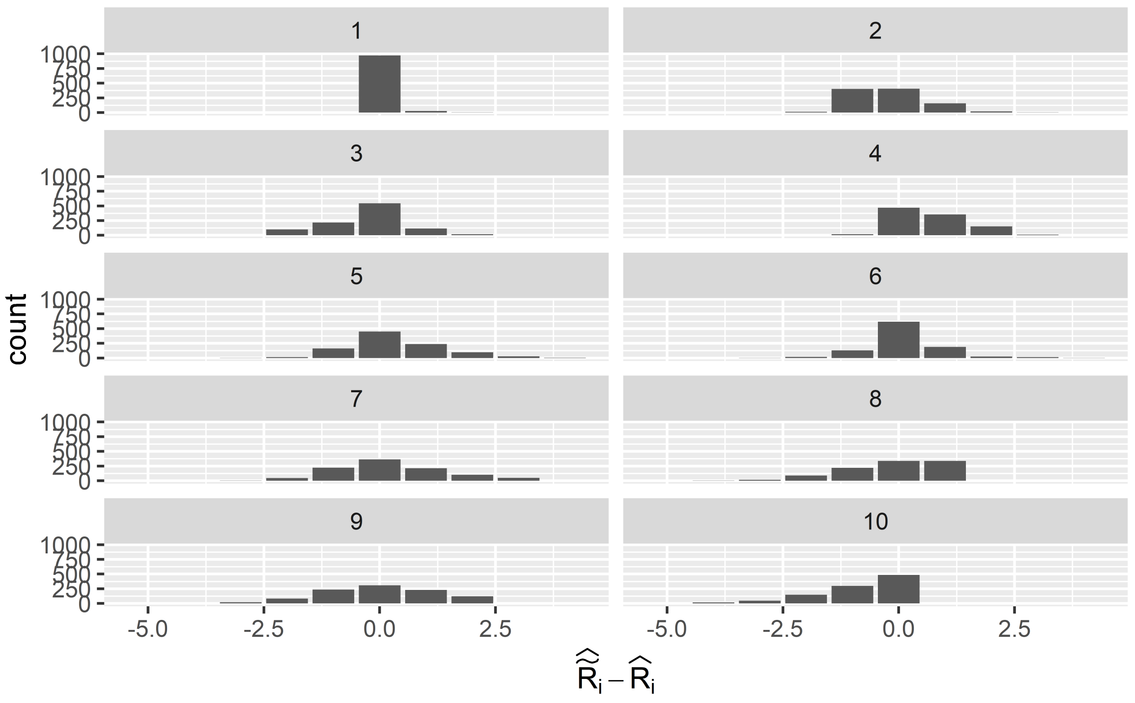
Notes: This table shows the estimates of the coverage probabilites $\texttt{CV}_i^{\texttt{pct}}$ and $\texttt{CV}_i^{\texttt{efr}}$. The number of simulations is $M = 5000$ and $N = 1000$ bootstrap iterations were used in each simulation.

There are several things to notice in the results reported in Table 1. First, let's note the following two effects that underlie the design:

1. **Boundary Effect.** The closer the population R_i is to the boundary of suppR_i, the easier it is to cover the true R_i. Say that $R_i = 0.5$ in either Design 1 or Design 2, then there are no other values of R_i to the left that we have to disentangle from the true one, whereas if $R_i = 4.5$ we have "competitors" from the left and from the right, making it harder to correctly cover the true value.

2. **Asymmetry of bootstrap distribution**. This is particularly true, the closer the population value of R_i is to the boundary of suppR_i. Figure 1 shows this occurrence for a randomly selected bootstrap distribution among the M ones in the simulation in Design 1. Figure 2 shows that the the bootstrap distribution is not truncated at 0 anymore when R_i is at the boundary in Design 3. This happens because in that design $\widehat{R}_i$ behaves worse because the θ_is are closer to each other, making $\widehat{R}_i$ possibly very different from R_i.

Figure 1: Asymmetry of Bootstrap Distribution (Design 1).



