ECON 8000/9000 Empirical Energy Economics
PS8: Exercise: Simple Multinomial Logit Demand Estimation
Christy Zhou

Instruction and Notes:

- Deadline: Sunday 5/31 @ 5pm
- Submission: Please compile all results into a single PDF.
- Group: Everyone can work together, but only **up to two students** can submit the assignment together. If so, make sure you include both names at the top when you submit.
- Here I write instructions using Stata. You can use other software if you like.
- This study has a replication package online. I recommend you follow my instructions below, which are slightly less complicated than the online package.

In this exercise, Q1 will ask you to estimate a simple multinomial Logit in Stata. Q2 will ask you to reproduce some key tasks in Python's linear regression model. Q3 will introduce you to the syntax of PyBLP in Python.

Q1. Estimate a simple multinomial Logit in Stata

Key Info:

- Dataset: "ps8data_df_at_productlevel.dta"
- This is a yearly dataset from model year 2014 to model year 2018
- I have aggregated confidential product-level vehicle data to a shareable level. In this exercise, a product is defined as a unique combination of make, fuel type, and body style (or regarded as segment). You can find them using "make_nw fuel_type_nw body_style_nw".
- Key P & Q information
  - sales (in sales_count)
  - price (in price_usd2022)
  - total market size defined as in GMY (2025), (already given in totalmktsize)
- Attributes that we will use
  - Dollar-per-mile (in dpm_usd2022)
  - Horsepower-to-weight ratio (in hpwt_w)
  - Footprint (in footprint_w)
  - Alternative vehicle fuel types (in i_fuel_hybrid and i_fuel_ev)
  - Segment/body styles types (in i_segsub_*, including 8 of them)

Tasks:

- 1. Compute key market share variables
  - Generate $s_{jt} = \frac{q_{jt}}{M_t}$, denoted as sales_share_permk_j
  - Generate $s_{0t} = 1 - \sum(s_{jt})$, denoted as sales_share_permk_0.
    - Here you will need to use bysort model_year_nw together with egen's sum() function.
  - Generate $\ln\left(\frac{s_{jt}}{s_{0t}}\right)$, denoted as ln_sales_share_permk_jnet0. This will be the key LHS variable when we estimate the multinomial Logit manually with an IV regression.
- 2. Scale some vars
  - We will scale price to 1k 2022 USD level, call it price_usd2022_1k
- 3. Generate IVs
  - In this exercise we will use classic BLP IVs. Recall the IVs are

$$z_{jt}^{\text{num}} = |\mathcal{J}_t| \qquad z_{jt}^{\text{own}} = \sum_{k \in \mathcal{J}_{ft},\, k \neq j} \mathbf{z}_{kt} \qquad z_{jt}^{\text{rival}} = \sum_{k \in \mathcal{J}_{-f,t}} \mathbf{z}_{kt}$$

no. of products in t | other products, same firm f | products by rival firms

- Denote IV1 as sumnum_all_m1. We will generate the number of products in market t minus 1 (j itself). It can be create using egen or gegen together with nunique()
- Denote IV2 as sum_ownF_otherj_weight_w.
  - It can be create using egen/gegen's sum() function at model_year_nw and make_nw level
- Denote IV3 as sum_otherFj_weight_w
  - It can be create using egen/gegen's sum() function at model_year_nw level, together with
  - egen/gegen's sum() function at model_year_nw and make_nw level

