The Commons: Puget Sound Journal of Politics

Volume 3 | Issue 1 Article 4

August 2022

Balancing Populations of Electoral Districts

Ethan Stern-Ellis
University of Puget Sound

BALANCING POPULATIONS OF ELECTORAL DISTRICTS

ETHAN STERN-ELLIS

PART I – INTRODUCTION

The purpose of this project is to help think about different ways of creating electoral districts in the United States. Rethinking how to draw districts can hopefully further the discussion on the problems that redistricting and gerrymandering pose to the United States. Gerrymandering is the drawing of electoral district lines to favor one group over another, which is a potential consequence of the routine process of redrawing district lines. The redrawing process happens every ten years when the US Census is carried out in order to account for population changes that may affect how populations are represented electorally. Gerrymandering is a threat to US democracy since many states allot the drawing of district lines to their state legislatures, allowing politicians who may redraw districts in order to satisfy personal or partisan objectives. One question that comes up through the process of redrawing districts, therefore, is what counts as a district that is drawn fairly.

To answer this question, one criterion that should be considered is the compactness of districts. Generally, districts should be as compact as possible. This has led non-politically motivated analysts to use a euclidean measure of space between groups of people. A euclidean distance is defined as the distance between two points using the straight-line distance formula. This kind of measure might be problematic since it does not take into account potential geographical or physical obstacles that can separate communities. For example, if the distance between group A and group B is one mile, but there is a mountain without a tunnel that would connect the two groups, then the real-world distance traveled from A to B would be much greater. A euclidean measure for the distance between A and B, therefore, is an inaccurate representation of how these groups of people would be connected by distance.

One potential solution to this misrepresentation of space is to use travel time as the measurement of distance when creating the boundaries of each district. The Spatial Models and Electoral Districting Research Experience for Undergraduates (SMED REU) of 2019 attempted this, using the state of Ohio as their model. They remodeled the districts by using multidimensional scaling (MDS) and a clustering algorithm. The clusters, in this case, are the congressional districts in Ohio. However, in Ohio, each district must satisfy two constraints: first, they must be contiguous; second, they must be balanced

by population as determined by the Ohio legislature. Contiguity in this case means that each electoral district in a cluster must touch the border of another district from the same cluster. Population balance means that each cluster's total population must be within plus or minus five percent of the average population for all clusters. In this case, that means that the range is from 684,980-757,082 people. The initial methods used by the SMED REU, which will be discussed in more depth in the next section, made it difficult to satisfy the population constraint. Building off of the work that the SMED REU completed, this project aimed to satisfy the population constraint by implementing an algorithm that clusters districts using the MDS data while balancing the populations of each district.

PART II – RELATED WORK

The SMED REU of 2019 was directly built from the work of former University of Washington professor of geography Richard Morrill. In 1972, Morrill was tasked with hand-drawing the legislative and congressional districts for the State of Washington.[1] Four years later, Morrill was able to access new computer technologies that allowed him to gather travel time data for these districts.[2] He used this data to create computer models of the districts in order to compare them to his original drawings.[3] Morrill found that, on average, the computer models created districts that were more compact with regards to population and more efficient in how they were divided.[4] The SMED REU team drew from Morrill's work in order to find comparisons between the 2010 maps of Ohio and their own computer models. Their goal was to be able to model Ohio's districts using travel time data in order to more accurately represent how Ohio residents inhabit and interact with space, allowing analysts to uncover potential problems with Ohio's existing maps.

As stated in the previous section, this project directly builds off of the work that was done by the SMED REU of 2019. In particular, the SMED REU team compiled all points to all points travel time data from Ohio using OpenStreetMap, an online mapping system. While compiling this data, the team generated a 9232 by 9232 matrix, which they then reduced to eleven dimensions using multidimensional scaling. After reducing the data, they ran the matrix through Python's K-means package, setting the initial centroids as the centroids from the 2010 US census. A centroid defined in this case is the point that is the mean distance between all of the points of a cluster. Running K-means on the MDS data generated sixteen clusters, representing Ohio's congressional districts. These clusters were almost contiguous, but not within Ohio's population requirements. In order to satisfy both Ohio's contiguity and population constraints, the SMED REU team took the data after K-means finished running and attempted to modify it to meet the constraints. They were able to satisfy the contiguity requirements, but were unable to meet the population requirements.

