```
# Industrial Organization II - Problem Set 1
# Demand Estimation (BLP)
# Gabriel Gonzalez Sutil - gg2718


# Preliminaries -------------------------------------------------------------
rm(list = ls())
pacman :: p_load('data.table','dplyr','bbmle','AER','gmm','stargazer', 'optimx')
dir <- 'C:/Users/Gabriel Gonzalez Sut/Google Drive/PhD Sustainable - Columbia/'
course <- 'Sem 5 - Industrial Organization II/Problem Sets/'
ps <- 'PS1'
setwd(paste0(dir,course,ps))

# Import Data ---------------------------------------------------------------------------------------
auto <-  read.delim('ps1_blp_data_no_header.txt', header = FALSE)
colnames(auto) <- c('price','quantity','weight','hp','AC','firm')
auto$price <- as.numeric(auto$price)
auto$quantity <- as.numeric(auto$quantity)
auto$weight <- as.numeric(auto$weight)
auto$hp <- as.numeric(auto$hp)
auto$AC <- as.numeric(auto$AC)
auto$firm <- as.numeric(auto$firm)
M <- 100 * 10^6 # Market Size
lambda <- 4 * 10^(-6)

# Functions
BLP.z <- function(data,covariates){
  n <- (length(covariates)*2)
  Z <- matrix(0,nrow(data),n)
  Z[,1:3] <- as.matrix(data[,covariates, with = FALSE])
  for (i in 1:nrow(data)){
    competitor <- as.matrix(data[firm != data$firm[i],covariates, with = FALSE])
    if (nrow(competitor) > 1) Z[i,4:6] <- colSums(competitor)
    if (nrow(competitor) == 1) Z[i,4:6] <- competitor
  }
  Z <- as.data.table(Z)
  colnames(Z) <- paste0('Z',1:n)
  return(Z)
}

#  ------------------------------ Random Coefficients 1 ---------------------------------------------

# Preliminaries
random <- as.data.table(auto)
random <- random[order(price),]
random[, share := (quantity/M)] # Define share
variables <- c('weight','hp','AC')
X <- cbind(rep(1,131),random[,variables, with = FALSE])
mean <- 35000
sd <- 45000
sim <- 100000 # Number of Simulations
par <- 1 # Number of Random Coefficients
random$delta <- log(share) - log(1 - sum(share)) # Initial delta
attach(random)

# Step 1: v
set.seed(123456)
v <- matrix(rnorm(sim * par, mean = 0, sd = 1), sim, 1)

# Step 2: construct random coefficient
sigma <- sqrt(log((sd^2/mean^2) + 1))
mu <- log(mean^2/sqrt(sd^2+mean^2))
alpha <- 1/exp(mu + sigma * v)

# Iterations limits
tolerance <- 10^(-13)
t <- 1000
iter <- 1
limit <- 1000
dist <- matrix(0,limit)
s <- matrix(0,nrow(random),sim)
s_hat <- rep(0,nrow(random))

# Step 3: Sample average over simulations
while(t > tolerance & iter <= limit){
  print(iter)
  # Simulated Data
  for(i in 1:sim){
    aux <- delta - (alpha[i] - exp(-mu + 0.5 * sigma^2)) * price
    numerator <- exp(aux)
    s[,i] <- numerator/(1 + sum(numerator))
  }
  s_sim1 <<- s
  s_hat <- rowMeans(s)
  # Step 5: Contracting Mapping
  deltaUpdated <- delta + log(share) - log(s_hat)
  t <- dist(rbind(deltaUpdated, delta), method = "maximum")
  dist[iter,1] <- t[1]
  delta <- deltaUpdated
  iter <- iter + 1
  random$s.est1 <- s_hat
  random$delta.est1 <- delta +  exp(-mu + 0.5 * sigma^2) * price
}
if(iter <= limit){
  print("Convergence achieved")
}else{
  print("Iteration Limits Achieved")
}

# Step 4: 2SLS
blp1 <- lm(delta.est1 ~ weight + hp + AC, data = random)
summary(blp1)

# GMM
moment <- function(theta){
  xi <- delta - as.matrix(X[,1:4]) %*% as.matrix(theta[1:4]) + exp(-mu + 0.5 * sigma^2) * as.matrix(price)
```

```
  g <- (t(X) %*% xi)*(1/nrow(X))
  W <- as.matrix(t(X)) %*% as.matrix(X)
  obj <- t(g) %*% solve(W) %*% g
}
gmm.blp1 <- opm(par = c(0,0,0,0), fn = moment, method=c('BFGS'))

# Results
blp1$coefficients
cbind(gmm.blp1$p1,gmm.blp1$p2,gmm.blp1$p3,gmm.blp1$p4)
rm(v, dist, s, alpha)

#  ------------------------------ Random Coefficients --------------------------------------------

# Preliminaries
Z <- BLP.z(random,variables)
mean <- 35000
sd <- 45000
sim <- 100000 # Number of Simulations
par <- 1 # Number of Random Coefficients
delta <- random$delta # Initial delta

# Step 1: v
set.seed(123456)
v <- matrix(rnorm(sim * par, mean = 0, sd = 1), sim, 1)

random.coefficients <- function(theta){
  print('Starting Inner Loop')
  alpha1 <- theta[1]
  alpha2 <- theta[2]
  beta <- theta[3:6]

