Data Science Pipeline for Clustering Asian Countries Based on Developmental, Geo-political and Economical data

Course Project - Algorithms, Data Structures and Databases

Àlex Martorell
Enric Reverter
Louis Van Langendonck

Facultat d'Informàtica de Barcelona

2021
Universitat Politècnica de Catalunya

Platform Access Instructions

All data, scripts and results can be accessed via `https://github.com/eReverter/adsdb_project`.

To run the pipeline, navigate to the main wrapper python script called *main.py*. This wrapper runs through all the scripts that are build per zone in one go. The parameters can be tuned to modify the outcome. To know what is happening while executing the pipeline, print statements serving as an auditing method are implemented in most of the main functions supporting the project. However, to further investigate how each zone works, the reader is encouraged to access the python scripts and notebooks, both stored in each zone folder. The `.py` scripts contain all the different functions used in the main pipeline, which are all described using docstrings. The notebooks are just a step wise representation of what happens in that part of the main pipeline.

Moreover, the profiling of the databases as well as the results of the analysis can be accessed in its corresponding folders. Inside the `formatted_zone` folder there is a sub-folder named `profiling`. The different source versions and the integrated sources can be depicted via the `.html` files. Each file contains a detailed profile of the variables in the database. The same type of information can be found in the `exploitation_zone` folder. Note that the factual tables from the schema are profiled in more detail (i.e., the correlation between variables is also depicted). Finally, the analysis results, consisting of interactive cluster plots in *.html*-format and a Principal Components Analysis (PCA) report in *.txt*-format, can be found inside the `cluster_results` sub-folder of the `analysis` folder.

Furthermore, the `environment.yml` contains all the dependencies that need to be installed in order to reproduce the project.

Contents

1 Project Aim & Report Structure

1.1 Project Aim

Although the title for this report could suggest otherwise, the main focus of this project for ADSDB is creating a Data Management Backbone and a Data Analysis Backbone for an array of data sets. The goal is to explore tools and develop good habits for general structuring of Data Science projects. The data analysis consists of clustering Asian countries based on several economical and governmental indicators. However, as explained before, the focus of the data analysis, although very interesting, is put in second place, as this is not the objective of the project.

1.2 Report Structure

The report is generally structured according to the different steps of the project.

Section 2 presents the data sources used throughout the project. Moreover, it briefly postulates the different tools used throughout the pipeline.

The main section of this report is Section 3, which summarizes the Data Management (DM) process step by step. For each zone contained in the DM, an initial summary is provided which lists the names of the files (notebooks or python files) in that zone as well as its most notable functions. Below each summary a more detailed explanation of the files and its functionality is given, in order to bring clarity to the whole process.

In section 4, the main steps for the Data Analysis are shown, as well as some of the important results presented in the form of plots and *txt* files.

Finally, in the conclusions part of the report, Section 5, the group members reflect on some of the difficulties encountered along the development of the Data Management backbone and indicate what aspects of the projects can be improved upon.

2 Data Sources & Tool Choices

2.1 Data sources

Three data sets were considered before building the Data Management backbone:

- World Governance Indicators (WGI): Extensive data set that considers aggregated, numerical indicators of six broad dimensions of governance for each officially recognized country in the world: *Voice and Accountability, Political Stability and Absence of Violence/Terrorism Government Effectiveness, Regulatory Quality, Rule of Law* and *Con-*

trol of Corruption . A *.dta* or *.xlsx* file can be retrieved from: `https://info.worldbank.org/governance/wgi`.

- World Bank Data: An incredibly large Data Bank (`https://data.worldbank.org/`) providing an API to access various geo-political, social, economical and infrastructural markers for all officially recognized countries in the world. Note that these can be filtered by data set or by topic. The latter option is chosen, and indicators from topic 11: *Poverty* and a few more from topic 15: *Social development* will be considered.

- United Nations Data Bank: From here, a data set with just the country names and its continent are retrieved. This can later be used to filter the previous data sets since the analysis part is focused on Asian countries.

2.2 Choice of Tools

Collaboration: The main tool used for collaboration is **Github**, whose repository can be accessed via:
`https://github.com/eReverter/adsdb_project`.

