

ps4_solutions

February 16, 2026

1 Anton Melnychuk ECON 3385 - Problem Set 5

February 16th, 2026

1.0.1 Question 1

```
[123]: import pandas as pd

data = pd.read_csv('airlines_long.csv')
airline_map = {1: 'AA', 2: 'DL', 3: 'Other', 4: 'UA'}
data['airline'] = data['airline_id'].map(airline_map)

data_wide = data.pivot(
    index=['route_city', 'quarter', 'avg_pop'],
    columns='airline',
    values=['price', 'logp', 'passengers', 'logq', 'avg_hub']
).reset_index()

data_wide.columns = [f'{val}_{airline}' if airline else val
                     for val, airline in data_wide.columns]
data_wide = data_wide.dropna()
print(len(data_wide))
data_wide.head()
```

167

```
[123]:
```

	route_city	quarter	avg_pop	price_AA	price_DL	price_Other	price_UA	\
0	ATL to DEN	2	634.0	257.0	44.0	139.25	150.0	
1	ATL to DEN	3	634.0	172.0	38.0	139.75	119.0	
2	ATL to DEN	4	634.0	100.0	58.0	78.00	93.0	
3	ATL to ORD	1	1640.5	106.0	70.0	52.00	96.0	
4	ATL to ORD	2	1640.5	137.0	47.0	30.00	149.0	

	logp_AA	logp_DL	logp_Other	...	passengers_Other	passengers_UA	\
0	5.549076	3.784190	4.936271	...	7450.0	1680.0	
1	5.147494	3.637586	4.939855	...	7440.0	1710.0	
2	4.605170	4.060443	4.356709	...	7230.0	3780.0	
3	4.663439	4.248495	3.951244	...	14610.0	42900.0	

```

4  4.919981  3.850147  3.401197  ...          13980.0      12260.0

      logq_AA      logq_DL  logq_Other      logq_UA  avg_hub_AA  avg_hub_DL  \
0  2.995732  9.836813  8.915969  7.426549      25.5      41.0
1  4.382027 10.057500  8.914626  7.444249      28.5      42.0
2  5.521461  9.877144  8.885994  8.237479      28.0      37.5
3  9.801455 11.233210  9.589461 10.666630      41.0      41.0
4  9.382611 11.297750  9.545383  9.414097      31.0      41.0

      avg_hub_Other  avg_hub_UA
0          16.625      36.0
1          18.000      35.0
2          18.000      33.5
3          17.000      38.5
4          14.000      35.5

```

[5 rows x 23 columns]

1.0.2 Question 2

Why is avg_pop in the demand equation?

avg_pop captures market size - larger population markets have higher demand. It controls for market size differences across routes. Since population is exogenous to airline pricing decisions, including avg_pop helps identify demand and isolates the price effects captured by the _jk coefficients.

What do the _jk coefficients mean under the log-log specification?

_jk = % change in Q_j / % change in p_k (cross-price elasticity if j ≠ k, own-price elasticity if j=k => expect to be negative).

1.0.3 Question 3

Pricing equations (Nash-Bertrand):

For each airline j, FOC from profit maximization:

$$p_{jct} = c_{jct} - \frac{Q_{jct}}{\frac{\partial Q_{jct}}{\partial p_{jct}}}$$

Under log-log demand: $\frac{\partial Q_{jct}}{\partial p_{jct}} = \frac{Q_{jct} \beta_{jj}}{p_{jct}}$

So:

$$p_{jct} = c_{jct} - \frac{p_{jct}}{\beta_{jj}}$$

Rearranging:

$$p_{jct} = c_{jct} \cdot \frac{\beta_{jj}}{\beta_{jj} + 1}$$

8 Valid Instruments for 4 prices:

For each airline j 's price p_{jct} , two types of instruments:

1. **Hausman instruments:** $\bar{p}_{(-j)ct}$ = average price of other airlines (excluding j) in the same route c at time t (4 instruments - one per airline)
2. **Same airline's prices in other routes:** $\bar{p}_{j(-c)t}$ = average price of airline j in all other routes at time t (4 instruments - one per airline)

Validity: Hausman instruments reflect supply conditions of competitors but are uncorrelated with airline j 's demand shock under the covariance assumptions. Prices in other routes share common cost factors across markets but are independent of the current route's demand shock.

