# Problem Set 1 – Industrial Organization I

Guilherme Jardim (Yale University)

1. Code provided in Appendix A.

2. I conduct the simulation using 500 draws of the random coefficients. Comparing with 1,000 and 10,000 draws, the results for the derivatives were similar. To compute the derivatives, I follow Morrow and Skerlos (2011) breaking up the matrix of demand derivatives into two parts, a Jt $\times$ Jt diagonal matrix $\Lambda_t$, and a Jt $\times$ Jt dense matrix $\Gamma_t$:

$$\begin{aligned}
\frac{\partial \boldsymbol{s}_t}{\partial \boldsymbol{p}_t}\left(\boldsymbol{p}_t\right) &= \Lambda_t\left(\boldsymbol{p}_t\right) - \Gamma_t\left(\boldsymbol{p}_t\right) \\
\Lambda_{jj,t} &= \int \alpha s_{ijt}\left(\boldsymbol{\mu}_{it}\right) f\left(\boldsymbol{\mu}_{it} \mid \widetilde{\theta}_2\right) \mathrm{d}\boldsymbol{\mu}_{it} \\
\Gamma_{jk,t} &= \int \alpha s_{ijt}\left(\boldsymbol{\mu}_{it}\right) s_{ikt}\left(\boldsymbol{\mu}_{it}\right) f\left(\boldsymbol{\mu}_{it} \mid \widetilde{\theta}_2\right) \mathrm{d}\boldsymbol{\mu}_{it}
\end{aligned} \tag{1}$$

where $s_{ijt} = \frac{\exp(\delta_{jt}+\mu_{ijt})}{\sum_{k\in J_t}\exp(\delta_{kt}+\mu_{ikt})}$. I approximate the integral using pseudo-Monte Carlo (pMC) integration.

To solve the system of equations implied by the First Order Conditions, I employ two distinct methods: Julia's "nlsolve" function (equivalent to MATLAB's "fsolve") and the approach outlined by Morrow and Skerlos (2011). The results show convergence for all prices, with 96% of products yielding identical prices across both methods. However, notable differences emerge in certain markets, with discrepancies reaching up to \$18, though the average difference is \$0.11.

Conlon and Gortmaker (2020) show that the Morrow and Skerlos (2011) method reliably identifies equilibria and recommend it as their preferred approach. Furthermore, in markets where the two methods diverge, "nlsolve" tends to produce anomalously high prices. Given these considerations, I opt to use the prices derived from the Morrow and Skerlos method for subsequent analysis. Table 1 shows summary statistics for prices.

| Statistic | Value |
|---|---|
| Mean | 2.750 |
| Minimum | 1.778 |
| 1st Quartile | 2.459 |
| Median | 2.699 |
| 3rd Quartile | 2.971 |
| Maximum | 5.051 |
| N | 2,400 |

Table 1: Summary Statistics for Prices using Morrow and Skerlos (2011)

3. Table 2 shows summary statistics for market shares.

| Statistic | Value |
|---|---|
| Mean | 0.183 |
| Minimum | 0.001 |
| 1st Quartile | 0.086 |
| Median | 0.166 |
| 3rd Quartile | 0.253 |
| Maximum | 0.741 |
| N | 2,400 |

Table 2: Summary Statistics for Market Shares

4. I use the BLP approximation of the optimal instruments in addition to $x$ and $w$. I present the result of the regressions of prices and market shares on the exogenous variables and instruments below.

Instruments appear to be providing enough variation, as shown by the significant coefficients and the high $R^2$ in the regression. The sign of the parameters are also consistent with what we would expect. Therefore, I keep the suggested parameter choices.

| | prices | shares |
|---|---|---|
| | (1) | (2) |
| (Intercept) | 2.265*** | 0.178*** |
| | (0.021) | (0.007) |
| x | 0.120*** | 0.086*** |
| | (0.009) | (0.003) |
| satellite | 0.015 | 0.007* |
| | (0.011) | (0.004) |
| w | 0.471*** | -0.082*** |
| | (0.009) | (0.003) |
| x_instrument | -0.023*** | -0.015*** |
| | (0.005) | (0.002) |
| w_instrument | 0.021*** | 0.015*** |
| | (0.005) | (0.002) |
| Estimator | OLS | OLS |
| $N$ | 2,400 | 2,400 |
| $R^2$ | 0.567 | 0.404 |

5. Below, I present the results from the plain multinomial logit model of demand by OLS (ignoring the endogeneity of prices). In practice, I'm estimating the regression:

$$\delta_{jt} \equiv \log(s_{jt}) - \log(s_0) = \beta_0 + \alpha \cdot \text{prices}_{jt} + \beta_1 \cdot x_{jt} + \beta_2 \cdot \text{satellite}_{jt} + \varepsilon_{jt} \qquad (2)$$

As expected, the price coefficient is lower, in absolute value, than the true one.

| | $\delta$ |
|---|---|
| | (1) |
| (Intercept) | 1.719*** |
| | (0.120) |
| prices | -1.090*** |
| | (0.044) |
| x | 0.814*** |
| | (0.029) |
| satellite | 0.044 |
| | (0.035) |
| Estimator | OLS |
| $N$ | 2,400 |
| $R^2$ | 0.335 |

6. To address endogeneity issues, I re-estimate the multinomial logit model from 2 using two-stage least squares (2SLS). This approach instruments for prices using excluded cost shifters ($w$), and $x, w$ summed over competing goods $-j$ in market $t$. The results of this estimation are presented below. The first-stage regression yields an $R^2$ of 0.57, providing strong evidence that the relevance condition is satisfied.