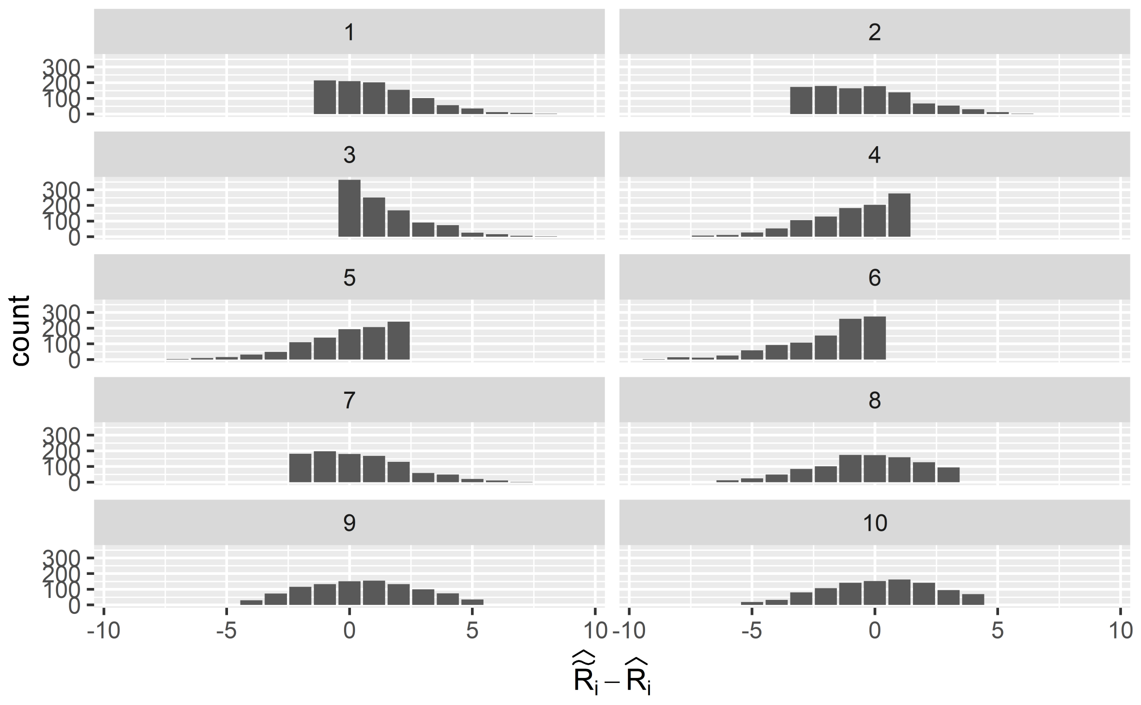
By using these two effects, we can analyze the results in Table 1:

Design 1 In this design the Efron's bootstrap works extremely well but shows over-coverage. The percentile bootstrap suffers close to the boundary but not at the boundary. To explain the non-monotonic coverage of the percentile bootstrap, first note that, as the population value of R_i gets closer to the boundary, the bootstrap distribution is becoming more and more skewed, and swapping the quantiles yields a poor approximation of the tails of the distribution. On the flip side, when R_i is exactly on the boundary, the boundary effect dominates the negative impact of the asymmetry of the bootstrap distribution and coverage increases again. Figures 3 and 4 show the confidence intervals obtained with the percentile and Efron's percentile bootstrap, respectively, for some simulations and report also the true values of $R_i, i = 1, \dots, k$.

Design 2 Both bootstrap procedures work extremely well with Design 2, showing over-coverage. The intuition for this result is that since the true θ_is are far apart from each other $\mathbb{P}(\widehat{R}_i \geq \widehat{R}_j) = 0$ a.s. for $i < j$. In words, there is no overlap between the distributions from which the θ_is are generated, hence $R_i = \widehat{R}_i$ almost surely, thus, by construction $R_i \in \widehat{\text{CI}}_{i,\texttt{efr}}$ and $R_i \in \widehat{\text{CI}}_{i,\texttt{pct}}$ $\forall\, i$ almost surely. The benign second effect always dominates in this design. We do not report any figure for this design since the confidence intervals are always the singleton $\{R_i\}$.

