1.1._B_gas_nongas_io_table

February 19, 2025

[ ]:
```
import numpy as np
import os
import pymrio
import pandas as pd
from IPython.display import display
import country_converter as coco
```

[ ]:
```
# Adjust Pandas options for better display
pd.set_option("display.max_columns", None)  # Show all columns
pd.set_option("display.max_rows", None)  # Show all rows
pd.set_option("display.width", 1000)  # Prevent wrapping
pd.set_option("display.expand_frame_repr", False)  # Prevent wrapping
```

[ ]:
```
### Fetch Exiobase3 data and convert it to a pymrio object ###

# Define the storing folder for Exiobase3 data
exio3_folder = 'C:/Users/danie/Nextcloud/Coding/Masterthesis/exiobase'

# Download Exiobase3 data
exio_downloadlog = pymrio.download_exiobase3(storage_folder=exio3_folder,␣
 ↪system="ixi", years=[2019, 2020, 2021, 2022])
print(exio_downloadlog)
```

[ ]:
```
# Parse Exiobase3 (2021) data
exio3 = pymrio.parse_exiobase3(path='C:/Users/danie/Nextcloud/Coding/
 ↪Masterthesis/exiobase/IOT_2021_ixi.zip')
```

[ ]:
```
# Assess meta data
print(exio3.meta)
```

[ ]:
```
### Check for geographical sampling differences between FIGARO and EXIOBASE 3␣
 ↪###

# FIGARO countries
figaro_countries = [
    'AR', 'AT', 'AU', 'BE', 'BG', 'BR', 'CA', 'CH', 'CN', 'CY', 'CZ', 'DE',␣
 ↪'DK', 'EE', 'ES', 'FI', 'FIGW1', 'FR', 'GB',
```

```
    'GR', 'HR', 'HU', 'ID', 'IE', 'IN', 'IT', 'JP', 'KR', 'LT', 'LU', 'LV',␣
 ↪'MT', 'MX', 'NL', 'NO', 'PL', 'PT', 'RO', 'RU',
    'SA', 'SE', 'SI', 'SK', 'TR', 'US', 'ZA'
]

# Extract country codes from EXIOBASE 3 dataset
exio_countries = exio3.get_regions()

# Compare country codes
common_countries = sorted(set(figaro_countries).intersection(exio_countries))
figaro_only_countries = sorted(set(figaro_countries) - set(exio_countries))
exio_only_countries = sorted(set(exio_countries) - set(figaro_countries))

print("Common countries:", common_countries)
print("Countries only in FIGARO:", figaro_only_countries)
print("Countries only in EXIOBASE 3:", exio_only_countries)

# Argentina and Saudi Arabia are not in EXIOBASE 3, but in FIGARO
# Taiwan is in EXIOBASE 3, but not in FIGARO
# FIGW1 is ROW in Figaro
# WA (Asia), WE (Europe), WF (Africa), WL (Latin America), WM (Middle East), WP␣
 ↪(Pacific) are ROW regions in Exiobase3

# Taiwan will be added to WA to avoid dimensinality issues
exio3.rename_regions({'TW': 'WA'})
```

[ ]:
```
# Check available classification data that contains possibly useful different␣
 ↪names and aggregation levels
mrio_class = pymrio.get_classification(mrio_name='exio3_ixi')

# Display the full mrio_class
display(mrio_class)

# Switch sector naming to ExioLabel due to better readability
```

[ ]:
```
# Create a conversion dictionnary from ExioName to ExioLabel and check for␣
 ↪correctness by displaying it
conv_dict = mrio_class.get_sector_dict(mrio_class.sectors.ExioName, mrio_class.
 ↪sectors.ExioLabel)
display(conv_dict)

# Rename sectors in the pymrio object
exio3.rename_sectors(conv_dict)

# Check if the renaming was successful
print(exio3.Z.index)
```

