Bussiere Methodology Script

# Introduction

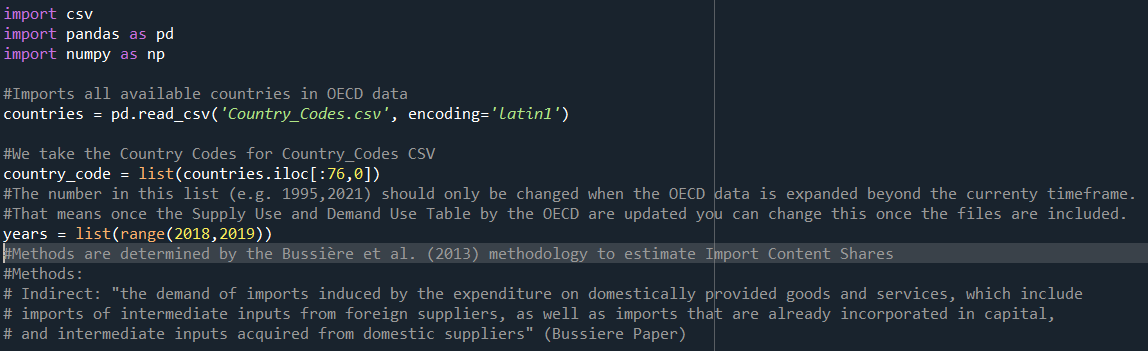
The **Bussiere Methdology Script** implements the methodology described in [Bussière et al., 2013](https://www.aeaweb.org/articles?id=10.1257/mac.5.3.118) (from here on referred to as the Bussière Paper or Methodology) to estimate the import content share of the major macroeconomic models (e.g. Private Consumption, Government Consumption, Investment, and Exports). The methodology was originally applied to the OECD’s [domestic output and import tables](https://www.oecd.org/en/data/datasets/input-output-tables.html#:~:text=The%20latest%20set%20of%20harmonised%20national%20IOTs%20presents,and%20several%20non-member%20economies%20%28including%20all%20G20%20countries%29.). The first script described below resulted in a strong replication of the paper’s results. However, given the time-series limitation data provided by the OECD, an additional combination of scripts will described. Specifically, a script which transforms Figaro Data into a replica of the OECD data format, and a modification of the Bussière methodology script.

Notably, the results are not identical and vary from country to country. This may arise from methodological differences (unlikely), currency differences (OECD in dollars while Figaro in Euros), or further reason which was not identified. Given the time and skill limitations, and inability to reach out to experts, the script is provided with risk of unreliability placed on the users.

# OECD Script

## Loading Data

The data is loaded by first loading all the countries available within the OECD datasets. These are stored in the variable **country\_code** and later used to loop over countries. Similarly, we set range of years within which we would like to load the data. Notably, this is constrained by data availability and will need to be determined by the user’s data stored within their working directory (see Figure 1).



## Selecting Type of Variable to be Generated

Given that the Bussière Methodology allows for three types of import content estimates, we need to preselect which we would like the script to ultimately provide. These are **Indirect, Direct, and Total.** These can be set manually in the python list called methods. Based on which methods you are interested in estimating, fill in one or all of the method names in “”, as well as putting a comma in between. It is important to note that the code is case sensitive. This requires us to ensure that all letter are small.

methods = ["indirect","direct","total"]

