MKTG 551-3, Homework 3

Rayyan Sayeed

July 31, 2018

1 Introduction

Here, we estimate a homogeneous aggregate logit model on a data set from BLP (1995). We begin by structuring the data for estimation and creating instruments for each IV considered, and then estimate the model using OLS, 2SLS and finally, GMM. This provides us with insight on the choice of our instruments, and we conclude by examining price markups and marginal costs.

2 Structuring Data

We begin by structuring the BLP data for estimation. This consists of creating a column of market shares for each product, calculating the market share of the outside good, creating a mean utility column, and setting up BLP instruments. The BLP instruments considered were cost, own-product, and other-product IVs, as well as all of these together. The cost instruments were already included in the datafile on the price of steel and wages, and instruments for own-product and other-product IVs were created and placed within the datafile. The instruments for these two IV scenarios were horsepower-to-weight ratio, AC, miles per dollar ratio, car size, and price. The code which accomplishes all of this is included below.

```
setwd("C:/Users/Rayyan Sayeed/Documents/Northwestern Spring 2018/MKTG 551-3/Homework 3")
library(dplyr, tidyr)
library(MASS)
```

```
df=read.csv("car_panel.csv",header=T)

# create market share vector, year sum (ownshare summed by year)

df$ownshare=df$q/df$num_hh
yrsum=aggregate(ownshare~year,data=df,FUN=sum)

# create vector to match dimension of yrsum with datafile

dim1=as.vector(t(table(df$year)))

# create outside share vector

df$outsideshare=1-as.numeric(rep(yrsum$ownshare,times=dim1))

# mean utility

df$meanutility=log(df$ownshare)-log(df$outsideshare)

############## create BLP-instruments now...

###### COST-IV vars already in df, steel_index and wages_index

###### OWN-IV

# create vector to match dimensions
dim2=as.vector(t(table(df$year,df$firm_id)))
dim2 = dim2[dim2!=0]

# Air

own_ag_ac=aggregate(air~firm_id+year,data=df,FUN=sum)
```

```
own_ac_vec=rep(own_ag_ac$air,times=dim2)
own_ac_vec=own_ac_vec/(rep(dim2,times = dim2)-1)

own_ac_vec[is.infinite(own_ac_vec)]=0
own_ac_vec[is.nan(own_ac_vec)]=0

# Miles/dollar ratio

own_ag_mpd=aggregate(mpd~firm_id+year,data=df,FUN=sum)
own_mpd_vec=rep(own_ag_mpd$mpd,times=dim2)
own_mpd_vec=own_mpd_vec/(rep(dim2,times = dim2)-1)

own_mpd_vec[is.infinite(own_mpd_vec)]=0
own_mpd_vec[is.nan(own_mpd_vec)]=0


# Size

own_ag_size=aggregate(log_car_size~firm_id+year,data=df,FUN=sum)
own_size_vec=rep(own_ag_size$log_car_size,times=dim2)
own_size_vec=own_size_vec/(rep(dim2,times = dim2)-1)

own_size_vec[is.infinite(own_size_vec)]=0
own_size_vec[is.nan(own_size_vec)]=0

# HP to Weight Ratio

own_ag_hp2wt=aggregate(hp2wt~firm_id+year,data=df,FUN=sum)
own_hp2wt_vec=rep(own_ag_hp2wt$hp2wt,times=dim2)
own_hp2wt_vec=own_hp2wt_vec/(rep(dim2,times = dim2)-1)

own_hp2wt_vec[is.infinite(own_hp2wt_vec)]=0
```

```
own_hp2wt_vec[is.nan(own_hp2wt_vec)]=0

########## OTHER-IV

dim3=as.vector(t(table(df$year)))
dim3=dim3[dim3 != 0]

# Air

other_ag_ac=aggregate(air~year,data=df,FUN=sum)
other_ac_vec=rep(other_ag_ac$air,times=dim3)
other_ac_vec=other_ac_vec/(rep(dim3,times = dim3)-1)

other_ac_vec[is.infinite(other_ac_vec)]=0
other_ac_vec[is.nan(other_ac_vec)]=0

# Miles/dollar ratio

other_ag_mpd=aggregate(mpd~year,data=df,FUN=sum)
other_mpd_vec=rep(other_ag_mpd$mpd,times=dim3)
other_mpd_vec=other_mpd_vec/(rep(dim3,times = dim3)-1)

other_mpd_vec[is.infinite(other_mpd_vec)]=0
other_mpd_vec[is.nan(other_mpd_vec)]=0

# Size

other_ag_size=aggregate(log_car_size~year,data=df,FUN=sum)
other_size_vec=rep(other_ag_size$log_car_size,times=dim3)
other_size_vec=other_size_vec/(rep(dim3,times = dim3)-1)

other_size_vec[is.infinite(other_size_vec)]=0
```

```
other_size_vec[is.nan(other_size_vec)]=0

# HP to Weight Ratio

other_ag_hp2wt=aggregate(hp2wt~year,data=df,FUN=sum)
other_hp2wt_vec=rep(other_ag_hp2wt$hp2wt,times=dim3)
other_hp2wt_vec=other_hp2wt_vec/(rep(dim3,times = dim3)-1)

other_hp2wt_vec[is.infinite(other_hp2wt_vec)]=0
other_hp2wt_vec[is.nan(other_hp2wt_vec)]=0
```