Design 3 This is the most complicated case to analyze and the one in which both techniques perform worse than in the other two cases. Let's start with the Efron's percentile bootstrap. The benign boundary effect

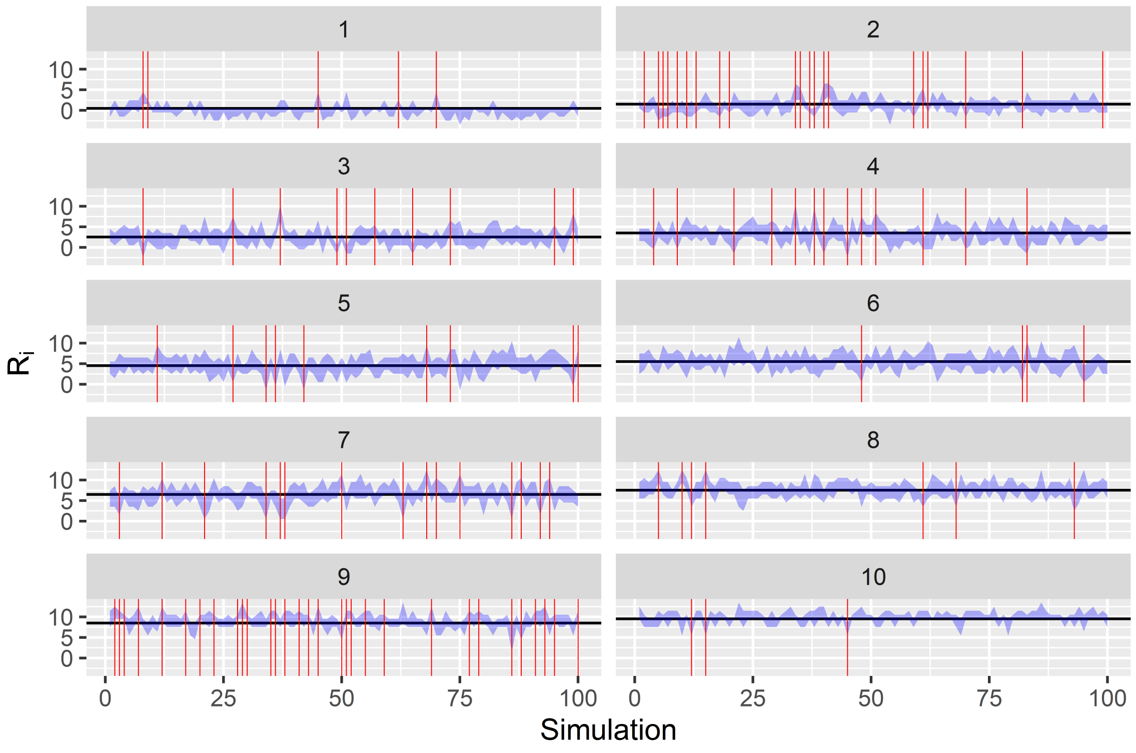
Figure 2: Asymmetry of Bootstrap Distribution (Design 3).



is dominated by the fact that the θ_is are very close to each other and when R_i is at the left boundary the bootstrap distribution has a long tail to the left of 0 (viceversa at the right boundary). This wastes the major part of the confidence interval and explains why coverage probability drops sharply at the two boundaries. This technique behaves nicely for all the other cases. If we instead analyze the percentile bootstrap, the fact that the bootstrap distribution puts "redundant mass" outside the support of R_i turns out to be beneficial at the boundaries in terms of coverage probability because quantiles are swapped. However, the strong asymmetry of the bootstrap distribution for all true values of R_i induced by the proximity of the θ_is dominates and hence the very low coverage probabilities.

