MIT Election Data and Science Lab
Hiring Assignment - Data Analysis Exercise

Juan Cruz Ferreyra
February 16, 2026

Part 1: Data Wrangling

1.1 Data Cleaning

The cleaned dataset (part1_clean.csv) has been produced in accordance with the codebook specifications. Key transformations include:
- Standardized office names (US PRESIDENT & VICE PRESIDENT → US PRESIDENT)
- Mapped party names to standardized format (DEMOCRATIC → DEMOCRAT)
- Obtained the year from the Election string field using Regex
- Aggregated individual write-in candidates to WRITE-IN
- Merged county FIPS codes from reference file
- Converted all text fields to uppercase where required
- Set all constant fields (stage, state, special, date)

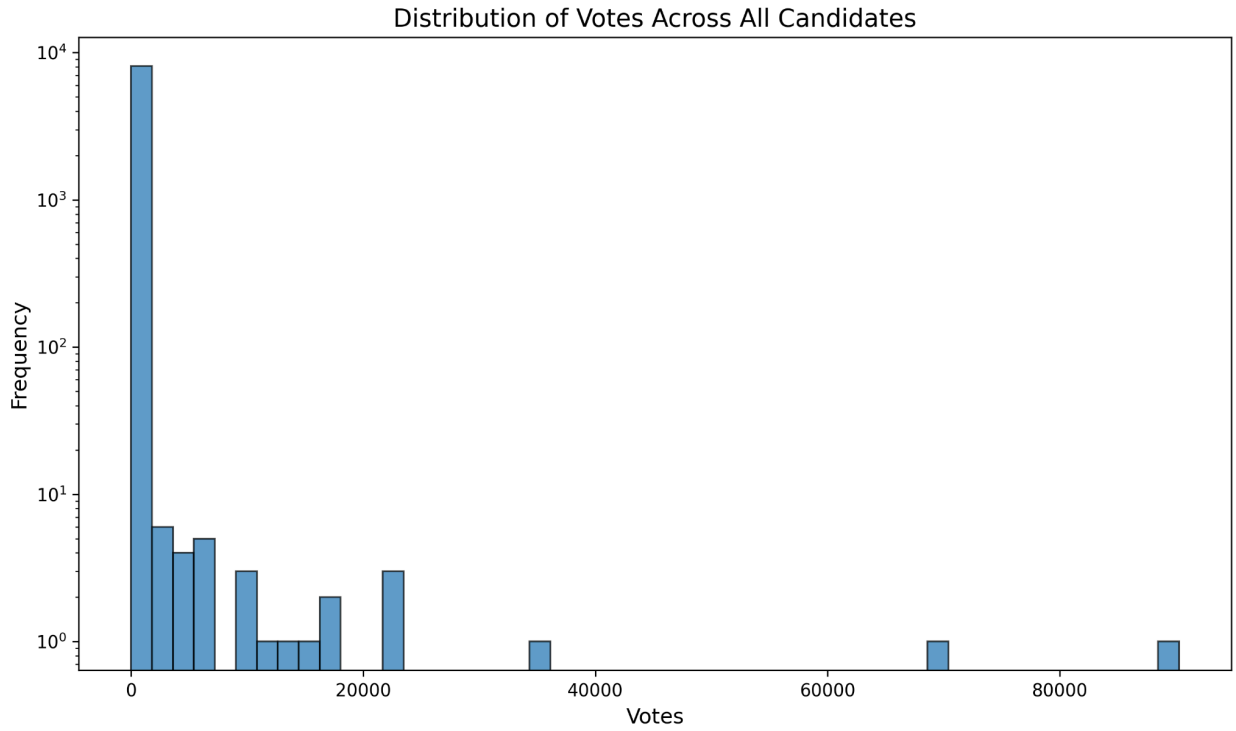
1.2 I found three issues during the data cleaning step:

1. Incorrect format in one of the records (row 323). This record has two extra empty fields at the beginning, so it cannot be easily read by pandas. I found it because the pandas ParserError traceback indicated the row number, and I inspected the raw CSV. I implemented a quick solution using LLM support to use the default package csv to read every raw row in the file, check the number of fields, and if the number of fields is higher than the number of columns, then we perform the check for the leading commas (extra empty fields). A more robust approach would imply searching not only for leading empty fields, but for empty fields in the whole row; this might be problematic if we have more empty fields than extra fields in the row (so some empty fields should be read as valid NA values). In that case, a better approach would be to print a warning to the console so the researcher can check it manually and extend the fix step to new formats.
2. The second error found is related to character encoding in candidate names. Particularly candidate André, which was displayed as "AndrÃ©". I found it by visual inspection of the candidate column output. I implemented a quick fix by directly replacing the error character 'Ã©' with 'E' during the candidate standardization step. A more robust long-term fix would imply ensuring the source file is properly encoded as UTF-8 when received from election offices. Alternatively, we could parse the file explicitly as Latin-1 encoding when loading, then map all accented characters to their unaccented counterparts (e.g., é → E, á → A) before uppercasing, which would handle all cases rather than addressing specific character errors individually.
3. The last error found is the presence of negative values in the votes column. It was found during exploratory data analysis when printing the summary of the column. The fix was

simply setting it to 0. Another approach would be setting it to the absolute value (e.g., -8 to 8), but that implies some assumptions, like assuming that during manual data entry, there was a problem and the operator misclicked the minus sign.

4. The fourth error found is an implausible vote count in the votes column. One record shows 999,999 votes for Trump in Riverside-North precinct (Cedar County). This makes the precinct total exceed 1 million votes, while other Cedar County precincts range from 0 to 1,156 votes. It was found during exploratory data analysis when checking the maximum value of votes. The value 999,999 is likely a data placeholder for missing or unknown data rather than an actual vote count. I did not implement a fix for this as it requires verification with the data provider. A potential approach would be to filter out records where votes equal 999,999 or exceed some threshold (e.g., > 50,000 for precinct-level data), but this assumes the value is indeed a placeholder and not a legitimate county-level aggregation that was mislabeled as a precinct.

1.3 The outlier value 999,999 was filtered for the following plot (see 1.2.4):



1.4 Both the raw and cleaned data are in long format with a single column for the votes. Wide format would be if we pivot the dataframe having one column per category (e.g., one column per candidate, or party) with the number of votes as values, indexed by precinct and county. For example, wide format would have columns like 'DONALD J TRUMP', 'KAMALA D HARRIS', etc., with each row representing a precinct and cells containing vote counts for each candidate.

Part 2: Thinking

Example variables to extract:

- Proposition ID (133-140)
- Proposition official title
- Proposition descriptive title
- Legislative Council analysis text
- Arguments FOR text
- Arguments AGAINST text
- Economic analysis (when present)
- Page counts for each section

Derived metrics:

- Text length (word count) for FOR vs. AGAINST arguments
- Number of FOR arguments vs AGAINST arguments
- Presence of fiscal analysis (binary)

Possible to measure: Balance of information (FOR vs AGAINST word counts), Relationship between argument length and voter outcomes.

Dataset structure: One record per proposition with the following fields:

(prop_id, official_title, descriptive_title, analysis_text, for_text, against_text, economic_analysis, for_word_count, against_word_count)

Part 3: Reasoning

The code drops a row based on the fipscode (line 4), and then replaces values in various columns based on the value in fipscode (or in state_abbv + fipscode combined in lines 66-70).

Improvements: First, the lack of comments explaining what the code is doing and why it is changing those values. Second, I don't program in Stata, but if doing this in Python or R I would perform a loop to correct those values by passing a dictionary or lookup table with the fipscode as key and correction values as values (e.g., {fipscode: {column: new_value}}). This would replace the 54+ repetitive replace statements with a few lines of code. Probably the inner functionality would be encapsulated in a function.