CS109 Lab: Polarized Politics[*]

Denison University

Fall 2024

1 Overview

The legislative branch of the United States government, with its two-party system, goes through phases in which the two parties work together productively to pass laws, and other phases in which partisan lawmaking is closer to the norm. In this project, we will use data available on the website of the Clerk of the U.S. House of Representatives (`https://clerk.house.gov/Votes`) to analyze the voting behavior and polarization of the U.S. House over time.

1.1 Learning Objectives

As part of this project, you will learn to read data from files, use string manipulation operations to extract meaningful data from structured data (here, these will be XML files, which we will describe later), and use what you have learned about conditionals and boolean expressions to analyze, visualize, and answer questions about real-world data.

The core intellectual question at the heart of this project is computing the extent to which the U.S. House of Representatives tends to conduct *party line* votes, suggesting political polarization.

2 Part 1: Party Line Vote

A vote is called a *party line* vote if more than half of the voting members of one party vote in one way, and more than half of the voting members of the other party **vote the other way**. For example, one case of a party-line vote is if more than half of the Democrats voted yes and more than half of the Republicans voted no. Note that if either pary's votes are evenly split (a tie), this cannot, by definition, be a party-line vote, including cases where nobody in the party records a vote on a particular bill (yes, this is possible).

In the House (of Representatives), when the vote of every member is recorded, it is called a *roll call* vote. Results of individual roll call votes are available online, which we will explore later.

We have included an example data file called `aca.xml` which contains the roll-call for the last vote on the Affordable Care Act in 2010 (a landmark bill about the US healthcare system).

The format for that file is called XML (short for extensible markup language). In XML, data elements are enclosed in matching pairs of tags. In this file, all of the data for this particular vote is enclosed in a pair of `<vote-data>` ⋯ `</vote-data>` tags, beginning on line 56. The lines at the beginning of the file are metadata that store information like the name and date of the vote. Between the `<vote-data>` ⋯ `</vote-data>` tags, each vote is enclosed in a pair of `<recorded-vote>` ⋯ `</recorded-vote>` tags, like these (shortened considerably):

```
<recorded-vote><legislator name-id="A000022" ··· </recorded-vote>
<recorded-vote><legislator name-id="A000055" ··· </recorded-vote>
<recorded-vote><legislator name-id="A000364" ··· </recorded-vote>
```

[*]Based on Project 6.1 from the textbook: `http://www.discoveringcs.net/6_text_documents_and_dna/project6.1.pdf`

You should study the file format carefully to determine where the actual votes are cast within these lines and think about how you might extract them, using what you know about working with strings.

Your job will be to write a program that counts the number of affirmative and dissenting votes from Republicans and Democrats in a given roll-call and determines whether the results were a party line vote or not.

2.1 Determining if the Affordable Care Act was a Party-Line Vote

To do this, you should write a function called `countVotesByParty(lines)` to count the number of Yes/No votes by party for a given roll call vote (provided within `lines`). You should implement this function inside the file `utilities.py`.

The function `countVotesByParty(lines)` should take as input an XML file that has been **already been converted** into a list of strings, one for each line (Hint: the file `readlines()` function will read a file as a list of lines for you). It should return a list containing, in order, the number of Democrats who voted for the bill, the number of Democrats who voted against the bill, the number of Republicans who voted for the bill, and the number of Republicans who voted against the bill. **Do not change this order!**

Use only the recorded vote elements, and no other parts of the XML files. You may use any built-in string methods that we covered in class, e.g., `find()`, but **you may not use any external libraries for parsing XML files**.

Be aware: some votes are recorded as Yea/Nay and some votes are recorded as Aye/No. Do not count votes that are recorded in any other way (e.g., do not count votes recorded as "Present" or "Not Voting"), even towards the total number of votes. Test your function thoroughly before moving to the next step.

In order to test your function, write a program in `acaPolarization.py` that opens the file `aca.xml` , converts it to a list of strings, and calls `countVotesByParty` on that list. Note that you will have to import your `countVotesByParty` function from the `utilties` module.

At this point, all of the programs and functions that you write should follow the style and structure best practices we have emphasized in class: good function decomposition, self-documenting code, file and function docstrings, avoiding global variables, encapsulating logic in main functions, etc. I will be looking for this as I grade your project!