The 2SLS estimates show parameters that align more closely with the true values compared to the OLS estimates. Notably, the price coefficient shows a higher absolute value, which is consistent with our expectations given the positive correlation between prices and the demand unobservable. This suggests that the 2SLS approach has successfully mitigated the upward bias in the price coefficient caused by endogeneity. It's interesting to note that the intercept term in the model captures the mean of the random coefficients.

| | $\delta$ |
|---|---|
| | (1) |
| (Intercept) | 4.000*** |
| | (0.172) |
| x | 0.921*** |
| | (0.031) |
| satellite | 0.068 |
| | (0.038) |
| prices | -1.955*** |
| | (0.064) |
| Estimator | 2SLS |
| $N$ | 2,400 |

7. I estimate the nested logit model by linear IV, treating "satellite" and "wired" as the two nests for the inside goods. The estimation incorporates not only the standard within-group share term $\log(\bar{s}_{j/g(j),t})$ (following Berry, 1994), but also its interaction with the satellite dummy variable. In practice, I'm estimating the regression:

$$\begin{aligned}\log(s_{jt}) - \log(s_0) = \beta_0 + \alpha \cdot \widehat{\text{prices}}_{jt} + \beta_1 \cdot x_{jt} + \beta_2 \cdot \text{satellite}_{jt} + \\ \sigma_1 \cdot \widehat{\log(\bar{s}_{j/g(j),t})} + \sigma_2 \cdot \text{satellite}_{jt} \widehat{\times \log}(\bar{s}_{j/g(j),t}) + \varepsilon_{jt}\end{aligned} \tag{3}$$

instrumenting for prices, the standard within-group share and its interaction by using excluded cost shifters ($w$), $x$ and $w$ summed over competing goods $-j$ in market $t$ and of the other good in the same nest as good $j$.

This specification enables the estimation of nest-specific parameters, allowing for potentially different correlation structures within each nest. The nesting parameters are derived as follows:

(i) For the wired group: The coefficient on $\log(\bar{s}_{j/g(j),t})$

(ii) For the satellite group: The sum of the coefficient on $\log(\bar{s}_{j/g(j),t})$ and the coefficient on its interaction with the satellite dummy

| | $\log(s_{jt}) - \log(s_0)$ |
|---|---|
| | (1) |
| (Intercept) | 3.918*** |
| | (0.203) |
| x | 0.874*** |
| | (0.042) |
| satellite | 0.038 |
| | (0.037) |
| prices | -1.887*** |
| | (0.094) |
| $\log(\bar{s}_{j/g(j),t})$ | 0.079 |
| | (0.087) |
| satellite $\times$ $\log(\bar{s}_{j/g(j),t})$ | -0.037 |
| | (0.112) |
| Estimator | 2SLS |
| $N$ | 2,400 |

8. First, let's derive the price elasticities implied by the nested logit model. We have 3 cases:

- Own-price:

$$\frac{\partial s_j}{\partial p_j}\frac{p_j}{s_j} = \frac{p_j}{s_j}\alpha \cdot \frac{1}{1-\sigma_{g(j)}} s_j \left(1 - \sigma_{g(j)}\bar{s}_{j|g} - (1-\sigma_{g(j)})s_j\right).$$

- $j$ and $k$ in same group $g(j)$:

$$\frac{\partial s_j}{\partial p_k}\frac{p_k}{s_j} = -\frac{p_k}{s_j}\alpha \cdot s_k \left(s_j + \frac{\sigma_{g(j)}}{1-\sigma_{g(j)}}\bar{s}_{j|g}\right)$$

- $j$ and $k$ in different groups:

$$\frac{\partial s_j}{\partial p_k}\frac{p_k}{s_j} = -\frac{p_k}{s_j}\alpha \cdot s_j s_k$$

| Statistic | Estimated Elasticities (Nested Logit) | True Elasticities | \|Difference\| |
|---|---|---|---|
| Mean | -4.684 | -4.189 | 0.495 |
| Minimum | -10.865 | -10.008 | 0.191 |
| 1st Quartile | -5.264 | -4.764 | 0.404 |
| Median | -4.473 | -3.988 | 0.493 |
| 3rd Quartile | -3.899 | -3.395 | 0.579 |
| Maximum | -2.097 | -1.782 | 1.040 |

Table 3: Comparison of Nested Logit Estimated and True Own-Price Elasticities

Then, I compute the estimated own-price elasticities and compare with the true ones. Results are shown in Table 3. Across all statistics, the estimated elasticities tend to be more elastic (more negative) than the true elasticities. The differences are relatively small.

