Problem Set #5

Anaya Hall and Christian Miller

5/2/2018

Part 1: Theory

(Optional – skip for now!)

Part 2: Instrumental Variables

Question 1: NLS80

Revisit the model from *Problem Set #3*, now including ability.

$log(wage) = \beta_0 + exper \cdot \beta_1 + tenure \cdot \beta_2 + married \cdot \beta_3 + south \cdot \beta_4 + urban \cdot \beta_5 + black \cdot \beta_6 + educ \cdot \beta_7 + abil \cdot \gamma + \epsilon$

```
# Read in CSV as data.frame
wage_df <- readr::read_csv("nls80.csv")

# Select only the variables in our model
wage_df %<>% select(lwage, wage, exper, tenure, married, south, urban, black, educ)
```

(a) Bias of coefficient on education

Derive the bias of β_7. Show which direction the bias goes in depending on whether the correlation between ability and education is positive or negative.

First, let's load our OLS function.

```
# Function to convert tibble, data.frame, or tbl_df to matrix
to_matrix <- function(the_df, vars) {
  # Create a matrix from variables in var
  new_mat <- the_df %>%
    #Select the columns given in 'vars'
    select_(.dots = vars) %>%
    # Convert to matrix
    as.matrix()
  # Return 'new_mat'
  return(new_mat)
}


b_ols <- function(y, X) {
  # Calculate beta hat
  beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y
  # Return beta_hat
  return(beta_hat)
```

```
}

ols <- function(data, y_data, X_data, intercept = T, H0 = 0, two_tail = T, alpha = 0.05) {
  # Function setup ----
    # Require the 'dplyr' package
    require(dplyr)

  # Create dependent and independent variable matrices ----
    # y matrix
    y <- to_matrix (the_df = data, vars = y_data)
    # X matrix
    X <- to_matrix (the_df = data, vars = X_data)
      # If 'intercept' is TRUE, then add a column of ones
      if (intercept == T) {
      X <- cbind(1,X)
      colnames(X) <- c("intercept", X_data)
      }

  # Calculate b, y_hat, and residuals ----
    b <- solve(t(X) %*% X) %*% t(X) %*% y
    y_hat <- X %*% b
    e <- y - y_hat

  # Useful -----
    n <- nrow(X) # number of observations
    k <- ncol(X) # number of independent variables
    dof <- n - k # degrees of freedom
    i <- rep(1,n) # column of ones for demeaning matrix
    A <- diag(i) - (1 / n) * i %*% t(i) # demeaning matrix
    y_star <- A %*% y # for SST
    X_star <- A %*% X # for SSM
    SST <- drop(t(y_star) %*% y_star)
    SSM <- drop(t(b) %*% t(X_star) %*% X_star %*% b)
    SSR <- drop(t(e) %*% e)

  # Measures of fit and estimated variance ----
    R2uc <- drop((t(y_hat) %*% y_hat)/(t(y) %*% y)) # Uncentered R^2
    R2 <- 1 - SSR/SST # Uncentered R^2
    R2adj <- 1 - (n-1)/dof * (1 - R2) # Adjusted R^2
    AIC <- log(SSR/n) + 2*k/n # AIC
    SIC <- log(SSR/n) + k/n*log(n) # SIC
    s2 <- SSR/dof # s^2

  # Measures of fit table ----
    mof_table_df <- data.frame(R2uc, R2, R2adj, SIC, AIC, SSR, s2)
    mof_table_col_names <- c("$R^2_\\text{uc}$", "$R^2$",
                             "$R^2_\\text{adj}$",
                             "SIC", "AIC", "SSR", "$s^2$")
    mof_table <-  mof_table_df %>% knitr::kable(
      row.names = F,
```

```
    col.names = mof_table_col_names,
    format.args = list(scientific = F, digits = 4),
    booktabs = T,
    escape = F
  )

# t-test----
  # Standard error
  se <- as.vector(sqrt(s2 * diag(solve(t(X) %*% X))))
  # Vector of _t_ statistics
  t_stats <- (b - H0) / se
  # Calculate the p-values
  if (two_tail == T) {
  p_values <- pt(q = abs(t_stats), df = dof, lower.tail = F) * 2
  } else {
    p_values <- pt(q = abs(t_stats), df = dof, lower.tail = F)
  }
  # Do we (fail to) reject?
  reject <- ifelse(p_values < alpha, reject <- "Reject", reject <- "Fail to Reject")

  # Nice table (data.frame) of results
  ttest_df <- data.frame(
    # The rows have the coef. names
    effect = rownames(b),
    # Estimated coefficients
    coef = as.vector(b) %>% round(3),
    # Standard errors
    std_error = as.vector(se) %>% round(4),
    # t statistics
    t_stat = as.vector(t_stats) %>% round(3),
    # p-values
    p_value = as.vector(p_values) %>% round(4),
    # reject null?
    significance = as.character(reject)
    )

  ttest_table <-  ttest_df %>% knitr::kable(
    col.names = c("", "Coef.", "S.E.", "t Stat", "p-Value", "Decision"),
    booktabs = T,
    format.args = list(scientific = F),
    escape = F,
    caption = "OLS Results"
  )

# Data frame for exporting for y, y_hat, X, and e vectors ----
  export_df <- data.frame(y, y_hat, e, X) %>% tbl_df()
  colnames(export_df) <- c("y","y_hat","e",colnames(X))

# Return ----
  return(list(n=n, dof=dof, b=b, vars=export_df, R2uc=R2uc,R2=R2,
```

```
                  R2adj=R2adj, AIC=AIC, SIC=SIC, s2=s2, SST=SST, SSR=SSR,
                  mof_table=mof_table, ttest=ttest_table))
}
```

