An Empirical Bayesian Framework for Assessing Partisan Bias in Redistricting Plans

Kevin Baas and Colin McAuliffe

December 6, 2018

Abstract

There are several legal and technical challenges to the establishment of a standard for limiting partisan gerrymandering, and a few methods have been proposed thus far. All methods for examining gerrymandering use observations of the results of one or more elections to make inferences about the tendency of a given redistricting plan to lead to biased outcomes. Here we propose an empirical Bayesian framework which allows an analyst to make such inferences accurately by using all available election results for the redistricting plan in question in a robust and consistent statistical model. The framework can be used with any gerrymandering metric.

Additionally, we propose a new measure of gerrymandering called the specific asymmetry which we believe will stand up better to judicial and technical tests than any other measure proposed thus far. The specific asymmetry does not rely on proportionality of seats and votes, is applicable to any level of statewide partisanship, does not require national results as a baseline, and measures bias at the popular vote that actually occurred as opposed to some other hypothetical popular vote. All other available metrics fall short in at least one of these aspects, which leaves them vulnerable to criticism and manipulation by those seeking to gerrymander or to defend an existing gerrymander.

We analyze the magnitude and persistence of partisan bias under a particular districting plan by using the empirical Bayesian model to compute the expected value of the specific asymmetry. This analysis technique is applied to the United States congressional elections from 1972-2016 to examine the total and net effects of partisan bias in recent history. We also examine the Act 43 map for the Wisconsin State Assembly, which is the subject of *Whitford v. Gill.*

Keywords: Redistricting, gerrymander, Whitford v. Gill,

1 Outline

Motivation The problem The solution Transparency Transparent data in Transparent development Transparent source code Transparent data out The Optimization Engine The Genetic Algorithm Rank normalization The Optimization Schedule Starting conditions Gradual refinement Ending conditions The Metrics Necessary and sufficient properties Metrics used in AutoRedistrict (mainline distribution) Balancing the metrics Closing Summary

2 Motivation

The Problem Every 10 years the apportionment of U.S. House seats to individual member states are re-calculated based on the latest census data. Because the number of districts apportioned to a given state can change after this new calculation, and the populations within a district can change over time, new districts have to be drawn that put about the same number of citizens in each district.

Traditionally the districts have been drawn by hand or more recently assisted by mapping software. This process is very time consuming and expensive, and often results in districts that give a large advantage in representation to one group of voters over others, whether by accident or deliberately.

In the end, millions of dollars are spent on maps that are, ultimately, pretty bad at achieving genuinely democratic government. Indeed these maps are often contested in lawsuits on precisely those grounds, and these lawsuits an additional couple of million dollars to the cost. The voters who have been harmed by these maps are then forced to pay the enormous legal fees to defend the maps, since the legal defense is paid for by the taxes that the state collects.

With the increasing power of computers and the sophistication of software, the tools that bad actors can use to take control of the composition of Congress away from the people it purportedly represents, grow more effective at doing just that.

The Solution But computers themselves are agnostic. They are simply calculating machines and as such can be used for any purpose that involves calculation. This includes guaranteeing that districts are fair and representative of their population. That is to say that while computers can multiply the impact of bad actors, they can also multiply the impact of good actors. Used wisely, computers can even serve as a defense against bad actors; a wall that protects against them, or a light that illuminates their motivations and machinations.

I wrote this software with that goal in mind: to shed light on the bad actors, to give power to the good actors, and above all, to guarantee a strong and lasting democracy.

3 Transparency

Transparent data in: Open standard data format The software operates on ESRI shapefiles, which is an open specification, meaning anyone has the legal right to read and write data in that format, and the information a developer needs in order to do so – the specification – is publically published, by ESRI. (https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf)

Transparent development: Copy-lefted (GNU GPL v3) The right to use, copy, and distribute the source code for AutoRedistrict, and to create derivative works (modified versions of the software), is explicitly granted to everyone by the GNU-GPL 3.0 license. (https://www.gnu.org/licenses/quick-guide-gplv3.html) Developers are protected from liability, provided they follow the terms of this license, and they prominently state that the program has been modified from the original. This is to enable an environment where developers can contribute without fear of unwarranted legal threat, while simultaneously protecting users of this software from deceptive practices, such as repurposing the program to do the opposite of what its intended for.

