pset 2 Labor

```
#=========================#
# ==== Metrics 675 ps 3 ====
#=========================#



#======================================#
# ==== load packages and clear data ====
#======================================#

library(data.table)
library(doParallel)
library(foreach)
library(ggplot2)
library(Matrix)
library(sandwich)
library(lmtest)
library(xtable)
library(boot)
library(gmm)
library(broom)

# clear data and consol
rm(list = ls(pos = ".GlobalEnv"), pos = ".GlobalEnv")
options(scipen = 999)
cat("\f")

# set options
opt_test_run <- FALSE

# set attributes for plot to default ea theme
plot_attributes <- theme( plot.background = element_rect(fill = "lightgrey"),
                          panel.grid.major.x = element_line(color = "gray90"),
                          panel.grid.minor  = element_blank(),
                          panel.background = element_rect(fill = "white", colour = "black") ,
                          panel.grid.major.y = element_line(color = "gray90"),
                          text = element_text(size= 30),
                          plot.title = element_text(vjust=0,
                                                    hjust = 0.5,
                                                    colour = "#0B6357",
                                                    face = "bold",
                                                    size = 40))


#====================#
# ==== Question 1 ====
#====================#

#==================#
# ==== part 9 a ====
```

```
#==================#


# load in data
p_dt <- fread("c:/Users/Nmath_000/Documents/MI_school/Second Year/675 Applied Econometrics/hw/hw3/pisof

# create si variable
p_dt[,s := 1-dmissing]


# create log variable
p_dt[, log_inc := log(S_incomepc + 1)]

# estimate the logic model
reg <- glm(s ~ S_age + S_HHpeople + log_inc,
           data = p_dt,
           family = binomial(link = 'logit'))


# get robust standard errors
reg_c_r <- coeftest(reg, vcov = vcovHC(reg, type="HC1"))

# tidy that up
reg_c_r <- data.table(tidy(reg_c_r))


# make a copy for tex results
table_q1_9_a <- copy(reg_c_r)

# make confidence interval for part a
table_q1_9_a[,CI_L := estimate - qnorm(.975)*std.error]
table_q1_9_a[,CI_H := estimate + qnorm(.975)*std.error]


#======================#
# ==== Q1 part 9 b ====
#======================#


# right a funciton to bootstrap
b_st1 <- function(in_data, sample, in_reg_c_r){

  # run regression
  b_reg <- glm(s ~ S_age + S_HHpeople + log_inc,
               data = in_data[sample],
               family = binomial(link = 'logit'))


  # get robust standard errors
  b_reg_r <- coeftest(b_reg, vcov = vcovHC(b_reg, type="HC1"))

  # tidy that up
  b_reg_r <- data.table(tidy(b_reg_r))
```

```
  # calculate t stat
  t_stat <- (b_reg_r[, estimate] - in_reg_c_r[, estimate])/in_reg_c_r[, std.error]

  # get the stuff I want
  return(t_stat)
}

# now run the bootstrap
boot_results <- boot(data = p_dt, R = 999, statistic = b_st1, in_reg_c_r = reg_c_r)

#Now get the stats from every sample
boot_samp <- data.table(boot_results$t)

# create balnk data.table for quantiles
quant_dt <- reg_c_r[, "term"]

quant_dt[, t_q.975 := unlist(lapply(boot_samp,  quantile, .975))]
quant_dt[, t_q.025 := unlist(lapply(boot_samp,  quantile, .025))]


# merge on quantiles to original estimates ans standard errors
table_q1_9_b <- merge(reg_c_r[, c("term", "estimate", "std.error")], quant_dt,  by = "term")

# now get bootstrap confidence intervals
table_q1_9_b[,CI_L := estimate + t_q.025*std.error]
table_q1_9_b[,CI_H := estimate + t_q.975*std.error]

# calculate p_values
p_value_fun <- function(vector, theta){

  mean(abs(vector) > abs(theta))
}

# write p value function
P_value_fun <- function(vector, theta){

  2 * max( mean(vector-theta>=abs(theta)), mean(vector-theta<=-1*abs(theta)))
}


table_q1_9_b[, pvalue := mapply(FUN = P_value_fun,
                                vector = as.list(data.table(boot_results$t)),
                                as.list(boot_results$t0))]

#======================#
# ==== Q1 Part 9 c ====
#======================#

# grab the data from the original regression
ests <- data.table(Estmate = reg$fitted.values)


plot_q1_9_c <-  ggplot(ests, aes(x=Estmate)) +
```

```
                geom_density(color = "blue", fill = "blue", kernel = 'gaussian',alpha = .3) +
                plot_attributes + ylab("Density") +
                ggtitle("Kernal Density Estimate")


#====================#
# ==== Question 2 ====
#====================#

#============================#
# ==== question 2 part 2 B ====
#============================#

# subset data to complete cases
p_dt <- na.omit(p_dt, c("danemia", "dpisofirme", "S_age", "S_HHpeople", "log_inc"))

# to test function
data = p_dt
theta = c(1,2,3,4)