Now, I compare the true diversion ratios and the ones implied by the estimates. Diversion ratio from $j$ to $k$ is given by $-\frac{\partial s_k/\partial p_j}{\partial s_j/\partial p_j}$. Table 4 presents the results. On average, the estimated diversion ratio is slightly higher than the true value, but the difference is small.

| Statistic | Estimated (Nested Logit) | True | \|Difference\| |
|---|---|---|---|
| Mean | 0.226 | 0.223 | 0.043 |
| Minimum | 0.001 | 0.001 | 0.000 |
| 1st Quartile | 0.114 | 0.107 | 0.021 |
| Median | 0.211 | 0.200 | 0.040 |
| 3rd Quartile | 0.316 | 0.317 | 0.061 |
| Maximum | 0.803 | 0.798 | 0.124 |

Table 4: Comparison of Nested Logit Estimated and True Diversion Ratios

9. Table 5 presents estimates of the demand parameters and standard errors when estimating demand alone. Table 6 presents results estimating jointly with supply. The differences suggest that using optimal instruments along with supply-side restrictions lead to more precise estimates, particularly for the nonlinear coefficients. However, the linear demand parameters appear to be stable across both estimation methods.

| | Variable | Estimate | Robust SE |
|---|---|---|---|
| Nonlinear Coefficients | satellite | 0.960 | 0.250 |
| | wired | 0.660 | 0.290 |
| Beta | Constant | 4.100 | 0.200 |
| | prices | -2.000 | 0.079 |
| | x | 0.970 | 0.043 |
| | satellite | 0.025 | 0.069 |

Table 5: Demand Parameters (Demand Only)

| | Variable | Estimate | Robust SE |
|---|---|---|---|
| Nonlinear Coefficients | satellite | 1.100 | 0.130 |
| | wired | 0.930 | 0.140 |
| Beta | Constant | 4.100 | 0.200 |
| | prices | -2.000 | 0.078 |
| | x | 0.970 | 0.040 |
| | satellite | 0.063 | 0.044 |
| Supply | Constant | 0.510 | 0.015 |
| | w | 0.250 | 0.006 |

Table 6: Demand Parameters (Estimated Jointly with Supply)

10. For my preferred estimates, I select those derived from the model jointly estimated with supply. Given the knowledge about the supply side, we know that it's correctly specified. Together with the optimal instruments, this gives us a good finite sample performance. Table 7 presents the true own-price elasticities and the ones implied by the preferred specification. Overall, these results suggest that the BLP model performs very well in estimating the own-price elasticities, even when compared to the nested logit.

| **Statistic** | **Estimated Elasticities (BLP)** | **True Elasticities** |
|---|---|---|
| Mean | -4.275 | -4.189 |
| Minimum | -10.217 | -10.008 |
| 1st Quartile | -4.859 | -4.764 |
| Median | -4.068 | -3.988 |
| 3rd Quartile | -3.474 | -3.395 |
| Maximum | -1.820 | -1.782 |

Table 7: Comparison of BLP Estimated and True Own-Price Elasticities

Table 8 presents the true diversion ratios and those estimated by the preferred BLP

specification. Similarly, results suggest that the estimates imply diversion ratios identical to the true ones.

| Statistic | Estimated Diversion Ratio (BLP) | True Diversion Ratio |
|---|---|---|
| Mean | 0.223 | 0.223 |
| Minimum | 0.001 | 0.001 |
| 1st Quartile | 0.107 | 0.107 |
| Median | 0.199 | 0.200 |
| 3rd Quartile | 0.317 | 0.317 |
| Maximum | 0.798 | 0.798 |

Table 8: Comparison of BLP Estimated and True Diversion Ratios

11. In the context of our merger simulation, we can narrow our focus to a specific aspect of merger effects, setting aside several other potential consequences. These include the internalization of inefficient pricing externalities, facilitation of collusion, alteration of innovation incentives, and enhanced bargaining power against upstream suppliers. Instead, I concentrate on the internalization of competitive externalities (unilateral effects) in a horizontal merger.

The unilateral effects of a merger arise when competition between the products of the merged firm is reduced because the merged firm internalizes substitution between its jointly-owned products (Conlon and Mortimer, 2020). As a result, I anticipate an increase in the prices of products from the merged firm ($j = 1, 2$). This change also affects the FOC of the other firms through the price elasticities. Consequently, prices for $j = 3, 4$ should also increase.

