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*.

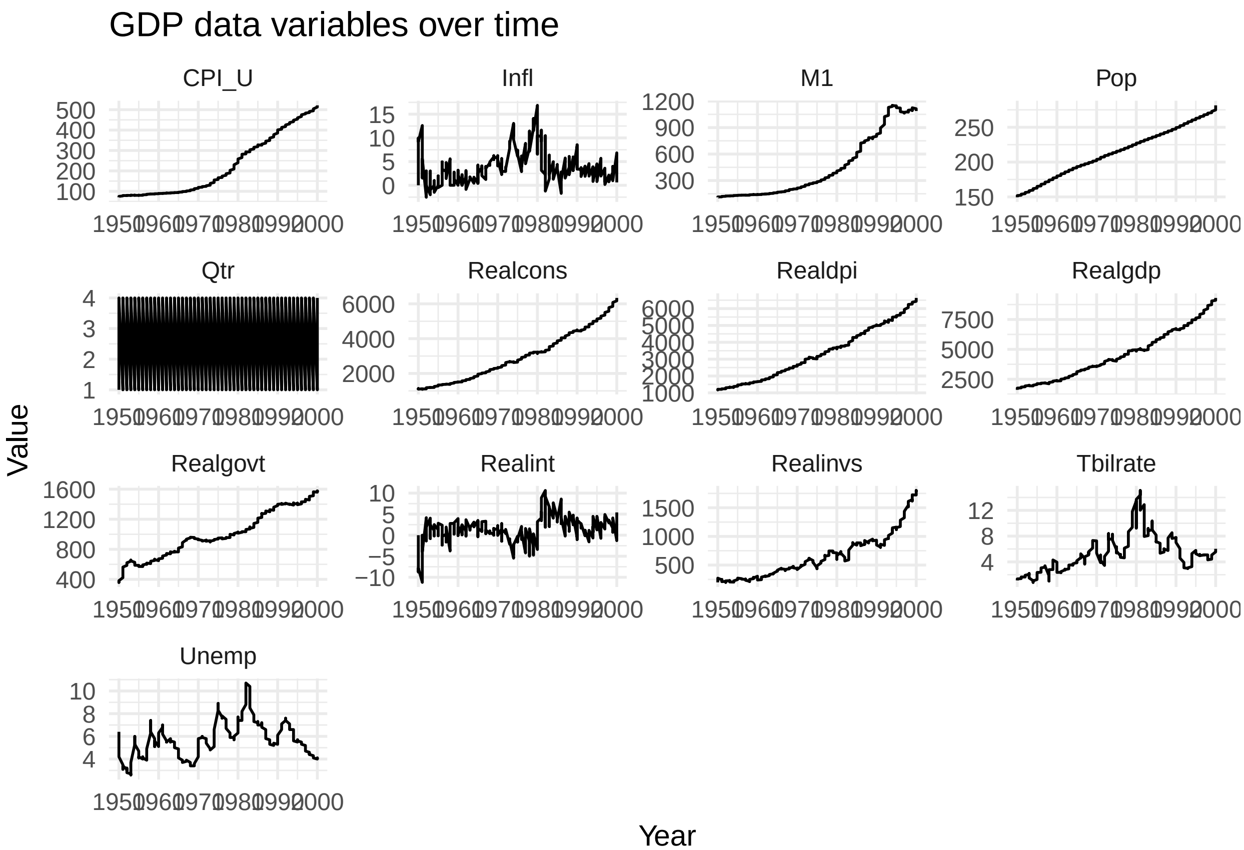
Question 1

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()
```




```r
write.table(x = gdp_data, file = "test_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 (%)

```r
# Generate Dependent Variable
for (i in 1:nrow(gdp_data)) {
                        if (i==1)
                        gdp_data$delta_p[i] = NA
                        else
                          gdp_data$delta_p[i] = gdp_data$Infl[i] - gdp_data$Infl[i-1] }
```

```
## Warning: Unknown or uninitialised column: 'delta_p'.
```

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

(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)
}

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
```

```r
  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),
```

```r
      # 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))
}
```

```r
# What is the right model here??--- all covariates?
covariates <- c("Year", "Qtr", "Realgdp", "Realcons", "Realinvs", "Realgovt", "Realdpi", "CPI_U",

model_1 <- ols(gdp_data,
               y_data = "delta_p",
               X_data =c("Year", "Realgdp", "Realcons", "Realinvs", "Realgovt", "Realdpi", "CPI_U'

