ECON832 Midterm

Daniel Sánchez Pazmiño

March 2024

Preliminaries

This is the report for my ECON832 midterm responses. The attached files include all replicating code to obtain the results. The folders are organized as follows:

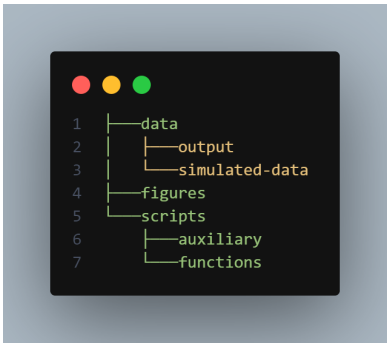

Figure 1: Repository files organization

Files have been attached to the Canvas submission along with this PDF document. A makefile `.sh` script has been included to run all relevant Julia scripts.

1 Problem 1: BLP Estimation

1.1 Data Analysis

In this part, I used `TidierData.jl`, Julia's implementation of R's *dplyr*, to reshape the simulated data to the format of the original BLP data, as seen in the `JuliaBLP` repository. This is present in the `problem1_1_data_analysis.jl` in the `scripts` subfolder of my project repository.

I assume that the provided data describes three products per market: MD, TH and SB, each produced by a different firm. Each will have a different market share for every `marketid`. I first pivot the attributes data to a *long format* (in a stepwise fashion, first the prices, then the caffeine scores), and then join them together based on their `marketid` and `product` columns. See below for a code snippet of the procedure for the price reshape procedure.

```
prices = @chain attributes_raw begin
    @select(marketid, price_MD, price_TH, price_SB)
    @rename(MD = price_MD, TH = price_TH, SB = price_SB)
    @pivot_longer(MD:SB, names_to = "product", values_to = "price")
end
```

After joining the prices and caffeine scores together based on auxiliary `markeitd` and `product` columns, I repeated the same procedure for the market shares data. I calculated an `outshr` variable (included in the original BLP data), even though I don't end up using it, since it is only used in the `Demand - OLS and 2SLS.jl` script, which is a biased estimation of elasticities, unlike BLP.

To calculate it, I calculated the total market share for each market and then obtained the `outshr` variable by subtracting the market share of each product from the total market share. After this was done, I joined the market shares data with the attributes data to obtain the final dataset. I noticed that market shares do *not* sum to 1; I discuss the implications of this in subsection 3 for this problem.

I also include the cost characteristic, w, in the dataset, which is the caffeine score cost attribute that is observed by firms. I reshape it in the same way as the other attributes, and then join it to the final dataset based on the `marketid` and `product` working columns.

I created a firm identifier `firmid`, as in the BLP dataset, by reassigning identifiers to the working `product` column so that MD is 1, TH is 2 and SB is 3. I also create a product id, `id`, which assumes that all products are different across markets, hence, it is simply a row number identifier when the dataset is ordered ascendently by T and J (the number of markets and firms, respectively). I added the constant term, `const`, which is a vector of ones, to the dataset, as it is used in the BLP estimation.

The final dataset was named `reshaped_product_data.csv` and saved to the `data/output` directory for later use. See Figure 2 for the required screenshot.

Note that even though this question did not ask for the reshaping of instruments for the following parts, I implemented procedures similar to the one I did here in the `problem1_2_elasticities.jl` script, which reshapes the instruments for the BLP estimation, as the provided instruments are not in a format that can be used directly in the code. I do not describe this process in those parts for brevity, but know that the same reshaping procedures were followed.

```
julia> print(first(product_data_final, 60))
60×9 DataFrame
```

