Insper

DOCTORAL PROGRAM IN BUSINESS ECONOMICS

Course Name: **Empirical IO**

Assignment: **Problem Set 1**

Date of Submission: **11/11/2024**

Submitted By: **Giovanni Cavalcanti**

Exercise 1.1, Structural Econometrics

Giovanni Cavalcanti

2024-10-30

Data and model

We have a panel with 50 products dispersed over 20 geographic markets. For each product, we have its market share, 4 product characteristics, prices and 6 instrumental variables for it. The consumer utility is defined as

$$U_{ijm} = -\alpha p_{jm} + \sum_{k=1}^{3} \beta_k x_{jmk} + \beta_{4i} x_{jm4} + \xi_j + \xi_{jm} + \varepsilon_{ijm}$$

where i denotes individuals, j denotes choices and m markets. $\beta_{4i} = \beta_4 + \sigma_1 v_1$ and $v_i \sim N(0,1)$, ξ_j is a product fixed effect, ξ_{jm} denotes characteristics of product j at market m that is observed by consumers and firms (but not to the econometrician) and ε_{ijm} is iid EV across (i,j,m). We further assume that there is a outside good such that $U_{im0} = \varepsilon_{im0}$ for each m.

We load the dataset and make preliminary manipulations on it

```
dataset <- read_excel(path = "dataset_ps1.xlsx") %>%
  mutate(market = as.factor(market),
         product = as.factor(product))
```

```
## # A tibble: 6 x 14
##   market product   share    x1    x2    x3    x4   price   iv1   iv2   iv3   iv4
##   <fct>  <fct>     <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1      1       0.00541 0.529  1.15     0  1.89 0.00494  5.44  5.40  5.90  5.69
## 2 1      2       0.00115 0.494  1.28     0  1.94 0.00413  5.89  6.21  6.18  6.11
## 3 1      3       0.00131 0.468  1.46     0  1.72 0.00590  7.89  7.48  7.77  7.15
## 4 1      4       0.00133 0.427  1.61     0  1.69 0.00541  7.58  7.33  7.01  7.60
## 5 1      5       0.00314 0.452  1.65     0  1.50 0.00820  8.93  9.58  8.93  8.95
## 6 1      6       0.00305 0.451  1.62     0  1.73 0.00616  7.79  7.22  7.37  7.77
## # i 2 more variables: iv5 <dbl>, iv6 <dbl>
```

Problem 1: Multinomial Logit

Assume that $\sigma_1 = 0$, and $\varepsilon_{ijm} \sim TIEV(\mu, \theta)$, where $\mu > 0$ is the scale parameter and θ is the location parameter of the distribution, i.e. its distribution is $F_\varepsilon(x) = exp(-exp(-\mu(x-\theta)))$

1.1 Derive the share of each choice in each market.

The probability that a consumer i in market m chooses product j is given by:

$$P(a_{im} = j) = P\left(U_{ijm} > U_{ikm},\ \forall k \neq j\right),$$

where U_{ijm} is the utility of product j for consumer i in market m.

Given the utility specification $U_{ijm} = V_{ijm} + \varepsilon_{ijm}$ with $V_{ijm} = \sum_{l=1}^{3} \beta_l x_{jml} + \beta_{4i} x_{jm4} - \alpha p_{jm}$, we assume that $\varepsilon_{ijm} \sim$ i.i.d. Type I Extreme Value.

Using the properties of the Extreme Value distribution, the choice probability for product j becomes:

$$P(a_{im} = j) = \frac{\exp(V_{ijm})}{\sum_k \exp(V_{ikm})}.$$

This implies that the market share s_{jm} of product j in market m is:

$$s_{jm} = \frac{\exp(V_{jm})}{\sum_k \exp(V_{km})}.$$

Since we already observe the market share for each product, we only need to calculate the share for the outside option and add it to the dataset. The outside option's share represents the probability of consumers choosing none of the observed products in each market, defined as $1 - \sum_j \text{share}_j$ for each market.