1.0.4 Question 4

```
[124]: from linearmodels.iv import IV2SLS
import statsmodels.api as sm

airlines = ['AA', 'DL', 'Other', 'UA']

# other airlines' prices in same route
data_wide['other_price_AA'] = data_wide[['price_DL', 'price_Other',
↪ 'price_UA']].mean(axis=1)
data_wide['other_price_DL'] = data_wide[['price_AA', 'price_Other',
↪ 'price_UA']].mean(axis=1)
data_wide['other_price_UA'] = data_wide[['price_AA', 'price_DL',
↪ 'price_Other']].mean(axis=1)
data_wide['other_price_Other'] = data_wide[['price_AA', 'price_DL',
↪ 'price_UA']].mean(axis=1)

# same airline's prices in other routes
for airline in airlines:
    quarter_means = data_wide.groupby('quarter')[f'price_{airline}'].mean()
    quarter_counts = data_wide.groupby('quarter')[f'price_{airline}'].count()

    data_wide[f'other_route_price_{airline}'] = data_wide.apply(
        lambda row: (quarter_means[row['quarter']] *
↪ quarter_counts[row['quarter']] - row[f'price_{airline}']) /
        (quarter_counts[row['quarter']] - 1),
        axis=1
    )

# all 8 instruments
instruments = data_wide[[
    'other_price_AA', 'other_route_price_AA',
    'other_price_DL', 'other_route_price_DL',
    'other_price_Other', 'other_route_price_Other',
    'other_price_UA', 'other_route_price_UA'
]]
```

```

results_2sls = {}
for airline in airlines:
    y = data_wide[f'logq_{airline}']
    exog = sm.add_constant(data_wide[['avg_pop']])
    endog = data_wide[[f'logp_{a}' for a in airlines]]

    model = IV2SLS(y, exog, endog, instruments).fit()
    results_2sls[airline] = model

```

```

[125]: print(f"AA Demand (2SLS):")
       print(results_2sls['AA'].summary)

```

AA Demand (2SLS):

IV-2SLS Estimation Summary

```

=====
Dep. Variable:          logq_AA    R-squared:                0.2409
Estimator:              IV-2SLS    Adj. R-squared:           0.2173
No. Observations:       167        F-statistic:              65.886
Date:                   Mon, Feb 16 2026    P-value (F-stat)          0.0000
Time:                   21:18:31           Distribution:             chi2(5)
Cov. Estimator:         robust

```

Parameter Estimates

```

=====
               Parameter  Std. Err.    T-stat    P-value    Lower CI    Upper CI
-----
const           17.815      2.4698     7.2133    0.0000     12.975     22.656
avg_pop          0.0006    9.322e-05    6.5507    0.0000      0.0004     0.0008
logp_AA         -2.8405      0.5599    -5.0731    0.0000     -3.9379    -1.7431
logp_DL          0.6045      0.2263     2.6712    0.0076      0.1610     1.0480
logp_Other       0.4386      0.2802     1.5651    0.1176     -0.1107     0.9878
logp_UA         -0.2852      0.3028    -0.9419    0.3462     -0.8786     0.3082
=====

```

Endogenous: logp_AA, logp_DL, logp_Other, logp_UA
Instruments: other_price_AA, other_route_price_AA, other_price_DL,
other_route_price_DL, other_price_Other, other_route_price_Other,
other_price_UA, other_route_price_UA
Robust Covariance (Heteroskedastic)
Debiased: False

```

[126]: print(f"DL Demand (2SLS):")
       print(results_2sls['DL'].summary)

```

DL Demand (2SLS):

IV-2SLS Estimation Summary

```

=====
Dep. Variable:          logq_DL    R-squared:                0.5714

```

Estimator: IV-2SLS Adj. R-squared: 0.5581
 No. Observations: 167 F-statistic: 275.56
 Date: Mon, Feb 16 2026 P-value (F-stat) 0.0000
 Time: 21:18:31 Distribution: chi2(5)
 Cov. Estimator: robust

