ocp_neer

Competitiveness indices are important indicators that tells how a country or an industry is performing with respect to its partners and peers, for the phosphate industry, a key sector for the Moroccan economy, having such a picture on the evolution and the positioning of its power is a must, in particular given the current setup of the industry, the increase of partners and the price movements along with exchange rates fluctuations.
By considering the latter possible causes, that is prices, exchange rates, exports and imports data of all countries in the industry, one can derive with the help of the effective exchange rate double weight based indicator useful information on the international competitiveness of the Moroccan phosphate industry.

the way to go is following the agreed on methodology to construct double weighted effective exchange rates (see Schmit & al. 2013 ECB), that consists in combining data on bilateral trade flows for the product or sector in question, data on production and consumption, data on exchange rates, and deflators for constructing the real version, all this merged into one dataset in order to generate the needed variables and indicators.
This exercise is data intensive, and a bit complex given the difference between the dimensionalities of the datasets (bilateral trade data is 4 dimensional, while all other are three dimensional).

The purpose of this Stata program is to automate all the processing tasks through the **project** built in command developed by Robert Picard, by creating a master do file that controls all sub do files that are responsible for each step of the indicator computation, which results in inputting the necessary raw datasets, sending the command to the master do file to be run, and then waiting for the results to be outputted in the desired folder, without caring about the intermediate data steps that are between. this is useful for managers that do not care about the different data processing steps, and also for analysts, that would like fast and manageable programs to derive the desired results, without having to care about the path dependencies and running each do file apart.

The OCP_EER Stata program is composed of the following folders :

- **data_raw** : contains raw excel and cvs files of the data as downloaded from their sources, no changes
- **data_derived** : contains all the intermediate data created through the whole process
- **results** : contains indicators in excel sheets and analytical graphs
- **Codes** : contains the do files for each step of the data processing.

On top of that the OCP_EER folder contains a **master** do file, this is the controlling point of the whole program,
We provide further details on the elements of each folder, the functioning of each do file and finally notes and recommendations on the associations between all three.

1. Folders decomposition :

The folders are decomposed in the following logic, each folder contains a subfolder that is specific to the dataset in question, if different datasets are concerned the datasets are stored in the merge folder,

1.1. the **data raw** folder is composed as follow :

COMTRADE : containing raw csv files of bilateral phosphate industry trade data, downloaded from uncomtrade website, the folder contains 4 files divided by years, each csv file contain a set of years, that each one of those will be imported and merged with the others, duplicates are then controlled for to avoid any extra information, if the user would like to add a year that is not present or to modify an existing one due to revision of trade data (that is common) by the website, the new added file should have the same structure as the previous files, so as to be correctly identified by the comrade.do program, to ensure this the download should consist of the following query :

- **periods** : select the years you would like (up yo 5 are allowed)
- **Exporters** : all
- **Importers** : all
- **trade flows** : exports
- **products** : 310530 310540 280920 2510 3103

then hit the download csv button, download the data and move it to the folder data_raw/comtrade.

Remark : open the dataset and check that is full, sometimes the UnComtrade API may crash and send complex queries, if the fills is empty, then do not put it in the data_raw/comtrade folder, since it will crash the program and stop it from running, instead retry the query and try downloading again from a different ip address (another laptop or another web browser).

IFA : contains excel files of production and consumption data from the international fertilizers association website, same as Comtrade, data are sliced by years due to limitation by the website to download bulk of data, moreover data should be downloaded as follow from https://www.ifastat.org/databases/supply-trade :

- **Country** : all countries
- **when** : select the desired period, up to 5 years are allowed
- **which products** : select only phosphate : DAP MAP TSP PA Phosphate rock all grade
- **which activities and units** : production exports imports apparent combustion home deliveries/ units product

and check hide empty lines, then hit show data and then download, again check that the files are not empty, and without changing them put them in data_raw/IFA
Another folder is prices exists within the subfolder IFA, it contains prices related to each phosphate byproduct, the files can be changed if the user would like to update prices.

Exchange : contains exchange rates data for each year from 2000 till 2020, downloaded from the imf stat website using the following query :
https://data.imf.org/regular.aspx?key=61545862

- select national currency per SDR, period average.

Deflators : the current version of the program relies on a single deflator which is the cpi, downloaded from https://data.imf.org/regular.aspx?key=61545861 by selecting consumer prices. newly added data are put in data_raw/deflators/cpi.