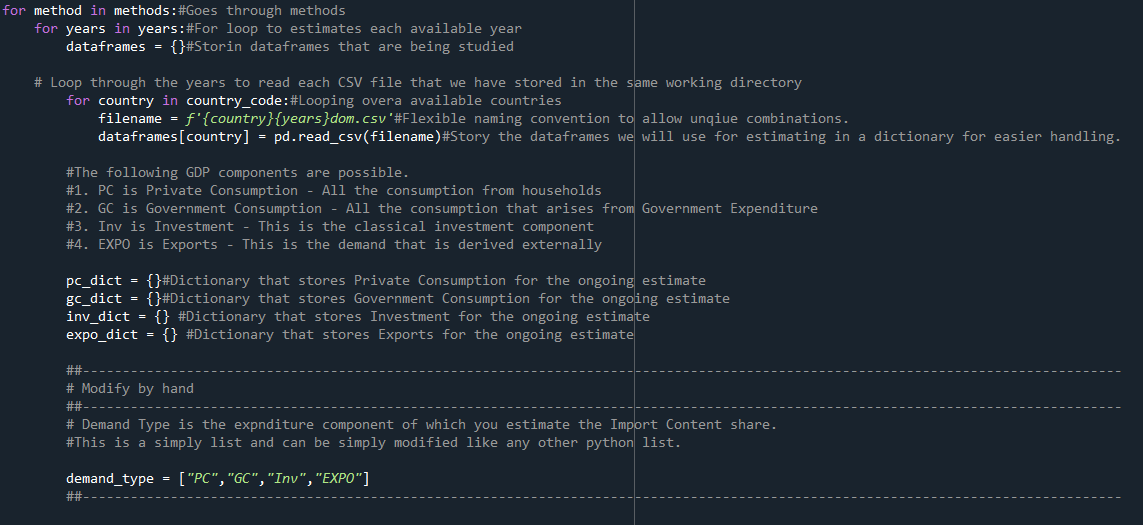
## Estimating the Import Content

The script is a simple nested for loop. The most “outside” loop is the method loop. This is because this because results will be stored together by method. Similarly, as can be seen in Figure x, the next loop goes over the range of years you set. For each year-method combination a unique element within a dictionary is generated to ensure comparability. This dictionary is called **dataframes** because it stored the dataframes we generate using the loop.

Then, the data is loaded with which unique combination of country and year is loaded based on estimation needs. This occurs because the OECD provides its data in single country-year combinations, requiring us to load the data based on our needs.