[ ]:
```
### Aggregate Exiobase3 data ###
# Done through renaming, which also helps to adapt it to eurostat data

# Renaming of sectors requires mappping of ExioLabel to NACE classification

rename_dict_exio3_NACE = {
    "A_PARI": "A01",
    "A_WHEA": "A01",
    "A_OCER": "A01",
    "A_FVEG": "A01",
    "A_OILS": "A01",
    "A_SUGB": "A01",
    "A_FIBR": "A01",
    "A_OTCR": "A01",
    "A_CATL": "A01",
    "A_PIGS": "A01",
    "A_PLTR": "A01",
    "A_OMEA": "A01",
    "A_OANP": "A01",
    "A_MILK": "A01",
    "A_WOOL": "A01",
    "A_MANC": "A01",
    "A_MANB": "A01",
    "A_FORE": "A02",
    "A_FISH": "A03",
    "A_GASE": "B_gas",
    "A_OGPL": "B_gas",
    "A_COAL": "B_nongas",
    "A_COIL": "B_nongas",
    "A_ORAN": "B_nongas",
    "A_IRON": "B_nongas",
    "A_COPO": "B_nongas",
    "A_NIKO": "B_nongas",
    "A_ALUO": "B_nongas",
    "A_PREO": "B_nongas",
    "A_LZTO": "B_nongas",
    "A_ONFO": "B_nongas",
    "A_STON": "B_nongas",
    "A_SDCL": "B_nongas",
    "A_CHMF": "B_nongas",
    "A_PCAT": "C10-12",
    "A_PPIG": "C10-12",
    "A_PPLT": "C10-12",
    "A_POME": "C10-12",
    "A_VOIL": "C10-12",
    "A_DAIR": "C10-12",
    "A_RICE": "C10-12",
```

```
    "A_SUGR": "C10-12",
    "A_OFOD": "C10-12",
    "A_BEVR": "C10-12",
    "A_FSHP": "C10-12",
    "A_TOBC": "C10-12",
    "A_TEXT": "C13-15",
    "A_GARM": "C13-15",
    "A_LETH": "C13-15",
    "A_WOOD": "C16",
    "A_WOOW": "C16",
    "A_PULP": "C17",
    "A_PAPR": "C17",
    "A_PAPE": "C17",
    "A_MDIA": "C18",
    "A_COKE": "C19",
    "A_REFN": "C19",
    "A_PLAS": "C20-21",
    "A_PLAW": "C20-21",
    "A_NFER": "C20-21",
    "A_PFER": "C20-21",
    "A_CHEM": "C20-21",
    "A_RUBP": "C22",
    "A_GLAS": "C23",
    "A_GLAW": "C23",
    "A_CRMC": "C23",
    "A_BRIK": "C23",
    "A_CMNT": "C23",
    "A_ASHW": "C23",
    "A_ONMM": "C23",
    "A_NUCF": "C24",
    "A_STEL": "C24",
    "A_STEW": "C24",
    "A_PREM": "C24",
    "A_PREW": "C24",
    "A_ALUM": "C24",
    "A_ALUW": "C24",
    "A_LZTP": "C24",
    "A_LZTW": "C24",
    "A_COPP": "C24",
    "A_COPW": "C24",
    "A_ONFM": "C24",
    "A_ONFW": "C24",
    "A_METC": "C24",
    "A_FABM": "C25_33",
    "A_MACH": "C25",
    "A_OFMA": "C26",
    "A_ELMA": "C27",
```

```
    "A_RATV": "C27",
    "A_MEIN": "C28_32",
    "A_MOTO": "C29",
    "A_OTRE": "C30",
    "A_FURN": "C31",
    "A_POWC": "D35",
    "A_POWG": "D35",
    "A_POWN": "D35",
    "A_POWH": "D35",
    "A_POWW": "D35",
    "A_POWP": "D35",
    "A_POWB": "D35",
    "A_POWS": "D35",
    "A_POWE": "D35",
    "A_POWO": "D35",
    "A_POWM": "D35",
    "A_POWZ": "D35",
    "A_POWT": "D35",
    "A_POWD": "D35",
    "A_GASD": "D35",
    "A_HWAT": "D35",
    "A_WATR": "E36",
    "A_RYMS": "E37-39",
    "A_BOTW": "E37-39",
    "A_INCF": "E37-39",
    "A_INCP": "E37-39",
    "A_INCL": "E37-39",
    "A_INCM": "E37-39",
    "A_INCT": "E37-39",
    "A_INCW": "E37-39",
    "A_INCO": "E37-39",
    "A_BIOF": "E37-39",
    "A_BIOP": "E37-39",
    "A_BIOS": "E37-39",
    "A_COMF": "E37-39",
    "A_COMW": "E37-39",
    "A_WASF": "E37-39",
    "A_WASO": "E37-39",
    "A_LANF": "E37-39",
    "A_LANP": "E37-39",
    "A_LANL": "E37-39",
    "A_LANI": "E37-39",
    "A_LANT": "E37-39",
    "A_LANW": "E37-39",
    "A_CONS": "F",
    "A_CONW": "F",
    "A_TDMO": "G45",
```

```
    "A_TDWH": "G46",
    "A_TDFU": "G47",
    "A_TDRT": "G47",
    "A_TRAI": "H49",
    "A_TLND": "H49",
    "A_TPIP": "H49",
    "A_TWAS": "H50",
    "A_TWAI": "H50",
    "A_TAIR": "H51",
    "A_TAUX": "H52",
    "A_PTEL": "H53",
    "A_HORE": "I",
    "A_COMP": "J62_63",
    "A_FINT": "K64",
    "A_FINS": "K65",
    "A_FAUX": "K66",
    "A_REAL": "L68",
    "A_RESD": "M_N",
    "A_OBUS": "M_N",
    "A_HEAL": "M_N",
    "A_MARE": "M_N",
    "A_PADF": "O84",
    "A_EDUC": "P85",
    "A_RECR": "R_S",
    "A_ORGA": "R_S",
    "A_OSER": "R_S",
    "A_PRHH": "T",
    "A_EXTO": "U"
}
```

