Economics 631 IO - Fall 2019
Problem Set 1

Nathan Mather and Tyler Radler

October 17, 2019

1 Binary Choice Model

1.1

The estimates and standard errors for the probit model are as follows:

Probit results

variable	Estimate	Std. Error
th0	-0.71	0.38
th1	-0.01	0.01
th2	0.10	0.02

1.2

The marginal effect of an additional year of education is given below:

$$\frac{\partial P[Y=1|X,\theta]}{\partial X_2} = \frac{\partial \Phi(\theta_0 + \theta_1 X_1 + \theta_2 X_2)}{\partial X_2} = \theta_2 \phi(\theta_0 + \theta_1 X_1 + \theta_2 X_2)$$

The marginal effect for the average woman, given our parameter estimates, is then simply

$$\widehat{\theta}_2 \phi(\widehat{\theta}_0 + \widehat{\theta}_1 \bar{X}_1 + \widehat{\theta}_2 \bar{X}_2) = 0.041$$

1.3

The estimates and standard errors for the logit model are as follows:

Logit results

variable	Estimate	Std. Error
th0	-1.13	0.61
th1	-0.02	0.01
th2	0.17	0.03

The coefficients are fairly different between the two models, which is likely due to the differences in the assumed error structure between the logit and the probit. The signs do not flip, however, which is reassuring.
The age/education ratio between the two models is more similar. For the probit this ratio is -0.0886, while for the logit it is -0.0905, which is fairly close.

1.4

It seems likely that completed education would be correlated with innate "ability", which would also cause higher wages and increase the likelihood of an individual working. In other words, it is likely that $x_2 i$ is correlated with ε_i. If this were the case we should expect those with high x_2 to also have high ε_i, which would cause our estimates to be biased upward.

1.5

The parameter ρ is the covariance between the errors in the assumed DGP for the work decision and the education decision. I'd expect ρ to be positive, which would mean who have higher conditional-on-observables educational attainment to have higher conditional-on-observables propensities to work. The assumption that z_i does not influence y_i directly means it can function as a valid excluded instrument for x_2 – high values of z_i increase the probability of work only through it's influence on the education status of the observed individual.

1.6

We are given that

$$p(y_i|x_{1i}, x_{2i}, z_i; \theta) = p(y_i|x_{1i}, x_{2i}, zi_;\theta)p(x_{2i}|x_{1i}, z_i; \theta)$$

and that

$$p(y_i = 1|x_{1i}, x_{2i}, zi_;\theta) = p(\theta_0 + \theta_1 x_{1i} + \theta_2 x_{2i} + \varepsilon_i > 0|x_{1i}, x_{2i}, z_i; \theta)$$

Rearranging gives us

$$p(\varepsilon_i > -\theta_0 - \theta_1 x_{1i} - \theta_2 x_{2i}|x_{1i}, x_{2i}, z_i; \theta) = 1 - F_\varepsilon(-\theta_0 - \theta_1 x_{1i} - \theta_2 x_{2i}|x_{1i}, x_{2i}, z_i; \theta)$$

where

$$\varepsilon_i|x_{1i}, x_{2i}, z_i; \theta \sim \mathcal{N}(\frac{\rho}{\sigma_w^2}(x_{2i} - (\theta_3 + \theta_4 x_{1i} + \theta_5 z_i)), (1 - \frac{\rho^2}{\sigma_w^2})) \equiv \mathcal{N}(\psi_i, \tau)$$

Note then that

$$p(\frac{\varepsilon_i - \psi_i}{\sqrt{\tau}} > \frac{-\theta_0 - \theta_1 x_{1i} - \theta_2 x_{2i} - \psi_i}{\sqrt{\tau}}|x_{1i}, x_{2i}, z_i; \theta) = 1 - \Phi(\frac{-\theta_0 - \theta_1 x_{1i} - \theta_2 x_{2i} - \psi_i)}{\sqrt{\tau}}) \equiv 1 - \Phi(m_i)$$

Note also that

$$p(x_{2i}|x_{1i}, z_i; \theta) = \phi(\frac{x_{2i} - \theta_3 - \theta_4 x_{1i} - \theta_5 z_i}{\sigma_w})\frac{1}{\sigma_w}$$

