

Economics 631 IO - Fall 2019

Problem Set 2

Nathan Mather and Tyler Radler

October 27, 2019

1 BLP - Random Coefficient

Preliminaries

Each firm chooses price to solve the problem

$$\max_{p_j} (p_j - mc_j) M s_j(\mathbf{p}, \mathbf{x}, \sigma)$$

The FOC is

$$0 = (p_j - mc_j) M \frac{\partial s_j}{\partial p_j} + M s_j$$

and so the price will be determined by the following condition:

$$p_j = mc_j - s_j \left(\frac{\partial s_j}{\partial p_j} \right)^{-1}$$

. The market share for product j is given by

$$s_j(\mathbf{p}, \mathbf{x}, \theta) = \int \frac{\exp(\beta_i x_j - \alpha p_j)}{1 + \sum_{j'} \exp(\beta_i x_{j'} - \alpha p_{j'})} dF(\beta_i)$$

Given our functional form assumptions we can rewrite $\beta_i = \beta + \sigma v_i$ where β is the mean of the distribution and σ is the standard deviation, $v_i \sim \mathcal{N}(0, 1)$. Additionally we can define the mean utility of purchasing product j as $\delta_j = \beta x_j - \alpha p_j$ and rewrite the market share expression in terms of δ_j and σ .

$$s_j(\mathbf{p}, \mathbf{x}, \boldsymbol{\delta}, \sigma) = \int \frac{\exp(\delta_j + \sigma x_j v_i)}{1 + \sum_{j'} \exp(\delta_{j'} + \sigma x_{j'} v_i)} dF(v)$$

Note that if we rewrite the above expression as

$$s_j(\mathbf{p}, \mathbf{x}, \boldsymbol{\delta}, \sigma) = \int \tilde{s}_j(\mathbf{p}, \mathbf{x}, \boldsymbol{\delta}, \sigma) dF(v_i)$$

we can get a fairly-nice expression for the own-price derivative with respect to the price:

$$\frac{\partial s_j}{\partial p_j} = \int (-\alpha) \frac{\partial \tilde{s}_j}{\partial \delta_j} dF(v) = -\alpha \int \tilde{s}_j (1 - \tilde{s}_j) dF(v)$$

where the last equality is due to properties of the logit error. Thus our final price condition is

$$p_j = mc_j - \int \tilde{s}_j dF(v) [-\alpha \int \tilde{s}_j (1 - \tilde{s}_j) dF(v)]^{-1}$$

2 Q1

We are given that $\alpha = 1, \beta = 1, \sigma = 1, x_1 = 1, x_2 = 2, x_3 = 3$, and $mc_j = x_j$.
The price vector is:

Q1 Prices	
variable	value
p1	2.17
p2	3.28
p3	4.85

write write write

3 Q2

Now $\alpha = .5, \beta = .5, \sigma = .5, x_1 = 1, x_2 = 2, x_3 = 3$, and $mc_j = x_j$. The price vector is:

Q2 Prices	
variable	value
p1	3.36
p2	4.45
p3	5.84

write write write. It is different because blah.

4 Q3

After the merger the profit maximization problem for the new firm is

$$\max_{p_1, p_2} (p_1 - mc_1)s_1(\mathbf{p}, \mathbf{x}, \sigma) + (p_2 - mc_2)s_2(\mathbf{p}, \mathbf{x}, \sigma)$$

The FOC for p_1 is

$$0 = s_1 + (p_1 - mc_1)\frac{\partial s_1}{\partial p_1} + (p_2 - mc_2)\frac{\partial s_2}{\partial p_1}$$

and so the price will be determined by the following condition:

$$p_1 = mc_1 - (s_1 + (p_2 - mc_2)\frac{\partial s_2}{\partial p_1})(\frac{\partial s_1}{\partial p_1})^{-1}$$

Note that using our previous notation,

$$\frac{\partial s_2}{\partial p_1} = -\alpha \int \tilde{s}_1 \tilde{s}_2 dF(v)$$

The FOC for p_2 is symmetric to that of p_1 , and thus the optimal p_1 and p_2 are given by

$$p_1 = mc_1 - [\int \tilde{s}_1 dF(v) + (p_2 - mc_2)(-\alpha \int \tilde{s}_1 \tilde{s}_2 dF(v))][-\alpha \int \tilde{s}_1(1 - \tilde{s}_1) dF(v)]^{-1}$$

$$p_2 = mc_2 - [\int \tilde{s}_2 dF(v) + (p_1 - mc_1)(-\alpha \int \tilde{s}_2 \tilde{s}_1 dF(v))][-\alpha \int \tilde{s}_2(1 - \tilde{s}_2) dF(v)]^{-1}$$

