# Studying Virginia Voting Pattern with Persistent Homology: Summer 2020 Report

Jiaying Chen

## Introduction

The project aims to quantify the 'voting islands' in Virginia. A 'voting island' refers to an area where residents' voting behaviors strongly disagree with its surrounding areas. Due to demographic properties in urban and rural areas, a Democratic dominant precinct can be surrounded by several Republican dominant precincts, or less commonly vice versa. Recent research suggests that people tend to move to neighborhoods with political opinions similar to their own, especially in urban areas. If we can pinpoint those voting islands and quantify that differences, social scientists might be interested in studying the formation of these areas and trends of political polarization. The changes in size and intensity of political islands may signal the trends of demographics and public opinions. The quantitative methods can also be applied to other social science problems such as detecting income and racial segregation in urban areas and potentially contribute to the debate of identifying gerrymandering. The developed techniques can be applied to any dataset where areas are connected by some characteristics and holes are of interest.

To quantify voting islands, Dr.Feng describes her simplicial persistence homology methods (i.e. the adjacency method and the level-set method) to quantify voting islands in her paper *Persistent Homology of Geospatial Data: A Case Study with Voting* [1], which lies as the starting point of this project. This summer project consists of two parts: 1) replicating Dr.Feng's adjacency method and 2) exploring a cubical approach. The cubical approach should enjoy both the voting-strength interpretation in the adjacency method and the visual contiguity of the map in the level-set method by constructing a sub-level-set filtration.

## Data

I used a precinct-level dataset that contains voting results and administrative boundary shapes for Democratic and Republican candidates in the 2016 presidential election in Virginia. The dataset is from Metric Geometry and Gerrymandering Group[2] and it is preprocessed in QGIS[3] for further analysis.

## Method: Simplicial

I built the simplicial complexes with the adjacency method described in Dr.Feng's paper (p. 6-8) and calculated the persistent homology using Perseus[4]. I made the modification to keep adding blue precincts by sequence when all red precincts are added. The purpose was to distinguish between a dark blue island and a light blue island. In Feng's paper, she evaluates the effectivity of the adjacency method by visualizing the identified loops on maps and checking if a loop is a meaningful feature. She made the visualization by modifying the source code of PHAT[5] to scrape off the generators of a hole. However, Feng did not publish the modified PHAT package. I tried to visualize detected loops with a software package called Persloop[6], but it sometimes draws larger loops than expected and there is also a mismatch between the Persloop results and the persistence diagram generated by Perseus.

## Method: Cubical

The cubical approach starts with the converting the precinct-level voting data into greyscale rasters, with pixel values ranging from 0 to 255. Cells within the boundary of a precinct share the same value. Pixel values are calculated as

$$128 + 127 * m$$

, where $m$ is Clinton's margin of victory ranging normalized to (-1,1). Thus, Trump-winning precincts have lower pixel values (i.e. darker color) and Clinton-winning precincts have higher pixel values (i.e.. lighter color). Out-of-bound areas are assigned with 255 (i.e. white). The persistence homology can be computed by Perseus, with the re-formated raster matrix as the input. I tested different resolutions for greyscale maps and examined the corresponding betti diagrams and persistence diagrams. As shown in Figure1(a), when the resolution drops, a precinct can break into two holes. Unnecessary holes are also created when polygons are converted to rasters (Figure 1(b)).

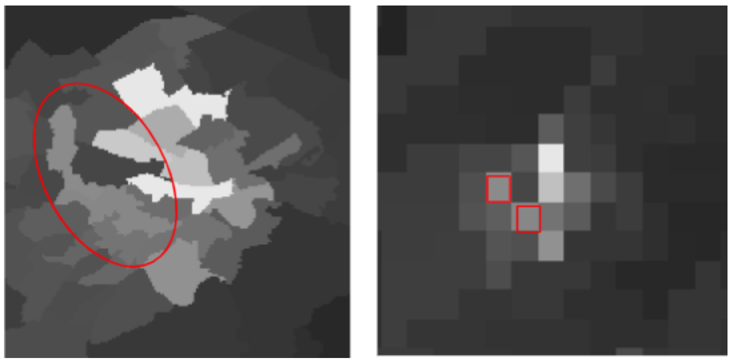
Figure 1(a): A few voting islands in Segment 12 with the resolution of 8184p (left) and 256p (right).

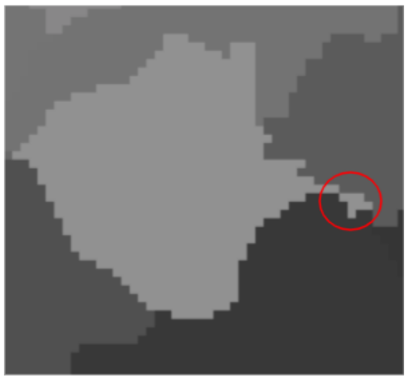
Figure 1(b): Rasterized precinct shape (8184p).