12. Table 9 presents the changes in equilibrium prices post-merger. I tried to compute the confidence intervals using a parametric bootstrap but it was taking a long time, code is provided in Appendix B.

| Statistic | Value |
|---|---|
| Mean | 0.122 |
| Std. Deviation | 0.163 |
| Minimum | 0.000 |
| 1st Quartile | 0.006 |
| Median | 0.030 |
| 3rd Quartile | 0.205 |
| Maximum | 0.978 |

Table 9: Summary of Price Changes After Merger of Firms 1 and 2

13. Table 10 compares the price changes resulting from the merger of firms 1 and 2 versus the merger of firms 1 and 3. We can see that the merger of firms 1 and 2 leads to larger price increases for the merged firms. This is consistent with what we would expect as firms 1 and 2 provide a more similar product (satellite television services) compared to firm 3. Consequently, the cross-price elasticities between the products of firms 1 and 2 are higher, which strengthens the unilateral effects of the merger.

| **Product** | **Firms 1 and 2** | **Firms 1 and 3** |
|---|---|---|
| Product 1 | 0.228 | 0.111 |
| Product 2 | 0.239 | 0.011 |
| Product 3 | 0.010 | 0.119 |
| Product 4 | 0.010 | 0.011 |

Table 10: Average Price Changes for Mergers

14. A merger-specific reduction in marginal cost can enhance welfare by improving overall efficiency in the market. When two companies merge, they may achieve economies of scale, streamline operations, or eliminate duplicate functions, leading to lower production costs. As a result, this increases welfare by increasing producer's profits and/or consumer surplus, if these cost savings are passed on to consumers in the form of lower prices.

15. Table 11 presents the predicted post-merger price changes for each product, averaged across markets. Assuming that the total measure of consumers is the same in each market, the predicted impact of the merger on consumer welfare is -$0.54. This additional assumption is needed in order to aggregate the consumer welfare, i.e. determine the weights in the consumer surplus calculation. With an identical measure of consumers, each market is given equal weight of 1.

To compute the predicted impact of the merger on total welfare, I estimate the effect on profits ($67.62). This leads to an overall increase in total welfare of $67.08.

| **Product** | **Firms 1 and 2** |
|---|---|
| Product 1 | 0.030 |
| Product 2 | 0.025 |
| Product 3 | -0.001 |
| Product 4 | -0.001 |

