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, iq)
```

(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.

$plimb_7 = \beta_7 + \gamma\delta_7$

Assume that all δ's are zero except for the one on the variable of interest (education)

$plimb_7 = \beta_7 + \gamma \cdot \frac{Cov[abil,educ]}{Var[educ]}$

Truth is β_7 , bias is $\gamma \cdot \frac{Cov[abil,educ]}{Var[educ]}$

We expect the sign on γ to be positive (higher abiltiy should lead to higher wage), the covariance of ability and education to also be positive (more able people acheive higher levels of education), and, of course, the variance of education is positive. Thus, the bias will also be *positive*.

(b) Proxy for ability

Estimate the model above excluding ability, record your parameter estimates, standard errors and R^2.

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)
```

```r
# 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)
    )
```

```r
    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))
}
```

```r
model_1 <- ols(wage_df, y_data = "lwage",
               X_data = c("exper", "tenure", "married", "south", "urban", "black", "educ"))

model_1$ttest
```

Table 1: OLS Results

	Coef.	S.E.	t Stat	p-Value	Decision
intercept	5.395	0.1132	47.653	0.0000	Reject
exper	0.014	0.0032	4.409	0.0000	Reject
tenure	0.012	0.0025	4.789	0.0000	Reject
married	0.199	0.0391	5.107	0.0000	Reject
south	-0.091	0.0262	-3.463	0.0006	Reject
urban	0.184	0.0270	6.822	0.0000	Reject
black	-0.188	0.0377	-5.000	0.0000	Reject
educ	0.065	0.0063	10.468	0.0000	Reject

```r
model_1$mof
```

R^2_{uc}	R^2	R^2_{adj}	SIC	AIC	SSR	s^2
0.9971	0.2526	0.2469	-1.963	-2.005	123.8	0.1336

(c) Include IQ

(c) Estimate the model including IQ as a proxy, record your parameter estimates, standard errors and R^2.

```r
model_iq <- ols(wage_df, y_data = "lwage",
               X_data = c("exper", "tenure", "married", "south", "urban", "black", "educ", "iq"))

model_iq$ttest
```

Table 3: OLS Results

	Coef.	S.E.	t Stat	p-Value	Decision
intercept	5.176	0.1280	40.441	0.0000	Reject
exper	0.014	0.0032	4.469	0.0000	Reject
tenure	0.011	0.0024	4.671	0.0000	Reject
married	0.200	0.0388	5.148	0.0000	Reject
south	-0.080	0.0263	-3.054	0.0023	Reject
urban	0.182	0.0268	6.791	0.0000	Reject
black	-0.143	0.0395	-3.624	0.0003	Reject
educ	0.054	0.0069	7.853	0.0000	Reject
iq	0.004	0.0010	3.589	0.0004	Reject

```r
model_iq$mof
```

R^2_{uc}	R^2	R^2_{adj}	SIC	AIC	SSR	s^2
0.9972	0.2628	0.2564	-1.97	-2.016	122.1	0.1319

(d) Returns on education.

What happens to returns to schooling? Does this result confirm your suspicion of how ability and schooling are expected to be correlated?

Magnitude of the parameter estimate on education decreased. If IQ is a good proxy for ability, this does confirm our suspicion that ability is correlated with education. In the first model, some of the returns on ability (IQ) were mis-attributed to education. In the second model, we correct for this, and see that the parameter estimate on ability is indeed significant.

