An American Redistricting and Gerrymandering Simulation: Can the Minority Outperform the Majority?

Nakul Mody, Matthew Thomas, Jacky Lin

EID: nm34494, mjt3248, jyl866

TACC: nakulmody27, mjt2005, jackyyl

nm34494@my.utexas.edu, mjt3248@my.utexas.edu, jyl866@my.utexas.edu

COE 322

Fall 2024

1 Introduction

Gerrymandering, the deliberate manipulation of electoral district boundaries to benefit specific political parties or groups, has been a persistent challenge in the pursuit of fair and representative democracy. The practice undermines the principle of equal representation, distorting electoral outcomes and often disenfranchising significant portions of the electorate. Addressing this issue requires a deep understanding of the mechanics and consequences of redistricting strategies.

This paper explores gerrymandering through a simulation framework developed using recursive algorithms in C++. The model generates random population samples, allowing for flexible adjustments to district size and composition. By employing recursion, the framework dynamically simulates redistricting processes, making it well-suited for analyzing diverse and complex scenarios. The integration of random sampling and customizable parameters enables the simulation to explore a wide range of possibilities, shedding light on the structural and demographic factors influencing gerrymandering outcomes.

2 Background

The United States House of Representatives serves as the lower legislative body, comprised of 435 members representing congressional districts across the fifty states. The United States Constitution requires that congressional districts must be redrawn every ten years to reflect the population data generated from the U.S. census. This power lies at the state level, where either an independent commission or the state legislature draws electoral maps. Each state creates a number of districts equal to the number of seats they are allotted in the House. These districts must be about equal in size and house the voters who will elect the congressperson who represents them.

Since the constitution does not provide any direct structure to where and how congressional districts must be drawn, it is possible to create an electoral map that gives one party a disproportionate number of seats in their state congressional delegation. The practice of drawing districts such that one party can effectively dilute the power of another party through the way citizens with known voting patterns are grouped is called gerrymandering.

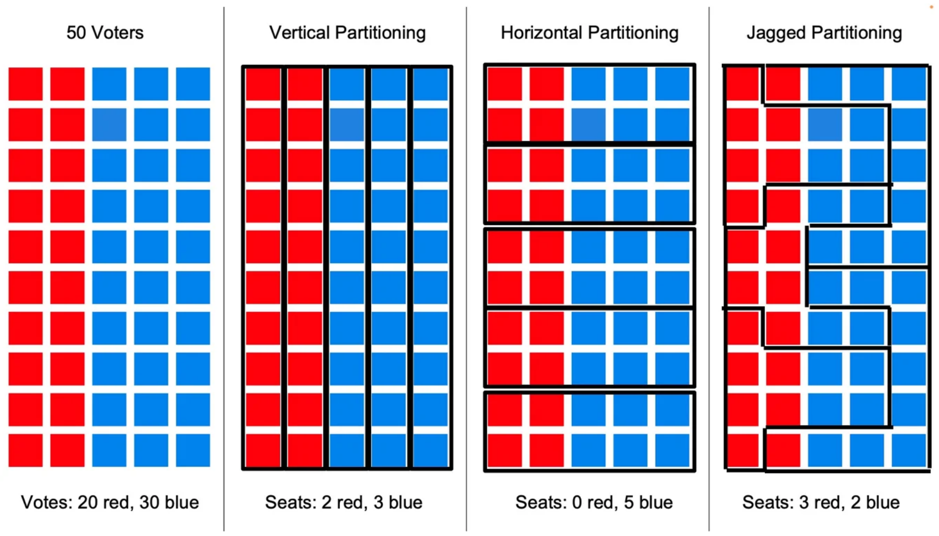


Figure 1: Visual example of redistricting and Gerrymandering

2.1 Types of Gerrymandering

Gerrymandering takes several forms, each with distinct political and social implications. Partisan gerrymandering is perhaps the most common and is the main focus of our simulation.This occurs when district boundaries are drawn to favor one political party by either concentrating opposing voters in a few districts or spreading them thinly across many districts to dilute their influence. Beyond partisan gerrymandering there are other forms such as racial gerrymandering which focuses on redistricting on the basis of race.

2.2 Ethical Implications

The ethical implications of gerrymandering and its countermeasures form a critical part of this discussion. Gerrymandering, when misused, undermines democratic principles by distorting electoral outcomes and disenfranchising voters. The primary ethical goal of this simulation is to highlight the extreme outcomes that gerrymandering can produce. The fragility of democratic systems is emphasized by demonstrating how a minority can consistently win through strategic district manipulation, highlighting the ease with which electoral boundaries can be drawn without accountability and redistricting weaponized to disenfranchise majority populations.

