pset 5 675

```
#====================#
# ==== ps_5_675R ====
#====================#


#=========================================#
# ==== Load packages, clear workspace ====
#=========================================#


library(MASS)
library(data.table)
library(broom)
library(AER)
library(xtable)
library(Matrix)
library(doParallel)
library(foreach)

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

#========================#
# ==== Q2 simulation ====
#========================#

# set final run parm for n simulations and if it should save
final_run <- TRUE


  #=============================#
  # ==== Write sim function ====
  #=============================#

    # parms for funciton
    f_stat = 0
    n = 200
    sim = 1 # a tag for the simulation number

    # sim function
    sim_fun2 <- function(sim, f_stat, n = 200){

      # make gamma
      gamma <- sqrt(f_stat/n)

     # make mu vector
      mu = c(0,0,0)

      # make sigma matrix
```

```r
    sigma <- matrix(c(1,0,0,0,1,.99,0,1,.99), 3,3)

    # make data
    rdt <- mvrnorm(n, mu, sigma)

    # make it a data.table
    rdt <- data.table(rdt)
    setnames(rdt, colnames(rdt), c("z","u","v"))

    # back out x
    rdt[, x := gamma*z + v]

    # back out y given b=0
    rdt[, y := u]

    # run ols
    ols_res <- data.table(tidy(lm(y~x, data = rdt)))

    # make column of rejecting the null
    ols_res[, rej := as.numeric(abs(statistic) > 1.96)]

    # take what we need
    ols_res <- ols_res[term == "x", c("estimate", "std.error", "rej")]

    # add on ols suffix
    ols_res[, reg_type := "ols"]

    # melt data for matias table
    ols_res <- melt.data.table(id.vars = "reg_type", data = ols_res)

    # run first stage of 2sls to get f test
    fst_stg <- lm(x~z, data = rdt)
    f_stat <- summary(fst_stg)$fstatistic[1]

    # now run 2sls
    iv_reg <- ivreg(y ~ x | z , data = rdt)
    summary(iv_reg)
    iv_reg <- data.table(tidy(ivreg(y ~ x | z , data = rdt)))

    # compute rej
    iv_reg[, rej := as.numeric(abs(statistic) > 1.96)]

    # take what we need
    iv_reg <- iv_reg[term == "x", c("estimate", "std.error", "rej")]

    # throw in the f stat
    iv_reg[, f_stat := f_stat]

    # add 2sls indicator
    iv_reg[, reg_type := "2sls"]

    # melt data for matias table
    iv_reg <- melt.data.table(id.vars = "reg_type", data = iv_reg)
    # stack  these tables
    out_dt <- rbind(ols_res, iv_reg)

    # add sim number
    out_dt[, sim := sim]

    # ruturn that shiz
    return(out_dt[])

    }# end sim funciton




#===========================#
# ==== run sim funciton ====
#===========================#

  # time this sucker
  start_time <- Sys.time()

  # initialize list to store output
  sim_list <- list()
  for(f_stat_i in c(0,.25,9,99)){

    # get number of sims
    n_sims <- ifelse(final_run, 5000, 50)

    # apply the function 5000 times
    sim_out <- lapply(c(1:n_sims), sim_fun2, f_stat = f_stat_i, n = 200)

    # bind the results
    sim_out <- rbindlist(sim_out)

    # take mean, std, quantiles by group
    results <- sim_out[, list("mean" = mean(value),
                              "st.dev" = sd(value),
                              "quant .1" = quantile(value, .1),
                              "quant .5" = quantile(value, .5),
                              "quant .9" = quantile(value, .9)), by = c("reg_type", "variable")]

    # store results in a list
    sim_list[[paste0(f_stat_i)]] <- results

  }# end loop over gamams


  # check time
  end_time <- Sys.time()

  # print time
  print(paste0(round(as.numeric(end_time - start_time, units = "mins"), 3), " minutes to run"))
  #================================================#
  # ==== save out these tables into a tex file ====
  #================================================#