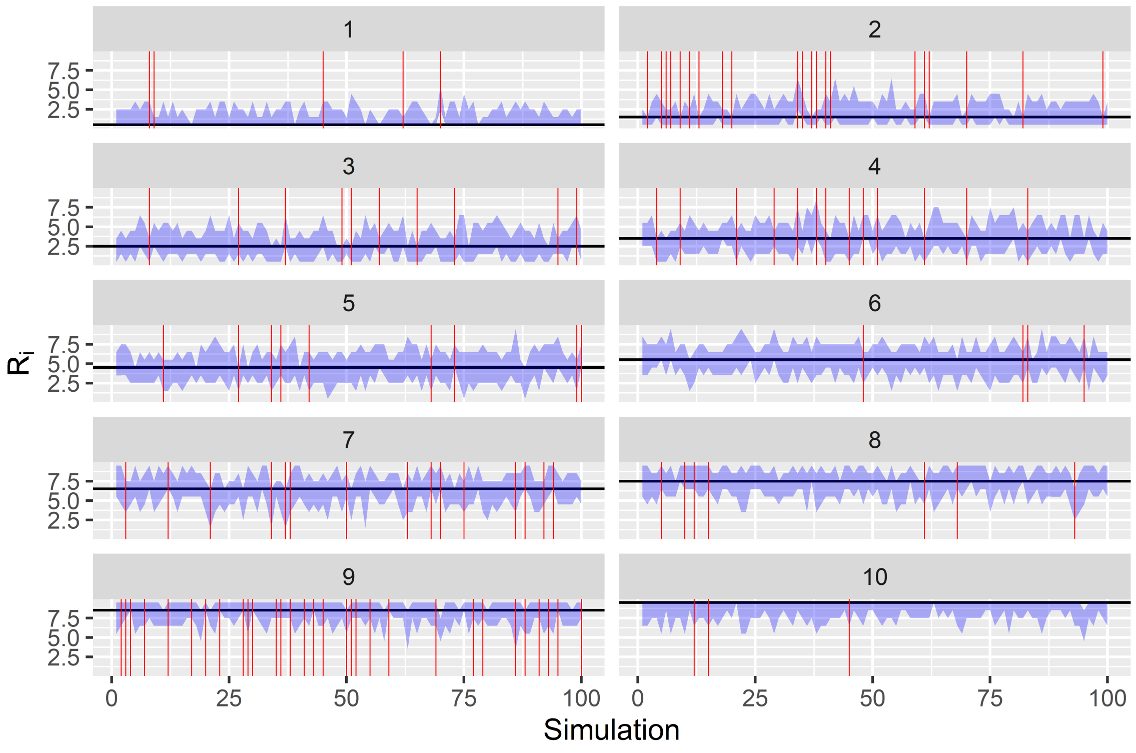
d) As we can see from Figure 3 and Figure 5, percentile bootstrap confidence intervals tend to have either the lower bound or the upper bound outside the support of R_i. This is an undesiderable feature because it increases the expected average length of the confidence interval without having a positive impact on the coverage probability.

Figure 3: Percentile bootstrap confidence intervals in Design 1.



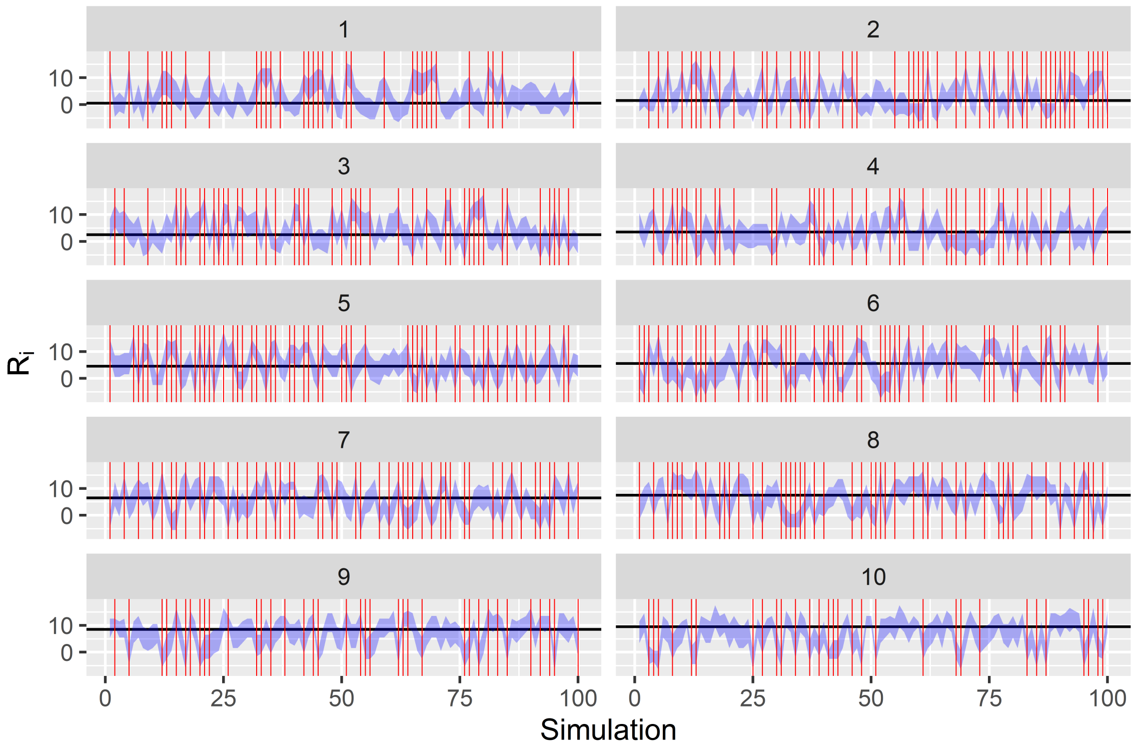
Notes: This figure reports the estimated 95% confidence intervals (blue shaded areas) for 100 MonteCarlo simulations. The true value of R_i is indicated by the horizontal black solid line. Vertical solid red lines highlight simulations where the 95% confidence interval does not cover the true value of R_i.