Row	firmid	cdid	id	price	const	caffeine_score	caffeine_score_cost	share	outshr
	Int64	Int64	Int64	Float64	Int64	Float64	Float64	Float64	Float64
1	1	1	1	1.87059	1	0.966159	0.0866591	0.244752	0.0409142
2	2	1	2	2.58938	1	0.105214	0.0485192	0.0126449	0.273021
3	3	1	3	2.2279	1	0.217468	0.732495	0.0282693	0.257397
4	1	2	4	1.90561	1	0.597104	0.00554706	0.0641745	0.270079
5	2	2	5	1.94316	1	0.843626	0.92502	0.153911	0.180342
6	3	2	6	1.65621	1	0.632427	0.0154237	0.116168	0.218086
7	1	3	7	2.5564	1	0.3281	0.964173	0.0133731	0.323254
8	2	3	8	1.67734	1	0.559408	0.491374	0.260673	0.0759548
9	3	3	9	1.71425	1	0.132648	0.175674	0.0625817	0.274046
10	1	4	10	1.51839	1	0.734724	0.892568	0.238495	0.326803
11	2	4	11	1.77727	1	0.942066	0.683135	0.310983	0.254314
12	3	4	12	2.39762	1	0.329828	0.0717879	0.0158193	0.549478
13	1	5	13	1.48157	1	0.797715	0.83818	0.204769	0.144393
14	2	5	14	1.48253	1	0.248942	0.488632	0.105723	0.243439
15	3	5	15	1.68817	1	0.0457809	0.137744	0.0386699	0.310492
16	1	6	16	1.36698	1	0.0486979	0.591229	0.113191	0.180313
17	2	6	17	2.30454	1	0.111331	0.530425	0.0170729	0.276432
18	3	6	18	1.78607	1	0.932807	0.290873	0.163241	0.130264
19	1	7	19	1.25816	1	0.403728	0.756938	0.198994	0.184889
20	2	7	20	1.83666	1	0.121596	0.606263	0.0530895	0.330794
21	3	7	21	2.03104	1	0.820487	0.315431	0.1318	0.252083
22	1	8	22	2.13364	1	0.303147	0.732461	0.0502769	0.24215
23	2	8	23	2.22037	1	0.684572	0.425535	0.109572	0.182854
24	3	8	24	1.74801	1	0.44812	0.530246	0.132577	0.159849
25	1	9	25	2.27644	1	0.188089	0.646151	0.0392727	0.199458
26	2	9	26	1.75579	1	0.0643051	0.374395	0.0646748	0.174056
27	3	9	27	2.2468	1	0.532166	0.547228	0.134783	0.103947
28	1	10	28	3.11279	1	0.036368	0.328578	0.00329356	0.439389
29	2	10	29	1.52469	1	0.882325	0.606821	0.253587	0.189096
30	3	10	30	1.34203	1	0.872665	0.471676	0.185803	0.25688
31	1	11	31	3.70696	1	0.0569499	0.959992	0.00352714	0.0971869
32	2	11	32	2.92582	1	0.243401	0.846911	0.0150344	0.0856797
33	3	11	33	1.50736	1	0.00323062	0.804092	0.0821525	0.0185615
34	1	12	34	1.31275	1	0.0565118	0.695863	0.0658198	0.160483
35	2	12	35	1.81501	1	0.0140558	0.761441	0.0210199	0.205283
36	3	12	36	1.3835	1	0.223043	0.602643	0.139463	0.0868397
37	1	13	37	1.67111	1	0.387713	0.0541614	0.111015	0.169912
38	2	13	38	2.25783	1	0.113435	0.528514	0.0175772	0.26335
39	3	13	39	2.06816	1	0.823256	0.370579	0.152335	0.128592
40	1	14	40	1.28716	1	0.237104	0.271388	0.158063	0.0423744
41	2	14	41	2.02063	1	0.0990223	0.88457	0.0288352	0.171602
42	3	14	42	2.75696	1	0.180969	0.91056	0.0135392	0.186898
43	1	15	43	1.67388	1	0.421931	0.554789	0.047067	0.320944
44	2	15	44	1.99908	1	0.613222	0.627893	0.158364	0.209646
45	3	15	45	1.34586	1	0.31688	0.349504	0.162579	0.205431
46	1	16	46	2.59174	1	0.260723	0.790881	0.0264511	0.392139
47	2	16	47	1.44792	1	0.850506	0.709705	0.28182	0.136771
48	3	16	48	1.5719	1	0.299419	0.603771	0.11032	0.308271
49	1	17	49	1.62782	1	0.871864	0.768669	0.272138	0.237835
50	2	17	50	2.21325	1	0.418754	0.459769	0.0370538	0.472919
51	3	17	51	1.50988	1	0.724352	0.770408	0.200782	0.309192
52	1	18	52	2.26916	1	0.902684	0.58837	0.0970515	0.305844
53	2	18	53	1.80543	1	0.912218	0.350186	0.216484	0.186411
54	3	18	54	1.67024	1	0.420255	0.834764	0.0893598	0.313536
55	1	19	55	1.39374	1	0.679047	0.0682939	0.158172	0.205174
56	2	19	56	1.23677	1	0.411877	0.581239	0.191493	0.171853
57	3	19	57	2.54424	1	0.134262	0.0139052	0.013681	0.349666
58	1	20	58	2.83829	1	0.207545	0.16674	0.012011	0.338399
59	2	20	59	1.60404	1	0.761308	0.0633265	0.215099	0.135311
60	3	20	60	2.17269	1	0.915269	0.152264	0.1233	0.22711

Figure 2: Product Data

1.2 BLP estimation with the provided instruments

Table 1 below shows the results of the BLP estimation using the provided instruments. The replicating code for these results can be found in the `problem1_2_estimation.jl` script in the `scripts` subfolder. Dependencies are `demand_functions.jl`, `demand_derivatives.jl`, and `demand_instruments.jl`, found in the `scripts/functions` directory. These are the modified functions from the `JuliaBLP` repository.

