MKTG 551-3, Homework 4

Rayyan Sayeed

August 3, 2018

1 Introduction

Here, we implement a heterogeneous logit demand model by extending the model used in HW3. The data file is the same. Our model includes unobserved heterogeneity in the coefficients. Through implementation of this model, we obtain the estimate for θ, and the objective function value at the minimum for each IV scenario. We then use this new estimate for θ as the starting point for GMM estimation using the same GMM implementation used in HW3, and compare results.

2 Code

Beyond the HW3 code, the above procedure is implemented in the code below:

```
######## START heterogeneous logit model

# Define Q as matrix of random normals

set.seed(1234)
R = 50; L = 6
Q.sobol = t(sobol(n=R, dim=L, normal=TRUE, scrambling=3, seed=1234))



# Create weight matrices, W, for each IV case
```

```
W_cost = ginv(t(Z_cost)%*%Z_cost)
W_own = ginv(t(Z_own)%*%Z_own)
W_other = ginv(t(Z_other)%*%Z_other)
W_all = ginv(t(Z_all)%*%Z_all)

# Redefine X matrix as data frame for convenience...

X_new = cbind(ones,df[,c("hp2wt","air","mpd","car_size","p_real")])


######## BEGIN FUNCTION

# Function uses a contraction mapping to get out an estimate of theta vector, returns
    min value estimate of obj. fn.

# Inputs are starting vector theta, mean utility vector delta, weight matrix W, &
    instrument matrix Z
# Here, set input theta to be sls estimates for that IV scenario

min_val_fn = function(theta, delta, W, Z){

  # Make "current" large enough to enter while loop
  current = as.matrix(c(rep(200,nrow(df))))

  last = as.matrix(delta)

  # Initialize step
  step = 0

  mu = as.matrix(X_new) %*% diag(as.vector(theta)) %*% (Q.sobol) #2217x50
  reps = as.matrix(table(df$year))
```

```
# Define tolerance and iteration

tol = 1e-6
iter = 200

###### BEGIN WHILE LOOP
while(max(abs(current - last)) > tol & step < 200 ){

  if(step == 0){
    last = delta
  }
  else{
    last = current
  }

  # Monte Carlo
  mat = mu + last[row(mu)]
  nums = exp(mat)

  # Sum over all products by year, remove year column

  denoms = 1 + aggregate(as.matrix(nums) ~ df$year, nums, FUN = sum)
  denoms = denoms[,2:ncol(denoms)]

  # New denoms for use in for loop below, 2217x50
  newdenoms = matrix(rep(0,nrow(df)*ncol(denoms)),ncol = ncol(denoms))

  for (i in 1:ncol(newdenoms)) {
    newdenoms[,i] = rep(denoms[,i],times = as.vector(reps))
  }
  # s is Pre-Monte Carlo mean
  s = nums/newdenoms

  # Average across rows, 2217x1 vector
  s = apply(X = s, MARGIN = 1, FUN = mean)

  print(dim(exp(last)*df$mktshare))
  current = log((exp(last) * df$ownshare)/s)
  current = as.matrix(current)

  # Increment step
  step = step+1
}

###### END WHILE LOOP

# After the while loop is broken, "current" should have convergent values of delta

delta = current
# Finding theta1

xtilda = X_new # xtilda is a data frame!!
xtilda$p_real = -xtilda$p_real
xtilda=as.matrix(xtilda) # change xtilda to a matrix

part1 = ginv(t(xtilda)%*%Z%*%ginv(W)%*%t(Z)%*%xtilda)
part2 = t(xtilda) %*% Z %*% ginv(W)
part3 = t(Z) %*% delta
theta1 = part1%*% part2%*% part3
#theta1 = ginv(t(xtilda)%*%Z%*%ginv(W)%*%t(Z)%*%xtilda) %*% t(xtilda) %*% Z %*%
    ginv(W) %*% t(Z) %*% delta
theta1 = as.matrix(theta1)
```