ISO : contains iso excel files for matching names of countries in each dataset in a common way to be merged, the logic of the file is to have a common names for all countries that will correspond to the country name in a specific dataset.

1.2. **data_derived** folder is decomposed as :

- **Comtrade**, **IFA**, **Exchange**, and **deflator** : contains as the dataset that where created by the respective do files,

- **Merge** : this folder contains common datasets that are intermediary between different steps of the data processing

- **NEER/REER** : contains respectively nominal effective exchange rates datasets and real effective exchange rates ones, it includes the more general out of computation data including all the variables that were used in the computation of the indices, along with testing ones. each folder is decomposed as follow :

  * A variation and an index folder, the former include the variation of the indices (nominal and real effective exchange rates), while the index is including the date based nominal/real effective exchange rate index.

  * the above folders are further decomposed into a geometric and an arithmetic method of computation, the former follows the conventional way of creating effective exchange rates, while the arithmetic one is based on specific needs of the project (further differences can be discussed).

  * Moreover, each one of the above folders are decomposed into either yearly rates or monthly ones.

  * Additional note : the need folder contains the double/simple weights as outputted from the weights function and with respect to each level of aggregation.

Last folder is **results**, it includes the output of the indices in excel format, variation decomposition graphs, and graphs for the reer/neer.

2. Codes decomposition :

The do file programs are composed as follow :

Comtrade : import, append, process and output ready to merge bilateral trade data from comrade

IFA : import, append, process and output ready to merge ifa country data

merge_trade_ifa : merge comrade and ifa datasets

exchange : import, append, process and output ready to merge exchange data

deflators : import, append, process and output ready to merge exchange data

weights : input the trade merged data, slice into chunks of year and product, and input into weights function to compute the double and simple weights (5 products)

weights_agg_1 : input the trade merged data, slice into chunks of year and product, and input into weights function to compute the double and simple weights (3 products)

weights_agg_2 : input the trade merged data, slice into chunks of year and product, and input into weights function to compute the double and simple weights (1 products)

neer_year : inputs the weights computed, merge with exchange year, process data, and compute the neer index and variation (geometric)

reer_year : inputs the weights computed, merge with exchange year, process data, and compute the reer index and variation (geometric)

neer_month : inputs the weights computed, expand them to months, merge with exchange month, process data, and compute the neer index and variation (geometric)

reer_month : inputs the weights computed, expand them to months, merge with exchange month, process data, and compute the reer index and variation (geometric)

neer_index_arithmetic_year : inputs the weights computed, merge with exchange year, process data, and compute the neer index and variation (arithmetic)

reer_index_arithmetic_year : inputs the weights computed, merge with exchange year, process data, and compute the reer index and variation (arithmetic)

neer_index_arithmetic_month

reer_index_arithmetic_month //

decomposition : input the computed effective exchange rates, and compute the variation decomposition by country, and outputs the results in graphs

output : output graphs and excel files for the nominal and real effective exchange rates, only for geometric

export_excel : output excel files

3. Guidelines for running the program :

As stated in the introduction, the program is managed by the project Stata built in command, the purpose of the latter is to automate the execution of all do files, provide a global path for any user and avoid executing unchanged programs.

After downloading the ocp_neer folder, put it wherever you want, in this folder you will have everything from the raw datasets used until the results, and each one in its specific folder.

First thing is to open the Stata program, install the project command as :

ssc install project

once done, than you tell Stata the master do file that is going to manage the project such as:

project, setup

then a pop up window will open telling you to choose the master do file, choose the one in the OCP_neer folder.

Now, before running the program there is few things to make in mind, first as you will see the master do file contains a lot of project do commands, as it states this commands will run each do file after one, in the order specified, changing the order is never recommended given the dependencies between each do file and another, by dependencies, I mean that each do file have associated to him a set of input files that he uses and output files that he creates, so each time an input file is changed by the previous do file, the project command will force the using do file of this input file to be run, and hence creates a chain of changes until the last do file. The other thing to mention, is that it is possible to comment out some do file if the user have no intention to use, such as arithmetic vs geometric neer computation do files, or weights aggregation do files or for any simulation reason.