3 Homogeneous Aggregate Logit Model – OLS

Here, we implement the homogeneous aggregate logit model with utility using the formula,

$$u_{itj} = x'_{tj} - \alpha_p p_{tj} + \xi_{tj} + \epsilon_{itj}.$$

We then estimate the model using OLS, and report the estimates and standard errors yielded by the estimation. The code that accomplishes this is included below.

```
# Implement homogenous aggregate logit model with utility

hal_model=lm(meanutility~hp2wt+air+mpd+car_size+p_real, data=df)

# Standard Errors
coef(summary(hal_model))[, "Std. Error"]
```

The OLS estimation yielded the following estimates and standard errors:

```
> hal_model

Call:
lm(formula = meanutility ~ hp2wt + air + mpd + car_size + p_real,
    data = df)

Coefficients:
(Intercept)        hp2wt          air          mpd     car_size       p_real
  -10.12825     -0.01064     -0.03211      0.27959      2.36114     -0.08862

> hal_errors
(Intercept)       hp2wt         air         mpd    car_size      p_real
0.256006033 0.027705147 0.072731235 0.044355691 0.125767416 0.004021899
```

4 2SLS Implementation

We then estimate the same logit model using a 2SLS approach. The 2SLS estimation is done using each of the IV scenarios: cost, own, other, and all. The code which implements this is included below.

```
############ START OF INSTRUMENTS, 2SLS

# Estimate model using 2SLS and a) COST-IV b) OWN-IV c) OTHER-IV d) All of the above

markup_vec=df$p_real+1/(1-df$ownshare)

# Estimate Price, p_hat, to use in 2nd stage utility estimation

p_hat=lm(p_real~steel_index+wages_index+1/(1-ownshare), data=df)

# Estimate utility using p_hat

utility=lm(meanutility~hp2wt+air+mpd+car_size+p_hat$fitted.values, data=df)

########################### 2SLS implementation with IVs

# Set up X, input matrix

ones = cbind(rep(1,length(df$id)))
X = cbind(ones,df$hp2wt,df$air,df$mpd,df$car_size,df$p_real)
M = nrow(df) # number of product-market observations, used in variance parts below

################## Part a) Cost-IV

# Z is IV matrix
```

```
Z_cost=cbind(df$steel_index,df$wages_index)

# Compute projection matrix of Z_cost, P_Z=Z(Z'Z)^-1*Z'

P_Z_cost=Z_cost%*%ginv((t(Z_cost)%*%Z_cost))%*%t(Z_cost)

