Exercise: Analyzing the Country Export Network with the BiCM

Overview of the Dataset

This exercise uses data from the global export matrix for 2022. The dataset consists of three key files:

1. **Export Matrix (`exportmatrix2022.csv`)**:
    - This is a 171×5,206171 \times 5,206171×5,206 matrix where each entry (i,j)(i, j)(i,j) represents the export value (in USD) of product jjj by country iii in 2022.
    - Rows correspond to countries, and columns correspond to products.
2. **Country List (`countries.csv`)**:
    - A file containing the names of 171 countries, in the same order as the rows of the export matrix.
    - Each entry in this file corresponds to a row in the export matrix.
3. **Product List (`products.csv`)**:
    - A file containing the names of 5,206 products, in the same order as the columns of the export matrix.
    - Each entry in this file corresponds to a column in the export matrix.

These files allow you to map the numerical values in the export matrix to specific countries and products.

In this exercise you will use the Bipartite Configuration Model. To do so you need to install the library bicm, here you can find the documentation

https://pypi.org/project/bicm/

Required Imports

Before starting, import the following libraries:

python
Copia codice
```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from bicm import BipartiteGraph
from community import community_louvain
```

Step 1: Data Import

Instructions:

1. Import the **export matrix** as a NumPy array using `pandas.read_csv`.
2. Load the **country** and **product lists** as Python lists.
3. Verify that the dimensions of the export matrix are 171×5206, and that the numbers of countries and products match their respective lists.

Step 2: Compute the Revealed Comparative Advantage (RCA)

What is RCA?

The **RCA** measures a country's comparative advantage in exporting a product. It is calculated as:

$$\mathrm{RCA}_{cp} = \frac{\frac{x_{cp}}{X_c}}{\frac{X_p}{X_{\text{tot}}}}$$

Where:

- x_{cp}: Export value of country c in product p,
- X_c: Total exports of country c,
- X_p: Total exports of product p,
- X_{tot}: Total global exports.

If $\mathrm{RCA}_{cp} > 1$, country c has a comparative advantage in product p.

Using the code below

1. compute the RCA for each country-product pair

2. get a binary adjacency matrix by setting to 1 values of RCA equal or larger than 1 and to 0 the other

Full Code for RCA:

```
# Compute totals
X_c = export_matrix.sum(axis=1, keepdims=True)  # country totals
X_p = export_matrix.sum(axis=0, keepdims=True)  # product totals
X_tot = export_matrix.sum()  # global total

# Compute RCA
RCA = (export_matrix / X_c) / (X_p / X_tot)

# Binarize the RCA matrix: 1 if RCA > 1, else 0
binary_matrix = (RCA > 1).astype(int)
```

Step 3: Naive Network Projection

Objective:

Create a country-country network where two countries are connected if they share at least one product in which both have an RCA > 1.

Instructions:

1. Compute the adjacency matrix using cooccurrences: link any two countries that export at least a common product.
2. You can do this in several way, with a for cycle or with a matrix multiplication for instance
3. Convert the adjacency matrix into a graph using `NetworkX`.
4. Visualize the graph using a spring layout and label a few key countries.

Step 4: Bipartite Configuration Model (BICM) Projection

What is the BICM?

The **Bipartite Configuration Model (BICM)** generates a random bipartite graph with the same degree distribution as the observed graph. This model filters out statistically insignificant connections, leaving only meaningful ones.

Key Functions from the `bicm` Library:

1. `BipartiteGraph()`:
    - Initializes a bipartite graph object.
2. `set_biadjacency_matrix()`:
    - Loads your binarized RCA matrix into the BICM model.
3. `compute_projection(rows=True, alpha=0.02, method='poisson', threads_num=4)`:
    - Computes the projection of the bipartite graph onto one of its layers (countries in this case).
        - `rows=True`: Projects onto countries.
        - `alpha=0.02`: Sets the significance threshold.
        - `method='poisson'`: Specifies the statistical model to use.
4. `get_rows_projection(fmt='edgelist')`:
    - Extracts the significant connections between countries as an edge list.

Instructions:

1. Use the above functions to compute the BICM projection.
2. Create a graph using `NetworkX` with the significant connections.
3. Visualize the graph and compare it with the naive projection.

Step 5: Community Detection with Louvain

Objective:

Identify groups of countries with similar export baskets in the BICM-projected network.

Instructions:

Apply the Louvain algorithm to the BICM-projected network using:
`partition = community_louvain.best_partition(G_bicm)`

1. This returns a dictionary mapping each country to its community.
2. Visualize the communities:
    - Assign each community a unique color.
    - Label key countries.

Bonus 1: Export Communities and Name Them with ChatGPT

1. **Group countries by community**:

    - Create a dictionary where keys are community IDs and values are lists of countries in each community.
2. **Export all communities**:
    - Save all communities together in a single text file or message format.

Example:
```
Community 0:
Country1, Country2, Country3, ...

Community 1:
CountryA, CountryB, CountryC, ...
```

3. **Ask ChatGPT to name the communities**:
    - Provide all communities to ChatGPT at once.
    - Explain that these communities represent groups of countries with **similar export baskets** and ask for meaningful names for each group.

Example Prompt for ChatGPT:
```
These are groups of countries based on similarity in their export
baskets. Please suggest descriptive names for each group:
- Community 0: USA, Germany, China, ...
- Community 1: Brazil, Argentina, ...
```

Bonus 2: Enhanced Visualization with Community Names

1. Use the names generated by ChatGPT to label each community.
2. Create a legend mapping colors to community names.
3. Highlight a few key countries (e.g., USA, China, Germany) in the graph.