**Advanced EEIOA course 2022**

**Week 16 Exercises: WIOT [April 19th 2022]**

**Objectives**

* Understand the structure of a WIOT
* Build a WIO model in Python
* Analyze data from a WIO model

This workshop will consist of two parts. **Part 1** will test and develop your conceptual understanding of waste input output tables and their treatment of economic and physical flows. **Part 2** will require you to construct a Waste Input-Output model in Python and analyse the waste streams linked to economic activity

**Part 1: Understanding the structure of a WIOT**

Table 1. WIOT of country C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | E | WT |  | Y |
| E | 52 | 22 |  | 100 |
| WT | 349 | 44 |  | 50 |

where:

E= economic sector, WT= Waste treatment, and Y=final demand

Considering table 1:

1a. What are the expected units in: E-WT quadrant (i.e., with value 22), WT-WT quadrant (i.e., with value 44), and in Y (i.e., with values 100 and 50)?

1b. Estimate the A-matrix. What do the coefficient represent in each quadrant?

**Part 2: Python exercises**

In this exercise, we will create a WIOT in Python

Table

Description automatically generated

Figure 2. Basic structure of a WIOT

1a. Import "week\_16\_data.xlsx" as a pd.dataframe, using the following code:

|  |
| --- |
| # A) Import data  data = 'week\_16\_data.xlsx'  ZYeconomy = pd.read\_excel(data, sheet\_name = 'ZYeconomy', index\_col = 0) # Input-output table, including intermediate and final demand matrix  ZYwaste = pd.read\_excel(data, sheet\_name = 'ZYwaste', index\_col = 0) # Waste account, including intermediate and final demand matrix  F = pd.read\_excel(data, sheet\_name = 'F', index\_col = 0) # VA and satellite account  Q = pd.read\_excel(data, sheet\_name = 'Allocation', index\_col = 0) # Waste allocation matrix  unit = pd.read\_excel(data, sheet\_name = 'Units', index\_col = False) # units |

2. Build a WIOT in Python

You can choose between 2.1 Python expert exercise (no code), and 2.2. Becoming a Python expert (with guiding code)

2.1. Python expert (no code)

- For question 2, there are 3 main steps:

2.1a Calculate the net flows of waste:

, [1]

where are the net waste flows for economic sector E, are the waste outflows for economic sector E, and is the waste inflow for economic sector E;

, [2]

where are the net waste flows for waste treatment sector WT, are the waste outflows for waste treatment sector WT, and is the waste inflow for waste treatment sector WT;

, [3]

where are the net waste flows for final demand Y, is the waste outflows for final demand Y, and is the waste inflow for final demand Y.

2.1b Covert waste flows into waste treatments by using the allocation matrix:

, [4]

where is a matrix of waste flows generated and/or absorbed by economic sector E, and is the allocation matrix;

, [5]

where is a matrix of waste flows generated and/or absorbed by waste treatment sector WT;

, [6]

where is a matrix of waste flows generated by final demand Y.

2.1c Merge elements into the WIOT (as shown in figure 2 above)

2.2. Becoming a python expert (with guiding code)

- For question 2, there are 3 main steps:

2.2a Calculate the net flows of waste:

, [1]

where is the net waste flows for economic sector E, are the waste outflows for economic sector E, and is the waste inflow for economic sector E;

, [2]

where is the net waste flows for waste treatment sector WT, are the waste outflows for waste treatment sector WT, and is the waste inflow for waste treatment sector WT;

, [3]

where are the net waste flows for final demand Y, are the waste outflows for final demand Y, and is the waste inflow for final demand Y.

|  |
| --- |
| # First, identify the number of sectors and sector headers  n\_e = 103 # number of economic sectors  n\_wt = 13 # number of waste treatment sectors  n\_y = 11 # number of final demand categories  n\_w = 79 # number of waste flows  # Second, Separate Z\_e\_e, Z\_e\_wt, Y, Z\_wo\_e, Z\_wi\_e, Z\_wi\_wt, Z\_wi\_wt, Y\_wi, Y\_wo  Z\_e\_e = ZYeconomy.iloc[:, 0:n\_e] # IO economy with the 103 economic sectors  Z\_e\_wt = ZYeconomy.iloc[:, n\_e:n\_e + n\_wt] # IO waste treatment with the 13 waste treatment categories  Y\_e = ZYeconomy.iloc[:, n\_e+n\_wt:n\_e+n\_wt+n\_y] # Final demand matrix  Z\_wo\_e = ZYwaste.iloc[0:n\_w, 0:n\_e] # Waste outflow of 103 economic sectors  Z\_wi\_e = ZYwaste.iloc[n\_w:n\_w+n\_w, 0:n\_e] # Waste inflow of 103 economic sectors  Z\_wo\_wt = ZYwaste.iloc[0:n\_w, n\_e:n\_e+n\_wt] # Waste outflow of 13 waste treatments  Z\_wi\_wt = ZYwaste.iloc[n\_w:n\_w+n\_w, n\_e:n\_e+n\_wt] # Waste inflow of 13 wasre treatments  Y\_wo = ZYwaste.iloc[0:n\_w, n\_e+n\_wt:n\_e+n\_wt+n\_y] # Waste outflow of 11 final demand categories  Y\_wi = ZYwaste.iloc[n\_w:n\_w+n\_w, n\_e+n\_wt:n\_e+n\_wt+n\_y] # Waste inflow of 11 final demand categories  # Then, calculate net waste flows Z\_w\_e, Z\_w\_wt, and Y\_w  Z\_w\_e = Z\_wo\_e.values - Z\_wi\_e.values # Net waste flow of 103 economic sectors. Note: Here pd.dataframe is converted first into np.array  Z\_w\_wt = Z\_wo\_wt.values - Z\_wi\_wt.values # Net waste flow of 13 waste treatment  Y\_w = Y\_wo.values - Y\_wi.values # Net waste flow of 11 final demand categories |

