Problem Set #4

Anaya Hall & Christian Miller

Due April 25th

Serial Correlation

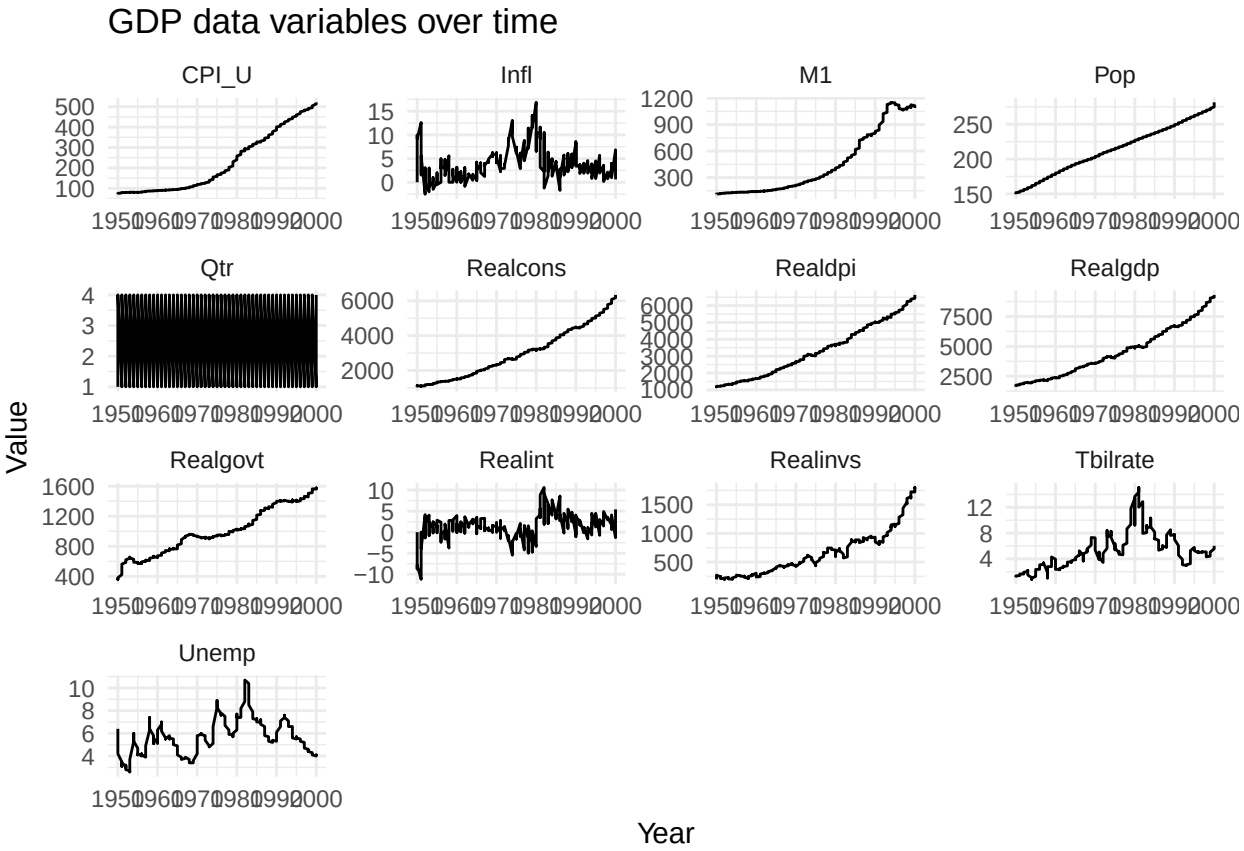
The goal of this problem set is to explore what happens when we have *serially correlated distrubances*.

Question 1

Read the data into R. Plot the series against time and make sure your data are read in correctly. Also, print out data as ascii file and compare the first and last row to make sure there's no funny business with how the data were read in. Check a few points in the middle too.

```
# Column names from codebook
names <- c("Year", "Qtr", "Realgdp", "Realcons", "Realinvs", "Realgovt", "Realdpi", "CPI_U", "M1",

# Read in txt file as data.frame using column names from codebook
gdp_data <- readr::read_table2("data.txt",
                               col_names = names)
```

```
#Plot the variables in our model against time
ggplot(data = gather(gdp_data, key, value, -Year), aes(x = Year, y = value)) +
  geom_line() +
  facet_wrap(~ key, scales = "free") +
  ggtitle("GDP data variables over time") +
  ylab("Value") +
  xlab("Year") + theme_minimal()
```




```
write.table(x = gdp_data, file = "data_ascii")

# ascii(x = gdp_data, include.rownames = T)
```

So far, everthing looks good.

