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

# Preliminaries -----------------------------------------------------------
rm(list = ls())
pacman :: p_load('data.table', 'optimx','readr','sandwich','AER', 'lfe', 'np', 'optimx', 'estprod','stargazer','plm')
dir <- 'C:/Users/Gabriel Gonzalez Sut/Google Drive/PhD Sustainable - Columbia/'
course <- 'Sem 5 - Industrial Organization II/Problem Sets/'
ps <- 'PS2'
setwd(paste0(dir,course,ps))

# Import Data -------------------------------------------------------------
food <-  read.table('ps_prod_data.txt', quote="\"", comment.char="")
colnames(food) <- c('id','year','output','capital','labor','material','l.output','l.capital','l.labor','l.material')
food <- as.data.table(food)
food <- food[,list(output, capital, labor, material,l.output,l.capital,l.labor,l.material, sort(year)), by = 'id']

# Table to store scale ----------------------------------------------------
scale <- data.table(model = c('OLS','FE','LP','ACF'),
                    coef = c(0,0,0,0),
                    std = c(0,0,0,0))

# Question 1: OLS ---------------------------------------------------------

mod1 <- lm(output ~ capital + labor, data = food)
summary(mod1)
vcv <- vcov(mod1)
scale[1,2] <- mod1$coefficients[2] + mod1$coefficients[3]
g <- matrix(c(0,1,1))
scale[1,3] <- sqrt(t(g) %*% vcv %*% g)

# Question 2: FE ---------------------------------------------------------

food.m <- aggregate(food[,c('output','capital','labor','id')], by = list(food$id), FUN = mean)
food.m <- food.m[,c('id','output','capital','labor')]
colnames(food.m) <- c('id','mean.y','mean.k','mean.l')
food.fe <- merge(food,food.m, by = 'id')
food.fe$diff.y <- food.fe$output - food.fe$mean.y
food.fe$diff.k <- food.fe$capital - food.fe$mean.k
food.fe$diff.l <- food.fe$labor - food.fe$mean.l

mod2 <- felm(output ~ capital + labor | id | 0  |, data = food)
summary(mod2) # Gives different sd. since there are repeated observations
# alternative: fe <- plm(output ~ capital + labor, data = food, index = c('id','V9'), mode = 'within')
mod2fe <- lm(diff.y ~ -1 + diff.k + diff.l, data = food.fe)
summary(mod2fe)
vcvfe <- vcov(mod2fe)

constant <- food.m$mean.y - (mod2$coefficients[1] * food.m$mean.k + mod2$coefficients[2] * food.m$mean.k)

png('fig1.png')
plot(density(constant), main = 'Intercept')
dev.off()

scale[2,2] <- mod2fe$coefficients[1] + mod2fe$coefficients[2]
g <- matrix(c(1,1))
scale[2,3] <- sqrt(t(g) %*% vcvfe %*% g)

# Question 3-4: LP --------------------------------------------------------

food.LP <- food

# Stage 1
#fs1 <- lm(output ~ capital + labor + material + I(capital^2) + I(material^2) + capital * material, data = food.LP)
fs1 <- lm(output ~ labor + poly(cbind(material,capital), degree = 3, raw = T), data = food.LP)
summary(fs1)

# bw <- npplregbw(output ~ capital + labor | capital + material, data = food.LP) #bandwidth
# fs2 <- npplreg(bw) # Partially linear kernel regression
# summary(fs2)

food.LP$p.output <- predict(fs1)
lag <- as.data.frame(cbind(1,food.LP$l.labor,poly(cbind(food.LP$l.material,food.LP$l.capital), degree = 3, raw = T)))
food.LP$lp.output <- as.matrix(lag) %*% as.matrix(fs1$coefficients)
food.LP$phi <- food.LP$p.output  -  fs1$coefficients[2] * food.LP$labor
food.LP$l.phi <- food.LP$lp.output  -  fs1$coefficients[2] * food.LP$l.labor

png('fig2.png')
plot(density(food.LP$phi), col = 'blue', main = 'Density Phi LP');
  lines(density(food.LP$l.phi), col = 'red');
  legend(9,0.6, legend = c('Phi at t', 'Phi at t-1'), col = c('blue', 'red'), lty=1:1, cex=0.8)
dev.off()