I also compared randomly generated graphs with the real data (Figure 2). We can see that the betti numbers reach to around 30 and 50 in Figure 2(a) and Figure 2(b) and it seems like the grouping of cells by precincts counts for the small magnitude of betti numbers. We can also see that Figure 2(a) and Figure 2(c) both reach the peak of $\beta_1$ before all red areas are added, possibly indicating that the proportion of red areas is responsible for the peak of $\beta_1$.

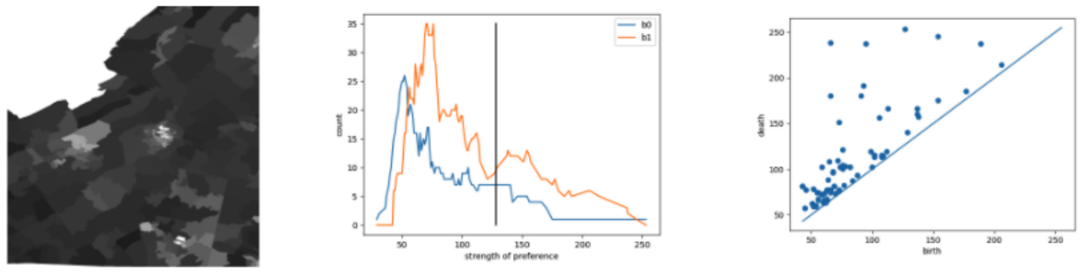

Figure 2(a): real data (Segment 12) (8184p)

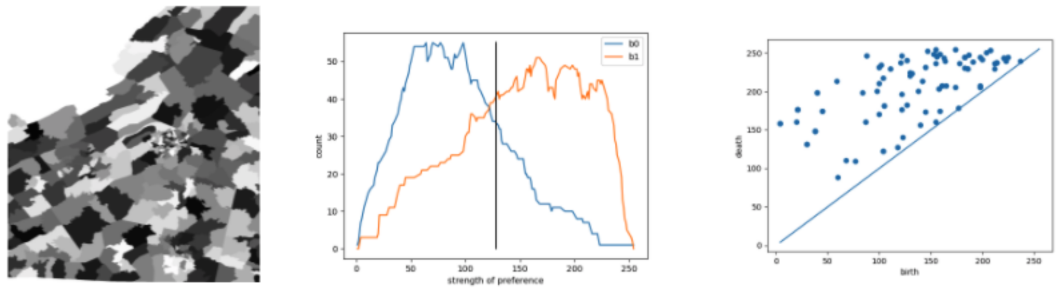

Figure 2(b): Values randomly assigned by precinct (Segment 12)(8184p)

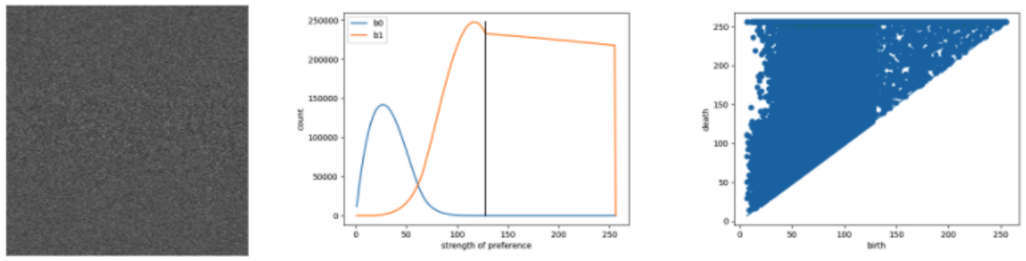

Figure 2(c): Values randomly assigned by pixel, with the same number of pixels$< 128$ as in the real data (Segment 12)(8184p)

## Future Steps

The focus of my future work would be primarily on developing a cubical method that identify and quantify voting islands. To achieve this goal, we need to define a measurement to determine whether the cubical method can precisely identify voting islands. An intuitive way would be to visualize identified features on map. However, this might not be feasible for a relatively small project. As mentioned before, Persloop sometimes yields undesirable results. Even when the mismatch is fixed, given the functionalities it provides, it is difficult to incorporate Persloop in a automatic procedure to generate a large number of visualizations. If I keep working on this direction, it is unavoidable to dive into the algorithm of scraping generators and modifying the C++ source code of a persistence homology package. Even if I were able to create the visualizations, it would be a tedious work to manually check every single feature. Thus, doing the visualizations on maps can be a huge amount of work.

Another evaluation method that I come up with, while not been implemented yet, is that for each precinct, we crop the raster by the same size of its polygon and and pad pixels around. This is illustrated in Figure 3, where a 6*6 raster is cropped by the height and width of a precinct and it is padded into a 8*8 raster. Then we can check if there is only one connected component in this piece of raster data. A Python package called GeoPandas can be used to convert the polygon of precincts and their attributes into data frames.