The following code calculates this share and appends it to the dataset:

```r
# Calculate outside good share and add outside good
outside_goods <- dataset %>%
  group_by(market) %>%
  summarize(
    product = as.factor("0"),  # Label for outside good
    share = 1 - sum(share),  # Calculated outside share
    x1 = 0, x2 = 0, x3 = 0, x4 = 0,  # Characteristics set to 0
    price = 0,  # Price of outside good is 0
    iv1 = 0, iv2 = 0, iv3 = 0, iv4 = 0, iv5 = 0, iv6 = 0  # Instrument variables set to 0
  )

# Bind the outside good rows to the original dataset and arrange by market
dataset <- bind_rows(dataset, outside_goods) %>%
  arrange(market)
```

To preview the shares for the outside option across markets:

```
## # A tibble: 6 x 3
##   market product share
##   <fct>  <fct>   <dbl>
## 1 1      0       0.433
## 2 2      0       0.455
## 3 3      0       0.436
## 4 4      0       0.463
## 5 5      0       0.437
## 6 6      0       0.427
```

1.2 Is μ identified? Why? And what about θ?

In general, the scale parameter μ cannot be identified. Suppose that $\varepsilon_{ij} \sim T1EV(\mu_j, \theta_j)$ and $\varepsilon_{ik} \sim T1EV(\mu_k, \theta_k)$ represent the unobserved idiosyncratic utility shocks, where we assume orthogonality between these terms. When we take the difference $\varepsilon_{ik} - \varepsilon_{ij}$, it yields a logistic distribution that depends only on the difference in location parameters $\theta_j - \theta_k$ and not on the scale μ:

$$\varepsilon_{ik} - \varepsilon_{ij} \sim \text{Logistic}(\theta_j - \theta_k).$$

This difference reflects the general case of choosing product j over any other product k in the market, represented by:

$$P(a_i = k) = \text{Prob}\left(\varepsilon_{ik} - \varepsilon_{ij} < V_{ij} - V_{ik}, \forall k \neq j\right).$$

Thus, μ cannot be directly identified from this structure since it cancels out in the difference.

Regarding θ, the location parameter, it is only partially identifiable. By anchoring θ to an outside option with an observed utility level of zero, we set a baseline against which the utility levels of all other options are identified. This normalization allows us to interpret and identify the utilities of other products relative to the outside good, assuming its utility level is fixed at zero.

1.3 Suppose that $\mu = 1, \theta = 0$. Suppose that you want to estimate the model using OLS.

a) Derive the equation you will use to estimate the model.

We first start with the closed logistic formula for the share in each market

$$P(a_i = j | V_{i1}, \dots, V_{i50}) = \frac{exp(V_j)}{\sum_{k=1}^{50} exp(V_k)}$$

Where V_k is the vector of observed characteristics and price. If we divide each probability, wich is the observed market share, the denominator cancels out, yielding

$$\frac{P(a_i = j | V_{i1}, \dots, V_{i50})}{P(a_i = 0 | V_{i1}, \dots, V_{i50})} = \frac{exp(V_j)}{exp(V_0)}$$

Taking the logs ob both sides, we linearize it

$$ln(P(a_i = j|\cdot)) - ln(P(a_i = 0|\cdot)) = V_j - V_0$$

Therefore, out estimating equation for this aggregated model is

$$ln(\hat{P}(a_i = j|\cdot)) - ln(\hat{P}(a_i = 0|\cdot)) = \sum_{l=1}^{L} \beta_l (x_{jl} - x_{ol}) + (\xi_j - \xi_o)$$

$$ln(\hat{P}(a_i = j|\cdot)) - ln(\hat{P}(a_i = 0|\cdot)) = \sum_{l=1}^{L} \beta_l (x_{jl} - x_{ol}) + (\xi_j^*)$$

b) Under what conditions the OLS estimator is consistent?

The model parameters β can only be consistently estimated if the vectors of observed characteristics V_j contains no unknowns or unobservables, as to have no endogeneity problem in the estimation.

1.4 Assume that $\xi_j = 0$. Estimate the parameters of the model by OLS and IV. Compare the estimates of α. Does the OLS bias have the expected sign? Explain.

