

Economics 631 IO - Fall 2019

Problem Set 3

Nathan Mather and Tyler Radler

December 3, 2019

1 Production Function Estimation

1.1 Summary Statistics

Summary Statistics – Mean and Variance

Variable	Full Mean	Bal. Mean	Tstat	Full VAR	Bal. VAR	Fstat
ldsal	5.67	6.91	-17.15	3.84	3.38	1.14
lemp	1.26	2.41	-17.94	3.15	2.63	1.20
ldnpt	4.47	5.92	-17.81	4.91	4.23	1.16
ldrst	3.40	4.89	-19.60	4.12	3.72	1.11
ldrnd	1.79	3.22	-18.42	4.21	3.98	1.06
ldinv	2.67	4.07	-17.27	4.71	4.22	1.12

Summary Statistics – Min and Max

Variable	Full Min	Bal. Min	Full max	Bal. max
ldsal	-0.86	1.66	11.70	11.70
lemp	-3.77	-2.07	6.73	6.73
ldnpt	-1.39	0.81	11.11	11.11
ldrst	-4.29	0.06	9.97	9.97
ldrnd	-5.31	-2.71	8.43	8.43
ldinv	-3.84	-2.08	8.99	8.89

There appear to be significant differences between the full sample and the balanced panel. In particular, the mean of each of the variables in the balanced panel is higher than that in the full sample, suggesting that firms which stay in the sample for the entire time period are generally larger and invest more. This makes sense, as we'd expect firms who eventually exit the market to be the least productive firms, and therefore produce and invest less. The full panel also has higher variance than the balanced panel, which would make sense as we are including firms with generally lower values relative to the balanced panel, which should increase the variance. The minimum values are similarly lower in the full sample relative to the balanced panel, and the maximum values all appear in the balanced panel. Broadly it seems clear that the balanced panel is a selected sample.

1.2 Production Function Estimation

Production Function Coefficients Balanced Panel

Parm	Statistic	Std.Error
B_0	0.76	0.80
K	0.24	0.22
L	0.45	0.21
P	1.00	0.02
RD	0.36	0.10

Production Function Coefficients Unbalanced Panel

Parm	Statistic	Std.Error
B_0	0.76	0.72
K	0.37	0.14
L	0.30	0.21
P	0.97	0.02
RD	0.38	0.06

The production function estimates are reported above, with the bootstrapped standard errors reported on the right. A few things – we interpreted “ignoring the selection issue” to mean using the balanced panel and not worrying about the fact this panel should consist of firms with the highest productivity. We were a bit unsure about this, however, so we included the results for the unbalanced Panel as well.

In our reported results we Blocked the bootstrap at the firm level. That is we selected firms randomly for our sample and included all observations for each firm. We have also experimented with iid bootstrapping at the observation level. Our concern with the latter was figuring out how to deal with the lagged variables.

We weren’t sure what to make of the ρ being so close to 1. This appears to be the case regardless of which sample we use. This means that any productivity shock is basically entirely persistent. If this were the case we could potentially treat the productivity shocks as a firm-specific fixed effect we could get rid of by differencing.

$$\mathbb{E}[w_{it} - w_{it-1}] = \mathbb{E}[w_{it-1} * \rho + \xi_{it} - w_{it-1}] \approx \mathbb{E}[\xi_{it}] = 0$$

This seems weird, although this potentially points to an error in our code.