- 4. Estimate the multinomial model $\ln\left(\frac{s_{jt}}{s_{0t}}\right) = \delta_{jt} = \alpha p_{jt} + X_{jt}\beta + \phi_f + \phi_t + \eta_{jt}$
  - Set up:
    - Use reghdfe for OLS and ivreghdfe for IV
    - Endogenous variable = price_usd2022_1k
    - Exogenous attributes include dpm, hpwt_w and footprint_w, together with two AFV dummies (i_fuel_ev i_fuel_hybrid) and segment dummies (i_segsub_* and omit station wagon as omitted group).
    - Absorb make FE ϕ_f (make_nw) and ϕ_t (model_year_nw) in abs(). Note that make is string so you need to create a group dummy with egen before feeding it to abs()
    - Cluster at make level
  - Estimate OLS and IV and compare estimates especially the price coefficient. Store the results and report the results. What do you find?
  - Examine the first stage of the IV and run an F-test. Also, check the correlation amongst the IVs. What do you find?
- 5. Compare own-price demand elasticity in OLS vs IV.
  - Recall that own-price demand elasticity is $\varepsilon_{jj}^d = \alpha p_j(1 - s_j)$
  - Compute the average own-price demand elasticity in OLS vs IV. Denote them as elas_own_p_naive and elas_own_p. What do you find? Does OLS bias towards zero or away from zero? Speculate why.
  - Then study the case of IV. When do you find elasticity to be large/small? Is that super realistic?
- 6. Compute cross-price demand elasticity in IV for a particular product
  - Recall $\varepsilon_{jk}^d = -\alpha p_k s_k$
  - Consider the Toyota Gasoline Sedan. Compute the cross-price elasticity of a shock in the Toyota Gasoline Sedan on other vehicles. Study the magnitude across vehicles. Are they realistic?
- 7. Compute diversion ratios if p_j varies
  - First, compute the diversion ratio for other vehicles to the Toyota Gasoline Sedan. Recall $D_{j\to k} = \frac{s_k}{1-s_j}$
  - Next, compute the diversion ratio to the outside option. Recall $D_{j\to 0} = \frac{s_0}{1-s_j}$
  - Study the magnitude of these diversion ratios. Are they realistic?
- 8. Study a counterfactual
  - Consider the 2018 market
  - Suppose the federal government has an America First subsidy of $5k on the following makes: "Buick", "Cadillac", "Chevrolet", "Chrysler", "Dodge", "Ford", "GMC", "Jeep", "Lincoln", and "Ram". To make it simple, suppose the $5000 subsidy is by 2022 USD, i.e., consider price_usd2022 will drop by 5 for these vehicles.
  - First denote the baseline mean utility δ_{jt} as delta_mu_0.
    - Note by construction, δ_{jt} is the same as $\ln\left(\frac{s_{jt}}{s_{0t}}\right)$, aka delta_mu_0 = ln_sales_share_permk_jnet0
    - I.e., do not run a "predict delta_mu_0, xbd" after ivreghdfe, as doing so will miss the residual η_{jt} which is included in the mean utility
  - Next, compute the counterfactual mean utility δ_{jt}^c for these affected products using price coefficient _b[price_usd2022_1k]. Denote δ_{jt}^c as delta_mu_new.
  - Compute the following Logit share using $s_{jt} = \frac{\exp(\delta_{jt})}{1+\sum_k(\exp(\delta_{kt}))}$
    - Compute share_0 under baseline mean utility
    - Compute share_new under counterfactual mean utility

- Compute share_change as share_new – share_0
- The first can be created using bysort model_year_nw together with egen's sum()
- You should find sales_share_permk_j and share_0 are identical
- Take a look at the mean of "share_0 share_new share_change" using summ
  - For 2018
  - For 2018, the affected makes
  - For 2018, the unaffected makes
- Do the same for sales. Compare actual sales, new sales, and the change in sales
  - You can create them by multiplying the share by the total market size.
- What happened to the total quantity sold in the market? What does your demand estimation imply in terms of the substitution pattern if US vehicles are subsidized?

Q2. Reproduce Q1 in Python Setting

In this exercise, we will reproduce Q1 in Python's linear model. The goal is to get you familiar with a bit of Python so that Q3 will be easier.

I would first specify the library and package to import, as well as some setting.

```
import pandas as pd
import numpy as np
from linearmodels.iv import IV2SLS as lm_iv
pd.options.display.width     = 150
pd.options.display.max_columns = 50
```

Tasks:

- 1. Define your path, then load the data using pd.read_csv() and define some key variables. Here I recommend you just follow my commands.

```
# 1. Load data
product_data = pd.read_csv(mybuild_output + "Data_for_student/" + "ps8data_df_at_productlevel.csv")


# 2.  Gen Key Vars
# 2.1 Typical Vars
product_data['sales_share_permk_j'] = product_data['sales_count']   / product_data['totalmktsize']
product_data['shares']              = product_data['sales_count']   / product_data['totalmktsize']
product_data['price_usd2022_1k']    = product_data['price_usd2022'] / 1000
product_data['prices']              = product_data['price_usd2022'] / 1000
product_data['dpm']                 = product_data['dpm_usd2022']
product_data['make']                = product_data['make_nw']
product_data['year']                = product_data['model_year_nw']
product_data['market_ids']          = product_data['model_year_nw']
product_data['firm_ids']            = product_data.groupby(['make_nw']).ngroup()
product_data['product_ids']         = product_data.groupby(['make_nw', 'fuel_type_nw', 'body_style_nw']).ngroup()
product_data['clustering_ids']      = product_data['firm_ids']
```