As firm 3 has not merged it's optimal price condition is the same:

$$p_3 = mc_3 - \int \tilde{s}_3 dF(v)[-\alpha \int \tilde{s}_3(1 - \tilde{s}_3) dF(v)]^{-1}$$

The price vector from the simulation is:

Q3 Prices

variable	value
p1	3.76
p2	4.83
p3	5.91

5 Q4

The change in consumer surplus is given by the compensating variation, which we can calculate as follows:

$$CV_i = \frac{\log(\sum_j \exp(V_{ij}^{old}) - (\sum_j \exp(V_{ij}^{new}))}{\alpha}$$

where $V_{ij} = \beta_i x_j - \alpha p_j$. The total consumer surplus is found by

$$CV = M * \int CV_i dF(v)$$

where M is the number of people in the market. The change in producer surplus is just the change in profits, and is given by

$$\Delta\pi = \sum_j (p_j^{new} - mc_j^{new}) Ms_j^{new} - (p_j^{olds} - mc_j^{old}) Ms_j^{old}$$

and the total change in welfare is

$$\Delta\text{Surplus} = \Delta\pi - CV = M[(\sum_j (p_j^{new} - mc_j^{new}) s_j^{new} - (p_j^{olds} - mc_j^{old}) s_j^{old}) - \int CV_i dF(v)]$$

The change in surplus when we normalize $M = 1$ is

Q4 Surplus Results

variable	value
Change In Cosumer Surplus Per Person	-0.29
Change In Producer Surplus Per Person	0.05
Change In Total Surplus Per Person	-0.24

insert ish here

6 Q5

If we allow the merging firm's marginal costs to decrease from x_j to $\frac{x_j}{2}$ we get the following pricing equilibrium and change in consumer, producer and total surplus:

Q5 Prices

variable	value
p1	3.91
p2	4.89
p3	5.92

Q5 Surplus Results

variable	value
Change In Cosumer Surplus Per Person	-0.36
Change In Producer Surplus Per Person	0.25
Change In Total Surplus Per Person	-0.11

insert ish here

7 Appendix

7.1 R Code

pset 2 631

```
#####  
# ==== pset 2 ====  
#####  
  
require(data.table)  
require(Matrix)  
library(xtable)  
  
# clear objects  
rm(list = ls(pos = ".GlobalEnv"), pos = ".GlobalEnv")  
options(scipen = 999)  
cat("\f")  
  
#set #note output location  
f_out <- "c:/Users/Nmath_000/Documents/Code/Econ_631/ps2/"  
  
#set #note option to save output  
opt_save <- TRUE  
  
#####  
# ==== Question 1 ====  
#####  
  
# set parameters  
x1 = 1  
x2 = 2  
x3 = 3  
n.sim = 10000  
  
# Function to compute shares for a given mean and random utility  
share_f <- function(delta.in, mu.in, opt_tidle = FALSE){  
  
  # get the numerator by exp(delta + xi*vi*sigma)  
  numer <- exp(mu.in) * matrix(rep(exp(delta.in), n.sim), ncol = n.sim)  
  
  # get the denominator by summing over all numerators and adding one  
  denom_i <- matrix(rep(1 + colSums(numer), 3), nrow = 1, ncol = n.sim)  
  
  # then replicated this three times so we can divide (probs better way to do this )  
  denom <- rbind(denom_i, denom_i, denom_i)  
  
  if(opt_tidle){  
  
    return(numer / denom)  
  
  }else{  
    # the shares are the mean of numerator/denominator accross simulations  
    shares <- rowMeans(numer / denom)  
  
    return(shares)  
  }  
}
```

```

}
}

# Function to compute the derivative of your own shares wrt own-good mean utility
#note we can just use output of shares function as input here
dSharedOwnP_f <- function(shares.in, alpha.in){

  # dS.i/dP.i is -alpha*share*(1-share)
  dSharedOwnP <- rowMeans(-alpha.in*shares.in*(1-shares.in))

  return(dSharedOwnP)
}

#note switcing this to take shares as the input so we dont recalculate it
dSharedOtherP_f <- function(shares.in, alpha.in){

  # Just using shares output from other function
  # share.i <- matrix(shares.in)

  # dS.i/dDelta.j is integral of -s.i*s.j
  sisj.matrix <- -shares.in%*%t(shares.in)/ncol(shares.in)

  # dS.i/dP.j is -alpha*dS.i/dDelta.j
  #note I don't understand why we are only grabbing 1,2
  dSharedOtherP <- -alpha.in*sisj.matrix[1,2]
  return(dSharedOtherP)
}

# create data.table of xs
xi <- as.matrix(c(x1,x2,x3))

# make simulation matrix
v = matrix(rnorm(1 * n.sim), nrow = 1, ncol = n.sim)

# Now we will guess the price to start and calcualte everything
# fill in an initial price guess to work through functions
# price in iteration k
p.init <- matrix(c(2, 3, 4))

tol <- 10^-10

p_solver <- function(beta.in, alpha.in, sigma.in, xi.in, mc.in, p.guess){
  #####
  # ==== Inside the loop ====
  #####
  i <- 1

  # Initial guess
  p.old <- matrix(c(0, 0, 0))