If the user want to launch the program for the first time then type in :

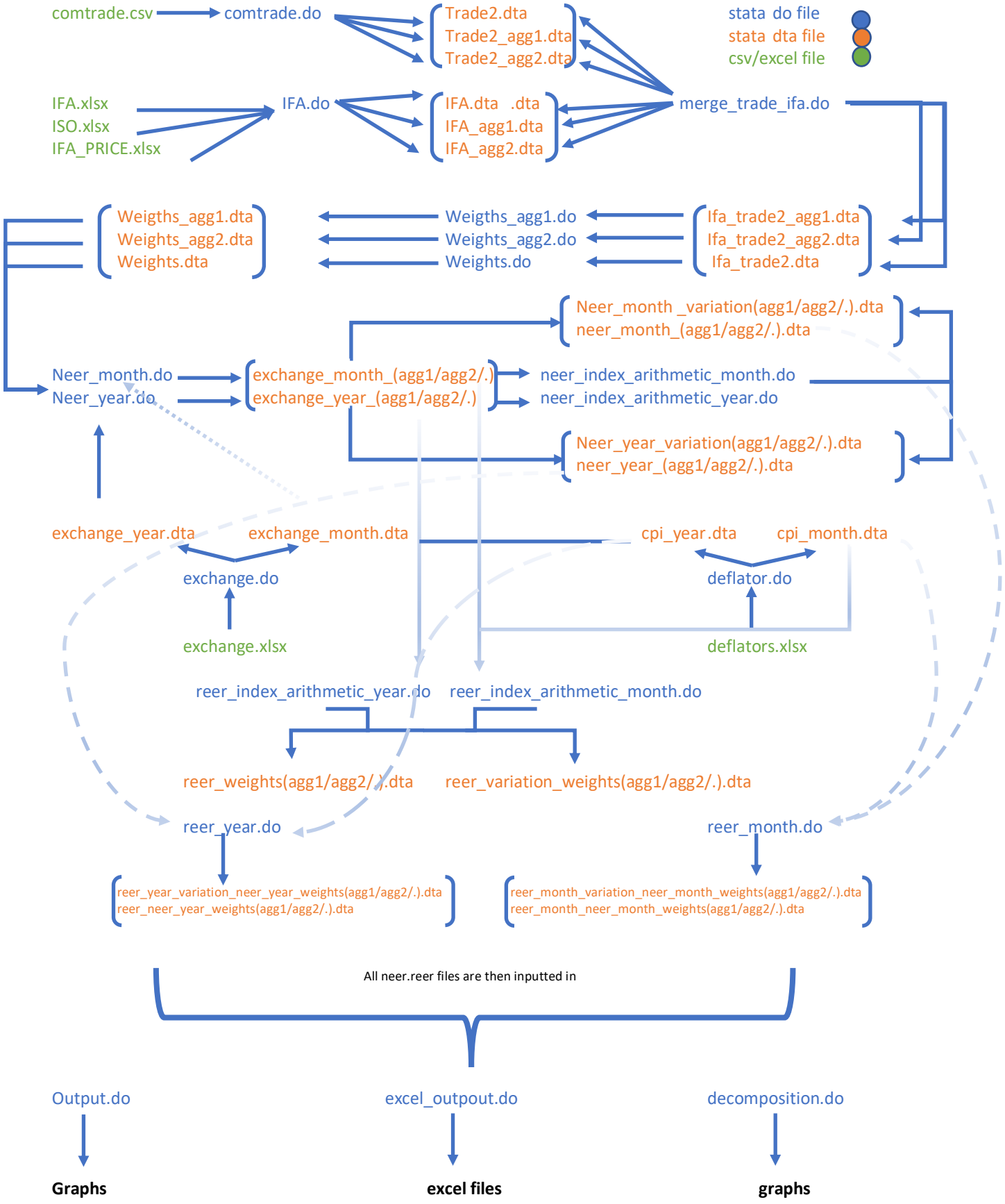
Project master, build

This will launch the program and it takes around 20 minutes to finish.

Now few things can happen, first is the program can crash if it does, send me an email with the error, the do file it crashed in at abselakari@gmail.com.

Once the program is finished, you can check your results at the results folder, another thing that is useful, is the smcl files, these files are the shell output of each do file, ie, each time a do file is run, what is displayed in the stata results window, will be saved into these smcl files. An example of usage, is checking the countries that have been dropped in each merge.

Remark regarding adding new files :
The user may would like to add new files, for example a new year that is available or replacing an existing one due to revision by the source data, to do so, download the desired year, check that is having the same query as above, and put it in the associated folder, once done, and if the program has been already run, I would suggest to rerun the program from scratch, to do so, project, setup/ and then project master, build.



OCP_NEER program data workflow from input to output.