3 Methodology

3.1 Objects

The primary data of this simulation is the Voter class. The Voter class was constructed with race, gender, age, education, and type of living attributes that were randomly generated and used to predict a voter's party affiliation. Data from the Pew Research Center [2] was used to gather information on American voting patterns based on each of these attributes, which were factored into probabilities that an attribute would increase the likelihood of a voter holding a certain party affiliation. A snippet of this code is shown.

```
if (race == "Asian") {
    if (number_race <= 53) {dem++;} // 53% of Asians vote dem
    else {gop++;}
    }
else if (race == "Black") {
    if (number_race <= 83) {dem++;} // 83% of blacks vote dem
    else {gop++;}
    }
else if (race == "Hispanic") {
    if (number_race <= 61) {dem++;} // 61% of Hispanics vote dem
    else {gop++;}
    }
else if (race == "White") {
    if (number_race <= 41) {dem++;} // 41% of whites vote dem
    else {gop++;}
    };
```

A random number generator was seeded with each type of attribute to ensure randomness. Whichever party is selected for 3/5 attributes will be designated as the party for that Voter through a *set_aff()* method. Although this method of predicting a voter's party affiliation by essentially gaining a "point" for each type of attribute is naive and not realistic, it serves its purpose acceptably in this simulation. Because a majority of these attributes correlate to a Democratic voter, the authors found that there were always a larger group of Democratic voters than Republican voters in large sample sizes. Although there was no general attribute that could offset the lean towards Democratic voters, turnouts of the past 10 elections show that there were more democratic voters than republican voters[4].

Building on the Voter class is the District class, which in itself is a one-dimensional vector of Voters. The District class holds attributes for the total amount of democrats and republicans in it as well as methods to calculate its partisan lean as a tuple containing the majority party and its percentage of victory.

The final unique object created for this simulation is the State class, which holds a vector of Districts and is used to determine population-level characteristics and demographics.

3.2 Redistricting

To contrast gerrymandering with a truly random redistricting method, two functions were written to redistrict a State into districts.

3.3 Random Redistricting

In order to provide a constant in this experiment, a vector of voters was redistricted without respect to partisan affiliation. For simplifying purposes, it is assumed that there is a maximum number of voters that can be packed into a singular district and that voters must be contiguous to be grouped into a district. This vector of voters that was produced from the *generate_voters()* function is then divided into districts with an equal number of voters in each district. The districts are then returned to become a State.

3.4 Realistic Redistricting

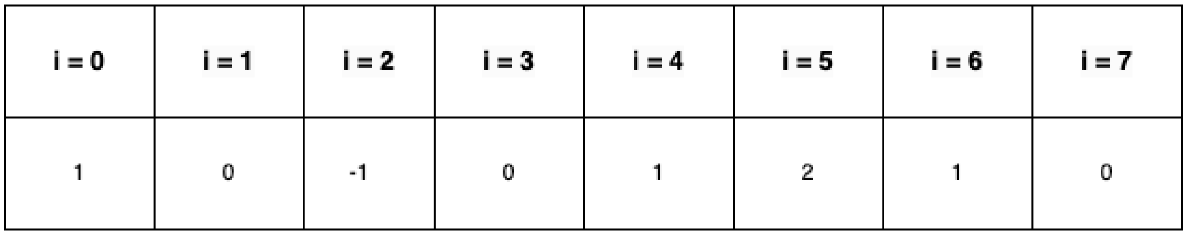
The second type of redistricting that is explored is one that more accurately reflects the real world. The population of diverse states in the U.S. is not evenly dispersed by race, education, and living type. It is much more likely that those with similar demographics live in proximity to each other, especially in densely populated urban cities. For that reason, the redistricting function was rerun with the vector of voters ordered by these attributes. To do this, voters of the exam's same race, education level, and living type were grouped to produce contiguity among them in the one-dimensional vector where they belong. When they were redistricted, districts would consist of voters of similar demographics, which better represents the real world. The function that achieves this redistricting variation is shown.

```
State ordered_districting(vector<Voter> &voters, int max_district_pop) {
    int number_of_districts = voters.size() / max_district_pop;
    voters = combine_like_voters(voters);
    vector<District> new_districts;
    for (int i = 0; i < number_of_districts; i++) {
        District district = vector<Voter>(voters.end() - max_district_pop,
voters.end());
        voters.erase(voters.end() - max_district_pop, voters.end());
        new_districts.push_back(district);
    }
    return new_districts;
};
```

3.5 Gerrymandering

The primary objective of this project is to simulate a version of gerrymandering in comparison to other redistricting methods. For this algorithm, the number of voters in each district is not fixed and can vary greatly. An analysis of this methodology will be explained in the following sections.