Thus

$$p(y_i, x_{2i}|x_{1i}, z_i; \theta) = (1 - \Phi(m_i))^{y_i}\Phi(m_i)^{1-y_i}\phi(\frac{x_{2i} - \theta_3 - \theta_4 x_{1i} - \theta_5 z_i}{\sigma_w})\frac{1}{\sigma_w}$$

and the log-likelihood is

$$\sum_{i=1}^{n} y_i \ln(1 - \Phi(m_i)) + (1 - y_i)\ln(\Phi(m_i)) + \ln(\phi(\frac{x_{2i} - \theta_3 - \theta_4 x_{1i} - \theta_5 z_i}{\sigma_w})) - \ln(\sigma_w)$$

where

$$m_i \equiv \frac{-\theta_0 - \theta_1 x_{1i} - \theta_2 x_{2i} - \frac{\rho}{\sigma_w^2}(x_{2i} - (\theta_3 + \theta_4 x_{1i} + \theta_5 z_i))}{\sqrt{(1 - \frac{\rho^2}{\sigma_w^2})}}$$

The results of the model are as follows:

Joint MLE results

variable	Estimate	Std. Error
th0	-0.43	0.64
th1	-0.01	0.01
th2	0.08	0.04
th3	9.11	0.49
th4	-0.00	0.01
th5	0.37	0.02
p	0.10	0.19
sig	1.98	0.05

2 BLP - Logit

2.1 Market Shares

The market share for product j in period ct is given by

$$s_{jct}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\xi}, \theta) = Pr(u_{ijct} \geq u_{ij'ct} \quad \text{for} \quad \forall j' = 0, 1, ..., J) = \frac{\exp(\boldsymbol{x}_{jt}\boldsymbol{\beta} - \alpha p_{jct} + \xi_{jct})}{(1 + \sum_{j'=1}^{J} \exp(\boldsymbol{x}_{j't}\boldsymbol{\beta} - \alpha p_{j'ct} + \xi_{j'ct}))}$$

2.2 Mean Utility

Because we have nonrandom coefficients the mean utility from product j in city-year ct is just

$$\delta_{jct} = \boldsymbol{x}_{jt}\boldsymbol{\beta} - \alpha p_{jct} + \xi_{jct}$$

and thus we can rewrite the mean utility in terms of the market shares as

$$s_{jct} = \frac{\exp(\boldsymbol{x}_{jt}\boldsymbol{\beta} - \alpha p_{jct} + \xi_{jct})}{(1 + \sum_{j'=1}^{J} \exp(\boldsymbol{x}_{j't}\boldsymbol{\beta} - \alpha p_{j'ct} + \xi_{j'ct}))} = \frac{\exp(\delta_{jct})}{(1 + \sum_{j'=1}^{J} \exp(\delta_{j'ct}))}$$

and so

$$\ln(s_{jct}) = \delta_{jct} - \ln(1 + \sum_{j'} \exp(\delta_{j'ct}))$$

if we define $\ln(s_{0ct}) = -\ln(1 + \sum_{j'} \exp(\delta_{j'ct}))$ then we can rewrite to

$$\delta_{jct} = \ln(s_{jct}) - \ln(s_{0ct})$$

2.3 part 2

2.3.1 sub part a

OLS Results

term	estimate	std.error	statistic	p.value
(Intercept)	-2.99	0.11	-26.80	0.00
mushy	0.05	0.05	1.00	0.32
sugar	0.05	0.00	10.49	0.00
price	-10.12	0.88	-11.51	0.00

2.3.2 sub part b

First IV Results

term	estimate	std.error	statistic	p.value
(Intercept)	-7.54	0.73	-10.27	0.00
mushy	0.25	0.08	3.26	0.00
sugar	-0.02	0.01	-1.21	0.22
price	29.92	6.59	4.54	0.00

2.3.3 sub part c

Second IV Results

term	estimate	std.error	statistic	p.value
(Intercept)	-2.36	0.72	-3.29	0.00
mushy	0.02	0.06	0.34	0.73
sugar	0.05	0.01	5.48	0.00
price	-15.59	6.21	-2.51	0.01

3 BLP - Random Coefficients

3.1

The market share for product j in period ct is given by

$$s_{jct}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\xi}, \theta) = \int \frac{\exp(\boldsymbol{x}_{jct}\boldsymbol{\beta}_i - p_{jct}\alpha + \xi_{jct})}{1 + \sum_{j'} \exp(\boldsymbol{x}_{j'ct}\boldsymbol{\beta}_i - p_{j'ct}\alpha + \xi_{j'ct})} dF_{\boldsymbol{\beta}_{i1}, \boldsymbol{\beta}_{i2}}(\boldsymbol{\beta}_{i1}, \boldsymbol{\beta}_{i2})$$