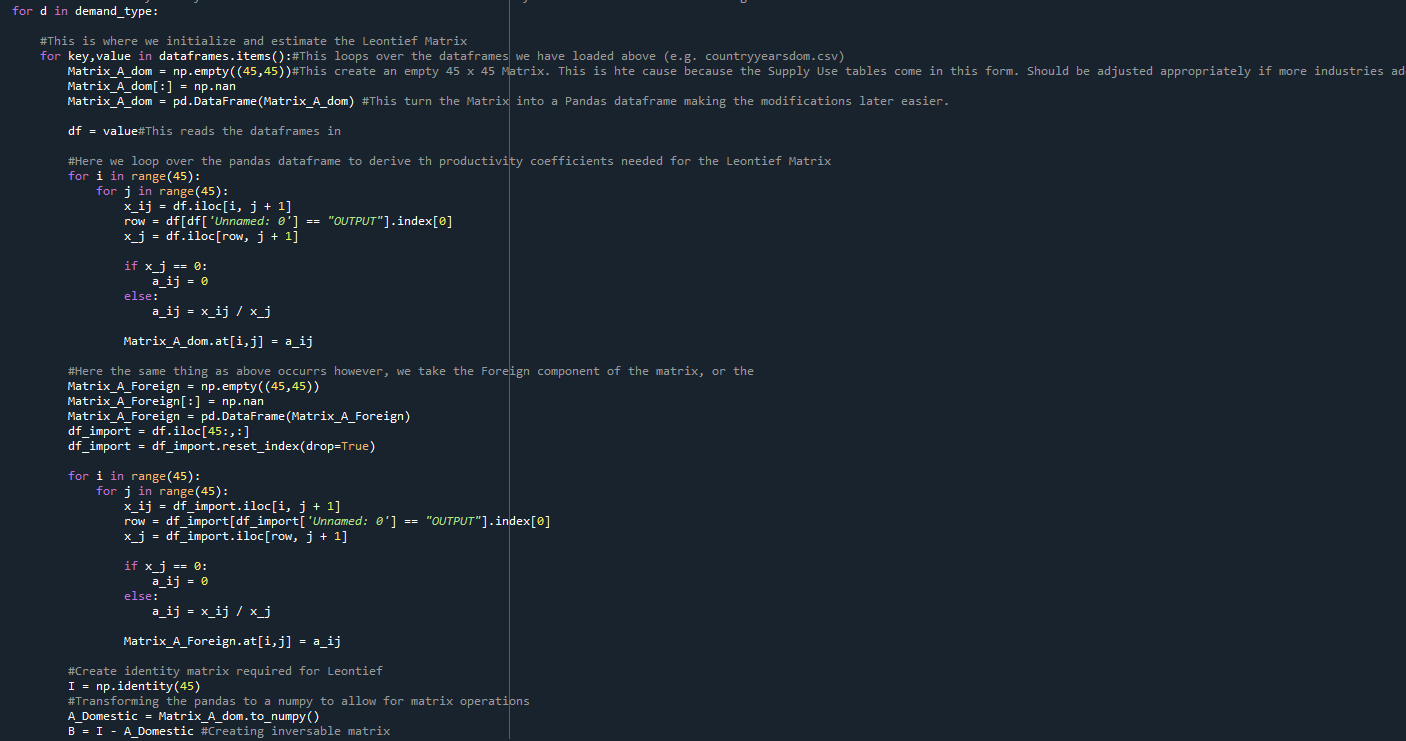
### GDP Components

The script will estimate the import content of four types of demand components: (1) Private Household consumption (composed both of households and non-profit entities), (2) Government Consumption, (3) Investment, and (4) Exports. Which ones you are interested in to estimating need to be once again edited manually. Specifically, a python list with the different consumption types, similar to that of methods is available and can be edited freely. Again, it is important to highlight the case sensitivity of this list, as it may cause problems in the storage of the estimated values later on.



### Estimating Demand Components

Given the redundancy in explaining the price methodology (given that it is provided in the Bussière paper), I will highlight certain details that might be important. First, the loop moves along the country-year combinations we stored. This is a relatively simplistic structure, however, functions overall well. If errors occur at this stage there might be issues with the data itself (accessing/corrupted data files). The 45 in the np.empty operation is important. This is because the structure of the data takes a 45-45 symmetric matrix structure (e.g. 45 industries horizontally, and 45 industries vertically). This can be adjusted, given that similar datasets could be employed using this code. In that case (assuming that the structure is similar enough, simply changing this number should hypothetically suffice).[[1]](#footnote-1)

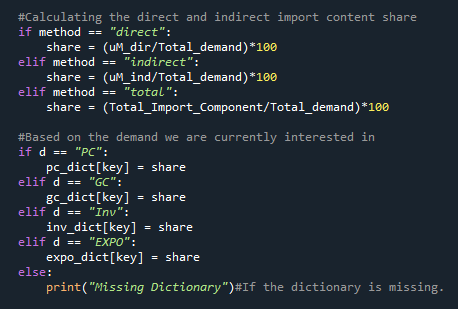


The Bussière methodology requires us to estimate two types of Matrices (after which we apply the standard Leontief equation treatments). What is important to note in coding here are the following. Thee reason why we see df\_import= df.iloc[45:,:] is because the OECD data is structured in such a way that it reports the domestic imports for each industries in the rows 45 and on.

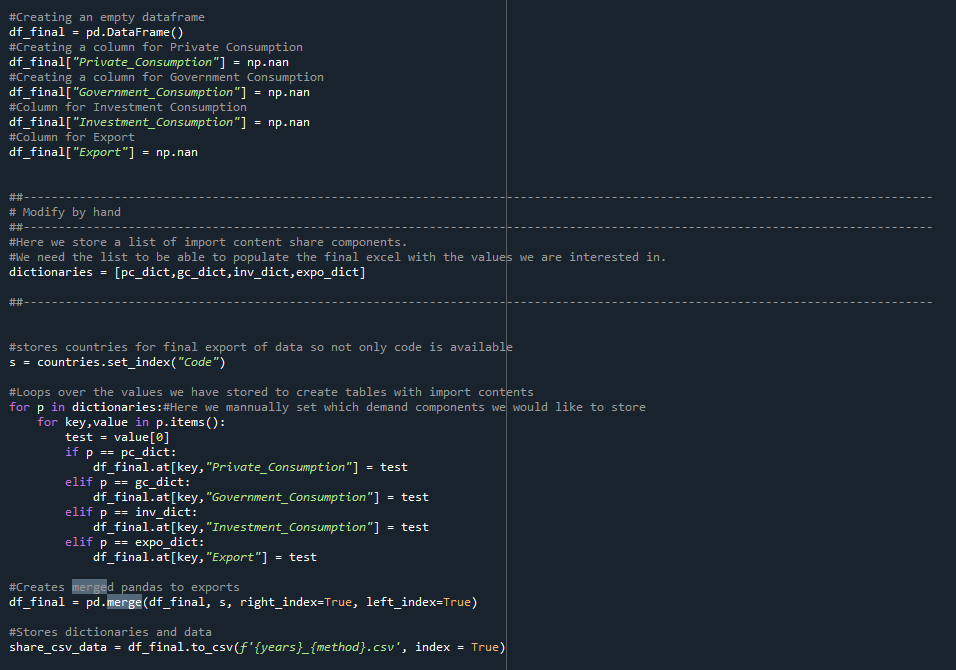
Furthermore, it is also important to elaborate on the naming convention of “Unnamed:0”. This is primarily the case as python automatically fills unnamed column rows with Unnamed: x. Assuming that this applies also to your data this should not be of concern. In the case this does not apply, bugs related to this line can be easily solved likely by replacing “Unnamed: 0” with the naming of the column available within the dataframe you load. The reason we are calling this row is because of the Bussière methodology.