  - While you can call the main dataframe anything, such as, df, here I call it product_data on purpose. The reason is later in Q3 we will use PyBLP and it needs the dataframe to be named in certain ways. Therefore you will see me do the following:
    - Denote the main product level data as product_data
    - Denote share s_{jt} as shares
    - Denote price as prices
    - Denote make identifiers and firm_ids
    - Denote model year as market_ids
    - Denote cluster level identifier as clustering_ids
    - Denote IVs as demand_instruments0, demand_instruments1, demand_instruments2, etc.
    - Although this exercise doesn't need us to name them this way, we will make Q2 and Q3 consistent by doing so
- 2. Generate $\ln\left(\frac{s_{jt}}{s_{0t}}\right)$.

-
```
# 2.2 Define other vars
product_data['shares_inner']        = product_data.groupby('year')['shares'].transform('sum')
product_data['shares_outside']      = 1 - product_data['shares_inner']
product_data['ln_shares_j_0']       = np.log(product_data['shares'] / product_data['shares_outside'])
```
    - This part is required if we do 2SLS manually, as we did in Stata. This part will not be needed in PyBLP (in Q3) as PyBLP will compute these in the background once you feed the program 'shares' and 'market_ids'.
- 3. Generate BLP IVs.
    - As you did in Q1, we found IV1 is quite colinear with other IVs. So here we will simply use IV2-IV3.
    -
```
product_data['sum_all_mkt']             = product_data.groupby('model_year_nw')['weight_w'].transform('sum')
product_data['sum_own_firm']            = product_data.groupby(['model_year_nw', 'make_nw'])['weight_w'].transform('sum')
product_data['sum_otherFj_weight_w']    = product_data['sum_all_mkt']  - product_data['sum_own_firm']    # rival firms
product_data['sum_ownF_otherj_weight_w'] = product_data['sum_own_firm'] - product_data['weight_w'] # same firm, other products
```
    - To make our Q2 and Q3 consistent, we will rename IVs as demand_instruments0, demand_instruments1.
- 4. Estimate the 2SLS
    - To make our replication easier. We will skip OLS and do the IV only.
    -
```
mlogit_lm_iv          = lm_iv.from_formula(formula_lm_iv, data = product_data)
mlogit_lm_iv_est_seClu = mlogit_lm_iv.fit(cov_type = "clustered", clusters = product_data['firm_ids'])
```
    - Syntax:
    - Here in the formula part, you are supposed to write in a syntax similar to lm() in R, aka 'yvar ~ 0 + control1 + control2 + … + dummies 1 + dummies 2 + … + fixed effects + [prices ~ iv1 + iv2]'
    - To make sure a constant is included, add "0+"
    - To allow linearmodel to understand FE, you can use C(.).
    - Here you can specify C(market_ids) + C(firm_ids)
    - The names of the estimated objects are not important. I call them mLogit_lm_iv and mLogit_iv_est_seClu
- 5. Access estimates
    - You can find the beta estimates in .params of the object
    - You can access all coefficients using mlogit_lm_iv_est_seClu.params
    - You can access the price coefficient using mlogit_lm_iv_est_seClu.params['prices']
- 6. Elasticity:
    - You can generate product-level own-price demand elasticity using the price coefficient multiplied by "product_data['prices'] * (1 - product_data['shares'])". Then take an average.
- 7. Report your estimates and average demand elasticity.
    - Compare to Q1. Are your estimates consistent between Q1 and Q2?