    # check if this is a final run
    if(final_run){

      # for each item in the list
      for(tab_i in ls(sim_list)){

        print(xtable(sim_list[[tab_i]], type = "latex"),
              file = paste0("C://Users/Nmath_000/Documents/Code/courses/econ 675/PS_5_tex/q2tab_fstat_"
              include.rownames = FALSE,
              floating = FALSE)
        }#end loop

    } # end if statement



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

  # clear enviroment
  rm(list = ls(pos = ".GlobalEnv"), pos = ".GlobalEnv")

  # load in data
  ak <- fread("C:/Users/Nmath_000/Documents/MI_school/Second Year/675 Applied Econometrics/hw/hw5/Angris


  #========================#
  # ==== 3.1 AK models ====
  #========================#

    #============================#
    # ==== regression set up ====
    #============================#


    # make YOB dummies
    ak[, .N, "YoB_ld"]
    for(year_i in unique(ak$YoB_ld)){

      ak[ ,temp := 0]
      ak[YoB_ld == year_i ,temp := 1]
      setnames(ak, "temp", paste0("d_YOB_ld_", year_i))

    }
    # get a list of all year dummies but one. Exclude the proper one to match coeffs
    year_dummies <- setdiff(grep("d_YOB", colnames(ak), value = TRUE), "d_YOB_ld_0")

    # make QoB dummies
```

```r
for(qob_i in unique(ak$QoB)){

  ak[ ,temp := 0]
  ak[QoB == qob_i ,temp := 1]
  setnames(ak, "temp", paste0("d_QoB_", qob_i))

}

# get qob dummy list.  Exclude the proper one to match coeffs
qob_dummies <- setdiff(grep("d_QoB", colnames(ak), value = TRUE), "d_QoB_1")

# make cross variables of year dummies and qob
#note there is almost certainly a better way to do this but here we are
inter_list <- NULL
for(d_year in year_dummies){

  for(d_qob in qob_dummies){

    ak[, temp := get(d_qob)*get(d_year)]
    setnames(ak, "temp", paste0(d_year, "X", d_qob))
    inter_list<- c(inter_list, paste0(d_year, "X", d_qob))
  }
}


# standard controls
# (i) race, (ii) marrital status, (iii) SMSA, (iv) dummies for
# region, and (iv) dummies for YoB ld.
std_cont <- c("non_white","married", "SMSA",
              "ENOCENT","ESOCENT", "MIDATL",
              "MT", "NEWENG", "SOATL", "WNOCENT",
              "WSOCENT",  year_dummies) # get year dummies but leave one out

# save extra controls
extra_cont <- c("age_q", "age_sq")

#=================#
# ==== ols 1 ====
#=================#

  # make the formula
  ols1_form <- as.formula(paste0("l_w_wage~educ +", paste(std_cont, collapse = " + ")))

  # run ols
  out_ols1 <- data.table(tidy(lm(ols1_form, data = ak)))

  # keep what I need
  out_ols1 <- out_ols1[term %chin% c("educ"), c("term", "estimate", "std.error")]
  out_ols1[, model := "OLS 1"]

#=================#
# ==== OlS 2 ====
#=================#
```

```
  # make the formula
  ols2_form <- as.formula(paste0("l_w_wage~educ +", paste(std_cont, collapse = " + "), " + ", paste

  #run ols
  out_ols2 <- data.table(tidy(lm(ols2_form, data = ak)))

  # keep what I need
  out_ols2 <- out_ols2[term %chin% c("educ"), c("term", "estimate", "std.error")]
  out_ols2[, model := "OLS 2"]

#================#
# ==== 2sls ====
#================#

  # write this part as a function so I can use it in 3.2
  # ACTUALLY, im gonna use different faster function but this is fine as a function too
  wrap_2sls <- function(in_data){

  #==================#
  # ==== 2sls 1 ====
  #==================#

    iv_form <- as.formula(paste0("l_w_wage~educ +", paste(std_cont, collapse = " + "),
                                 "| ",
                                 paste(std_cont, collapse = " + "), " + ", paste0(inter_list, colla
    iv_reg1 <- data.table(tidy(ivreg(iv_form , data = in_data)))

    # keep what I need
    iv_reg1 <- iv_reg1[term %chin% c("educ"), c("term", "estimate", "std.error")]