2 Appendix

2.1 R Code

pset 3 631

```
#####  
# ==== IO Pset 3 ====  
#####  
  
#####  
# ==== load packages/data ====  
#####  
  
rm(list = ls(pos = ".GlobalEnv"), pos = ".GlobalEnv")  
options(scipen = 999)  
cat("\f")  
  
library(package)  
library(data.table)  
library(xtable)  
library(boot)  
library(broom)  
library(rmarkdown)  
  
# set option for who is running this  
opt_nate <- TRUE  
  
# load data and set directories  
if(opt_nate){  
  # load data  
  gmdt <- fread("c:/Users/Nmath_000/Documents/MI_school/Third Year/Econ 631/ps3/GMdata.csv")  
  f_out <- "c:/Users/Nmath_000/Documents/Code/Econ_631/ps3/"  
  
  # if running on tyler's computer  
}else{  
  # load data from tyler's locaiton  
  gmdt <- fread("C:/Users/tyler/Box/coursework/Econ_631/ps3/GMdata.csv")  
  f_out <- "C:/Users/tyler/Box/coursework/Econ_631/ps3/"  
  
}  
  
#####  
# ==== summary stats ====  
#####  
  
#####  
# ==== Make balanced panel ====
```

```

#####

# make the balanced panel version of data
# first get number of years by firm
gmdt[, num_years := .N, index]

# check how many are in each group
gmdt[, .N, num_years]

# make balanced panel
gmdt_b <- gmdt[num_years == max(gmdt$num_years)]

# drop variable
gmdt[, num_years := NULL]
gmdt_b[, num_years := NULL]

#####
# ==== do comparison ====
#####

# make list of variables
vars <- grep("l", colnames(gmdt_b), value = TRUE)

# get each column of summart stats
sum_stats_li <- list()
sum_stats_li[[1]] <- data.table(Variable = vars)
# means
sum_stats_li[[2]] <- gmdt[, list("Full Mean" = lapply(.SD, mean)), .SDcols = vars]
sum_stats_li[[3]] <- gmdt_b[, list("Bal. Mean" = lapply(.SD, mean)), .SDcols = vars]

# t.test for mean equlity
# function to do it
tstat_fun <- function(var_i){
  res <- t.test(gmdt[, get(var_i)], gmdt_b[, get(var_i)])

  return(res$statistic)
}

# apply over variables
sum_stats_li[[4]] <- data.table( "Tstat" = unlist(lapply(vars, tstat_fun)))

# varianvce
sum_stats_li[[5]] <- gmdt[, list("Full VAR" = lapply(.SD, var)), .SDcols = vars]
sum_stats_li[[6]] <- gmdt_b[, list("Bal. VAR" = lapply(.SD, var)), .SDcols = vars]

# fstat for equalit of variance
ftest_fun <- function(var_i){
  res <- var.test(gmdt[, get(var_i)], gmdt_b[, get(var_i)])

  return(res$statistic)
}

```

```

}

#apply over variables
sum_stats_li[[7]] <- data.table( "Fstat" = unlist(lapply(vars, ftest_fun)))

#min
sum_stats_li[[8]] <- gmdt[, list("Full Min" = lapply(.SD, min)), .SDcols = vars]
sum_stats_li[[9]] <- gmdt_b[, list("Bal. Min" = lapply(.SD, min)), .SDcols = vars]
#max
sum_stats_li[[10]] <- gmdt[, list("Full max" = lapply(.SD, max)), .SDcols = vars]
sum_stats_li[[11]] <- gmdt_b[, list("Bal. max" = lapply(.SD, max)), .SDcols = vars]

# get variance
sum_stats <- do.call(cbind, sum_stats_li)
#check it out!
sum_stats

#=====#
# ==== save table to tex ====
#=====#

# split this into two tables
sum_stat_1 <- sum_stats[, 1:7, with=FALSE]
sum_stat_2 <- sum_stats[, c(1, 8:11), with = FALSE]

# save summary stats
print(xtable(sum_stat_1, type = "latex"),
      file = paste0(f_out, "sum_stats_1.tex"),
      include.rownames = FALSE,
      floating = FALSE)

print(xtable(sum_stat_2, type = "latex"),
      file = paste0(f_out, "sum_stats_2.tex"),
      include.rownames = FALSE,
      floating = FALSE)

#=====#
# ==== Estimate Production Function ==== #
#=====#

# Step 1: Get measurement error from second-order polynomial
# (including interactions) of emp, dnpt, drst and investment
setnames(gmdt, c("ldsai", "ldnpt", "ldrst"), c("lsales", "lcap", "lrdcap"))

gmdt[, dummy:=1]

```

```

# do it for balanced too just for fun
setnames(gmdt_b, c("ldsai", "ldnpt", "ldrst"), c("lsales", "lcap", "lrdcap"))

gmdt_b[,dummy:=1]

#####
# === internal functions ===
#####

# in_data <- gmdt_b
get_resid_fun <- function(in_data){

  dt_copy <- copy(in_data)

  X <- as.matrix(dt_copy[,.(dummy, lemp,lcap,lrdcap,ldinv, lemp*lemp,lemp*lcap, lemp*lrdcap,
    lemp*ldinv,lcap*lcap,lcap*lrdcap,lcap*ldinv,lrdcap*lrdcap,lrdcap*ldinv,ld

  Y <- as.matrix(dt_copy[,.(lsales)])

  # Get the coefficients of all these interactions
  beta1 <- solve(t(X)%*%X)%*%t(X)%*%Y

  # Get the fitted values
  theta <- X%*%beta1

  # Add the fitted values to the datatable
  dt_copy[,theta:= theta]

  # get lags of emp, cap, rdcap and theta, remove observations without lag values
  cols = c("lemp","lcap","lrdcap", "theta")
  anscols = paste("lag", cols, sep="_")
  dt_copy[, (anscols) := shift(.SD, 1, NA, "lag"), .SDcols=cols, index]

  dt_copy <- dt_copy[!is.na(lag_lemp),]

  return(dt_copy)

}

# # ttest the funciton
# gmdt2 <- get_resid_fun(gmdt)
#

# function to run 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,
  Y.in = Y,
  X.in = X,
  lX.in = lX,
  ltheta.in = ltheta,