Furthermore, all distributions and derivatives works must inherit this license, and the rights enumerated by it. Developers and users are prohibited from restricting access to the software or source code, including in derivative works (modified versions of the software).

While this license grants the freedom to alter and share the software and its source code, it does not restrict the right to sell it or otherwise profit from it (for instance, by providing support). This software, or derivative works, can be sold and profited from, provided the above mentioned terms of the license are met.

The reason for licensing the software this way is to ensure lasting and meaningful transparency, to enable continuous and unfettered improvement of the software, and to effectively transfer ownership of AutoRedistrict to the public sphere, while simultaneously protecting it from being repurposed for malicious intent.

Transparent source code: Open source AutoRedistrict is hosted on a public repository where everyone can see all the source code and the entire history of changes to it. Additionally the source code is included with every copy of AutoRedistrict. (https://github.com/happyjack27/autoredis)

Transparent data out: Automatic statistics AutoRedistrict automatically computes a variety of statistics, pie charts, and map overlays, all of which can be exported from the program with the click of a button. Furthermore the district assignments are written back to the shapefile, allowing them to be processed and/or analyzed by other software.

4 The Optimization Engine

To find the number one best map, one would have to search through and evaluate every possible map. This is not computationally feasible. So instead, a heuristic search algorithm is used to evaluate only those maps that are probably good.

AutoRedistrict uses a heuristic search algorithm known as the Genetic Algorithm. The genetic algorithm consists of a few basic steps:

Evaluate. The fitness of each map is evaluated on each of the criteria (in our case, compactness, equal population, competitiveness, proportional representation, etc.) Normalize the scores. Adjust the scores of the different criteria to all be in about the same range. Weight and add together the scores. The scores are then multiplied by their respective weights (controlled by the user), and added together into a grand total score. Select. The best scoring maps are selected for recombination. Recombine. These maps are then randomly recombined to form new maps. Mutate. Finally these maps are "mutated" slightly so that other potential maps that are similar to them are explored. (Repeat). This is the new population of potential maps. The process repeats from step 1.

Each iteration through these steps is called a generation. The probability of mutation is slowly reduced over many generations, allowing the algorithm to make smaller adjustments as it approaches an optimal solution.

Rank normalization While some scores, such as compactness, range from 0 to 1, others, such as population imbalance, can go into the thousands. To prevent the high-range metrics from drowning out the low-range metrics, we need to bring both metrics into the same range.

Furthermore, metrics such as contiguity can change abruptly in value based on the inclusion or exclusion of a single precinct. These sudden jumps can drown out all other metrics. To account for this, we need to constantly re-adjust our ranges, so that large jumps appear smooth.

This process of bringing different metrics into the same range is called normalization.

AutoRedistrict uses rank normalization. In rank normalization, instead of using the raw score, the scores are sorted from lowest to highest, and then their place or rank in that sorted list replaces the raw score. This produces normalized scores that are distributed perfectly evenly.

5 The Optimization Schedule

The starting conditions AutoRedistrict starts by generating hundreds of plans using a breadth-first flood fill (https://en.wikipedia.org/wiki/Flood$_{fill}).Thefactthatdistrictsareinitializedviaflood-fillensuresthatdistrictshavegoodcontiguity.Thefactthatitisbreadth-firstasopposedtodepth-firstensuresthattheinitialplanshavegoodcompactness.Additionally,theflood-fillisdoneoneprecinctatin$

Alternatively, AutoRedistrict provides the option of starting with an existing plan, such as a manually-drawn plan. In this way, AutoRedistrict can be used not only to generate new pans, but also to improve on existing plans.

Gradual refinement AutoRedistrict starts with a high mutation rate, and slowly reduces the mutation rate over time. Specifically, the mutation rate at iteration number t+1 is the mutation rate at iteration t, times a constant.

This is equivalent to saying that after x iterations, y

The ending conditions The optimization is considered complete when the map changes very little over many iterations. This is approximated by the mutation rate, since the mutation rate determines how much the map changes each iteration.

6 The Metrics

Necessary and sufficient properties for a metric to be usable in a genetic algorithm.

Since a genetic algorithm is a heuristic search, it does not need to know how to construct districts that meet the objective or objectives; it does not need to know a generating algorithm or method. A genetic algorithm only needs to know how to evaluate how well any given candidate meets the objective or objectives.

But there are a few qualities that such an evaluation needs to have in order to be practical and well-suited for use in a genetic algorithm.

