ENSF 692: Final Project (Spring 2025)

Authors: Conner Castle & William Watson

# Introduction

This report describes the functionality of the accompanying python file ‘data\_analysis.py’. The purpose is to:

1. Document the data processing workflow used to transform three separate sets of macroeconomic data in csv format into a unified data frame that has undergone several processing and analysis steps and eventually been converted back to a new csv file.
2. Describe the command line interface functionality that allows for user specified analysis of subsets of the data.
3. Explain how and where the project requirements have been met. These will be referenced to either function names or processing steps in main().

Ancillary files such as Jupyter notebooks are included in the GitHub repository for reference, as these were the primary method for exploratory analysis and processing.

# Datasets

This investigation focuses on the relationship between the GDP of nations and their levels of domestic income inequality. The observation that motivated this investigation is the following: Ireland currently has the highest GDP Per Capita in the world. However, it is widely known that this is not the result of an increasingly powerful economy. Rather, it is due to the massive influx of direct foreign investment (DFI) because of a sharp reduction in national corporate tax rate. Ireland has become a tax haven for multinationals operating in Europe. The spike in GDP (and by extension, GDP Per Capita) has been labeled an accounting trick, and satirically dubbed 'Leprechaun Economics'. So, how does this spike in GDP affect the quality of life for the inhabitants of Ireland? Is it a merely a vapid statistic, or is there a tangible benefit to the individual people?

Data Sources:

Total GDP (USD, Inflation Adjusted) https://www.gapminder.org/data/

GDP Per Capita (USD, Inflation Adjusted) https://www.gapminder.org/data/

Gini Coefficient (USD, Inflation Adjusted) https://www.gapminder.org/data/

Note: The Gini Coefficient is a measure of national income inequality, represented as a value between zero and 100 where zero is perfect equality, and 100 in maximum inequality.

# Data Processing Workflow

The data was processed in the following steps:

## Data Cleaning and Merging

1. Read in raw data from CSV files to Pandas data frames (‘Total GDP’, ‘GDP Per Capita’, and ‘Gini Coeff’ data.)
2. Sort and copy the raw data to perform manipulation without modifying the original data.
3. Parse the Total GDP and GDP Per Capita data frame values to convert from strings to floats.
   * E.g., ‘1.78TN’ 🡪 1,780,000,000,000.0
4. Merge the datasets using an inner join on the rows and columns. This ensures only common countries and years are preserved. The combined dataset is MultiIndexed on the rows, where the outer index is the ‘Metric’: [‘Total GDP’, ‘GDP Per Capita’, ‘Gini Coeff’], and the inner index is the ‘Country’.
5. Filter rows (countries) based on the number of nan values. In this case, a country is dropped if less than 50% of years contained real values in either ‘Total GDP’ or ‘GDP Per Capita’.

## General Aggregate Statistics

1. Show aggregate statistics for the cleaned datasets using the *describe()* method applied to each metric.

## Derive New Data

1. Create new dataset from existing data to reflect the GDP Per Capita adjusted for inequality. To do this, a new metric is defined and calculated as follows:

Where is a value between (0 – 1).

This metric for each year and country is then added as rows to the data frame under the new metric index *‘Gini Dollars’.*

## Add Level to MultiIndex Rows

1. Create a new classification label for each metric, that when combined will be added as an additional row index layer called ‘Classification’. For each metric, these are defined as follows:
   * ‘Total GDP’ classified as economic weight class:
     + ‘Superpower’ : > 10 trillion USD
     + ‘Heavyweight’ : 1 trillion – 10 trillion USD
     + ‘Middleweight’ : 100 billion – 1 trillion USD
     + ‘Lightweight’ : < 100 billion USD
   * ‘GDP Per Capita’ classified as income level:
     + ‘Lower Income’ : < 1,135 USD
     + ‘Lower Middle Income’ : 1,135 – 4,465 USD
     + ‘Upper Middle Income’ : 4,465 – 13,845 USD
     + ‘High Income’ : 13,845 – 40,000 USD
     + ‘Ultra-High Income’ : > 40,000 USD
   * ‘Gini Coeff’ classified as inequality level:
     + ‘Very Low Inequality’ : < 30
     + ‘Low Inequality’ : 30 – 35
     + ‘Low-Moderate Inequality’ : 35 – 40
     + ‘Moderate-High Inequality’ : 40 – 45
     + ‘High Inequality’ : 45 – 50
     + ‘Very High Inequality’ : > 50
   * ‘Gini Dollars’ classified as livability level:
     + ‘Low Livability’ : < 30
     + ‘Low-Moderate Livability’: 30 – 35
     + ‘Moderate Livability’: 35 – 40
     + ‘Moderate-High Livability’: 40 – 45
     + ‘High Livability’: 45 – 50
2. Use the *describe()* method again to show aggregate stats for all data, including the newly derived additional data.
3. Show a pivot table of averaged values for a specific year (default is end year of user input range), pivoting on ‘Metric’ and ‘Classification’.

