Part 0 – Open Stata, and make your own do-file

- Using windows explorer, make a new folder called `H:\rproject\clab\clab12.`
- Start Stata through the start menu button
- In de white command window type `doedit` to start de do-file editor. Place the Stata screen on the left and the do file editor on the right such that you can easily switch between the two.
- In the first two lines of the do-file type

```
cls                                 //this clears the screen
clear all                           //this clears the memory
cd "H:\rproject\clab\clab12"        //this is your path
```

Part 1 – Create a working dataset using `crwork.do`

In part 1 of these exercises you are going to create a working dataset from excel sheets that have been downloaded from `https://stats.oecd.org/`. The excel datafiles contain information on gross domestic product (gdp), consumer prices (cpi), (short term) interest rates (i), and money supply (m), for the OECD and some additional countries. The Stata file `work_merged.dta` already contains measures of `gdp`, `cpi`, and `i`, and needs to be merged (linked) to `m` such that we can investigate the relationship between prices and money supply.

1. Download the data files `M3.xlsx` and `work_merged.dta` from Bb, and store them in your folder for the current computer lab session (e.g. `H:\rproject\clab\clab12`).
2. Import the excel data on money supply using the menus File->Import->Excel. Think about the options to use. In particular, are the column headers visible in row 1 or lower? Row 7 perhaps? (Don't forget to apply Rules 1&2: paste the command in your `crwork.do` file, press control-s to save and control-d to run).
3. Type `browse` to familiarize yourself with the data. Just look around. Notable things we see are i) the `country` variable is empty after row 22; ii) all variable names (except `country`) are letters of the alphabet; and iii) the contents of columns `d - ad` are in red; iv) in column `ar` all values are 100.
4. Open the datafile also with excel, to compare it to how Stata imported it. Can you give an explanation for the variable names? And for the red contents? Why are all values in column `ar` equal to 100?
5. We can create a single country variable by typing (copy-pasting)

```
replace b = country if mi(b)  //replaces b missing
ren (country b) (a country)   //renames
drop a c                      //drops redundant variables
```

6. We need to give the variables proper names. From the excel file we know that column `d` is M3 (money supply) from 1970, while column `ay` is M3 from 2017. Using the command `rename (d - ay) m#, addnumber(1970)`, we can make easy names out of this.

7. Then, we need to make all variables numeric. Type `destring m*, force replace` to achieve this. Use `browse` to inspect the data again. Notice a difference? What do you think the `force` and replace `options` are for?
8. Look at column `m2010`. The final observation is missing because it does not correspond to a country. We can drop it by typing `drop if mi(m2010)`.
9. Type `tab country` to look at all our countries. It seems that the countries are not sorted by name. Also, there are some long names. By typing
   ```
   replace country = strtrim(country)
   replace country = word(country,1) if wordcount(country) > 2
   ```
   we get rid of spaces before the names, and shorten long country names. Type `help string functions` to see more functions.
10. Now we are nearly ready to merge the data on money supply `m` to the other dataset that contains prices and GDP. We will do this using the `merge` command, which is tremendously useful. You can check it out by typing `help merge`. Then click on Also see->pdf to see some good examples. The command to use is `merge 1:1 country using work_merged`
11. After a merge, the variable _merge contains information about whether observations (countries) could be matched. Type
    ```
    tab country if _merge==1
    tab country if _merge==2
    drop if _merge==1
    drop _merge
    ```
    The tables show that Brazil, OECD, and Turkey are only present in the *master data*, which is the data currently active (the money supply data), while a bunch of European countries are only present in the *using data* that contains GPD and prices. We will drop the former, but keep the latter. Why do you think there is no information on money supply in the European countries?
12. You now have a clean dataset on gross domestic product (gdp), consumer prices (cpi), (short term) interest rates (i), and money supply (m) for OECD countries from 1970 to 2017. You data are in *wide form*, meaning each row in the data is one observation (country). Type `browse` to see the data. To be able to analyze the data, and do transformations on it, we need to convert it to *long form*. This is done using the reshape command. Type `help reshape` to figure out what it does. Then type
    ```
    reshape long gdp cpi i m, i(country) j(year)
    ```
    to restructure your data. Use `browse` to see the result.

You now have a well-structured dataset that can be analyzed. Usually your model or theory will point to what variable you need in the analyses. For example, Fig 5.1 of the macro book of Mankiw and Taylor (2014) shows a scatterplot of the (log of) inflation and money growth. The quantity theory of money implies both should be strongly positively related. If you want to investigate this with the current dataset, you need to create the growth rates of prices (inflation) and money.

13. You can create growth rates by typing
    ```
    bysort country (year): gen pi =100*(cpi-cpi[_n-1])/cpi[_n-1]
    bysort country (year): gen g  =100*(gdp-gdp[_n-1])/gdp[_n-1]
    ```

```
bysort country (year): gen mgrowth =100*(m-m[_n-1])/m[_n-1]
```

The `bysort` statement makes Stata generate the variable *per country, sorted by year*. The `_n-1` refers to the previous observation. Because the data are in longform we need only do this for three variables. In wide form, we would have had to do it for each variable for each year.

14. With the growth rates available, it it easy to generate the logarithm of it:
```
gen lnpi       = log(pi)
gen lng        = log(g)
gen lnmgrowth  = log(mgrowth)
```
15. The final and essential step is to save your clean and structured dataset in a file for further analyses:
```
compress
save work.dta, replace
```
The `compress` statement makes sure the data are stored in the most efficient way. The `replace` option allows you to make changes in the do-file and run it again. Your workfile `work.dta` will then be overwritten. The raw data never change.

Part 2 – Making using `graphs.do`

In part 2 of these exercises we will make figures similar to fig. 5.1 and 5.2 from Mankiw and Taylor (2014).

16. Start by making a new do-file by the name of `graphs.do`. Start with the statements from Part 0, and then `use work.dta` to open the structured dataset that you have just created in Part 1.
17. To mimic figure 5.1, type
```
scatter lnpi lnmgrowth, xtitle(Money supply growth (log))
  ytitle(Inflation (log))
```
18. To mimic figure 5.2, type
```
line pi i year if country=="United Kingdom", lc(green red)
  ytitle(Per cent) xtitle(Year) legend(pos(0) bplace(ne) col(1)
  order(1 "Inflation rate" 2 "Nominal interest rate"))
```
19. To save the graph type `graph export uk_pi_i.emf, replace`. Put it in a word document to see what it looks like.
20. Repeat question 18 (press page-up to get it back) but first type `set scheme economist` to get another nice layout. Try `set scheme sj` for another.

In the next tutorials we will practice the final, and most interesting, script where you do your main analysis: `analysis.do`. Also, we will make more fine-tuned graphs as we go along.

References

Mankiw, N.G., and Taylor, M.P., 2014. Macroeconomics, 2nd European edition. New York: Worth.