Table 11: Average Price Changes Accounting for Cost Reductions from the Merger

# Appendices

## A Code for "Generate Fake Data" and "Estimate Some Mis-specified Models" (Julia)

```
using Random
using Distributions
using LinearAlgebra
using NLsolve
using CSV
using GLM
using DelimitedFiles
using DataFrames
using Econometrics
using RegressionTables
using Plots
Random.seed!(1);

# set home directory
HOME_DIR = "C:/Users/guilh/Dropbox/ECON 600 - Industrial
   Organization I/First Part/Problem Sets/"

# problem parameters
N_draws = 500;
T = 600;
J = 4;

# functions
### function for derivatives and market shares
function derivatives_sh(p, x, beta_1, beta_2, beta_3, alpha,
   N_draws, T)

    # delta
    delta_jt = alpha .* p + beta_1 .* x[:, 3] + x[:, 6] #
       column 3: x_jt; column 6: xi_jt

    # loop over i
```

```
mu_ijt = zeros((N_draws, T*J))
s_ijt = zeros((N_draws, T, J))
D = zeros((J, J, T))
G = zeros((J, J, T))

for i in 1:N_draws
    mu_ijt[i, :, :] = beta_2[i] .* x[:, 4] + beta_3[i] .*
       x[:, 5]

    # reshape data (each line is a market, each column is
       a product)
    mu_ijt_reshaped = reshape(mu_ijt[i, :, :], (T, J))
    delta_jt_reshaped = reshape(delta_jt, (T, J))

    num_ijt = exp.(delta_jt_reshaped + mu_ijt_reshaped)
    denom_ijt = 1 .+ sum(num_ijt, dims=2)

    s_ijt[i, :, :] = num_ijt ./ repeat(denom_ijt, outer =
       [1, 4])

    # elasticities
    for t in 1:T

        cross_terms = s_ijt[i, t, :] * s_ijt[i, t, :]'

        # for the cross-price elasticities
        G[:, :, t] = G[:, :, t] .+ alpha .* cross_terms

        # for the own-price elasticities
        for j in 1:J
            D[j, j, t] = D[j, j, t]  .+ alpha .* s_ijt[i,
               t, j]
        end

    end
end

s_jt = dropdims(sum(s_ijt, dims=1), dims=1) ./ N_draws
D = D ./ N_draws
```

```
    G = G ./ N_draws

    derivatives = D - G

    return s_jt, derivatives, D, G
end

### function for marginal cost
function mc(w, gamma_0, gamma_1)
    marg_cost = exp.(gamma_0 .+ gamma_1 .* w[:, 3] + w[:, 4]
       ./ 8) # column 3: w_jt; column 4: omega_jt
end

### function for solving equilibrium prices
function foc(p, mg_cost, x, beta_1, beta_2, beta_3, alpha,
   N_draws, T)

    sh, deriv, D, G = derivatives_sh(p, x, beta_1, beta_2,
       beta_3, alpha, N_draws, T)

    # get only own-price derivatives
    own_deriv = zeros(T, J)
    for t in 1:T
        for j in 1:J
            own_deriv[t, j] = deriv[j, j, t]
        end
    end

    # FOC
    FOC = (p .- mg_cost) .* own_deriv' .+ sh'
    return FOC
end

# 1. Draw the exogenous product characteristics...
   ------------------------

# distributions
d = MvNormal([0,0], [1 0.25; 0.25 1]);
beta_23_dist = Normal(4, 1)
```

```
# non-random parameters
beta_1 = 1;
alpha = -2;
gamma_0 = 1/2;
gamma_1 = 1/4;

# draw betas
beta_2 = rand(beta_23_dist, N_draws)
beta_3 = rand(beta_23_dist, N_draws)

# generate xi_jt and omega_jt
x_gen = rand(d, T*J);
xi = x_gen[1, :];
om = x_gen[2, :];

# generate matrix of characteristics X_jt and xi_jt -- columns
   (1: Market ID; 2: Product ID; 3: x_jt; 4: satellite; 5:
   wired; 6: xi_jt)
X_1t = hcat(1:T, ["1" for i in 1:T], abs.(randn((T, 1))), ones
   (T), zeros(T), xi[1:T])       # satellite
X_2t = hcat(1:T, ["2" for i in 1:T], abs.(randn((T, 1))), ones
   (T), zeros(T), xi[T+1:2*T])   # satellite
X_3t = hcat(1:T, ["3" for i in 1:T], abs.(randn((T, 1))),
   zeros(T), ones(T), xi[2*T+1:3*T]) # wired
X_4t = hcat(1:T, ["4" for i in 1:T], abs.(randn((T, 1))),
   zeros(T), ones(T), xi[3*T+1:4*T]) # wired

# generate cost observables w_jt and unobservables omega_jt --
    columns(1: Market ID; 2: Product ID; 3: w_jt; 4: omega_jt)
W_1t = hcat(1:T, ["1" for i in 1:T], abs.(randn((T, 1))), om
   [1:T])
W_2t = hcat(1:T, ["2" for i in 1:T], abs.(randn((T, 1))), om[T
   +1:2*T])
W_3t = hcat(1:T, ["3" for i in 1:T], abs.(randn((T, 1))), om
   [2*T+1:3*T])
W_4t = hcat(1:T, ["4" for i in 1:T], abs.(randn((T, 1))), om
   [3*T+1:4*T])
```

```
# get marginal cost
mg_cost = reshape(mc([W_1t; W_2t; W_3t; W_4t], gamma_0,
   gamma_1), (T, J))


# 2. Solve for the equilibrium prices for each good in each
   market -------