# Stage 2:
moment <- function(theta){
  beta0 <- theta[1]
  betak <- theta[2]
  omega <- food.LP$phi - beta0 - betak * as.matrix(food.LP$capital)
  omega_1 <- food.LP$l.phi - betak * food.LP$l.capital - beta0
  fit.lp <<- lm(omega ~ poly(omega_1, degree = 3, raw = T))
  etak <- (omega - predict(fit.lp)) * food.LP$capital
  g <- rbind(mean(omega),mean(etak))
  obj <- t(g) %*% g
}
gmm.lp<- opm(par = c(0,0), fn = moment, method=c("BFGS"))

lp <- levinsohn_petrin(food.LP, output ~ labor | capital | material , id = 'id', time = 'V9')

results <- data.table(Variables = c('intercept','labor','capital'),
                      gabo = c(gmm.lp$p1, fs1$coefficients[2], gmm.lp$p2),
                      LP_package = c(' ',round(lp$t0[1],7),round(lp$t0[2],7)))

# Variance:
omega <- food.LP$phi - gmm.lp$p1 -  gmm.lp$p2 * food.LP$capital
omega_1 <- food.LP$l.phi - gmm.lp$p2 * food.LP$l.capital - gmm.lp$p1
```

```
ss1_coef <- fit.lp$coefficients

gamma <- matrix(0,2,2)
gamma[1,1] <- -1
gamma[1,2] <- -mean(food.LP$capital)
gamma[2,1] <- mean((-1 + ss1_coef[2] + 2 * ss1_coef[3] * omega_1 + 3 * ss1_coef[4] * omega_1^2) * food.LP$capital)
gamma[2,2] <- mean((-food.LP$capital + food.LP$l.capital * (ss1_coef[2] + 2 * ss1_coef[3] * omega_1 + 3 * ss1_coef[4] * omega_1^2)) *
food.LP$capital)

g.omega <- ss1_coef[1] +  ss1_coef[2] * omega_1 + ss1_coef[3] * omega_1^2 + ss1_coef[4] * omega_1^3
etak <- (omega - g.omega) * food.LP$capital
g <- cbind(omega,etak)
V <- cov(g) # (t(g) %*% g)/nrow(food)
vcov_lp <- solve(gamma) %*% as.matrix(V) %*% solve(t(gamma))/length(omega)
sqrt(vcov_lp[1,1])
sqrt(vcov_lp[2,2])

# Bootstrap:

loops <- 1000
beta.lp <- matrix(0,loops,3)
indeces <- seq(1:nrow(food))
for (i in 1:loops){
  if ((i %% 100) == 0){
    print(i)
  }
  # Select Indexes
  ind <- sample(indeces, length(indeces), replace=T)
  foodb <- food.LP[ind, ]
  # firms <- unique(food$id) if want to sample by id
  #ind <- data.table(id = sample(firms, length(firms), replace=T)) if want to sample by id
  #foodb <- food[ind, on = .(id = id), roll = TRUE] if want to sample by id

  # First Stage
  fs1b <- lm(output ~ labor + poly(cbind(material,capital), degree = 3, raw = T), data = foodb)
  foodb$p.output <- predict(fs1b)
  lagb <- as.data.frame(cbind(1,foodb$l.labor,
                              poly(cbind(foodb$l.material,foodb$l.capital), degree = 3, raw = T)))
  foodb$lp.output <- as.matrix(lagb) %*% as.matrix(fs1b$coefficients)
  foodb$phi <- foodb$p.output  -  fs1b$coefficients[2] * foodb$labor
  foodb$l.phi <- foodb$lp.output  -  fs1b$coefficients[2] * foodb$l.labor

  # Second Stage
  #phi <- phi
  #l.phi <- foodb$l.phi
  #capital <- foodb$capital
  #l.capital <- foodb$l.capital
  momentb <- function(theta){
    beta0b <- theta[1]
    betakb <- theta[2]
    omegab <- foodb$phi - beta0b - betakb * as.matrix(foodb$capital)
    omegab_1 <- foodb$l.phi - betakb * foodb$l.capital - beta0b
    etakb <- (omegab - predict(lm(omegab ~ poly(omegab_1, degree = 3, raw = T)))) * foodb$capital
    gb <- rbind(mean(omegab),mean(etakb))
    obj <- t(gb) %*% gb
  }
   gmm.lpb<- opm(par = c(1,1), fn = momentb, method=c("BFGS"))
#  gmm.lpb<- opm(par = c(1,1), fn = momentb, lower=c(0,0), method=c("L-BFGS-B"))
  beta.lp[i,1] <- gmm.lpb$p1
  beta.lp[i,2] <- fs1b$coefficients[2]
  beta.lp[i,3] <- gmm.lpb$p2
}

