Fall 2019

Final Project Report: Automating Creation of Voting Districts

William Austin *CS 633 Computational Geometry*

Abstract

Gerrymandering poses a significant threat to the fairness of our elections by creating artificial advantages for one political party over another. The effect of this practice is easy to understand just by looking at the highly irregular districts produced, clearly engineered to include or exclude certain neighborhoods. In recent years, there has been significant interest in improving the process of how these boundaries are drawn, and the movement to create districts that are compact and impartial is gaining bipartisan support.

Starting with information from the US census, we can use a data-driven approach that will employ algorithmic techniques to produce maps that are based solely on population distribution. This results in maps with that have favorable characteristicts and can be generated at a low cost and without political bias. The method that we focus on is based on 'balanced centroidal power diagrams', which is related to the k-means clustering algorithm. We also identify a few metrics that can be used to quantitatively analyze the overall quality of districts so we can differentiate between one that is simple, compact, and contiguous, as opposed to one that is more complicated and likely to be gerrymandered. Our work is focused on maps for the state of Virginia, for both the state legislatures and the US House of Representatives.

1 Motivation and Introduction

In representative democracies, running fair elections that represent the will of the people is of critical importance. Gerrymandering undermines this effort by creating a set of maps that do not achieve this goal. However, as we will see, the practice is extremely common because the politicians that are in the current majority are usually also the ones that approve the maps. This leads to the common description of "politicians picking their voters instead of voters picking their politicians". Therefore, this phenomenon is extremely hard to prevent, despite the fact that most people can identify a gerrymandered map with relative ease. As an example, the map below shows the current Virginia Congressional districts in northern Virginia. Congressional District 11 is good example of a non-compact district that has likely been modified for a political advantage.

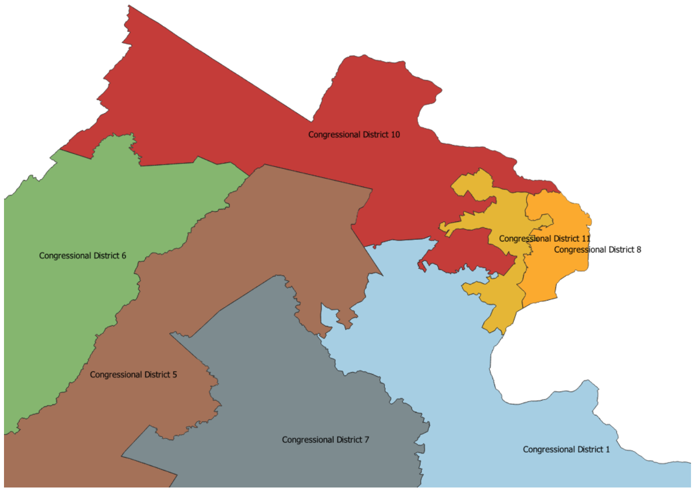


Figure 1: Current Northern Virginia Congressional Districts

Another example comes from southern Virginia, where the State Senate Districts 15 (brown) and 20 (green) look like a yin and yang symbol.

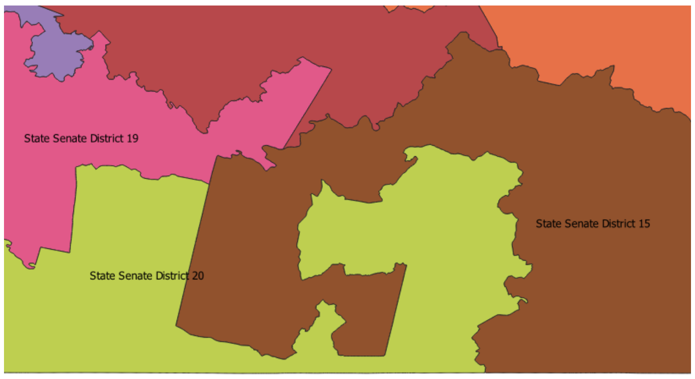


Figure 2: Current Southern Virginia State Senate Districts

1.1 Brief History of Gerrymandering

The term "gerrymandering" comes from the an 1812 political cartoon in the Boston Gazette satirizing the map signed into law by then Governor Elbridge Gerry. The cartoon relates the shape of the districts to that of a salamander, which led people to refer to the map as the "Gerry-mander". Subsequently, this phrase was shortened to gerrymander [13].