Given our functional form assumptions we can rewrite $\boldsymbol{\beta}_{ik} = \boldsymbol{\beta}_k + \sigma_k v_{ik}$ where β_k is the mean of the coefficient and σ_k is the standard deviation.

$$s_{jct}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\xi}, \theta) = \int \frac{\exp(\boldsymbol{x}_{jct}\boldsymbol{\beta} - p_{jct}\alpha + \xi_{jct} + \sum_k \sigma_k x_{jct} v_{ki})}{1 + \sum_{j'} \exp(\boldsymbol{x}_{j'ct}\boldsymbol{\beta} - p_{j'ct}\alpha + \xi_{j'ct} + \sum_k \sigma_k x_{j'ct} v_{ki})} dF_v(v_{i1}, v_{i2})$$

Finally we can define $\delta_j = \boldsymbol{x}_j\boldsymbol{\beta} - p_j\alpha + \xi_j$ as the mean utility for purchasing product j, and rewrite the above expression only in terms of δ_j and the σ_k.

$$s_{jct}(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{\delta}, \boldsymbol{\sigma}) = \int \frac{\exp(\delta_j + \sigma_1 x_j v_{1i} + \sigma_2 x_j v_{2i})}{1 + \sum_{j'} \exp(\delta_{j'} + \sigma_1 x_j v_{1i} + \sigma_2 x_j v_{2i})} dF_v(v_{i1}, v_{i2})$$

3.2

The moment condition are are trying to match is

$$\mathrm{E}[z_j \xi_j] = 0$$

Where for the sample moments we get ξ_j from the relationship $\delta_j(\boldsymbol{p}, \boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\sigma}) = \boldsymbol{x}_j\boldsymbol{\beta} - p_j\alpha + \xi_j$. If z_j is the average characteristics for other products produced by the same firm we can write

$$z_j = \frac{(\sum_{j'}(\text{j and j' produced by same firm})x'_j) - x_j}{(\sum_{j'}(\text{j and j' produced by same firm})) - 1}$$

. Note that z_j is a vector over mushiness and sugar content. Then the objective function is

$$\min_{\alpha,\beta,\sigma}(\delta(\boldsymbol{p},\boldsymbol{x},\boldsymbol{s},\boldsymbol{\sigma}) - \boldsymbol{x}\boldsymbol{\beta} + p\alpha)'\boldsymbol{z}W\boldsymbol{z}'(\delta(\boldsymbol{p},\boldsymbol{x},\boldsymbol{s},\boldsymbol{\sigma}) - \boldsymbol{x}\boldsymbol{\beta} + p\alpha)$$

Where z is defined as above and W is the identity matrix.
In practice we note that the moment conditions are linear in $\boldsymbol{\beta}$ and α conditional on the $\boldsymbol{\sigma}$ and can be calculated from the overidentified IV formula referenced in the notes. Thus we actually seek to minimize the objective function below:

$$\min_{\boldsymbol{\sigma}}(\delta(\boldsymbol{p},\boldsymbol{x},\boldsymbol{s},\boldsymbol{\sigma}) - \boldsymbol{x}\hat{\boldsymbol{\beta}}(\boldsymbol{\sigma}) + p\hat{\alpha}(\boldsymbol{\sigma}))'\boldsymbol{z}W\boldsymbol{z}'(\delta(\boldsymbol{p},\boldsymbol{x},\boldsymbol{s},\boldsymbol{\sigma}) - \boldsymbol{x}\hat{\boldsymbol{\beta}}(\boldsymbol{\sigma}) + p\hat{\alpha}(\boldsymbol{\sigma}))$$

3.3

The results of the model are as follows:

BLP Random Coefficients Results

Variable	value	SE
Sigma Sugar	9.66	21514140.63
Sigma Mushy	43.27	118558266.19
Beta Sugar	-0.00	247973982.90
Deta Mushy	-0.27	26746725.82
Alpha	29.67	172095717.80

The standard errors seem to be wrong. They are unbelievably large. This could certainly be an error in our code, but it may also be related to the answer to part 3.4.

3.4

Based off of our understanding of the necessary conditions for identifying $\boldsymbol{\sigma}$ this dataset does **not** have the necessary exogenous variation to identify the $\boldsymbol{\sigma}$. To estimate $\boldsymbol{\sigma}$ we need variation in the choice sets across markets, which does not appear to be the case here – each product in the dataset has positive market share in every market. Therefore we do not have the variation in the choice sets needed to identify $\boldsymbol{\sigma}$.

