Marra, Marleen. (2024), "Estimating an Auction Platform Game With Two-Sided Entry", *Journal of Political Economy*

Replication Package

July 23, 2024

1 Requirements and set-up

The code requires Matlab and R. The results are most recently generated with Matlab version R2023a, Update 3 and R version 4.2.1. To reproduce all results, create the following folder structure
base/Code
base/Output
All code and data files should be downloaded in the `Code` folder. Executing `Run_descriptive_stats.R` generates all results from the descriptive analysis of the data including Tables 2-4. Executing `Main.m` generates all results from the structural analysis including Tables 5-6 and Figures 1-4. More details about all subfunctions are provided below. If the base folder is saved in Dropbox, the code should run without any modification. Otherwise, the path at the top of `Run_descriptive_stats.R` should be modified.

2 Details of all files in the replication package

Data and main scripts

- `BFWauctiondata.csv` contains auction-level input data from the Bid for Wine auction platform. The data has 3,481 observations, identified by `auctionid`.
- `BFWpotentialsellerdata.csv` contains potential seller-level input data from the Bid for Wine auction platform. The data has 2,581 observations, identified by `sellerid`. This data is only used to generate Table 3.
- `Run_Descriptive_Stats.R` reproduces Tables 2-4 and can be run independently of all following code.
- `Main.m` reproduces Tables 5-6 and Figures 1-4, and generates additional structural estimation results that are referred to in the text. The main inputs to the functions can be tailored at the top of the script, in the `PAR` structure.

Other high-level files

- `estimation_func.m` obtains the structural estimates. It is called from `Main.m` and outputs a `Results` structure (called `Results_high` for the high-end sample), with the estimation results. It is also called from the bootstrap function.
- `bootstrap_func.m` samples from the relevant data and obtains bootstrapped standard errors across these samples. Results are outputted in a `BS` structure in the command window when calling the function from `Main.m`. Results can also be loaded from `Output/Bootstrap_results.mat`. This function can be run at any point after running `estimation_func.m`.
- `modelfit_func.m` runs model fit exercises. It is called from `Main.m`. The plots in Figure 1 are saved in the Output/ folder. The function also outputs additional statistics that are referenced in the text, in matrix form (called Modelfit). This function can be run at any point after running `estimation_func.m`.
- `Simulate_CF.m` simulates equilibrium outcomes on the platform. It is called from `Main.m` to produce the simulated baseline. It is also called from `counterfactual_func.m` to produce counterfactual outcomes for different fee structures, etc. Besides the fee structure, it can simulate results under the scenario that the entry decisions of bidders and/or sellers is not affected. This function can be run at any point after running `estimation_func.m`.
- `counterfactual_func.m` performs various counterfactual policy simulations. It is called from `Main.m`. Figures are saved in the Output/ folder. The raw results from the different counterfactuals are outputted to the workspace (CF* indicating a certain counterfactual, as detailed in the function). This function can be run at any point after running `estimation_func.m` and obtaining the baseline simulation results.

Lower-level files

These files contain functions that are needed to estimate the parameters (the first five files), and to compute the two-sided entry equilibrium (all remaining files), and they are listed roughly in order of first occurrence when running `Main.m`

- `bidderLL.m` Obtains the log likelihood of bids given values, to estimate parameters of the Generalized Gaussian distribution of bidder values, in auctions with no reserve price. Called from `estimation_func()`.
- `bidderLLlambda.m` Obtains the log likelihood of bids and number of observed bidders given values and number of bidders, to estimate the parameters of the Generalized Poisson distribution of the number of bidders, in auctions with positive reserve prices. Called from `estimation_func()`.

- `ggd_cdf.m` Obtains the Generalized Gaussian CDF given parameters.
- `ggd_pdf.m` Obtains the Generalized Gaussian CDF given parameters.
- `seller_LL.m` Obtains the log likelihood of implied seller values for sellers on the platform given seller valuation parameters. Corresponds to equation 24 in the paper. Called from `estimation_func.m`.
- `piBS_MC.m` Obtains matrices of listing-level bidder and seller surpluses, given commissions and value distribution parameters. Obtains matrices with in the first dimension the seller's value v_0 and in the second dimension the number of bidders n. Corresponds to equations 4-7 in the paper. The function runs `optreserve()`, `sampleHSHvalues()`, and `EBSsurplus()`, described directly below, and interpolates the results on a finer grid.
- `optreserve.m` Compute the optimal reserve price given bidder valuation parameters and seller valuation v_0.
- `sampleHSHvalues.m` Sample highest- and second-highest values of bidders and given the distribution of bidder values, given n and bidder valuation parameters.
- `EBSsurplus.m` Obtains the expected surplus of entering the platform for bidders and sellers, based on sampled values in `sampleHSHvalues()`, and applied on the same grid with in the first dimension the seller's value v_0 (relevant for the expected seller surplus and expected bidder surplus in $r > 0$ listings) and in the second dimension the number of bidders n.
- `knnimputation.m` K-nearest neighbour imputation of missing values if they occur in `sampleHSHvalues.m`. Not written by me. Provided Reference: *Speed, T. Statistical Analysis of Gene Expression Microarray Data (2003), Chapman & Hall. Copyright 2003-2004 The MathWorks, Inc. Revision: 1.1.12.1, Date: 2004/12/24 20:42:47.*
- `eSofunc.m` Obtains the seller entry cost as the value that sets the expected surplus from entering the platform equal to zero, given the listing-level surplus obtained with `piBS_MC()`, valuation parameters, and the equilibrium number of bidders in positive reserve auctions. Corresponds to equation 17 in the paper.
- `PiSeller.m` Obtains the expected seller surplus when entering with a certain v_0 when competing sellers' adopt entry threshold $\tilde{v_0}$ and bidders respond optimally. Corresponds to equation 12 in the paper.
- `lambdastarfunc.m` Obtains the equilibrium number of bidders for a given seller entry threshold value that satisfies the zero profit condition. Corresponds to equations 9 ($r = 0$) and 11 ($r > 0$) in the paper.
- `Pibidder.m` Obtains the expected bidder surplus when entering the platform given values, equilibrium number of bidders, and, in the case of $r > 0$ the seller's entry threshold $\tilde{v_0}$. Corresponds to equations 8 ($r = 0$) and 10 ($r > 0$) in the paper.