### (i) Using "fsolve":
conv = 0
prices_solve = zeros(T, J)
for t in 1:T
    f!(p) = foc(p, mg_cost[t, :], [X_1t; X_2t; X_3t; X_4t][[t,
        t+T, t+2*T, t+3*T], :], beta_1, beta_2, beta_3, alpha,
       N_draws, 1)
    res = nlsolve(f!, 1.3 .* mg_cost[t, :], iterations = 1000)

    # check convergence
    conv = conv + res.f_converged

    # get prices (zero)
    prices_solve[t, :] = res.zero
end

conv == 600 # all converged

### (ii) Using the algorithm of Morrow and Skerlos (2011):
H = I(4)
p_init = reshape(mg_cost, T*J)
diff = 1
iter = 0

p_n = p_init
while diff > 1e-10
    sh_n, deriv_n, D_n, G_n = derivatives_sh(p_n, [X_1t; X_2t;
        X_3t; X_4t], beta_1, beta_2, beta_3, alpha, N_draws, T)

    # reshape prices (each line is a market, each column is a
       product)
    p_n = reshape(p_n, (T, J))
```

```
    # loop over markets
    Zt_n = zeros((T, J))
    for t in 1:T
        Zt_n[t, :] = inv(D_n[:, :, t]) * (H .* G_n[:,:,t]) * (
           p_n[t, :] - mg_cost[t, :]) - inv(D_n[:, :, t]) *
           sh_n[t, :]
    end

    # new guess
    p_n1 = mg_cost + Zt_n

    # update guess
    diff = maximum(abs.(p_n1 - p_n))
    p_n = reshape(p_n1, T*J)
    iter = iter + 1

end

prices_ms_2011 = reshape(p_n, (T, J))

# checking difference between both methods
maximum(abs.(prices_solve - prices_ms_2011)) # large maximum
   difference
mean(abs.(prices_solve - prices_ms_2011)) # not that high
   average difference

# choose Morrow and Skerlos (2011) as observed prices - Conlon
    and Gortmaker (2020)
p_obs = prices_ms_2011

# 3. Calculate "observed" market shares
   ----------------------------------
sh_obs, deriv_obs = derivatives_sh(p_n, [X_1t; X_2t; X_3t;
   X_4t], beta_1, beta_2, beta_3, alpha, N_draws, T)

# build fake dataset -- columns(1: Market ID; 2: Product ID;
   3: prices; 4: shares; 5: x_jt; 6: satellite; 7: wired; 8:
   w_jt)
```

```
G1 = hcat(X_1t[:, 1:2], p_obs[:, 1], sh_obs[:, 1], X_1t[:,
   3:5], W_1t[:, 3])
G2 = hcat(X_2t[:, 1:2], p_obs[:, 2], sh_obs[:, 2], X_2t[:,
   3:5], W_2t[:, 3])
G3 = hcat(X_3t[:, 1:2], p_obs[:, 3], sh_obs[:, 3], X_3t[:,
   3:5], W_3t[:, 3])
G4 = hcat(X_4t[:, 1:2], p_obs[:, 4], sh_obs[:, 4], X_4t[:,
   3:5], W_4t[:, 3])
data = [G1; G2; G3; G4]

# as DataFrame
df = DataFrame(data, :auto)
df = select(df, :x1 => :market_ID, :x2 => :product_ID, :x3 =>
   :prices, :x4 => :shares, :x5 => :x, :x6 => :satellite, :x7
   => :wired, :x8 => :w)

# export csv
CSV.write(joinpath(HOME_DIR, "data.csv"), df)


# 4. Check instruments in your fake data
   ---------------------------------
# read table
data, header = readdlm(joinpath(HOME_DIR, "data.csv"), ',',
   header=true)
df = DataFrame(data, vec(header))

# create instruments (using BLP approximation)
df_inst = transform(groupby(df, :market_ID), :x => ((v) -> sum
   (v) .- v) => :x_instrument, :w => ((v) -> sum(v) .- v) => :
   w_instrument)

# regressions of prices and market shares on the exogenous
   variables
reg_p = lm(@formula(prices ~ x + satellite + w + x_instrument
   + w_instrument), df_inst)
reg_s = lm(@formula(shares ~ x + satellite + w + x_instrument
   + w_instrument), df_inst)
regtable(reg_p, reg_s; renderSettings = latexOutput(joinpath(
```

```
   HOME_DIR, "tables/reg.tex")))


# 5. Estimate the plain multinomial logit model of demand by
   OLS ---------
df_logit = transform(groupby(df_inst, :market_ID), :shares =>
   ((v) -> 1 - sum(v)) => :s0)
df_logit = transform(df_logit, [:shares, :s0] => ((sj, s0) ->
   log.(sj) - log.(s0)) => :delta)

logit = lm(@formula(delta ~ prices + x + satellite), df_logit)
regtable(logit; renderSettings = latexOutput(joinpath(HOME_DIR
   , "tables/logit.tex")))


# 6. Re-estimate the multinomial logit model of demand by 2SLS
    -----------
iv_logit = fit(EconometricModel,
               @formula(delta ~ x + satellite + (prices ~ x +
                  satellite + w + x_instrument + w_instrument))
                  ,
               df_logit)

regtable(iv_logit; renderSettings = latexOutput(joinpath(
   HOME_DIR, "tables/iv_logit.tex")))

first_stage = lm(@formula(prices ~ x + satellite + w +
   x_instrument + w_instrument), df_logit)