    iv_reg1[, model := "2sls 1"]

  #==================#
  # ==== 2sls 2 ====
  #==================#


    iv_form2 <- as.formula(paste0("l_w_wage~educ +", paste(std_cont, collapse = " + "), "+", paste0
                                 "| ",
                                 paste(std_cont, collapse = " + "),
                                 " + ", paste0(inter_list, collapse = " + "),
                                 "+", paste0(extra_cont, collapse = " + ")))
    iv_reg2 <- data.table(tidy(ivreg(iv_form2 , data = in_data)))

    # keep what I need
    iv_reg2 <- iv_reg2[term %chin% c("educ"), c("term", "estimate", "std.error")]
    iv_reg2[, model := "2sls 2"]

    # stack 2sls
    out_2sls <- rbind(iv_reg1, iv_reg2)

    return(out_2sls)

  }#end 2sls function
```

```
    # run function
    ak_2sls <- wrap_2sls(ak)

  #========================#
  # ==== output tables ====
  #========================#

    output_3.1 <- rbind(out_ols1, out_ols2, ak_2sls)
    setcolorder(output_3.1, c("model", "term", "estimate", "std.error"))

    # out put it

    print(xtable(output_3.1, type = "latex"),
          file = paste0("C://Users/Nmath_000/Documents/Code/courses/econ 675/PS_5_tex/q3.1_table.tex"
          include.rownames = FALSE,
          floating = FALSE)



#================#
# ==== Q 3.2 ====
#================#

  #=========================================#
  # ==== whats the fastest 2sls method? ====
  #=========================================#

  # #================#
  # # ==== ivreg ====
  # #================#
  #
  #
  #
  #   # using ivreg
  # start1 <- Sys.time()
  #
  #   iv_form <- as.formula(paste0("l_w_wage~educ +", paste(std_cont, collapse = " + "),
  #                                "| ",
  #                                paste(std_cont, collapse = " + "), " + ", paste0(inter_list, colla
  #   iv_reg1 <- data.table(tidy(ivreg(iv_form , data = ak_perm)))
  #
  # end1 <- Sys.time()
  # print(paste0(round(as.numeric(end1 - start1, units = "secs"), 3), " seconds to run"))
  #
  # #=========================#
  # # ==== using matrix ====
  # #=========================#
  #
  #   ak_perm[, const := 1]
  #   start1 <- Sys.time()
  # # make x z and y matrices
  # y <- as.matrix(ak_perm[, l_w_wage])
  # x <- as.matrix(ak_perm[, c("educ", std_cont, 'const'), with = FALSE])
# z <- as.matrix(ak_perm[, c(inter_list, std_cont, "const"), with = FALSE])
#
# # get 2sls
# out_2sls1 <- solve(crossprod(x,z)%*%solve(crossprod(z))%*%crossprod(z,x))%*%crossprod(x,z)%*%solv
# end1 <- Sys.time()
# print(paste0(round(as.numeric(end1 - start1, units = "secs"), 3), " seconds to run"))
#
# #====================#
# # ==== using lm ====
# #====================#
#
# start1 <- Sys.time()
#
# form_1st <- as.formula(paste0("educ~", paste(std_cont, collapse = " + "), " + ", paste0(inter_lis
# first_stage <- lm(form_1st, data = ak_perm)
#
#
# X_hat <- fitted(first_stage)
# form_2nd <- as.formula(paste0("l_w_wage~"," X_hat +", paste(std_cont, collapse = " + ")))
#
# ols_second <- lm(form_2nd, data = ak_perm)
# coef(ols_second)
# end1 <- Sys.time()
# print(paste0(round(as.numeric(end1 - start1, units = "secs"), 3), " seconds to run"))
#
#
#========================#
# ==== matrix try 2 ====
#========================#

# # LOOKS LIKE THIS IS THE WAY TO GO
# start1 <- Sys.time()
#
# # make x z and y matrices
# y <- as.matrix(ak_perm[, l_w_wage])
# x <- as.matrix(ak_perm[, educ])
# cont <- as.matrix(ak_perm[, c( std_cont, 'const'), with = FALSE])
# z <- as.matrix(ak_perm[, c(inter_list, std_cont, "const"), with = FALSE])
#
# first_stage_fit <-  z%*%Matrix::solve(Matrix::crossprod(z))%*%(Matrix::crossprod(z, x))
#
# # make x' matrix
# x_prime <- cbind(first_stage_fit, cont)
#
#
# form_2nd <-  Matrix::solve(Matrix::crossprod(x_prime))%*%(Matrix::crossprod(x_prime, y))
#
# form_2nd[1,1]
# end1 <- Sys.time()
# print(paste0(round(as.numeric(end1 - start1, units = "secs"), 3), " seconds to run"))
#
#====================================#