Data Storage: In order to store data in databases, **PgAdmin 4** is used for hosting a **Postgres** database and **DBeaver** is used to access it.

Data Transformation: All transformations and analysis on the data are done using *Python*-scripts and extensive use of open-source packages. Some of the most notable packages are: **psycopg2** and **sqlalchemy** for accessing, loading, and updating the aforementioned databases, **pandas** for all data frame transformations, and **pandas_profiling** for automatic data profiling.

Pipelining: All scripts and transformations are glued together within **python** by the use of wrapper functions.

Data Analysis: Dimension reduction and clustering, used in the data analysis part, mostly relies on the **scikit-learn** package in **python**. Interactive result *.html* plots are generated using the **plotly** package.

Constraint 1. Note that the origin of the three sources fulfills this first constraint.

3 Data Management Backbone

Constraint 2. The naming convention is `zone_name`. Note that some additional files may be found in each of the zone with their corresponding names.

Constraint 3. Since all of the project has been wrapped into a main pipeline, the files found in the different zones are of auxiliary type and contain the functions required for the analysis. They are named: `aux_zonename.py`.

3.1 Landing zone

3.1.1 Summary

- **Available notebooks and files:** `aux_landing.py`, `01_wbd_api.ipynb`, and `02_landing.ipynb`.
- **Important functions:** `move()` and `rename_csv()`
- **Temporal zone:** Contains the data sources that can be downloaded directly (i.e. not using an API).
- **Persistent zone:** Contains the three data sources with its corresponding timestamp.

3.1.2 Methodology

The World Bank source data is retrieved using the API provided. This process is documented in the *01_wbd_api* notebook. Using this notebook, a different selection of topics can easily be chosen such that this process can be considered flexible and scalable. However, to keep it simple, some topics are predefined for the analysis backbone and thus, a *.csv* file is created, containing the chosen indicators for a time span of 20 years (2000-2019).

The core of the landing zone consists of a simple and brief script that moves the files from the temporal zone to the persistent zone using the built-in `shutil` package whilst automatically adding a timestamp. The renaming of files is sustained by the `rename_csv` function. This function takes a file, an optional new name, and file path and adds the current date to the end of the file name. For example, the file obtained from the WBD is automatically moved into the persistent folder and renamed to wbd_*timestamp*.

3.2 Formatted zone

3.2.1 Summary

- **Available notebooks and files:** `aux_formatted.py`, `01_to_formatted.ipynb`, and `02_data_quality.ipynb`.
- **Important functions:** `tables_to_load()`, `load_database()`, and `outliers_duplicated_profiling()`.
- **Other folders:** `profiling`.

3.2.2 Methodology

In the formatted zone a connection to a local database is initiated. This is where the source tables are stored. Note that the user is expected to have access to and/or host an operational *PostgreSQL* database. The `tables_to_load` function acts as the wrapper that creates and fills the different tables in the database. It relies mostly on another custom method called `load_database`. This checks the file path and type (.xlsx, .dta, .csv), reads its content into a data frame, and creates a database table accordingly. The pandas `df.to_sql` then bulk loads that table using `sqlalchemy` tools, where we also specify the method used to load it, called `psql_insert_copy`. This method allows for a fast and efficient insertion of the data into a *PostgreSQL* database. The wrapper function loops through the persistent zone files and then checks for existing tables. If the table already exists, it is not loaded unless specified in the arguments. Note that only the different versions of a source are loaded at this stage.

An important aspect in this zone is profiling the different variables. As such, data quality processes are performed. Therefore, `outliers_duplicated_profiling()` acts as a wrapper function that generates *.html* files for each table (using the `pandas-profiling` package), which contain an extensive profiling report for each column. Moreover, the wrapper function method allows for some basic data cleaning. The number of extreme outliers and duplicates are checked and are handled according to the parameters chosen by the user. That is, extreme outliers can either be kept, removed, or imputed as missing. Then, duplicated rows and columns can also be either kept or removed.

3.3 Trusted zone

3.3.1 Summary

- **Available notebooks and files:** `aux_trusted.py` and `01_trusted_integration.ipynb`.
- **Important functions:** `integrate_source_versions()`.