```

```

        Z.in = Z,
        W.in = W){

beta <- as.matrix(parm_vector.in[1:4])
rho <- parm_vector.in[5]

current.resid <- Y.in - X.in**beta

lag.w <- as.matrix(ltheta.in) - lX.in**beta

m <- current.resid - rho*lag.w

m <- as.matrix(m)

distance <- t(Z.in)**m

result <- t(distance/length(m))** W.in ** (distance/length(m))

# get function value
return(result)
}

# TEST
## parms to test the function
# parm_vector <- c(1, .6, .2, .2, .8)
#
## Get X's and lag X's and Zs for GMM estimation
# X <- as.matrix(gmdt2[,.(dummy, lemp, lcap, lrdcap)])
# lX <- as.matrix(gmdt2[,.(dummy, lag_lemp, lag_lcap, lag_lrdcap)])
# Z <- as.matrix(gmdt2[,.(dummy, lag_lemp, lcap, lrdcap)])
# Y <- as.matrix(gmdt2[,.(lsales)])
# ltheta <- as.matrix(gmdt2[,.(lag_theta)])
# W <- diag(1, 4, 4)
#
## test it out
# f <- gmm_obj_f(parm_vector.in = parm_vector,
#               Y.in           = Y,
#               X.in           = X,
#               lX.in          = lX,
#               ltheta.in      = ltheta,
#               Z.in           = Z,
#               W.in           = W)
#
## Run the initial GMM using the identity matrix as the weighting matrix
# Results.step1 <- optim(par      = parm_vector,
#                       fn        = gmm_obj_f,
#                       Y.in      = Y,
#                       X.in      = X,
#                       lX.in     = lX,
#                       ltheta.in = ltheta,
#                       Z.in      = Z,
#                       W.in      = W)

# Function to calculate optimal weighting matrix

```

```

find.optimal.W <- function(results.in,
                           Y.in = Y,
                           X.in = X,
                           lX.in = lX,
                           ltheta.in = ltheta,
                           Z.in = Z){
  beta <- as.matrix(results.in$par[1:4])
  rho <- results.in$par[5]

  current.resid <- Y.in - X.in%%beta

  lag.w <- as.matrix(ltheta.in) - lX.in%%beta

  m <- current.resid - rho*lag.w

  m <- as.matrix(m)

  distance <- t(Z.in*cbind(m, m, m, m))%(Z.in*cbind(m, m, m, m))
  W <- distance/length(m)

  W.inv <- solve(W)
  return(W.inv)
}

# test
# # Get optimal weighting matrix
# W.opt <- find.optimal.W(results.in = Results.step1,
#                           Y.in = Y,
#                           X.in = X,
#                           lX.in = lX,
#                           ltheta.in = ltheta,
#                           Z.in = Z)
#
# # Run again with optimal weighting matrix
# Results.final <- optim(par      = parm_vector,
#                         fn       = gmm_obj_f,
#                         Y.in = Y,
#                         X.in = X,
#                         lX.in = lX,
#                         ltheta.in = ltheta,
#                         Z.in = Z,
#                         W.in = W.opt)

#####
# ==== boot function ====
#####

# in_data <- gmdt
# sample <- 1:nrow(in_data)
to_boot_fun <- function(in_data, sample){

  boot_dt <- in_data[sample]