A genetic algorithm can meet an unlimited number of objectives simultaneously, so long as each objective can be summarized in a single number "score", and each such score has the following properties:

High score (or conversely low score) represents good fulfillment of the objective. This is obvious and needs no explanation, but Ive included if for the sake of formal completeness.

A score can take on many different values. Ideally a score is continuously valued or nearly so. This allows the algorithm to make meaningful comparisons between two maps: one map always scores somewhat higher than the other; they rarely have equal scores. Maps with equal scores don't provide useful information for improving the score. (Note, this is not a strictly necessary property, but it has a large impact on convergence.)

Randomly combining two maps with good scores is more likely than not to be produce a map with a better score than randomly combining two maps with bad scores. This is the underlying assumption of a genetic algorithm. If this is false, the algorithm cant work.

Time to calculate the score does not grow too fast with the size of the solution (in bits). In computer science, computation time is measured in whats called Big-O notation. In Big -O notation, we are concerned only with how the number of computations scale with the number of data points. N signifies the number of data points. For instance if we are sorting a 52-card deck, N=52. If to sort them, we have to compare every card to every other card, then the number of computations is proportional to N*N. In Big-O notation, we write this as O(N*N). When we use Big-O notation, we only care about the largest factor, and dont care about any constant multiplier on it. So if the number of computations was 3*N*N + 2*N + 8, wed still just write O(N*N). Since a genetic algorithm will be running these calculations over and over again, its total running time is going to be proportional to the time to calculate the metric that takes the most time to calculate (expressed in Big-O). Thus the time to calculate a metric should not scale too fast in proportion to N. Roughly, O(N) or better is manageable and most metrics can be computed in this time, including compactness, contiguity, and equal population. Metrics that cannot be computed in this time can often be approximately scored in O(N) time or better by training an AI algorithm to estimate the score. For instance, Deep Learning can be used to evaluate the positional strength in a Chess game or Go board in O(1) time, despite a complete evaluation being intractable. For a genetic algorithm to be effective, an exact score is not needed, only a score that fulfills properties 1, 2, and 3, and a trained AI can fulfill these properties.

Available metrics in AutoRedistrict (mainline distribution) The following are the objectives (optimization criteria) available in the mainline distribution. and how they meet the required properties. In the following, N represents the number of precincts (aka voting wards).

Shared evaluation (before scores are calculated) Connected regions (contiguity) are identified via breadth-first flood fill. This is an O(N) algorithm. Precincts (aka voting wards) that make up a district are collected via iterating through all precincts and appending them to a linked list. This is an O(N) algorithm.

Geometric criteria

Equal population The equal population metric is calculated as the sum of the squares of the deviation of the population from the average population of a district.

Property 1: The score reflects how unequal the population of districts are. Minimizing the score maximizes how equal their population is. Property 2: This score varies close to continuously, with the granularity being approximately the population of the state. Property 3: The combination of two equal population maps is more likely to produce an equal population map than the combination of two non- equal population maps. Property 4: The computational complexity time of the algorithm to compute the score is O(N).

Compactness Compactness is measured as the sum of the reciprocals of the isoperimetric quotient of the districts. Isoperimetric quotient is calculated by dividing the area by the the square of the perimeter, and then multiplying by 4*pi. The reciprocals is taken so that non-compact districts are given more weight. This helps ensure that each district has about the same compactness.

Property 1: The most compact shape possible is a circle. The least compact is a line. A circle has an isoperimetric quotient of exactly 1, which is the highest isoperimetric quotient possible. A line has an isoperimetric quotient of exactly 0, which is the lowest possible. Property 2: The isoperimetric quotient varies continuously. Property 3: The combination of two compact maps is more likely to produce a compact map than the combination of two non-compact maps. Property 4: The computational complexity time of the algorithm to compute the score is O(N).

County splits Simply counting the number of counties split would not meet requirement 2, since then there would only be as many different scores as different counties. To remedy this, we consider the number of people that would be removed from a district, if you remove a split. This reflects how close the county is to having one less split. This also has the advantage of favoring optimization of the split that is most easily removed.

For each district, we find the split county with the fewest amount of people in that county and district. And then we add that together over all districts.