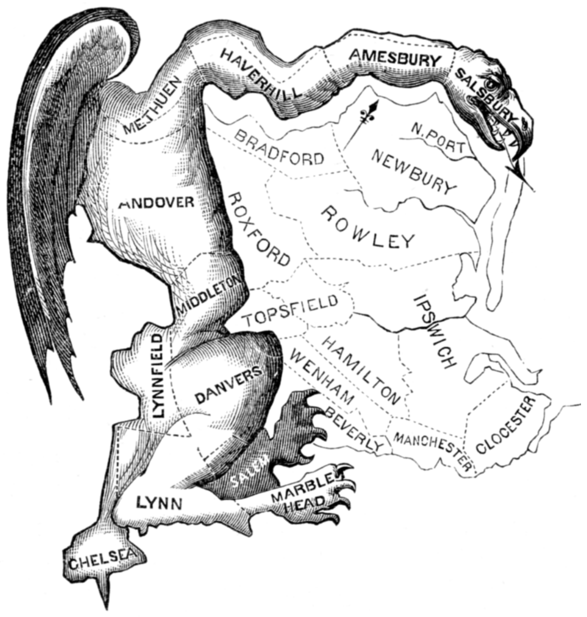


Figure 3: The Gerry-mander

Since this original instance, gerrymandering has spread around the globe and there have been documented examples of the practice in more than 25 countries[13].

1.2 Recent Developments

Clearly, the practice of gerrymandering has been around for a long time, but the last 20 years have seen it become much more common. In addition, as the effectiveness of these techniques have become evident, with REDMAP [14], we have seen efforts at a national party level to make gerrymandering a core component of the long term political strategy. This practice is sometimes referred to as "cracking and packing" because a political organization wishes to organize districts so that opposing voters are split up, or "cracked" among districts that they have an advantage in, or consolidated ("packed") into districts that they are likely to lose.

In addition to the effectiveness of previous applications, there are a few other factors that explain why this practice has become so common:

- Running for public office and getting elected requires incredibly large amounts of money and effort. Maintaining a campaign staff, holding events, and purchasing advertisements are all required activities for winning an election, but not directly related to the role of a legislator. Therefore, there is an incredible incentive for government officials to rig the system to make re-election efforts easier.
- In most states (36 out of 50) [13], the redistricting process is controlled by the state legislature, which is clearly going to be biased in favor of maps that keep the currently elected officials in office.

- More data is available than ever about how voters are distributed. By looking at poll results, historical data, and demographic trends, legislators can use gerrymandering systematically makes elections less competitive. This reduces the number of neighborhoods that will need to be targeted for a successful campaign, saving time and costs.
- Improved hardware makes analysis of all this data feasible in a reasonable amount of time. In addition, there are now more sophisticated software tools that can be used to determine how to most effectively alter district boundaries for political. In this paper, we explore ways of using algorithms and software to draw better districts, but deploying technology for political gain has been around much longer.

There are a few different strategies for dealing with this problem. Some states, including Virginia, are considering or have made structural changes that place the burden of drawing maps on an independent redistricting commission. The leading group supporting this type of reform is a bipartisan group called OneVirginia2021, with the long term goal of passing an amendment to the Virginia Constitution that maps describes how the redistricting process would work. An emphasis would be placed on process transparency, public involvement and removing the political influence of the state legislature [6].

Another national bipartisan group that is pushing for reform is FairVote. Their agenda includes gerrymandering reform as well as other structural changes in the voting process, such as Ranked Choice Voting. In 2011, FairVote sponsored a public competition to redraw the Virginia Congressional districts [5].