# logistic bootstrap function, need to put it all in one functon for parallel to function.
boot.T_logistic <- function(boot.data, ind) {
  require(data.table)
  g_logistic <- function(theta, data) {
    a <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*dpisofirme]

    b <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*S_age]

    c <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*S_HHpeople]

    d <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*log_inc]

    cbind(a, b, c, d)
  }

  gmm::gmm(g_logistic, boot.data[ind], t0=c(0,0,0,0), wmatrix="ident", vcov="iid")$coef
}

# run bootsrtap in parallel
ptm <- proc.time()
```

```r
set.seed(123)
temp <- boot(data=p_dt[s==1, ],
             R=499,
             statistic = boot.T_logistic,
             stype = "i",
             parallel = "snow",
             ncpus=4 )
proc.time() - ptm

# create data.table of results
table_q2_2_b <- data.table(term =  c("dpisofirme", "S_age", "S_HHpeople", "log_inc"),
                           Estimate = temp$t0,
                           sd = apply(temp$t, 2, sd))

# write p value function
P_value_fun <- function(vector, theta){

  2 * max( mean(vector-theta>=abs(theta)), mean(vector-theta<=-1*abs(theta)))
}

# apply p value function to data
table_q2_2_b[, p_value := mapply(FUN = P_value_fun,
                                 vector = as.list(data.table(temp$t)),
                                 as.list(temp$t0))]

# create t stat
table_q2_2_b[, t := Estimate/sd]

#  CI function
CI_fun <- function(vector, theta, quant){

  2 * theta - quantile(vector, quant)

}

# get quantiles
table_q2_2_b[, CI_L := mapply(FUN = CI_fun,
                              vector = as.list(data.table(temp$t)),
                              theta = as.list(temp$t0), quant = .975)]
table_q2_2_b[, CI_H := mapply(FUN = CI_fun,
                              vector = as.list(data.table(temp$t)),
                              theta = as.list(temp$t0), quant = 0.025)]


#==============================#
# ==== question 2 part 3 C ====
#==============================#

# GMM moment condition
g_MAR <- function(theta, data) {
  data <- data[data$s==1, ]
  a <- (data$danemia - plogis(theta[1]*data$dpisofirme +
                                theta[2]*data$S_age +
```

```
                                theta[3]*data$S_HHpeople +
                                theta[4]*log(1+data$S_incomepc)))*data$dpisofirme * data$weights

  b <- (data$danemia - plogis(theta[1]*data$dpisofirme +
                                theta[2]*data$S_age +
                                theta[3]*data$S_HHpeople +
                                theta[4]*log(1+data$S_incomepc)))*data$S_age * data$weights

  c <- (data$danemia - plogis(theta[1]*data$dpisofirme +
                                theta[2]*data$S_age +
                                theta[3]*data$S_HHpeople +
                                theta[4]*log(1+data$S_incomepc)))*data$S_HHpeople * data$weights

  d <- (data$danemia - plogis(theta[1]*data$dpisofirme +
                                theta[2]*data$S_age +
                                theta[3]*data$S_HHpeople +
                                theta[4]*log(1+data$S_incomepc)))*log(1+data$S_incomepc)*data$weights
  cbind(a, b, c, d)
}

# logistic bootstrap
boot.T_MAR <- function(boot.data, ind) {
  data.temp <- boot.data[ind, ]
  fitted <- glm(s ~ dpisofirme + S_age + S_HHpeople +I(log_inc) - 1,
                data = data.temp,
                family = binomial(link = "logit"))$fitted
  data.temp$weights <- 1 / fitted
  gmm(g_MAR, data.temp, t0=c(0,0,0,0), wmatrix="ident", vcov="iid")$coef
}

ptm <- proc.time()
set.seed(123)
temp <- boot(data=p_dt, R=499, statistic = boot.T_MAR, stype = "i")
proc.time() - ptm
table5 <- matrix(NA, ncol=6, nrow=4)
for (i in 1:4) {
  table5[i, 1] <- temp$t0[i]
  table5[i, 2] <- sd(temp$t[, i])
  table5[i, 3] <- table5[i, 1] / table5[i, 2]
  table5[i, 4] <- 2 * max( mean(temp$t[, i]-temp$t0[i]>=abs(temp$t0[i])), mean(temp$t[, i]-temp$t0[i]<=
  table5[i, 5] <- 2 * temp$t0[i] - quantile(temp$t[, i], 0.975)
  table5[i, 6] <- 2 * temp$t0[i] - quantile(temp$t[, i], 0.025)
}

table_q2_3_c <- data.table(cbind(table_q2_2_b$term, table5))
setnames(table_q2_3_c, colnames(table_q2_3_c), c("Term","Estimate", "Std.Error", "t", "p-value", "CI_L"

#=================#
# ==== part d ====
#=================#