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	17.980	2.1570	8.3358	0.0000	13.753	22.208
avg_pop	0.0004	6.943e-05	5.6803	0.0000	0.0003	0.0005
logp_AA	0.0958	0.5586	0.1716	0.8638	-0.9990	1.1906
logp_DL	-3.0928	0.2466	-12.543	0.0000	-3.5760	-2.6095
logp_Other	0.3638	0.2266	1.6054	0.1084	-0.0803	0.8080
logp_UA	0.2462	0.2305	1.0680	0.2855	-0.2056	0.6979

Endogenous: logp_AA, logp_DL, logp_Other, logp_UA
 Instruments: other_price_AA, other_route_price_AA, other_price_DL,
 other_route_price_DL, other_price_Other, other_route_price_Other,
 other_price_UA, other_route_price_UA
 Robust Covariance (Heteroskedastic)
 Debiased: False

```
[127]: print(f"Other Demand (2SLS):")
        print(results_2sls['Other'].summary)
```

Other Demand (2SLS):

IV-2SLS Estimation Summary

Dep. Variable: logq_Other R-squared: 0.3530
 Estimator: IV-2SLS Adj. R-squared: 0.3329
 No. Observations: 167 F-statistic: 106.15
 Date: Mon, Feb 16 2026 P-value (F-stat) 0.0000
 Time: 21:18:31 Distribution: chi2(5)
 Cov. Estimator: robust

Parameter Estimates

	Parameter	Std. Err.	T-stat	P-value	Lower CI	Upper CI
const	15.024	2.1448	7.0046	0.0000	10.820	19.227
avg_pop	0.0006	7.163e-05	7.7799	0.0000	0.0004	0.0007
logp_AA	0.1721	0.3698	0.4653	0.6417	-0.5527	0.8968
logp_DL	0.1427	0.3028	0.4713	0.6375	-0.4508	0.7363
logp_Other	-1.9037	0.3225	-5.9022	0.0000	-2.5359	-1.2715
logp_UA	-0.0304	0.2720	-0.1118	0.9110	-0.5635	0.5027

```

=====

Endogenous: logp_AA, logp_DL, logp_Other, logp_UA
Instruments: other_price_AA, other_route_price_AA, other_price_DL,
other_route_price_DL, other_price_Other, other_route_price_Other,
other_price_UA, other_route_price_UA
Robust Covariance (Heteroskedastic)
Debiased: False

```

```

[ ]: # Simulate AA+UA merger
import numpy as np

# Extract coefficients
airlines = ['AA', 'DL', 'UA', 'Other']
beta = {j: {k: results_2sls[j].params[f'logp_{k}'] for k in airlines} for j in airlines}
alpha = {j: results_2sls[j].params['const'] for j in airlines}
beta_pop = {j: results_2sls[j].params['avg_pop'] for j in airlines}

# Get residuals (linearmodels uses .residuals, not .resid)
data_cf = data_wide.copy()
data_orig = data_wide.copy()
for j in ['AA', 'DL', 'UA', 'Other']:
    data_cf[f'resid_{j}'] = results_2sls[j].residuals

# Iterate to equilibrium
tol, max_iter = 0.01, 500
for it in range(max_iter):
    # Update log prices
    for a in ['AA', 'DL', 'UA', 'Other']:
        data_cf[f'logp_{a}'] = np.log(np.maximum(data_cf[f'price_{a}'], 0.01))

    # Predict quantities
    for j in ['AA', 'DL', 'UA']:
        data_cf[f'logq_{j}'] = (alpha[j] + beta_pop[j] * data_cf['avg_pop'] +
                                sum(beta[j][k] * data_cf[f'logp_{k}'] for k in airlines) +
                                data_cf[f'resid_{j}'])

    qty = {j: np.exp(data_cf[f'logq_{j}']) for j in ['AA', 'DL', 'UA']}
    ratio_UA_AA = qty['UA'] / qty['AA']
    ratio_AA_UA = qty['AA'] / qty['UA']

# FOCs: merged AA+UA, separate DL, Other fixed
p_new = {}
p_new['AA'] = (data_cf['mc_AA'] - data_cf['price_AA']/beta['AA']['AA'] -