Q3. Reproduce Q1 in PyBLP

I would first specify the library and package to import, along with some settings.

```
import pandas as pd
import numpy as np
import pyblp
from scipy import stats
pd.options.display.width     = 150
pd.options.display.max_columns = 50
```

Tasks

- 1. Load the data and create the same variable as I did in Q2. Make sure you call the two IVs by the name that PyBLP prefers, aka redo Tasks 1 and 3 in Q2.
- 2. Estimate the multinomial logit
```
mlogit_pyblp_formulation      = pyblp.Formulation(formula_pyblp, absorb = formula_str_FE)
mlogit_pyblp                  = pyblp.Problem(mlogit_pyblp_formulation, product_data)
mlogit_pyblp_est              = mlogit_pyblp.solve(method = '1s', se_type = 'clustered')
```
    - Syntax:
    - In the formula part, you are supposed to write in a syntax similar to lm() in R, aka '0 + prices + control1 + control2 + … + dummies 1 + dummies 2'
    - You don't have to write 'yvar ~ ' part in the formula as this formula part is to denote what goes into the mean utility δ_{jt}

- In the absorb part, you can feed 'C(market_ids) + C(firm_ids)' to PyBLP and it will do an FWL operation.
- To make sure a constant is included, add "0 + "
- No need to do [price ~ demand_instruments0 + demand_instruments1] as we did in Q2 as PyBLP automatically take all demand_instruments* as IVs
- The names of the estimated objects are not important. I call them mlogit_pyblp and mlogit_pyblp_est

- 3. Access estimates
  - You can find the beta estimates in .beta of the object
    - For example,
    - price coefficient can be found at mlogit_pyblp_est.beta.flat[0]
    - dpm coefficient can be found at mlogit_pyblp_est.beta.flat[1]
  - All SE can be found using .beta_se
    - E.g., price coefficient SE can be found at mlogit_pyblp_est.beta_se.flat[0]
- 4. To obtain own-price elasticity:
  - Again, you can manually generate product-level own-price demand elasticity using the price coefficient multiplied by "product_data['prices'] * (1 - product_data['shares'])". Then take an average.
  - Alternatively, PyBLP has an automated feature called .compute_elasticities(). For example, you can specify mlogit_pyblp_est.compute_elasticities(name = varname, market_id = mkt). The name can be price, and market id can loop over a local called mkt over model years 2014, 2015 etc. Then take an average.
- 5. Report your estimates and average demand elasticity.
  - Compare to Q1. Are your estimates consistent between Q1 and Q3?
- 6. Produce the same counterfactual as in Q1 Part 8.

```
# 4. Counterfactual
# 4.1 Focus on 2018 market
counterfactual_market = 2018
counterfactual_data   = product_data.loc[product_data['market_ids'] == counterfactual_market].copy()


# 4.2 Define affected products
us_makes = ['Buick', 'Cadillac', 'Chevrolet', 'Chrysler', 'Dodge', 'Ford', 'GMC', 'Jeep', 'Lincoln', 'Ram']
counterfactual_data['i_affected_make'] = counterfactual_data['make_nw'].isin(us_makes).astype(int)


# 4.3 Consider counterfactual price
counterfactual_data['new_prices']   = counterfactual_data['prices']
counterfactual_data['new_prices']   = np.where(counterfactual_data['i_affected_make'] == 1,
                                               counterfactual_data['prices'] - 5,
                                               counterfactual_data['prices'])
counterfactual_data['new_shares']   = mlogit_pyblp_est.compute_shares(market_id = counterfactual_market,
                                                                      prices = counterfactual_data['new_prices'])
counterfactual_data['delta_shares'] = counterfactual_data['new_shares'] - counterfactual_data['shares']
```

  - While we can manually recompute logit shares $s_{jt} = \frac{\exp(\delta_{jt})}{1+\sum_k(\exp(\delta_{kt}))}$ for both baseline shares and counterfactual shares, PyBLP automates this part using .compute_shares() as long as you feed the counterfactual prices as new prices. PyBLP also allows you to simulate a counterfactual by changing exogenous attributes, such as the horsepower-to-weight ratio. Here we are only changing the price.
  - Then you should be able to compare shares with new shares and changes in shares, and you should also be able to do the same exercise for sales. Aka, do the following:
  - Take a look at the mean of "shares new_shares delta_shares"
    - For 2018
    - For 2018, the affected makes
    - For 2018, the unaffected makes
  - Do the same for sales. Compare actual sales, new sales, and the change in sales
    - You can create them by multiplying the share by the total market size.
  - Are your counterfactuals consistent between Q1 and Q3?
  - What happened to the total quantity sold in the market? What does your demand estimation imply in terms of the substitution pattern if US vehicles are subsidized?