The most influential individual over the past decade in the area of programmatic creation of district boundaries is Brian Olson. He has created used data from the 2010 census to create new maps for almost every legislature in all 50 states. In addition, his site contains a collection of links to many other resources related to redistricting [3]. Olson is also very engaged trying to raise public awareness and is considered an expert in the field. His 2016 TED talk about the gerrymandering problem and his proposed solution has been viewed by thousands [2]. The image below shows his generated map for Virginia Congressional districts.

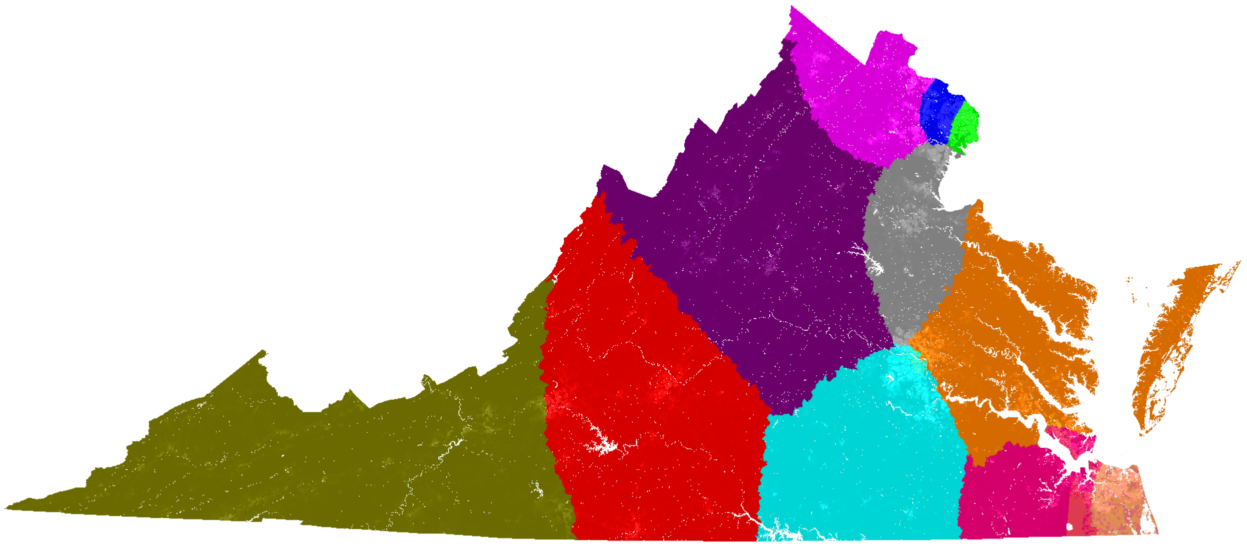


Figure 4: Proposed Virginia Senate Districts by Brian Olson

1.3 Overall Research Strategy

The inspiration for this project was based on the 2018 paper, “Balanced power diagrams for redistricting” [4]. This paper provides the algorithmic basis for drawing maps with desirable properties that could represent legislative districts. More specifically, their criteria includes convexity, equal population distribution across regions, and shapes that have no more than six sides. Having these guaranteed properties is critical for producing fair and compact districts.

Because the original authors did not include sample maps for Virginia in their original paper, my primary goal was to generate these maps, using their methodology. However, I also wanted to enhance the work of the original authors by performing an additional postprocessing step that creates more useful maps that respect existing political boundaries, rather than the straight line Voronoi cells produced by their algorithm. Lastly, I wished to calculate some of the metrics proposed by Olson [1] and others [16]. Therefore, to summarize, the three main activities I performed were:

1. Reproduce the work described in the paper by Cohen-Addad, Klein, and Young. Use the software available in their Bitbucket repository and run it on geospatial and demographic data collected for the state of Virginia. Generate maps for:
    - Virginia Congressional Districts (11 districts)
    - Virginia State Senate Districts (40 districts)
    - Virginia House of Delegates Districts (100 districts)
2. Expand the set of maps produced in the first step to make them more useful by updating electoral district boundaries to respect census block divisions.
3. Investigate metrics for compactness and create scripts to perform calculations. These metrics should be calculated for both the old and new maps and compared. This establishes an argument for why this automated method of district creation may lead to better results than the traditionally hand-drawn districts.

