Problem Set 3 - Grant Jackson

September 30, 2024

[1]:
```
# Importing dataset
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_stata("C:\\Users\gmoor\Documents\Applied Microeconomics\Data\cars1.
 ↪dta")

df.head()
```

[1]:

	year	country	co	type	segment	domestic	firm	brand	loc	\
0	1983	Belgium	1	alfa 33	compact	0	AlfaRomeo	AlfaRomeo	Italy	
1	1984	Belgium	1	alfa 33	compact	0	AlfaRomeo	AlfaRomeo	Italy	
2	1985	Belgium	1	alfa 33	compact	0	AlfaRomeo	AlfaRomeo	Italy	
3	1986	Belgium	1	alfa 33	compact	0	AlfaRomeo	AlfaRomeo	Italy	
4	1987	Belgium	1	alfa 33	compact	0	Fiat	AlfaRomeo	Italy	

	qu	…	weight	pop	ngdp	ngdpe	country1	\
0	729.0	…	890	9860000.0	4.188800e+12	234000000.0	1	
1	1860.0	…	890	9860000.0	4.512600e+12	234000000.0	1	
2	1771.0	…	890	9860000.0	4.834400e+12	234000000.0	1	
3	2047.0	…	890	9860000.0	5.084900e+12	234000000.0	1	
4	2147.0	…	910	9870000.0	5.318700e+12	234000000.0	1	

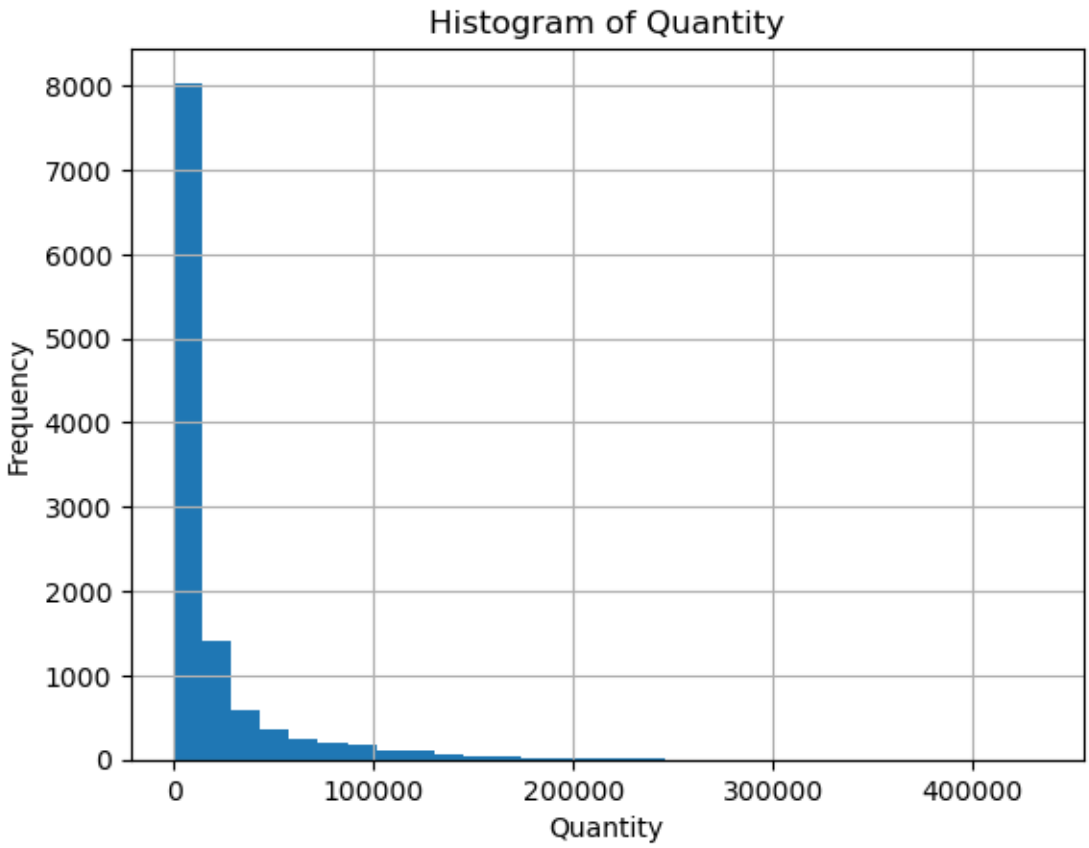
	country2	country3	country4	country5	yearsquared
0	0	0	0	0	3932289.0
1	0	0	0	0	3936256.0
2	0	0	0	0	3940225.0
3	0	0	0	0	3944196.0
4	0	0	0	0	3948169.0