Researchers have tried to tackle the problem of clustering groups given certain constraint requirements. Graphically weighted regression (GWR) and automatic zoning procedures (AZP) are two techniques that exist for this task. A 2014 analysis produced by Professor of Computational Spatial Science Stewart Fotheringham of Arizona State University and others stated that, "In essence, GWR measures the inherent relationships around each regression point *i*, where each set of regression coefficients is estimated by weighted least squares."[5] The coefficients in this case would be the population constraint, and GWR would model the relationship between Ohio's travel time data and the population at a given regression point *i*. Alternatively, the AZP technique would contiguously aggregate spatial data for a given number of zones (which, in this case, would be electoral districts) into a given number of regions (which, in this case, would be congressional districts). The AZP algorithm was further developed to include equality and inequality constraints.[6] For this project, GWR was not used, since the SMED REU used K-means and this project intends to expand on the work of SMED REU. A Python package for the AZP technique was examined for this project, but it was abandoned after we found that documentation for the AZP implementation that we were considering was no longer supported. Instead, this project drew its inspiration from a 2001 paper written by Professor of Statistics Giuseppina Damiana Costanzo of University of Calabria, in which Costanzo describes a modified K-means algorithm.[7] In particular, this paper shows a version of K-means where, at each iteration of K-means, the algorithm was updated based on different constraints.[8] From this paper, we decided to implement a K-means algorithm by hand that updates the clusters created by K-means at each iteration based on the population constraint.

PART III – METHOD

K-MEANS IMPLEMENTATION

In this section, we will first discuss the K-means algorithm that we used for this project and then transition into discussing how we attempted to implement a population check at each iteration of K-means. Much of the code used for our K-means algorithm was drawn from the work of programmer Harrison Kinsley.[9] The K-means algorithm begins by picking initial centroids. There are sixteen centroids, since there are sixteen congressional districts in Ohio. The centroids that were chosen initially are the same centroids that were used in the 2010 census. The algorithm then enters a loop where two actions occur. First, the distance between each data point to each centroid is calculated. The data point is then assigned to the cluster whose centroid to which it is closest. Second, for each cluster, a new centroid is assigned based on the average distances between every data point in the cluster. These actions occur for either X number of iterations, or until the centroids do not move more than a predefined threshold that was set at .001 per Kinsley's recommendation. We tested the algorithm for correctness using a version of

unit tests, where the algorithm was broken up into small sections and verified for correctness through the use of mock data and print statements.

POPULATION CHECK

We then implemented three versions of the population check. The population check comes directly after the data points are assigned to clusters. Together, K-means and the population check make up the whole algorithm. For clarity, the entire algorithmic process will be outlined, and then the three versions of the population check will be discussed. The entire process of the algorithm is as follows:

1. Start by initializing the centroids.
2. Assign the data points to clusters.
3. Check the populations of each cluster.
4. Recalculate the centroids.
5. Repeat steps 2-4 for the given number of iterations or until the centroids do not move more than the predefined threshold.

Population Check V0 – The first version of our population check goes through a fairly straightforward process involving two steps. First, out of all the clusters that were computed at the previous iteration of K-means, the check identifies the cluster with the largest population. Let us call this cluster C. Then, the check reduces the population size of cluster C until it is within the population threshold required by the State of Ohio. It does this by moving the data points that are farthest away from the centroid to the cluster they were second closest to when they were originally assigned to cluster C.

Population Check V1 – The second version of our population check does the same process as the first version, barring one modification. In this version, when picking the largest population, all the clusters that were modified in the previous iteration of K-means are excluded from selection. For example, suppose that cluster C1 is picked as the cluster with the largest population in the first population check. Then, suppose that data points are moved from C1 to clusters C2 and C3. In the second population check, clusters C1, C2, and C3 would then be excluded from selection as the cluster with the largest population.

Population Check V2 – The third version of our population check again does the same process as the first version, barring one modification. This time, after the cluster with the largest population is selected, the population reduction ends after one of two conditions becomes true: either the population is reduced to the required size, or a given number of data points are moved. When we tested, we used either 50 or 100 data points as the maximum number for points that could be moved.