### Storing Data

Once we have calculated the matrices required for estimating the different types of import content demand sources, we stores these into the demand type dictionaries. Therefore, if you have not selected a certain type dictionary, they will simply be empty.



The final outcomes are then loaded into an empty pandas dataframe. This is populated with empty columns for the different sources of demand. The dictionaries with the individual demand types are loaded into a list of dictionaries, which is looped over in the process of population the pandas dataframe. This allows us to store the different methods, and demand sources into a single dataframe for all available countries. The different countries. Finally, we merge the data we have generated, and dataframe of long form names of countries. This allows us to read the actual countries without having to google them. Finally, the import content data is stored as a csv file. Notably, all individual years are uploaded as individual dataframes. Therefore, you will the for year x all the countries where data is available, as well as the different demand source components, and import content methods (e.g. indirect, direct, and total).



# Bussière Methdology Using Figaro Tables

The same methodology can be applied to Figaro Tables (published by Eurostat). However, the use of the Bussière Methodology with Figaro Tables should be done with care. The reason is that while the structure is similar, the script takes certain assumptions and attempts to generate a similar domestic imports and exports structure as provided by the OECD. However, given that there were no papers or reports upon which one could confidently rely on building this structure, I was only able to “relatively” confidently rebuild the OECD like structures.

## Building OECD Domestic Import and Exports Tables

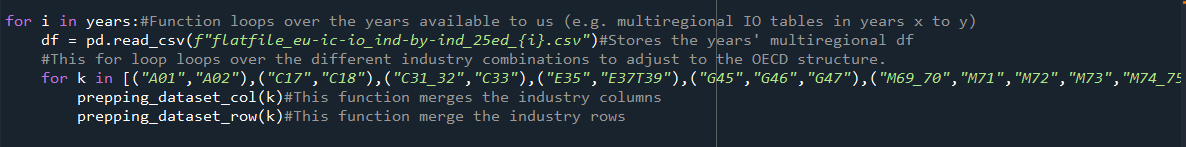
To begin with we discuss how to build the Domestic Imports and Exports Tables. The base data upon which we build are the Multi-Country Industry-Industry Input Output Tables. These tables provide both data on domestic production, consumption, imports, and exports. Notably, it does this for all EU-27 as well as relevant partners. Given Figaro’s close collaboration with the OECD, and various mentions of strong methodological interoperability with the OECD’s input-output tables, we more confidently argue that definitions such as Household or Government consumption will likely overlap. One notable difference between the Figaro and OECD data (an issue we confront below) is the aggregation of industries. While the Figaro tables’ industry disaggregation is relatively high, the OECD Domestic Import and Export tables combine various industries in the published.

The script goes through 5 phases when prepping the data:

1. Prepping Dataset Columns & Rows
2. Generating Domestic Columns
3. Generating Domestic Rows
4. Creating Final Domestic Import and Export Dataframe