Figure 4: Efron's Percentile bootstrap confidence intervals in Design 1.



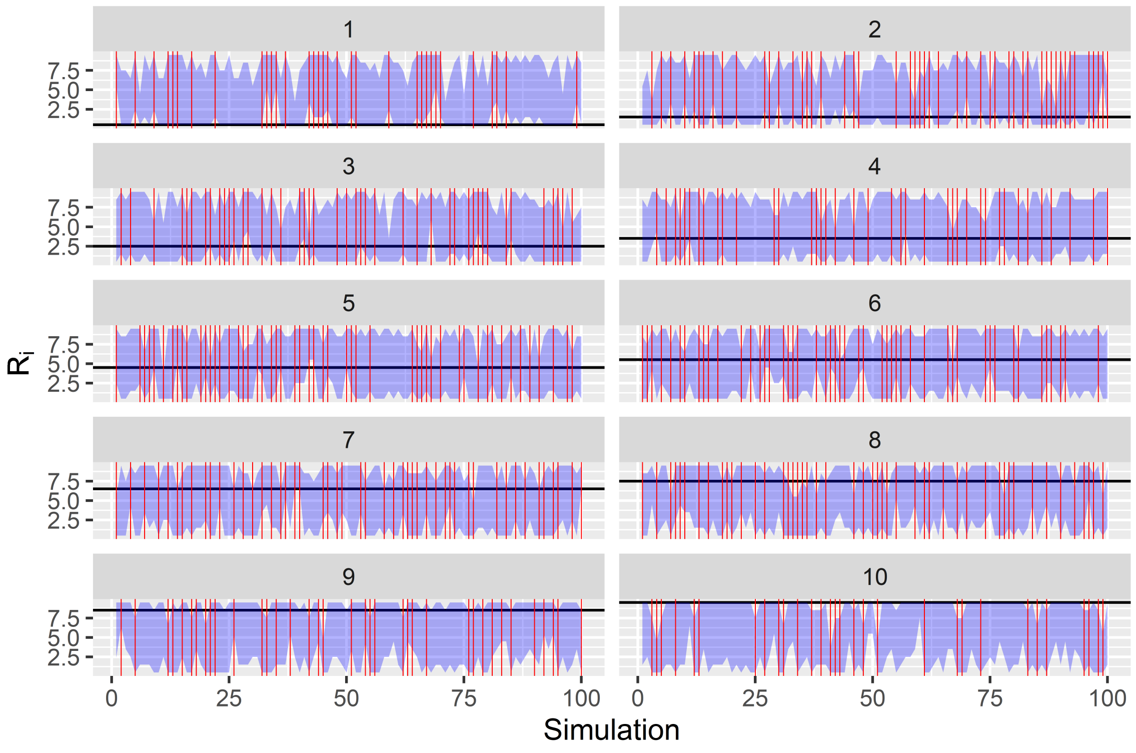
Notes: This figure reports the estimated 95% confidence intervals (blue shaded areas) for 100 MonteCarlo simulations. The true value of R_i is indicated by the horizontal black solid line. Vertical solid red lines highlight simulations where the 95% confidence interval does not cover the true value of R_i.

Figure 5: Percentile bootstrap confidence intervals in Design 3.



Notes: This figure reports the estimated 95% confidence intervals (blue shaded areas) for 100 MonteCarlo simulations. The true value of R_i is indicated by the horizontal black solid line. Vertical solid red lines highlight simulations where the 95% confidence interval does not cover the true value of R_i.

Figure 6: Efron'sPercentile bootstrap confidence intervals in Design 3.



