ECON 370: Assignment 2

Drew Van Kuiken

Due Wednesday, October 16, 2024

No AI on this assignment!!![1]

First, at the beginning of your script, set your seed to 123. That is, add set.seed(123) in the first part in your document. Remember to include all group members names as a comment in the header (i.e., the beginning section of comments) in your group's script.

Logic

Subsetting Data with Logical Statements

Read in the HTdata_labels.xlsx. Call this data HT_data **(5 points)**. Note: you cannot read excel files into R using read.csv or read_csv(). Instead, check out the **readxl** package. To use the readxl package, you'll need to load the tidyverse, then load the readxl package, and finally, use the function read_excel().

Create subsets of the data using the following conditions:

1. Auctions with plots in Zone 2. Call this subset HT_data_Zone2 **(3 points)**.
2. Auctions with Delta values between 0.04 and 0.05. Call this HT_data_Delta45 **(3 points)**.
3. Auctions with plots of Species Concentration greater than 0.5. Call this HT_data_Con **(3 points)**.
4. Auctions with 2 or 3 bidders. Call this HT_data_Bidders23 **(3 points)**.
5. Auctions with 4 bidders in 1982. Call this HT_data_4Bidders1982 **(3 points)**.

[1]I know I said you can use AI in the intro to our class. But I don't see how I can make problem sets difficult enough to be useful if you all are using AI on them. So: new policy is that you can use AI on the final project and for the problem set that involves prepping your final project, but you cannot use it for problem sets that ask you to solve coding problems like this one.

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.*** **(3 points)**

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 points)**

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". **(3 points)**

4. Create a function that returns the sign of a vector of numbers. Call this function "my_sign_vec". **(3 points)**

5. Write a function called CRRA that takes in two inputs: a vector 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 *a vector of length greater than one* for c but only one value of η. **(3 points)**

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. **(3 points)**

7. Create the following object and calculate the mean, median, min, max and standard deviation of each row and column. Name each output "x_y" where x is the statistic that is being calculated and y is either row or column. **(3 points)**

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

The Wage Gap: Bad Controls

Policymakers (and laypeople) have spent lots of time discussing the wage gap, which measures the difference in average earnings between men and women. Estimating the wage gap involes some interesting and relevant econometric issues. The average wage of women is roughly 80% of the average wage of men. Some argue though that this statistic is misleading because men and women choose jobs that pay differently. As such, these people argue that occupation should be controlled for in order to get a better picture of the "true" wage gap.

For this question, we'll assume that discrimination exists in the workplace. There's lots of good research on how discrimination leads to a wage gap. Last year, economist Claudia Goldin won the Nobel prize for her work on gender and the workplace. To read more on this, check out **this overview of Claudia Goldin's work**.

Model

- Demographics: There are $N = 10,000$ men and women where each agent has equal probability of being assigned male or female.
- Markets: There is a labor market with two types of jobs, a low-paying and high-paying job. Men are assigned the high-paying and low-paying job with equal probability. However, women are assigned the high paying job only a quarter of the time. Note that this is a very simplified version of a model where there are actual choices and those choices depend on unobservables.
- Wages: The low-paying job pays \$15 and the high-paying job pays \$30.
- Timing: The timing of the model is as follows,
  1. Agents are born.
  2. Agents are assigned to jobs
  3. Agents realize their wages.

Questions

1. (Points 10) What is the data generating process and what are the data?
2. (Points 10) Run a Monte Carlo simulation of this model. There should be $N_{rep} = 10,000$ Monte Carlo replications of the model. Remember, this is different from the sample size. After you have the wage, job, and sex data, run three different regressions: one with wage as the dependent variable and sex as the only independent variable, one with wage as the dependent variable and job as the only independent variable, and one with wage as the dependent variable and sex and job as the independent variables. Save the coefficients from the regression in three separate matrices.

   Note: This may take some time to run on your computer.
3. (Points 10) Look at the mean of the coefficients in each regression across models. Compare the mean of the sex coefficient in the two models that control for sex. What does this tell us about controlling for occupation in this model? What kind of bias does adding controls for occupation give rise to?