```

```

        (data_cf['price_UA'] - data_cf['mc_UA']) * ratio_UA_AA *
↪beta['UA']['AA']/beta['AA']['AA'])
    p_new['UA'] = (data_cf['mc_UA'] - data_cf['price_UA']/beta['UA']['UA'] -
        (data_cf['price_AA'] - data_cf['mc_AA']) * ratio_AA_UA *
↪beta['AA']['UA']/beta['UA']['UA'])
    p_new['DL'] = data_cf['mc_DL'] - data_cf['price_DL']/beta['DL']['DL']

    # Check convergence
    diff = max([abs(p_new[j] - data_cf[f'price_{j}'])).max() for j in ['AA',
↪'DL', 'UA']])
    if diff < tol:
        print(f"Converged after {it+1} iterations (diff={diff:.6f})")
        break

    # Update prices with damping
    for j in ['AA', 'DL', 'UA']:
        data_cf[f'price_{j}'] = 0.1 * p_new[j] + 0.9 * data_cf[f'price_{j}']

# Final quantities
for a in ['AA', 'DL', 'UA', 'Other']:
    data_cf[f'logp_{a}'] = np.log(np.maximum(data_cf[f'price_{a}'], 0.01))
for j in ['AA', 'DL', 'UA']:
    data_cf[f'logq_{j}'] = (alpha[j] + beta_pop[j] * data_cf['avg_pop'] +
        sum(beta[j][k] * data_cf[f'logp_{k}'] for k in
↪airlines) +
        data_cf[f'resid_{j}'])

# Price changes
print("\nAverage Price Changes (%):")
for a in ['AA', 'DL', 'UA', 'Other']:
    pct_change = ((data_cf[f'price_{a}'] - data_orig[f'price_{a}']) /
↪data_orig[f'price_{a}'] * 100).mean()
    print(f" {a}: {pct_change:.2f}%")

# Profit changes
profit_pre = {j: (data_orig[f'price_{j}'] - data_orig[f'mc_{j}']) * np.
↪exp(data_orig[f'logq_{j}']})
    for j in ['AA', 'DL', 'UA']}
profit_post = {j: (data_cf[f'price_{j}'] - data_cf[f'mc_{j}']) * np.
↪exp(data_cf[f'logq_{j}']})
    for j in ['AA', 'DL', 'UA']}

print("\nAverage Profit Changes ($)")
print(f" AA+UA: ${((profit_post['AA'] + profit_post['UA'] - profit_pre['AA'] -
↪profit_pre['UA'])).mean():.2f}")
print(f" DL: ${((profit_post['DL'] - profit_pre['DL'])).mean():.2f}")

```

```
[128]: print(f"UA Demand (2SLS):")
print(results_2sls['UA'].summary)
```

UA Demand (2SLS):

IV-2SLS Estimation Summary

```
=====
Dep. Variable:          logq_UA    R-squared:                0.1010
Estimator:              IV-2SLS    Adj. R-squared:           0.0731
No. Observations:       167        F-statistic:              103.95
Date:                   Mon, Feb 16 2026    P-value (F-stat)          0.0000
Time:                   21:18:31    Distribution:              chi2(5)
Cov. Estimator:         robust
```

Parameter Estimates

```
=====
               Parameter  Std. Err.    T-stat    P-value    Lower CI    Upper CI
-----
const           25.036      3.0251     8.2763    0.0000     19.107     30.965
avg_pop          0.0003      0.0001     2.4930    0.0127     7.258e-05    0.0006
logp_AA         -0.7037      0.7723    -0.9112    0.3622     -2.2174     0.8099
logp_DL         -0.1084      0.3498    -0.3100    0.7566     -0.7941     0.5772
logp_Other      -0.9085      0.3641    -2.4954    0.0126     -1.6222    -0.1949
logp_UA         -1.9261      0.6300    -3.0571    0.0022     -3.1610    -0.6912
=====
```

Endogenous: logp_AA, logp_DL, logp_Other, logp_UA

Instruments: other_price_AA, other_route_price_AA, other_price_DL,
other_route_price_DL, other_price_Other, other_route_price_Other,
other_price_UA, other_route_price_UA

Robust Covariance (Heteroskedastic)

Debiased: False

Estimated Coefficients (2SLS):

See regression summaries above for actual coefficients.