The following code fits the two requested models

Table 1: Comparison of OLS and IV Model Results with Clustered SEs

	OLS Model	IV Model
(Intercept)	-7.357***	-7.424***
	(0.471)	(0.472)
x1	0.350	0.778*
	(0.391)	(0.387)
x2	1.623***	1.664***
	(0.242)	(0.242)
x3	0.581***	0.797***
	(0.098)	(0.104)
x4	0.329**	0.326**
	(0.117)	(0.117)
price	-50.024***	-70.161***
	(6.157)	(7.250)
Num.Obs.	1000	1000
R2	0.089	0.084
R2 Adj.	0.084	0.079
AIC	2994.3	2999.8
BIC	3028.7	3034.1
Log.Lik.	-1490.160	
RMSE	1.07	1.08
Std.Errors	Custom	Custom

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

```r
# Prepare the dataset with log difference in shares
dataset_with_outside <- dataset %>%
  left_join(dataset %>%
              filter(product == "0") %>%
              select(market, share) %>%
              rename(outside_share = share),
            by = "market") %>%
  filter(product != "0") %>%  # Exclude the outside good
  mutate(log_diff_share = log(share) - log(outside_share))

# Fit OLS and IV models
ols_model <- lm(log_diff_share ~ x1 + x2 + x3 + x4 + price, data = dataset_with_outside)
iv_model <- ivreg(log_diff_share ~ x1 + x2 + x3 + x4 + price |
                    . - price + iv1 + iv2 + iv3 + iv4 + iv5 + iv6,
                  data = dataset_with_outside)

# Clustered standard errors by product
ols_se_clustered <- vcovCL(ols_model, cluster = ~product)
iv_se_clustered <- vcovCL(iv_model, cluster = ~product)
```

Controlling for the instruments given yields an α estimand with higher magnitude, this is expected as the endogeneity problem would yield an upward bias

1.5 Based on the IV parameters estimated above compute own- and cross-price elasticities for each good in market one. Discuss the results

The following code calculates the elasticities for market 1

```r
# Select data for market 1
market_data <- dataset %>% filter(market == 1, product != 0)
```

```r
# Extract coefficients from the model
alpha_price <- iv_model$coefficients["price"]  # Coefficient for price

# Calculate Own-Price Elasticities
market_data <- market_data %>%
  mutate(own_price_elasticity = price * alpha_price * (1 - share))

# Calculate Cross-Price Elasticities
# Create a dataframe for cross-price elasticities
cross_price_elasticities <- expand.grid(j = market_data$product, k = market_data$product)
cross_price_elasticities <- merge(cross_price_elasticities, market_data[, c("product", "price", "share"
cross_price_elasticities <- merge(cross_price_elasticities, market_data[, c("product", "price", "share"

# Calculate cross-price elasticities
cross_price_elasticities <- cross_price_elasticities %>%
  mutate(cross_price_elasticity = -price_k * alpha_price * share_k)
```

The first results for the elasticity matrix are

```r
# Create a full elasticity matrix
# Initialize a matrix with NA values
num_products <- nrow(market_data)
elasticity_matrix <- matrix(NA, nrow = num_products, ncol = num_products)

# Fill the diagonal with own-price elasticities
rownames(elasticity_matrix) <- market_data$product
colnames(elasticity_matrix) <- market_data$product

# Fill in cross-price elasticities
for (row in 1:nrow(cross_price_elasticities)) {
  j <- as.character(cross_price_elasticities$j[row])
  k <- as.character(cross_price_elasticities$k[row])
  elasticity_matrix[j, k] <- cross_price_elasticities$cross_price_elasticity[row]
}

for (i in 1:num_products) {
  elasticity_matrix[i, i] <- market_data$own_price_elasticity[i]
}

# Print the elasticity matrix
print(elasticity_matrix[1:10,1:10])
```

The elasticity matrix results align with theoretical expectations: own-price elasticities are negative, indicating that an increase in the price of a product leads to a decrease in its demand. Additionally, the cross-price elasticities exhibit the Independence of Irrelevant Alternatives (IIA) property, where a change in the price of one product affects the demand for all other products uniformly. This outcome reinforces the suitability of the multinomial logit model for this analysis, albeit with the known limitation that IIA may oversimplify substitution patterns among products.

