Data Cleaning, Automated Gathering and Analysis of European Parliament Voting Data

By **Franciszek Swiderski**

B. Sc. International Politics and Government, Bocconi University

September 2024

1 Introduction

The European Parliament voting data spanning the 6th to the 9th legislatures was inherited by the Institute for European Policy at Bocconi from the VoteWatch project, initially developed by Abdoul Noury and Simon Hix. To ensure the continuity and usability of this dataset, significant efforts were required in data cleaning, reformatting, and supplementation, followed by the automation of data gathering processes to facilitate ongoing maintenance. This dissertation details the systematic approach undertaken to clean and restructure the data, the challenges encountered, and the subsequent automation using the European Parliament's API, as well as the analysis conducted on using the data.

1.1 Dataset Structure and Initial Challenges

The dataset comprises three primary components:

- MepInfo: This dataset contains information on Members of the European Parliament (MEPs).
- Votes: This dataset records the votes cast by MEPs on specific pieces of legislation.
- Votings: This dataset describes the legislative items that were voted upon.

The Votes dataset alone includes approximately 42,800 roll-calls, with 800-900 MEPs participating in each legislature, resulting in an estimated 40 million data points. Upon initial examination, several issues with the legacy data became apparent:

- Missing Variables: The MepInfo dataset lacked several critical variables, including gender and age.
- Inconsistencies and Missing Observations: Some variables such as Party affiliation and European Parliament Group (EPG) affiliation were either inconsistent or incomplete.
- Non-Standard MEP IDs in EP7: MEP IDs in the 7th Parliament did not conform to the standard European Parliament API format. Instead of unique identifiers, incrementing integers were used.
- Inconsistent Naming Conventions: MEP names and surnames were not consistent with those in the European Parliament API.
- Incorrect and Inconsistent Encoding of Missing Data: Missing data was inconsistently encoded, with symbols like - or . being used in place of null values.
- Inconsistent Datetime Formatting: Datetime values were formatted inconsistently across the dataset, making them difficult to import and process.

- Non-Standard Encoding of Binary Categorical Variables: For example, the binary categorical variable "Vote," which indicates whether a vote passed or not, was encoded as + and - instead of the standard 1/0 used elsewhere in the dataset.

Additionally, the dataset was stored in a long format tailored to the specific analytical requirements of the original VoteWatch team. While this structure was suited to their immediate needs, it posed challenges for long-term usability, warehousing, and broader analytical purposes.

2 Data Cleaning Process

2.1 Supplementing and Correcting the MepInfo Dataset

The initial challenge involved supplementing and correcting the MepInfo dataset to fill in missing variables and rectify inconsistencies. The European Parliament's API played a crucial role in this process. An API (Application Programming Interface) is a set of defined rules and protocols that allows one software application to interact with another. In the context of scraping data, an API serves as an intermediary that lets users request and retrieve specific data from a website or service in a structured format, typically JSON or XML, without the need to manually scrape the HTML content of a webpage. Using an API for data scraping is often more efficient and reliable than traditional web scraping, as it provides direct access to the desired data, reducing the risk of encountering issues like changes in webpage structure or content restrictions. This API provides endpoints, such as '/meps', which return JSON data containing basic MEP information, including a unique identifier (MepId). By utilizing this API, we were able to standardize the names and identifiers of MEPs across most legislatures by joining the tables by MepId.

However, the 7th Parliament (EP7) presented a unique challenge. The MEP IDs in this legislature were not unique and were instead represented by incrementing integers. To address this issue, I employed the 'fuzzywuzzy' Python package, which uses the Levenshtein distance algorithm to calculate the similarity between strings. This allowed for making an approximate match of full names from the original dataset to the 'sortLabel' field in the API data, providing the correct MEP IDs. Manual verification and correction of edge cases were necessary to ensure accuracy.

Despite these efforts, some data remained incomplete, particularly regarding MEPs who changed parties or EPGs during their tenure, as well as demographic data such as gender and birth dates. These gaps required further manual supplementation, which is discussed in detail in the Automation section.

2.2 Addressing Inconsistencies in the Votings Dataset

The Votings dataset required significant work to address encoding inconsistencies and improperly formatted datetime values. The first issue was relatively

easy to reconcile using renaming dictionaries to replace inconsistent encodings in some variables.

However, the latter issue proved particularly problematic due to the original data being gathered in Excel. Manual formatting likely led to inconsistent datetime values formatting. To standardize the format, manual correction of the dates was necessary before applying the Pandas 'to datetime' function.