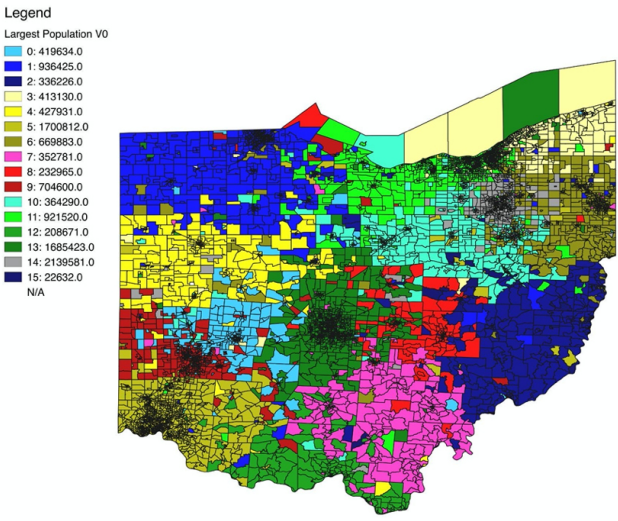

Figure 1: The results of algorithm Population Check V0 after 1 iteration.

PART IV – RESULTS AND ANALYSIS

The discussion of the results of the algorithm will begin by showing the results of Population Check V0 after several different iterations of the algorithm. We will begin by showing the results after one iteration.

The first map, Figure 1, shows the results of Population Check V0 after one iteration. It is useful to present this map as it provides a baseline for comparison to the second and third maps. The legend format follows a specific pattern: from left to right, each entry begins with a color, the cluster that is associated with that color, and the total population for that cluster.

The second map, Figure 2, shows the results of Population Check V0 after 300 iterations of the algorithm. Compared to the first iteration, 9 out of the 16 clusters got closer to the population threshold required by the State of Ohio. This was a promising result, so we tested the algorithm after 3000 iterations. The third map, Figure 3, shows the results of Population Check V0 after 3000 iterations of the algorithm. Compared to 300 iterations, 9 out of the 16 clusters again got closer to the population threshold, but with significantly lower returns.

This prompted us to examine the output between iteration 300 and iteration 3000. A worrying pattern emerged. During the first iteration, a cluster with the largest population passed some of its data points to a cluster with a smaller population, thus reducing the size of the largest cluster's population. This was expected. The problem was revealed during the second

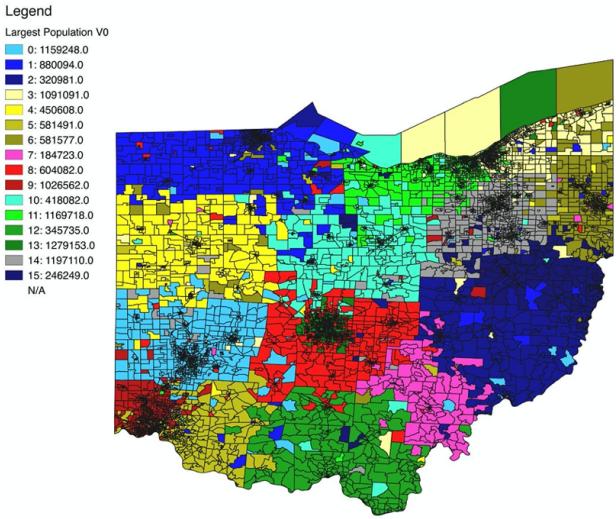

Figure 2: The results of algorithm Population Check V0 after 300 iterations.

iteration. The cluster with the smaller population, instead of the cluster with the largest population, was selected as the cluster with the largest population. Consequently, the cluster with the smaller population passed some of its data points back to the cluster with the largest population. Between iterations 300 and 3000, we noticed that this process happened continuously.

To try to tackle this problem, we implemented Population Check V1 and Population Check V2 respectively (see Part III). For Population Check V1, the same problem of population swapping between two clusters arose. For Population Check V2, reducing the number of data points that could be moved proved to be problematic as well. In particular, this prevented the centroids from moving very much after they were recalculated in Step 4 of the algorithm, ending the overall algorithm before the populations could be balanced.