Interpretation:

- **Own-price elasticities** (β_{jj}): Negative coefficients indicate demand decreases with own price.
- **Cross-price elasticities** (β_{jk} , β_{kj}): Positive coefficients indicate substitutes. Negative coefficients indicate complements.
- **Population effect** (β_{POP}): Positive and significant - larger markets have higher demand for all airlines.
- **Intercepts** (β_j): Capture airline-specific baseline demand levels.

1.0.5 Question 5

```
[133]: # Compute marginal costs:  $MC = P + P / \beta_{jj}$ 
airlines_mc = ['AA', 'DL', 'UA']

for airline in airlines_mc:
    beta_jj = results_2sls[airline].params[f'logp_{airline}']
    data_wide[f'mc_{airline}'] = data_wide[f'price_{airline}'] +
    ↪data_wide[f'price_{airline}'] / beta_jj

# Display summary statistics
print("Implied Marginal Costs Summary:")
for airline in airlines_mc:
    mc_col = f'mc_{airline}'
    valid_mc = data_wide[mc_col].dropna()

    print(f"\n{airline}:")
    print(f"  Mean MC: ${valid_mc.mean():.5f}")
    print(f"  Std Dev MC: ${valid_mc.std():.5f}")
    print(f"  Min MC: ${valid_mc.min():.5f}")
    print(f"  Max MC: ${valid_mc.max():.5f}")
```

Implied Marginal Costs Summary:

AA:

Mean MC: \$104.54141
Std Dev MC: \$50.63613
Min MC: \$29.80579
Max MC: \$429.59210

DL:

Mean MC: \$127.00208
Std Dev MC: \$86.91077
Min MC: \$21.65324
Max MC: \$494.64116

UA:

Mean MC: \$75.65881
Std Dev MC: \$46.71486
Min MC: \$1.44246
Max MC: \$232.71750

1.0.6 Question 6

```
[137]: # simulate AA+UA merger
import numpy as np

beta = lambda j, k: results_2sls[j].params[f'logp_{k}']
```

```

data_orig = data_wide.copy()
for a in ['AA', 'UA', 'DL']:
    data_orig[f'resid_{a}'] = results_2sls[a].resids.values
data_cf = data_orig.copy()

# find equilibrium
for it in range(500):
    for a in ['AA', 'UA', 'DL', 'Other']:
        data_cf[f'logp_{a}'] = np.log(np.maximum(data_cf[f'price_{a}'], 0.01))

    for a in ['AA', 'UA', 'DL']:
        data_cf[f'logq_{a}'] = (results_2sls[a].params['const'] +
                                results_2sls[a].params['avg_pop'] * data_cf['avg_pop'] +
                                sum(beta(a, k) * data_cf[f'logp_{k}'] for k in ['AA', 'DL', '
↪Other', 'UA'])) +
                                data_cf[f'resid_{a}'])

    q_AA, q_UA = np.exp(data_cf['logq_AA']), np.exp(data_cf['logq_UA'])
    r_UA_AA, r_AA_UA = q_UA / q_AA, q_AA / q_UA

    # FOCs: merged firm (AA+UA) internalizes cross-effects, DL Nash-Bertrand
    p_new_AA = (data_cf['mc_AA'] - data_cf['price_AA']/beta('AA', 'AA') -
                (data_cf['price_UA'] - data_cf['mc_UA']) * r_UA_AA * beta('UA', '
↪AA')/beta('AA', 'AA'))
    p_new_UA = (data_cf['mc_UA'] - data_cf['price_UA']/beta('UA', 'UA') -
                (data_cf['price_AA'] - data_cf['mc_AA']) * r_AA_UA * beta('AA', '
↪UA')/beta('UA', 'UA'))
    p_new_DL = data_cf['mc_DL'] - data_cf['price_DL']/beta('DL', 'DL')

    # check for convergence
    diff = max(np.abs(p_new_AA - data_cf['price_AA']).max(),
               np.abs(p_new_UA - data_cf['price_UA']).max(),
               np.abs(p_new_DL - data_cf['price_DL']).max())
    if diff < 0.01:
        print(f"converged after {it+1} iterations")
        break

    # update prices with damping
    data_cf['price_AA'] = 0.1*p_new_AA + 0.9*data_cf['price_AA']
    data_cf['price_UA'] = 0.1*p_new_UA + 0.9*data_cf['price_UA']
    data_cf['price_DL'] = 0.1*p_new_DL + 0.9*data_cf['price_DL']

# quantities
for a in ['AA', 'UA', 'DL']:
    data_cf[f'logp_{a}'] = np.log(np.maximum(data_cf[f'price_{a}'], 0.01))
    data_cf[f'logq_{a}'] = (results_2sls[a].params['const'] +