	1	2	3	4	5	6	7	8	9	10
1	-0.3444	0.0003	0.0005	0.0005	0.0018	0.0013	0.0033	0.0630	0.0167	0.0236
2	0.0019	-0.2895	0.0005	0.0005	0.0018	0.0013	0.0033	0.0630	0.0167	0.0236
3	0.0019	0.0003	-0.4133	0.0005	0.0018	0.0013	0.0033	0.0630	0.0167	0.0236
4	0.0019	0.0003	0.0005	-0.3788	0.0018	0.0013	0.0033	0.0630	0.0167	0.0236
5	0.0019	0.0003	0.0005	0.0005	-0.5736	0.0013	0.0033	0.0630	0.0167	0.0236
6	0.0019	0.0003	0.0005	0.0005	0.0018	-0.4308	0.0033	0.0630	0.0167	0.0236
7	0.0019	0.0003	0.0005	0.0005	0.0018	0.0013	-0.6111	0.0630	0.0167	0.0236
8	0.0019	0.0003	0.0005	0.0005	0.0018	0.0013	0.0033	-0.6786	0.0167	0.0236
9	0.0019	0.0003	0.0005	0.0005	0.0018	0.0013	0.0033	0.0630	-0.7500	0.0236
10	0.0019	0.0003	0.0005	0.0005	0.0018	0.0013	0.0033	0.0630	0.0167	-0.8403

Table 2: Elasticity Matrix for the first 10 goods

Exercise 1.2, Structural Econometrics

Giovanni Cavalcanti

2024-11-06

Problem 2: Random Coefficients Logit

Assume that $\sigma_1 \neq 0$ and $\epsilon_{ijm} \sim T1EV(1,0)$. Assume that prices are correlated with ξ_{jm}.

2.1 Derive the choice probabilities.

We can rewrite the utility function of individual i as:

$$U_{ijm} = V_{ijm} + \epsilon_{ijm} \quad ; \quad V_{ij} = (\delta_{jm} + \mu_{ijm} + \xi_j + \xi_{jm})$$

where $\delta_{jm} = -\alpha p_{jm} + \sum_{k=1}^{4} \beta_k x_{jmk} + \xi_{jm} \quad , \quad \mu_{ijm} = \sigma_1 v_i x_{jm4}$

μ_{ijm} captures correlation across choices, this correlation depends on characteristics t and unobserved taste shocks ν. So, at any market m, the probability of choosing j is given by

$$A_{ijm} = \{(\epsilon_i, \beta_i) : V_{ijm} + \epsilon_{im} \geq V_{ikm} + \epsilon_{ikm} \; \forall \; k \neq j\}$$
$$P(a_{im} = j) = P(V_{ijm} + \epsilon_{jm} \geq V_{ikm} + \epsilon_{km} \; \forall \; k \neq j)$$

Therefore, using the assumption that $\epsilon_i = (\epsilon_1, \ldots, \epsilon_{50})$ is i.i.d. across J, and the β_i vary over i with population density $f(\beta|\theta)$,

$$\begin{aligned} P(a_{im} = j) &= \int_{A_{ijm}} f((\epsilon_i, \beta_i))\, d(\epsilon_i, \beta_i) \\ &= \int_{\beta_i} f(\beta_i|\theta) \left(\int_{A_{ijm}} f(\epsilon_i)\, d\epsilon_i \right) d\beta_i \\ &= \int_{\beta_i} \frac{\exp(\beta_i' x_j)}{\sum_k \exp(\beta_i' x_k)} f(\beta_i|\theta)\, d\beta_i \end{aligned}$$

2.2 Assume for simplicity that $\xi_j = 0$. Estimate the parameters of the model (including σ_1 and a constant) using a consistent estimator.

First, we load the original dataset for further manipulation.

```r
dataset <- read_excel(path = "dataset_ps1.xlsx") %>%
  mutate(market = as.factor(market),
         product = as.factor(product))
```

And for our starting guesses for the parameters, we will use the IV estimates from question 1.

