Simulation of Gerrymandering and Districting Outcomes

Chibueze Onuoha

December 9th, 2024

Contents

1 Introduction

District boundaries have a great deal of consequence for the outcome of democratic systems. For countries such as the United States with several district elections, methods in districting are crucial in determining the winning political parties. Some methods for setting the boundaries of these districts include **gerrymandering**, or changing the boundaries to favor one party or group over others. This manipulation generally leads to results that do not represent the actual distribution of preference among voters, and as such, it is destructive to the democratic process. In this paper, we analyze the operational impact of gerrymandering through a simulation-based approach. Our goal is to examine how different districting methods-regular and gerrymandered-affect the distribution of electoral outcomes.

The key goal of this paper is to examine the effect of gerrymandering on election results by simulating the process of districting and analyzing the resultant political landscape. Since gerrymandering is capable of giving unfair advantages to one party, knowledge of its mechanics is highly essential for both policymakers and the common man. This simulation will provide an overview of how district boundaries can be manipulated to create favorable outcomes for a specific political interest and, more importantly, the ethical implications of such practices.

2 Implementation

2.1 Program Layout

Rather than merely dividing the voter population into districts based on geographic or demographic criteria, this model focuses on the manipulation of district boundaries in favor of the losing political party to achieve a skewed electoral outcome. In regular districting, boundaries are set without interference, and representation comes out properly reflecting the distribution of voters. However, in the case of gerrymandering, the boundaries are manipulated in such a way that the losing party gets a disproportionate number of districts even when it has fewer votes overall.

The technique of gerrymandering depends on moving voters between contiguous districts. In the event of a district having a clear majority for one party, a chunk of the voters of the losing party is shifted to a neighboring district. This raises the probability of a win in that district by the losing party while leaning more in the favor of the winning party, that is shifting from one district. The percentage determining how many to shift decides just about, but does not over-alter, how many to move around and make each manipulated district gain adequate political advantage on the side of the losers.

This dynamic manipulation scheme thus alters the composition of the districts during simulation. When the model has divided the districts into an initial configuration without manipulation, a gerrymandering technique of shifting voter affiliations to create more politically favorable outcomes for the targeted political party is applied. In this model, both the number of controlled districts and the voter distribution in each district are taken into consideration in trying to maximize the effectiveness of manipulation.

It also allows for new districts or voters to be created as part of the gerrymandering,

including the dynamic shifting of boundaries. C++ is used to simulate the voters and districts and the manipulation techniques involved in the program; Python is used with libraries such as matplotlib to visualize the manipulated district boundaries and electoral outcomes.

2.1.1 Voter

The 'Voter' class represents individual voters. The voters are represented by two things, an id and their affiliation.

Listing 1: Voter Class

```
class Voter{
private:
   int id;
   int affiliation; // 1,0,-1 for party A, Undecided, and party B
      respectively
public:
    Voter(int voter_id, int party_affiliation)
        : id(voter_id), affiliation(party_affiliation) {}

    int get_affiliation() const { return affiliation; }
};
```

2.1.2 District Class

The 'District' class contains a list of voters and determines the majority party. This is done through the "majority()" method which calculates the majority through addding all the voter affiliation with eachother and determining which one is bigger, the one that's bigger gets their affilation repsentation returned

Listing 2: District majority method

```
...
    int majority() const {
        int countA = 0, countB = 0;

        for (int i = 0; i < voters.size(); ++i) {
            if (voters[i].get_affiliation() == 1) countA++;
            else if (voters[i].get_affiliation() == -1) countB++;
        }

        if (countA > countB) return 1;
        else if (countB > countA) return -1;
        else return 0;
    }
};
```

2.1.3 Districting

The 'Districting' class is designed to create districts and enable gerrymandering manipulation. To simplify the implementation, I assumed the state is perfectly square-shaped. This shape allows for an even distribution of the population across districts, making it easier to achieve equal representation[1]. As a result, the congressional districts are also square-shaped to maintain balance and fairness.

2.1.4 Normal Districting

As stated earlier, the standard districting method I envisioned was creating a multiple equal sized congressional districts each with a certain amount of people in them

The first step to achieve this is to find out how many congressional districts there even are. this is done by dividing the population size by the district size. In this example, assume 1000000 and 20000 respectively.

Listing 3: Calculating congressional districts

```
1
2 public:
3     // Constructor that creates districts from the population of
          voters
4     Districting(vector<Voter> population, int district_size) {
5         int num_districts = population.size() / district_size; //
              Calculate how many districts we can create
```

From there A loop runs from i = 0 to the total number of districts (num_districts). Each iteration corresponds to the creation of a new district.

After that, the code initializes a list of voters for each district (district_voters). The voters are assigned to a district in chunks of the specified district_size. This is achieved by iterating through the population, adding voters from the population list to the current district's list until the required number of voters is reached for that district.