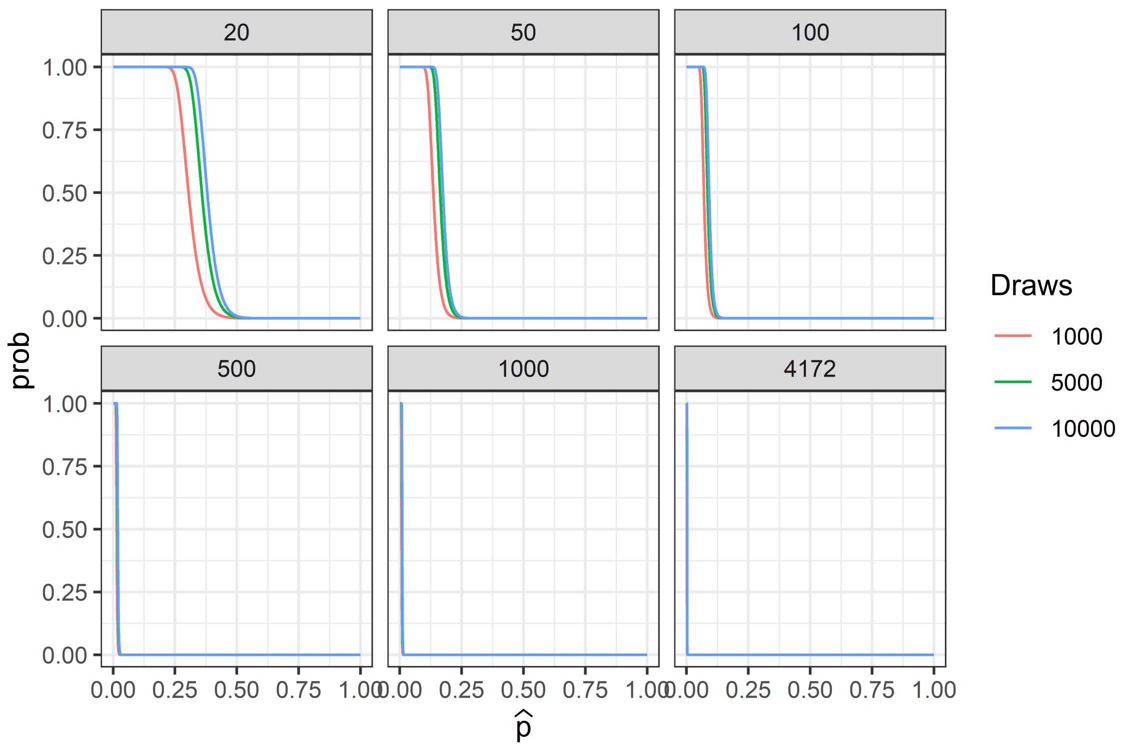
Notes: This figure reports the estimated 95% confidence intervals (blue shaded areas) for 100 MonteCarlo simulations. The true value of R_i is indicated by the horizontal black solid line. Vertical solid red lines highlight simulations where the 95% confidence interval does not cover the true value of R_i.

Question 2

a) Yes, the non-parametric bootstrap should yield asymptotically correct standard errors if both $\widehat{\beta}$ and $\overline{Y}$ are recomputed in each draw of the bootstrap to take into account estimation uncertainty in both steps. However, we have to **rule out** some DGPs. This problem is closely related to the weak IV one, where a first stage parameter in the neighborhood of 0 breaks the asymptotic normality of the IV estimator. Indeed, if the population value of $p := \mathbb{E}[Y_i]$ is in a neighborhood of 0, this model becomes weakly identified, hence the bootstrap procedure fails to yield a consistent estimate for the standard errors of $\widehat{\theta}$. The main difference of this setting with the weak IV one is that here the sign restriction holds by definition of p, thus the moments of $\beta/\mathbb{E}[Y_i]$ are well defined.

Moreover, note that Y_i is dichotomous, thus if the fraction of people paying the property tax p is not sufficiently high, it would be likely that in at least one iteration, the bootstrappped $\mathbf{Y}^\star = (Y_1^\star, \dots, Y_N^\star)'$ would be a vector of zeros. Specifically, for each iteration there is a probability $(1-\widehat{p})^N$ that this happens, where $\widehat{p} = N^{-1}\sum_{i=1}^N Y_i$. Therefore, $1-[1-(1-\widehat{p})^N]^B$ is the probability of drawing at least one such vector in the whole bootstrap procedure, where B are the bootstrap draws. If such $\mathbf{Y}^\star$ is drawn, then standard errors are going to be infinite. Figure 7 shows the probability of this occurrence as a function of $\widehat{p}, B$, and N. The south-east panel sets $N = 4172$, the lowest sample size in Bergeron, Tourek, and Weigel (2020).