# Compute fitted values of X using X_hat=P_Z*X

X_hat_Z_cost=P_Z_cost%*%X

# 2nd stage estimates, beta_hat_2sls=(X_hat'*X)^-1*X_hat'*delta, delta is mean utility

sls_estimates_Z_cost=ginv(t(X_hat_Z_cost)%*%(X_hat_Z_cost))%*%t(X_hat_Z_cost)%*%df$meanutility

# Residuals, epsilon_hat=delta-X*beta_hat_2sls

residuals_Z_cost= df$meanutility-X%*%(sls_estimates_Z_cost)

# Variance, V_2sls=1/M(epsilon_hat')(epsilon_hat)(X'P_zX)^-1, M = # of product-market
    obsdim

resid_term_cost=(t(residuals_Z_cost)%*%residuals_Z_cost)

V_2sls_Z_cost=(1/M)*resid_term_cost[1,1]*ginv(t(X)%*%P_Z_cost%*%X)

# Standard Error

std_err_cost=V_2sls_Z_cost/sqrt(M)


################## Part b) Own-IV
```

```
Z_own=cbind(own_ac_vec,own_mpd_vec,own_size_vec,own_hp2wt_vec)

P_Z_own=Z_own%*%ginv((t(Z_own)%*%Z_own))%*%t(Z_own)

X_hat_Z_own=P_Z_own%*%X

sls_estimates_Z_own=ginv(t(X_hat_Z_own)%*%(X_hat_Z_own))%*%t(X_hat_Z_own)%*%df$meanutility

residuals_Z_own= df$meanutility-X%*%sls_estimates_Z_own

resid_term_own=(t(residuals_Z_own)%*%residuals_Z_own)

V_2sls_Z_own=(1/M)*resid_term_own[1,1]*ginv(t(X)%*%P_Z_own%*%X)

std_err_own=V_2sls_Z_own/sqrt(M)


################## Part c) Other-IV

Z_other=cbind(other_ac_vec,other_mpd_vec,other_size_vec,other_hp2wt_vec)

P_Z_other=Z_other%*%ginv((t(Z_other)%*%Z_other))%*%t(Z_other)

X_hat_Z_other=P_Z_other%*%X

sls_estimates_Z_other=ginv(t(X_hat_Z_other)%*%(X_hat_Z_other))%*%t(X_hat_Z_other)%*%df$meanutility

residuals_Z_other= df$meanutility-X%*%sls_estimates_Z_other

resid_term_other=(t(residuals_Z_other)%*%residuals_Z_other)
```

```
V_2sls_Z_other=(1/M)*resid_term_other[1,1]*ginv(t(X)%*%P_Z_other%*%X)

std_err_other=V_2sls_Z_other/sqrt(M)


################## Part d) All of the above

Z_all=cbind(df$steel_index,df$wages_index,own_ac_vec,own_mpd_vec,own_size_vec,own_hp2wt_vec,other_

P_Z_all=Z_all%*%ginv((t(Z_all)%*%Z_all))%*%t(Z_all)

X_hat_Z_all=P_Z_all%*%X

sls_estimates_Z_all=ginv(t(X_hat_Z_all)%*%(X_hat_Z_all))%*%t(X_hat_Z_all)%*%df$meanutility

residuals_Z_all= df$meanutility-X%*%sls_estimates_Z_all

resid_term_all=(t(residuals_Z_all)%*%residuals_Z_all)

V_2sls_Z_all=(1/M)*resid_term_all[1,1]*ginv(t(X)%*%P_Z_all%*%X)

std_err_all=V_2sls_Z_all/sqrt(M)

############ END OF INSTRUMENTS, 2SLS
```