Table 1: Average elasticities with provided instruments

Elasticity	**Estimate**
Price	-1.05184
Caffeine Score	2.04923
Caffeine Cost	0.228144

Compared to the simulation parameters of β = -1.5 and $\gamma = 0.1$, the estimation does not fall too far from the true values. My estimated price elasticity of demand is -1.05, which means that an additional increase of 1% in the price of a product will decrease the share of that product by 1.05%. On the other hand, my estimated caffeine score elasticity is 2.04, which means that an additional increase of 1% in the caffeine score of a product will increase the share of that product by 2.04%.

Finally, regarding my supply side parameter, I estimate a caffeine score cost elasticity of 0.22, which means that an additional increase of 1% in the caffeine score of a coffee product will increase the marginal cost of that product by 0.22%. Note that for this elasticity, I performed 2SLS to instrument for the endogenous cost characteristic in the regression of the marginal cost (in log form) on the caffeine score cost characteristic. The instruments were the ones provided in the simulated data for this exam.

Regarding best practices, I follow recommendations set out in the `PyBLP` article (Conlon and Gortmaker 2020) to choose my tuning parameters and optimizers for the estimation process. I only applied the recommendations that applied to the case of this simplified BLP problem, since best practices are only applicable to more complex problems.

First, to define my simulated consumers, I used a multivariate normal distribution with mean zero and variance one, using Julia's `MvNormal` function from the `Distributions` package. I tried different values until I eventually selected a sample size s of for the demand side simulation, but it can be changed, conditional on computer performance.

Additionally, following the article, I changed tolerance for solving shares, ϵ_{tol}, to 1e-12, which falls within the recommended range of 1e-12 to 1e-14. Previously, the tolerance had been set higher, which propagated error to the estimation process. This change was reflected in the `demand_functions.jl` script.

Another recommendation was to not use the Nelder-Mead algorithm in the optimization routine for the estimation process. I used the `Optim` package in Julia, which has a variety of optimization algorithms, and choose not to use the Nelder-Mead algorithm. Currently, the script includes the `LBFGS` algorithm, `BFGS` and gradient descent methods, which are derivative based and are recommended for BLP estimation. I also decreased the tolerance to 1e-12 and increased the maximum amount of iterations for each of these algorithms.

1.3 BLP estimation with code-generated instruments

Table 2: Average elasticities with the code-generated instrument

Elasticity	**Estimate**
Price	-1.72942
Caffeine Score	2.12977
Caffeine Cost (OLS)	1.05857
Caffeine Cost (2SLS)	1.29218

Table 2 shows the results of the BLP estimation using the code-generated instrument(s). The replicating code for these results can be found in the `problem1_3_estimation.jl` script. Dependencies are the same as in the previous subsection. This estimation has several caveats which I discuss below.

My estimate for β is not too far off from its true value. I report two estimates for γ, one from the OLS procedure which simply regresses the price on the caffeine cost attribute, and another from the 2SLS procedure, which uses the generated instrument to estimate the caffeine score cost elasticity. The -1.73 estimate for the average price elasticity is higher than the one I found in the last part, the caffeine score elasticity in this part is only marginally different, and the caffeine score cost elasticity estimate are both higher than the ones I found in the last part. Both estimates for the cost elasticity are likely biased, as they are higher from their true value of 0.2. This is likely due to the fact that the generated instrument is not valid, as I discuss below.

The provided code in the `JuliaBLP` repository states that there are two sets of instruments: characteristics of *other* products from *same company* in *same market*, as well as the characteristics of other products from *other companies* in *same market*. Because in this setting every company only produces one product, the first set of instruments does not apply, so I cannot construct that instrument[1]. I am definitely able to construct the second set of instruments,

[1]By fixing the dimensions of the code of the `BLP_instruments` function in the `JuliaBLP` repository, I am actually able to get the `IV` matrix, but it produces a matrix that induces a singular matrix in the 2SLS regression. This is because the procedure of getting instruments returns the same cost characteristics, thus there is perfect collinearity, which explains the presence of a singular matrix.

which is the instrument that I use in the demand-side estimation of average price and caffeine score elasticities. However, because the inclusion of both instruments was needed, I am unable to ensure the exclusion restriction and thus the validity of BLP in this part. This might make my θ biased.