### Prepping Dataset Columns & Rows

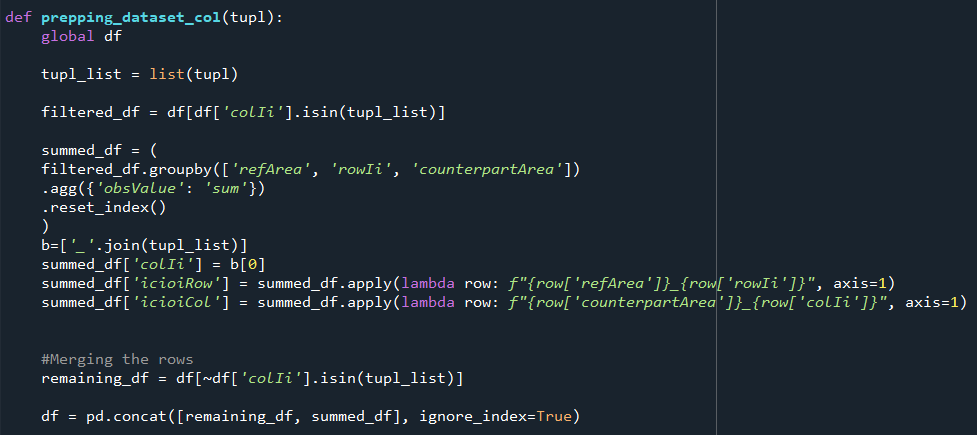
As mentioned above a notable difference between the Figaro and OECD data is the degree of industry disaggregation. Therefore, to construct a usable Domestic Imports and Exports table we need to merge said industries. This is done by looping over a python list of tuples storing the combination of industries (to liken that of the OECD). This is done by calling two types of functions. These are **prepping\_dataset\_col** and **prepping\_dataset\_row**.



#### Prepping Dataset Col

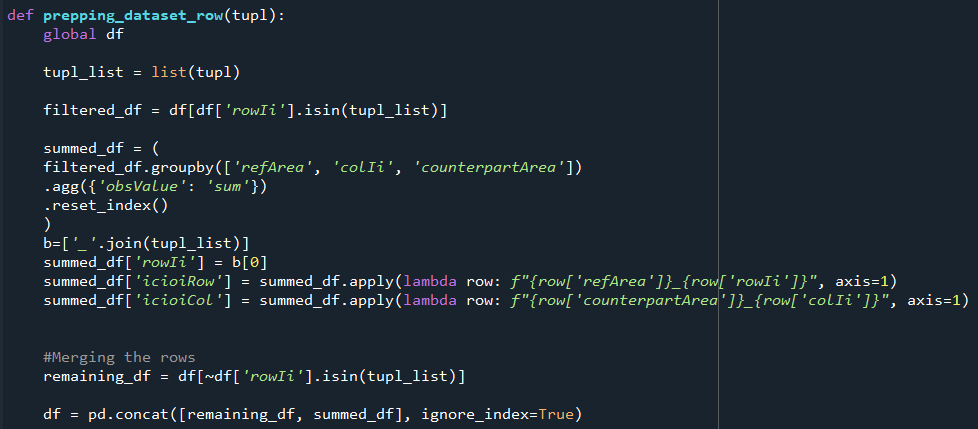
This function begins by loading the dataset stored as df in our working directory. Then it takes the tupls that we have previously (as shown above) as k. We then filter for these, which allows us to sum them together into a single row in our panel data. This is possible because we work with the Figaro Tables structured as panel datasets (allowing ease of operations).

We then combine the list of columns names into a single name, and then apply the new name, to each row afflicted by the merging operation. After summing up the values we take the remaining rows which did have not been merged, and combine them with merged columns.



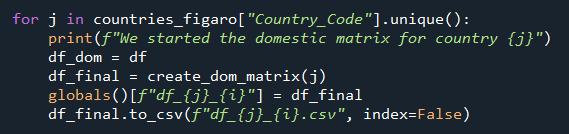
#### Prepping Dataset Rows

The same logic as before is applied to colIi to ensure consistency and that both rowIi and colIi are combined.



### Creating Dom Matrix

After prepping the original Figaro Table dataset, we begin a for loop over the unique countries. This is done by first storing the dataframe we have imported and modified as described above, after which we begin apply the modifications through the **create\_dom\_matrix**.



The **create\_dom\_matrix** function is outer function which calls different sub-functions to modify the dataset appropriately. This can be seen in Figure X. The first step in this function is call and applying **columns\_dom(country).**

#### columns\_dom(country) & rows\_dom(country)

In short, these two function allow us to differentiate between the country whose Domestic Imports and Exports Table we are interested in generating.

## Applying Bussière Methodology to Figaro Tables

The script for applying the Bussière Methdoology to the Figaro Tables is similar, and only contains minor changes. These will be described below.