You can find the results for any roll-call vote by going to `https://www.congress.gov/roll-call-votes`, clicking on the appropriates session, and searching for the vote number. The ACA vote was vote 194 in the second session of the 111th Congress. Compare the results you are getting to the results recorded there.

Once you have written `countVotesByParty(lines)`, you will write another function, `isPartyLine(demYes, demNo, repYes, repNo)` that determines whether, given the number of Democrats and Republicans who voted for and against a bill, whether the vote was a party-line vote or not. This should also go in `utilities.py`. Hint: think about input validation for this function–are there potential vote totals that could cause an error in how you calculate whether the vote was party-line?

Modify your `acaPolarization.py` to have a function, `isACAPolarized()` that loads the `aca.xml` file, converts it to a list of strings, and calls `countVotesByParty` and `isPartyLine` to determine whether the ACA vote was polarized or not, printing out the result. Your program should follow the standard style and structure guidelines that we have followed in this class.

2.2 Applying Your Logic to a Different Vote

Now that you've analyzed the provided `aca.xml` file, you will write a program, in `explorePolarization.py` that downloads the XML file for a different roll call vote, of your choice, and analyzes whether it is polarized or not.

Individual roll call votes are available online at URLs that look like the following:

```
https://clerk.house.gov/cgi-bin/vote.asp?year=<year>&rollnumber=<number>
```

We have provided you with a function called `getVoteFileFromWeb(url)` in `utilities.py` that, given a properly formed URL of the above type, returns a list of the lines (each line being a string) of the file representing the roll call.

Use this function, along with the ones that you wrote in the last section, to download and analyze whether a different roll call vote, of your choice, was a party-line vote or not. Implement this in a function called `isMyChoicePolarized`.

2.3 Analyzing Congress's Polarization Over Time

While it is interesting to look at individual votes, it is more informative to examine trends in how Congress has voted over time. To do this, you will write the program `polarization.py`, which will track and plot the fraction of roll call votes that were party-line votes every year for the past 20 years (2004 to 2023).

You should write at least two functions as a part of this program:

1. `countPartyLine(year, maxNumber)` will take a year and the maximum number of roll-calls to count for that year, and return the fraction of those roll-calls that were party-line votes The roll-call votes each year start at number 1, and we will cap the number of votes at 450. Thus, `countPartyLine(2004, 450)` should return the fraction of roll-call votes between Vote 1 and Vote 450, inclusive, that were party-line votes.

2. `plotPartyLine()` should plot, for years 2004-2023 inclusive, the fraction of party-line votes each year (with a max of 450 votes from each year). It should also write a file called `partylines.txt` containing, on each line, the fraction of votes that were party-line votes that year. A sample (incorrect) output file has been provided for you.

3 Part 2: State Divides

We can use the roll call votes to infer other useful statistics/information. For example, for each representative, in addition to their party affiliation, you can see their state. Each state is allocated a certain number of representatives that can be Republican or Democrat. Another question that we might want to answer is how the distribution of representatives between parties changes over time in a specific state.

Implement the function `stateDivide(state)` that plots, for the last 20 years (i.e., 2004–2023, inclusive) the number of Democratic representatives, and on the same plot, adds a second curve to indicate the number of Republican representatives over the same time period in a specific state. You may include additional curves for other parties if you wish. Make sure to test your function on several states.

To count the number of representatives from each party, use the first roll-call vote for each year. Do not use other roll-call votes other than the first one for each year for the purposes of this project.

The state input should be a string containing the abbreviation for the state you wish to analyze. For your convenience, we've included a list of state abbreviations in `utilities.py` called `STATES` (the reason the variable is capitalized is to indicate that we view it as a *constant*, i.e., it is a variable whose value should never change). Your function should ensure that the state input is a valid state abbreviation, and it should raise an `AssertionError` if it is not.