This seems to affect the cost elasticities more than the demand elasticities. I performed 2SLS by recreating the `BLP_instruments` function code to obtain cost charactersitics for products made from other companies in the same companies. I then used 2SLS to instrument for the endogenous cost characteristic in the regression of the marginal cost (in log form) on the caffeine score cost characteristic. I am not fully sure that I meet exclusion restrictions with only one instrument. Further, it's possible that because the shares do not sum to one, my procedure to find instruments is not adequate.

2 Problem 2: Merger Simulation

2.1 Setup

To simulate a merger between firms TH and MD (`firmid`'s 1 and 2), I was easily able to calculate the caffeine scores x and caffeine score cost attributes w for both scenarios using the provided data and the scenarios. Using my estimates for the baseline scenario which I obtained in Problem 1.1, I used the following equation to estimate marginal cost.

$$\ln mc_j = w_j\theta_3 + \omega_j$$

where w_j contains the modified caffeine score cost attributes for merged firms and the old cost attributed for firm SB (`firmid` $= 3$). Everything else would stay the same as in the baseline scenario.

The most important principle is that firms profit maximize as follows:

$$\Pi_f = \sum_{j \in \mathcal{F}_j} \left(p_j - mc_j\right) Ms_j(p, x, \xi; \theta)$$

which involves the following first order conditions:

$$s_j(p, x, \xi, \theta) + \sum_{r \in \mathcal{F}_f} \left(p_r - mc_r\right) \frac{\partial s_r(p, x, \xi; \theta)}{\partial p_j} = 0.$$

$$\Delta_{jr} = \begin{cases} -\frac{\partial s_r}{\partial p_j} & \exists f : r, j \in \mathcal{F}_f \\ 0 & \text{otherwise} \end{cases}$$

Given that this is a counterfactual setting, this is the inverse problem to the one I solved in the previous problems, I have θ, ξ, x, mc yet need to find p, with which I can s. This a non-linear problem, which I can solve by minimizing the first equation (recognizing that symbolically, it should be as small as possible or equal to zero.) I minimize with respect to p to find the equilibrium price.

$$\min_p s_j(p, x, \xi, \theta) \sum_{j \in \mathcal{F}_f} \left(p_j - mc_j\right) \frac{\partial s_r}{\partial p_j}$$

The share function with respect to price is given by the predicted share function σ_j, which was shown to have only δ with any set of market shares s_j and a contraction mapping Θ. The function in its simplified format (without an integral) is the following:

$$s_j = \sigma_j\left(\delta, \theta; P^{ns}\right) = \sum_{r=1}^{ns} \frac{\exp\left(\delta_j + \sum_k x_{jk}\theta_{2k}\nu_{rk}\right)}{1 + \sum_s \exp\left(\delta_s + \sum_k x_{jk}\theta_{2k}\nu_{rk}\right)}$$

I would already have everything in this equation except for δ. I know that δ can be expressed as follows:

$$\delta_j = x_j\theta + \xi_j x_j^*\theta_1 + p_j\theta_2 + \xi$$

which are all values that I already have except for the price. I would also need to estimate the $\frac{\partial s_r}{\partial p_j}$, which is the derivative of the share function with respect to price. This is already given to me in the code of the `JuliaBLP` repository in the following way:

$$\frac{\partial s_r}{\partial p_j} = \int_v \left(\alpha + v_{ikp}s_v\right)\tau_j(1 - \tau_j)f(v)dv$$

where τ_j is the interior of the integral of σ_j.

With all this in mind, this can be solved using Julia's `Optim` package. Unfortunately, due to time constraints, I was unable to implement this in code, but the problem is defined in this subsection.

After being able to do this, I would be able to compute consumer and producer surplus for this counterfactual scenario. I should also compare to a calculation for baseline surplus. If the merger counterfactual surplus is higher than the baseline surplus, then the merger could be accepted by the regulatory authority. Intuition tells me than in the efficiency scenario and an elastic demand of $\beta <$ -1, the firms would be able to exert market power to increase prices and approach monopoly pricing. In this scenario, the deadweight loss would make total society surplus lower than in the baseline scenario.

In the average cost scenario, the firms are not able to exert market power to increase prices that much, so the contrary would happen. The deadweight loss is not enough to make total consumer surplus lower than in the baseline scenario.

References

Conlon, Christopher, and Jeff Gortmaker. 2020. "Best Practices for Differentiated Products Demand Estimation with PyBLP." *The RAND Journal of Economics* 51 (4): 1108–61. https://doi.org/10.1111/1756-2171.12352 .