4 Appendix

4.1 R Code

pset 1 631

```r
#=============================================#
# ==== Inustrial Organization Problem Set 1 ====
#=============================================#


# clear objects
rm(list = ls(pos = ".GlobalEnv"), pos = ".GlobalEnv")
options(scipen = 999)
cat("\f")


# Load packages
library(data.table)
library(stats4)
library(broom)
library(AER)
library(xtable)
library(Matrix)
library(BLPestimatoR)
library(SQUAREM)
library(BB)

# set save option

opt_save <- TRUE

# set path for output
f_out <- "C:/Users/Nmath_000/Documents/Code/courses/econ 631/ps1/tex/"

#=====================#
# ==== Question 1 ====
#=====================#

# laod data
q1dt <- fread("file:///C:/Users/Nmath_000/Documents/MI_school/Third Year/Econ 631/ps1/ps1.dat")



# create variable names
setnames(q1dt, colnames(q1dt), c("y", "x1", "x2", "z"))


#=================#
# ==== part 1 ====
#=================#

# write log likelihood function
Probit_llf <- function(th0, th1, th2){
```

```
  mu <- q1dt[,  pnorm(th0 + th1*x1 + th2*x2)]

  -sum(q1dt[, y*log(mu) + (1-y)*log(1-mu)])
}

# create starting values
prob_start <- list(th0 = 0,
                   th1 = .01,
                   th2 = .01)

# run the mle function
probit_res <- mle(Probit_llf, start = prob_start)

# get the results I need
probit_res <- data.table(variable = rownames(summary(probit_res)@coef),
                         summary(probit_res)@coef)

# check my results
check <- glm( y ~ x1+ x2,
              family = binomial(link = "probit"),
              data = q1dt)

# looks good
summary(check)
probit_res

#================#
# ==== Part2 ====
#================#

probit_res$Estimate[3]*dnorm(probit_res$Estimate[1] + probit_res$Estimate[2]*mean(q1dt$x1) + probit_res$

#================#
# ==== Part3 ====
#================#

# define logit log likelihood
logit_llf <- function(th0, th1, th2){


  mu <- q1dt[,  plogis(th0 + th1*x1 + th2*x2)]

  -sum(q1dt[, y*log(mu) + (1-y)*log(1-mu)])
}

# create startign values
logit_start <- list(th0 = 0,
                    th1 = .01,
                    th2 = .01)

# run the mle function
logit_res <- mle(logit_llf, start = logit_start)
```

```r
# get the results I need
logit_res <- data.table(variable = rownames(summary(logit_res)@coef),
                        summary(logit_res)@coef)

# check my results
check <- glm( y ~ x1+ x2,
              family = binomial(link = "logit"),
              data = q1dt)
# looks good
summary(check)
logit_res

#=================#
# ==== Part 6 ====
#=================#


p6_llf <- function(th0, th1, th2, th3, th4, th5, p, sig){

  # define some intermediate terms
  q1dt[, psi := (p/sig^2)*(x2 - (th3 + th4*x1 + th5*z))]
  q1dt[, tau := 1-(p^2/sig^2)]
  q1dt[, m:= (-th0 - th1*x1 - th2*x2 - psi)/(tau^.5)]

  # get prob y equals one conditional on x1, x2, z, theta
  q1dt[, p_y_1 := 1- pnorm(m)]

  # get pro x2 = x2i given x1 z theta
  q1dt[, p_x := dnorm((x2 - th3 - th4*x1 - th5*z)/sig)]

  # get log likelihoods
  q1dt[, llf := y*log(p_y_1) + (1-y)*log(1-p_y_1) + log(p_x) - log(sig)]

  # sum them
  sum_llf <- q1dt[, sum(llf)]

  # remove extra vars
  q1dt[ ,`:=` (psi = NULL, tau = NULL, m = NULL, p_y_1 = NULL, p_x = NULL, llf = NULL)]

  #  return negative sum
  return(-sum_llf)
}

second_stage <- glm(y ~ x1+ x2,
                    family = binomial(link = "probit"),
                    data = q1dt)

first_stage <- lm(x2 ~ x1 + z, data = q1dt)

p6_start <- as.list(c(second_stage$coefficients, first_stage$coefficients, .1, 1))

names(p6_start) <- c( paste0("th", 0:5), "p", "sig")
```

```r
# run the mle function
p6_res <- mle(p6_llf, start = p6_start)

logLik(p6_res)
# get the results I need
p6_res_tab <- data.table(variable = rownames(summary(p6_res)@coef),
                         summary(p6_res)@coef)

p6_res_tab




#====================#
# ==== Question 2 ====
#====================#

  #=================#
  # ==== part 3 ====
  #=================#


    # load cereal data
    cereal <- data.table(readxl::read_excel("C:/Users/Nmath_000/Documents/MI_school/Third Year/Econ 631/


    # get total market share by city year