1.4 Technology and Software Tools Utilized

Many different software products and technologies are required to perform the processing and analysis required for this project. Below, I give an overview of the main tools used, and what the function of each was.

Python 3.7.2 Python is used extensively in the GIS (geographic information systems) domain and has excellent libraries that support common geospatial operations. In addition, Python is an extremely powerful general purpose scripting language, which makes it valuable for many other required tasks. In this project, the original authors used Python to format input and output, as well as to verify results and generate plotting data. Likewise, I used Python extensively to read and write shapefiles, aggregate data, compute metrics, and generate output files. Some of the critical Python libraries that I utilized were:

- **PyShp 2.1.0**. This library makes it easy to read and write shapefiles, with support for operations like streaming and attribute retrieval.
- **Shapely 1.6.4** Shapely is a library that supports many geometric operations. For this project, we can create a Shapely polygon from the district definition and Shapely will perform operations such as centroid location, perimeter, area, and convex hull.
- **PyProj 2.4.2** This library allows us to read in data that is specified with a particular datum and CRS (coordinate reference system), and it will give us projected plane coordinates that we can use to perform geometric operations on.

C++ 11 The algorithm provided by the paper's author has been implemented in C++. This code allows us to compute the desired balanced centroidal power diagram. I was able to clone their repository and run a successful build to create the desired executable, but a few minor changes to their original source code were required.

QGIS 3.10.0 QGIS is a general purpose tool for viewing, analyzing, and editing geospatial data. In this project, the main purpose was for modifying shapefiles. Some of the key operations supported are:

- Geospatial joins, which allow us to merge data from different maps together, based on a common key attribute. In this project, we added the census block population data from a spreadsheet to the shapefile as an attribute by using this feature.
- Centroid calculations, which allow us to compute the centroid of a district or census block. This is useful because the balanced centroidal power diagram algorithm requires single points as input, so we represent census blocks by their centroid. In addition, the average distance metrics depend on knowing what the district centroid is.
- Unions and intersection operations. We can use these capabilities to transform the Voronoi cells in the algorithm output into a valid shapefile by retrieving the intersection with the Virginia state shapefile and the removing unwanted features. In addition, we can merge census blocks into a single region to produce our final modified output.

- Styling maps with colors, labels, and transparency to generate images for our output.

Gnuplot 5.2 This tool was used by the original authors to generate visual output for the generated maps.

1.5 Project Artifacts

All code and project aritifacts are avaiable on my public GitHub account. The URL is:

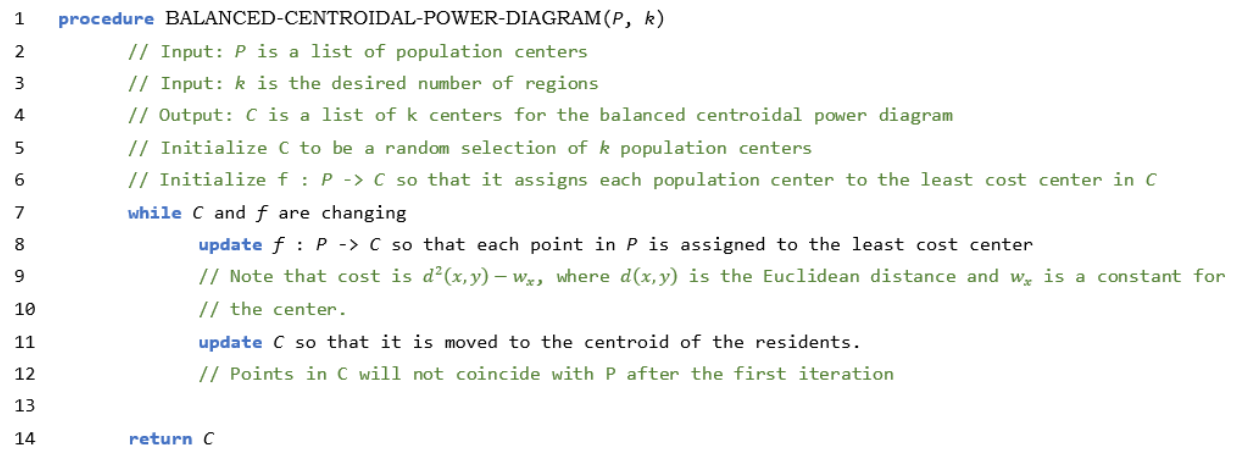
https://github.com/william-r-austin/district2