png('fig3.png')
par(mfrow=c(1,3))
plot(density(beta.lp[beta.lp[,3]>0,1]), main = 'Intercept');
  abline(v = gmm.lp$p1, col = 'blue')
plot(density(beta.lp[beta.lp[,3]>0,2]), main = 'Labor');
  abline(v = fs1$coefficients[2], col = 'blue')
plot(density(beta.lp[beta.lp[,3]>0,3]), main = 'Capital');
  abline(v = gmm.lp$p2, col = 'blue')
par(mfrow=c(1,1))
dev.off()

sd(beta.lp[,1])
sd(beta.lp[,2])
sd(beta.lp[,3])

# Scale

scale[3,2] <- fs1$coefficients[2] + gmm.lp$p2
scale[3,3] <- sqrt(cov(beta.lp)[2,2] + cov(beta.lp)[3,3] + 2 * cov(beta.lp)[2,3])

# Question 5: ACF ----------------------------------------------------------

food.ACF <- food

# Stage 1

acf1 <- lm(output ~ poly(cbind(capital, labor, material), degree = 3, raw = T), data = food.ACF)
food.ACF$phi <- predict(acf1)
lag <- as.data.frame(cbind(1,poly(cbind(food.ACF$l.capital, food.ACF$l.labor, food.ACF$l.material), degree = 3, raw = T)))
food.ACF$l.phi <- as.matrix(lag) %*% as.matrix(acf1$coefficients)

png('fig4.png')
plot(density(food.ACF$phi), col = 'blue', main = 'Density Phi ACF');
lines(density(food.ACF$l.phi), col = 'red');
legend(9,0.4, legend = c('Phi at t', 'Phi at t-1'), col = c('blue', 'red'), lty=1:1, cex=0.8)
dev.off()

# Stage 2
moment <- function(theta){
    beta0 <- theta[1]
    betal <- theta[2]
    betak <- theta[3]
```

```
    omega <- food.ACF$phi - beta0 - betak * as.matrix(food.ACF$capital) - betal * as.matrix(food.ACF$labor)
    omega_1 <- food.ACF$l.phi - beta0 - betak * as.matrix(food.ACF$l.capital) - betal * as.matrix(food.ACF$l.labor)
    fit.acf <<- lm(omega ~ poly(omega_1, degree = 3, raw = T))
    eta <- (omega - predict(fit.acf))
    etak <- eta * food.ACF$capital
    etal <- eta * food.ACF$labor
    g <- rbind(mean(omega), mean(etak), mean(etal))
    obj <- t(g) %*% g
}
gmm.acf <- opm(par = c(0,0,0), fn = moment, method=c("BFGS"))

# Variance:
omega <- food.ACF$phi - gmm.acf$p1 - gmm.acf$p2 * food.ACF$labor -  gmm.acf$p3 * food.ACF$capital
omega_1 <- food.ACF$l.phi - gmm.acf$p1 - gmm.acf$p2 * food.ACF$l.labor - gmm.acf$p3 * food.ACF$l.capital

ss2_coef <- fit.acf$coefficients

gamma <- matrix(0,3,3)
gamma[1,1] <-   -1
gamma[1,2] <-   -mean(food.ACF$labor)
gamma[1,3] <-   -mean(food.ACF$capital)
gamma[2,1] <-   mean((-1 + ss2_coef[2] + 2 * ss2_coef[3] * omega_1 + 3 * ss2_coef[4] * omega_1^2) * food.ACF$capital)
gamma[2,2] <-   mean((-food.ACF$labor + food.ACF$l.labor * (ss2_coef[2]  + 2 * ss2_coef[3] * omega_1 + 3 *  ss2_coef[4] * omega_1^2)) *
food.ACF$capital)
gamma[2,3] <-   mean((-food.ACF$capital + food.ACF$l.capital * (ss2_coef[2] + 2 * ss2_coef[3] * omega_1 + 3 * ss2_coef[4] * omega_1^2)) *
food.ACF$capital)
gamma[3,1] <-   mean((-1 + ss2_coef[2] + 2 * ss2_coef[3] * omega_1 + 3 * ss2_coef[4] * omega_1^2) * food.ACF$labor)
gamma[3,2] <-   mean((-food.ACF$labor + food.ACF$l.labor * (ss2_coef[2]  + 2 * ss2_coef[3] * omega_1 + 3 * ss2_coef[4] * omega_1^2)) *
food.ACF$labor)
gamma[3,3] <-   mean((-food.ACF$capital + food.ACF$l.capital * (ss2_coef[2] + 2 * ss2_coef[3] * omega_1 + 3 * ss2_coef[4] * omega_1^2)) *
food.ACF$labor)