    cereal[, s0 := 1-sum(share), c("city", "year", "quarter" )]


    # create column for mean utility
    cereal[, m_u := log(share) - log(s0)]

    #============#
    # ==== a ====
    #============#

    # run ols, tidy it up, make it a data.table
    q2_p3_ols <- data.table(tidy(lm(m_u ~ mushy + sugar + price , data = cereal)))

    # round p value
    q2_p3_ols[, p.value := round(p.value, 6)]

    #============#
    # ==== b ====
    #============#


    # create  sugar instroment
    cereal[, (.N-1), c('firm_id', "city", "quarter")]
    cereal[, i1_sugar := (sum(sugar) - sugar)/ (.N-1), c('firm_id', "city", "quarter")]

    # create mush intrument
```

```
    cereal[, i1_mushy := (sum(mushy) - mushy)/(.N-1), c('firm_id', "city", "quarter")]

    # create price instrument
    cereal[, i1_price := (sum(price) - price)/ (.N-1), c('firm_id', "city", "quarter")]

    # now do 2sls, tidy it up, make it a data.table
    q2_p3_iv1 <- data.table(tidy(ivreg(m_u ~ mushy + sugar + price | i1_sugar + i1_mushy +  mushy + suga

    q2_p3_iv1[, p.value := round(p.value,6)]


    #============#
    # ==== c ====
    #============#

    # now create second set of instroments
    # would be ideal to write functino for this, but who has the time
    # create  sugar instroment
    # start by getting sum of sugar for firm
    cereal[, f_sugar_sum := sum(sugar), c( "city", "quarter", "firm_id")]

    # get number of products by firm
    cereal[, f_nprod := .N, c( "city", "quarter", "firm_id")]

    # get sum of sugar in market, subtract off sum of sugar for the firm
    # divide by number of products minus the products for this firm that we subtracted off
    cereal[, i2_sugar := (sum(sugar) - f_sugar_sum)/ (.N-f_nprod), c("city", "quarter")]

    # now do the same for the other
    cereal[, f_mushy_sum := sum(mushy), c( "city", "quarter", "firm_id")]
    cereal[, i2_mushy := (sum(mushy) - f_mushy_sum)/ (.N-f_nprod), c("city", "quarter")]

    # #note dont actually need this
    # cereal[, f_price_sum := sum(price), c( "city", "quarter", "firm_id")]
    # cereal[, i2_price := (sum(price) - f_price_sum)/ (.N-f_nprod), c("city", "quarter")]

    # now do 2sls
    ivreg_out <- ivreg(m_u ~ mushy + sugar + price | i2_sugar + i2_mushy + mushy + sugar , data = cereal
    q2_p3_iv2 <- data.table(tidy(ivreg(m_u ~ mushy + sugar + price | i2_sugar + i2_mushy + mushy + sugar

    q2_p3_iv2[, p.value := round(p.value,6)]



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

  #===================================#
  # ==== set up data for functions ====
  #===================================#
```

```r
# load cereal data
cereal <- data.table(readxl::read_excel("C:/Users/Nmath_000/Documents/MI_school/Third Year/Econ 631/

#note these are data manipulations that can happen outside the outer loop. We
# just need to convert some things to matrices since we actually need to
# use matrix algebra in this question



# create a single market variale. Don't have to referece two variables then
cereal[, mkt := paste0(city, "_", quarter)]

# get total market share by city year
#note: should we do this before or after dropping obs with missing instruments?
cereal[, s0 := 1-sum(share), mkt]

# create column for mean utility
cereal[, m_u := log(share) - log(s0)]

# create  sugar instrument
cereal[, (.N-1), c('firm_id', "city", "quarter")]
cereal[, i1_sugar := (sum(sugar) - sugar)/ (.N-1), c('firm_id', "city", "quarter")]

# create mush instrument
cereal[, i1_mushy := (sum(mushy) - mushy)/(.N-1), c('firm_id', "city", "quarter")]

# drop observations with missing instruments. Missing becasue firm only has one product
cereal <- cereal[!is.na(i1_sugar)]

# defint some variables
# I'm not crazy about doing this. I think this either needs to be written as a function
# or else we should just hard code the column names. This is like a weird middle ground
# of generalizability that isn't that helpful
share.fld =     "share"
prod.id.fld =   "product_id"
mkt.id.fld =    "mkt"
prc.fld =       "price"
x.var.flds =    c("sugar",
                  "mushy")

# order data and get nrows
cereal <- setorder(cereal, "city", "product_id")
