Empirical exercise 3 – Part in Stata

1. Open the data file

Clear the system meory and load the data file using the commands at the end of the section.

```
clear all
use "C:\Users\username\Desktop\exercisethree.dta"
```

2. Rename variables

Rename the variables for ease of reference using the command at the end of the section.

```
rename q quarter
rename YD income
rename CE consumption
```

3. Drop observations

Using the command `tab year` observe that the data runs from 1947:1 to 1996:4. `tab` is shorthand for the `tabulate` command. Since we are asked to run the regression for the period 1953:1 to 1996:4, drop the observations before year 1953 using the command `drop if year < 1953`.

```
tab year
drop if year < 1953
```

4. Generate a time indicator

In the data, variables `year` and `quarter` together define each quarter of a year. Generate a time indicator that indicates each quarter of a year so that we can use it throughout the exercise. We can use the command `gen t = _n` to generate the time indicator. `gen` is shorthand for the `generate` command. `_n` is called the underscore variable. It is a built-in syntax. `gen t = _n` assigns each observation the row number as they are stored in the data. Because we want advancing observations of `t` to correspond to advancing calendar dates, we need to make sure that the values of `year` and `quarter` are sorted in an ascending order before we generate `t`. Use the command `sort year quarter` to sort the observations of `year` in an ascending order, and then to sort the observations of `quarter` in an ascending order within the sorted observations of `year`. Type `order year quarter t` in the command prompt to group the three time indicators in your data sheet so that you can easily inspect them using the data browser.

```
sort year quarter
gen t = _n
order year quarter t
```

5. Carry out regression and analyse the residuals against time

Carry out a regression where the outcome variable is consumption and the predictor variable is income, using the command `reg consumption income`. The coefficient estimate of `income` gives you an estimate of `the marginal propensity to consume` based on the underlying sample data. The command syntax for obtaining the residuals is `predict e_hat, residuals` where `e_hat` is a custom name. Once the command is executed a new variable will appear in the Variables window.

Produce a scatter plot of `e_hat` against `t`, and name the scatter plot as `scatter_ehat_t` using the command `scatter e_hat t, name(scatter_ehat_t,replace)`. We name the plot as otherwise it will be overwritten when a next plot is produced which does not allow us to inspect different plots at a time. We add the `replace` option because otherwise we cannot display the graph for a second time as it already exists in the system memory with the name given to it.

Inspect in the scatter plot that the estimated errors are serially correlated especially at advancing values of the time indicator. Furthermore, the estimated errors do not depict a constant spread around the mean across the values of the time indicator suggesting that the errors are heteroskedastic. These show that the variance-covariance matrix of the residuals contain structure that is not effectively controlled by the explanatory variables we currently employ.

Inspect also the histogram of the residuals using the `histogram` command. The histogram shows that the residuals are not really normally distributed, and in particular the distribution is skewed to the left in favour of the values less than zero. This is in line with the conclusions we have drawn from the scatter plot.

```
reg consumption income
predict e_hat, residuals
scatter e_hat t, name(scatter_ehat_t,replace)
histogram e_hat, name(hist_ehat,replace)
```

6. An attempt to curb heteroskedasticity

Heteroskedasticity may have different sources. It could be that outliers are causing heteroskedasticity. However simple descriptive inspections of the variables do not favour this reason. Another reason could be that as income increases, people have more freedom about how they consume and consumption may become more volatile with increasing income.

It is possible to restrain heteroskedasticity to a certain extent by transforming the variables into logarithmic forms. Think of the shape of the logarithmic function: across the larger values of a variable, the transformation brings these values closer to each other. This explains why the logarithmic transformation helps to limit heteroskedasticity. Convince yourself that this is true. Consider the commands `gen income_ln = ln(income)` and `gen consumption_ln = ln(consumption)` to transform the variables into logarithmic forms.

Consider the regression `reg consumption_ln income_ln`. Since both the outcome and the predictor variables are in logarithmic forms, the regression is now evaluating proportionate changes in both variables. Convince yourself that this is true. The interpretation of the coefficient of `income_ln` has therefore changed. Proportionate changes are usually more stable throughout time. Obtain the residuals using the command `predict e_hat_ln, resid`.

Inspect the scatter plot of the residuals against time using the `scatter` command presented at the end of the section. Observe that heteroskedasticity is limited to some extent, but the economic shocks in the later periods still renders the data heteroskedastic.

```
gen income_ln = ln(income)
gen consumption_ln = ln(consumption)
reg consumption_ln income_ln
predict e_hat_ln, resid
scatter e_hat_ln t, name(scatter_e_hat_ln_t,replace)
hist e_hat_ln, name(hist_e_hat_ln,replace)
```

7. Regression with simulated consumption data

We want to draw random errors from the normal distribution. `gen variable = rnormal(m,s)` instructs Stata to draw random numbers from the normal distribution with mean `m` and standard deviation `s`. Let us draw random numbers with mean zero, and hence set `m` to zero.