r2(first_stage)


# 7. Now estimate a nested logit model by linear IV
   ----------------------
### adding group share
df_nestedl = transform(groupby(df_logit, [:market_ID, :
   satellite]), :shares => ((v) -> v/sum(v)) => :s_jg)
df_nestedl = transform(groupby(df_nestedl, [:market_ID, :
   satellite]), :x => ((v) -> sum(v) .- v) => :x_g_instrument,
   :w => ((v) -> sum(v) .- v) => :w_g_instrument)
```

```
nested_logit = fit(EconometricModel,
                   @formula(delta ~ x + satellite + (prices +
                      log(s_jg) + satellite&log(s_jg) ~ x + w +
                       x_instrument + w_instrument +
                      x_g_instrument + w_g_instrument)),
                   df_nestedl)

regtable(nested_logit; renderSettings = latexOutput(joinpath(
   HOME_DIR, "tables/nested_logit.tex")))

# 8. Compare the estimated own-price elasticities to the true
   ones -------
sigma_wired = coef(nested_logit)[5]
sigma_satellite = coef(nested_logit)[5] + coef(nested_logit)
   [6]

df_nestedl.sigma = @. ifelse(df_nestedl.satellite == 1,
   sigma_satellite, sigma_wired)
df_elast = transform(df_nestedl, [:shares, :s_jg, :sigma] =>
   ((sj, s_jg, sigma) -> alpha ./ (1 .- sigma) .* sj .* (1 .-
   sigma .* s_jg .- (1 .- sigma) .* sj)) => :own_elast)

# derivatives matrix
price_deriv = zeros((J, J, T))
for t in 1:T
    for j in 1:J
        for k in 1:J
            sj = df_elast[(df_elast.market_ID .== t), :shares
               ][j]
            sj_g = df_elast[(df_elast.market_ID .== t), :s_jg
               ][j]
            sk = df_elast[(df_elast.market_ID .== t), :shares
               ][k]
            sigma = df_elast[(df_elast.market_ID .== t), :
               sigma][j]

            if j == k # own-price
                price_deriv[j, j, t] = df_elast[(df_elast.
```

```
                    market_ID .== t), :own_elast][j]

            elseif df_elast[(df_elast.market_ID .== t), :wired
               ][j] == df_elast[(df_elast.market_ID .== t), :
               wired][k] # same group
                price_deriv[j, k, t] = -alpha .* sk .* (sj .+
                   sj_g .* sigma ./ (1 .- sigma))

            else # different groups
                price_deriv[j, k, t] = -alpha .* sj .* sk
            end

        end
    end
end

# derivatives to elasticities
true_elasts = zeros((J, J, T))
price_elast = zeros((J, J, T))
for t in 1:T
    for j in 1:J
        for k in 1:J
            sj = df_elast[(df_elast.market_ID .== t), :shares
               ][j]
            pk = df_elast[(df_elast.market_ID .== t), :prices
               ][k]

            price_elast[j, k, t] = pk/sj * price_deriv[j, k, t
               ]
            true_elasts[j, k, t] = pk/sj * deriv_obs[j, k, t]
        end
    end
end

est_own_price_elast = [price_elast[j, j, t] for j in 1:J for t
    in 1:T]
true_own_price_elast = [true_elasts[j, j, t] for j in 1:J for
   t in 1:T]
```

```julia
# compare
describe(est_own_price_elast)
describe(true_own_price_elast)
describe(abs.(est_own_price_elast .- true_own_price_elast))

# plot
p = scatter(est_own_price_elast, true_own_price_elast,
    xlabel="Estimated Own-Price Elasticity",
    ylabel="True Own-Price Elasticity",
    legend=false,
    marker=:circle,
    ms=2,           # marker size
    grid=true,      # add grid
    title="Comparison of True and Estimated Own-Price
       Elasticities",
    aspect_ratio=1,
    alpha=0.5
)

plot!(-11:-2, -11:-2,
    label="y = x",
    lw=2,         # line width
    color=:red
)

savefig(p, joinpath(HOME_DIR, "tables/plot_elasticities.png"))

# diversion ratios
true_diversion = zeros((J, J, T))
estd_diversion = zeros((J, J, T))
for t in 1:T
    for j in 1:J
        for k in 1:J
            estd_diversion[j, k, t] = -price_deriv[j, k, t] /
               price_deriv[j, j, t]
            true_diversion[j, k, t] = -deriv_obs[j, k, t] /
               deriv_obs[j, j, t]
        end
    end
```

```
end

# compare
describe(estd_diversion[estd_diversion .!= 1])
describe(true_diversion[true_diversion .!= 1])
describe(abs.(estd_diversion[estd_diversion .!= 1] .-
   true_diversion[true_diversion .!= 1]))
```