Here is an example of a plot generated by calling `stateDivide("OH")` for Ohio:

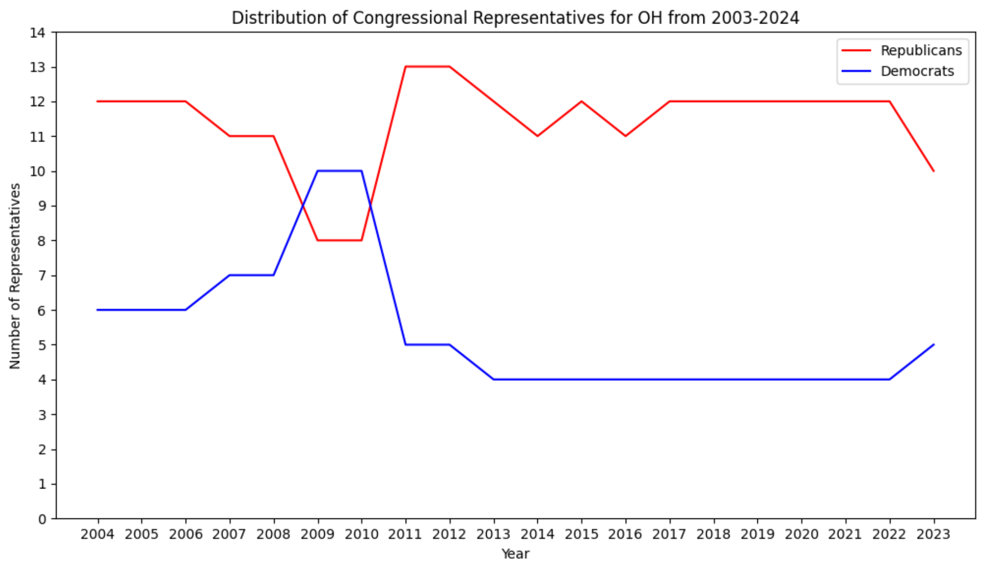

Figure 1: A sample output for the `stateDivide` function, using Ohio as the output. Note the title, legend, and axis labels. From this plot, we can see that the distribution of representatives in Ohio has remained fairly steady over time, with a brief interlude where there were more Democrats than Republicans from 2009 to 2010.

4 Implementation

The following function(s) have been provided to help you get started:

1. `getVoteFileFromWeb(url)` (in `utilities.py`): This function takes the web address (url) where the data for a specific roll-call vote is specified. It returns a list of lines, which are strings corresponding to each of the lines in that data file.

You should implement at least the following functions in the corresponding files, importing them as needed where you want to use them. Do not change the function definitions or behaviors of these specific functions, as this may cause your project to fail automatic testing.

1. `countVotesByParty(lines)`: This function should iterate over a list of lines in a voting data file and return a list containing, **in order**, the number of Democrats who voted for the bill, the number of Democrats who voted against the bill, the number of Republicans who voted for the bill, and the number of Republicans who voted against the bill. Do not change this order.

2. `isPartyLine(demYes, demNo, repYes, repNo)`: This function should take the votes for and against the bill, by Democrats and Republicans, as four arguments. It should return a `bool` indicating whether the vote was a party-line vote or not (see Section 2) for the definition of party-line).

3. `isACAPolarized()`: This function should open the included `aca.xml` file. This file is the voting record for the final vote on the Affordable Care Act in 2010. The function should return a `bool` indicating whether the ACA vote was a party-line vote or not. You should call the other functions you've written to make this determination.

4. `isMyChoicePolarized()`: This function should use `getVoteFileFromWeb` to get the voting data file for a roll-call vote of your choice—ideally something you care about. It should return a `bool` indicating whether the ACA vote was a party-line vote or not.

5. `countPartyLine(year, maxNumber)`: This function returns the fraction of votes during `year` that were party line votes, using roll call numbers 1 through `maxNumber` (inclusive).

6. `plotPartyLine()`: This function plots, for each of the last 20 years (2004–2023, inclusive), the fraction of party line votes in Congress out of the first 450 votes for that year. **Your plot should have axis labels and a title.** It also generates a text file with the fraction for each year recorded on each line. This function may take a very long time to run, so make sure it works on smaller values first (e.g., try it out over the last 3 years, for the first 30 votes each year). Additionally, I would recommend writing each result to file as soon as you get it, so you can pick up where you left off if something happens while you are running your code. **IMPORTANT: once you've generated the plot, save it to a file (you can do this by hitting the save icon in the matplotlib window)!**

7. `stateDivide(state)`: This function plots the number of Republican and Democrat representatives in a given `state` each year over the last 20 years. **Your plot should have axis labels and a title.** If `state` is not one of the two-letter state abbreviations, your program should raise an `AssertionError`. **IMPORTANT: once you've generated a plot, save it to a file (you can do this by hitting the save icon in the matplotlib window)!**

You are encouraged to decompose the functions further as you design your own logic. However, you MUST implement these functions exactly as they are defined above. Changing any of the input or output behaviors from what is enumerated may cause you to fail graded test cases.