3.6 Methods



i = 0	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7
1	0	-1	0	1	2	1	0

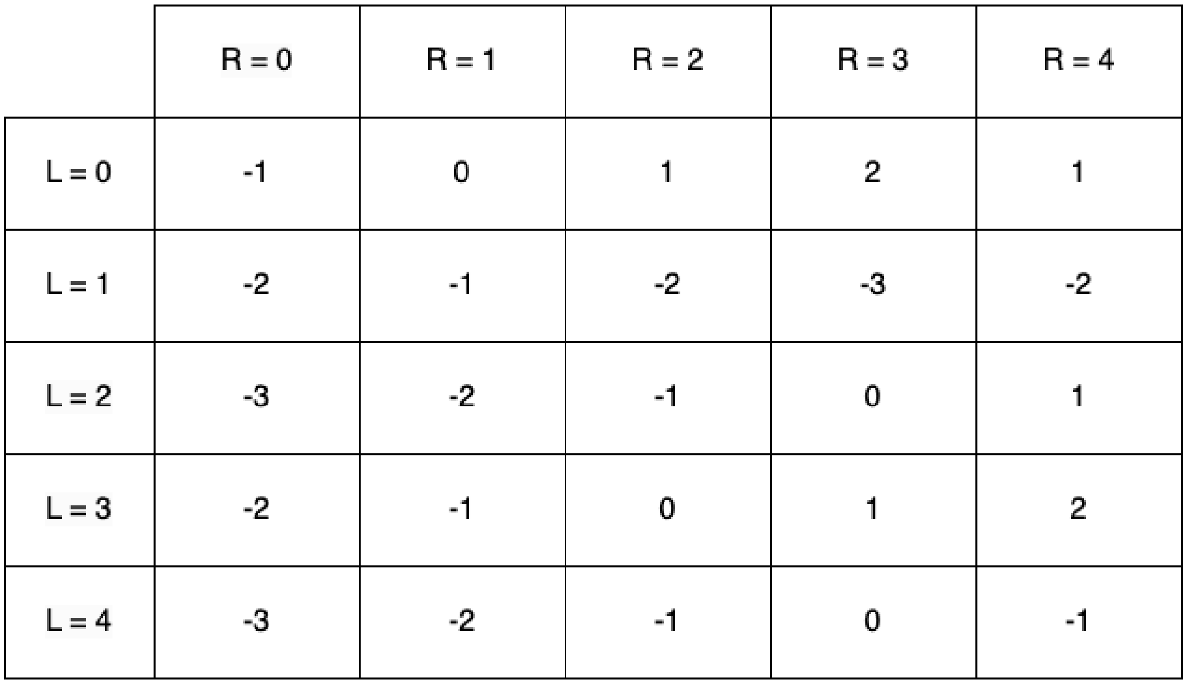
Figure 2: Prefix Sum

```
vector<int> computePrefixSum(const vector<int>& affiliation_converted) {
    vector<int> prefix_sum(affiliation_converted.size() + 1, 0);
    for (int i = 1; i <= affiliation_converted.size(); i++) {
        prefix_sum[i] = prefix_sum[i - 1] + affiliation_converted[i - 1];
    }
    return prefix_sum;
}
```

Figure 2 depicts a prefix sum vector designed to simplify the calculation of district leanings efficiently. Each voter's position is fixed at a specific index within the vector, ensuring the order cannot be altered. The prefix sum technique allows us to compute the cumulative leaning of voters incrementally, without repeatedly iterating over the entire vector.

To construct the prefix sum vector, voters are first categorized into majority and minority groups. Each minority voter is assigned a value of -1, while majority voters are assigned a value of +1. The vector is then iterated through, with the prefix sum at each position being updated by adding the previously computed sum to the current voter's value (-1 or +1, depending on their group).

This approach significantly enhances computational efficiency. Once the prefix sum vector is built, the leaning of any contiguous group of voters can be quickly determined by subtracting the prefix sum value at the start of the group from the value at the end of the group. This eliminates the need for repeated traversal of the voter vector, making it an optimal solution for analyzing district leanings.



