pset 2 631

```
#==================#
# ==== pset 2 ====
#==================#

require(data.table)
require(Matrix)
library(xtable)

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

#set #note output location
f_out <- "c:/Users/Nmath_000/Documents/Code/Econ_631/ps2/"

#set #note option to save output
opt_save <- TRUE

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

# set parameters
x1    = 1
x2    = 2
x3    = 3
n.sim = 10000

# Function to compute shares for a given mean and random utility
share_f <- function(delta.in, mu.in, opt_tidle = FALSE){

  # get the numerator by exp(delta + xi*vi*sigma)
  numer <- exp(mu.in) * matrix(rep(exp(delta.in), n.sim), ncol = n.sim)

  # get the denominator by summing over all numerators and adding one
  denom_i <- matrix(rep(1 + colSums(numer),3), nrow = 1, ncol = n.sim)

  # then replicated this three times so we can divide (probs better way to do this )
  denom <- rbind(denom_i, denom_i, denom_i)

  if(opt_tidle){

    return(numer / denom)

  }else{
    # the shares are the mean of numerator/denominator accross simulations
    shares <- rowMeans(numer / denom)

    return(shares)
  }
}

# Function to compute the derivative of your own shares wrt own-good mean utility
#note we can just use output of shares function as input here
dSharedOwnP_f <- function(shares.in, alpha.in){

  # dS.i/dP.i is -alpha*share*(1-share)
  dSharedOwnP <- rowMeans(-alpha.in*shares.in*(1-shares.in))

  return(dSharedOwnP)
}


#note switcing this to take shares as the input so we dont recalculate it
dSharedOtherP_f <- function(shares.in, alpha.in){

  # Just using shares output from other function
  # share.i <- matrix(shares.in)

  # dS.i/dDelta.j is integral of -s.i*s.j
  sisj.matrix <- -shares.in%*%t(shares.in)/ncol(shares.in)

  # dS.i/dP.j is -alpha*dS.i/dDelta.j
  #note I don't understand why we are only grabbing 1,2
  dSharedOtherP <- -alpha.in*sisj.matrix[1,2]
    return(dSharedOtherP)
}

# create data.tabe of xs
xi <- as.matrix(c(x1,x2,x3))

# make simulation matrix
v = matrix(rnorm(1 * n.sim), nrow = 1, ncol = n.sim)


# Now we will guess the price to start and calcualte everything
# fill in an initial price guess to work through functions
# price in iteration k
p.init <- matrix(c(2, 3, 4))

tol <- 10^-10


p_solver <- function(beta.in, alpha.in, sigma.in, xi.in, mc.in, p.guess){
  #==========================#
  # ==== Inside the loop ====
  #==========================#
  i <- 1

  # Initial guess
  p.old <- matrix(c(0, 0, 0))
```

```
    while (sum(abs(p.guess - p.old)) > tol)
    {
      print(paste0("Iteration:", i, ", Difference:", sum(abs(p.guess - p.old))))

      p.old <- p.guess

      # using the guess, calualte deltas
      delta  <- xi.in*beta.in - alpha.in*p.guess

      # calculate x times sigma times v
      mu <- xi.in%*%v*sigma.in

      # Calculate shares and derivative
      shares <- as.matrix(share_f(delta, mu))
      shares_tilde <- share_f(delta, mu, opt_tidle = TRUE)

      dSharedOwnP <- as.matrix(dSharedOwnP_f(shares_tilde, alpha.in))

      # using the shares and derivative, calculate the equilibrium price
      p.guess <- mc.in - shares*(dSharedOwnP)^-1

      i <- i + 1
    }
  p.final <- p.guess
  return(p.final)

}

# get answer for question 1
p_q1 <- p_solver(1, 1, 1, xi, mc.in = xi,  p.init)


#=====================#
# ==== question 2 ====
#=====================#

p_q2 <- p_solver(.5, .5, .5, xi, mc.in = xi, p.init)



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


p_postmerge_solver <- function(beta.in, alpha.in, sigma.in, xi.in, mc.in, p.guess){
  #==========================#
  # ==== Inside the loop ====
  #==========================#
  i <- 1
  p.old <- matrix(c(0, 0, 0))
  while (sum(abs(p.guess - p.old)) > tol)
  {
```

```r
    print(paste0("Iteration:", i, ", Difference:", sum(abs(p.guess - p.old))))

    p.old <- p.guess

    # using the guess, caluate deltas
    delta  <- xi.in*beta.in - alpha.in*p.guess

    # calculate x times sigma times v
    mu <- xi.in%*%v*sigma.in

    # You care about the markup of the other product you own, so create a variable for 2's markup for 1
    markup <- p.guess - mc.in