3.3.2 Methodology

In the trusted zone, the different versions of a source are integrated into a single table. This is done using the `integrate_source_versions` function. It takes as arguments the source to be considered and the url to the database containing the tables to be integrated. It then checks for table names which have the so-called source name incorporated into their version name. If there is a match, each different version from the database if read after which they are concatenated into a single data frame while deleting duplicates. After that, it is bulk loaded into a new table using the `psql_insert_copy` method described earlier. The name of the table can be specified in the arguments, otherwise the source name is used for it. It is worth to mention that the different versions are integrated even if the number of columns differs. If a version is missing some variables its rows will just contain missing values. Notice that at this point, both the integrated source and its different versions are allocated into the same database.

3.4 Exploitation zone

3.4.1 Summary

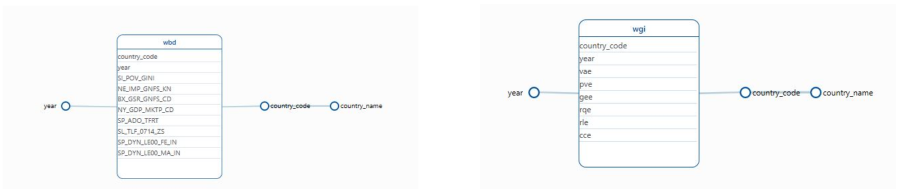
- **Available notebooks and files:** `01_exploitation`, `aux_exploitation.py`
- **Important functions:** `populate_facts()`

3.4.2 Methodology

In the last step of the Data Management Backbone, the final schema where the data has to be inserted is considered. The schema has to be provided by the user in a .sql file. That allows for a higher flexibility on the definition of the exploitation zone. In this particular case, the schema contains factual tables about WGI and WBD indicators and dimension tables about the years and countries. Figure 3.1 displays the final schema. Both years and countries dimensions are shared between the fact tables. Then, the tables are populated once the schema has been generated. It is worth to mention that until this point, all the steps can be automatically reproduced for any data. However, to integrate and transform the dimensions, some specific transformations to the analyzed data is needed. First, the dimensions are populated to avoid violation constraints in the relations. Then, the facts are populated.

Regarding the dimensions, the years dimension is populated without issue, but the countries dimension has a problem. The country coding of WBD and WGI differs from the standard ISO 3166 [1]. For example, according to the ISO 3166 standards Andorra should be encoded into *AND*, but the WBD data encodes it as *ADO*. That is, entity resolution is needed. Otherwise, the relations between facts and dimensions cannot be properly defined. As such, two columns are added into the countries dimension in order to map the different sources. This is done using the Levenshtein distance since for this particular problem is more suitable than other similarity metrics such as Jaccard distance. Nevertheless, some codes are still mismatched and thus, this mappings should be done manually. For this reason, the mapping columns are kept into the dimensions as a proof of concept, but the relations between primary keys and foreign keys are not defined in the schema. This is all computed using the `populate_dimensions` wrapper function.

Regarding the facts, the `populate_facts` wrapper function is used. It automatically loops through the schema facts and the source tables until it finds a match. Then, since the schema table might contain less columns, it cross-checks this with the source table and only loads the subset of variables. In this particular case, there are no constraints set to the rows, but this should be taken into account as well. According to the documentation, the method `on_conflict_do_nothing` from `sqlalchemy` allows to do that [2]. It checks if some rows are violating some constraints and skips them when loading into the tables. Thus, this is the method with which data is inserted into the schema.



(a) WBD fact (b) WGI fact

Figure 3.1: Final Schema visualization using the Indyco Tool.

Constraint 4. Data Quality and Profiling results can be found in separate folders which contain `.html` files.

Constraint 5. The *.html* quality reports describe all the created tables extensively.

4 Data Analysis Backbone

4.1 Summary

In this project, analysis is kept to simple methods, which however manage to yield remarkable results. It seems clear that many interesting conclusions could be retrieved for a further study. However, in this case, we reduce this to a simple preprocessing (imputation of NA's, disregarding duplication of dimensions) and clustering (Dimensionality reduction and k-means clustering). Plots are outputted in a *.html* page displaying the cluster results and a *.txt*-file containing cluster info, meta-data and PCA profiling.

