Anton Melnychuk ECON 3385 - Problem Set 4

February 10th, 2026

In [5]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
import statsmodels.stats.api as sms

hausman_data = pd.read_csv("hausman_data.csv")
hausman_data.head()
```

Out[5]:

	city	time	prod	price	qty
0	1	1	1	6.028006	238.995267
1	1	1	2	8.232000	646.243673
2	1	1	3	6.974984	767.854482
3	1	1	4	8.879458	559.638749
4	1	1	5	7.007685	708.385041

Question 1

Transform the data from long format to wide format, creating separate columns for each product's price and quantity.

In [6]:

```
hausman_pivot = hausman_data.pivot(
    index=["city", "time"],
    columns="prod",
    values=["price", "qty"]
)

# Flatten the multi-level column
hausman_pivot.columns = [
    f'{val}{col}' for val, col in hausman_pivot.columns
]

hausman_pivot = hausman_pivot.reset_index()
hausman_pivot.head()
```

Out[6]:

	city	time	price1	price2	price3	price4	price5	qty1	qty2	qty3
0	1	1	6.028006	8.232000	6.974984	8.879458	7.007685	238.995267	646.243673	767.854482
1	1	2	3.589721	3.884272	3.361197	5.561177	5.595291	490.060439	365.277518	397.480928

	city	time	price1	price2	price3	price4	price5	qty1	qty2	qty3
2	1	3	2.891599	4.292659	4.164220	4.983339	7.509970	378.983007	455.782289	532.482876
3	1	4	4.623883	6.725099	6.297546	6.153429	11.684874	473.928773	712.214672	716.641624
4	1	5	4.115155	5.619486	4.986857	6.594416	6.705489	302.054122	584.416389	345.371783

Question 2

Create Hausman instruments for each product. The Hausman instrument for product in city at time is the average price of product in all other cities (excluding city) at time .

In [7]:

```
# want average price of each product in other cities at same time

products = [1, 2, 3, 4, 5]
n_cities = 20

for prod in products:
    own_price = f'price{prod}'
    instrument_name = f'otherprice{prod}'

    # total sum minus own city price, then divide by number of cities
    total_by_time = hausman_pivot.groupby('time')[own_price].transform('sum')
    hausman_pivot[instrument_name] = (total_by_time - hausman_pivot[own_price]) / n_citi

hausman_pivot.head()
```

Out[7]:

	city	time	price1	price2	price3	price4	price5	qty1	qty2	qty3
0	1	1	6.028006	8.232000	6.974984	8.879458	7.007685	238.995267	646.243673	767.854482
1	1	2	3.589721	3.884272	3.361197	5.561177	5.595291	490.060439	365.277518	397.480928
2	1	3	2.891599	4.292659	4.164220	4.983339	7.509970	378.983007	455.782289	532.482876
3	1	4	4.623883	6.725099	6.297546	6.153429	11.684874	473.928773	712.214672	716.641624
4	1	5	4.115155	5.619486	4.986857	6.594416	6.705489	302.054122	584.416389	345.371783

In [8]:

```
hausman_pivot[["city"]].nunique()
```

Out[8]:

```
city    20
dtype: int64
```

Question 3

a) Linear Demand Equation

- = how much of product people want to buy in city at time
- = baseline demand for product in city (some cities just like certain products more)
- = how sensitive demand is to prices

- = price of product in city at time
- = random demand shocks (things we can't observe that affect demand)

b) Marginal Cost Specification

When firms maximize profits, they set prices where marginal revenue equals marginal cost:

For linear demand , the derivative with respect to own price is:

Since , raising price lowers demand.

Substitute into the marginal cost equation:

Marginal cost = price + markup

(c) Assumptions for Hausman Instruments Validity

The Hausman instrument for product in city at time is:

Where is the total number of cities.

Relevance Condition: The instrument must be correlated with the endogenous variable (own price). This holds because:

- Prices across cities are correlated due to common cost shocks, market conditions, or competitive forces

Exclusion Restriction: The instrument must affect quantity demanded only through its effect on own price, not directly. This requires:

- demand shocks in city do not affect prices in other cities
- city-specific demand shocks () are uncorrelated across cities
- price differences across cities reflect supply-side factors rather than demand-side factors

Exogeneity: The instrument is uncorrelated with the error term .

Question 4

Estimate 5 demand equations using Two-Stage Least Squares (2SLS).

For each product, we regress quantity on all prices, using Hausman instruments.

In [10]:

```
from statsmodels.sandbox.regression.gmm import IV2SLS

price_vars = ['price1', 'price2', 'price3', 'price4', 'price5']
instrument_vars = ['otherprice1', 'otherprice2', 'otherprice3', 'otherprice4', 'otherpri

demand_models = {}

# estimate demand for each product
```

```python
for prod in range(1, 6):
    y = hausman_pivot[f'qty{prod}']
    # endogenous regressors
    X_endog = sm.add_constant(hausman_pivot[price_vars])
    # Hausman instruments for all prices
    Z = sm.add_constant(hausman_pivot[instrument_vars])

    model = IV2SLS(y, X_endog, Z).fit()
    demand_models[f'product_{prod}'] = model
```

In [14]:

```python
print(demand_models[f'product_{1}'].summary())
```

```
                          IV2SLS Regression Results
==============================================================================
Dep. Variable:                   qty1   R-squared:                      -0.840
Model:                         IV2SLS   Adj. R-squared:                 -0.855
Method:                     Two Stage   F-statistic:                     27.44
                        Least Squares   Prob (F-statistic):           5.10e-25
Date:                Tue, 10 Feb 2026
Time:                        01:40:10
No. Observations:                 600
Df Residuals:                     594
Df Model:                           5
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        686.8071    184.419      3.724      0.000     324.614    1049.000
price1      -155.7746     13.995    -11.131      0.000    -183.260    -128.290
price2        20.0050     19.982      1.001      0.317     -19.240      59.250
price3        16.4413     10.210      1.610      0.108      -3.611      36.493
price4         1.6614     19.567      0.085      0.932     -36.768      40.091
price5        21.5251     12.789      1.683      0.093      -3.592      46.642
==============================================================================
Omnibus:                        0.896   Durbin-Watson:                   2.099
Prob(Omnibus):                  0.639   Jarque-Bera (JB):                0.912
Skew:                           0.094   Prob(JB):                        0.634
Kurtosis:                       2.961   Cond. No.                         89.0
==============================================================================
```

In [15]:

```python
print(demand_models[f'product_{2}'].summary())
```

```
                          IV2SLS Regression Results
==============================================================================
Dep. Variable:                   qty2   R-squared:                      -1.643
Model:                         IV2SLS   Adj. R-squared:                 -1.665
Method:                     Two Stage   F-statistic:                     5.425
                        Least Squares   Prob (F-statistic):           6.85e-05
Date:                Tue, 10 Feb 2026
Time:                        01:40:17
No. Observations:                 600
Df Residuals:                     594
Df Model:                           5
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        791.0331    184.426      4.289      0.000     428.827    1153.239
price1        14.8583     13.995      1.062      0.289     -12.628      42.344
price2       -99.7369     19.983     -4.991      0.000    -138.983     -60.491
```

```
price3        17.3388     10.210      1.698      0.090      -2.714      37.391
price4        20.1118     19.568      1.028      0.304     -18.319      58.542
price5        -3.7507     12.789     -0.293      0.769     -28.869      21.367
==============================================================================
Omnibus:                        2.003   Durbin-Watson:                   2.047
Prob(Omnibus):                  0.367   Jarque-Bera (JB):                1.780
Skew:                          -0.003   Prob(JB):                        0.411
Kurtosis:                       2.733   Cond. No.                         89.0
==============================================================================
```

In [16]:

```
print(demand_models[f'product_{3}'].summary())
```

```
                          IV2SLS Regression Results
==============================================================================
Dep. Variable:                   qty3   R-squared:                      -0.600
Model:                         IV2SLS   Adj. R-squared:                 -0.614
Method:                     Two Stage   F-statistic:                     37.08
                        Least Squares   Prob (F-statistic):           4.20e-33
Date:                Tue, 10 Feb 2026
Time:                        01:40:22
No. Observations:                 600
Df Residuals:                     594
Df Model:                           5
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        874.2595    183.880      4.755      0.000     513.125    1235.394
price1         5.0519     13.954      0.362      0.717     -22.353      32.457
price2        52.3294     19.924      2.626      0.009      13.199      91.459
price3      -132.0943     10.180    -12.976      0.000    -152.088    -112.101
price4       -12.3873     19.510     -0.635      0.526     -50.704      25.929
price5        16.7322     12.752      1.312      0.190      -8.311      41.776
==============================================================================
Omnibus:                        0.173   Durbin-Watson:                   1.888
Prob(Omnibus):                  0.917   Jarque-Bera (JB):                0.114
Skew:                           0.032   Prob(JB):                        0.944
Kurtosis:                       3.024   Cond. No.                         89.0
==============================================================================
```

In [17]:

```
print(demand_models[f'product_{4}'].summary())
```

```
                          IV2SLS Regression Results
==============================================================================
Dep. Variable:                   qty4   R-squared:                      -0.961
Model:                         IV2SLS   Adj. R-squared:                 -0.978
Method:                     Two Stage   F-statistic:                     10.08
                        Least Squares   Prob (F-statistic):           2.80e-09
Date:                Tue, 10 Feb 2026
Time:                        01:40:27
No. Observations:                 600
Df Residuals:                     594
Df Model:                           5
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        862.0068    168.223      5.124      0.000     531.622    1192.392
price1         8.6417     12.766      0.677      0.499     -16.430      33.713
price2        66.5466     18.228      3.651      0.000      30.748     102.345
```

```
price3        11.9620      9.313      1.284      0.200     -6.329     30.253
price4      -115.8626     17.849     -6.491      0.000   -150.917    -80.808
price5        11.1725     11.666      0.958      0.339    -11.739     34.084
==============================================================================
Omnibus:                        0.456   Durbin-Watson:                   1.917
Prob(Omnibus):                  0.796   Jarque-Bera (JB):                0.314
Skew:                           0.033   Prob(JB):                        0.855
Kurtosis:                       3.091   Cond. No.                         89.0
==============================================================================
```

In [18]:

```
print(demand_models[f'product_{5}'].summary())
```

```
                          IV2SLS Regression Results
==============================================================================
Dep. Variable:                   qty5   R-squared:                      -0.891
Model:                         IV2SLS   Adj. R-squared:                 -0.907
Method:                     Two Stage   F-statistic:                     20.17
                        Least Squares   Prob (F-statistic):           1.30e-18
Date:                Tue, 10 Feb 2026
Time:                        01:40:31
No. Observations:                 600
Df Residuals:                     594
Df Model:                           5
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        963.2208    173.152      5.563      0.000    623.156    1303.286
price1        26.8079     13.140      2.040      0.042      1.002     52.614
price2        -4.2221     18.762     -0.225      0.822    -41.069     32.625
price3        38.0241      9.586      3.967      0.000     19.197     56.851
price4        22.1720     18.372      1.207      0.228    -13.909     58.253
price5       -96.8683     12.008     -8.067      0.000   -120.451    -73.286
==============================================================================
Omnibus:                        0.616   Durbin-Watson:                   1.959
Prob(Omnibus):                  0.735   Jarque-Bera (JB):                0.467
Skew:                          -0.049   Prob(JB):                        0.792
Kurtosis:                       3.094   Cond. No.                         89.0
==============================================================================
```

Question 5

Calculate the elasticity matrix. The elasticity of quantity with respect to price is:

In [22]:

```
# calculate average prices and quantities
price_cols = ['price1', 'price2', 'price3', 'price4', 'price5']
qty_cols = ['qty1', 'qty2', 'qty3', 'qty4', 'qty5']

P_bar = np.array([hausman_pivot[col].mean() for col in price_cols])
Q_bar = np.array([hausman_pivot[col].mean() for col in qty_cols])
```

In [24]:

```
# extract coefficients from demand models
B = np.zeros((5, 5))

for i in range(5):
```

```
    model = demand_models[f'product_{i+1}']
    for j, price_var in enumerate(price_vars):
        if price_var in model.params.index:
            B[i, j] = model.params[price_var]
        else:
            B[i, j] = 0

print(B.round(4))
```

```
[[-155.7746   20.005    16.4413    1.6614   21.5251]
 [  14.8583  -99.7369   17.3388   20.1118   -3.7507]
 [   5.0519   52.3294 -132.0943  -12.3873   16.7322]
 [   8.6417   66.5466   11.962  -115.8626   11.1725]
 [  26.8079   -4.2221   38.0241   22.172   -96.8683]]
```

In [26]:

```
# elasticity matrix
# ε_ij = β_ij × (P_j / Q_i)
elasticity_matrix = np.zeros((5, 5))

for i in range(5):  # quantity (row)
    Q_i_mean = Q_bar[i]
    for j in range(5):  # price (column)
        P_j_mean = P_bar[j]
        beta_ij = B[i, j]

        elasticity_matrix[i, j] = beta_ij * (P_j_mean / Q_i_mean)

elasticity_df = pd.DataFrame(
    elasticity_matrix,
    index=[f'Q{i+1}' for i in range(5)],
    columns=[f'P{j+1}' for j in range(5)]
)

print("Elasticity Matrix:")
print(elasticity_df.round(4))
```

```
Elasticity Matrix:
        P1      P2      P3      P4      P5
Q1 -1.5765  0.2731  0.2143  0.0255  0.3970
Q2  0.1281 -1.1600  0.1925  0.2628 -0.0589
Q3  0.0398  0.5563 -1.3408 -0.1479  0.2403
Q4  0.0523  0.5430  0.0932 -1.0620  0.1232
Q5  0.1696 -0.0360  0.3097  0.2125 -1.1166
```

Question 6

Positive values (off-diagonal entries) here mean that products are substitutes: if product 's price rises (column), demand for product increases (row). The bigger the positive number, the stronger the substitution effect.

The strongest substitute relationships:

- Q3 with respect to P2: 0.5563 — Product 3 is a substitute for Product 2
- Q4 with respect to P2: 0.5430 — Product 4 is a substitute for Product 2
- Q1 with respect to P5: 0.3970 — Product 1 is a substitute for Product 5
- Q5 with respect to P3: 0.3097 — Product 5 is a substitute for Product 3

- Q1 with respect to P2: 0.2731 — Product 1 is a substitute for Product 2
- Q2 with respect to P4: 0.2628 — Product 2 is a substitute for Product 4
- Q3 with respect to P5: 0.2403 — Product 3 is a substitute for Product 5
- Q1 with respect to P3: 0.2143 — Product 1 is a substitute for Product 3
- Q5 with respect to P4: 0.2125 — Product 5 is a substitute for Product 4

Question 7

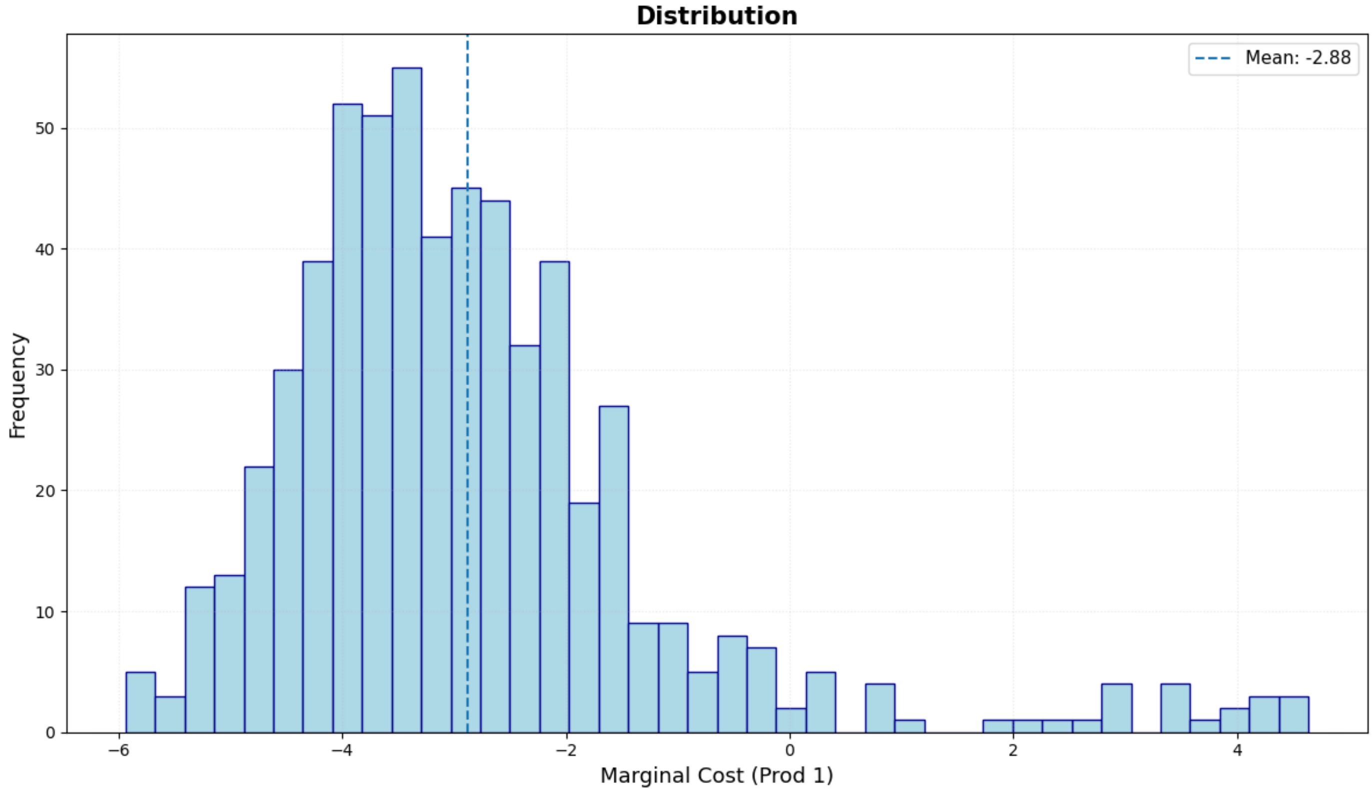
Calculate marginal costs for each product using the formula derived in Question 3b:

In [37]:

```
B_inv = np.linalg.inv(B)

prices = hausman_pivot[['price1', 'price2', 'price3', 'price4', 'price5']].values
quantities = hausman_pivot[['qty1', 'qty2', 'qty3', 'qty4', 'qty5']].values

# MC = P + Q @ B_inv.T
marginal_costs = prices + (quantities @ B_inv.T)
hausman_pivot['MC1'] = marginal_costs[:, 0]
```

In [50]:

```
plt.figure(figsize=(12, 7))
plt.hist(hausman_pivot['MC1'], bins=40, edgecolor='darkblue',color='lightblue')
plt.xlabel('Marginal Cost (Prod 1)', fontsize=13)
plt.ylabel('Frequency', fontsize=13)
plt.title('Distribution', fontsize=15, fontweight='bold')
plt.axvline(hausman_pivot['MC1'].mean(), linestyle='--',
            label=f'Mean: {hausman_pivot["MC1"].mean():.2f}')
plt.legend(fontsize=11)
plt.grid(True, alpha=0.25, linestyle=':', linewidth=0.8)
plt.tight_layout()
plt.show()

print("\nMarginal Cost:")
print(f"Mean: {hausman_pivot['MC1'].mean():.4f}")
print(f"Std Dev: {hausman_pivot['MC1'].std():.4f}")
print(f"Min: {hausman_pivot['MC1'].min():.4f}")
print(f"Max: {hausman_pivot['MC1'].max():.4f}")
```




```
Marginal Cost:
Mean: -2.8778
Std Dev: 1.7270
Min: -5.9379
Max: 4.6408
```