	R = 0	R = 1	R = 2	R = 3	R = 4
L = 0	-1	0	1	2	1
L = 1	-2	-1	-2	-3	-2
L = 2	-3	-2	-1	0	1
L = 3	-2	-1	0	1	2
L = 4	-3	-2	-1	0	-1

Figure 3: Cost Matrix

```
vector<vector<int>> computeCostMatrix(const vector<int>& prefix_sum, int n) {
    vector<vector<int>> cost(n + 1, vector<int>(n + 1, 0));
    for (int l = 1; l <= n; l++) {
        for (int r = l; r <= n; r++) {
            cost[l][r] = prefix_sum[r] - prefix_sum[l - 1];
        }
    }
    return cost;
}
```

Figure 3 illustrates the cost matrix, a critical optimization tool for improving computational efficiency. Districts can be formed by voters starting and ending at any position, but relying solely on the prefix sum vector to calculate the leaning for a voter group requires performing the calculation each time. The equation used for this calculation is "leaning = *prefix_sum*[R] – *prefix_sum*[L - 1]". Here, L - 1 ensures that the voter at position L is included in the group. While this calculation is simple, its repeated use for overlapping or identical voter ranges becomes computationally expensive as the problem scales. The cost matrix solves this inefficiency by precomputing the leaning for all possible voter groups defined by L (start index) and R (end index). Each combination is calculated once and stored in the matrix. Similar voter ranges can then be retrieved through the cost matrix constant time (O(1)). By eliminating redundant calculations, the cost matrix ensures that the program efficiently handles multiple queries for the same or overlapping ranges. This precomputation significantly reduces the overall workload, improving the performance of the algorithm.

3.7 Dynamic Programming

	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5
i = 0	0	Max_Int	Max_Int	Max_Int	Max_Int	Max_Int
i = 1	Max_Int	-1	Max_Int	Max_Int	Max_Int	Max_Int
i = 2	Max_Int	0	0	Max_Int	Max_Int	Max_Int
i = 3	Max_Int	-1	1	1	Max_Int	Max_Int
i = 4	Max_Int	-1	0	2	0	Max_Int
i = 5	Max_Int	1	0	1	1	-1

Figure 4: Gerrymandering DP table

Figure 4 illustrates a dynamic programming (DP) table, implemented in the “solveDistricting” function within the “*redistricting_method.h*” file. The concept of dynamic programming involves decomposing a complex problem into smaller, manageable subproblems, solving these subproblems, and using their results to build up the solution to the original problem. This approach typically reduces the computational workload, leading to improved efficiency, often represented by a better time complexity in terms of Big O notation.

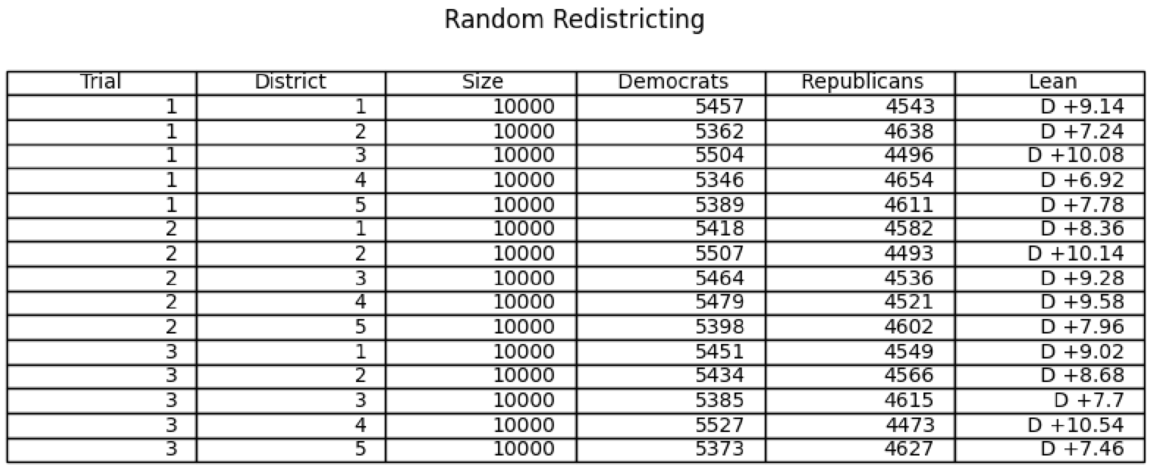
In this DP table, instead of processing all the voters in the state at once, the problem is tackled incrementally by considering only the first i voters at a time. The rows of the table correspond to these first i voters, while the columns represent splitting them into j districts.