```

```

        results_2sls[a].params['avg_pop'] * data_cf['avg_pop'] +
        sum(beta(a, k) * data_cf[f'logp_{k}'] for k in ['AA', 'DL', 'Other',
        ↪ 'UA']) +
        data_cf[f'resid_{a}'])

```

converged after 122 iterations

```

[142]: # price changes
price_changes = {}
for a in ['AA', 'UA', 'DL', 'Other']:
    if a == 'Other':
        price_changes[a] = 0.0
    else:
        price_changes[a] = ((data_cf[f'price_{a}'] - data_orig[f'price_{a}']) /
        ↪ data_orig[f'price_{a}'] * 100).mean()

print("Average Price Changes (%):")
for a in ['AA', 'UA', 'DL', 'Other']:
    print(f"  {a}: {price_changes[a]:.2f}%")

```

Average Price Changes (%):

```

AA: -13.24%
UA: -21.99%
DL: 0.00%
Other: 0.00%

```

```

[141]: # Profit changes
profit_pre = {}
profit_post = {}
for a in ['AA', 'UA', 'DL']:
    profit_pre[a] = (data_orig[f'price_{a}'] - data_orig[f'mc_{a}']) * np.
    ↪ exp(data_orig[f'logq_{a}'])
    profit_post[a] = (data_cf[f'price_{a}'] - data_cf[f'mc_{a}']) * np.
    ↪ exp(data_cf[f'logq_{a}'])

print("Average Profit Changes ($):")
for a in ['AA', 'UA', 'DL']:
    print(f"  {a}: ${ (profit_post[a] - profit_pre[a]).mean():.2f}")
print(f"\n AA+UA (combined): ${ (profit_post['AA'] + profit_post['UA'] -
    ↪ profit_pre['AA'] - profit_pre['UA']).mean():.2f}")

```

Average Profit Changes (\$):

```

AA: $113,277.69
UA: $127,521.50
DL: $-23,511.78

