Lab 03: Migration Network Analysis with Python Classes

Course: Programming for Data Science
Professor: Dr. Carlos J. Costa

February 18, 2026

In this lab, you will take on the role of a data analyst studying global migration flows using the `international_migration_flow.csv` dataset, available at:
`https://data.humdata.org/dataset/international-migration-flows`

The dataset contains information about migration between countries over time, including the sending country (`country_from`), the receiving country (`country_to`), the month of migration (`migration_month`), and the number of migrants (`num_migrants`).

Your task is to design and implement Python classes that load, analyze, and visualize this dataset. You will practice object-oriented programming, network analysis with `networkx`, data manipulation with `pandas`, and data visualization with `matplotlib`. You are encouraged to enhance the classes by exploring migration characteristics that you consider most relevant, and to interpret your analyses while taking into account the limitations of the data collection process.

Task 1: Data Loader Class

The first task is to create a Python class that reads the CSV file and validates the presence of the required columns. The class should handle errors gracefully, such as missing columns or invalid file paths. After implementing the class, test it by loading the dataset and displaying the first few rows to verify the import.

Expected output: a pandas DataFrame containing the migration data.

Task 2: Temporal Network Constructor

Next, you will implement a class that constructs a directed weighted network for each month in the dataset. Each node represents a country, and each directed edge represents migration from one country to another with a weight equal to the number of migrants. The class should provide a method that returns a dictionary mapping each month to its corresponding `networkx.DiGraph` object.

Expected output: a dictionary of monthly networks and inspection of nodes and edges for at least one month.

Task 3: Centrality Analyzer Class

In this task, create a class that computes centrality measures on the monthly networks:

- **PageRank:** identifies structurally important countries based on the recursive importance of their migration connections.
- **Betweenness:** identifies corridor countries that connect otherwise weakly connected regions.
- **Centralization:** measures how concentrated the migration network is around the most central countries.

The class should provide methods to compute each measure over all months and to identify the top countries according to each centrality metric.

Expected output: DataFrames showing PageRank and Betweenness over time, a series showing centralization over time, and lists of the top countries.

Task 4: Visualization Functions

Implement separate functions to visualize the results:

- Plot the evolution of PageRank for the top countries over time.
- Plot the centralization of the migration network over time.

Expected output: clear line plots showing temporal trends in centrality and network structure.

Task 5: Incremental Analysis and Interpretation

Using the classes from previous tasks, perform a full analysis:

1. Load the dataset using your DataLoader class.
2. Build monthly networks using the TemporalNetwork class.
3. Compute PageRank, Betweenness, and Centralization using the CentralityAnalyzer class.
4. Identify the persistent structural hubs and corridor countries.
5. Generate plots showing centrality evolution and network centralization.

Interpret the results:

- Which countries are consistently central hubs (high PageRank and low volatility)?
- Which countries act as corridors (high betweenness)?
- How centralized is the global migration network over time?

Task 6: Refactoring and Extension

Finally, review your classes and methods. Consider improvements such as:

- Adding handling for zero migrant flows to avoid errors in betweenness computation.
- Extending the analyzer to compute volatility of PageRank over time.
- Adding functionality to detect structural role switches or emerging hubs.
- Creating helper methods for repeated plotting operations.

The goal is to produce a clean, modular Python module or notebook where all classes work seamlessly together and produce interpretable results.

Submission Requirements

Students should submit a Jupyter Notebook (`.ipynb`) or Python script (`.py`) containing:

- Implementation of all classes and functions.
- Demonstration of the analysis workflow.
- Visualizations generated from the dataset.
- Brief interpretation of the results for each centrality measure.