The value at any cell (i, j) in the DP table is computed using previously computed values. Specifically, the table assumes that the optimal solution for dividing i-1 voters into j - 1 districts is already available. A for-loop iterates through potential configurations by considering the last t voters in the final district. Using the cost matrix (refer to Figure 3) for these t voters, the solution is updated if a more optimal configuration is found. This iterative process continues until the final cell of the table is computed, providing the solution to the original problem. The DP table ensures that each intermediate calculation represents the most efficient way to divide voters into districts up to that point.

4 Analysis

After running the simulation multiple times, a set of graphs has been created to illustrate comparisons between trials. This section will introduce these data sets and provide a deeper analysis of their meaning in the real world.

4.1 Data and Evaluation



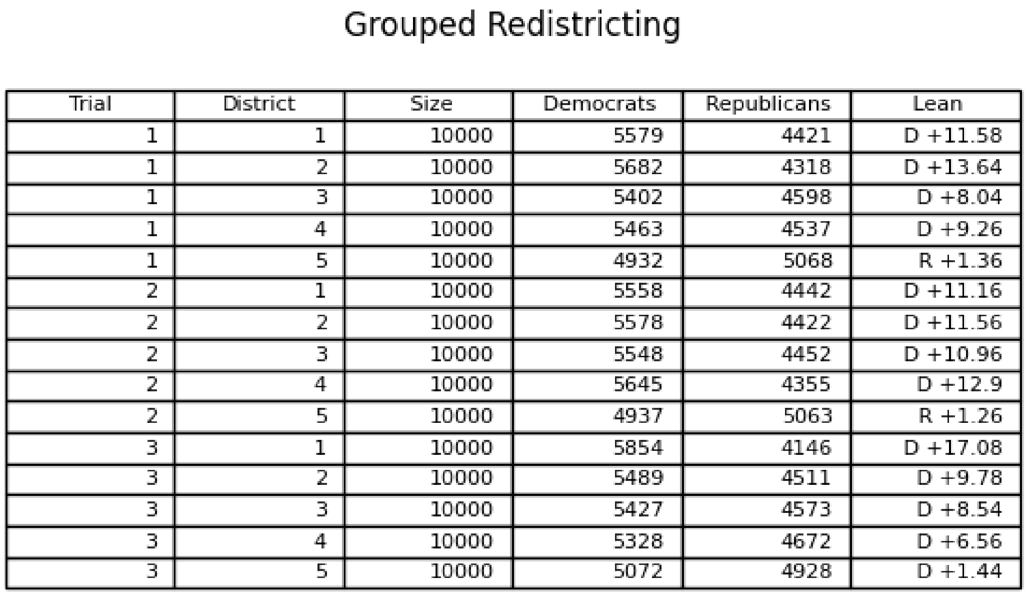
Random Redistricting

Trial	District	Size	Democrats	Republicans	Lean
1	1	10000	5457	4543	D +9.14
1	2	10000	5362	4638	D +7.24
1	3	10000	5504	4496	D +10.08
1	4	10000	5346	4654	D +6.92
1	5	10000	5389	4611	D +7.78
2	1	10000	5418	4582	D +8.36
2	2	10000	5507	4493	D +10.14
2	3	10000	5464	4536	D +9.28
2	4	10000	5479	4521	D +9.58
2	5	10000	5398	4602	D +7.96
3	1	10000	5451	4549	D +9.02
3	2	10000	5434	4566	D +8.68
3	3	10000	5385	4615	D +7.7
3	4	10000	5527	4473	D +10.54
3	5	10000	5373	4627	D +7.46

Figure 5: Random Population Table - 5 Districts, 50000 Voters

The table presented in Figure 5, illustrates the results of a simulation analyzing the effects of random redistricting on electoral outcomes. The simulation consists of three trials, each dividing voters into five districts, with each district containing 10,000 voters to ensure uniform population sizes. The Lean column quantifies the political advantage in each district, expressed as D (Democrats) followed by the percentage margin of victory. Due to the bias towards Democrats when finding the affiliation for a voter, there is a high volume of Democrats throughout the state. Because of this, it is not surprising that Democrats won each district even when shuffling randomly after a district is drawn.

These results are significant because they highlight the inherent variability in outcomes generated by random redistricting. The consistent Democratic advantage may reflect an underlying population skew toward Democratic voters or demonstrate that random redistricting does not eliminate natural biases in voter distribution. Although the districts are evenly sized, the variability in partisan lean also shows that even random districting can produce unequal partisan outcomes. Ultimately, the significance of the figure lies in its ability to illustrate the baseline variability in electoral outcomes, offering a foundational reference for further comparisons.