Revenue Equivalence: Auction Simulation

Relevant Auction Theory Background

Empirical auctions, which are rooted in auction theory, is a popular and important area in my subfield of economics known as industrial organization. Auctions have been one of the recent successes in economic theory as they have played a vital role in various aspects of our daily lives including allocating electromagnetic spectrum ("radio waves"), selling timber plots to forestry companies, and most notably, selling advertising space on websites to companies based on the characteristics of site visitors. In fact, Paul Milgrom and Robert Wilson won the 2020 Nobel Memorial Prize in Economic Sciences[2] for their work in developing auction theory. Designing auctions to maximize revenue or social welfare are two common goals of auction theory. One surprising result from auction theory is that given rather general conditions on bidder preferences and the auction format[3], every auction format that meets these conditions will result in the same average amount of revenue.[4] In this question, will consider two different auction formats:

1. First-price sealed bid,
2. Second-price sealed bid.

We have covered the difference between these auction formats in class, but to reiterate: in a first-price sealed bid auction, bidders submit their bids anonymously and whoever submitted the highest bid wins and pays their bid whereas with a second-price sealed bid auctions, bidders submit their bids anonymously and the person who submits the highest bid wins, however, she only pays the bid of the *second highest* bidder (or the highest, losing bid).

To be clear, revenue equivalence means that **on average**, both of these auctions will result in the same revenue. This might be surprising given that, for example, second-price auctions seem to have lower revenue than first-price auctions since in the first-price auction bidders pay their own bid whereas in the second-price auctions, winners pay the second highest bid. However, this reasoning fails to consider how the *auction format itself* affects *optimal bidding behavior.* That is, the optimal bid in the first-price auction is different (and specifically, lower) than the optimal bid in the second-price auction due to the design of the auction. It turns out that these two effects perfectly cancel out and the first-price and second-price auction result in the same revenue on average.

Your optimal bid in the first-price sealed bid auction is as follows:

$$b_i = v_i - \frac{\int_0^{v_i} F_v(x)^{N-1}dx}{F_v(v_i)^{N-1}} \tag{3}$$

[2]The "Nobel Prize in Economics."

[3]Utility functions are quasi-linear in "money," the object always goes to agent with the highest signal, and any bidder with the lowest possible signal receives 0 net utility.

[4]That is, the auctions will result in the same revenue "on average" (or in expectation) regardless of the format.

where N is the number of bidders (including yourself), v_i is your valuation, and F_v is the distribution function that v_i comes from. If this scares you or does not mean anything to you, that is fine. I have given you the code for the bidding function to compute the optimal bids, i.e. the formula above, in HW3_Code.R. All you need to understand conceptually is that the optimal bid in the first-price sealed bid auction is to "shave down" one's valuation (v_i) based on how competitive the auction is and how "varied" the valuations are.[5] This is intuitive since if you're paying your bid, you don't want to pay exactly $b_i = v_i$ as if you win as then your utility is $v_i - b_i = v_i - v_i = 0$. If you could "shave down" (or decrease) your bid by some small amount (that is $b_i = v_i - \varepsilon_i$ where $\varepsilon_i > 0$), you could get a higher level of utility since $v_i - b_i = v_i - (v_i - \varepsilon_i) = \varepsilon_i > 0$. That said, if you decrease your bid by shaving down your valuation *too much*, you risk no longer having the highest bid and losing the auction. Optimal bidding in a first-price sealed bid auction balances these two effects.

However, the optimal bid in the second-price auction is simply,

$$b_i = v_i, \tag{4}$$

that is, it is optimal for someone to bid exactly their valuation.[6] However, the *revenue* for this auction is not the highest bid, but the second highest bid. Again, note that in the first-price auction, the revenue is simply the highest bid; however, in the second-price auction, the revenue is the second-highest bid as the winner pays the second highest bid.

Testing Revenue Equivalence

We will run a simulation that tests the idea of revenue equivalence. To do this, suppose that $v_i \sim \text{Exp}(1/10)$ and each auctions has $N = 5$ bidders. So for each simulation, you will have 5 new draws from Exp(1/10). Use these five draws to determine the revenue in the first-price and second-price sealed bid auctions. So in the second-price sealed bid auction, the revenue is simply the *second largest* of the five draws. For the first-price sealed bid auction, use the function b_i that I supplied you with this problem set. It has two arguments that you need to supply: v and N. v should be the *largest* of the five draws in each simulation and N should be set to 5.

For this question, it will be helpful to write a function called SimulateAuctions that has one argument, $Nsim$. For a given value of $Nsim$, SimulateAuctions should run all $Nsim$ auctions, store the revenue of the two auction formats in vectors, and return a *list* of the two revenue vectors. The steps for the simulation should be as follows[7]:

[5] **If you're curious where this comes from, see Subsections 1.1 and 1.3 of this linked document.**

[6] **If you're wondering why this is the case, see Subsection 1.2 of this linked document.**

[7] Throughout these steps, I will tell you what to name objects and then in parentheses and quotes, I will give an explanation as to why.

