Emma Carlson

CSE-515

Final Project

15 March 2022

International Data Preprocessing Python Package

**Introduction**

This International Data Preprocessing Python package allows users to clean data frames that include international currency or metrics. This package is lightweight and is easily integrated into current workflows. To properly analyze international data, it is necessary to undergo a process of normalization. This normalization process allows the end-users to make true comparisons of data values from one specific viewpoint, usually defined by country and year. This Python package allows end-users to complete this process by preparing data columns for function input, passing in a data frame and selected columns to a function, and finally executing an intended calculation. Functionality includes adjusting currency for inflation by comparison year, converting international currency values using exchange rates, and providing per capita metrics using country populations by year. The design focuses on accessibility and ease of use, so that end-users may utilize this tool quickly and effectively.

**Use Cases**

Potential users of this package are any individuals interested in utilizing lightweight data cleaning tools. This application of course only applies to individuals wishing to clean datasets involving international data metrics. In order to fully expand upon potential use cases, an example interaction with each major function is presented.

Example 1: User needs to adjust a data frame column from “Total Millionaires” associated with a year and country value to “Millionaires Per Capita”

Process:

1. Install the International Data Preprocessing package for use in Python
2. Read in data frame and any dependencies
3. Assess data frame columns according to input guidelines
   1. Ensure year columns are 4-digit year values
   2. Ensure country column is in alpha 3 format
4. Parse any columns that do not meet input guidelines
5. Execute desired function to complete calculation
   1. For per capita calculations, use adjust\_per\_capita
6. Verify output data meets expectations

Example 2: User needs to adjust a data frame column “Median Income” associated with a year and country value to reflect monetary value in 2022 US Dollars

Process:

1. Install the International Data Preprocessing package for use in Python
2. Read in data frame and any dependencies
3. Assess data frame columns according to input guidelines
   1. Ensure year columns are 4-digit year values
   2. Ensure country column is in alpha 3 format
   3. Ensure currency type is in standard 3-digit code format
4. Parse any columns that do not meet input guidelines
5. Execute desired function to complete calculation
   1. For inflation adjustments, use adjust\_for\_inflation
6. Verify output data meets expectations

Example 3: User needs to adjust a data frame column “Average Credit Card Debt” associated with a year and country from local currency values to US Dollar values

Process:

1. Install the International Data Preprocessing package for use in Python
2. Read in data frame and any dependencies
3. Assess data frame columns according to input guidelines
   1. Ensure year columns are 4-digit year values
   2. Ensure country column is in alpha 3 format
   3. Ensure currency type column is represented by standard 3 letter code
4. Parse any columns that do not meet input guidelines
5. Execute desired function to complete calculation
   1. For currency exchange calculations, use convert\_currency
6. Verify output data meets expectations

These use cases identify three potential occurrences for which an individual might utilize this Python package. The general process for implementation is largely similar for each function. This process structure allows users to learn package implementation quickly.

**Software Modules**

This python package is structured around three central modules. Aside from the init and util files, the three major modules are titled adjustments, country, and currency. The country and currency modules define the classes Country and Money, respectively. These classes help identify these data types within inputted data frames and enables parsing functions. The adjustments file contains the main functionality of this package, housing the major preprocessing functions.

The country module defines the Country class, initialized with the attributes alpha2\_code (standard 2 letter country code), alpha3\_code (standard 3 letter country code), un\_code (UN assigned 3-digit code), full\_name (official full name of country), and short\_name (official abbreviated name of country). These attributes assist in the identification of a given country in an inputted data frame column. A country could be identified using any of these attributes and still be considered valid input. This structure allows for increased input flexibility, so long as the input is one of the country’s standard identifiers. Within this class, there are several functions. There is the standard get function that returns any of the listed attributes. Additionally, there are two main parsing functions. The parsing functions of the country class allow users to parse a value or column of values from one identifying country attribute to another. This allows users to ensure that their data frame columns match expected input without taking additional data cleaning steps. The expected input for country identifiers is a standard alpha 3 code. Users are therefore able to ensure their column values match this input type before function execution using these parsing functions.

Similar to the country module is the currency module. This module defines the Money class with attributes amount (numeric value), currency\_abbv (3 letter standard currency code), and currency\_full (full official currency name). This class allows a given value to be of type Money and receive the assigned attributes. Similar to the Country class, the Money class includes parsing functionality. The three parsing functions allow the user to parse currency identifiers to the expected function input. For currency-related functionality, a three-letter currency code is expected. As many data frames may identify currency values differently, the ability to parse data in this way is especially helpful.