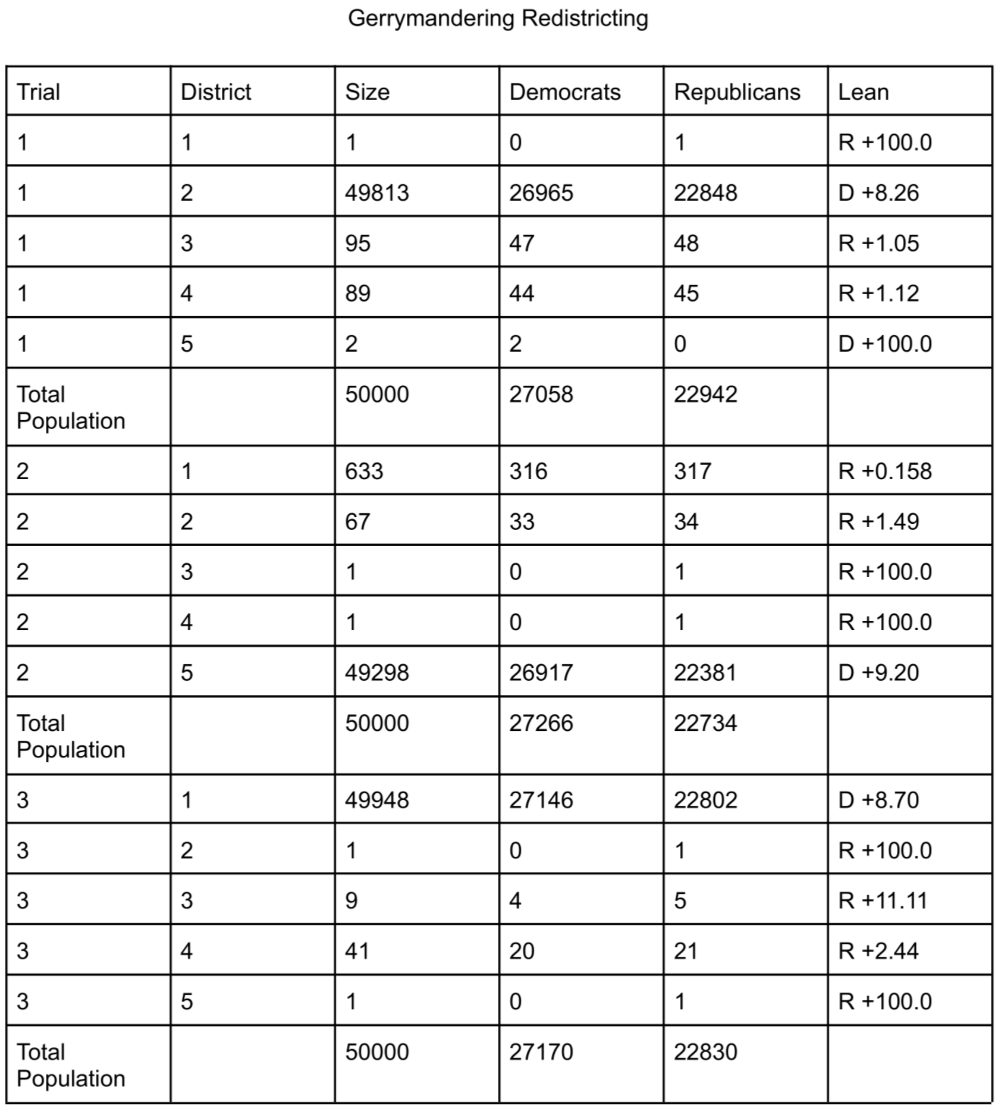


Grouped Redistricting

Trial	District	Size	Democrats	Republicans	Lean
1	1	10000	5579	4421	D +11.58
1	2	10000	5682	4318	D +13.64
1	3	10000	5402	4598	D +8.04
1	4	10000	5463	4537	D +9.26
1	5	10000	4932	5068	R +1.36
2	1	10000	5558	4442	D +11.16
2	2	10000	5578	4422	D +11.56
2	3	10000	5548	4452	D +10.96
2	4	10000	5645	4355	D +12.9
2	5	10000	4937	5063	R +1.26
3	1	10000	5854	4146	D +17.08
3	2	10000	5489	4511	D +9.78
3	3	10000	5427	4573	D +8.54
3	4	10000	5328	4672	D +6.56
3	5	10000	5072	4928	D +1.44

Figure 6: Grouped Population Table - 5 Districts, 50000 Voters

Figure 6 runs the same test as Figure 5, however, its voters have been rearranged to fit the typical behaviors of the randomly generated voters and where they would likely live in a real-world scenario. By adjusting the voter distribution to reflect more realistic scenarios, the simulation provides insight into how real-world factors, such as population density, geographic location, and political tendencies can influence electoral results. This run is more realistic as this redistricting data shows a more balanced and realistic political landscape, although still with a Democratic advantage. This is evident in how out of 15 total districts across three trials, Democrats won 13 districts while Republicans secured 2 districts. Whereas in the previous, Republicans claimed no victories.