# set this up to use data.table and run in parallel so it doesn't take half my life
# logistic bootstrap function, need to put it all in one functon for parallel to work
```

```r
boot.trim <- function(boot.data, ind) {

  require(data.table)

  data.temp <- boot.data[ind, ]

  fitted <- glm(s ~ dpisofirme + S_age + S_HHpeople +I(log_inc) - 1,
                data = data.temp,
                family = binomial(link = "logit"))$fitted
  data.temp[,weights := 1 / fitted]

  # GMM moment condition with trimming
  g_logistic <- function(theta, data) {
    data.temp <- data.temp[s==1 & weights<=1/0.1, ]
    a <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*dpisofirme *weights]

    b <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*S_age*weights]

    c <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*S_HHpeople*weights]

    d <- data[, (danemia -plogis(theta[1]*dpisofirme +
                                   theta[2]*S_age +
                                   theta[3]*S_HHpeople +
                                   theta[4]*log_inc))*log_inc*weights]

    cbind(a, b, c, d)
  }

  gmm::gmm(g_logistic, data.temp, t0=c(0,0,0,0), wmatrix="ident", vcov="iid")$coef
}




ptm <- proc.time()
set.seed(123)
temp <- boot(data=p_dt, R=499, statistic = boot.trim, stype = "i",parallel = "snow", ncpus=4 )
proc.time() - ptm

table6 <- matrix(NA, ncol=6, nrow=4)
for (i in 1:4) {
  table6[i, 1] <- temp$t0[i]
  table6[i, 2] <- sd(temp$t[, i])
  table6[i, 3] <- table6[i, 1] / table6[i, 2]
```

```
  table6[i, 4] <- 2 * max( mean(temp$t[, i]-temp$t0[i]>=abs(temp$t0[i])), mean(temp$t[, i]-temp$t0[i]<=
  table6[i, 5] <- 2 * temp$t0[i] - quantile(temp$t[, i], 0.975)
  table6[i, 6] <- 2 * temp$t0[i] - quantile(temp$t[, i], 0.025)
}


table_q2_3_d <- data.table(cbind(table_q2_2_b$term, table6))
setnames(table_q2_3_d,
         colnames(table_q2_3_d),
         c("Term","Estimate", "Std.Error", "t", "p-value", "CI_L", "CI_U"))

# convert to numeric
to_change <- c("Estimate", "Std.Error", "t", "p-value", "CI_L", "CI_U")
table_q2_3_c[, to_change] <- lapply(table_q2_3_c[,to_change, with = FALSE],  as.numeric)
table_q2_3_d[, to_change] <- lapply(table_q2_3_d[,to_change, with = FALSE],  as.numeric)


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


#=================#
# ==== Part 1 ====
#=================#

n <- 1000
set.seed(123)

# generate random data
r_dt <- runif(n, 0,1)

# get max of data
r_max <- max(r_dt)

# write functin for bootstrap
b_fun <- function(in_data, sample){

  n*(r_max-max(in_data[sample]))

}

# run bootsrtap
u_b <- boot(data=r_dt, R=499, statistic = b_fun, stype = "i" )

# get distribution of statistic
u_b_d <- data.table(u_b$t)


# plot data
plot_q3_1 <- ggplot(u_b_d) +
  geom_density(aes(x = V1, linetype = "Boot_density") ) +
  stat_function(fun = function(x) dexp(x), size = 1, aes(linetype = 'Exponential')) +
```

```
  plot_attributes +
  scale_linetype_manual(name = 'Legend',
                        values = c(Boot_density=1,
                                   Exponential=2))


plot_q3_1

#=================#
# ==== part 2 ====
#=================#

# take random data and do parametric bootstrap
# just write loop for bootstrap
stat_list <- vector("list", length = 599)
for(iter in 1:599){

  # generate random uniform data
  r_data_i <- runif(n, 0,r_max)
  stat_list[[iter]] <- n*(r_max-max(r_data_i))
}
param_b <- data.table(unlist(stat_list))

# plot data
plot_q3_2 <- ggplot(param_b) +
  geom_density(aes(x = V1, linetype = "Boot_density") ) +
  stat_function(fun = function(x) dexp(x), size = 1, aes(linetype = 'Exponential')) +
  plot_attributes +
  scale_linetype_manual(name = 'Legend',
                        values = c(Boot_density=1,
                                   Exponential=2))


plot_q3_2


#======================#
# ==== save output ====
#======================#

# grab all tables to save
tab_list <- grep("table_q", ls(), value=TRUE)

# save all the tables by name

for(tab_i in tab_list){

  print(xtable(get(tab_i), type = "latex",
               digits = 3),
        file = paste0("C:/Users/Nmath_000/Documents/Code/courses/econ 675/PS_3_tex/", tab_i, ".tex"),
        include.rownames = FALSE,
        floating = FALSE)
}
```

```
# save plot

# save plots
plot_list <- grep("plot_q", ls(), value = TRUE)
for(plot_i in plot_list){
png(paste0("c:/Users/Nmath_000/Documents/Code/courses/econ 675/PS_3_tex/",
           plot_i, ".png"), height = 800, width = 800, type = "cairo")
print(get(plot_i))
dev.off()
}
```