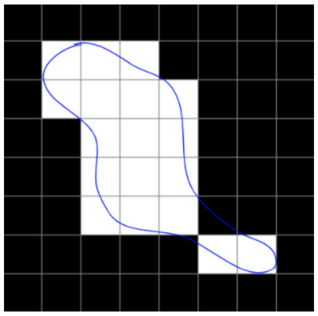

Figure 3: illustration of the evaluation method, where the blue polygon represents the boundary of a precinct and the white cells represent the rasterized precinct.

Once the evaluation method works, we can explore a decent way to lower the resolution, which preserves the connectivity within a precinct (i.e. the pixels within a precinct should only be one connected component in the input raster for Perseus). Once this is achieved, we might further explore the betti numbers and the persistence diagrams and see how they reveal patterns in voting data.

An additional goal might be to properly incorporate the population data when necessary. The value of population data I see now is that it removes some large water and mountain areas, which can affect the number of voting islands. As shown in Figure 4(a), there is a light hole at the lower left corner of the left image, but the hole disappears in the right image where waterways are removed. Such differences rarely occur in my dataset, and generally I find it less important to incorporate population data, since the precinct-level is the highest granularity available for voting data and we are not looking at features smaller than a precinct. However, adding population data can be meaningful for other socioecomonic datasets at a higher granularity, so it could be worthwhile to explore methods for incorporating population data. The problem with population data is that it introduces noises to the map (see Figure 4(b)). Possible solutions might be to let the noises disappear when surround areas are filtered, or to include only significant unpopulated areas. Currently we are thinking about only removing areas where population equals zero, but we might also use the varying population density depending on the purpose of the project. Another way to look at the problem may be to use population grid as the unit of analysis, and averaging the voting result in a cell. Overall, it remains unclear how I can properly use the population data and I will give this task a lower priority due to the purpose of this specific project.

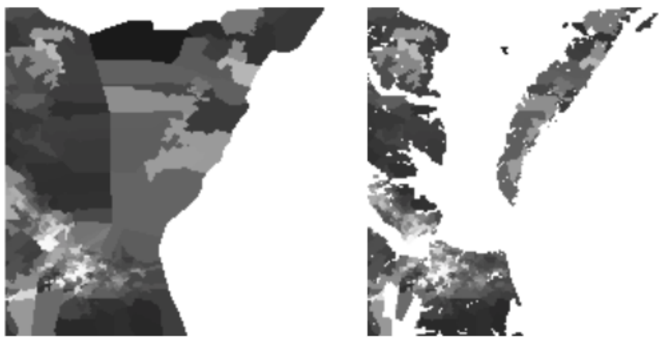

Figure 4(a): voting map vs. voting map with unpopulated areas removed

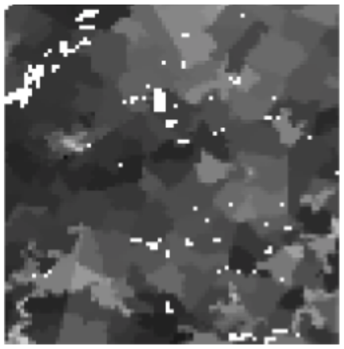
Figure 4(b): unnecessary holes appears when unpopulated areas are removed

## Appendix

### Code

All the scripts are in */Downloads/TDA/scripts*, where *./simplicial* contains the code for the simplicial method and ./cubical has the code for the cubical method. Future works would primarily involve the scripts in ./cubical. Within this folder, note that *cubical_raster.py* should be run in QGIS.

### Important Data Files

original data (converted into EPSG4326): /Users/jiaying/Downloads/TDA/data/va_data_MGGG/VA-shapefiles-master/VA_precincts/va_precincts_epsg4326.shp
8184p raster converted from shapefile: /Users/jiaying/Downloads/TDA/cubical/8184p/virginia8184p.tif
8184p raster with unpopulated areas removed: /Users/jiaying/Downloads/TDA/cubical/8184p/pop_8184p/virginia8184p_final.tif

## References

[1] Persistent Homology of Geospatial Data: A Case Study with Voting. Access online: https://arxiv.org/abs/1902.05911

[2] mggg-states/VA-shapefiles. (2019, May 13). GitHub. https://github.com/mggg-states/VA-shapefiles

[3] QGIS 3.4. Access online: https://qgis.org/en/site/

[4] The Perseus Software Project for Rapid Computation of Persistent Homology. Access online: http://people.maths.ox.ac.uk/nanda/perseus/

[5] Persistent Homology Algorithm Toolkit. Access online: https://pythonhosted.org/phat/

[6] PersLoop: Persistent 1-Cycles: Definition, Computation, and Its Application. Access online: http://web.cse.ohio-state.edu/ dey.8/PersLoop/