Question 2: Phillips Curve

Estimate the estimations augmented Phillips Curve (see Greene p. 251)

Equation:

$\Delta p_t - \Delta p_{t-1} = \beta_1 + \beta_2 \cdot u_t + \epsilon_t$

(a) Generate dependent variable

Hint: Check the codebook; may need to drop one of our variables.

Need to drop the first row because the first observation for Infl is missing Phillip's curve regresses inflation (%) on unemployment (%)

```
# Drop first observation (row)
gdp_data <- gdp_data[-1,]

# Generate Dependent Variable
gdp_data %<>% mutate(delta_p = Infl - lag(Infl))
gdp_data <- gdp_data[-1,] # to drop the first NA
```

```
# SHOULD I CONCATENATE YEAR and QUARTER? That way I don't have to drop quarter!
gdp_data %<>%  transform(yr_qtr = as.numeric(paste(Year, Qtr, sep=".")))
```

(b) Estimate relationship

Estimate relationship above. Report parameter estimates, standard errors, t-statistics 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
```

```r
# 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 # Centered 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))
}
```

```
model_1 <- ols(gdp_data,
               y_data = "delta_p",
               X_data = "Unemp")

model_1$ttest
```

Table 1: OLS Results

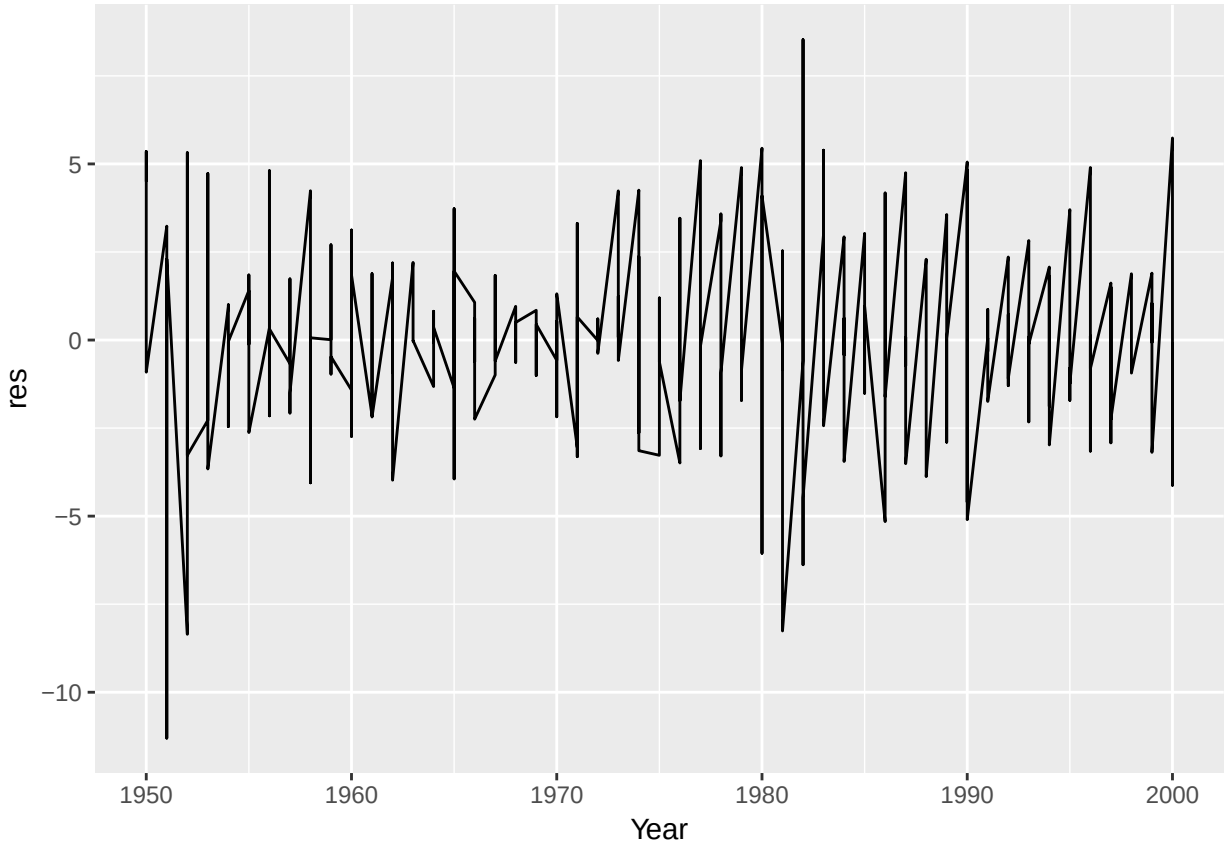
	Coef.	S.E.	t Stat	p-Value	Decision
intercept	0.518	0.7432	0.697	0.4868	Fail to Reject
Unemp	-0.091	0.1263	-0.719	0.4731	Fail to Reject