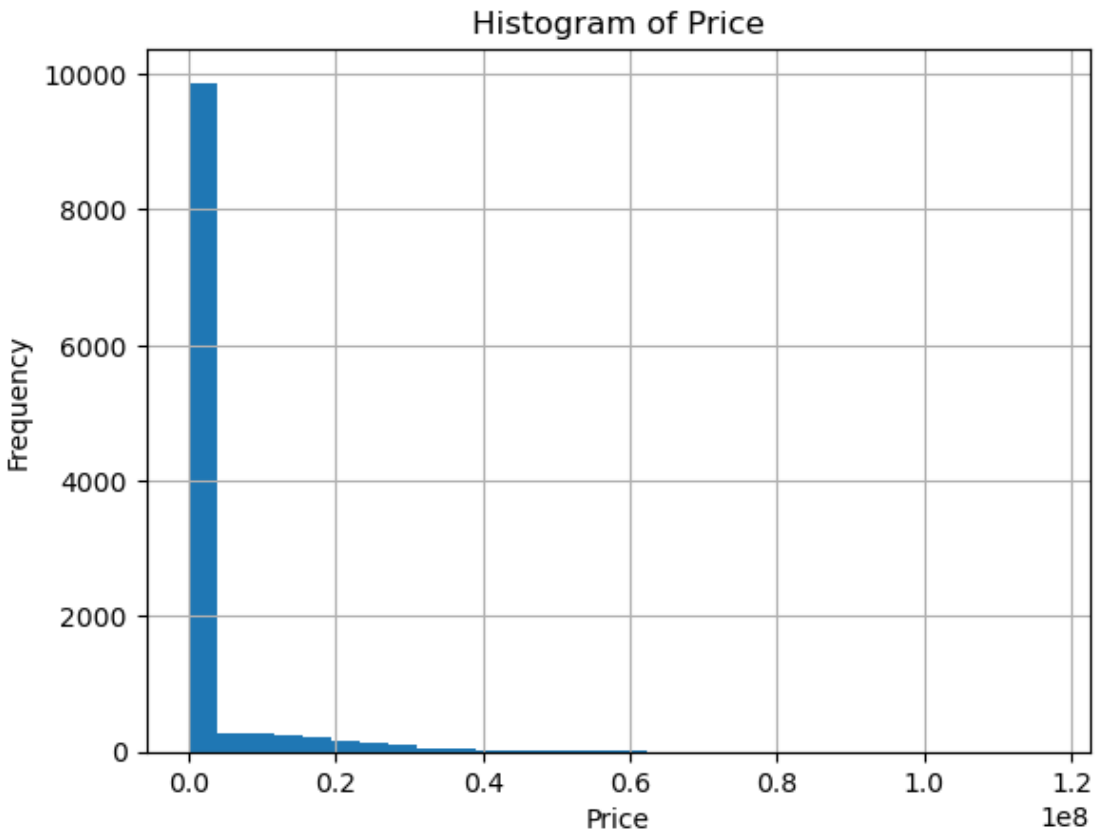
[5 rows x 27 columns]

1 (a) Summary statistics and correlation between LogPrice and LogQuantity:

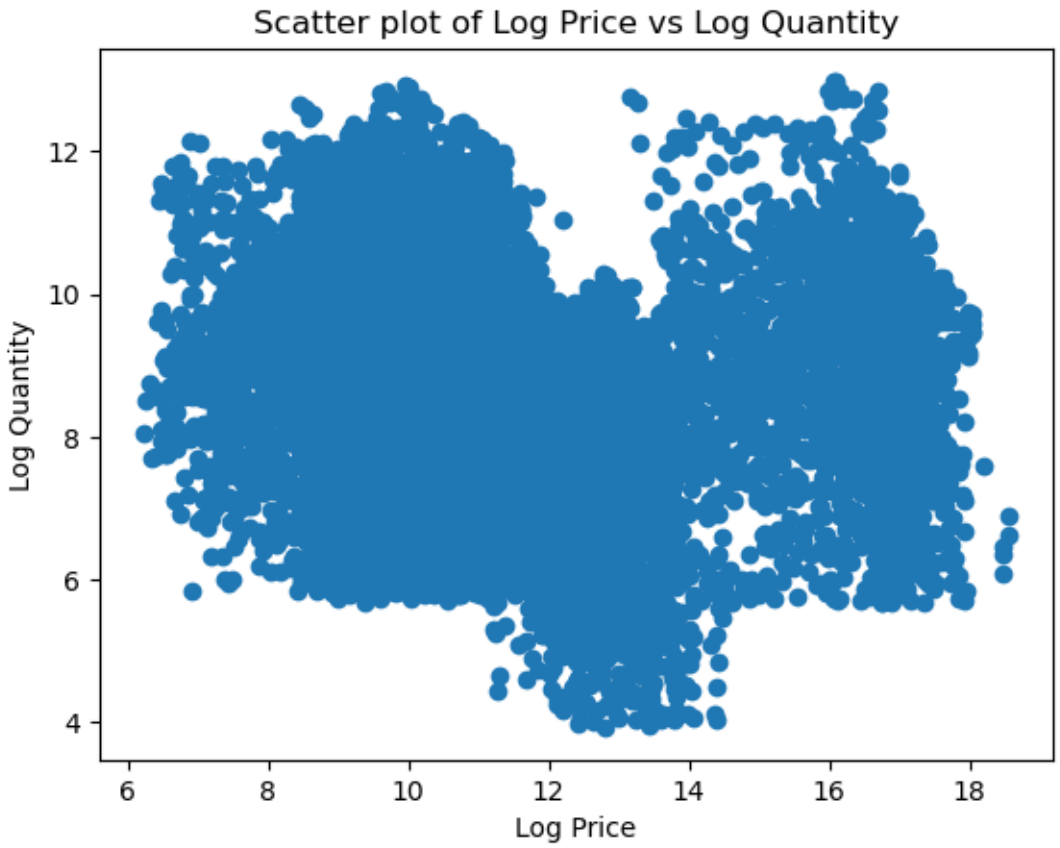
[2]:
```
# Summary statistics for 'quantity' and 'price'
print(df[['qu', 'pr']].describe())

# Histogram of 'quantity'
df['qu'].hist(bins=30)
plt.title('Histogram of Quantity')
plt.xlabel('Quantity')
plt.ylabel('Frequency')
plt.show()

# Histogram of 'price'
df['pr'].hist(bins=30)
plt.title('Histogram of Price')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()

# Tabulate 'country'
print(df['country'].value_counts())

# Log transformations
df['lnq'] = np.log(df['qu'])
df['lnp'] = np.log(df['pr'])

# Scatter plot of log(price) vs log(quantity)
plt.scatter(df['lnp'], df['lnq'])
plt.title('Scatter plot of Log Price vs Log Quantity')
plt.xlabel('Log Price')
plt.ylabel('Log Quantity')
plt.show()

# Calculate and print correlation between log price and log quantity
print("Correlation between log(price) and log(quantity):", df[['lnp', 'lnq']].
  ↪corr())
```

```
                 qu            pr
count   11483.000000  1.148300e+04
mean    19911.439453  2.857566e+06
std     37803.589844  8.237671e+06
min        51.000000  4.980000e+02
25%      1992.500000  1.324500e+04
50%      6262.000000  5.590000e+04
75%     18855.500000  4.136750e+05
max    433694.000000  1.166610e+08
```


Histogram of Quantity
Frequency
Quantity
0
1000
2000
3000
4000
5000
6000
7000
8000
0
100000
200000
300000
400000




```
country
Belgium    2641
UK         2289
Germany    2281
France     2252
Italy      2020
Name: count, dtype: int64
```




```
Correlation between log(price) and log(quantity):          lnp       lnq
lnp  1.000000 -0.135325
lnq -0.135325  1.000000
```

2 (b) OLS-Fixed effects estimator of the standard logic model:

[3]:
```
import statsmodels.formula.api as smf

# Construct market size, shares, and log-odds ratio
df['logpop'] = np.log(df['pop'])
df['loggdp'] = np.log(df['ngdp'])
df['msize'] = df['pop'] / 4
df['share'] = df['qu'] / df['msize']

# Calculate the outside goods market share
df['sum_share'] = df.groupby(['country', 'year'])['share'].transform('sum')
df['share0'] = 1 - df['sum_share']

# Generate log odds ratio
```

```
df['lsj_ls0'] = np.log(df['share'] / df['share0'])

# Run the fixed-effects OLS regression
fixed_effects_model = smf.ols(formula='lsj_ls0 ~ price + horsepower + fuel +␣
 ↪width + domestic + height + weight + logpop + loggdp + C(country) + C(year)␣
 ↪+ C(brand)', data=df).fit(cov_type='HC3')

# Print regression results
print(fixed_effects_model.summary())
```

```
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2935626821.py:10:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df['sum_share'] = df.groupby(['country', 'year'])['share'].transform('sum')
```

OLS Regression Results

Dep. Variable:	lsj_ls0	R-squared:	0.574
Model:	OLS	Adj. R-squared:	0.571
Method:	Least Squares	F-statistic:	215.7
Date:	Mon, 30 Sep 2024	Prob (F-statistic):	0.00
Time:	22:31:41	Log-Likelihood:	-16060.
No. Observations:	11483	AIC:	3.228e+04
Df Residuals:	11403	BIC:	3.287e+04
Df Model:	79		
Covariance Type:	HC3		

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-25.4813	3.676	-6.931	0.000	-32.687	-18.276
C(country)[T.France]	-2.1647	0.337	-6.425	0.000	-2.825	-1.504
C(country)[T.Germany]	-2.1353	0.377	-5.663	0.000	-2.874	-1.396
C(country)[T.Italy]	-2.5403	0.453	-5.605	0.000	-3.429	-1.652
C(country)[T.UK]	-1.4538	0.341	-4.265	0.000	-2.122	-0.786
C(year)[T.1971]	-0.0607	0.097	-0.628	0.530	-0.250	0.129
C(year)[T.1972]	0.0047	0.094	0.050	0.960	-0.180	0.189
C(year)[T.1973]	0.0329	0.099	0.333	0.739		