- `eBofunc.m` Obtains the bidder entry cost as the value that sets the expected surplus from entering the platform equal to zero, given the listing-level surplus obtained with `piBS_MC()`, valuation parameters, and the equilibrium number of bidders. Corresponds to equation 18 in the paper when $r = 0$ (zeroreserve, the second-to-last function entry is set to 1), and equation 19 when $r > 0$.

- `v0starfunc.m` Obtains the equilibrium entry threshold v_0^* given valuation parameters and fees. Calls `Piseller()`, which computes for each candidate $\tilde{v_0}$ the equilibrium mean number of bidders per listing with `lambdastarfunc()`, and how that translates into the expected seller surplus for a seller with $V_0 = \tilde{v_0}$. Corresponds to equation 13 in the paper.

- `my_ecdf.m` Obtains the empirical CDF of values on a grid.

- `Piplatform.m` Obtains a set of expected surpluses / revenues / equilibrium outcomes of platform configuration given fees, listing-level surpluses, values, sale probabilities, prices, volumes, all pre-calculated with `piBS_MC()`. Called in counterfactual simulations by the function `Simulate_CF()`.

- `matrix2latex` Function to take data in a 2-dimensional numerical or cell array and output it as a table in latex format. Adapted from the function written by M. Koehler and published under the GNU GPL by the free software foundation.

3 Reproducing Tables and Figures

The paper contains 6 Tables and 4 Figures. Table 1 below details where in the code they are created and where they are saved.

Table 1: List of all Tables and Figure in the paper and how to reproduce them

	Code file	Notes / Output
Table 1	Based on fees quoted on platform's website at the time	
Table 2	`Run_descriptive_stats.R`	saved to *Output/Table2.tex*
Table 3	`Run_descriptive_stats.R`	saved to *Output/Table3.tex*
Table 4	`Run_descriptive_stats.R`	saved to *Output/Table4.tex*
Table 5 - Panel I	`Main.m` [`estimation_func()` and `bootstrap_func()` both with `PAR.highvalue=0`]	saved to *Output/Table5_PanelI.tex*
Table 5 - Panel II	`Main.m` [`estimation_func()` and `bootstrap_func()` both with `PAR.highvalue=1`]	saved to *Output/Table5_PanelII.tex*
Table 6	`Main.m` [`counterfactual_func()`]	saved to *Output/Table6.tex*
Figure 1a	`Main.m` [`modelfit_func()`]	saved to *Output/Figure1a.png* (bidder value distribution)
Figure 1b	`Main.m` [`modelfit_func()`]	saved to *Output/Figure1b.png* (seller value distribution)
Figure 1c	`Main.m` [`modelfit_func()`]	saved to *Output/Figure1c.png* (homogenized bids)
Figure 1d	`Main.m` [`modelfit_func()`]	saved to *Output/Figure1d.png* (reserve prices)
Figure 1e	`Main.m` [`modelfit_func()`]	saved to *Output/Figure1e.png* (hammer prices $r > 0$)
Figure 1f	`Main.m` [`modelfit_func()`]	saved to *Output/Figure1f.png* (Poisson approximation)
Figure 2a	`Main.m` [`counterfactual_func()`]	saved to *Output/Figure2a.png*
Figure 2b	`Main.m` [`counterfactual_func()`]	saved to *Output/Figure2b.png*
Figure 3a	`Main.m` [`counterfactual_func()`]	saved to *Output/Figure3a.png*
Figure 3a	`Main.m` [`counterfactual_func()`]	saved to *Output/Figure3a.png*
Figure 4	`Main.m` [`counterfactual_func()`]	saved to *Output/Figure4.png*