```
model_1$mof_table
```

R^2_{uc}	R^2	R^2_{adj}	SIC	AIC	SSR	s^2
0.002565	0.002564	-0.002398	2.125	2.092	1613	8.023

(c) Plot residuals against time

```
gdp_data$res <- model_1$vars$e

ggplot(gdp_data, aes(x=Year, y=res)) + geom_line()
```



Looking at this plot, we likely have a **NEGATIVE** auto-correlation issue!

(d) Breusch-Godfrey

Use Breusch-Godfrey test to test for first order correlation

Procedure 1. Run OLS & save residuals 2. Augment with column of lagged residuals –> X0 (fill in any NAs with zeros) 3. Auxillary regression: regress residuals et on X0t 4. Test Statistic:

$LM = T \cdot R_0^2$

Null hypothesis is no serial correlation, or $H_0 : LM < \chi^2$ critical value. Our alternate hypothesis is first order serial correlation, or $H_A : LM > \chi^2$ critical value.

```
# 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
  Z <- cbind(1,X) # adding a column of ones
  colnames(Z)[1] <- "intercept"

  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e %>% as.matrix()
  # Generate time period lagged residuals
  e_lag <- as.data.frame(matrix(nrow = nrow(Z), ncol = order))
  for (i in 1:order) {
        e_lag[i] <- lag(e0,i)
  }
  e_lag[is.na(e_lag)] <- 0 # Replace NA with zeros
  Z %<>% cbind(e_lag) %>% as.matrix() # add to regressor matrix


  # Regress, Calculate b, y_hat, and residuals
  b <- solve(t(Z) %*% Z) %*% t(Z) %*% e0
  y_hat <- Z %*% b
  resid <- e0 - y_hat

  # Calculate R^2
  n <- nrow(Z) # number of observations
  i <- rep(1,n) # column of ones for demeaning matrix
  A <- diag(i) - (1 / n) * i %*% t(i) # demeaning matrix
  y_star <- A %*% e0 # for SST
  SST <- drop(t(y_star) %*% y_star)
  SSR <- drop(t(resid) %*% resid)
  R2 <- 1 - SSR/SST # Centered R^2

  LM_stat <- R2*(n-order)

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

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

```
}

(BGtest(data = gdp_data, y_data = "delta_p", X_data = "Unemp", order = 1))
```

```
## # A tibble: 1 x 2
##   `Test Statistic`     `P-Value`
##              <dbl>         <dbl>
## 1             33.2 0.00000000839
```

Given that our test statistic is 36.41983 and p-value of $1.591 * 10^{-9}$, we reject the null hypothesis of no serial correlation.

(e) Box-Pierce

Use Box-Pierce test to test for first order correlation. Report test statistic and pvalue. Our null hypothesis is no auto correlation, or $H_0 : Q < \chi^2$ critical value. Our alternate hypothesis is first order auto correlation, or $H_A : Q > \chi^2$ critical value.