Question 2: Recreate results from Card

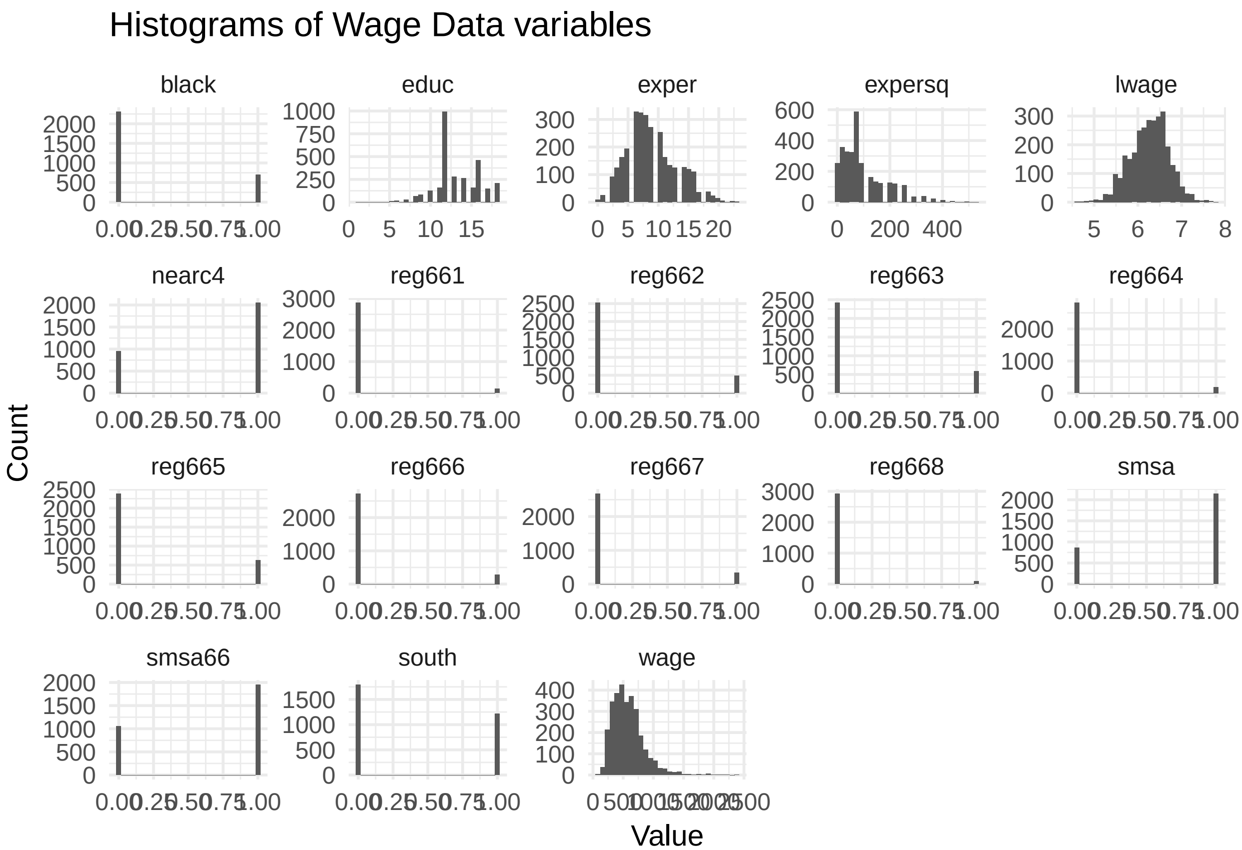
(a) Read in data & plot

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

# Select only the variables in our model
card_df %<>% select(lwage, wage, educ, exper, expersq, black, south, smsa, smsa66, reg661, reg662,

head(card_df)
```

```
## # A tibble: 6 x 18
##   lwage  wage  educ exper expersq black south  smsa smsa66 reg661 reg662
##   <dbl> <int> <int> <int>   <int> <int> <int> <int>  <int>  <int>  <int>
## 1  6.31   548     7    16     256     1     0     1      1      1      0
## 2  6.18   481    12     9      81     0     0     1      1      1      0
## 3  6.58   721    12    16     256     0     0     1      1      1      0
## 4  5.52   250    11    10     100     0     0     1      1      0      1
## 5  6.59   729    12    16     256     0     0     1      1      0      1
## 6  6.21   500    12     8      64     0     0     1      1      0      1
## # ... with 7 more variables: reg663 <int>, reg664 <int>, reg665 <int>,
## #   reg666 <int>, reg667 <int>, reg668 <int>, nearc4 <int>
```

```
ggplot(data = gather(card_df), aes(x = value)) +
  geom_histogram() +
  facet_wrap(~ key, scales = "free") +
  ggtitle("Histograms of Wage Data variables") +
  ylab("Count") +
  xlab("Value") + theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