Figure 7: Finite sample probability of drawing a vector of zeros in the entire bootstrap.



An interesting alternative here is to use a Bayesian bootstrap, as it won't suffer from the problem of resampling. Indeed, the Bayesian bootstrap does not rely on sampling with replacement as the frequentist bootstrap. Rather, it requires a multinomial likelihood and a Dirichlet prior on the relative frequency of each observed value in the support of Y. As such, as long as the posterior distribution of the frequency weights is not degenerate on a single point, the Bayesian bootstrap does not fail. The posterior of the frequency weights will be Dirichlet with parameters $n_k + \alpha_k, k = 0, 1$, where n_k is the number of times $Y = k$ has been observed in the sample and α_k is the corresponding prior parameter.

b) An alternative to the non-parametric bootstrap would be to use a just-identified GMM estimator with moment conditions

$$\mathbb{E}[g(W_i; \alpha, \beta, \boldsymbol{\gamma}, p)] = \mathbb{E}\begin{bmatrix} Y_i - p \\ \boldsymbol{Z}_i \epsilon_i \end{bmatrix} = 0,$$

where $\boldsymbol{Z}_i = (1, X_i, \mathbf{H}_i)$, where $\mathbf{H}_i$ is a $h \times 1$ vector of indicators, where h is the number of neighborhoods. In such case, the estimator is asymptotically normal with asymptotic variance given by

$$\mathtt{V} = \mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\psi}'} g(W_i; \boldsymbol{\psi})\right]^{-1} \mathbb{E}\left[g(W_i; \boldsymbol{\psi}) g(W_i; \boldsymbol{\psi})'\right] \mathbb{E}\left[\frac{\partial}{\partial \boldsymbol{\psi}'} g(W_i; \boldsymbol{\psi})\right]^{-1'},$$

and then use the Delta-Method to work out the variance of $\theta = h(\boldsymbol{\psi}) = \beta / p$. However, even this procedure would suffer the same problem. Indeed, Theorem 3.1. in Newey and McFadden (1994) requires the population value of the parameter to lie in the interior of the parameter space.

Main Functions

Rank Estimator

```r
rankEst <- function(theta) {
  R <- matrix(NA, nrow = length(theta), 1)
  for (i in seq_len(length(theta))) {
    R[i, 1] <-  1/2 + sum(theta < theta[i])  +  (sum(theta == theta[i]) - 1)/2
  }
  return(R)
}
```

Generate fixed effect coefficients

```r
dataGen <- function(k, M = 5000) {

  ## First design
  theta1 <- c(1:k)
  rank1 <- c(1:k) - 1/2
  data.design1 <- MASS::mvrnorm(n = M, mu = theta1, Sigma = diag(rep(1, k)))

  ## Second design
  theta2 <- 10*theta1
  rank2 <- c(1:k) - 1/2
  data.design2 <- MASS::mvrnorm(n = M, mu = theta2, Sigma = diag(rep(1, k)))

  ## Third design
  theta3 <- 1/c(1:k)
  rank3 <- c(k:1) - 1/2
  data.design3 <- MASS::mvrnorm(n = M, mu = theta3, Sigma = diag(rep(1, k)))

  # Transpose output so each columns is a simulation
  return(list(data.design1 = t(data.design1),
              data.design2 = t(data.design2),
              data.design3 = t(data.design3),
              theta.design1 = theta1,
              theta.design2 = theta2,
              theta.design3 = theta3,
              rank.design1 = rank1,
              rank.design2 = rank2,
              rank.design3 = rank3))

}
```