```

```

while (sum(abs(p.guess - p.old)) > tol)
{
  print(paste0("Iteration:", i, ", Difference:", sum(abs(p.guess - p.old))))

  p.old <- p.guess

  # using the guess, calculate deltas
  delta <- xi.in*beta.in - alpha.in*p.guess

  # calculate x times sigma times v
  mu <- xi.in%*%v*sigma.in

  # Calculate shares and derivative
  shares <- as.matrix(share_f(delta, mu))
  shares_tilde <- share_f(delta, mu, opt_title = TRUE)

  dSharedOwnP <- as.matrix(dSharedOwnP_f(shares_tilde, alpha.in))

  # using the shares and derivative, calculate the equilibrium price
  p.guess <- mc.in - shares*(dSharedOwnP)^-1

  i <- i + 1
}
p.final <- p.guess
return(p.final)
}

# get answer for question 1
p_q1 <- p_solver(1, 1, 1, xi, mc.in = xi, p.init)

#####
# ==== question 2 ====
#####

p_q2 <- p_solver(.5, .5, .5, xi, mc.in = xi, p.init)

#####
# ==== Question 3 ====
#####

p_postmerge_solver <- function(beta.in, alpha.in, sigma.in, xi.in, mc.in, p.guess){
  #####
  # ==== Inside the loop ====
  #####
  i <- 1
  p.old <- matrix(c(0, 0, 0))
  while (sum(abs(p.guess - p.old)) > tol)
  {

```

```

print(paste0("Iteration:", i, ", Difference:", sum(abs(p.guess - p.old))))

p.old <- p.guess

# using the guess, calculate deltas
delta <- xi.in*beta.in - alpha.in*p.guess

# calculate x times sigma times v
mu <- xi.in%*%v*sigma.in

# You care about the markup of the other product you own, so create a variable for 2's markup for 1
markup <- p.guess - mc.in

# Definitely a better way to do this...
othergood.markup <- rbind(markup[2,1], markup[1, 1], 0)

# Calculate shares, own price elasticities
shares <- as.matrix(share_f(delta, mu))
shares_tilde <- share_f(delta, mu, opt_title = TRUE)
dSharesdOwnP <- as.matrix(dSharedOwnP_f(shares_tilde, alpha.in))

# Calculate price elasticities wrt the other product we care about.
#note: would rather use an ownership matrix somehow but yolo
dSharesdOtherP <- as.matrix(c(dSharedOtherP_f(shares_tilde, alpha.in), dSharedOtherP_f(shares_tilde

# using the shares and derivative, calculate the equilibrium price
p.guess <- xi.in - (shares + othergood.markup*dSharesdOtherP)*(dSharesdOwnP)^-1

  i <- i + 1
}

p.final <- p.guess

return(p.final)
}

p_q3 <- p_postmerge_solver(.5, .5, .5, xi,mc.in = xi, matrix(c(2, 3, 4)))

#####
# ==== question 4 ====
#####

#####
# ==== change in consumer surplus ====
#####

# define variables for debug
v.in = v

```



```

pv = p_q2

# function for getting sum of value funcitons
vi_f <- function(v.in, pv, xi.in, beta.in, alpha.in, sigma.in){

  # make beta_i
  beta_i <- beta.in + sigma.in*v.in

  # get beta_i times xs
  #note this is old. It does not doe the exponent. Can delet when we are sure it is wrong
  # Vi <- colSums( xi.in %*% beta_i ) - colSums(alpha*pv )
  Vi <- colSums( exp(xi.in %*% beta_i - matrix(rep(alpha.in*pv, ncol(beta_i)), ncol = ncol(beta_i))) )

  return(Vi)
}

# #note temp define these for deubg
# pv_pre = p_q2
# pv_post = p_q3

# NOW write funciton to get cv_i
cv_i_f <- function(v.in, pv_pre, pv_post, xi.in, beta.in, alpha.in, sigma.in ){

  # get vi for pre
  vi_pre <- vi_f(v.in, pv = pv_pre, xi.in, beta.in, alpha.in, sigma.in)

  # et vi for post
  vi_post <- vi_f(v.in, pv = pv_post, xi.in, beta.in, alpha.in, sigma.in)

  # get cv_i
  cv_i <- (log(vi_post) - log(vi_pre))/-alpha.in

  return(cv_i)
}

# run it with correct values
cv_i <- cv_i_f(v.in      = v,
               pv_pre    = p_q2,
               pv_post   = p_q3,
               xi.in     = xi,
               beta.in   = .5,
               alpha.in  = .5,
               sigma.in  = .5)

# now get mean cv
mean_cv <- mean(cv_i)

#####
# ==== Change in producer surplus ====
#####