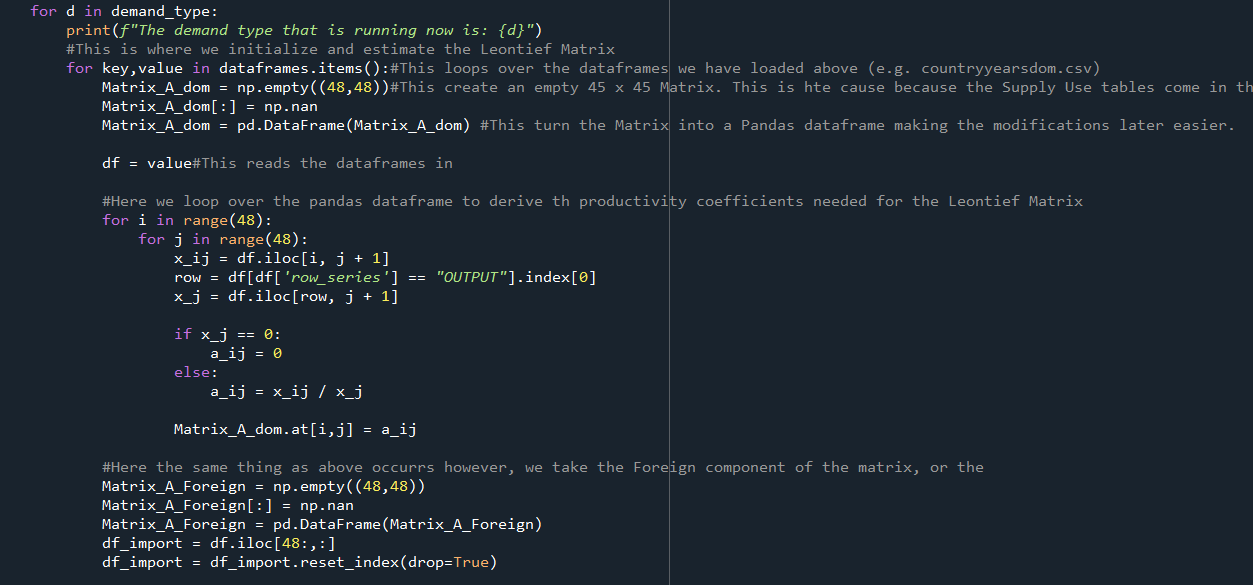
### Countries Available

Figaro tables include less countries overall, constraining the type of estimates we can perform. Therefore, we load a different set of countries using:

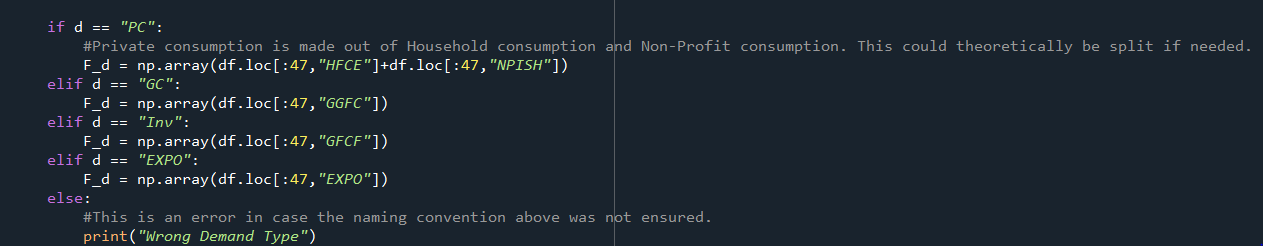
countries\_figaro = pd.read\_csv("Countries\_FIGARO.csv", encoding='latin1')

### Number of Industries

A further notable difference arises from the fact that we cannot merge the number of industries to match that of the OECD. Therefore, the number of cells required in the domestic matrix will change from 45 to 48.



This issue needs to be handled as well in the selection of the Final Demand columns. This is achieved in the way shown below. Avoid to change any of these numbers when working with the two scripts.



### Naming Conventions Change

Finally, we slightly change the naming convention to ensure consistency between the two types of scripts.

share\_csv\_data = df\_final.to\_csv(f'FIGARO\_{year}\_{method}\_new.csv', index = True)

1. Assuming identical naming conventions regarding the demand components. [↑](#footnote-ref-1)