Test-statistic: $Q = T \cdot \Sigma \hat{\rho}_j^2$ distributed χ_p^2

where $\hat{\rho}_j^2$ are the OLS coefficients from a regression of e_t on the j^{th} lag (without an intercept!) $\hat{\rho}_j^2 = \frac{\Sigma_{t=j+1} e_t e_{t-j}}{\Sigma_{t=1} e_t^2}$

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

  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e %>% as.matrix()

  # Generate time period lagged residuals
  e_lag <- data.frame(matrix(nrow = nrow(e0), ncol = order))
  for (i in 1:order) {
        e_lag[i] <- lag(e0,i)
  }
  e_lag[is.na(e_lag)] <- 0 # Replace NA with zeros
  e_lag %<>% as.matrix

  p = (t(e_lag) %*% e0) %*% solve((t(e0) %*% e0))

  Q_stat <- nrow(e0) * sum(p^2)

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

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

  }

(BPtest(data = gdp_data, y_data = "delta_p", X_data = "Unemp"))
```

```
## # A tibble: 1 x 2
##   `Test Statistic`     `P-Value`
##              <dbl>         <dbl>
## 1             33.2 0.00000000836
```

We have a test statistic of 36.43994 and p-value of $1.575 * 10^{-9}$, and again, we reject the null hypothesis that there is no autocorrelation!

(f) Durbin Watson

Use the Durbin Watson test to test for first order autocorrelation. Report test statistic and interpret. Our null hypothesis is no auto correlation, or $H_0 : d > 2$ or $d > 1.779$ at 95 percent confidence level. Our alternate hypothesis is first order positive auto correlation, or $H_A : d < 2$ or $d < 1.758$at 95 percent confidence level.

Test statistic: $d = \frac{\Sigma_{t=2}(e_t - e_{t-1})^2}{\Sigma_{t=1} e_t^2}$

```
DWtest <- function(data, y_data, X_data) {
  require(stats)
  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 %>% as.matrix()


  # Generate time period lagged residuals
  e_lag <- lag(e0)
  e_lag[is.na(e_lag)] <- 0 # Replace NA with zeros

  # numerator summing from t=2
  d_numer <- sum((e0[2:nrow(e0),] - e_lag[2:nrow(e_lag),])^2)
  d_denom <- sum((e0)^2)

  # Test statistic
  d <- d_numer/d_denom

  return("Test Statistic" = d)

}

(DWtest(data = gdp_data, y_data = "delta_p", X_data = "Unemp"))
```

```
## [1] 2.792357
```

Given our test statistic of 2.828, we reject the null hypothesis of no auto correlation. Since our test statistic is greater than 2, we want to also test the null hypothesis of no serial correlation versus negative serial correlation. Using 4-d (1.172) as the test statistic, we cannot reject the null that there is no serial correlation.

(g) Prais Winsten procedure

Use the Prais Winsten procedure to correct for the serial correlation problem. Report parameter estimates, standard errors, and t-statistics. Plot new residuals against time.

Instead of $\hat{\rho}$ use $\hat{\rho}\frac{T-k}{T-1}$ in FLGS.

FGLS function from Problem Set #3

Function to return $\hat{\rho}$ estimator (for use in Prais Winsten and Cochrane Orcutt procedures)

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

  # Run OLS and save residuals to new covariate matrix
  e0 <- ols(data, y_data, X_data)$vars$e %>% as.matrix()
  # Generate time period lagged residuals
  e_lag <- as.data.frame(matrix(nrow = nrow(X), ncol = order))
  for (i in 1:order) {
        e_lag[i] <- lag(e0,i)
  }
  e_lag[is.na(e_lag)] <- 0 # Replace NA with zeros
  e_lag <- cbind(rep(1,nrow(e_lag)),e_lag)
  e_lag %<>% as.matrix()

  # Regress, Calculate b, y_hat, and residuals
  b <- solve(t(e_lag) %*% e_lag) %*% t(e_lag) %*% e0
  rho_hat <- b[2]
  # return(lag_coef)
  return(rho_hat)

}

# Check if function works
# (rho_ols(data = gdp_data, y_data = "delta_p", X_data = "Unemp"))
```

```
fgls_sc <- function(data, y_var, X_vars, Z_vars, intercept = T, procedure = "prais") {
  # Which procedure
  if (procedure == "prais") {
      sc_data <- data
      # Turn data into matrices
      y <- to_matrix(sc_data, y_var)
      X <- to_matrix(sc_data, X_vars)
      Z <- to_matrix(sc_data, Z_vars)
  } else if (procedure == "cochrane"){
      sc_data <- data[-1,]
        # Turn data into matrices
      y <- to_matrix(sc_data, y_var)
      X <- to_matrix(sc_data, X_vars)
      Z <- to_matrix(sc_data, Z_vars)
  }
```

```r
  # Add intercept
  if (intercept == T) X <- cbind(1, X)
  if (intercept == T) Z <- cbind(1, Z)
  # Calculate n and k for degrees of freedom
  t <- nrow(X)
  k <- ncol(X)

  # 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
   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^2)
        } else if (i == j & i != 1) { 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)
}

# Define covariates (again)
rhs_vars <- c("yr_qtr", "Realgdp", "Realcons", "Realinvs", "Realgovt", "Realdpi", "CPI_U", "M1", "

# Run the FGLS function
prais <- fgls_sc(
  data = gdp_data,
  y_var = "delta_p",
  X_vars = rhs_vars,
  Z_vars = rhs_vars,
```

```
  intercept = T)

# Report parameter values, standard errors and t-statistics
prais$ttest
```