2.2b Convert waste flows into waste treatments by using allocation matrix:

, [4]

where is a matrix of waste flows generated and/or absorbed by economic sector E, and is the allocation matrix;

, [5]

where is a matrix of waste flows generated and/or absorbed by waste treatment sector WT;

, [6]

where is a matrix of waste flows generated by final demand Y.

|  |
| --- |
| # Covert waste flows into waste treatments Z\_wt\_e, Z\_wt\_wt, Y\_wt  Z\_wt\_e = Q.values @ Z\_w\_e # Matrix of waste flows generated and/or absorbed by 103 economic sector  Z\_wt\_wt = Q.values @ Z\_w\_wt # Matrix of waste flows generated and/or absorbed by 13 waste treatment sectors sector  Y\_wt = Q.values @ Y\_w # Matrix of waste flows generated and/or absorbed by 11 final demand categories |

2.2c Merge elements into the WIOT (as shown in figure 2 above)

|  |
| --- |
| # Merge matrices in a WIOT arrange  ## First, let's create pd.dataframes  Z\_wt\_e = pd.DataFrame(Z\_wt\_e, index=Q.index, columns=Z\_e\_e.columns)  Z\_wt\_wt = pd.DataFrame(Z\_wt\_wt, index=Q.index, columns=Z\_e\_wt.columns)  Y\_wt = pd.DataFrame(Y\_wt, index=Q.index, columns=Y\_e.columns)  wiot\_e = pd.concat([Z\_e\_e, Z\_e\_wt], axis=1) # Concat Z\_e\_e and Z\_e\_wt block  wiot\_wt = pd.concat([Z\_wt\_e, Z\_wt\_wt], axis=1) # Concat Z\_wt\_e and Z\_wt\_wt block  Z\_wiot = pd.concat([wiot\_e, wiot\_wt], axis=0) # Integrate WIOT  Y\_wiot = pd.concat([Y\_e, Y\_wt], axis=0) # Concat Y\_e and Y\_wt |

3. **Extra:** From a consumption-based perspective, which are the major contributors to landfill area footprint?

You can choose between 3.1 Python expert exercise (no code), and 3.2. Becoming a Python expert (with guiding code)

3.1. Python expert (no code)

3.1a Using the WIOT, calculate the total output vector (x), A matrix, and L matrix

3.1b Calculate the landfill area intensity (f)

3.1c Calculate the embodied landfill area (i.e., landfill footprint) for the total final demand

3.1d Display the top-5 contributors. What can you infer from the results?

3.2. Becoming a python expert (with guiding code)

With the WIOT, calculate the total output vector (x), A matrix, and L matrix

|  |
| --- |
| x = Z\_wiot.sum(axis = 1).values + Y\_wiot.sum(axis = 1).values  xinv = np.diag(((x != 0)/(x + (x == 0))).flatten())  A = Z\_wiot @ xinv  I = np.identity(len(A)) # An identy matrix of the size as A-matrix  L = np.linalg.inv(I - A) |

3.2a Calculate the landfill area intensity (f)

|  |
| --- |
| ## Landfill intensity  F = F.iloc[8,: n\_e+n\_wt] #Landfill area in square meter is in position 8  f = F @ xinv |

3.2b Calculate the embodied landfill area (i.e., landfill footprint) for the total final demand

|  |
| --- |
| ## Calculate land footprint per economic sector(LF)  y\_sum = Y\_wiot.sum(1).values # Final demand vector  LF = f @ L @ np.diag(y\_sum.flatten()) # Landfill footprint calculation  LF = pd.DataFrame(LF, index=Y\_wiot.index, columns=['Landfill footprint']) # Create new dataframe with LF  LF\_sort = LF.sort\_values(by='Landfill footprint', ascending=False) # Sort results |

3.2c Display the top-5 contributors. What can you infer from the results?

|  |
| --- |
| print(LF\_sort.head(5)) # Print top-5 contributors |