```

AA+UA (combined): \$240,799.20

1.0.7 Question 7

```
[157]: # simulate AA+DL and UA+DL mergers
def simulate_merger(merged_airlines, data_orig):
    data_cf = data_orig.copy()
    beta = lambda j, k: results_2sls[j].params[f'logp_{k}']

    for it in range(500):
        # update log prices
        for a in ['AA', 'UA', 'DL', 'Other']:
            data_cf[f'logp_{a}'] = np.log(np.maximum(data_cf[f'price_{a}'], 0.
↪01))

        # predict quantities
        for a in ['AA', 'UA', 'DL']:
            data_cf[f'logq_{a}'] = (results_2sls[a].params['const'] +
                                   results_2sls[a].params['avg_pop'] * data_cf['avg_pop'] +
                                   sum(beta(a, k) * data_cf[f'logp_{k}'] for k in ['AA', 'DL',
↪'Other', 'UA'])) +
                                   data_cf[f'resid_{a}'])

        # get quantities and ratios for merged firms
        q = {a: np.exp(data_cf[f'logq_{a}']) for a in merged_airlines}

        # FOCs: merged firms internalize cross-effects, others Nash-Bertrand
        p_new = {}
        if set(merged_airlines) == {'AA', 'DL'}:
            r_DL_AA, r_AA_DL = q['DL'] / q['AA'], q['AA'] / q['DL']
            p_new['AA'] = (data_cf['mc_AA'] - data_cf['price_AA']/beta('AA',
↪'AA')) -
                        (data_cf['price_DL'] - data_cf['mc_DL']) * r_DL_AA *
↪beta('DL', 'AA')/beta('AA', 'AA'))
            p_new['DL'] = (data_cf['mc_DL'] - data_cf['price_DL']/beta('DL',
↪'DL')) -
                        (data_cf['price_AA'] - data_cf['mc_AA']) * r_AA_DL *
↪beta('AA', 'DL')/beta('DL', 'DL'))
            p_new['UA'] = data_cf['mc_UA'] - data_cf['price_UA']/beta('UA',
↪'UA')
        elif set(merged_airlines) == {'UA', 'DL'}:
            r_DL_UA, r_UA_DL = q['DL'] / q['UA'], q['UA'] / q['DL']
            p_new['UA'] = (data_cf['mc_UA'] - data_cf['price_UA']/beta('UA',
↪'UA')) -
                        (data_cf['price_DL'] - data_cf['mc_DL']) * r_DL_UA *
↪beta('DL', 'UA')/beta('UA', 'UA'))
            p_new['DL'] = (data_cf['mc_DL'] - data_cf['price_DL']/beta('DL',
↪'DL')) -
```

```

        (data_cf['price_UA'] - data_cf['mc_UA']) * r_UA_DL *
↪beta('UA', 'DL')/beta('DL', 'DL'))
        p_new['AA'] = data_cf['mc_AA'] - data_cf['price_AA']/beta('AA',
↪'AA'))
        else: # AA+UA
            r_UA_AA, r_AA_UA = q['UA'] / q['AA'], q['AA'] / q['UA']
            p_new['AA'] = (data_cf['mc_AA'] - data_cf['price_AA']/beta('AA',
↪'AA')) -
                (data_cf['price_UA'] - data_cf['mc_UA']) * r_UA_AA *
↪beta('UA', 'AA')/beta('AA', 'AA'))
            p_new['UA'] = (data_cf['mc_UA'] - data_cf['price_UA']/beta('UA',
↪'UA')) -
                (data_cf['price_AA'] - data_cf['mc_AA']) * r_AA_UA *
↪beta('AA', 'UA')/beta('UA', 'UA'))
            p_new['DL'] = data_cf['mc_DL'] - data_cf['price_DL']/beta('DL',
↪'DL'))

        # check convergence
        diff = max([np.abs(p_new[a] - data_cf[f'price_{a}'])).max() for a in
↪p_new.keys()])
        if diff < 0.01:
            break

        # Update prices
        for a in p_new.keys():
            data_cf[f'price_{a}'] = 0.1*p_new[a] + 0.9*data_cf[f'price_{a}']

        # Final quantities
        for a in ['AA', 'UA', 'DL', 'Other']:
            data_cf[f'logp_{a}'] = np.log(np.maximum(data_cf[f'price_{a}'], 0.01))
        for a in ['AA', 'UA', 'DL']:
            data_cf[f'logq_{a}'] = (results_2sls[a].params['const'] +
                results_2sls[a].params['avg_pop'] * data_cf['avg_pop'] +
                sum(beta(a, k) * data_cf[f'logp_{k}'] for k in ['AA', 'DL',
↪'Other', 'UA'])) +
            data_cf[f'resid_{a}'])

        return data_cf

# from previous question
results_mergers = {'AA+UA': data_cf}

for merger in [['AA', 'DL'], ['UA', 'DL']]:
    merger_name = '+'.join(merger)
    print(f"Simulating {merger_name} merger...")
    results_mergers[merger_name] = simulate_merger(merger, data_orig)

```

Simulating AA+DL merger...
Simulating UA+DL merger...