Instructions for SimulationAuction

1. While it might seem easier to do this with a loop, due to the way I wrote b_i, it is *much* faster to do all the simulations at once. To do this, draw a vector of length $N \times Nsim = 5 \times Nsim$ from Exp(1/10). Call this vector sims ("simulations"),

2. Create a *matrix* that is $Nsim \times 5$ using the vector sims created in step 1. Call this matrix Vs ("valuations"),

3. Sort *each row* of Vs from smallest to largest and store this new sorted matrix as a new object called Vs_s ("valuations sorted"). Hint: Due to the way apply works, this will flip the rows and the columns. So Vs_s[,1] are the five valuations for the first simulation, Vs_s[1,1] is the smallest valuation in the first simulation, Vs_s[4,1] is the second largest valuation in the first simulation, Vs_s[5,1] is the largest valuation in the first simulation, and Vs_s[,i] is equivalent for any other $i = 1, 2, ..., Nsim$. So in Vs_s, the *columns* are the simulations and the *rows* are the 5 valuations for each simulation,

4. Create a vector that is the fourth row from Vs_s ($N - 1$ since $N = 5$) called rev_SP ("revenue second-price") as this is the revenue in each simulation of the second-price auction,

5. Use the supplied bidding function, $b_i(v, N)$, to get the bid of the largest valuation in each simulation, or the fifth row of Vs_s (since $N = 5$), and store this as a vector rev_FP ("revenue first-price"). That is, the optimal winning bid (and hence the revenue in a first-price auction) in each simulation can be obtained using b_i(Vs_s[5,],5),

6. Lastly, return a list with two elements, rev_FP and rev_SP, with the same names.

Questions

1. (Points 25) Run the simulation described above using $Nsim = 5$ and compare the average revenue of the first-price and second-price sealed bid auctions. Make sure to do this by writing the function SimulateAuction described above that has one argument: $Nsim$. Hint: Use the sapply function with the output of SimulateAuctions(5) as the X argument and the mean function as the FUN argument. How close are the estimated average ("expected") revenues?

2. (Points 5) Rerun question 1 using $Nsim =$ 10, 100, 1000, and 10000. What happens to the expected revenue of the two auctions? Does this match what you would expect under revenue equivalence?

3. (Points 15) Write a function called SimulateAuction_Loop that also takes in one argument, $Nsim$, that runs the simulation as a loop instead. See the modified instructions below. Compare how long it takes to run SimulateAuction(100) and SimulateAuction_Loop(100) using the system.time function. That is, run the following code on two different lines:

   - VecTime = system.time(SimulateAuctions(100))
   - LoopTime = system.time(SimulateAuctions_Loop(100))

and compare the outputs using this code: as.numeric(LoopTime[3]/VecTime[3]). This will tell you how many times faster VecTime is than LoopTime. On my computer, it is roughly 21 times faster.

4. (Points 5) If we were to "generalize" the function SimulateAuctions by adding more arguments than just $Nsim$ to allow for more general simulations, what arguments could we add to the function? *This question will require some thinking.* Hint: Think of numbers and functions we "hard-coded" in the simulation that I told you their value in the first paragraph of the "Testing Revenue Equivalence" subsection above i.e. we set specific values for this simulation when in actuality, they could've been variables. There are at least three potential answers that I am thinking of; two are easy, one is a bit harder. To receive full credit, I want to you name at least two of these three.

Modified Instructions for SimulateAuction_Loop

1. We will loop through all $Nsim$ auctions. To do this, initialize vectors to store the revenues from the simulations first and second price auctions. It is easiest to just use rep(0,$Nsim$). Call these two vectors rev_FP ("revenue first-price") and rev_SP ("revenue second-price"),
2. Start a loop. For each simulation, draw a vector of length $N = 5$ from Exp(1/10). Call this vector Vs ("valuations"). Suppose we are on simulation i,
3. Sort Vs from smallest to largest and store this sorted vector as a new object called Vs_s ("valuations sorted"),
4. Store the fourth value of Vs_s ($N - 1$ since $N = 5$) in the i^{th} spot in rev_SP as this is the revenue of the second-price auction,
5. Use the supplied bidding function, $b_i(v, N)$, to get the bid of the largest valuation, or the fifth value of Vs_s (since $N = 5$), and store this in the i^{th} spot in rev_FP. That is, the optimal winning bid in this simulation can be obtained using b_i(Vs_s[5],5),
6. Continue looping through all $i = 1, 2, 3, ..., Nsim$ simulations.
7. Once all $Nsim$ simulations are complete, create and return a list with two objects, rev_FP and rev_SP, with the same names.