```



```

# get resids
boot_dt <- get_resid_fun(boot_dt)

parm_vector <- c(1, .6, .2, .2, .8)

# Get X's and lag X's and Zs for GMM estimation
X <- as.matrix(boot_dt[,.(dummy, lemp, lcap, lrdcap)])
lX <- as.matrix(boot_dt[,.(dummy, lag_lemp, lag_lcap, lag_lrdcap)])
Z <- as.matrix(boot_dt[,.(dummy, lag_lemp, lcap, lrdcap)])
Y <- as.matrix(boot_dt[,.(lsales)])
ltheta <- as.matrix(boot_dt[,.(lag_theta)])
W <- diag(1, 4, 4)

# test it out
f <- gmm_obj_f(parm_vector.in = parm_vector,
               Y.in          = Y,
               X.in          = X,
               lX.in         = lX,
               ltheta.in     = ltheta,
               Z.in          = Z,
               W.in          = W)

# Run the initial GMM using the identity matrix as the weighting matrix
Results.step1 <- optim(par      = parm_vector,
                      fn       = gmm_obj_f,
                      Y.in     = Y,
                      X.in     = X,
                      lX.in    = lX,
                      ltheta.in = ltheta,
                      Z.in     = Z,
                      W.in     = W)

# Get optimal weighting matrix
W.opt <- find.optimal.W(results.in = Results.step1,
                        Y.in = Y,
                        X.in = X,
                        lX.in = lX,
                        ltheta.in = ltheta,
                        Z.in = Z)

# Run again with optimal weighting matrix
Results.final <- optim(par      = parm_vector,
                      fn       = gmm_obj_f,
                      Y.in     = Y,
                      X.in     = X,
                      lX.in    = lX,
                      ltheta.in = ltheta,
                      Z.in     = Z,
                      W.in     = W.opt)

return(Results.final$par)
}

```

```

# to_boot_fun(gmdt, 1:(nrow(gmdt)))

#####
# ==== Do bootstrapping ====
#####

#####
# ==== write out own boot ====
#####

n_sim <- 1000

# block boot function to iterate
block_boot <- function(iteration, in_data){

  # if iteration is 1 use full sample
  if(iteration == 1){

    sample_i <- 1:1400
  }else{
    sample_i <- sample(1:1400, 1400, replace = TRUE)
  }

  # grab a random sample
  dt_i <- in_data[index %in% sample_i]

  # run function on subsample
  res_i <- to_boot_fun(dt_i, 1:nrow(dt_i))

  return(data.table(var = parms , res = res_i))
}

# apply function n_sim times
re_list_b <- lapply(1:n_sim, block_boot, in_data = gmdt_b)

# do it again on unbalanced sample
re_list_u <- lapply(1:n_sim, block_boot, in_data = gmdt)

# bind the lists together
re_dt_b <- rbindlist(re_list_b)
re_dt_u <- rbindlist(re_list_u)

# Now get standard errors for balanced
re_dt_b[, mean := mean(res), var]
re_dt_b[, to_sum := (res-mean)^2 ]
out_dt_b <- re_dt_b[, list(std.error = sqrt(1/(n_sim - 1) * sum(to_sum))), var]

```

```

# Now get standard errors fo unbalanced
re_dt_u[, mean := mean(res), var]
re_dt_u[, to_sum := (res-mean)^2 ]
out_dt_u <- re_dt_u[, list(std.error = sqrt(1/(n_sim - 1) * sum(to_sum))), var]

# Now add in estiamtes
est_b <- re_list_b[[1]]
est_u <- re_list_u[[1]]
out_dt_b <- merge(est_b, out_dt_b, "var")
out_dt_u <- merge(est_u, out_dt_u, "var")

# change the variable names
setnames(out_dt_b, colnames(out_dt_b), c("Parm", "Statistic", "Std.Error"))
setnames(out_dt_u, colnames(out_dt_u), c("Parm", "Statistic", "Std.Error"))

# save table
print(xtable(out_dt_b, type = "latex"),
      file = paste0(f_out, "boot_res.tex"),
      include.rownames = FALSE,
      floating = FALSE)

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

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

# load data and set directories
if(opt_nate){
  rmarkdown::render(input = "C:/Users/Nmath_000/Documents/Code/Econ_631/ps3/ps3_r_markdown.Rmd",
                    output_format = "pdf_document",
                    output_file = paste0(f_out, "assignment_3_r_code_pdf.pdf"))

  # if running on tyler's computer
}else{
  rmarkdown::render(input = "C:/Users/tyler/Box/coursework/Econ_631/ps3/ps3_r_markdown.Rmd",
                    output_format = "pdf_document",
                    output_file = paste0(f_out, "assignment_3_r_code_pdf.pdf"))
}

```