  # Step 2: construct random coefficient
  sigma <- sqrt(log((sd^2/mean^2) + 1))
  mu <- log(mean^2/sqrt(sd^2+mean^2))
  mu_hat <- -mu + log(alpha2)
  alpha <- alpha1 + exp((-mu + log(alpha2)) + sigma * v)

  # Iterations limits
  tolerance <- 10^(-13)
  t <- 1000
  iter <- 1
  limit <- 1000
  dist <- matrix(0,limit)
  s_hat <- rep(0,nrow(random))

  # Step 3: Sample average over simulations
  while(t > tolerance & iter <= limit){
    print(iter)
    s <- matrix(t(alpha), nrow = nrow(random), ncol= sim, byrow=TRUE)
    delta_m <- matrix(delta, nrow = nrow(random), ncol= sim, byrow=FALSE)
    # Simulated Data
    aux <-  delta_m - (s - (alpha1  + exp(mu_hat + 0.5 * sigma^2))) * price
    numerator <- exp(aux)
    for(i in 1:sim){
      s[,i] <- numerator[,i]/(1 + sum(numerator[,i]))
    }
    s_sim2 <<- s
    s_hat <- rowMeans(s, na.rm = TRUE)
    random$s.est2 <<- s_hat

    # Step 5: Contracting Mapping
    deltaUpdated <- delta + log(share) - log(s_hat)
    t <- dist(rbind(deltaUpdated, delta), method = "maximum")
    dist[iter,1] <- t[1]
    random$delta.est2 <- delta
    delta <- deltaUpdated
    random$delta.est2 <<- delta
    iter <- iter + 1
  }
  if(iter <= limit){
    print("Convergence achieved")
  }else{
    print("Iteration Limits Achieved")
  }

  # Step 4: Outer Loop
  xi <- delta - as.matrix(X[,1:4]) %*% as.matrix(beta) + (alpha1  + exp(mu_hat + 0.5 * sigma^2)) * as.matrix(price)
  g <- (t(Z) %*% xi)*(1/nrow(X))
  W <- as.matrix(t(Z)) %*% as.matrix(Z)
  obj <- t(g) %*% solve(W) %*% g
}

gmm.blp2 <- opm(par = c(0,0.1,0,0,0,0), fn = random.coefficients, method=c('BFGS'))

# If want to use L-BFGS: lower = c(-Inf,0,-Inf,-Inf,-Inf), method=c('L-BFGS-B'))

# Results
blp.coefficient <- data.table(Variable = c("alpha.1","alpha.2",'constant',"beta.wg","beta.hp","beta.ac"),
                              Case.1 = c(0,1,round(gmm.blp1$p1,5),round(gmm.blp1$p2,5),round(gmm.blp1$p3,5),
                                         round(gmm.blp1$p4,5)),
                              Case.2 = c(round(gmm.blp2$p1,5),round(gmm.blp2$p2,5),round(gmm.blp2$p3,5),
                                         round(gmm.blp2$p4,5),round(gmm.blp2$p5,5),round(gmm.blp2$p6,5)))

############ Compare Alpha ##################

alpha1 <- exp(-mu + sigma * v)
mean(alpha1)
sd(alpha1)

alpha2 <- gmm.blp2$p1 + exp(-mu + log(gmm.blp2$p2) + sigma * v)
mean(alpha2)
sd(alpha2)

png('alpha1.png')
plot(density(alpha1), col = 'blue', xlab = 'Price Sensitivity (alpha)', main = NA)
dev.off()
```

```
png('alpha2.png')
plot(density(alpha2), col = 'red', xlab = 'Price Sensitivity (alpha)', main = NA)
dev.off()

############ Elasticities Case 1 ##################

integral <- matrix(0,nrow(random),sim)
integral <- as.data.table(integral)
for (j in 1:sim){
    integral[,j] <- alpha1[j] * s_sim1[,j] * (1- s_sim1[,j])
}

elast.own.r1 <- -price/random$share * (rowMeans(integral))
png('ownr1.png')
plot(price, elast.own.r1, xlab = 'Car Price', ylab = 'Own-price Elasticity')
dev.off()

integral <- matrix(0,nrow(random),sim)
integral <- as.data.table(integral)
for (j in 1:sim){
    integral[,j] <- alpha1[j] * s_sim1[,j] * s_sim1[66,j]
}

elast.cross.r1 <- price[66]/random$share * (rowMeans(integral))
png('crossr1.png')
plot(price, elast.cross.r1, xlab = 'Car Price', ylab = 'Cross-price Elasticity vs Median Priced Car')
dev.off()


############ Elasticities Case 2 ##################

integral <- matrix(0,nrow(random),sim)
integral <- as.data.table(integral)
for (j in 1:sim){
    integral[,j] <- alpha2[j] * s_sim2[,j] * (1- s_sim2[,j])
}

elast.own.r2 <- -price/random$share * (rowMeans(integral))
png('ownr2.png')
plot(price, elast.own.r2, xlab = 'Car Price', ylab = 'Own-price Elasticity')
dev.off()

integral <- matrix(0,nrow(random),sim)
integral <- as.data.table(integral)
for (j in 1:sim){
    integral[,j] <- alpha2[j] * s_sim2[,j] * s_sim2[66,j]
}

elast.cross.r2 <- price[66]/random$share * (rowMeans(integral))
png('crossr2.png')
plot(price, elast.cross.r2, xlab = 'Car Price', ylab = 'Cross-price Elasticity vs Median Priced Car')
dev.off()
```