ME_Assignment_4

Jia Ru

2016-03-22

(I use R instead of STATA for this homework)

Question 1

Consider the following wage rate equation specifications:

$$w_{it} = \gamma w_{i,t-1} + \beta' x_{it} + \delta d_{it} + \alpha_i + u_{it}$$

$$w_{it} = \beta' x_{it} + \delta d_{it} + \alpha_i + u_{it}$$

where x_{it} stand for years of schooling, experience, industry dummies and occupational dumming, and d_{it} is the union status dummy.

estimate (1) and (2) by:
1. covariance method (Least Squares dummy variable)
2. generalized method of moments estimator
3. Random Effects Estimator

Use your results to answer the following questions:
(a) Your preferred specification.
(b) Does union membership raise wage rate?

```
##########################################################
# Q1
#########################################################

# Import data
rm(list = ls())
library(readxl)
data <- read_excel(path = "data_assignment4.xls", col_names = TRUE)

# generate lag_LWAGE
data <- data[order(data$id,data$time),]  # sort
data$lag_LWAGE <- rep(NA,nrow(data))
for (i in 1:nrow(data)){
  if (i%%7==1) next
  data[i,"lag_LWAGE"] <- data[i-1,"LWAGE"]
}
# set panel data structure
library(plm)
pdata <- plm.data(x = data,indexes = c("id","time"))
```

(1) covariance method (Least Squares dummy variable)

```r
#  (1) LSDV

lsdv <- lm(LWAGE~ED+EXP+IND+OCC+UNION+factor(id),data = data)
lsdv_lag <- lm(LWAGE~ED+EXP+IND+OCC+UNION+factor(id)+lag_LWAGE,data = data)

summary(lsdv_lag)$coefficients[1:6,]
summary(lsdv)$coefficients[1:6,]
```

```
##                Estimate  Std. Error    t value      Pr(>|t|)
## (Intercept)  3.74552012 0.315090685 11.8871179  7.207466e-32
## ED           0.07857988 0.028731307  2.7349913  6.275242e-03
## EXP          0.07651207 0.002318759 32.9969982 1.055277e-203
## IND          0.01259265 0.016705404  0.7538067  4.510250e-01
## OCC         -0.02704422 0.015185098 -1.7809708  7.501940e-02
## UNION        0.01820191 0.016487117  1.1040081  2.696790e-01
##                Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)  4.47462629 0.291439115 15.353554 1.446003e-51
## ED           0.10008975 0.027430646  3.648830 2.672098e-04
## EXP          0.09677881 0.001189885 81.334562 0.000000e+00
## IND          0.02018713 0.015580279  1.295685 1.951679e-01
## OCC         -0.02384978 0.013846470 -1.722445 8.507572e-02
## UNION        0.03420595 0.015047170  2.273248 2.307046e-02
```

(2) generalized method of moments estimator

```r
library(gmm)

gmm <- gmm(g = LWAGE~ED+EXP+IND+OCC+UNION,
           x = ~ED+EXP+IND+OCC+UNION,
           data = data)

gmm_lag <- gmm(g = LWAGE~ED+EXP+IND+OCC+UNION+lag_LWAGE,
           x = ~ED+EXP+IND+OCC+UNION+lag_LWAGE,
           data = data)

gmm$coefficients
gmm_lag$coefficients
```

```
## (Intercept)          ED         EXP         IND         OCC       UNION
##  5.51357587  0.07028259  0.01212397  0.10838322 -0.14879002  0.14411194
##   (Intercept)            ED           EXP           IND           OCC
##  0.6669967774  0.0087994828  0.0002018259  0.0235619691 -0.0226658954
##         UNION     lag_LWAGE
##  0.0145896230  0.8957378543
```

