ECON 390: Assignment 2

Alex Marsh

Due September 16, 2021

Problem Set Preparation

At the very beginning of your script, set your rng seed to 123. This can be done by placing "set.seed(123)" at the very beginning of your code. This is very important to get the right answers.

Logic

We're going to "test" DeMorgan's laws empirically. Draw 100 simulations from $N(2, 1)$ and call these "norm_draws". Likewise, draw 100 simulations from Exp(2) called "exp_draws". For the latter, use the function "rexp()" with a lambda of 2. Remember to look at the help documentation if you don't know how this works. Let the statements be defined as follows:

- P is the statement norm_draws > 2.
- Q is the statement exp_draws $> \frac{1}{2}$.

Show that each version of DeMorgan's Laws hold with these statements. Take the mean across all the logical statements. If you did this correctly, the mean should evaluate to 1 in both cases.

Note: Be careful with negating $>$. While if you do it incorrectly, you might still get the right answer for a few reasons, I will be looking to make sure you did it correctly.

If Statements and Functions

1. Create your own absolute value function that takes in a single value and returns it's absolute value. Call it "my_abs". Note: ***Do not use the abs() function.***
2. Now, create your own absolute value function that takes in *a vector* and returns it's absolute value. Call it "my_abs_vec". Note: ***Do not use the abs() function.***
3. Create a function that returns "the sign" of a single number. That is, if the number is positive, return 1, if the number is negative, return -1, and if the number if 0, return 0. Call this function "my_sign".

4. Create a function that returns the sign of a vector of numbers. Call this function "my_sign_vec".

5. Write a function called CRRA that takes in two inputs c and η (spelled eta) and returns

$$u(c\,;\eta) = \begin{cases} \frac{c^{(1-\eta)}-1}{1-\eta}, & \text{if } \eta \geq 0 \text{ and } \eta \neq 1 \\ \log c, & \eta = 1. \end{cases} \tag{1}$$

   Note, if $\eta < 0$, the function should error out. Likewise, you should write this function to be able to take in *multiple* values of c but only one value of η.

6. Create a function called "my_funct" that calculates the following:

$$f(x) = \begin{cases} x^2 + 2x + |x|, & \text{if } x < 0 \\ x^2 + 3 + \log(x+1), & \text{if } 0 \leq x < 2 \\ x^2 + 4x - 14, & \text{if } x \geq 2. \end{cases} \tag{2}$$

   Note: This function should be able to take multiple values of x.

7. Create the following object and calculate the mean, median, min, max and standard deviation of each row and call. Name each output "x_y" where x is the statistic that is being calculated and y is either row or column.

   ```
   my_mat = matrix(c(rnorm(20,0,10), rnorm(20,-1,10)), nrow = 20, ncol = 2)
   ```

An Economic Application of Loops: Auction Simulation

Auction theory is a popular and important area in industrial organization. Designing auctions to maximize revenue or maximizing social welfare are two common goals of auction theory. While auction theory is nice (in theory), empirical applications are important when applying auction theory for actual auction design. The following is inspired by Haile and Tamer (2003).

Consider the following (incomplete[1]) model of bidding. Let v_i be the *valuation* of bidder i and let b_i be bidder i's bid. We only make two behavioral assumptions:

1. No one bids more than how much they value the item: for all bidders i, $b_i \leq v_i$,
2. No one lets someone else win, if they're willing to bid enough to win: for losing bidders j, $v_j \leq b_n + \Delta$, where b_n is the *winning* bid and Δ is the minimum bid increment.[2]

Item 1. says no one will bid more than they value the object. This is fairly intuitive. Item 2. says that if someone *does not* win, then it must be that their valuation is smaller than the winning bid plus the minimum bid increment. This just means that no one will let someone else win if they're willing to bid more.

While it is possible to use this model to recover the valuations from actual bids, [3] we will not be doing that.[4] Instead, we are going to simulate a bunch of auctions and see what the resulting bids would be (rather than backing out the valuations from the bids).

[1] Incomplete in the sense that we are not defining what agents are maximizing.

[2] Practically speaking, Δ is at least \$0.01 since the smallest someone can bid is one penny. However, an auction might have an official minimum bid increment.

[3] This is what we would do with actual bid data.

[4] See Haile and Tamer (2003).

1. To start, suppose there are $N_{\text{sim}} = 1000$ auctions, each auction with only two bidders in them. We need to draw valuations for bidders from a distribution. To do this, assume each bidders' valuation comes from a log-Normal distribution with a mean μ and a variance σ^2. Suppose $\mu = 1$ and $\sigma^2 = 1$. Draw $2 * N_{\text{sim}}$ valuations for all bidders in all auctions.

   Hint: In base R, there is not a way to draw from a log-Normal distribution. However, if we say a random variable X is distributed log-Normal, that means that $\log X$ is distributed Normal. So if we draw variables from a Normal distribution and then exponentiate them (the inverse of log), then these random variables will be distributed log-Normal.

2. We will now loop over auctions. For each auction, we need to decide who bids first between bidder 1 and 2. To do this, "flip a coin" in R: if 0, bidder 1 goes first; if 1, bidder 2 goes first. If the first bidder's valuation is more than the minimum bid increment Δ, start the first bid at Δ. If Δ is greater than the first bidder's valuation, the first bidder bids her valuation. Set $\Delta = 0.01$ (one penny).

3. After the first bid is placed, alternate between bidders. If the current bid *plus* the minimum bid increment Δ is less than the current bidder's valuation, they up the current bid to be the last bid plus the minimum bid increment.

4. Continue this process until someone is not willing to bid more. The last bid is the winning bid. Save this bid and the winning bidder.

5. Loop over all N_{sim} auctions.

Once you are finished simulating the auction, I want you to do the following analysis.

1. Show me summary statistics of the winning bids i.e. use the summary() function on the winning bids and also show me the variance. Compare these to the mean and variance of the log normal distribution. Can you confidently say the distribution of the valuations and the distribution of the bids are different[5]?

   Hint: The mean of the log normal distribution is $e^{\mu+\frac{\sigma^2}{2}}$ and the variance is $(e^{\sigma^2}-1)(e^{2\mu+\sigma^2})$. In our case $\mu = 1$ and $\sigma^2 = 1$.

2. Create a data set called "winners" that contains three variables for *only* winning bidders:

   (a) The bidder number: called "bidder_num,"

   (b) Her valuation: called "valuation,"

   (c) Her bid: called "bid."

3. Run a regression of "bid" on "valuation" (that is, bid is the dependent variable and valuation is the independent variable). Print the summary of the regression and interpret the coefficients.

[5] You do not need to do a formal statistical test to answer this question.