Gerrymandering Redistricting

Trial	District	Size	Democrats	Republicans	Lean
1	1	1	0	1	R +100.0
1	2	49813	26965	22848	D +8.26
1	3	95	47	48	R +1.05
1	4	89	44	45	R +1.12
1	5	2	2	0	D +100.0
Total Population		50000	27058	22942	
2	1	633	316	317	R +0.158
2	2	67	33	34	R +1.49
2	3	1	0	1	R +100.0
2	4	1	0	1	R +100.0
2	5	49298	26917	22381	D +9.20
Total Population		50000	27266	22734	
3	1	49948	27146	22802	D +8.70
3	2	1	0	1	R +100.0
3	3	9	4	5	R +11.11
3	4	41	20	21	R +2.44
3	5	1	0	1	R +100.0
Total Population		50000	27170	22830	

Figure 7: Gerrymandering Data Table - 5 Districts, 50000 Voters

In this figure, there are 3 trials that are run with variable sizes in each district. In each trial, there are 50,000 voters that are split among 5 districts. As all 3 trials show, the Republicans are the minority, but through the redistricting algorithm, they are able to hold a majority of the districts. This shows that the redistricting function works and is able to successfully manipulate the minority to hold a majority of the districts. While district boundaries may not perfectly reflect reality, this figure demonstrates that redistricting can be achieved with sound reasoning.

Trial	Democratic Population	Republican Population	Democrat Districts	Republican Districts
1	27415	22585	1	9
2	27241	22759	1	9
3	27017	22983	2	8

Figure 8: Gerrymandered Population Table - 10 Districts, 50000 Voters

Figure 8 displays the breakdown for the 3 trials done with 50,000 voters and 10 districts. The authors decided to not include the district breakdown, as the tables were unnecessarily repetitive to the previous trial of 10 districts. Similar to the previous gerrymandering in the 5 district case, the Republicans were the minority and ended up with control over the majority of the districts. In regards to the random and grouped results for these trials, the overall analysis is the same as Figure 5 and 6. (Republicans begin to gain an upper hand) The most significant difference between the previous trial of 5 district voters is that the increase in the number of districts helped the Republicans win a greater percentage of districts when compared to the 5 district case. This is significant because it highlights how increasing the number of districts creates more opportunities for gerrymandering, allowing a voter minority (Republicans in this case) to gain disproportionate control.

Trial	Democratic Population	Republican Population	Democrat Districts	Republican Districts
1	27153	22487	2	18
2	27323	22677	2	18
3	27085	22915	2	18

Figure 9: Gerrymandered Population Table - 20 Districts, 50000 Voters

Figure 9 illustrates the outcomes of three trials involving 50,000 voters distributed across 20 districts, where Democrats consistently outnumber Republicans in total population. Despite this numerical advantage where the Democratic population ranges between 27,085 and 27,323, compared to the Republican population of 22,487 to 22,915 Republicans dominate district outcomes, winning 18 out of 20 districts in every trial. This stark disparity highlights how increasing the number of districts amplifies the effects of gerrymandering. By dividing voters into more districts, Republicans are able to manipulate boundaries to "pack" Democratic voters into a few districts and "crack" their influence across others. This strategic redistricting ensures Republicans maximize their control, winning the vast majority of districts even as the voter minority. The increase in districts thus significantly enhances the party's ability to secure disproportionate representation.

4.2 Key Takeaways

The simulations conducted over several trials demonstrates that district manipulation can have a major impact on political representation.

In all scenarios, Democrats maintain a majority of around 54 to Republicans' 46. Despite this, through gerrymandering Republican dominance was achieved. For example, in the 5-district gerrymander simulation, trial 1 it demonstrates the overall basic gerrymandering strategies. One of the districts district has a massive Democratic majority (D +8.26), which concentrated Democratic votes. The other districts are then altered to maintain Republican advantages (R +1.05 to R +2.44). This overall pattern is then made more obvious when looking at the 10-district scenario, where Democrats, despite their numerical advantage, win only 1-2 seats while Republicans win 8-9 districts.