Question 2: Card

```r
# Read in CSV as data.frame
card_df <- readr::read_csv("card.csv")

# Select only the variables in our model
wage_df %<>% select(lwage, wage, exper, tenure, married, south, urban, black, educ)
```

FUNCTIONS FROM PS4

```r
# First order only
BGtest <- function(data, y_data, X_data, order = 1) {
  y <- to_matrix(data, y_data)
  X <- to_matrix(data, X_data)
  Z <- X #duplicate rhs matrix

  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e
  # Generate 1 time period lagged residuals
  e_lag <- lag(e0)
  e_lag[is.na(e_lag)] <- 0 # Replace NA with zeros

  Z <- cbind(Z, e0)

  # Add column of lagged residuals
  Z <- cbind(NA, Z)
  colnames(Z)[1] <- "e_lag"

  # First, convert Z to dataframe for lagging operation
  c <- as.data.frame(Z)

  # Create lagged residuals
    for (i in 1:nrow(Z)) {
    if (i == 1)
      c$e_lag[[i]] = 0
    else
      c$e_lag[[i]] = c$e0[i-1]
    }

  # Back to matrix
  X0 <- c[-ncol(c)] %>% as.matrix()
  # BG_df <- cbind(data[y_data], X0)
  BG_df <- c

  # Regress
  # Not regressing on all lhs vars (colnames(X0)), just e_lag
  R2_stat <- ols(BG_df, "e0", "e_lag")$R2
  test_stat <- R2_stat*nrow(data)
```

```r
  pvalue <- 1 - pchisq(test_stat, df = 1)

  return(data_frame("Test Statistic" = test_stat,
             "P-Value" = pvalue))

}


# ols(data = gdp_data, y_data = "delta_p", X_data = c("Year", "Realgdp", "Realcons", "Realinvs",
```

```r
BPtest <- function(data, y_data, X_data, order = 1) {
  y <- to_matrix(data, y_data)
  X <- to_matrix(data, X_data)
  Z <- to_matrix(data, X_data)

  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e
  Z <- cbind(Z, e0)

  # Add column of for lagged residuals
  Z <- cbind(NA, Z)
  colnames(Z)[1] <- "e_lag"

  # First, convert Z to dataframe for lagging operation
  c <- as.data.frame(Z)

  # Create lagged residuals
    for (i in 1:nrow(Z)) {
    if (i == 1)
      c$e_lag[[i]] = 0
    else
      c$e_lag[[i]] = c$e0[i-1]
    }

  # Back to matrix
  X0 <- c[-ncol(c)] %>% as.matrix()
  # BG_df <- cbind(data[y_data], X0)
  BP_df <- c

  # Regress e on lagged variables, save coefficient on e_lag
  # Not regressing on all lhs vars (colnames(X0)), just e_lag
  lag_coef <- ols(BP_df, "e0", "e_lag")$b[2]
  test_stat <- nrow(BP_df) * lag_coef^2

  pvalue <- 1 - pchisq(test_stat, df = order)

  # return(lag_coef)
  return(data_frame("Test Statistic" = test_stat,
             "P-Value" = pvalue))
```

```
  }
```

```
DWtest <- function(data, y_data, X_data) {
   y <- to_matrix(data, y_data)
  X <- to_matrix(data, X_data)
  Z <- to_matrix(data, X_data)

  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e
  Z <- cbind(Z, e0)

  # Add column of for lagged residuals
  Z <- cbind(NA, Z)
  colnames(Z)[1] <- "e_lag"

  # First, convert Z to dataframe for lagging operation
  c <- as.data.frame(Z)

  # Create lagged residuals
    for (i in 1:nrow(Z)) {
    if (i == 1)
      c$e_lag[[i]] = 0
    else
      c$e_lag[[i]] = c$e0[i-1]
    }

  # # Back to matrix
  # X0 <- c[-ncol(c)] %>% as.matrix()
  # # BG_df <- cbind(data[y_data], X0)
  # DW_df <- c
  #
  # # Regress e on lagged variables, save coefficient on e_lag
  # lag_coef <- ols(DW_df, "e0", colnames(X0))$b[2]

  # numerator summing from t=2
  c1 <- c[-1,]
  d_numer <- sum((c1$e0 - c1$e_lag)^2)
  d_denom <- sum((c$e0)^2)

  # Test statistic
  d <- d_numer/d_denom

  return("Test Statistic" = d)

}
```

```
rho_ols <- function(data, y_data, X_data, order = 1) {
  y <- to_matrix(data, y_data)
  X <- to_matrix(data, X_data)
  Z <- to_matrix(data, X_data)
```

```r
  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e
  Z <- cbind(Z, e0)

  # Add column of for lagged residuals
  Z <- cbind(NA, Z)
  colnames(Z)[1] <- "e_lag"

  # First, convert Z to dataframe for lagging operation
  c <- as.data.frame(Z)

  # Create lagged residuals
    for (i in 1:nrow(Z)) {
    if (i == 1)
      c$e_lag[[i]] = 0
    else
      c$e_lag[[i]] = c$e0[i-1]
    }

  # Back to matrix
  X0 <- c[-ncol(c)] %>% as.matrix()
  # BG_df <- cbind(data[y_data], X0)
  lagged_df <- c

  # Regress e on lagged variables, save coefficient on e_lag
  # (this is index position 2, after the intercept)
  rho_hat <- ols(lagged_df, "e0", "e_lag")$b[2]

  # Alternate way of calculating, gets the same value!
  # c1 <- c[-1,]
  # rho_numer <- sum((c1$e0 * c1$e_lag))
  # rho_denom <- sum((c$e0)^2)
  # rho_1 <- rho_numer/rho_denom

  # return(lag_coef)
  return(rho_hat)

}

# Check if function works
# rho_ols(data = gdp_data, y_data = "delta_p", X_data = c("Year", "Realgdp", "Realcons", "Realinvs
```

```r
fgls_sc <- function(data, y_var, X_vars, Z_vars, intercept = T, procedure = "prais") {
  # Turn data into matrices
  y <- to_matrix(data, y_var)
  X <- to_matrix(data, X_vars)
  Z <- to_matrix(data, Z_vars)
  # Add intercept
  if (intercept == T) X <- cbind(1, X)
```

```r
  if (intercept == T) Z <- cbind(1, Z)
  # Calculate n and k for degrees of freedom
  t <- nrow(X)
  k <- ncol(X)
  # Update names
  #if (intercept == T) rownames(b)[1] <- "Intercept"

  # ESTIMATE RHO
  if (procedure == "prais")
      sc_data <- data
  else if (procedure == "cochrane")
      sc_data <- data[-1,]

  # Estimate rho for use in FGLS
  rho <- rho_ols(sc_data, y_var, X_vars)
  rho_hat <- rho * (t-k)/(t-1)

  # CREATE LOOP THAT BUILDS C MATRIX (is this the right number of rows/columns??)
   G <- matrix(NA, nrow=t, ncol =t)
    for (i in 1:t) {
      for (j in 1:t) {
        if (j == 1 & i == 1) G[i,j] <- sqrt(1-rho)
        else if (i == j) G[i,j] <- 1
        else if (i == j + 1) G[i,j] <- -rho
        else G[i,j] <- 0
      }
    }

  # Re-weight y and X
  y_tilde <- G %*% y
  X_tilde <- G %*% X

  # Combine the transformed data and run OLS on them
  colnames(X_tilde)[1] <- "Intercept"
  tilde <- cbind(y_tilde, X_tilde) %>% data.frame()

  results <- ols(
    data = tilde,
    y_data =  colnames(tilde)[1],
    X_data  = colnames(tilde)[2:ncol(tilde)],
    intercept = F)
  # Return the results

  return(results)
}
```