1.0.8 Compare Results

```
[158]: # compare price changes across all 3 mergers
print("Price Changes (%) by Merger:\n")
price_comparison = pd.DataFrame({
    'Airline': ['AA', 'UA', 'DL', 'Other']
})

for merger_name in ['AA+UA', 'AA+DL', 'UA+DL']:
    data_cf_merger = results_mergers[merger_name]
    price_changes = []
    for a in ['AA', 'UA', 'DL', 'Other']:
        if a == 'Other':
            price_changes.append(0.0)
        else:
            pct = ((data_cf_merger[f'price_{a}'] - data_orig[f'price_{a}']) /
data_orig[f'price_{a}'] * 100).mean()
            price_changes.append(pct)
    price_comparison[merger_name] = price_changes

print(price_comparison.round(2))
```

Price Changes (%) by Merger:

	Airline	AA+UA	AA+DL	UA+DL
0	AA	-13.24	6.61	0.00
1	UA	-21.99	0.00	8.06
2	DL	0.00	6.58	-20.63
3	Other	0.00	0.00	0.00

```
[160]: # compare profit changes across all 3 mergers
print("profit Changes by Merger:\n")
profit_comparison = pd.DataFrame({
    'Airline': ['AA', 'UA', 'DL', 'Combined']
})

for merger_name in ['AA+UA', 'AA+DL', 'UA+DL']:
    data_cf_merger = results_mergers[merger_name]
    profit_changes = []

    # individual airline profits
    for a in ['AA', 'UA', 'DL']:
        profit_pre = (data_orig[f'price_{a}'] - data_orig[f'mc_{a}']) * np.
exp(data_orig[f'logq_{a}'])
```

```

        profit_post = (data_cf_merger[f'price_{a}'] -
↳data_cf_merger[f'mc_{a}']) * np.exp(data_cf_merger[f'logq_{a}'])
        profit_changes.append((profit_post - profit_pre).mean())

    # combined merged firm profit
    merged = merger_name.split('+')
    profit_pre_combined = sum([(data_orig[f'price_{a}'] - data_orig[f'mc_{a}'])
↳ np.exp(data_orig[f'logq_{a}']) for a in merged])
    profit_post_combined = sum([(data_cf_merger[f'price_{a}'] -
↳data_cf_merger[f'mc_{a}']) * np.exp(data_cf_merger[f'logq_{a}']) for a in
↳merged])
    profit_changes.append((profit_post_combined - profit_pre_combined).mean())

    profit_comparison[merger_name] = profit_changes

print(profit_comparison.round(2))

```

profit Changes by Merger:

	Airline	AA+UA	AA+DL	UA+DL
0	AA	113277.69	2881.77	-420449.65
1	UA	127521.50	-44246.22	73643.26
2	DL	-23511.78	6007.54	-13545.20
3	Combined	240799.20	8889.31	60098.05

Price Effects:

1. **AA+UA merger:** Both merged airlines reduce prices (AA: -13.24%, UA: -21.99%), while DL's price remains unchanged. This suggests AA and UA are close substitutes, and internalizing cross-price effects leads to lower prices.
2. **AA+DL merger:** The merged firm (AA+DL) internalizes cross-effects, while UA adjusts competitively.
3. **UA+DL merger:** Similar pattern with UA+DL merged and AA as the competitor.

Profit Effects:

1. **AA+UA merger:** The merged firm gains \$240,799 in combined profits, while DL loses \$23,512. The price reductions increase market share and total demand, offsetting lower margins.
2. **AA+DL merger:** The small combined profit gain suggests AA and DL are less substitutable than AA and UA, so internalizing cross-effects provides less strategic advantage.
3. **UA+DL merger:** AA is highly vulnerable when its two main competitors merge. This suggests AA competes closely with both UA and DL.

Overall, despite price reductions, merged firms can increase profits by capturing market share from competitors and internalizing competitive externalities. We also see, that the non-merged competitor adjusts prices according to Nash-Bertrand, but may still lose profits due to the merged

firm's strategic advantage. The magnitude of price and profit changes depends on the cross-price elasticities between merging firms. Higher substitutability leads to larger strategic effects.