Table 3: OLS Results

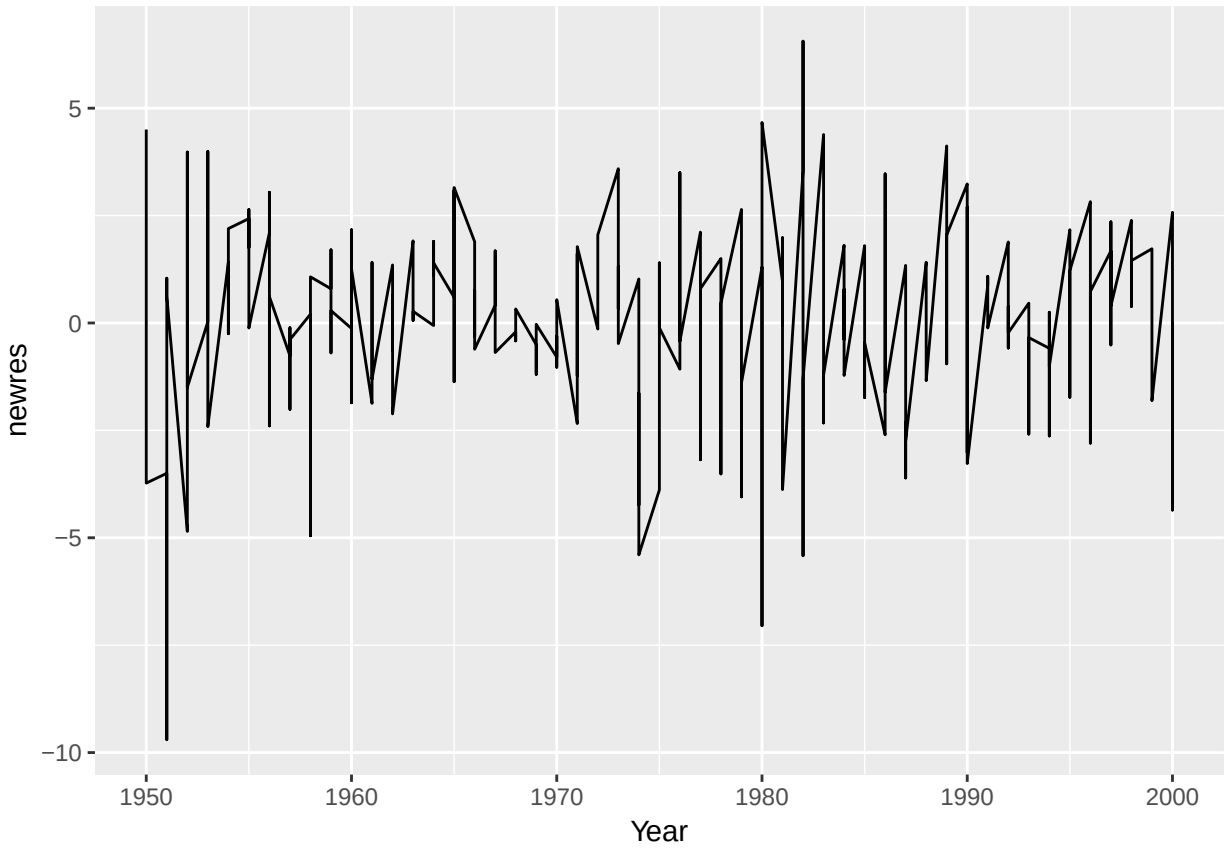
	Coef.	S.E.	t Stat	p-Value	Decision
Intercept	-1754.556	1004.7277	-1.746	0.0824	Fail to Reject
yr_qtr	0.910	0.5276	1.725	0.0861	Fail to Reject
Realgdp	-0.022	0.0055	-3.965	0.0001	Reject
Realcons	0.017	0.0065	2.590	0.0103	Reject
Realinvs	0.021	0.0052	4.102	0.0001	Reject
Realgovt	0.019	0.0058	3.305	0.0011	Reject
Realdpi	-0.006	0.0043	-1.300	0.1951	Fail to Reject
CPI_U	0.039	0.0169	2.303	0.0223	Reject
M1	0.002	0.0053	0.438	0.6619	Fail to Reject
Tbilrate	82.378	63.1323	1.305	0.1935	Fail to Reject
Unemp	-0.307	0.3349	-0.918	0.3599	Fail to Reject
Pop	-0.080	0.1663	-0.480	0.6321	Fail to Reject
Infl	-82.118	63.1202	-1.301	0.1949	Fail to Reject
Realint	-82.849	63.1291	-1.312	0.1910	Fail to Reject

```
newres <- prais$vars$e

(rho_ols(
  data = gdp_data,
  y_data = "delta_p",
  X_data = rhs_vars
))
```

```
## [1] -0.03262727
```