The following code manually implements the BLP estimation procedure, using the gmm package. To do so, a gmm_fn(.) function is defined based on the 3 step procedure described in class.

```r
# Add the outside option
outside_option <- dataset %>%
  group_by(market) %>%
  summarize(share = 1 - sum(share), .groups = 'drop') %>%
  mutate(product = "outside",
         x1 = 0, x2 = 0, x3 = 0, x4 = 0, price = 0,
         iv1 = 0, iv2 = 0, iv3 = 0, iv4 = 0, iv5 = 0, iv6 = 0)

# Combine the original dataset with the outside option
dataset_with_outside <- bind_rows(dataset, outside_option) %>%
  mutate(product = as.factor(product))

# Load initial parameter guesses from iv_model coefficient
intercept <- iv_model$coefficients["(Intercept)"]
alpha <- iv_model$coefficients["price"]
beta <- iv_model$coefficients[c("x1", "x2", "x3", "x4")]
sigma <- 0.1  # Initial value for sigma

# Set up initial parameters for GMM estimation
initial_params <- c(intercept = intercept, alpha = alpha, beta = beta, sigma = sigma)

set.seed(123)  # For reproducibility

# Set up constants
num_consumers <- 500
tolerance <- 1e-10  # Set a convergence tolerance
max_iterations <- 10000  # Maximum number of iterations

# Generate consumer taste shocks
taste_shocks <- dataset_with_outside %>%
  distinct(market) %>%
  mutate(consumer_id = list(1:num_consumers)) %>%
  unnest(consumer_id) %>%
  mutate(v_i = rnorm(n()))

# Expand dataset with product-consumer-market combinations
expanded_dataset <- crossing(
  dataset_with_outside %>% select(market, product),
  consumer_id = unique(taste_shocks$consumer_id)
) %>%
  left_join(taste_shocks, by = c("market", "consumer_id")) %>%
  left_join(dataset_with_outside, by = c("market", "product"))

# Define the GMM function
gmm_fn <- function(params, data, max_iterations = 10000, tolerance = 1e-6) {

  # Extract parameters
```

```r
intercept <- params["intercept.(Intercept)"]
alpha <- params["alpha.price"]
beta <- params[grep("^beta", names(params))]
sigma <- params["sigma"]

# Initialize delta based on the current parameter values
data <- data %>%
  mutate(delta = intercept + alpha * price + beta[1] * x1 + beta[2] * x2 + beta[3] * x3 + beta[4] * x
         mu_i = sigma * v_i * x4)

# Main iteration loop to update delta
for (iteration in 1:max_iterations) {

  # Calculate expected utility
  data <- data %>%
    mutate(exp_utility = exp(delta + mu_i)) %>%
    group_by(market, consumer_id) %>%
    mutate(total_exp_utility = sum(exp_utility)) %>%
    ungroup() %>%
    mutate(div = exp_utility / total_exp_utility) %>%
    group_by(market, product) %>%
    mutate(s_jm = sum(div) / num_consumers) %>%
    ungroup()

  # Calculate the new delta values
  data <- data %>%
    mutate(delta_new = delta + log(share) - log(s_jm))  # Update delta

  # Check for NA values in delta_new and delta
  if (any(is.na(data$delta_new))) {
    cat("NA detected in delta_new values. Exiting iteration.\n")
    break
  }

  # Calculate the maximum difference for convergence check
  delta_diff <- max(abs(data$delta_new - data$delta), na.rm = TRUE)

  # Check for convergence
  if (delta_diff < tolerance) {
    # cat("Converged in", iteration, "iterations.\n")
    # cat("Converged with", params, "\n")
    break  # Exit the loop if converged
  }

  # Update delta for the next iteration
  data$delta <- data$delta_new
}

# Define the matrix of instruments (Z)
Z <- as.matrix(data[, c("iv1", "iv2", "iv3", "iv4", "iv5", "iv6")])
```

```r
  # Compute residuals based on the final delta
  X <- as.matrix(data[, c("price", "x1", "x2", "x3", "x4")])
  residuals <- data$delta - X %*% c(alpha, beta)

  # Calculate the moment conditions
  moments <- c(residuals) * Z

  # Return the average of the moments across observations
  return(colMeans(moments))
}

# Perform GMM estimation
gmm_results <- gmm(
  g = gmm_fn,
  x = expanded_dataset,
  t0 = initial_params,
  vcov = 'HAC',
  method = 'Nelder-Mead',
  control = list(reltol = 1e-25, maxit = 100000)
)
```

```
## Warning in FinRes.baseGmm.res(z, Model_info): The covariance matrix of the
## coefficients is singular
```