Constraint 6. No train and test separation is needed in the Data Analysis Backbone, because the algorithm of choice is K-means Clustering, which is an unsupervised learning method and thus, the results are not tested.

Constraint 7. As the results of clustering are best stored as a cluster plot instead of a data frame, it is chosen to only store the generated reports (both *.html* and *.txt* files).

4.2 Methodology

The final schema is queried for joining by time dimension and country code. Note that there are six indicators of WGI type (six broad categories) and of WBD type (very specific and related to a given topic). A relation illustrating the meaning of each indicator is found in Table 1 in the Appendix.

As it has just been mentioned, the core of Data Analysis Backbone consists of two important functions: `preprocess` and `cluster`. `preprocess` first removes rows that only contain NA's and imputes the others. The function then standardizes the data after which the data is organized into different data frames, organized per year, predefined by the user as a hyper parameter. The `cluster` function uses this list of data frames as input. For each of them, it reduces the dimensions to 2 using PCA after which it applies a k-means clustering algorithm for which the user specifies the desired amount of clusters. The cluster plots are stored in a interactive *.html*-file and a profiling of the dimensions and clusters is exported to a *.txt*-file. An example of the cluster results are found in the Appendix, in 5.1

Constraint 8. All the functions are found in separate auxiliary files in each zone, and a `main.py` which acts as the **Operations** file runs the whole pipeline while outputting its progress.

Constraint 9. Both Data Management and Data Analysis backbones are automatized up to the point where loading new data sources would not imply any major changes in code. Adding more variables into an existing table is not a problem either since the said tables are integrated in the trusted zone.

5 Conclusions

We must acknowledge some of the particularities that this Data Science project has with respect to the other ones that we have done along this first semester. In some of our work we completed a thorough data analysis step, but the data management aspect of the job - introduced with all detail in this project - was rather scarce; one data frame was usually enough to contain all the data needed for the analysis. Nevertheless, we were now confronted with a bigger problem: what if we use data from many sources? Furthermore, what if we have different versions of data coming from the same source?. With the scope now widened, our knowledge fell short, and the Data Management aspect became the central part of the project.

Having many different tools for Data Management available, we chose the ones that we were more familiar with. Since it is considered a bad practice not to store our data in a database when we have many sources and possible updates in our data, we had to learn how to combine tools such as

Pandas in Python for the reading and exploration of data with the SQL querying language. Tools like psycopg2 and sqlalchemy came in handy to manage the intricacies and complexities of reading from, bulk-loading, and inserting data to an SQL table. Combining different environments is always hard, and a considerable amount of time was put into solving some of the cul-de-sac situations we encountered. The goal was to automatise each process as much as possible, as it is a good practice to pipeline all the operations we do. Many of the obstacles have already been mentioned, but for the sake of this conclusion, let us recall one. When loading the final schema, we are inserting row by row into a given table, the methods provided by PostgreSQL database adapter psycopg2 were not enough, and made it impossible to automatize the loading procedure. However, extensions from the SQLalchemy toolkit made automatization of the insert statement feasible. In other words, by constant trial and error, we were able to come out with a final pipeline, and track its execution. The final output (in text) is shown in the Appendix.

Being a First Semester Masters project and our first attempt to scrupulously and coherently design a Data Management Backbone, as we were coding, we came to realize that many improvements were possible. Most of them have been mentioned in the sections above. For example, wrong coding of the country names (i.e. different codes according to the source) caused a violation of the primary-foreign key constraint in the final schema. Entity resolution methods fell short (c.f. Section 3.4) in solving the issue. A solution would call for manual one to one mapping, but it would clearly compromise the automatization of the Data Management. Hence, the final product is not considered a definitive solution to all possible problems at hand but can surely serve as a scale-able backbone to further build on in the following semesters.

References

[1] ISO 3166 Maintenance Agency. Iso 3166 country codes.

[2] SQLAlchemy. Sqlalchemy 1.4 documentation, 2022.