```

```

# # for debug
# pv      = p_q2
# mc_v     = xi
# v.in     = v
# xi.in    = xi
# alpha.in = .5
# beta.in  = .5
# sigma.in = .5
profit_f <- function(pv,mc_v, v.in, alpha.in, beta.in, xi.in, sigma.in){

  # using the guess, calculate deltas
  delta <- xi.in*beta.in - alpha.in*pv

  # calculate x times sigma times v
  mu <- xi.in**v.in*sigma.in

  shares <- share_f(delta, mu)
  # get profits before and after
  profits <- (pv - mc_v)*shares

  return(profits)
}

# get profits before
profits_before <- profit_f(pv      = p_q2,
                           mc_v    = xi,
                           v.in    = v,
                           xi.in   = xi,
                           alpha.in = .5,
                           beta.in  = .5,
                           sigma.in = .5)

# get profits after
profits_after <- profit_f(pv      = p_q3,
                          mc_v    = xi,
                          v.in    = v,
                          xi.in   = xi,
                          alpha.in = .5,
                          beta.in  = .5,
                          sigma.in = .5)

# get the total difference in profits i.e. producer surplus
change_ps <- sum(profits_after) - sum(profits_before)

# get change in total surplus
total_surplus_change <- change_ps - mean_cv

# table all the info
q4_table <- data.table(variable = c("change in consumer surplus per person",
                                   "change in Producer surplus per person",
                                   "change in total surplus per person"),
                       value = c(-mean_cv, change_ps, total_surplus_change))

```

```

#####
# ==== Question 5 ====
#####
mc_new <- c(.5,1,3)

p_post_q5 <- p_postmerge_solver(.5, .5, .5, xi, mc.in = mc_new, p.init)

# get cv
cv_i <- cv_i_f(v.in      = v,
               pv_pre    = p_q2,
               pv_post    = p_post_q5,
               xi.in      = xi,
               beta.in     = .5,
               alpha.in    = .5,
               sigma.in    = .5)

# now get mean cv
mean_cv <- mean(cv_i)

# get profits after
profits_after_q5 <- profit_f(pv      = p_post_q5,
                             mc_v    = mc_new,
                             v.in     = v,
                             xi.in    = xi,
                             alpha.in = .5,
                             beta.in   = .5,
                             sigma.in = .5)

# get the total difference in profits i.e. producer surplus
change_ps <- sum(profits_after_q5) - sum(profits_before)

# get change in total surplus
total_surplus_change <- change_ps - mean_cv

# table all the info
q5_table <- data.table(variable = c("change in cosumer surplus per person",
                                   "change in Producer surplus per person",
                                   "change in total surplus per person"),
                       value = c(-mean_cv, change_ps, total_surplus_change))

#####
# ==== save output to latex ====
#####

# make these tables pretty
p_q1_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_q1))
p_q2_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_q2))
p_q3_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_q3))
p_post_q5_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_post_q5))

```

```

# capitolize first letters
q4_table[, variable := sapply(variable, function(x) paste0(sapply(strsplit(x, " "), Hmisc::capitalize),
q5_table[, variable := sapply(variable, function(x) paste0(sapply(strsplit(x, " "), Hmisc::capitalize),

if(opt_save){

  print(xtable(p_q1_out, type = "latex"),
        file = paste0(f_out, "p_q1.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(p_q2_out, type = "latex"),
        file = paste0(f_out, "p_q2.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(p_q3_out, type = "latex"),
        file = paste0(f_out, "p_q3.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(q4_table, type = "latex"),
        file = paste0(f_out, "q4_table.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(p_post_q5_out, type = "latex"),
        file = paste0(f_out, "p_post_q5.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(q5_table, type = "latex"),
        file = paste0(f_out, "q5_table.tex"),
        include.rownames = FALSE,
        floating = FALSE)

}

#####
# ==== run r markdown for tex file ====
#####

rmarkdown::render(input = "C:/Users/Nmath_000/Documents/Code/Econ_631/ps2/ps2_r_markdown.Rmd",
                  output_format = "pdf_document",
                  output_file = paste0(f_out, "assignment_2_r_code_pdf.pdf"))

```