We are required that the variance of the random numbers we draw is equal to the variance of the errors of the regression `reg consumption income`. How do we obtain the variance of the errors of a regression? Once again, carry out the regression `reg consumption income`. In the regression output, consider the columns SS, df and MS. SS stands for sum of squares. Residual sum of squares is given by $\sum(y_i - \hat{y}_i)^2$ which is a measure of the spread of $\hat{u}_i$ from its mean $\bar{\hat{u}}$. df stands for degrees of freedom. The residual df is N-2 which is the number of observations minus the number of model parameters including the intercept. MS stands for mean square and it represents the ratio of the SS column divided by the df column. Given these, the Residual MS is the estimate of the variance of the errors. Square root of it is the Root MSE, standing for root mean squared error, also presented in the regression output. It is the standard error of the regression and it is a measure of the spread of the observations around the regression line. It is a measure of fit: the less the observations are scattered around the regression line, the better the fit is. Consider the command `ereturn list` which shows the values of the various parameters and statistics calculated in the most recent regression. `e(rmse)` is what we are looking for. It is the root mean squared error, which is the estimate of the standard deviation of the errors. Type `help return` in the command prompt to study how parameters are stored in the system memory after a regression is carried out.

We we can replace `s` with `e(rmse)` in the command syntax `gen e_new = rnormal(0,s)` since `s` represents the standard deviation. However, it turns out that calling a parameter within a built-in command in this fashion is demanding a lot of system memory. Therefore, we will assign the parameter to a 'local macro'. Consider the command `local rmse = e(rmse)` where `rmse` is an arbitrary name. A macro is an input argument that is string in nature consisting of characters. In Stata macros can be local or global. For our purpose, they make no difference. Local macro names can be up to 31 characters long.

We can now generate the new error terms using the command `gen e_new = rnormal(0, `rmse')` where `e_new` is a name we give to the new error terms.

Using the new error terms, generate the new consumption data using the estimated parameters from the regression `reg consumption income`. For this we can use the command `gen consumption_sim = _b[_cons] + _b[income] * income + e_new`. The syntax `_b[income]` calls the coefficient estimate of `income` from the last regression we have carried out.

Next, use the simulated consumption data to carry out a regression of consumption on income using the command `reg consumption_sim income`. Note in the regression output

that the parameter estimates are very similar to those obtained when real data is used. The small differences are due to the random draws of the errors.

Inspect the scatter plot of the new residuals against the time indicator `t`. Inspect also the new residuals themselves using a histogram which you can obtain by executing the command `hist e_hat_sim`. The new residuals appear to be more independently and identically distributed. If the original errors looked like these residuals, we could claim that the model was correctly specified.

```
reg consumption income
ereturn list, all
local rmse = e(rmse)
display `rmse'
gen e_new = rnormal(0,`rmse')
gen consumption_sim = _b[_cons] + _b[income] * income + e_new
reg consumption_sim income
predict e_hat_sim, resid
scatter e_hat_sim t, name(scatter_e_hat_sim,replace)
hist e_hat_sim, name(hist_e_hat_sim,replace)
```

8. Exploiting the time series structure of the data

The analyses of the residuals and the attempts to curb heteroskedasticity in the preceding sections did not really help. Our conclusion could be that the model is not correctly specified. In fact, note that so far we did not exploit the time series dimension of the data. Doing so is beyond the scope of this exercise but we can still consider a slightly different yet simple model that accounts for the structure in the variables so far left to the error term.

Consider the time series graphs of consumption and income using the commands `twoway (line consumption t)` and `twoway (line income t)`. The graphs suggest that both time series are non-stationary. This means that in the linear regression model considered so far, we have a non-stationary series as a dependent variable, a non-stationary series as an independent variable, but have an error term that we assume is stationary. This shows that our specification has a problem. Let us turn the non-stationary consumption and income data series to stationary series. To do this, first set `t` as a time series indicator using the command `tsset t`. This is needed to use the difference operator explained next. Obtain the stationary consumption and income series using the commands `gen consumption_dif = d.consumption` and `gen income_dif = d.income` where `d.` is the built-in difference operator.

Carry out the regression using the stationary consumption and income series. Inspect the residuals. The scatter plot suggests that the model in first differences is a more reasonable model. The histogram of the residuals `e_hat_dif` looks more like a normal distribution compared to the histogram of the residuals `e_hat`.

```
twoway (line consumption t), name(line_consumption_t,replace)
twoway (line income t), name(line_income_t,replace)
tsset t
gen consumption_dif = d.consumption
gen income_dif = d.income
reg consumption_dif income_dif
predict e_hat_dif, resid
```

```
scatter e_hat_dif year, name(scatter_e_hat_dif_t,replace)
hist e_hat_dif, name(hist_e_hat_dif,replace)
```