(3) Random Effects Estimator

```r
# (3) Random Effects

re <- plm(LWAGE~ED+EXP+IND+OCC+UNION, model = "random", data = pdata)
re_lag <- plm(LWAGE~ED+EXP+IND+OCC+UNION+lag_LWAGE, model = "random", data = pdata)

summary(re_lag)$coefficients
re_lag$ercomp  # estimation of the components of the errors of RE model

summary(re)$coefficients
re$ercomp
```

```
##                  Estimate   Std. Error    t-value     Pr(>|t|)
## (Intercept)  0.368600977 0.0379089967   9.723311 4.500404e-22
## ED           0.005446663 0.0011052008   4.928211 8.676635e-07
## EXP         -0.000419706 0.0002141759  -1.959632 5.011668e-02
## IND          0.017568651 0.0047506165   3.698183 2.204215e-04
## OCC         -0.011465674 0.0061317760  -1.869878 6.158270e-02
## UNION        0.006007399 0.0050617266   1.186828 2.353746e-01
## lag_LWAGE    0.949113832 0.0061341387 154.726503 0.000000e+00
##                    var   std.dev   share
## idiosyncratic  0.021979  0.148254  1.095
## individual    -0.001915        NA -0.095
## theta:  -0.4476
##                 Estimate  Std. Error    t-value     Pr(>|t|)
## (Intercept)  4.13875825 0.093716241 44.1626573 0.000000e+00
## ED           0.11137433 0.006400367 17.4012413 1.605957e-65
## EXP          0.05542692 0.001097691 50.4941092 0.000000e+00
## IND          0.01571681 0.017506697  0.8977599 3.693655e-01
## OCC         -0.04569672 0.016588057 -2.7547962 5.898424e-03
## UNION        0.06498127 0.017227059  3.7720468 1.641656e-04
##                   var std.dev share
## idiosyncratic 0.02353 0.15341 0.191
## individual    0.09939 0.31527 0.809
## theta:  0.8191
```

```r
# estimation result

library(texreg)

screenreg(
  l = list(lsdv,lsdv_lag,gmm,gmm_lag,re,re_lag),
  omit.coef = "id",
  custom.model.names = c("lsdv","lsdv_lag","gmm","gmm_lag","re","re_lag")
  )
```

```
##
## ==============================================================================================
##                      lsdv         lsdv_lag     gmm          gmm_lag      re           re_lag
## ----------------------------------------------------------------------------------------------
## (Intercept)             4.47 ***     3.75 ***     5.51 ***     0.67 ***     4.14 ***     0.37 ***
##                        (0.29)       (0.32)       (0.10)       (0.09)       (0.09)       (0.04)
## ED                      0.10 ***     0.08 **      0.07 ***     0.01 ***     0.11 ***     0.01 ***
```

```
##                          (0.03)        (0.03)        (0.01)        (0.00)        (0.01)        (0.00)
## EXP                       0.10 ***      0.08 ***      0.01 ***      0.00          0.06 ***     -0.00
##                          (0.00)        (0.00)        (0.00)        (0.00)        (0.00)        (0.00)
## IND                       0.02          0.01          0.11 ***      0.02 ***      0.02          0.02 ***
##                          (0.02)        (0.02)        (0.03)        (0.01)        (0.02)        (0.00)
## OCC                      -0.02         -0.03         -0.15 ***     -0.02 ***     -0.05 **      -0.01
##                          (0.01)        (0.02)        (0.03)        (0.01)        (0.02)        (0.01)
## UNION                     0.03 *        0.02          0.14 ***      0.01 **       0.06 ***      0.01
##                          (0.02)        (0.02)        (0.03)        (0.01)        (0.02)        (0.01)
## lag_LWAGE                               0.18 ***                    0.90 ***                    0.95 ***
##                                        (0.02)                      (0.02)                      (0.01)
## ----------------------------------------------------------------------------------------------------
## R^2                       0.91          0.91                                      0.40          0.91
## Adj. R^2                  0.89          0.89                                      0.40          0.91
## Num. obs.              4165          3570          4165          3570          4165          3570
## RMSE                      0.15          0.15
## Criterion function                                  0.00          0.00
## ====================================================================================================
## *** p < 0.001, ** p < 0.01, * p < 0.05
```

Answer the question:

(a) Your preferred specification.

I prefer the LSDV specification. As I argued in Assignment-3, there is endogeneity problem, i.e, the heterogeneity term u_i is correlated with covariates x_{it}, so both GMM and random effect model is not appropreate.

Also I think the equation (1) (with lag term of LWAGE) is more appropriate, the reason is same as above: FD(first-order difference) model eliminates the individual effect u_i.

(b) Does union membership raise wage rate?

According to the reasoning above, I consider `ladv_lag` model. The coefficent of `UNION` in the model is 0.01, not significant. So I think union membership does not necessarily raise wage rate.

Moreover, although in other models the coefficient is significant, it only stands for correlation relationship, not causal relationship.

Question 2

Consider the model

$$y_{it} = \gamma y_{i,t-1} + \alpha_i + u_{it}$$

$$\gamma = 0.5, \alpha_i \sim N(0,1), u_{it} \sim N(0,1)$$

Generate $200 + T$ observations of y_{it} and throw away the first 200 observations. Consider the case of $N = 200,\ T = 5$ and $N = 200,\ T = 54$.

Estimate γ by the
1. simple instrumental variable method
2. GMM
3. MLE
Construct the t-statitic for the null:$\gamma = 0.5$

Replicate the experiment 1000 times. Find the actual size based on different estimators using the critical value of 1.96. (for the nominal significance level of 5%).