[ ]:
```
# Apply mapping with the rename_sectors tool of pymrio
exio3.rename_sectors(rename_dict_exio3_NACE)
print(exio3.Z.index)

# Aggregate duplicates
exio3.aggregate_duplicates()
print(exio3.Z)
```

[ ]:
```
### Code snippet that retrieves B_gas and B_nongas for each country ###
# Remember Multiindex, create two row matrix for each country
# Do a re-order according to country
# Then calculate weights
```

[ ]:
```
# Extraction snippet of the energy outputs

# Extract the energy matrix for B_gas and B_nongas
energy_matrix = exio3.Z.loc[(slice(None), ['B_gas', 'B_nongas']), :]
```

```
# Reorder the matrix according to the country
energy_matrix = energy_matrix.sort_index(level=0)

# Display the reordered energy matrix
print(energy_matrix)
```

[ ]:
```
# Extract the energy inputs for each sector and country
energy_inputs = exio3.Z.loc[(slice(None), ['B_gas', 'B_nongas']), :]

# Compute the total energy inputs by summing the gas and nongas inputs
total_energy_inputs = energy_inputs.groupby(level=0).sum()

# Calculate the share of gas and nongas in the total energy inputs
energy_shares = energy_inputs.div(total_energy_inputs, level=0)

# Sort the energy shares so that gas and nongas of each country are next to␣
 ↪each other
energy_shares = energy_shares.sort_index(level=0)

# Display the energy shares
print(energy_shares)
energy_shares.to_csv('C:/Users/danie/Nextcloud/Coding/Masterthesis/exiobase/
 ↪energy_shares.csv')
```

[ ]:
```
### Code snippet that brings Figaro Dataset in accordance with our weights␣
 ↪dataset ###
# Duplicate B sector, since the weights will be applied for B_gas and B_nongas␣
 ↪of the B cell value anyway
# Change indexing (either weights to figaro format, but rather figaro to␣
 ↪weights format)
```