Property 1: Minimizing this score minimizes the number of county splits. Property 2: This score varies close to continuously, with the granularity being approximately the population of the state. Property 3: The combination of two low scoring maps is more likely to produce a low scoring map than the combination of two high scoring maps. Property 4: The computational complexity time of the algorithm to compute the score is O(N).

Fairness criteria

Partisan symmetry Party symmetry is based on the idea that every political party should have an equal ability to translate votes into seats. This is the anti-gerrymandering criteria.

This does not mean, however, that seats should be proportional to votes. In a collection of single-winner elections, the relation of seats to votes naturally follows a cumulative Beta distribution, which resembles an S or sigmoid, not a diagonal line. While its simply not possible to design districts that diagonalize this line, it is possible to design districts that make it very symmetric. This is what we mean by partisan symmetry. How symmetric is the seats-votes curve to itself, when flipped on both axi (x (votes) and y (seats)) We measure asymmetry by measuring the total area between the seats-votes curve and its 2-axis reflection.

Property 1: Minimizing this score minimizes an advantage one party has over another, that is not a consequence of the sigmoidal nature of the seats-votes curve. Property 2: This score varies close to continuously, with the granularity being approximately the population of the state. Property 3: The combination of two equal partisanly symmetric maps is more likely to produce a partisanly symmetric map than the combination of two non-partisanly symmetric maps. Property 4: The computational complexity time of the algorithm to compute the score is O(N).

Descriptive representation Intuitively, subtracting the fraction of seats one for each demographic, from the fraction of the population thats part of that demographic, and then adding together all positive such values, to get total fraction of undescribed voters, would seem like a good score for descriptive representation. However, such a score can only change by an entire seat at a time, and that violates requirement 2. So instead we take into account how many more votes are needed to win the next seat. The pseudocode for this below.

pps = population per seat seat$_target = numberofseatsthatwouldbeproportionaltopopulation(roundedn$
$numberofseatsthatdemographicwouldwinifeveryonevotedalongdemographiclinesvotes_for_next =$
$numberofvotesneededforthatdemographictowinthenextseat$

if(seat$_target > seat_current)score = votes_for_next; seat_current++; score+ = (seat_target - seat_currer$

Property 1: The score reflects the disproportion of the demographics of the elected representatives to that of the population. Minimizing the score makes the descriptive representation maximally proportional. Property 2: This score varies close to continuously, with the granularity being approximately the population of the state. to make it meet continuous requirement, used undescribed voters Property 3: The combination of two low scoring maps is more likely to produce a low scoring map than the combination of two high scoring maps. Property 4: The computational complexity time of the algorithm to compute the score is O(N).

All metrics meet all necessary and sufficient properties for use in a genetic algorithm. The worst-performing metric, in terms of computation time, is an O(N) metric, and the initial overhead is also O(N). Therefore, the evaluation step is done in O(N) time. This is manageable computation time for a genetic algorithm.

Balancing the metrics

Every metric is optional; every metric can be turned on or off. Additionally, the relative weight of each metric compared to the other metrics can be fully and continuously controlled, from 0 to 1. This allows different users or agencies with differing values on each of the metrics to feed those values into the program and get back a result that meets them in those proportions.

For instance, an agency can decide that its not important to keep counties whole, or to keep districts compact, but that avoiding an undue advantage or disadvantage to a political party (preventing gerrymandering) is paramount. A different agency might decide the reverse: that redistricting should be blind to partisan bias, and compactness should be the only criteria. Both agencies can use AutoRedistrict to achieve these entirely different ends, and it will produce redistricting plans that meet their different goals. AutoRedistrict will still report on all criteria, and a third party could compare the two maps, and how they measure up on all of the criteria, and choose between them.

7 Closing Summary

While a human can generate roughly one redistricting plan a day, and evaluate it on a number of metrics in another day, automated redistricting using a heuristic search algorithm such as the Genetic Algorithm can generate and evaluates hundreds of maps per second. Furthermore, heuristic search guarantees that each iteration generates better plans than the previous. It mathematically guarantees continuous improvement, whereas successive iterations of manually-drawn maps provide no such mathematical guarantee.

AutoRedistrict provides this capability to everyone, regardless of technical expertise, free of charge, while maintaining full transparency through every step of the process.

Furthermore, the metrics that it uses to evaluate maps are mathematically sound, ensure that constitutional requirements are met, and ensure equality and the absence of bias in the result.

The resulting plans are then amenable to further evaluation and/or refinement.