2 Methods Used and Studied

In this section, we will describe the algorithms and processes needed to produce our output. In addition, we give the mathematical basis for our metrics.

2.1 Balanced Centroidal Power Diagrams

This algorithm is the basis for our work. As described by the authors of the original paper [4], this is an iterative algorithm that repeatedly assigns population points to districts based on a weighted distance, and then updates the center of the district according to these assignments. This algorithm, like other ones similar to it, must balance the shape of the produced districts and how compact they are with the how evenly the population is distributed. In this case, the authors use a balanced power diagram, which guarantees equal population distribution, and by requiring district centers to coincide with district centroids, they produce Voronoi cell shaped districts, which must be compact. The high-level logic pseudocode for the method is given here.



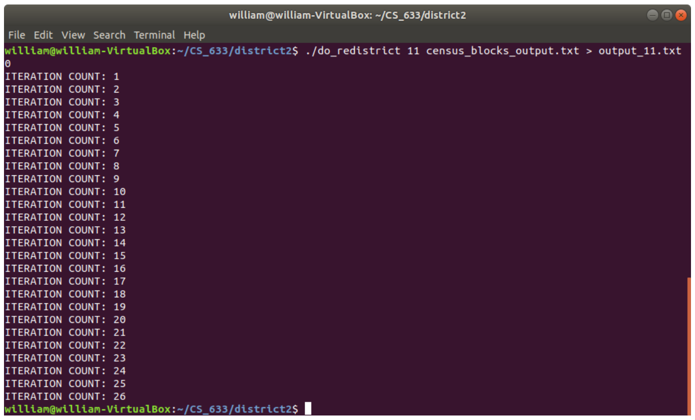
```
procedure BALANCED-CENTROIDAL-POWER-DIAGRAM($P$, $k$)
    // Input: $P$ is a list of population centers
    // Input: $k$ is the desired number of regions
    // Output: $C$ is a list of k centers for the balanced centroidal power diagram
    // Initialize C to be a random selection of $k$ population centers
    // Initialize f : $P$ -> $C$ so that it assigns each population center to the least cost center in $C$
    while $C$ and $f$ are changing
        update $f$ : $P$ -> $C$ so that each point in $P$ is assigned to the least cost center
        // Note that cost is $d^2(x,y) - w_x$, where $d(x,y)$ is the Euclidean distance and $w_x$ is a constant for
        // the center.
        update $C$ so that it is moved to the centroid of the residents.
        // Points in C will not coincide with P after the first iteration

    return $C$
```

Figure 5: Pseudocode for Computing Balanced Centroidal Power Diagrams

Note that this problem can be transformed to a version of the transshipment problem which can be solved using minimum-cost flow algorithms [15]. This makes the solution much more efficient a practical than a brute force approach.