JM <- nrow(cereal)

# make matrix of controls X
X <- as.matrix(cereal[, c(x.var.flds), with = FALSE])
K <- ncol(X)

# make a matrix that includes price
cereal[, n_price := -price]
XP <- as.matrix(cereal[, c("sugar", "mushy", "n_price"), with = FALSE])

# make a vector for price
```

```
  P <- as.matrix(cereal[, price])

  # put market into a vector
  mkt.id <- cereal[, get(mkt.id.fld)]

  # put shares into a vector
  s.jm <- as.vector(cereal[, get(share.fld)]);

  # put shates into a vector
  s.j0 <- cereal[, s0 ]

  # set number of simulated consumers
  n.sim = 20

  # Standard normal distribution draws, one for each characteristic in X
  # columns are simulated consumers, rows are variables in X (including constant and price)
  # as Ying pointed out we want to do this once outside the outer loop to get faster convergence
  v = matrix(rnorm(K * n.sim), nrow = K, ncol = n.sim)

  # Z matrix needed for gmm function
  Z <- as.matrix(cereal[!is.na(i1_sugar),c("sugar", "mushy", "i1_sugar", "i1_mushy"), with = FALSE])

  # PZ matrix needed for GMM function
  PZ <- Z %*% solve(t(Z) %*% Z) %*% t(Z)

  # make weighting matrix. Also used in GMM function
  W.inv <- diag(1, 4, 4 )

#================================#
# ==== functions for search ====
#================================#

  # these are all the functions we will need for a given sd_mush sd_sug guess

  # hard code estimates for variance paremter to test funcitons
  # this is what the outside loop will be optimizing over
  sd_mush_test <- .1
  sd_sug_test <- .2

  # get inital delta guess
  #note doesn't need to come from cereal data.table, this is just an initial guess
  delta.initial <- cereal[, m_u]

  # this function takes the data.table, V sims,  and the sd parameter estimates and returns
  # the mu (individual utility) estimates
  #note I am hard coding this for mushy and sugar as the relavent variables.
  # could write this in a more general way if we wanted and have it take a list of theta parms
  # and a list of corresponding variabe names
  find_mu <- function(X, sd_sug.in,  sd_mush.in, v.in){

     X %*% diag(x = c(sd_sug.in, sd_mush.in ), 2, 2) %*%v.in
  }
```

```r
# test it out
mu_mat <- find_mu(X, sd_sug_test, sd_mush_test, v )



# This computes a matrix of share estiamtes. rows are estimates for every product
# columns are estimates for every simulated V
ind_sh <- function(delta.in, mu.in, mkt.id.in){
  # This function computes the "individual" probabilities of choosing each brand
  numer <- exp(mu.in) * matrix(rep(exp(delta.in), n.sim), ncol = n.sim)

  denom <- as.matrix(do.call("rbind", lapply(mkt.id.in, function(tt){
    1 + colSums(numer[mkt.id.in %in% tt, ])

  })))
  return(numer / denom);
}

# test it out
sj_mat <- ind_sh(delta.initial, mu_mat, mkt.id)


blp_inner <- function(delta.in, mu.in, s.jm.in, mkt.id.in) {
  # Computes a single update of the BLP (1995) contraction mapping.
  # of market level predicted shares.
  # This single-update function is required by SQUAREM, see Varadhan and
  # Roland (SJS, 2008), and Roland and Varadhan (ANM, 2005)
  # INPUT
  #   delta.in : current value of delta vector
  #   mu.in: current mu matrix
  # OUTPUT
  #   delta.out : delta vector that equates observed with predicted market shares
  pred.s <- rowMeans(ind_sh(delta.in, mu.in, mkt.id.in));
  delta.out <- delta.in + log(s.jm.in) - log(pred.s)
  return(delta.out)
}

# test it out
delta_new <- blp_inner(delta.initial, mu_mat, s.jm, mkt.id)


# The function to iterate the contraction mapping is genereic. WE don't need to write it
# here is an example #note it is dependent on output of above examples
#note I have to pass additonal variables so that we are not pulling objects from global enviorment
  squarem.output <- squarem(par       = delta.initial,
                            fixptfn   = blp_inner,
                            mu.in     = mu_mat,
                            s.jm.in   = s.jm,
                            mkt.id.in = mkt.id,
                            control   = list(trace = TRUE))
  delta <- squarem.output$par
  summary(delta.initial - delta)
```