-0.160	0.226				
C(year)[T.1974]		-0.2245	0.093	-2.407	0.016
-0.407	-0.042				
C(year)[T.1975]		-0.1367	0.094	-1.455	0.146
-0.321	0.047				
C(year)[T.1976]		-0.0794	0.098	-0.812	0.417
-0.271	0.112				
C(year)[T.1977]		-0.1148	0.101	-1.132	0.258
-0.314	0.084				
C(year)[T.1978]		-0.1405	0.104	-1.357	0.175
-0.343	0.062				
C(year)[T.1979]		-0.1670	0.107	-1.563	0.118
-0.376	0.042				
C(year)[T.1980]		-0.2003	0.108	-1.849	0.064
-0.413	0.012				
C(year)[T.1981]		-0.2450	0.111	-2.203	0.028
-0.463	-0.027				
C(year)[T.1982]		-0.2763	0.115	-2.396	0.017
-0.502	-0.050				
C(year)[T.1983]		-0.4077	0.118	-3.447	0.001
-0.640	-0.176				
C(year)[T.1984]		-0.4909	0.123	-3.989	0.000
-0.732	-0.250				
C(year)[T.1985]		-0.5242	0.126	-4.145	0.000
-0.772	-0.276				
C(year)[T.1986]		-0.3772	0.130	-2.897	0.004
-0.632	-0.122				
C(year)[T.1987]		-0.2583	0.133	-1.946	0.052
-0.518	0.002				
C(year)[T.1988]		-0.2103	0.136	-1.548	0.122
-0.477	0.056				
C(year)[T.1989]		-0.1965	0.140	-1.407	0.159
-0.470	0.077				
C(year)[T.1990]		-0.1241	0.140	-0.889	0.374
-0.398	0.150				
C(year)[T.1991]		-0.3117	0.148	-2.100	0.036
-0.603	-0.021				
C(year)[T.1992]		-0.2175	0.150	-1.448	0.148
-0.512	0.077				
C(year)[T.1993]		-0.3626	0.151	-2.403	0.016
-0.658	-0.067				
C(year)[T.1994]		-0.3273	0.153	-2.135	0.033
-0.628	-0.027				
C(year)[T.1995]		-0.4114	0.158	-2.608	0.009
-0.721	-0.102				
C(year)[T.1996]		-0.3800	0.160	-2.372	0.018
-0.694	-0.066				
C(year)[T.1997]		-0.3237	0.161	-2.016	0.044

```
-0.638      -0.009
C(year)[T.1998]                -0.2896      0.163     -1.772      0.076
-0.610       0.031
C(year)[T.1999]                -0.2987      0.165     -1.810      0.070
-0.622       0.025
C(brand)[T.Audi]                0.3391      0.079      4.318      0.000
0.185       0.493
C(brand)[T.BMW]                 1.2500      0.066     18.933      0.000
1.121       1.379
C(brand)[T.Citroën]             0.0385      0.064      0.604      0.546
-0.086       0.164
C(brand)[T.Daihatsu]           -1.8908      0.093    -20.279      0.000
-2.073      -1.708
C(brand)[T.Fiat]                0.1324      0.062      2.129      0.033
0.010       0.254
C(brand)[T.Ford]                0.4308      0.066      6.495      0.000
0.301       0.561
C(brand)[T.Honda]              -0.0663      0.069     -0.963      0.336
-0.201       0.069
C(brand)[T.Hyundai]            -1.2078      0.091    -13.225      0.000
-1.387      -1.029
C(brand)[T.Innocenti]          -1.5915      0.184     -8.644      0.000
-1.952      -1.231
C(brand)[T.Lancia]             -0.6938      0.066    -10.590      0.000
-0.822      -0.565
C(brand)[T.Mazda]              -0.5570      0.078     -7.112      0.000
-0.710      -0.403
C(brand)[T.Mercedes]            1.9318      0.069     28.071      0.000
1.797       2.067
C(brand)[T.Mitsubishi]         -0.8001      0.078    -10.286      0.000
-0.953      -0.648
C(brand)[T.NissanDatsun]        0.1087      0.065      1.662      0.097
-0.020       0.237
C(brand)[T.OpelVauxhall]        0.1819      0.066      2.743      0.006
0.052       0.312
C(brand)[T.Peugeot]             0.3781      0.064      5.928      0.000
0.253       0.503
C(brand)[T.Renault]             0.4264      0.062      6.902      0.000
0.305       0.548
C(brand)[T.RoverTriumph]       -0.5540      0.067     -8.219      0.000
-0.686      -0.422
C(brand)[T.Saab]                0.0122      0.077      0.158      0.875
-0.139       0.163
C(brand)[T.Seat]               -0.6095      0.087     -7.029      0.000
-0.779      -0.440
C(brand)[T.Skoda]              -0.5198      0.109     -4.790      0.000
-0.733      -0.307
C(brand)[T.Suzuki]             -1.2152      0.085    -14.360      0.000
```

```
-1.381      -1.049
C(brand)[T.Toyota]                -0.0785      0.068     -1.153      0.249
-0.212       0.055
C(brand)[T.Volkswagen]             0.4506      0.069      6.511      0.000
0.315       0.586
C(brand)[T.Volvo]                  0.5539      0.072      7.689      0.000
0.413       0.695
C(brand)[T.Talbot]                -0.5677      0.102     -5.588      0.000
-0.767      -0.369
C(brand)[T.Kia]                   -2.0625      0.132    -15.595      0.000
-2.322      -1.803
C(brand)[T.Daewoo]                -0.7605      0.115     -6.638      0.000
-0.985      -0.536
C(brand)[T.Rover]                 -0.4990      0.126     -3.964      0.000
-0.746      -0.252
C(brand)[T.MCC]                   -0.7572      0.624     -1.214      0.225
-1.979       0.465
C(brand)[T.DAF]                    0.2415      0.126      1.912      0.056
-0.006       0.489
C(brand)[T.Autobianchi]           -0.5020      0.199     -2.518      0.012
-0.893      -0.111
C(brand)[T.Triumph]               -0.5173      0.188     -2.750      0.006
-0.886      -0.149
C(brand)[T.Princess]              -1.0146      0.273     -3.721      0.000
-1.549      -0.480
C(brand)[T.TalbotHillmanChrysler] -0.9766      0.243     -4.023      0.000
-1.452      -0.501
C(brand)[T.TalbotSimca]            0.4144      0.098      4.223      0.000
0.222       0.607
C(brand)[T.TalbotMatra]           -1.7697      0.150    -11.833      0.000
-2.063      -1.477
price                             -0.0540      0.005    -11.646      0.000
-0.063      -0.045
horsepower                        -0.0170      0.002    -11.122      0.000
-0.020      -0.014
fuel                              -0.0719      0.013     -5.451      0.000
-0.098      -0.046
width                              0.0533      0.002     21.341      0.000
0.048       0.058
domestic                           1.7841      0.027     66.975      0.000
1.732       1.836
height                            -0.0147      0.003     -5.743      0.000
-0.020      -0.010
weight                            -0.0008      0.000     -5.146      0.000
-0.001      -0.001
logpop                             0.7517      0.197      3.823      0.000
0.366       1.137
loggdp                             0.1071      0.051      2.112      0.035
```

```
0.008       0.206
==============================================================================
Omnibus:                      648.617   Durbin-Watson:                   0.604
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              777.159
Skew:                          -0.589   Prob(JB):                    1.75e-169
Kurtosis:                       3.488   Cond. No.                     4.25e+05
==============================================================================

Notes:
[1] Standard Errors are heteroscedasticity robust (HC3)
[2] The condition number is large, 4.25e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

2.1 (b) Interpretation: The OLS regression results indicate price, horsepower, fuel comsumption, and several other product characterisitcs signficantly impact the log-oods ration of market shares. The fixed effects for countries, years, and brands capture varations that are not directly explaiend by product characteristics. We can assume that higher prices reduce market share and domestics brands perform better.

3 (c) IV estimation of the standard logic model:

3.1 [i]:

[4]:
```
# Create the 'ones' column explicitly
df['ones'] = 1