The results of estimation in each IV case are included below. Note that the matrices below include the estimates from each scenario as the first column, and the standard errors, defined as standard deviation divided by the square root of the number of observations, as the 6x6 block remaining within each matrix. We note that the estimate values across the IV scenarios are relatively similar in magnitude and agree in sign.

```
> slsinfo_cost
           [,1]         [,2]         [,3]         [,4]         [,5]         [,6]         [,7]
[1,] -0.5253985  0.003250691  0.007533562  0.001847120 -0.004654969  0.007217689 -0.002842382
[2,] -1.2946500  0.007533562  0.017459231  0.004280746 -0.010787999  0.016727179 -0.006587249
[3,] -0.2828222  0.001847120  0.004280746  0.001049577 -0.002645066  0.004101262 -0.001615118
[4,]  0.5856917 -0.004654969 -0.010787999 -0.002645066  0.006665913 -0.010335691  0.004070372
[5,] -1.1232258  0.007217689  0.016727179  0.004101262 -0.010335691  0.016025834 -0.006311123
[6,] -0.1369587 -0.002842382 -0.006587249 -0.001615118  0.004070372 -0.006311123  0.002485713
> slsinfo_own
           [,1]          [,2]         [,3]          [,4]          [,5]          [,6]          [,7]
[1,] -0.39171315  2.662794e-05 -3.978428e-05  1.375621e-05 -3.462199e-07  8.936834e-05  1.140779e-06
[2,] -1.02344370 -3.978428e-05  9.513973e-05 -3.465429e-05 -3.946408e-05 -1.306699e-04 -6.155835e-06
[3,]  0.04359612  1.375621e-05 -3.465429e-05  1.315794e-05  1.929620e-05  4.385109e-05  1.947578e-06
[4,] -0.93521123 -3.462199e-07 -3.946408e-05  1.929620e-05  7.298625e-05 -1.347522e-05  1.843943e-06
[5,] -0.56402968  8.936834e-05 -1.306699e-04  4.385109e-05 -1.347522e-05  3.031123e-04  4.485412e-06
[6,] -0.03233887  1.140779e-06 -6.155835e-06  1.947578e-06  1.843943e-06  4.485412e-06  1.083604e-06
> slsinfo_other
           [,1]          [,2]         [,3]          [,4]          [,5]          [,6]          [,7]
[1,] -1.26415832  4.190242e-04 -5.222679e-04 -1.632626e-04  1.377499e-04  0.0007911904  3.197005e-05
[2,] -0.20572929 -5.222679e-04  7.661682e-04  1.932195e-04 -2.217303e-04 -0.0009880621 -6.850646e-05
[3,]  0.59665951 -1.632626e-04  1.932195e-04  6.528966e-05 -3.082511e-05 -0.0003089760 -1.302968e-05
[4,]  0.06559123  1.377499e-04 -2.217303e-04 -3.082511e-05  5.107453e-04  0.0002397086 -5.296029e-05
[5,] -2.32958061  7.911904e-04 -9.880621e-04 -3.089760e-04  2.397086e-04  0.0014949525  6.447070e-05
[6,] -0.22867076  3.197005e-05 -6.850646e-05 -1.302968e-05 -5.296029e-05  0.0000644707  2.281842e-05
> slsinfo_all
           [,1]          [,2]         [,3]          [,4]          [,5]          [,6]          [,7]
[1,] -12.8701350  5.619754e-03 -4.557049e-04  3.924081e-05 -5.888760e-04 -2.176977e-03  2.291029e-05
[2,]   0.1084292 -4.557049e-04  2.799899e-04  9.942553e-04 -1.290936e-04  8.441806e-05 -6.277138e-05
[3,]  -0.1794150  3.924081e-05  9.942553e-04  5.169523e-03 -7.713624e-04 -1.297228e-04 -2.944162e-04
[4,]   0.4780466 -5.888760e-04 -1.290936e-04 -7.713624e-04  2.170348e-04  2.506106e-04  4.321574e-05
[5,]   3.6941111 -2.176977e-03  8.441806e-05 -1.297228e-04  2.506106e-04  1.023235e-03  8.959927e-07
[6,]  -0.0770281  2.291029e-05 -6.277138e-05 -2.944162e-04  4.321574e-05  8.959927e-07  1.755231e-05
>
```