g.omega <- ss2_coef[1] +  ss2_coef[2] * omega_1 + ss2_coef[3] * omega_1^2 + ss2_coef[4] * omega_1^3
etak <- (omega - g.omega) * food.ACF$capital
etal <- (omega - g.omega) * food.ACF$labor
g <- cbind(omega,etak,etal)
V <- cov(g)
vcov_acf <- solve(gamma) %*% as.matrix(V) %*% solve(t(gamma))/length(omega)
sqrt(vcov_acf[1,1])
sqrt(vcov_acf[2,2])
sqrt(vcov_acf[3,3])

# Bootstrap

loops <- 1000
beta.acf <- matrix(0,loops,3)
indeces <- seq(1:nrow(food))
for (i in 1:loops){
  if ((i %% 100) == 0){
    print(i)
  }
  # Select Indexes
  ind <- sample(indeces, length(indeces), replace=T)
  foodb <- food.ACF[ind, ]

  # First Stage
  acf1b <- lm(output ~ poly(cbind(capital, labor, material), degree = 3, raw = T), data = foodb)
  foodb$phi <- predict(acf1b)
  lagb <- as.data.frame(cbind(1,poly(cbind(foodb$l.capital, foodb$l.labor, foodb$l.material), degree = 3, raw = T)))
  foodb$l.phi <- as.matrix(lagb) %*% as.matrix(acf1b$coefficients)

  # Second Stage

  momentb1 <- function(theta){
    beta0b <- theta[1]
    betalb <- theta[2]
    betakb <- theta[3]
    omegab1 <- foodb$phi - beta0b - betakb * as.matrix(foodb$capital) - betalb * as.matrix(foodb$labor)
    omegab1_1 <- foodb$l.phi - beta0b - betakb * as.matrix(foodb$l.capital) - betalb * as.matrix(foodb$l.labor)
    etab1 <- (omegab1 - predict(lm(omegab1 ~ poly(omegab1_1, degree = 3, raw = T))))
    etakb1 <- etab1 * foodb$capital
    etalb1 <- etab1 * foodb$labor
    gb1 <- rbind(mean(omegab1), mean(etakb1), mean(etalb1))
    obj <- t(gb1) %*% gb1
  }
  gmm.acfb <- opm(par = c(0,0,0), fn = momentb1, method=c("BFGS"))
# gmm.acfb <- opm(par = c(0,0,0), fn = momentb1, lower= c(0,0,0), method=c("L-BFGS-B"))
  beta.acf[i,1] <- gmm.acfb$p1
  beta.acf[i,2] <- gmm.acfb$p2
  beta.acf[i,3] <- gmm.acfb$p3
}

png('fig5.png')
par(mfrow=c(1,3))
plot(density(beta.acf[beta.acf[,1]>0,1]), main = 'Intercept');
abline(v = gmm.acf$p1, col = 'blue')
plot(density(beta.acf[beta.acf[,1]>0,2]), main = 'Labor');
abline(v = gmm.acf$p2, col = 'blue')
plot(density(beta.acf[beta.acf[,1]>0,3]), main = 'Capital');
abline(v = gmm.acf$p3, col = 'blue')
par(mfrow=c(1,1))
dev.off()

sd(beta.acf[,1])
sd(beta.acf[,2])
sd(beta.acf[,3])

# Scale

scale[4,2] <- gmm.acf$p2 + gmm.acf$p3
scale[4,3] <- sqrt(cov(beta.acf)[2,2] + cov(beta.acf)[3,3] + 2 * cov(beta.acf)[2,3])


# Output -----------------------------------------------------------------------

stargazer(mod1, mod2fe, title="Production Function Estiamtion OLS",
```

```
style = "qje", notes.append = FALSE, notes.align = "l",
notes = "This will be replaced")
```