ANS:

First I define a funciton to do **one time** simulation, then when I do 1000 times simulation I will just invoke this function.

```
###################################################################
# This chunk defines the one time simulation function :
# arguments: N(number of individuals) and TT(number of time)
# STEP_1  simulate DGP
# STEP_2  do IV, GMM, MLE regression and hypothesis test
# return: a list of 3, which indicates whether to reject H_0: gamma==0.05 in the three model specificat
###################################################################

rm(list=ls())

SM <- function(N,TT) {

################# DGP ##################################

#N <- 200
#TT <- 5 or 54
# in R, "T"" stands for boolean "True" , so use "TT" to escape.
gamma <- 0.5

# initialize data
data2 <- as.data.frame(matrix(NA, nrow = N*TT, ncol = 6), colnames)
names(data2) <- c("y","lag_y","a","u","id","time")

a <- rnorm(N)
for (i in 1:N ){

  data2[((i-1)*TT+1):(i*TT),"id"] <- i         # id
  data2[((i-1)*TT+1):(i*TT),"time"] <- 1:TT    # time
  data2[((i-1)*TT+1):(i*TT),"a"] <- a[i]       # for the same individual (i) alpha is same

  y <- rep(0,200+TT)     # initialize y_i   # vector
  u <- rnorm(200+TT)     # generate u_i     # vector
  for (t in 2:(200+TT)) {
      y[t] <- gamma*y[t-1] + a[i] + u[t]
  }

  data2[((i-1)*TT+1):(i*TT),"y"] <- y[201:(200+TT)]
  data2[((i-1)*TT+1):(i*TT),"lag_y"] <- y[200:(200+TT-1)]
  data2[((i-1)*TT+1):(i*TT),"u"] <- u[201:(200+TT)]

}
data2$id <- factor(data2$id) # in order to treat id as dummy in regression


######################## simple IV #######################
library(magrittr)
library(AER)
```

```r
iv <- ivreg(y~lag_y|lag_y, data = data2)
# equivalent to
lm <- lm(y~lag_y,data=data2)

summary(iv)
# hypothesis test
hyp <- linearHypothesis(iv,"lag_y=0.5",test = "F")
reject <- hyp$`Pr(>F)`[2]>0.05
iv_test <- as.numeric(reject) # whether reject H_0



######################## GMM ########################
gmm <- gmm(g = y~lag_y,
           x = ~lag_y,
           data = data2)
gmm
# hypothesis test
hyp <- linearHypothesis(gmm,"lag_y=0.5",test = "F")
reject <- hyp$`Pr(>F)`[2]>0.05
gmm_test <- as.numeric(reject) # whether reject H_0



######################## MLE  ########################
library(stats4)
y <- data2$y
x <- data2$lag_y

LL <- function(gamma, mu, sigma) {
    R = y - x * gamma
    R = suppressWarnings(dnorm(R, mu, sigma, log = TRUE))
    -sum(R)
}
mle <- mle(LL, start = list(gamma = 0.5, mu = 0, sigma=1))

mle

# construct t test
gamma_hat <- summary(mle)@coef["gamma","Estimate"]
gamma_se <- summary(mle)@coef["gamma","Std. Error"]
t <- (gamma_hat-gamma)/gamma_se
reject <-  abs(t)>1.96
mle_test <- as.numeric(reject)

return(list("iv"=iv_test,"gmm"=gmm_test,"mle"=mle_test))

}
```

Next, define a function of 1000 times simulation procedure. This function returns the simulated size of the hypothesis testing of the three regression methods

```r
###################################################################
# This chunk defines the 1000 times simulation function :
# arguments: N and TT, and n_sim=1000 is fixed.
# STEP_1  invoke function SM, replicated it 1000 times
# STEP_2  calculate the proportion that reject/accepts H_0, this is the simulated size.
# return: a list of 3, which is the simulated size.
###################################################################

n_sim <- 1000

SMsize <- function(N,TT) {
  N_v <- rep(N, times = n_sim)
  TT_v <- rep(TT, times = n_sim)

  m <- mapply(FUN=SM, N=N_v,TT=TT_v)
  mat <- matrix(
    unlist(m),
    ncol=n_sim, nrow=3, byrow=F,
    row.names(c("iv","gmm","mle"))
    )
  size <- rowSums(mat)/n_sim
  size3 <- list("size_iv"=size[[1]],
                "size_gmm"=size[[2]],
                "size_mle"=size[[3]]
                )
  return(size3)
}
```

Now invoke the function SMsize() to do two type of simulation and see the sizes:

```r
print("(1) N=200, T=5")
SMsize(N=200,TT=5)

print("(1) N=200, T=54")
SMsize(N=200,TT=54)
```

```
## [1] "(1) N=200, T=5"
## $size_iv
## [1] 0.7272727
##
## $size_gmm
## [1] 0.6363636
##
## $size_mle
## [1] 0.3636364
##
## [1] "(1) N=200, T=54"
## $size_iv
## [1] 0.1818182
##
## $size_gmm
## [1] 0.1818182
##
```

```
## $size_mle
## [1] 0.8181818
```

from the results we can see:
(1) when T is small, iv and gmm is closer to the "real world", while mle is not.
(2) when T is large, however, mle is performs better.