5 GMM Implementation

Here, we estimate the logit model using GMM. We set the starting point of the GMM estimation to be the 2nd-stage estimates attained from the previous estimation. The implementation of the GMM estimation is provided below.

```
############ GMM estimation

gmm = function(theta,data,y,Z){
  # GMM estimation function
  # Inputs:
  # data: here, X matrix
  # theta: starting pt of vector of parameters, use 2SLS estimates here
  # y: vector of outcomes (here, mean utility)
  # Z: matrix of instruments
  # Output: GMM val

  residuals = y - as.matrix(data)%*%as.matrix(theta)

  # Projection matrix: Z(Z'Z)^{-1}Z'

  proj_matrix = Z%*%ginv(t(Z)%*%Z)%*%t(Z)

  weight = ginv(t(Z)%*%Z)

  gmm_val = t(residuals)%*%Z%*%weight%*%t(Z)%*%residuals

  return(gmm_val)
}

######## a) GMM estimation, Cost-IV
```

```
mean_utility_col = cbind(df$meanutility)

gmm_cost <- nlm(gmm,sls_estimates_Z_cost,X,mean_utility_col,Z_cost,print.level = 2,
    hessian=TRUE)

# Standard Error

H_cost=gmm_cost$hessian
V_cost=solve(H_cost)
stderr_cost=sqrt(diag(V_cost))

######## b) GMM estimation, Own-IV

gmm_own <- nlm(gmm,sls_estimates_Z_own,X,mean_utility_col,Z_own,print.level = 2,
    hessian=TRUE)

# Standard Error

H_own=gmm_own$hessian
V_own=solve(H_own)
stderr_own=sqrt(diag(V_own))


######## c) GMM estimation, Other-IV

gmm_other <- nlm(gmm,sls_estimates_Z_other,X,mean_utility_col,Z_other,print.level = 2,
    hessian=TRUE)

# Standard Error

H_other=gmm_other$hessian
```

```
V_other=solve(H_other)
stderr_other=sqrt(diag(V_other))


######## d) GMM estimation, All-IV

gmm_all <- nlm(gmm,sls_estimates_Z_all,X,mean_utility_col,Z_all,print.level = 2,
    hessian=TRUE)

# Standard Error

H_all=gmm_all$hessian
V_all=solve(H_all)
stderr_all=sqrt(diag(V_all))
```