The final module is the adjustments module. This module contains all preprocessing functionality that can be executed once parsing steps are taken. This module contains two instances each of three major functions: adjust\_for\_inflation, adjust\_per\_capita, and convery\_currency. The two instances of each function allow users to execute preprocessing on single values or a column of values. This report will focus on preprocessing column values. The first function, adjust\_for\_inflation, receives a data frame, numeric amount column, associated country column, original year column, and resulting calculation year column. The user must also input whether they want the resulting values in a new column, and a new column name, if necessary. This input allows the function to look up inflation rate values for a given country and year from the main data file, apply the inflation rate to the inputted numeric value column, and return the resulting column as requested. The next function, adjust\_per\_capita, receives a data frame, numeric value column, associated country column, and associated year column as input. This function also requires input from the user if the resulting values should be placed in a new column, and a new column name if necessary. This input allows the function to retrieve country population values from the main data source file and execute the calculation to change numeric values to per capita values. The resulting column will be returned in requested format. The final function, covert\_currency, takes input of a data frame, numeric amount column, original currency column, resulting calculation currency column, year column, and arguments concerning the resulting column location and its name. This input allows the function to look up exchange rates for currencies from a given year from the main data file and apply these rates to execute the function calculation. The results are then returned to the user as requested.

**Design Decisions**

The design decisions made for this Python package centered on user experience and ease of use. One major design decision was to create Country and Money classes that would improve the overall user experience. The creation of these classes allowed for the assignment of different identifier types, which could be parsed by the user easily. As identifiers for countries and currencies can be provided in so many different formats, it was clear a decision needed to be made about what input would be acceptable. These classes allow the users to preprocess data that includes any standard identifier. While the functions themselves expect alpha 3 code identifiers or standard 3-digit currency code, the users are able to utilize the parsing functions to easily alter other standard identifiers to the expected identifiers for function input. Therefore, the use of these functions requires no additional cleaning outside of this package use, simply a parsing step built-in for the user. This allows so much more flexibility on the part of the user and saves the headaches of cleaning data simply so it can be used by another cleaning tool. Therefore, this design decision worked well for our project, and added to the overall user experience.

Another major design decision focuses on extensibility and potential updates. This centered around the data source feeding our three main functional calculations. For this project, the most realistic decision was to use a series of CSV files containing population values, exchange rates, inflation percentages, etc. While a more complex route could have been taken, it did not seem reasonable within the scope and timeline of this project. The use of the CSV files, however, allowed great flexibility and extensibility potential for the project. CSV files allowed for collection and input of calculation metrics outside of the python script. This means that these files can be updated, changed, or expanded upon without changing the software component. In addition to this flexibility, there is also a clear path for adding metrics as needed for additional populations, inflation rates, etc. that will be needed as time goes on. Therefore, reading in source data this way was a crucial design decision that worked incredibly well within the scope of this project.

**Design Comparisons**

Another package that has similar data cleaning capabilities is the pandas package in Python. This package is much more expansive in functionality than simply data cleaning, but it provides those capabilities as one of its major components. Data cleaning using with pandas happens using a different structure than the international data preprocessing package. In order to use the pandas package, one must convert the data frame itself to a pandas data frame. In this way, pandas has its own data frame type in order to conduct cleaning and preprocessing methods. Some pandas data cleaning functionality includes functions like drop, set\_index, and to\_numeric. Each of these functions simply require a pandas data frame column to be passed in to execute their desired function. These functions conduct more general data cleaning, rather than more specific tasks accomplished using the international data preprocessing package. Therefore, the input for each of these pandas functions are much simpler, as only the column itself is required. As mentioned before, the major difference in design is the way pandas handles the data frame itself. Rather than passing the data frame in at the function level, pandas deals with the data frame in its own class, establishing a pandas data frame type. This difference is indicative of the increased functionality the pandas package provides beyond data cleaning.

**Extensibility**

The design structure of this package acknowledges the need for updates to be made easily and efficiently. Due to the nature of the metrics calculated, it is important to make calculations with the most up-to-date metrics possible. For this reason, this package emphasizes the potential for versioning in its design. The main data source for executing calculations is a maintained data file that is easily updated to provide relevant and accurate calculation metrics. This provides an integrated component that promotes package and data updates.

Beyond this component, the package's potential to provide more functions in relation to international data preprocessing is evident. Conversions such as time-zone adjustments and percentage of global populations are just a few potential functions that could be added to this package in future versions using a similar structure and method. These additions could easily be added using the same process and implementation structure. The ideal extensibility scenario would be to provide users with any and all international data cleaning tools that would provide simple solutions to otherwise tedious and cumbersome avenues.