    # Definitely a better way to do this...
    othergood.markup <- rbind(markup[2,1], markup[1, 1], 0)

    # Calculate shares, own price elasticities
    shares <- as.matrix(share_f(delta, mu))
    shares_tilde <- share_f(delta, mu, opt_tidle = TRUE)
    dSharesdOwnP <- as.matrix(dSharedOwnP_f(shares_tilde, alpha.in))

    # Calculate price elasticities wrt the other product we care about.
    #note: would rather use an ownership matrix somehow but yolo
    dSharesdOtherP <- as.matrix(c(dSharedOtherP_f(shares_tilde, alpha.in), dSharedOtherP_f(shares_tilde

    # using the shares and derivative, calculate the equilibrium price
    p.guess <- xi.in - (shares + othergood.markup*dSharesdOtherP)*(dSharesdOwnP)^-1

    i <- i + 1
  }

  p.final <- p.guess

  return(p.final)

}


p_q3 <- p_postmerge_solver(.5, .5, .5, xi,mc.in = xi, matrix(c(2, 3, 4)))


#======================#
# ==== question 4 ====
#======================#


#====================================#
# ==== change in consumer surplus ====
#====================================#


  # define variables for debug
  v.in = v
  pv = p_q2


  # function for getting sum of value funcitons
  vi_f <- function(v.in, pv, xi.in, beta.in, alpha.in, sigma.in){

    # make beta_i
   beta_i <-  beta.in + sigma.in*v.in

   # get beta_i times xs
   #note this is old. It does not doe the exponent. Can delet when we are sure it is wrong
   # Vi <- colSums( xi.in %*% beta_i  ) - colSums(alpha*pv )
   Vi <- colSums( exp(xi.in %*% beta_i - matrix(rep(alpha.in*pv, ncol(beta_i)), ncol = ncol(beta_i))) )

   return(Vi)
  }


  # #note temp define these for deubg
  # pv_pre = p_q2
  # pv_post = p_q3

  # NOw write funciton to get cv_i
  cv_i_f <- function(v.in, pv_pre, pv_post, xi.in, beta.in, alpha.in, sigma.in ){

    # get vi for pre
    vi_pre <- vi_f(v.in, pv = pv_pre, xi.in, beta.in, alpha.in, sigma.in)

    # et vi for post
    vi_post <- vi_f(v.in, pv = pv_post, xi.in, beta.in, alpha.in, sigma.in)

    # get cv_i
    cv_i <- (log(vi_post) - log(vi_pre))/-alpha.in

    return(cv_i)

  }

  # run it with correct values
  cv_i <- cv_i_f(v.in     = v,
                 pv_pre   = p_q2,
                 pv_post  = p_q3,
                 xi.in    = xi,
                 beta.in  = .5,
                 alpha.in = .5,
                 sigma.in = .5)

  # now get mean cv
  mean_cv <- mean(cv_i)

#=====================================#
# ==== Change in producer surplus ====
#=====================================#
```

```r
# # for debug
# pv       = p_q2
# mc_v     = xi
# v.in     = v
# xi.in    = xi
# alpha.in = .5
# beta.in  = .5
# sigma.in = .5
profit_f <- function(pv,mc_v, v.in, alpha.in, beta.in, xi.in, sigma.in){

  # using the guess, calualte deltas
  delta  <- xi.in*beta.in - alpha.in*pv

  # calculate x times sigma times v
  mu <- xi.in%*%v.in*sigma.in

  shares <- share_f(delta, mu)
  # get profits before and afte
  profits <-( pv - mc_v)*shares

  return(profits)
}


# get proffts before
profits_before <- profit_f(pv       = p_q2,
                           mc_v     = xi,
                           v.in     = v,
                           xi.in    = xi,
                           alpha.in = .5,
                           beta.in  = .5,
                           sigma.in = .5)
# get profits after
profits_after <- profit_f(pv        = p_q3,
                           mc_v     = xi,
                           v.in     = v,