In the authors code, the main logic for iteratively choosing the centers is implemented in *redistrict.cpp*. The mincostflow.cpp class is a minimum cost flow solver that was included to efficiently compute the assignments to each district center. The image below shows an example of running the application and this particular example takes about 4 seconds to run. In addition to the console output, a file is produced that contains the exact district centers, as well as the district assignments for all of the input points.


william@william-VirtualBox: ~/CS_633/district2
File Edit View Search Terminal Help

```
william@william-VirtualBox:~/CS_633/district2$ ./do_redistrict 11 census_blocks_output.txt > output_11.txt
0
ITERATION COUNT: 1
ITERATION COUNT: 2
ITERATION COUNT: 3
ITERATION COUNT: 4
ITERATION COUNT: 5
ITERATION COUNT: 6
ITERATION COUNT: 7
ITERATION COUNT: 8
ITERATION COUNT: 9
ITERATION COUNT: 10
ITERATION COUNT: 11
ITERATION COUNT: 12
ITERATION COUNT: 13
ITERATION COUNT: 14
ITERATION COUNT: 15
ITERATION COUNT: 16
ITERATION COUNT: 17
ITERATION COUNT: 18
ITERATION COUNT: 19
ITERATION COUNT: 20
ITERATION COUNT: 21
ITERATION COUNT: 22
ITERATION COUNT: 23
ITERATION COUNT: 24
ITERATION COUNT: 25
ITERATION COUNT: 26
william@william-VirtualBox:~/CS_633/district2$
```

Figure 6: Sample Execution of the `do_redistrict` Application with k = 11

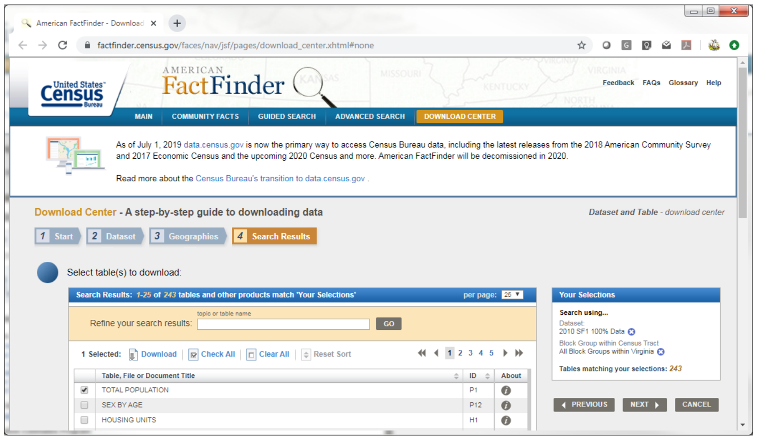
2.2 Methodology for Map Creation

Now, we will give an overview of the all the steps needed to generate a map based on population data. This process must integrate data from many different sources and we will explain in more detail how the different tools that we mentioned earlier are used.

Step 1. Data Collection for Virginia The first step for creating maps is to collect the required geospatial information and population data for the state. The most commonly used file type for representing a this information is the shapefile (SHP) format, and there are many different sources that we can use to obtain these and not all of them give us the information that we need. Below is a list of the data sources used for this project.

- The final census block data and VA State Senate district shapefiles came from the Census bureau website [8] [7]. These shapefiles distributed by the Census bureau are called TIGER files because they are stored in the Topoligically Integrated Geographic Encoding and Referencing database.
- The population data came from the American Fact Finder website [9]. Understanding the layout of the data sources and creating a query to return the data that we were looking for took some time. Understanding the hierarchy of geographic entities and how they are related is critical to being able to efficiently retrieve data. Many of the conventions rely on the

listof LSAD (legal/statistical area description) codes. However there are plenty of resources available on the Census site that describe these relationships[10]. The final search criteria and resulting CSV file are shown below.



As of July 1, 2019 data.census.gov is now the primary way to access Census Bureau data, including the latest releases from the 2018 American Community Survey and 2017 Economic Census and the upcoming 2020 Census and more. American FactFinder will be decomissioned in 2020.

Read more about the Census Bureau's transition to data.census.gov .