## User Input

1. Prompt the user to input a range for the years to include in the analysis. For example, [‘1960’, ‘2023’] (full dataset), or [‘1990’, ‘2010’] (partial dataset).
2. Prompt the user to input countries one-at-a-time, where detailed analysis will be performed on that subset. E.g., [‘Canada’, ‘Germany’, ‘Ireland’, ‘South Korea’].
3. Plot trends and aggregate statistics for the selected countries across the selected time frame.
4. Write the final MultiIndex data frame to an Excel file.

# Project Requirements

This section describes how and where this project meets the stated requirements.

|  |  |
| --- | --- |
| Project Requirement | Description of Meeting Requirement |
| Must use at least 3 separate dataset files. | Project uses 3 datasets from separate csv files, containing data on Total GDP, GDP Per Capita, and Gini Coefficient. |
| Final data must have at least 10 columns and 200 rows. | Final data frame shape is: (732,64) |
| Program must not modify excel files directly. | Program reads in and makes copies of original data, preserving the integrity of the data. |
| Must not hard-code / copy paste values. | All data is read in and modified via variables. |
| Must use at least 2 merge/joins on the data. | We do not use the merge or join functions explicitly. This is because the shape of each file is 2D, and it makes more sense to stack vertically. Therefore, we use the *pd.concat()* method to stack our data, and the MultiIndex of [‘Metric’, ‘Country’]. |
| You must create a hierarchical index of at least two levels (row or column). | We use a MultiIndex on the rows with 3 levels: [‘Metric’, ‘Classification’, and ‘Country’] |
| All data should be presented in the correctly sorted order, depending on the index. | The countries are sorted alphabetically, and this applies in each Metric. |
| You may not use global variables. You must import the data within your main function. | No global variables used. Data imported from csv files in *main().* |
| Remember to check for null values or data mismatches. | Data is combined using inner join on both rows and columns, ensuring consistent shape of dataset. Additionally, rows that do not meet a specified threshold of valid data values are dropped. Remaining missing values are ignored. |
| Your application must return useful information. Design an interface that allows users to search based on some sort of criteria or keywords.  The user must provide at least two pieces of information/selection (e.g. "school name" and "grade"). | User interface asks user to specify a time range for detailed analysis by entering a  Start Year:  End Year:  Then, the user is asked to enter an arbitrary number of countries one-at-a-time for analysis. For example, user could enter  Country 1: Canada  [‘Canada’]  Country 2: Germany  [‘Canada’, ‘Germany’]  Country 3: Ireland  [‘Canada’, ‘Germany’, ‘Ireland’] |
| Give the user clear input instructions. If an invalid entry is given, use try/except statements to handle the error and continue to prompt for user input. | User can press 1 for a list of valid country names to enter.  Two functions: *check\_years()* and *check\_country()* use error handling to validate user input. |
| Any output information must be clearly defined using printed headers (DataFrame tables) or sentences (scalar values). | General information output is separated by header lines, user specific information is plotted in figures with appropriate title and legend. |
| Use the describe method to print aggregate stats for the entire dataset. | Our *describe\_all()* function uses the *describe()* method applied to each Metric in the data frame. |
| Add at least two columns to the combined dataset. | We add a column ‘Classification’ as an additional layer of row indexing. And we also derive a new value (Gini Dollars) for each year/country. However, this is added as additional rows vertically. Again, this makes more sense given the shape of our data. |
| Use an aggregation computation for a subset of the data. | For the set of user selected countries, the mean values are plotted along with the individual countries in *plotter().* |
| Use a masking operation. | In *df\_index\_inner\_join()* masking is used to only include countries that are present in all datasets. |
| Use the 'groupby' operation at least once. | \*TODO\* |
| Create and print a pivot table. | A pivot table is created of averaged values for a specific year (default is end year of user input range), pivoting on ‘Metric’ and ‘Classification’. |
| Include at least two user-defined functions or a class that contains two methods. | *drop\_countries\_by\_nan()* takes in a threshold as a percentage, and drops a country if the percent of nan values exceeds the threshold.  *plotter()* takes in the user defined lists of years and countries, and plots the various metrics for the specified data. |
| export your entire merged, hierarchical dataset to an Excel file in the working directory. | Final data frame is exported to “final\_data.xlsx”. \*\*Conner to add\*\* |
| Use your data to create at least one plot using Matplotlib. Save the plot as a .png file and upload it to the repository. | Program generates 4 plots, one for each metric showing the user specified groups of countries. |
| All classes, methods, and functions must contain docstring documentation | All functions contain docstrings, and steps in the main() are commented. |
|  |  |
|  |  |
|  |  |