                           xi.in    = xi,
                           alpha.in = .5,
                           beta.in  = .5,
                           sigma.in = .5)


# get the total difference in profits i.e. producer surplus
change_ps <- sum(profits_after) - sum(profits_before)

# get change in total surplus
 total_surplus_change <- change_ps - mean_cv

# table all the info
q4_table <- data.table(variable = c("change in cosumer surplus per person",
                                    "change in Producer surplus per person",
                                    "change in total surplus per person"),
                      value = c(-mean_cv, change_ps,total_surplus_change))
```

```r
#=====================#
# ==== Question 5 ====
#=====================#
mc_new <- c(.5,1,3)

p_post_q5 <- p_postmerge_solver(.5, .5, .5, xi, mc.in = mc_new, p.init)

# get cv
cv_i <- cv_i_f(v.in     = v,
               pv_pre   = p_q2,
               pv_post  = p_post_q5,
               xi.in    = xi,
               beta.in  = .5,
               alpha.in = .5,
               sigma.in = .5)

# now get mean cv
mean_cv <- mean(cv_i)

# get profits after
profits_after_q5 <- profit_f(pv      = p_post_q5,
                          mc_v      = mc_new,
                          v.in      = v,
                          xi.in     = xi,
                          alpha.in  = .5,
                          beta.in   = .5,
                          sigma.in  = .5)


# get the total difference in profits i.e. producer surplus
change_ps <- sum(profits_after_q5) - sum(profits_before)

# get change in total surplus
total_surplus_change <- change_ps - mean_cv

# table all the info
q5_table <- data.table(variable = c("change in cosumer surplus per person",
                                    "change in Producer surplus per person",
                                    "change in total surplus per person"),
                       value = c(-mean_cv, change_ps,total_surplus_change))

#================================#
# ==== save output to latex ====
#================================#

# make these tables pretty
p_q1_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_q1))
p_q2_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_q2))
p_q3_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_q3))
p_post_q5_out <- data.table(variable = c("p1", "p2", "p3"), value = as.numeric(p_post_q5))
```

```
 # capitolize first letters
q4_table[, variable := sapply(variable, function(x) paste0(sapply(strsplit(x, " "), Hmisc::capitalize),
q5_table[, variable := sapply(variable, function(x) paste0(sapply(strsplit(x, " "), Hmisc::capitalize),

if(opt_save){


  print(xtable(p_q1_out, type = "latex"),
        file = paste0(f_out, "p_q1.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(p_q2_out, type = "latex"),
        file = paste0(f_out, "p_q2.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(p_q3_out, type = "latex"),
        file = paste0(f_out, "p_q3.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(q4_table, type = "latex"),
        file = paste0(f_out, "q4_table.tex"),
        include.rownames = FALSE,
        floating = FALSE)

  print(xtable(p_post_q5_out, type = "latex"),
        file = paste0(f_out, "p_post_q5.tex"),
        include.rownames = FALSE,
        floating = FALSE)


  print(xtable(q5_table, type = "latex"),
        file = paste0(f_out, "q5_table.tex"),
        include.rownames = FALSE,
        floating = FALSE)


}
```