# ==== write fast 2sls function ====
 #===================================#

 fast_2sls <- function(in_data){

  #===============#
  # ==== reg1 ====
  #===============#

  # make x z and y matrices
  y <- as.matrix(in_data[, l_w_wage])
  x <- as.matrix(in_data[, educ])
  cont <- as.matrix(in_data[, c( std_cont, 'const'), with = FALSE])
  z <- as.matrix(in_data[, c(inter_list, std_cont, "const"), with = FALSE])

  first_stage_fit <-  z%*%Matrix::solve(Matrix::crossprod(z))%*%(Matrix::crossprod(z, x))

  # make x' matrix
  x_prime <- cbind(first_stage_fit, cont)


  form_2nd <-  Matrix::solve(Matrix::crossprod(x_prime))%*%(Matrix::crossprod(x_prime, y))

  reg1 <- data.table( term = "educ", estimate = form_2nd[1,1], model = "2sls 1")

  #===============#
  # ==== reg2 ====
  #===============#

  cont <- as.matrix(in_data[, c( std_cont, extra_cont, 'const'), with = FALSE])
  z <- as.matrix(in_data[, c(inter_list, std_cont, extra_cont, "const"), with = FALSE])

  first_stage_fit <-  z%*%Matrix::solve(Matrix::crossprod(z))%*%(Matrix::crossprod(z, x))

  # make x' matrix
  x_prime <- cbind(first_stage_fit, cont)


  form_2nd <-  Matrix::solve(Matrix::crossprod(x_prime))%*%(Matrix::crossprod(x_prime, y))

  reg2 <- data.table( term = "educ", estimate = form_2nd[1,1], model = "2sls 2")

  # stack results and retur n
  out_results <- rbind(reg1, reg2)
 }


#=========================#
# ==== run simulation ====
#=========================#

 # copy data for permutation
 ak_perm <- copy(ak)
```

```
# add constant
ak_perm[, const := 1]

# write a function so I can parallel this shiz
sim_warper <- function(sim_i, in_data = ak_perm ){

 # get random sampel
 perm <- sample(c(1:nrow(in_data)))

 # purmute data
 in_data[, QoB := QoB[perm]]

 # clear out dummy variables
 in_data <- in_data[, -c(grep("d_QoB", colnames(in_data), value = TRUE), inter_list), with = FALSE]

 # redo dummy vars
 for(qob_i in unique(in_data$QoB)){

  in_data[ ,temp := 0]
  in_data[QoB == qob_i ,temp := 1]
  setnames(in_data, "temp", paste0("d_QoB_", qob_i))

 }
 # recalculate interactions
 inter_list <- NULL
 for(d_year in year_dummies){

  for(d_qob in qob_dummies){

   in_data[, temp := get(d_qob)*get(d_year)]
   setnames(in_data, "temp", paste0(d_year, "X", d_qob))
   inter_list<- c(inter_list, paste0(d_year, "X", d_qob))
  }
 }

 # run 2sls funciton on new data
 ak_2sls_i <- fast_2sls(in_data)

 # add simulation
 ak_2sls_i[, sim := sim_i]

 # retunrn it
 return(ak_2sls_i)

} # end funciton

# time this sucker
start_time <- Sys.time()

# parallel setup
cl <- makeCluster(4, type = "PSOCK")
registerDoParallel(cl)
```

```
# run simulations in parallel
output_list <- foreach(sim = 1 : 5000,
                       .inorder = FALSE,
                       .packages = "data.table",
                       .options.multicore = list(preschedule = FALSE, cleanup = 9)) %dopar% sim_war

# stop clusters
stopCluster(cl)

# check time
end_time <- Sys.time()

# print time
print(paste0(round(as.numeric(end_time - start_time, units = "mins"), 3), " minutes to run"))


#===========================#
# ==== organize output ====
#===========================#

  # stack data
  sim_res3.2 <- rbindlist(output_list)

  # make table
  output3.2 <- sim_res3.2[, list(mean = mean(estimate), std.dev = sd(estimate)), "model"]

  # save it

  print(xtable(output3.2, type = "latex"),
        file = paste0("C://Users/Nmath_000/Documents/Code/courses/econ 675/PS_5_tex/q3.2_table.tex"
        include.rownames = FALSE,
        floating = FALSE)
```