The results of the GMM algorithm in each IV scenario is included below. We note that the 2nd-stage estimates were good starting points in every scenario except the all-IV case; this is evidenced in the fact that the GMM algorithm took only 1 iteration to run in the other three scenarios, but took 22 iterations to run in the all-IV scenario. The minimum values found at the top of each printout gives the value of Hansen's J test of overidentifying restrictions. The fact that all four minimum values are positive suggests that the model in each IV scenario is overidentified.

```
> gmm_cost
$`minimum`
[1] 1.376648e-19

$estimate
[1] -0.5253985 -1.2946500 -0.2828222  0.5856917 -1.1232258 -0.1369587

$gradient
[1] -3.166614e-08 -1.246468e-07 -7.530719e-09 -6.557705e-08 -4.146396e-08 -3.691905e-07

$hessian
           [,1]      [,2]       [,3]      [,4]      [,5]      [,6]
[1,]  3644.9215  14349.04   866.5113  7551.498  4771.822  42507.44
[2,] 14349.0408  56488.98  3411.0505 29729.874 18784.886 167346.14
[3,]   866.5113   3411.05   206.0300  1794.873  1134.502  10104.08
[4,]  7551.4979  29729.87  1794.8731 15648.831  9885.223  88079.71
[5,]  4771.8224  18784.89  1134.5025  9885.223  6247.381  55645.99
[6,] 42507.4397 167346.14 10104.0785 88079.714 55645.990 495773.93

$code
[1] 3

$iterations
[1] 1



> gmm_own
$`minimum`
[1] 3.868533e-23

$estimate
[1] -0.39171315 -1.02344370  0.04359612 -0.93521123 -0.56402968 -0.03233887

$gradient
[1] -1.792789e-10 -4.632801e-10  2.717336e-11 -3.292483e-10 -2.939900e-10 -8.011170e-11

$hessian
          [,1]       [,2]       [,3]       [,4]      [,5]      [,6]
[1,]  4125.559  16445.481  1024.2933   8523.901  5466.836  49660.50
[2,] 16445.481  66668.328  4372.0889  33760.962 21683.473 207508.20
[3,]  1024.293   4372.089   573.4822   2189.951  1338.603  18542.27
[4,]  8523.901  33760.962  2189.9513  18518.541 11040.767 100927.09
[5,]  5466.836  21683.473  1338.6028  11040.767  7354.220  65654.36
[6,] 49660.495 207508.204 18542.2687 100927.092 65654.361 740875.48

$code
[1] 3

$iterations
[1] 1
```

```
> gmm_other
$`minimum`
[1] 9.677435e-21

$estimate
[1] -1.26415832 -0.20572929  0.59665951  0.06559123 -2.32958061 -0.22867076

$gradient
[1] -1.423890e-09 -3.908715e-09 -6.478558e-10 -3.014296e-09 -1.875192e-09 -1.990542e-08

$hessian
          [,1]       [,2]       [,3]       [,4]      [,5]      [,6]
[1,]  4407.149  17486.399  1072.0394   9146.144  5784.180  51932.84
[2,] 17486.399  69687.868  4344.3569  36561.451 22893.330 207399.50
[3,]  1072.039   4344.357   361.7781   2445.485  1361.739  14010.75
[4,]  9146.144  36561.451  2445.4847  19764.170 11890.140 110605.39
[5,]  5784.180  22893.330  1361.7387  11890.140  7613.237  67541.87
[6,] 51932.842 207399.501 14010.7455 110605.389 67541.873 630984.52

$code
[1] 3

$iterations
[1] 1



> gmm_all
$`minimum`
[1] 9.234029

$estimate
[1] -12.8701350   0.1084292  -0.1794150   0.4780466   3.6941111  -0.0770281

$gradient
[1] 1.255307e-09 5.080381e-09 4.352074e-10 2.131628e-09 1.365647e-09 1.385558e-08

$hessian
          [,1]       [,2]       [,3]       [,4]      [,5]      [,6]
[1,]  4408.258  17487.533  1070.2027   9145.622  5786.282  51924.09
[2,] 17487.533  70816.701  4568.4735  36280.171 22866.819 216569.05
[3,]  1070.203   4568.474   585.3575   2313.976  1390.839  18991.41
[4,]  9145.622  36280.171  2313.9756  20072.375 11748.435 106603.08
[5,]  5786.282  22866.819  1390.8392  11748.435  7715.489  68234.41
[6,] 51924.093 216569.049 18991.4084 106603.079 68234.415 762382.83

$code
[1] 3

$iterations
[1] 1
```

6 Price Markup and Marginal Cost

Here, we examine the price markup and marginal cost information for the cost-IV scenario only. The code used to calculate these two quantities is included below. The results are included below.

```
############ Marginal cost, markup for Cost-IV

alpha = gmm_cost$estimate[6]
own_price = alpha*mean(df$p_real%*%(1-df$meanutility))
part=alpha*mean(df$meanutility%*%(1-df$meanutility))
marg_cost=mean(df$p_real+alpha*(1-df$meanutility))
markup=mean((1-df$meanutility)- alpha)

######### END OF CODE
```

We find a marginal cost value of \$10.59, and a markup value of \$8.69.