```r
# function to runn GMM
# a lot of inputs here but this is how you get around using global objects
# THis is supposed to be better practice but it doe sget a bit wild with all these
gmm_obj_f <- function(parm_vector.in, delta.in, X.in,  XP.in,  Z.in, PZ.in, W.inv.in, s.jm.in, mkt.

  sd_sug.in <- exp(parm_vector.in[[1]])
  sd_mush.in <- exp(parm_vector.in[[2]])

   # get new MU matrix
  mu <- find_mu(X.in, sd_sug.in = sd_sug.in, sd_mush.in = sd_mush.in, v.in)

  # update delta
  squarem.output <- squarem(par       = delta.in,
                            fixptfn   = blp_inner,
                            mu.in     = mu,
                            s.jm.in   = s.jm.in,
                            mkt.id.in = mkt.id.in,
                            control   = list(trace = TRUE))
  delta_new <- squarem.output$par

  # first step
  PX.inv <- solve(t(XP.in) %*% PZ.in %*% XP.in)

  # finsih getting theta
  theta1 <- PX.inv %*% t(XP.in) %*% PZ.in %*% delta_new

  # get xi hat
  xi.hat <- delta_new - XP.in %*% theta1

  result <- t(xi.hat) %*% Z.in %*% W.inv.in %*% t(Z.in) %*% xi.hat
  # get function value
  return(result)

}

# make sd_vector
parm_vector <- c(3, 3)

# test it out
f <- gmm_obj_f(parm_vector.in = parm_vector,
               delta.in    = delta.initial,
               X.in        = X,
               XP.in       = XP,
               Z.in        = Z,
               PZ.in       = PZ,
               W.inv.in    = W.inv,
               s.jm.in     = s.jm,
               mkt.id.in   = mkt.id,
               v.in        = v)


Results <-  optim(par         = c(3,3),
```

```
                    fn          =  gmm_obj_f,
                   # # uncomment if you want fast convergence
                    # control       = list(reltol = 5),
                    delta.in    = delta.initial,
                    X.in        = X,
                    XP.in       = XP,
                    Z.in        = Z,
                    PZ.in       = PZ,
                    W.inv.in    = W.inv,
                    s.jm.in     = s.jm,
                    mkt.id.in   = mkt.id,
                    v.in        = v)

# save these real quick in case something happens
save(Results, file = paste0(f_out, "results_save.R"))

# grab out values
sd_final <- exp(Results$par)
sd_final



# get betas
# get new MU matrix
mu <- find_mu(X, sd_sug.in = sd_final[[1]], sd_mush.in = sd_final[[2]], v)

# update delta
squarem.output <- squarem(par       = delta.initial,
                          fixptfn   = blp_inner,
                          mu.in     = mu,
                          s.jm.in   = s.jm,
                          mkt.id.in = mkt.id,
                          control   = list(trace = TRUE))
delta_final <- squarem.output$par

# first step
PX.inv <- solve(t(XP) %*% PZ %*% XP)

# finsih getting theta
theta_final <- PX.inv %*% t(XP) %*% PZ %*% delta_new


# get m(parms) moment condition
temp <- delta_final - XP%*%theta_final
temp <- cbind(temp, temp, temp, temp)
m_final <- Z*(temp)


S_final <- (1/nrow(m_final))*t(m_final)%*%m_final

# get derivative of moment condition for Beta
temp <- cbind(X[,1],X[,1],X[,1],X[,1])
dmdb1 <- -Z*temp
```

```
temp <- cbind(X[,2],X[,2],X[,2],X[,2])
dmdb2 <- -Z*temp

# now for p
temp <- cbind(P,P,P,P)
dmda <- Z*temp


#=======================#
# ==== now get dmds ====
#=======================#

  #=======================#
  # ==== Get first one ====
  #=======================#

    # set phi
    phi <- .001
    # first get delta with + phi
    # get new MU matrix
    mu <- find_mu(X, sd_sug.in = sd_final[[1]] + phi, sd_mush.in = sd_final[[2]], v)

    # update delta
    squarem.output <- squarem(par       = delta.initial,
                              fixptfn   = blp_inner,
                              mu.in     = mu,
                              s.jm.in   = s.jm,
                              mkt.id.in = mkt.id,
                              control   = list(trace = TRUE))
    delta_phi_p0 <- squarem.output$par
    temp <- delta_phi_p0 - XP%*%theta_final
    temp <- cbind(temp, temp, temp, temp)
    m_p0 <- Z*(temp)


    # do it again