Listing 4: Standard Districting method

```
1
2         // Create the districts by dividing the population into
              chunks of district_size
3         for (int i = 0; i < num_districts; ++i) {
4             vector<Voter> district_voters; // A list to store voters
                  in this district
5             for (int j = i * district_size; j < (i + 1) *
                  district_size && j < population.size(); ++j) {
6                 district_voters.push_back(population[j]); // Add
                      voters to the district
7             }
8             districts.push_back(District(district_voters)); //
                  Create a new district and add it to the list
```

2.1.5 Gerrymandering

Once the initial districting is completed, we implement the gerrymandering process. The idea is to manipulate districts in favor of the losing party by shifting voters strategically. This section describes the process of determining which party is losing and how voters from the winning party are moved to the next district, effectively changing the outcome.

The gerrymandering method starts by calculating how many districts each party currently controls. Then, it decides which party is "losing" based on the district control and manipulates the voter population by shifting a percentage of voters from the winning party to the losing party's district. This helps to make the district outcomes more favorable to the losing party.

Listing 5: Gerrymandering Method

```
void gerrymander() {
    // Count the number of districts controlled by each party
    int countA = 0, countB = 0;
    for (size_t i = 0; i < districts.size(); ++i) {
        int majority = districts[i].majority();
        if (majority == 1) countA++;
        else if (majority == -1) countB++;
    }
    // Identify the losing party
    int losing_party = (countA < countB) ? 1 : -1;

    // Move voters from the winning party to the losing party's
       district
    for (size_t i = 0; i < districts.size() - 1; ++i) {
        District& current = districts[i];
        District& next = districts[i + 1];

        vector<Voter>& current_voters = current.get_voters();
        vector<Voter>& next_voters = next.get_voters();

        int move_count = current_voters.size() / 10; // Move 10% of
           voters
        int moved = 0;

        // Shift voters from the winning party to the next district
        for (auto it = current_voters.begin(); it != current_voters.
           end() && moved < move_count;) {
            if (it->get_affiliation() == losing_party) {
                next_voters.push_back(*it);
                it = current_voters.erase(it);
                moved++;
            } else {
                ++it;
```

The inspiration behind this algorithm was taken from the observation of the pattern in standard districting, whereby the electoral outcomes tend to be closely contested, and the winning party tends to control a bunch of contiguous districts.

A party with a larger share of districts will more often than not, just like Party A in this example, have multiple contiguous districts in their control. This is the clustering effect: the stronger presence of a party increases the likelihood of adjacent districts being similarly dominated. Based on this observation, I hypothesized that transferring a significant portion of voters from one district into another district already leaning toward Party A would solidify the latter district in Party A's favor. In contrast, the first district would have, after voter redistribution, a much better chance of flipping to Party B.

The result of this would be that Party A, with this strategy, would hold a far larger share of the districts-some 25% more than in the baseline case-whereas Party B would retain a greater number of smaller districts, each with an increased chance of victory. This is the redistribution tactic most commonly associated with gerrymandering: in congressional districting, the total number of districts a party holds is more important than the size of the districts.

My algorithm uses a "packing" approach[2]: it condenses the opposition into just a few large districts and spreads the remaining voters into a number of smaller districts where it is more likely to win. This approach shows how effective districting can alter the balance of power: a party doesn't have to win the largest districts, it just has to win enough of them.

3 Results

3.1 District Results

Table 1 and Table 2 summarize the results for regular and gerrymandered districting, respectively.

Table 1: Results for Regular Districting

District Type	**Party A Wins**	**Party B Wins**	**Toss-ups**
Regular Districting	27	23	0

Table 2: Results for Gerrymandered Districting

District Type	**Party A Wins**	**Party B Wins**	**Toss-ups**
Gerrymandered Districting	17	33	0

3.2 Comparative Analysis

As shown above, Table 1 Shows an instance of a natural distribution of congressional districts and Table 2 compares the outcomes of the popuation, after it has been gerrymandered. The

once minority but politically competitive party B has become essentially a supermajority monipoly through gerrymandering with 33 districts to just party B's 17.

3.3 Visual representation

While the previous sections provided extensive code to model the districting process, I believe it's important to give a clearer understanding of what this looks like in practice. To that end, the following figures visually demonstrate how the program generates and divides the population across the grid. These visuals offer a more intuitive sense of the districting procedure and its implications.

First The program first generates a population of people in random locations on a 10x10 grid such as this

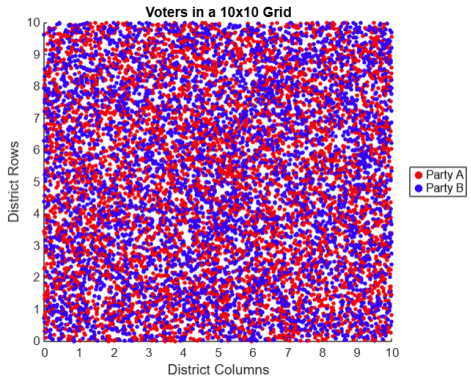

Figure 1: population

The standard districting then breaks them into evenly sized districts where the winner is the one who had more votes in said district.