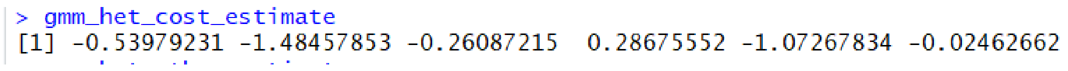
```
  # Finishing the GMM outer loop

  xi = as.matrix(delta - xtilda%*%theta1)

  # non-conformable argument error occurs in following line:
  return(t(t(xi)%*%Z%*%ginv(W)%*%t(Z)%*%xi))
}

############# END FUNCTION

# Calculate theta1 for each IV scenario
ests = gmm_cost$estimate
#test_vec=matrix(1,nrow=6,ncol=1)

# Finding obj. fn. minimums in each IV case, using sls estimates as starting point

min_cost=min_val_fn(sls_estimates_Z_cost, mean_utility_col,W_cost,Z_cost)
min_own=min_val_fn(sls_estimates_Z_own, mean_utility_col, W_own,Z_own)
min_other=min_val_fn(sls_estimates_Z_other,mean_utility_col,W_other,Z_other)
min_all=min_val_fn(sls_estimates_Z_all,mean_utility_col,W_all,Z_all)

### Rewrite out function to return theta1

theta1_fn=function(theta,delta,W,Z){

  # Make "current" large enough to enter while loop
  current = as.matrix(c(rep(200,nrow(df))))

  last = as.matrix(delta)
```

```
# Initialize step
step = 0

mu = as.matrix(X_new) %*% diag(as.vector(theta)) %*% (Q.sobol) #2217x50
reps = as.matrix(table(df$year))

# Define tolerance and iteration

tol = 1e-6
iter = 200

###### BEGIN WHILE LOOP
while(max(abs(current - last)) > tol & step < 200 ){

  if(step == 0){
    last = delta
  }
  else{
    last = current
  }

  # Monte Carlo
  mat = mu + last[row(mu)]
  nums = exp(mat)

  # Sum over all products by year, remove year column

  denoms = 1 + aggregate(as.matrix(nums) ~ df$year, nums, FUN = sum)
  denoms = denoms[,2:ncol(denoms)]

  # New denoms for use in for loop below, 2217x50
  newdenoms = matrix(rep(0,nrow(df)*ncol(denoms)),ncol = ncol(denoms))
  for (i in 1:ncol(newdenoms)) {
    newdenoms[,i] = rep(denoms[,i],times = as.vector(reps))
  }

  # s is Pre-Monte Carlo mean
  s = nums/newdenoms

  # Average across rows, 2217x1 vector
  s = apply(X = s, MARGIN = 1, FUN = mean)

  print(dim(exp(last)*df$mktshare))
  current = log((exp(last) * df$ownshare)/s)
  current = as.matrix(current)

  # Increment step
  step = step+1
}

###### END WHILE LOOP

# After the while loop is broken, "current" should have convergent values of delta

delta = current
# Finding theta1

xtilda = X_new # xtilda is a data frame!!
xtilda$p_real = -xtilda$p_real
xtilda=as.matrix(xtilda) # change xtilda to a matrix

part1 = ginv(t(xtilda)%*%Z%*%ginv(W)%*%t(Z)%*%xtilda)
part2 = t(xtilda) %*% Z %*% ginv(W)
```

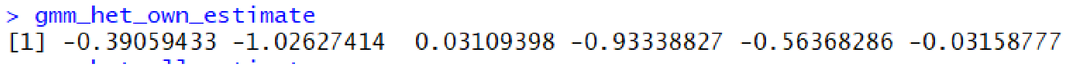
```
  part3 = t(Z) %*% delta

  theta1 = part1%*% part2%*% part3

  theta1 = as.matrix(theta1)


  return(theta1)


}


# Getting theta1 for each IV case for use in gmm estimation


theta1_cost=theta1_fn(sls_estimates_Z_cost, mean_utility_col, W_cost, Z_cost)

theta1_own=theta1_fn(sls_estimates_Z_own, mean_utility_col, W_own, Z_own)

theta1_other=theta1_fn(sls_estimates_Z_other, mean_utility_col, W_other, Z_other)

theta1_all=theta1_fn(sls_estimates_Z_all, mean_utility_col, W_all, Z_all)


# Use above theta1 vectors as starting point for GMM estimation, using hw3 gmm function
    written above


gmm_het_cost = nlm(gmm,theta1_cost,X,mean_utility_col,Z_cost,print.level = 2,
    hessian=TRUE)

gmm_het_own = nlm(gmm,theta1_own,X,mean_utility_col,Z_own,print.level = 2, hessian=TRUE)

gmm_het_other = nlm(gmm,theta1_other,X,mean_utility_col,Z_other,print.level = 2,
    hessian=TRUE)

gmm_het_all = nlm(gmm,theta1_all,X,mean_utility_col,Z_all,print.level = 2, hessian=TRUE)


gmm_het_cost_estimate=gmm_het_cost$estimate

gmm_het_own_estimate=gmm_het_own$estimate

gmm_het_other_estimate=gmm_het_other$estimate

gmm_het_all_estimate=gmm_het_all$estimate
```

3 Results

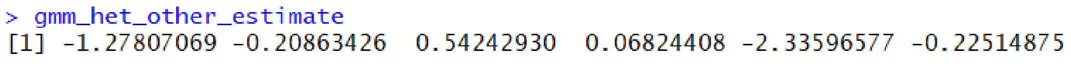
The implementation of the heterogeneous model with random coefficients yields the following estimates of θ for each IV scenario:



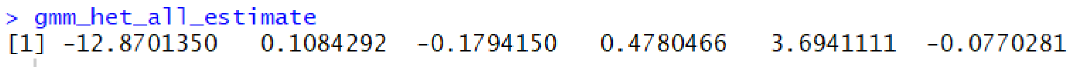
```
> gmm_het_cost_estimate
[1] -0.53979231 -1.48457853 -0.26087215  0.28675552 -1.07267834 -0.02462662
```



```
> gmm_het_own_estimate
[1] -0.39059433 -1.02627414  0.03109398 -0.93338827 -0.56368286 -0.03158777
```



```
> gmm_het_other_estimate
[1] -1.27807069 -0.20863426  0.54242930  0.06824408 -2.33596577 -0.22514875
```



```
> gmm_het_all_estimate
[1] -12.8701350   0.1084292  -0.1794150   0.4780466   3.6941111  -0.0770281
```

Further, the estimation routine returned the following values for the objective function minimums in each IV scenario:

$$min_{cost} = 2.04e{-}21$$

$$min_{own} = 3.932259e - 07$$

$$min_{other} = 3.564756e - 06$$

$$min_{all} = 3.932259e - 07$$