(i) Hildreth Lu procedure

```
rho <- 0.4
data$one <- 1

hildreth_lu <- function(data, y, X1, one, rho) {

#Converting variables to vectors
y <- select_(data, y) %>% unlist()
X1 <- select_(data, X1) %>% unlist()
one <- select_(data, one) %>% unlist()

#Modifying data based on rho
y_lag <- lag(y)
y_star <- y - rho * y_lag
y_star[1] <- sqrt(1-rho^2) * y[1]


X1_lag <- lag(X1)
X1_star <- X1 - rho * X1_lag
X1_star[1] <- sqrt(1-rho^2) * X1[1]


X2_star <- one - rho
X2_star[1] <- sqrt(1-rho^2)

# Turn data into matrices
  y <- cbind(y_star)

  X <- cbind(X1_star, X2_star)

#Defining n and k
n <- nrow(X)
k <- ncol(X)

# Estimate coefficients
  b <- b_ols(y, X)

# Calculate OLS residuals
  e <- y - X %*% b
# Calculate s^2
  s2 <- (t(e) %*% e) / (n-k)

#Calculate Log likelihood function
  log_L <- -1*((t(e) %*% e)/ 2*s2) +0.5*log(1-rho^2) -n/2*(log(2*pi) + log(s2))

  (log_L)

  # Return the results
  return(log_L)
```

```
}

#Testing for rho = 0.4 before running loop
hildreth_lu(data, "y", "unemployment", "one", rho)

#Defining vector of rho's
rhos <- seq(-0.95,0.95, by=0.01)

#Running in a loop
(LL <- sapply(X = rhos,
             FUN = hildreth_lu,
             data = data, y = "y", X1 = "unemployment",
             one = "one"))

(rhos[which(LL==max(LL))])
```

Notes on Parallelizing

library(parallel) res <- mclapply(query, GET, mc.cores = 4) map_df(res, function)