NONE 2. NONE 3.

The goal of this project is to create a collection of nodes based on survey data from a New York House of Representatives voting precinct. Using Rust programming language, the project handles data loading, graph construction, and analysis of voting patterns. The project successfully visualizes relationships and voting behaviors across different precincts, identifying key candidates and summarizing their support base.

Introduction

This project arises from the need to understand detailed voting patterns within precincts for the New York House of Representatives elections. By analyzing this data, we aim to identify how precincts align in terms of candidate support, which could be crucial for strategic planning in political campaigns and understanding voter behavior.

Dataset Description

The dataset used comprises precinct-level returns from the New York House of Representatives elections, including details such as precinct identifiers, candidate names, and vote counts. The data encompasses over 836,424 records, ensuring a robust basis for constructing a meaningful graph of precinct relationships.

Methodology

The project is implemented in Rust, utilizing modular programming principles. Data is loaded and parsed from a CSV file using custom deserialization to handle inconsistencies in data formatting. The graph is constructed to represent precincts as nodes, with edges indicating shared top candidate preferences, using the petgraph library.

Implementation

The implementation spans several modules:

NONE 2. NONE 3.

- **data_loading.rs**: Handles the reading and parsing of CSV data into usable structures.
- **graph_construction.rs**: Constructs an undirected graph where nodes represent precincts. Edges are added based on shared voting patterns.
- **analysis.rs**: Analyzes the graph to provide insights such as the top ten candidates by total votes and a summarized degree distribution of the graph.

The code effectively uses Rust's features like enums, structs, and iterators, facilitating efficient data handling and graph operations.

Testing

The project includes unit tests that verify the correctness of data loading, graph construction, and analysis functionalities. These tests ensure that the data pipeline works as expected and that the graph accurately reflects the dataset.

Results

The constructed graph included over 1,000 nodes, satisfying the project's scale requirements. The analysis revealed the top ten candidates by total votes, providing insights into the most influential figures across the precincts. The degree distribution indicated a varied level of connectivity, reflecting diverse voting alliances and patterns.

Conclusion

NONE 2. NONE 3.

This project achieved its objective of analyzing voting patterns in a structured and insightful manner. Future work could expand on this by incorporating more detailed demographic data for deeper insights or applying machine learning techniques to predict voting outcomes based on trends.