(b) OLS on log(wage)

```r
rhs_vars <- c("educ", "exper", "expersq", "black", "south", "smsa", "smsa66", "reg661", "reg662",

model1 <- ols(card_df, "lwage", rhs_vars)

model1$ttest
```

Table 5: OLS Results

	Coef.	S.E.	t Stat	p-Value	Decision
intercept	4.739	0.0715	66.259	0.0000	Reject
educ	0.075	0.0035	21.351	0.0000	Reject
exper	0.085	0.0066	12.806	0.0000	Reject
expersq	-0.002	0.0003	-7.223	0.0000	Reject
black	-0.199	0.0182	-10.906	0.0000	Reject
south	-0.148	0.0260	-5.695	0.0000	Reject
smsa	0.136	0.0201	6.785	0.0000	Reject
smsa66	0.026	0.0194	1.349	0.1773	Fail to Reject
reg661	-0.119	0.0388	-3.054	0.0023	Reject
reg662	-0.022	0.0283	-0.786	0.4321	Fail to Reject
reg663	0.026	0.0274	0.949	0.3427	Fail to Reject
reg664	-0.063	0.0357	-1.780	0.0753	Fail to Reject
reg665	0.009	0.0361	0.262	0.7935	Fail to Reject
reg666	0.022	0.0401	0.547	0.5842	Fail to Reject

	Coef.	S.E.	t Stat	p-Value	Decision
reg667	-0.001	0.0394	-0.015	0.9881	Fail to Reject
reg668	-0.175	0.0463	-3.777	0.0002	Reject

(c) Reduced Form

```
rhs_vars <- c("nearc4", "exper", "expersq", "black", "south", "smsa", "smsa66", "reg661", "reg662'

rf <- ols(card_df, "educ", rhs_vars)

rf$ttest
```

Table 6: OLS Results

	Coef.	S.E.	t Stat	p-Value	Decision
intercept	16.849	0.2111	79.805	0.0000	Reject
nearc4	0.320	0.0879	3.641	0.0003	Reject
exper	-0.413	0.0337	-12.241	0.0000	Reject
expersq	0.001	0.0017	0.526	0.5987	Fail to Reject
black	-0.936	0.0937	-9.981	0.0000	Reject
south	-0.052	0.1354	-0.381	0.7032	Fail to Reject
smsa	0.402	0.1048	3.837	0.0001	Reject
smsa66	0.025	0.1058	0.241	0.8096	Fail to Reject
reg661	-0.210	0.2025	-1.039	0.2991	Fail to Reject
reg662	-0.289	0.1473	-1.961	0.0500	Reject
reg663	-0.238	0.1426	-1.670	0.0950	Fail to Reject
reg664	-0.093	0.1860	-0.501	0.6167	Fail to Reject
reg665	-0.483	0.1882	-2.566	0.0103	Reject
reg666	-0.513	0.2096	-2.448	0.0144	Reject
reg667	-0.427	0.2056	-2.077	0.0379	Reject
reg668	0.314	0.2417	1.298	0.1945	Fail to Reject
Yes, the par	tial corr	elation I	S statisti	cally sign	ificant!

(d) Single IV

Estimate the log(wage) equation by instrumental variables, using nearc4 as an instrument for educ. Compare the 95% confidence interval for the return to educutioan to that obtained from the Least Squares regression above.

(e) Multiple IV

Use nearc2 and nearc4 as instruments for educ. Comment on the significance of the partial correlations of both instruments in the reduced form. Show your standard errors from the second stage and compare them to the correct standard errors.

(f) Hausman test

Conduct a Hausman test for endogeneity of educ. Report your test statistic, critical value and p-value.

FUNCTIONS FROM PS4

```
# 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)

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

  return(data_frame("Test Statistic" = test_stat,
```

```r
              "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)

  # 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", "Realinv
```

```
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)
  if (intercept == T) Z <- cbind(1, Z)
  # Calculate n and k for degrees of freedom
```

```r
  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

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

```r
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)
```

```r
#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)