The model results are

```
## intercept.(Intercept)           alpha.price               beta.x1
##           -2.88347845          -71.16020851            1.47127215
##               beta.x2               beta.x3               beta.x4
##            1.68081209            2.15183723            4.72792619
##                 sigma
##            0.02315001
```

2.3 Compute own- and cross-price elasticities for each good in market one. Compare with the elasticities obtained in 1.5.

```r
# Constants
alpha <- coefs['alpha.price']
beta <- coefs[3:6]
sigma <- coefs['sigma']

# Filter data for market 1
market_data <- expanded_dataset %>%
  filter(market == 1, product != "outside")

# Calculate individual choice probabilities s_ij for each product and consumer in market 1
market_data <- market_data %>%
  group_by(consumer_id) %>%
  mutate(
    delta = intercept + alpha * price + beta[1] * x1 + beta[2] * x2 + beta[3] * x3 + beta[4] * x4,
    mu = sigma * v_i * x4,
    exp_delta_mu = exp(delta + mu),
    total_exp_utility = sum(exp_delta_mu),
    s_ij = exp_delta_mu / total_exp_utility
  ) %>%
  ungroup()

# Calculate average market share for each product (s_j)
average_shares <- market_data %>%
  group_by(product) %>%
  summarize(s_j = mean(s_ij))

# Sort products in ascending order
products <- sort(as.numeric(unique(market_data$product)))

# Initialize an empty matrix to store elasticities
elasticity_matrix <- matrix(0, nrow = length(products), ncol = length(products),
                            dimnames = list(products, products))

# Calculate own- and cross-price elasticities
for (j in products) {
  # Filter data for product j
  product_data_j <- market_data %>% filter(product == j)
  s_j <- average_shares %>% filter(product == j) %>% pull(s_j)
  p_j <- product_data_j$price[1]  # Assuming price is constant across consumers for each product

  # Calculate own-price elasticity for product j
  elasticity_matrix[j, j] <- mean((p_j / s_j) * alpha * product_data_j$s_ij * (1 - product_data_j$s_ij))

  # Calculate cross-price elasticities for product j with respect to other products k
  for (k in products[products != j]) {
    product_data_k <- market_data %>% filter(product == k)
    p_k <- product_data_k$price[1]  # Price of product k
    elasticity_matrix[j, k] <- -mean((p_k / s_j) * alpha * product_data_j$s_ij * product_data_k$s_ij)
  }
}
```

Comparing the new results to those observed in 1.5, the IAA is not valid anymore.

	1	2	3	4	5	6	7	8	9	10
1	-0.347177198	0.005299763	0.003086103	0.003082090	0.001784594	0.004257424	0.007630117	0.093878490	0.052453340	0.033284590
2	0.004048745	-0.288637041	0.003086080	0.003082065	0.001784572	0.004257394	0.007630063	0.093878900	0.052453390	0.033284560
3	0.004048802	0.005299800	-0.416630548	0.003082177	0.001784673	0.004257530	0.007630307	0.093877040	0.052453160	0.033284700
4	0.004048810	0.005299806	0.003086196	-0.381573431	0.001784686	0.004257548	0.007630339	0.093876790	0.052453120	0.033284720
5	0.004048858	0.005299845	0.003086282	0.003082286	-0.581821559	0.004257663	0.007630543	0.093875240	0.052452930	0.033284830
6	0.004048799	0.005299798	0.003086178	0.003082172	0.001784668	-0.434034241	0.007630296	0.093877120	0.052453170	0.033284690
7	0.004048799	0.005299797	0.003086178	0.003082172	0.001784668	0.004257524	-0.615454194	0.093877130	0.052453170	0.033284690
8	0.004048677	0.005299697	0.003085960	0.003081933	0.001784453	0.004257233	0.007629776	-0.658280020	0.052453670	0.033284390
9	0.004048713	0.005299727	0.003086024	0.003082004	0.001784516	0.004257319	0.007629930	0.093879910	-0.725158920	0.033284480
10	0.004048734	0.005299744	0.003086061	0.003082043	0.001784552	0.004257368	0.007630016	0.093879250	0.052453430	-0.842938800

Table 1: Elasticity Matrix for first 10 goods in market 1