    mu <- find_mu(X, sd_sug.in = sd_final[[1]] - phi, sd_mush.in = sd_final[[2]], v)

    # update delta
    squarem.output <- squarem(par       = delta.initial,
                              fixptfn   = blp_inner,
                              mu.in     = mu,
                              s.jm.in   = s.jm,
                              mkt.id.in = mkt.id,
                              control   = list(trace = TRUE))
    delta_phi_m0 <- squarem.output$par
    temp <- delta_phi_m0 - XP%*%theta_final
    temp <- cbind(temp, temp, temp, temp)
    m_m0 <- Z*(temp)

    # now get first dmds
    dmds1 <- (m_p0 - m_m0)/(2*phi)
```

```
  #=========================#
  # ==== get second one ====
  #=========================#

    mu <- find_mu(X, sd_sug.in = sd_final[[1]], sd_mush.in = sd_final[[2]] + phi, v)

    # update delta
    squarem.output <- squarem(par       = delta.initial,
                              fixptfn   = blp_inner,
                              mu.in     = mu,
                              s.jm.in   = s.jm,
                              mkt.id.in = mkt.id,
                              control   = list(trace = TRUE))
    delta_phi_0p <- squarem.output$par

    temp <- delta_phi_0p - XP%*%theta_final
    temp <- cbind(temp, temp, temp, temp)
    m_0p <- Z*(temp)


    mu <- find_mu(X, sd_sug.in = sd_final[[1]], sd_mush.in = sd_final[[2]] - phi, v)

    # update delta
    squarem.output <- squarem(par       = delta.initial,
                              fixptfn   = blp_inner,
                              mu.in     = mu,
                              s.jm.in   = s.jm,
                              mkt.id.in = mkt.id,
                              control   = list(trace = TRUE))
    delta_phi_0m <- squarem.output$par


    temp <- delta_phi_0m - XP%*%theta_final
    temp <- cbind(temp, temp, temp, temp)
    m_0m <- Z*(temp)

    # Now finish getting dmds

    dmds2 <- (m_0p - m_0m)/(2*phi)

  #==============================#
  # ==== get standard errors ====
  #==============================#

    isSymmetric(S_final)

    S_final
    m_final

    dmdb1
    dmdb2
    dmda
    dmds1
```

```
      dmds2

      Rho <- t(rbind(colMeans(dmdb1),
                     colMeans(dmdb2),
                     colMeans(dmda),
                     colMeans(dmds1),
                     colMeans(dmds2)))

    SE_mat_final <- (1/JM)*(solve(t(Rho) %*% Rho, tol = 10^(-25)) %*% (t(Rho) %*% S_final %*% Rho) %*% s

    SE_final <- sqrt(diag(SE_mat_final))

    # put estiamates and SE into a table
    q3_p5 <- data.table(Variable = c("Sigma Sugar", "Sigma Mushy", "Beta Sugar", "Deta Mushy", "Alpha")



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

  if(opt_save){


    print(xtable(probit_res, type = "latex"),
          file = paste0(f_out, "q1_p1.tex"),
          include.rownames = FALSE,
          floating = FALSE)


    print(xtable(logit_res, type = "latex"),
          file = paste0(f_out, "q1_p3.tex"),
          include.rownames = FALSE,
          floating = FALSE)


    print(xtable(p6_res_tab, type = "latex"),
          file = paste0(f_out, "q1_p6.tex"),
          include.rownames = FALSE,
          floating = FALSE)

    print(xtable(q2_p3_ols, type = "latex"),
          file = paste0(f_out, "q2_p3a.tex"),
          include.rownames = FALSE,
          floating = FALSE)

    print(xtable(q2_p3_iv1, type = "latex"),
          file = paste0(f_out, "q2_p3b.tex"),
          include.rownames = FALSE,
          floating = FALSE)

    print(xtable(q2_p3_iv2, type = "latex"),
          file = paste0(f_out, "q2_p3c.tex"),
```

```
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(q3_p5, type = "latex"),
        file = paste0(f_out, "q3_p5.tex"),
        include.rownames = FALSE,
        floating = FALSE)


}

  #====================================#
  # ==== run markdown to print code ====
  #====================================#

  rmarkdown::render(input =  "C:/Users/Nmath_000/Documents/Code/courses/econ 631/ps1/ps1_r_markdown.Rm
                    output_format = "pdf_document",
                    output_file = paste0(f_out, "assignment_1_r_code_pdf.pdf"))
```