The most district imbalance is seen in the 20-district simulation, where the Democrats consistently received two districts while Republicans dominated with 18 districts in all three trials. This great imbalance shows that by how increasing the number of districts elected officials want to create, they can increase the impacts of gerrymandering. The consistency of these results indicates that they are not random, but rather the result of effectiveness of gerrymandering to maximize partisan advantage.

These simulations overall, mirror real-world challenges, where parties and groups are able to create partisan advantages even when the voter preferences would suggest a different outcome. This demonstrates how in a democratic republic, it is possible for the minority to maintain power through legal means, such as drafting district lines, it can ultimately undermine overall ethics and fairness.

5 Discussion

5.1 Run Time

This simulation saw many code updates and changes to the workflow over the course of the month it was worked on. The gerrymandering algorithm was altered about three times to maximize efficiency. Also, based on the data generated, it is evident that the algorithm to generate party affiliations for voters is biased towards Democrats. The authors believe this is due to 3 out of 4 of the races favoring Democrats as well as any education level. In the simulation, randomness resulted in roughly equal numbers of voters belonging to each race, gender, age, education level, and living situation. However, all know that in America, the population skews white, older, uneducated, and urban, respectively.

The original goal of this project was to conduct the simulation on 1 million + voters, however, memory allocation errors were encountered when the computer attempted to conduct the dynamic programming aspect of the program. After several trials with varying numbers of population sample sizes, it was determined that 50,000 voters would be the upper bound of what was reasonable to run the simulation with, as it resulted in the program taking about 5 minutes to run. Although accessing a node on Stampede 3 did allow for the program to run successfully for 100,000 voters, its run time of more than one hour made that option too difficult for data to be generated.

5.2 Efficiency Gap

In a separate computation, the efficiency gap was the main analysis topic. The efficiency gap is used to quantify wasted votes in an electoral map. Wasted votes are defined as those cast for losing candidates or votes cast more than what is needed for a candidate to win. The efficiency gap compares each party's wasted votes to determine if one party benefits disproportionately from how district boundaries are drawn.

A table is produced in the program output that shows all the districts and a breakdown of democratic votes, republican votes, the wasted votes for their respective parties, as well as the net wasted votes which is used for efficiency gap calculation.

In the simulation, a negative efficiency gap displays that the votes for Republicans in that scenario did not waste votes as much as the Republicans. While a positive value displays the opposite.

This is significant in regard to the idea of maintaining equitable political systems. Imbalances can lead to policies that do not reflect the will of the majority. The efficiency gap is utilized to challenge unfair district maps, ensuring that elections are made more representative in a democracy.

5.3 Limitations

For the gerrymandering simulation, each district could contain a variable number of voters, whereas in reality, electoral districts are drawn to maintain roughly equal numbers of voters. Also, this model assumes a one-dimensional representation of a state, so contiguity stems from voters being located either to the left or right of them. However, in the real world, contiguous voters can be located in proximity through all directions because a state is two-dimensional.

6 Conclusion

After many tests and analyses, a clear conclusion can be drawn from the data. While minority groups are often at a disadvantage in terms of representation, it is physically possible to manipulate district lines in a way that amplifies their voices and increases their chances of winning. Our analysis revealed a clear trend: the likelihood of minority representation improves as the number of districts increases. This finding demonstrates how strategic redistricting can shift power dynamics, allowing for greater inclusivity in electoral outcomes.

However, these results also highlight the broader implications of gerrymandering. While it can be used to achieve positive goals, such as empowering underrepresented groups, it underscores the critical need for ethical and transparent redistricting processes to prevent abuse and ensure fair representation for all.

7 References

[1] M. Bischoff, "Geometry reveals the tricks behind gerrymandering," Scientific American, `https://www.scientificamerican.com/article/geometry-reveals-the-tricks-behind-gerrymandering/`

[2] Nadeem, R. (2024, April 9). 2. Partisanship by race, ethnicity and education. Pew Research Center. `https://www.pewresearch.org/politics/2024/04/09/partisanship-by-race-ethnicity-and-education/`

[3] Petry, E. (n.d.). How the efficiency gap works. `https://www.brennancenter.org/sites/default/files/legal-work/How_the_Efficiency_Gap_Standard_Works.pdf`

[4] Statista, "Share of popular votes for major parties in US presidential elections 1860-2020," Statista, Jul. 04, 2024. `https://www.statista.com/statistics/1035521/popular-votes-republican-democratic-parties-since-1828/`