Most of the signs on the parameter estimates are the same, but the magnitude has changed. Some significance results have changed as well! $\hat{\rho}$ is -0.04408.

```
# plot(gdp_data$Year, newres)

ggplot(gdp_data, aes(x=Year, y=newres)) + geom_line()
```



Doesn't seem to have corrected the problem too much!

(h) Cochrane Orcutt procedure

Use the Cochrane Orcutt procedure to correct for the serial correlation problem. Report parameter estimates, standard errors, t-statistics and a plot of the new residuals against time. *Did dropping the initial observation change the results a lot?*

```
# Define covariates (again)
rhs_vars <- c("yr_qtr", "Realgdp", "Realcons", "Realinvs", "Realgovt", "Realdpi", "CPI_U", "M1", '

# Run the FGLS function
cochrane <- fgls_sc(
  data = gdp_data,
  y_var = "delta_p",
  X_vars = rhs_vars,
  Z_vars = rhs_vars,
  procedure = "cochrane",
  intercept = T)

# Report parameter values, standard errors and t-statistics
cochrane$ttest
```

Table 4: OLS Results

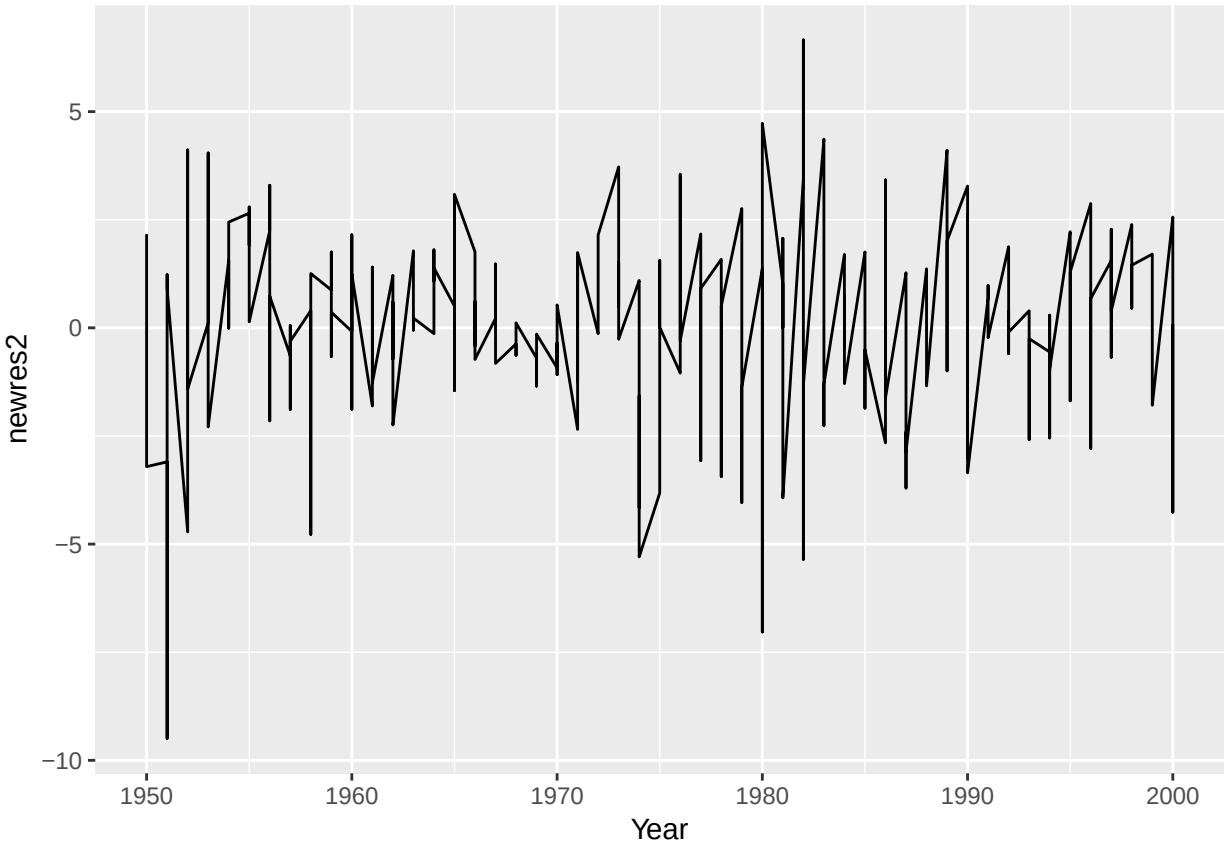
	Coef.	S.E.	t Stat	p-Value	Decision
Intercept	-1884.023	991.9992	-1.899	0.0591	Fail to Reject

	Coef.	S.E.	t Stat	p-Value	Decision
yr_qtr	0.977	0.5209	1.875	0.0623	Fail to Reject
Realgdp	-0.022	0.0054	-4.034	0.0001	Reject
Realcons	0.016	0.0064	2.584	0.0105	Reject
Realinvs	0.022	0.0052	4.245	0.0000	Reject
Realgovt	0.020	0.0057	3.489	0.0006	Reject
Realdpi	-0.006	0.0043	-1.488	0.1383	Fail to Reject
CPI_U	0.043	0.0168	2.546	0.0117	Reject
M1	0.001	0.0052	0.264	0.7917	Fail to Reject
Tbilrate	71.104	62.7208	1.134	0.2584	Fail to Reject
Unemp	-0.367	0.3303	-1.111	0.2682	Fail to Reject