Run Parametric Bootstrap

```
bootEst <- function(theta.est, N = 1000, level = .95) {

  # estimate rank given estimated thetas
  rank.est <- rankEst(theta.est)

  # For each theta.est draw N bootstrap samples from parametric distribution
  # rows are bootstrap draws
  theta.boot <- MASS::mvrnorm(n = N, mu = theta.est,
                             Sigma = diag(rep(1, length(theta.est))))

  # compute ranks (rows are bootstrap draws)
  rank.boot <- t(apply(theta.boot, 1, rankEst))
  rank.est.mat <- matrix(rep(rank.est, N), length(theta.est), N)

  J.boot <- rank.boot - t(rank.est.mat)

  # get quantiles for each parameter (within columns)
  alpha <- 1 - level
  J.hat <- t(apply(J.boot, 2, quantile, probs = c(alpha/2, 1-alpha/2)))

  # get lb and ub using percentile bootstrap
  percentile.lb <- rank.est - J.hat[,2]
  percentile.ub <- rank.est - J.hat[,1]

  # get lb and ub using efron's percentile bootstrap
  efron.lb <- rank.est + J.hat[,1]
  efron.ub <- rank.est + J.hat[,2]

  percentile.CI <- cbind(percentile.lb, percentile.ub)
  efron.CI <- cbind(efron.lb, efron.ub)

  return(list(percentile.CI = percentile.CI,
              efron.CI = efron.CI,
              rank.est = rank.est))
}
```

Get Coverage Probabilities

```
coverageGet <- function(CI.list, rank.true) {

  percentile.coverage <- CI.list$percentile.CI[,1] <= rank.true &
```

```r
                         CI.list$percentile.CI[,2] >= rank.true

  efron.coverage <- CI.list$efron.CI[,1] <= rank.true &
                    CI.list$efron.CI[,2] >= rank.true

  return(list(percentile.coverage=percentile.coverage,
              efron.coverage=efron.coverage,
              rank.est = CI.list$rank.est))
}
```

Transform List in Matrix

```r
matrixGet <- function(list.covers){
  k <- length(list.covers[[1]]$percentile.coverage)
  M <- length(list.covers)

  percentile.covers <- matrix(NA, nrow = k, ncol = M)
  efron.covers <- matrix(NA, nrow = k, ncol = M)
  rank.est <- matrix(NA, nrow = k, ncol = M)

  for (m in seq_len(M)) {
    percentile.covers[, m] <- list.covers[[m]]$percentile.coverage
    efron.covers[, m] <- list.covers[[m]]$efron.coverage
    rank.est[, m] <- list.covers[[m]]$rank.est
  }

  return(list(percentile.cvg.mat = percentile.covers,
              efron.cvg.mat = efron.covers,
              rank.est = rank.est))
}
```

Run MonteCarlo

```r
simulRun <- function(k = 10, M = 5000, N = 1000, level = .95) {

  # simulate estimated thetas M times
  designs <- dataGen(k = k, M = M)

  # estimate confidence intervals in each design
  design1.CI <- apply(designs$data.design1, 2, bootEst)
  design2.CI <- apply(designs$data.design2, 2, bootEst)
  design3.CI <- apply(designs$data.design3, 2, bootEst)

  # get coverage for each confidence interval
  design1.covers <- lapply(design1.CI, coverageGet, designs$rank.design1)
  design2.covers <- lapply(design2.CI, coverageGet, designs$rank.design2)
  design3.covers <- lapply(design3.CI, coverageGet, designs$rank.design3)

  # extract information into (k X M) matrices
  design1.cvg.mat <- matrixGet(design1.covers)
  design2.cvg.mat <- matrixGet(design2.covers)
  design3.cvg.mat <- matrixGet(design3.covers)

  # compute coverage for each parameter
  coverages <- cbind(rowMeans(design1.cvg.mat$percentile.cvg.mat),
                     rowMeans(design1.cvg.mat$efron.cvg.mat),
                     rowMeans(design2.cvg.mat$percentile.cvg.mat),
                     rowMeans(design2.cvg.mat$efron.cvg.mat),
                     rowMeans(design3.cvg.mat$percentile.cvg.mat),
                     rowMeans(design3.cvg.mat$efron.cvg.mat))*100

  colnames(coverages) <- c('Percentile', 'Efron',
                           'Percentile', 'Efron',
                           'Percentile', 'Efron')
  rownames <- c()
  for (i in seq_len(k)) {
    rownames <- c(rownames, paste0('R_', i))
  }
  rownames(coverages) <- rownames
  return(list(coverages = coverages,
              design1.rank.est = design1.cvg.mat$rank.est,
              design2.rank.est = design2.cvg.mat$rank.est,
              design3.rank.est = design3.cvg.mat$rank.est,
              design1.rank.true = designs$rank.design1,
              design2.rank.true = designs$rank.design2,
              design3.rank.true = designs$rank.design3,
              design1.CI = design1.CI,
              design2.CI = design2.CI,
              design3.CI = design3.CI))
}
```