model_1$ttest
```

Table 1: OLS Results

	Coef.	S.E.	t Stat	p-Value	Decision
intercept	-1932.304	811.8926	-2.380	0.0183	Reject
Year	1.004	0.4265	2.353	0.0196	Reject
Realgdp	-0.023	0.0056	-4.072	0.0001	Reject
Realcons	0.018	0.0065	2.760	0.0064	Reject
Realinvs	0.022	0.0053	4.189	0.0000	Reject
Realgovt	0.020	0.0059	3.321	0.0011	Reject
Realdpi	-0.006	0.0041	-1.491	0.1376	Fail to Reject
CPI_U	0.040	0.0172	2.301	0.0225	Reject
M1	0.002	0.0054	0.456	0.6488	Fail to Reject
Tbilrate	84.176	62.7045	1.342	0.1811	Fail to Reject
Unemp	-0.328	0.3383	-0.970	0.3333	Fail to Reject
Pop	-0.103	0.1444	-0.716	0.4751	Fail to Reject
Infl	-83.903	62.6919	-1.338	0.1824	Fail to Reject
Realint	-84.650	62.7006	-1.350	0.1786	Fail to Reject

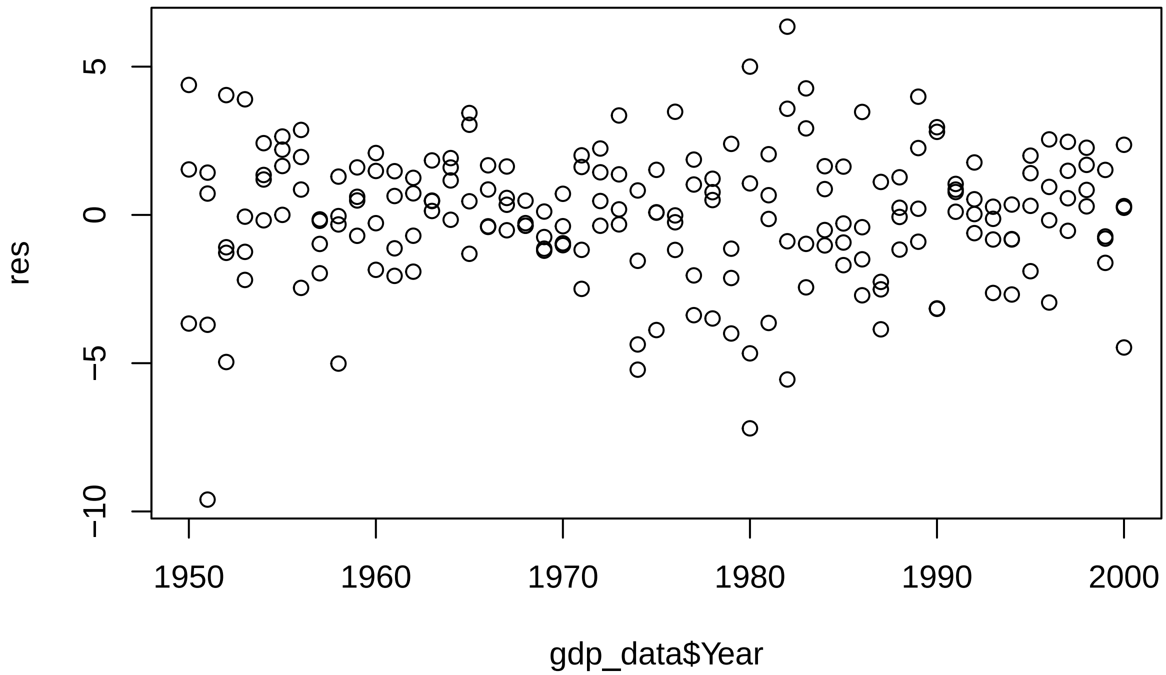
```r
model_1$mof_table
```

R^2_{uc}	R^2	R^2_{adj}	SIC	AIC	SSR	s^2
0.386	0.386	0.3438	1.954	1.725	992.7	5.252

(c) Plot residuals against time

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

plot(gdp_data$Year, res)
```



Looking at this plot, we likely have a POSITVE 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

```
# BGX_data <- select(gdp_data, c("Year", "Realgdp", "Realcons", "Realinvs", "Realgovt", "Realdpi"

# 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 <- 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
```

```r
    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
  R2_stat <- ols(BG_df, "e0", colnames(X0))$R2
  test_stat <- R2_stat*nrow(data)

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

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

}

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

Test Statistic	P-Value
0.3518258	0.5530814

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

We cannot reject the null that there is no serial correlation (that is, we might have a problem with serial correlation!)

(e) Box-Pierce

Use Box-Pierce test to test for first order correlation. Report test statistic and pvalue.

```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"
```

```r
  # 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
  lag_coef <- ols(BP_df, "e0", colnames(X0))$b[2]
  test_stat <- nrow(BP_df) * lag_coef^2

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

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

  }

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

Test Statistic	P-Value
0.3676114	0.5443092
_(I'm not sure th	is is correct. . . . .)_
Again, we fail to	reject the null- we likely have an issue with serial correlation!!

(f) Durbin Watson

Use the Durbin Watson test to test for first order autocorrelation. Report test statistic and interpret.