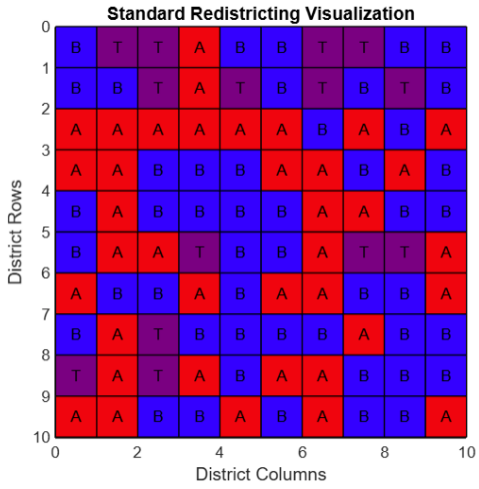

Figure 2: Standard redistricting

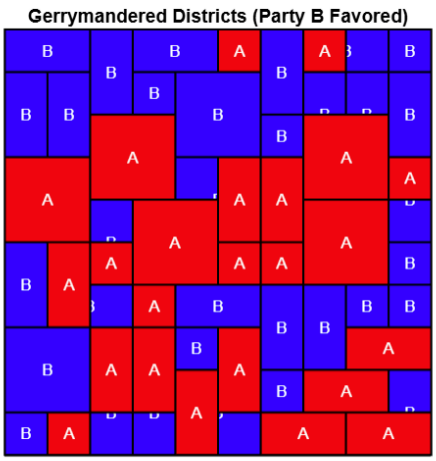

Figure 3: Gerrymandered districts

The Gerrymander method then changes the bounds of the districts by adding voters to the winning district, making that district bigger but also creating multiple smaller districts

4 Conclusion

Ultimately, the process of gerrymandering successfully distorted the outcome of districting in favor of the losing party. By comparison, fair districting methods yielded results that more accurately reflected the will of the voters and thus had equitable outcomes. However, gerrymandering created highly skewed results by strategically altering district boundaries to concentrate political power, often undermining the democratic principle of fair representation. This analysis underlines the importance of the examination and understanding of districting methods, as the way boundaries are drawn may have a great impact on election outcomes, with far-reaching implications for political equity and fairness. While gerrymandering may offer temporary advantages to certain political parties, it ultimately distorts the true will of the electorate, underlining the need for transparent, impartial districting practices.

4.1 Extensions

A major limitation for my program is that it limits itself to only rectangle shaped districts. say we get a 4 districts, each in 2x2. These districts are predominantly party A but in each of the 4 ditricts around the borders of each of the 4 districts there exists a substantial amount of party B voters that form the shape of a circle. In my program, those 4 districts would be go towards party A but they would not accurately represent the will of the people in that area.

Another limitation of my program is that it assumes randomness of votes spread evenly. In real life, people live in communites and whether it be because of social pressure or shared socio-economic backgrounds they tend to also share political leanings. A method to create

better simulations would be a sort of "infection" function where the chance of someone being the same party leaning as the person before them is increased by a certain percentage. This would accurately simulate communites and diveristy in areas such as Travis county and Oldham county where there is a large congregation of one-party voters in those communites.

4.2 Ethics

Although I would like to say Gerrymandering is absolutely bad or absolutely good, like most things in life, it's complicated. Our nation was founded on the fact that everyone should get a voice in their government in all levels and even if you're in the minority, you shouldn't be bullied by the majoirty. The U.S. senate exemplifies this as it was founded on the fear that states with less population would be pushed out of importance in house of representatives and would have no voice in congress so they created a system where every state gets two senators, regardless of how many citizens you have in your state. Gerrymandering also exemplifies helping the minority out. Take my circle example from earlier, if standard districting was done, those people would have never had a chance to determine what goes on in their communnites. In this instance, Gerrymandering helped an area of people get their representative voices back.

While Gerrymandering is at times used as a tool in strategy, at other instances it has more serious effects: sometimes, the impression may occur that such practices are at the expense of the representation of all citizens and exist solely in favor of maintaining a preeminent status for selected political parties. This consequently arouses suspiciousness towards the democratic process for distorting principles of equality in representation into unduly obtained results through defective representation.

5 References

Works Cited

1. "Districting." *Aceproject.org*, 2024, https://aceproject.org/main/english/ei/eie03.htm.

2. "Gerrymandering 101." *Southern Coalition for Social Justice*, 9 Oct. 2017, https://southerncoalition.o
101/.

3. The Editors of Encyclopaedia Britannica. "United States Senate — Definition, History, & Facts." *Encyclopædia Britannica*, 2019, https://www.britannica.com/topic/Senate-United-States-government.