# Group by country and year
group_cols = ['country', 'year']

# IV1 generation for each characteristic
df['numJ'] = df.groupby(group_cols)['ones'].transform('sum')

for col in ['horsepower', 'fuel', 'width', 'height', 'domestic', 'weight']:
    df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
    df[f'IV1_{col}'] = df['numJ'] * df[col] - df[f'sum_{col}']
    df.drop(columns=[f'sum_{col}'], inplace=True)

# IV2 generation for each characteristic
for col in ['horsepower', 'fuel', 'width', 'height', 'domestic', 'weight']:
    df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
    df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.
 ↪var(x, ddof=0))
    df[f'IV2_{col}'] = df['numJ'] * (df[col] - df[f'mean_{col}']) ** 2 +␣
 ↪df['numJ'] * df[f'var_{col}']
    df.drop(columns=[f'mean_{col}', f'var_{col}'], inplace=True)
```

```
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:8:
```

```
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df['numJ'] = df.groupby(group_cols)['ones'].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:11:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:11:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:11:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:11:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:11:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:11:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'sum_{col}'] = df.groupby(group_cols)[col].transform('sum')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:17:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:18:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.var(x,
ddof=0))
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:17:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
```

```
behavior or observed=True to adopt the future default and silence this warning.
  df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:18:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.var(x,
ddof=0))
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:17:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:18:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.var(x,
ddof=0))
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:17:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:18:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.var(x,
ddof=0))
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:17:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:18:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.var(x,
ddof=0))
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:17:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'mean_{col}'] = df.groupby(group_cols)[col].transform('mean')
C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\2348775012.py:18:
FutureWarning: The default of observed=False is deprecated and will be changed
```

```
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  df[f'var_{col}'] = df.groupby(group_cols)[col].transform(lambda x: np.var(x,
ddof=0))
```

3.2 [ii]:

[5]:
```
from linearmodels.iv import IV2SLS

# Define the endogenous variable and instruments
endog = df['price']
exog = df[['horsepower', 'fuel', 'width', 'domestic', 'height', 'weight',␣
 ↪'logpop', 'loggdp']]
iv = df[['IV1_horsepower', 'IV1_fuel', 'IV1_width', 'IV1_domestic',␣
 ↪'IV1_height', 'IV1_weight',
         'IV2_horsepower', 'IV2_fuel', 'IV2_width', 'IV2_domestic',␣
 ↪'IV2_height', 'IV2_weight']]