# B Code for "Estimate the Correctly Specified Model" and "Merger Simulation" (Python)

```
# setup
import pyblp
import numpy as np
import pandas as pd
from pathlib import Path

pyblp.options.digits = 2
pyblp.options.verbose = False

# set home directory
HOME_DIR = "C:/Users/guilh/Dropbox/ECON 600 - Industrial
   Organization I/First Part/Problem Sets/"

# read data
df = pd.read_csv(Path(HOME_DIR, 'data.csv'))
df.rename(columns={'market_ID': 'market_ids',
                   'product_ID': 'product_ids'}, inplace=True)

df_demandonly = df.copy()
df_demandonly.rename(columns={'w': 'demand_instruments0'},
   inplace=True)

# 9.
   ----------------------------------------------------------------------

# define X1 and X2
X1_formulation = pyblp.Formulation('1 + prices + x + satellite
   ')
X2_formulation = pyblp.Formulation('0 + satellite + wired')
product_formulations = (X1_formulation, X2_formulation)

# integration and optimization
mc_integration = pyblp.Integration('monte_carlo', size=500,
   specification_options={'seed': 0})
```

```
# estimate model with defaults (demand only)
prb = pyblp.Problem(product_formulations, df_demandonly,
   integration=mc_integration)
results_noopt = prb.solve(sigma = 0.5*np.eye((2)))

# optimal instruments
opt_instr = results_noopt.compute_optimal_instruments()
prb_opt = opt_instr.to_problem()
results_opt = prb_opt.solve(sigma = 0.5*np.eye((2)))


# with supply
df['firm_ids'] = df.product_ids
product_formulations_supply = (
   X1_formulation, X2_formulation,
   pyblp.Formulation('1 + w')
)

prb_supply = pyblp.Problem(product_formulations_supply, df,
   integration=mc_integration, costs_type='log')
results_noopt_supply = prb_supply.solve(sigma = 0.5*np.eye((2)
   ), beta = [2, -1, 0.5, 0.05])

# optimal instruments
optimization_tr = pyblp.Optimization('trust-constr')

opt_instr_supply = results_noopt_supply.
   compute_optimal_instruments()
prb_opt_supply = opt_instr_supply.to_problem(
   demand_shifter_formulation = pyblp.Formulation('x'))
results_opt_supply = prb_opt_supply.solve(sigma = 0.5*np.eye
   ((2)), beta = [2, -1, 0.5, 0.05],
                                          optimization =
                                             optimization_tr)


# export model results
results_opt.to_pickle(Path(HOME_DIR, 'models/demand_only_blp.
   pkl'))
```

```
results_opt_supply.to_pickle(Path(HOME_DIR, 'models/full_blp.
   pkl'))
#results_opt_supply = pyblp.read_pickle(Path(HOME_DIR, 'models
   /full_blp.pkl'))


# 10.
   ------------------------------------------------------------------

# compute own-price elasticities
own_price_elasts = results_opt_supply.extract_diagonals(
   results_opt_supply.compute_elasticities())
pd.DataFrame(own_price_elasts).describe()

# compute diversion ratios
all_diversion_ratios = np.zeros((4, 4, 600))
for t in range(600):
   all_diversion_ratios[:, :, t] = results_opt_supply.
      compute_diversion_ratios(market_id = (t+1))


# extract only non-outside good diversion ratios (compare with
    previous results)
diversion_ratios = np.zeros((4, 4, 600))
for j in range(4):
   for k in range(4):
      for t in range(600):
         if j != k:
            diversion_ratios[j, k, t] = all_diversion_ratios[j
               , k, t]

diversion_ratios = diversion_ratios[diversion_ratios != 0]
pd.DataFrame(diversion_ratios).describe()


# 12.
   ------------------------------------------------------------------

# compute costs
```

```
costs = results_opt_supply.compute_costs()

# simulate merger 1-2
merger_df_12 = df.copy()
merger_df_12['merger_ids'] = merger_df_12['firm_ids'].replace
   (2, 1)

# post-merger equilibrium prices
changed_prices12 = results_opt_supply.compute_prices(
    firm_ids = merger_df_12['merger_ids'],
    costs = costs
)

changes_12 = changed_prices12 - merger_df_12[["prices"]]

# bootstrap (was taking too long)
#bootstrapped_results = results_opt_supply.bootstrap(draws
   =100, seed=0)
#
#bounds = np.percentile(
#    bootstrapped_results.compute_prices(
#      firm_ids = merger_df_12['merger_ids'],
#      costs = bootstrapped_results.compute_costs()
#   ) - merger_df_12[["prices"]],
#    q=[10, 90],
#    axis=0
#)

# 13.
   ----------------------------------------------------------------------

# simulate merger 1-3
merger_df_13 = df.copy()
merger_df_13['merger_ids'] = merger_df_13['firm_ids'].replace
   (3, 1)

# post-merger equilibrium prices
changed_prices13 = results_opt_supply.compute_prices(
    firm_ids = merger_df_13['merger_ids'],
```

```
    costs = costs
)

changes_13 = changed_prices13 - merger_df_13[["prices"]]

# compare average across markets from 12 and 13
changes_12_reshaped = changes_12.values.reshape(600, 4, order=
   'F')
changes_13_reshaped = changes_13.values.reshape(600, 4, order=
   'F')

changes_12_averages = pd.DataFrame(changes_12_reshaped,
   columns=['Product1', 'Product2', 'Product3', 'Product4']).
   mean(axis=0)
changes_13_averages = pd.DataFrame(changes_13_reshaped,
   columns=['Product1', 'Product2', 'Product3', 'Product4']).
   mean(axis=0)


# 15.
   ----------------------------------------------------------------------

merger_costs = costs.copy()
merger_costs[merger_df_12.merger_ids == 1] = 0.85*merger_costs
   [merger_df_12.merger_ids == 1]

# post-merger equilibrium prices
changed_prices = results_opt_supply.compute_prices(
    firm_ids = merger_df_12['merger_ids'],
    costs = merger_costs
)

changes_12 = changed_prices - merger_df_12[["prices"]]

# compare average across markets
changes_12_reshaped = changes_12.values.reshape(600, 4, order=
   'F')
changes_12_averages = pd.DataFrame(changes_12_reshaped,
   columns=['Product1', 'Product2', 'Product3', 'Product4']).
```

```
   mean(axis=0)

# changes in shares
changed_shares = results_opt_supply.compute_shares(
   changed_prices)

# welfare
cs_m = results_opt_supply.compute_consumer_surpluses(prices=
   changed_prices)
cs_p = results_opt_supply.compute_consumer_surpluses(prices=df
   [["prices"]])
cs_change = np.sum(cs_m - cs_p)

profits = results_opt_supply.compute_profits(changed_prices,
   changed_shares, merger_costs)
profits_p = results_opt_supply.compute_profits(prices=df[["
   prices"]])
profits_change = np.sum(profits - profits_p)
np.sum(cs_change) + np.sum(profits_change)
```