This step ensured that the datetime values were correctly parsed and made the data suitable for further analysis. Once the initial cleaning was complete, a new challenge arose. In 2022, the original VoteWatch team altered their data gathering methodology. Consequently, data from Parliament 9, covering the period until 2022, was consistent with earlier practices. However, data collected from 2022 to March 2024 exhibited several discrepancies. The order of VoteIds had changed, some columns particularly those in the Votings dataset were missing, and special characters were corrupted due to encoding issues. To resolve these issues, I employed a multi-step approach. First, I used a dictionary to replace the corrupted characters systematically. Next, I reverse-engineered the missing Votings columns (such as finalVote, a binary variable of whether a vote is a last one in a section) from the available data. Despite the complexity of these tasks, they were necessary to restore the integrity and consistency of the dataset, ensuring that it could be seamlessly integrated with the existing data from earlier legislative periods.

2.3 Data Restructuring for Long-Term Usability

With the Votes and MepInfo datasets cleaned and supplemented, the next step was to restructure the data into a format suitable for long-term storage, analysis, and future updates. The data was initially stored in a long format, where each row represented an observation, and columns represented variables. To facilitate analysis, I separated the MepInfo and Votes datasets into distinct tables:

- MepInfo table contained all variables describing the MEPs, with 'MepId' serving as the primary key.
- The Votes table used a composite primary key consisting of 'MepId' and 'VoteId', a unique identifier for each voting event. This table also included a column encoding the MEPs' votes.

These tables were linked by primary keys, allowing for efficient querying and data retrieval. For example, using the 'MepId', one could easily retrieve all votes cast by a particular MEP, and by further referencing the 'VoteId', additional context from the Votings table could be added.

To support the development of the European Parliament Vote Monitor website, the voting data was additionally exported in '.csv' format, organized by month and year. This format was chosen to optimize data storage and accessibility, ensuring that the data could be easily updated and queried. Additionally, for specialized analytical purposes, the Votes data was also retained in its original matrix format, now enhanced with the newly cleaned and supplemented variables.

3 Automation of Data Gathering

3.1 Overview of the European Parliament API

With the historical data cleaned and restructured, the next phase of the project focused on automating the future data gathering process. Consistency and reliability were paramount in this process. The previous VoteWatch team relied on scraping XML files from the human-readable official minutes on the European Parliament's website, coupled with downloading MepInfo from the API. This approach was fraught with challenges, including the potential for inconsistencies in the scraped data, lack of unique identifiers, and the inherent unreliability of scraping human-readable content.

To address these issues, we transitioned to using the European Parliament's open API, which provides direct access to structured data in a reliable and consistent manner. The API is publicly accessible, meaning no API key is required, and it allows users to specify parameters in the URI to retrieve data from various endpoints.

3.2 MepInfo Data Collection Automation

The '/meps' endpoint of the API, which provides a list of MEPs along with their unique identifiers, was the starting point for automating the MepInfo data collection. However, the data returned by this endpoint was not exhaustive. To gather more detailed information such as MEPs' gender, age, and political affiliations additional API calls were required to other endpoints within the 'meps' group.

For example, to retrieve data on MEPs' party memberships and affiliations with European Political Groups (EPGs) and National Parties, separate API calls were made specyfing the MepId . The resulting data was then filtered and joined with the MepInfo dataset to create a comprehensive record of each MEP s political affiliations and demographic information.

One key challenge in this process was optimizing the API calls to minimize the load on the server. Each MEP required an individual API call to gather detailed data, which could result in approximately 850 GET requests per scraping session. To mitigate this, the metadata was included in the same call to avoid the need for additional requests.

Another consideration was the use of non-human-readable IDs for certain values, such as the organization field, which required further API calls to retrieve the corresponding labels from the 'corporate-bodies' endpoint. For example org/1537 was an identifier for GUE/NGL EU Political Group. These labels were essential for understanding the data and were subsequently joined with the MEPs' political affiliation data.

3.3 Votings Data Collection Automation

The ‘meetings‘ endpoint of the API was utilized to automate the collection of data on plenary sessions and the decisions made during these sessions. An initial API call returned a list of all plenary sessions that took place in a given year. This list was then filtered by month to identify the specific sessions relevant to the current data collection period.

For each identified session, further API calls were made to retrieve detailed voting data, such as the votes cast by each individual member and all of the supplementary data that was previously found in the Votings database. This data was then merged with the previously gathered MepInfo and Votes data, ensuring that all relevant information was captured and linked across the collection.

In order to automatically gather the data in the future, working with a Bocconi IT Department team, we created a cloud-based solution that is deployed and triggers every month, gathering and exporting the data to the .csv format, as well as to a relational SQL database.

4 Analysis

4.1 Ideal Points Estimation workflow reproduction