5 Writeup

You should submit, with your code, a writeup (.doc or .pdf) with the following content:

5.1 Questions

Answer the following questions in a single section:

1. Was the vote on the Affordable Care Act a party line vote?
2. Briefly (in one or two sentences) describe the vote that you chose for your implementation of the function `isMyChoicePolarized`. Was it a party line vote?
3. Describe the plot generated by `plotPartyLine()`. How has the fraction of votes that went along party lines changed over the past 20 years? Do you see any trends or interesting patterns in the data? What conclusions can you draw about polarization in American politics over the last 20 years? **IMPORTANT: include the plot in your report.**
4. Many news outlets report on the issue of polarization. Find a news story about this topic online, and compare your results to the story. It might be helpful to think about the motivations of the news outlets (e.g., if the news outlets align with a particular party). You should also think about the reliability and credibility of particular news outlets.
5. Call `stateDivide(state)` on at least 5 different states. Describe the curves you get for each state. Interpret your results in light of the results from the earlier questions. Has the House of Representatives become more polarized? **IMPORTANT: include the plots in your report.**

5.2 Personal Reflections

What challenges did you face in completing this project and how did you overcome them? What concepts from class were useful in the completion of this project? What are three takeaways that you learned over the course of this project?

5.3 Ethical Reflection

How could the kind of analysis you developed in this project be used positively and/or negatively? Are there any potential benefits or ethical pitfalls that you can see with this kind of data analysis? Are there any best practices that you can think of to doing this kind of work responsibly and ethically?

5.4 Contribution Statement

How did every teammate contribute to this project? As always, I reserve the right to adjust scores individually based on the contribution statement. Please also submit your group contract files on GitHub.

6 Submission and Rubric

Push all of your code to GitHub. You should also include your final `partylines.txt` file and all of your plot files in your repo, as well as your group contract files (pictures or any other format are ok). Your writeup should also be included in your repo in a format of your choice and pushed to GitHub.

The project will be graded as follows:

Functionality	Does the code properly fetch the corresponding roll call data for a given year and vote number? Does it parse the xml data using the specified tags and count the number of votes cast for and against a bill by each party? Does it correctly determine, given a set of votes, whether the vote was party-line or not? Does it correctly iterate over the past 20 years and calculate the fraction of party-line votes for each year? Does it correctly generate a plot with a meaningful title and axis labels? Does it save the data in a file as specified? Does the state divide function correctly count the number of Republican and Democrat voters in each year for the correct state? Does it correctly generate a plot indicating how each of these totals have changed over the past 20 years, with a meaningful title and axis labels?
Structure and Style	Is the code readable? Is there a reasonable decomposition into functions? Are the functions implemented in the correct modules? Is the code needlessly repetitive? Are there excessive or unnecessary global variables? Does the program effectively use a main function as a single entry point? Do variables and functions have meaningful name? Are there docstrings on each file and function definition? Does the code have sufficient meaningful comments to document its functionality?
Writeup	Completeness: Are all the sections and questions completed? Are the plots included in the writeup? There should at least be one plot for the party line vote data and five individual state plots for the state divide question. Clarity: Does the writeup clearly explain the project and answer the questions in a thoughtful manner? Are there significant grammatical errors or typos that make the report difficult to understand? Ethical and Personal Reflections: Is there a thorough discussion of both benefits and risks of this kind of technology? Are personal reflections on the project discussed? Teamwork Does the contribution statement make it clear how both teammates contributed to the project and is the distribution of work even? This score and the overall lab score may be adjusted individually but group work issues should be brought to the professor BEFORE submission. Part of this score is completing and submitting the group contract.