Appendix

WBD Indicators	
SI.POV.GINI	Gini index (World Bank estimate)
NE.IMP.GNFS.KN	Imports of goods and services (constant LCU)
BX.GSR.GNFS.CD	Exports of goods and services (BoP, current US $)
NY.GDP.MKTP.CD	GDP (current US$)
SP.ADO.TFRT	Adolescent fertility rate (births per 1,000 women ages 15-19)
SP.DYN.LE00.FE.IN	Life expectancy at birth, female (years)
SP.DYN.LE00.MA.IN	Life expectancy at birth, male (years)
SL.TLF.0714.MA.ZS	Children in employment, male (% of male children ages 7-14)
WGI Indicators	
vae	Voice and Accountability index Estimate
pve	Political Stability and No Violence index Estimate
qee	Goverment Efectiveness index Estimate
rqe	Regulatory quality index Estimate
rle	Rule of law index estimate
cce	Control of corruption index estimate

Table 1: Detailed description of each indicator used

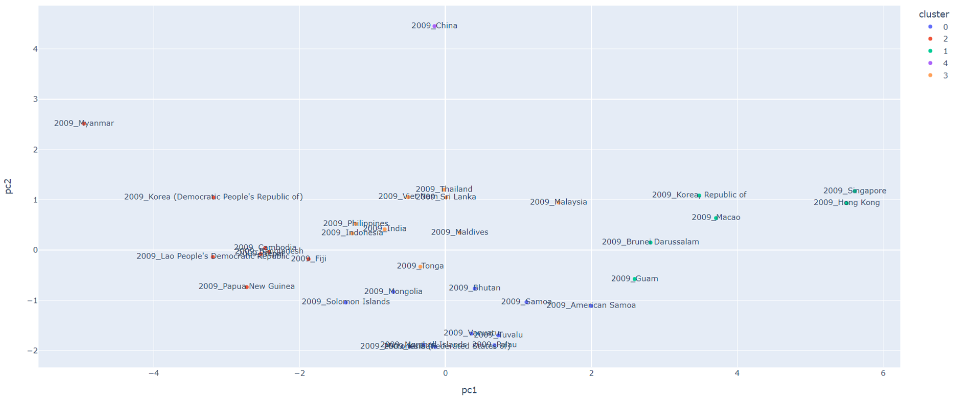


Figure 5.1: *.html* clustering plot output example.

```
############################## Landing Zone ##############################

countries.csv moved to persistent and renamed to countries_20220123.csv and timestamped.

############################## Formatted Zone ##############################

countries_20220123 succesfully loaded into the database.
countries_20220123.csv succesfully loaded the specified tables.
wbd_20220122 succesfully loaded into the database.
wbd_20220122.csv succesfully loaded the specified tables.
wgi_20220122 succesfully loaded into the database.
wgi_20220122.dta succesfully loaded the specified tables.
Quality report for wbd_20220122 already exists and is not replaced.
Quality report for wgi_20220122 already exists and is not replaced.
Quality report for wbd already exists and is not replaced.
Quality report for wgi already exists and is not replaced.
Quality report for countries_dim already exists and is not replaced.
Summarize dataset: 100%|
Generate report structure: 100%|
Render HTML: 100%|
Export report to file: 100%|
Quality report for countries_20220123 succesfully generated.

############################## Trusted Zone ##############################

Tables from wbd integrated into one table.
Single source table succesfully loaded into database.
Tables from wgi integrated into one table.
Single source table succesfully loaded into database.
Tables from countries integrated into one table.
Single source table succesfully loaded into database.

############################## Exploitation Zone ##############################

./schemas/conflict_schema.sql succesfully created into the database.
database dimensions succesfully populated.
database facts succesfully populated.
Quality report for years_dim already exists and is not replaced.
Quality report for countries_dim already exists and is not replaced.
Quality report for wgi already exists and is not replaced.
Quality report for wbd already exists and is not replaced.

############################## Analysis ##############################

PCA is finished.
Data has been clustered.
The reports have been generated and exported.
```

Figure 5.2: Output displayed trough the runtime of the operations.