Pop	-0.076	0.1638	-0.464	0.6432	Fail to Reject
Infl	-70.871	62.7080	-1.130	0.2598	Fail to Reject
Realint	-71.595	62.7168	-1.142	0.2551	Fail to Reject

```
newres2 <- cochrane$vars$e
```

The value of the coefficients have changed by a small amount. Th ecoefficient on Unemployment is only 0.014 bigger for Cochrane that that of teh Prais Winston method. Overall, the results are not significantly different.

```
# plot(gdp_data$Year, newres2)

ggplot(gdp_data[-1,], aes(x=Year, y=newres2)) + geom_line()
```



The plot still depicts serial correlation. Thus, maybe we have not fully corrected the problem of autocorrelation.

(i) Hildreth Lu procedure

Use the Hildreth Lu procedure discussed in the notes to correct for serial correlation. (Do a gridsearch for ρ over the interval [-0.95;0.95] using a 0.01 spacing.)

```
hildreth_lu <- function(data, y_var, X_vars, one, rho) {

#Converting variables to vectors
y <- select_(data, y_var) %>% unlist()
X1 <- select_(data, X_vars) %>% 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)
}
```

```
rho <- 0.4
gdp_data$one <- 1
#Testing for rho = 0.4 before running loop
hildreth_lu(gdp_data, "delta_p", "Unemp", "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 = gdp_data, y_var = "delta_p", X_vars = "Unemp",
             one = "one")

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

Using this method, we calculate the MLE for the log likelihood function and get a rho of -0.43. This is an order of magnitude off from what we got for the Prais-Winston and Cochrane-Orcutt estimate!

Notes on Parallelizing

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