# Run 2SLS
iv_model = IV2SLS(df['lsj_ls0'], exog, endog, iv).fit(cov_type='robust')

# Print IV regression results
print(iv_model.summary)
```

```
                          IV-2SLS Estimation Summary
==============================================================================
Dep. Variable:                 lsj_ls0   R-squared:                      0.9759
Estimator:                     IV-2SLS   Adj. R-squared:                 0.9759
No. Observations:                11483   F-statistic:                 4.739e+05
Date:                Mon, Sep 30 2024   P-value (F-stat)                0.0000
Time:                        22:31:41   Distribution:                  chi2(9)
Cov. Estimator:                robust

                             Parameter Estimates
==============================================================================
            Parameter  Std. Err.     T-stat    P-value    Lower CI    Upper CI
------------------------------------------------------------------------------
horsepower    -0.0403     0.0016    -24.631     0.0000     -0.0435     -0.0371
fuel          -0.1257     0.0090    -13.986     0.0000     -0.1433     -0.1081
width          0.0407     0.0020     20.643     0.0000      0.0368      0.0445
domestic       1.8663     0.0277     67.311     0.0000      1.8120      1.9206
height        -0.0212     0.0021    -10.342     0.0000     -0.0252     -0.0172
weight         0.0009     0.0002     5.8332     0.0000      0.0006      0.0012
logpop        -0.5107     0.0133    -38.260     0.0000     -0.5369     -0.4846
loggdp        -0.0074     0.0053    -1.3823     0.1669     -0.0179      0.0031
price          0.0074     0.0054     1.3780     0.1682     -0.0031      0.0179
==============================================================================
```

```
Endogenous: price
Instruments: IV1_horsepower, IV1_fuel, IV1_width, IV1_domestic, IV1_height,
IV1_weight, IV2_horsepower, IV2_fuel, IV2_width, IV2_domestic, IV2_height,
IV2_weight
Robust Covariance (Heteroskedastic)
Debiased: False
```

3.3 (c) Interpretation: IV-2SLS regression results indicate that price does not significantly affect the logg-odds ratio of market shares, unlike the OLS regression results. Product characteristics like fuel efficincy, horsepower, width, and whether the product is dommestic or not are a critical role in determining the market share.

4 (d) Willingness to pay (WTP) for fuel efficiency:

[6]:
```python
# Extract coefficients
alpha_price = iv_model.params['price']
beta_fuel = iv_model.params['fuel']

# Calculate WTP
wtp = -beta_fuel / alpha_price
print("WTP =", wtp)
```

```
WTP = 17.02297811927135
```

5 (e) Price elasticity of demand:

[7]:
```python
# Calculate price elasticity for each observation
df['elasticity'] = -alpha_price * (1 - df['share']) * df['price']

# Summary statistics for elasticity
print(df['elasticity'].describe())

# Grouped summary by country
print(df.groupby('country')['elasticity'].describe())
```

```
count    11483.000000
mean        -0.136380
std          0.065894
min         -1.109948
25%         -0.167440
50%         -0.120858
75%         -0.090494
max         -0.038727
Name: elasticity, dtype: float64
          count      mean       std       min       25%       50%       75%  \
country
```

```
Belgium  2641.0 -0.122916  0.057286 -0.584867 -0.148144 -0.108314 -0.080914
France   2252.0 -0.126195  0.054265 -0.605196 -0.151067 -0.113383 -0.088205
Germany  2281.0 -0.115675  0.047870 -0.434030 -0.139046 -0.102258 -0.081249
Italy    2020.0 -0.139380  0.068858 -0.754498 -0.172105 -0.121909 -0.093703
UK       2289.0 -0.179921  0.077428 -1.109948 -0.218538 -0.162612 -0.128583

              max
country
Belgium -0.041893
France  -0.044899
Germany -0.042839
Italy   -0.038727
UK      -0.055180

C:\Users\gmoor\AppData\Local\Temp\ipykernel_14212\3733935762.py:8:
FutureWarning: The default of observed=False is deprecated and will be changed
to True in a future version of pandas. Pass observed=False to retain current
behavior or observed=True to adopt the future default and silence this warning.
  print(df.groupby('country')['elasticity'].describe())
```