PART V – CONCLUSION

Clearly, the results from this algorithm do not accomplish the task of balancing the populations of each cluster to meet the requirements of the State of Ohio. However, the idea of having a K-means algorithm that modifies the populations of clusters at each iteration is still promising, as there are other methods of checking the population that can be implemented. The question that remains is whether or not this is an improvement from using Python's K-means package before modifying the data to meet the constraints. We believe that it is, as this method gives the programmer more freedom in addressing the problem of checking population balance and contiguity. Even if checking

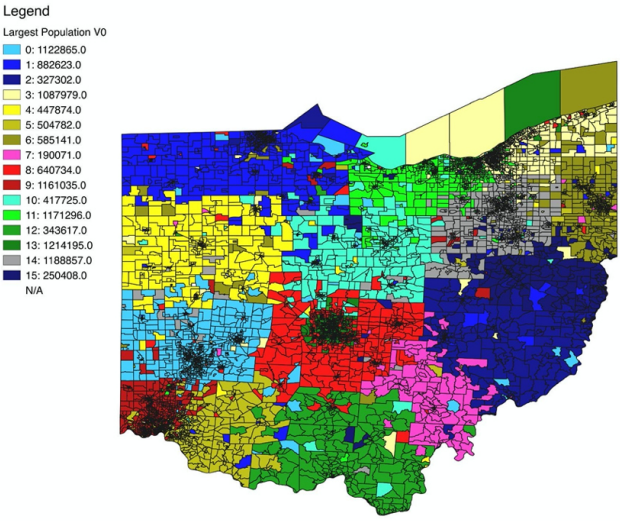

Figure 3: The results of algorithm Population Check V0 after 3,000 iterations.

the population at each iteration of K-means is a flawed way to approach this problem, having a baseline K-means algorithm allows the researcher to change directions with relative ease instead of being forced to modify data after K-means has run its course.

PART VI – FUTURE WORK

The next steps for this project are twofold. First, instead of excluding clusters from the previous iteration the way that the Population Check V2 does, it may be useful to exclude specific data points. This should be the next change to the current algorithm. Second, future work should involve implementing a contiguity check. We imagine that this would occur at each iteration as is the case for the population check.

PART VII – ACKNOWLEDGMENTS

We would like to acknowledge support for this project from Dr. Courtney Thatcher, Dr. Jim Thatcher, and Dr. America Chambers. Dr. Courtney Thatcher and Dr. Jim Thatcher were both crucial mentors and provided all of the materials for this project. Dr. Chambers was an incredible mentor and leader of our capstone class. All three of these professors made this work possible. We hope that the SMED REU, which is headed by Dr. Jim Thatcher and Dr. Courtney Thatcher, will have success in their endeavors.

END NOTES

1 Richard Morill, "Redistricting Revisited," *Annals of the Association of American Geographers* 66 (4): 548–556, 1976, 548.

2 Morril, 548.

3 Morril, 548.

4 Morril, 552–553.

5 Stewart Fotheringham, Martin Charlton, Paul Harris, and Bibin Lu, "Geographically Weighted Regression: The Analysis of Spatially Varying Relationships," *International Journal of Geographical Information Science* 28 (4): 660–81, 2014, 661.

6 S. Openshaw, and L. Rao, "Algorithms for Reengineering 1991 Census Geography," *Environment and Planning A* 27 (3): 425–446, 1995, 427.

7 G. Damiana Costanzo. "A Constrained K-Means Clustering Algorithm for Classifying Spatial Units." *Statistical Methods and Applications* 10: 237–256, 2001, 237.

8 Costanzo, 243-245.

9 Harris Kinsley, "K-means from Scratch in Python." Python Programming, https://pythonprogramming. net/k-means-from-scratch-machine-learning-tutorial/.

BIBLIOGRAPHY

Costanzo, G. Damiana. "A Constrained K-Means Clustering Algorithm for Classifying Spatial Units." *Statistical Methods and Applications* 10: 237–256, 2001, 237.

Fotheringham, Stewart, Martin Charlton, Paul Harris, and Bibin Lu. "Geographically Weighted Regression: The Analysis of Spatially Varying Relationships." *International Journal of Geographical Information Science* 28 (4): 660–81, 2014.

Kinsley, Harris. "K-means from Scratch in Python." Python Programming. https://pythonprogramming. net/k-means-from-scratch-machine-learning-tutorial/.

Morril, Richard. "Redistricting Revisited." *Annals of the Association of American Geographers* 66 (4): 548–556, 1976.

Openshaw, S., and L. Rao. "Algorithms for Reengineering 1991 Census Geography." *Environment and Planning A* 27 (3): 425–446, 1995, 427.