Download Center - A step-by-step guide to downloading data

Dataset and Table - download center

Select table(s) to download:

Search Results: *1-25* of *243* tables and other products match 'Your Selections' per page: 25

1 Selected:

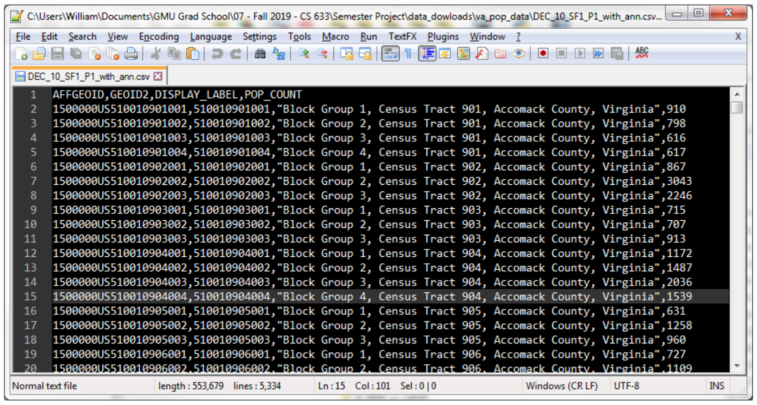
	Table, File or Document Title	ID	About
✓	TOTAL POPULATION	P1	
	SEX BY AGE	P12	
	HOUSING UNITS	H1	

Your Selections

Search using...
Dataset:
2010 SF1 100% Data
Block Group within Census Tract
All Block Groups within Virginia

Tables matching your selections: *243*

Figure 7: American Fact Finder Population by Block Group Search


C:\Users\William\Documents\GMU Grad School\07 - Fall 2019 - CS 633\Semester Project\data_dowloads\va_pop_data\DEC_10_SF1_P1_with_ann.csv...
File
Edit
Search
View
Encoding
Language
Settings
Tools
Macro
Run
TextFX
Plugins
Window
?
DEC_10_SF1_P1_with_ann.csv

```
AFFGEOID,GEOID2,DISPLAY_LABEL,POP_COUNT
1500000US510010901001,510010901001,"Block Group 1, Census Tract 901, Accomack County, Virginia",910
1500000US510010901002,510010901002,"Block Group 2, Census Tract 901, Accomack County, Virginia",798
1500000US510010901003,510010901003,"Block Group 3, Census Tract 901, Accomack County, Virginia",616
1500000US510010901004,510010901004,"Block Group 4, Census Tract 901, Accomack County, Virginia",617
1500000US510010902001,510010902001,"Block Group 1, Census Tract 902, Accomack County, Virginia",867
1500000US510010902002,510010902002,"Block Group 2, Census Tract 902, Accomack County, Virginia",3043
1500000US510010902003,510010902003,"Block Group 3, Census Tract 902, Accomack County, Virginia",2246
1500000US510010903001,510010903001,"Block Group 1, Census Tract 903, Accomack County, Virginia",715
1500000US510010903002,510010903002,"Block Group 2, Census Tract 903, Accomack County, Virginia",707
1500000US510010903003,510010903003,"Block Group 3, Census Tract 903, Accomack County, Virginia",913
1500000US510010904001,510010904001,"Block Group 1, Census Tract 904, Accomack County, Virginia",1172
1500000US510010904002,510010904002,"Block Group 2, Census Tract 904, Accomack County, Virginia",1487
1500000US510010904003,510010904003,"Block Group 3, Census Tract 904, Accomack County, Virginia",2036
1500000US510010904004,510010904004,"Block Group 4, Census Tract 904, Accomack County, Virginia",1539
1500000US510010905001,510010905001,"Block Group 1, Census Tract 905, Accomack County, Virginia",631
1500000US510010905002,510010905002,"Block Group 2, Census Tract 905, Accomack County, Virginia",1258
1500000US510010905003,510010905003,"Block Group 3, Census Tract 905, Accomack County, Virginia",960
1500000US510010906001,510010906001,"Block Group 1, Census Tract 906, Accomack County, Virginia",727
1500000US510010906002,510010906002,"Block Group 2, Census Tract 906, Accomack County, Virginia",1109
```

```
Normal text file    length : 553,679    lines : 5,334    Ln : 15    Col : 101    Sel : 0 | 0    Windows (CR LF)    UTF-8    INS
```

Figure 8: Format of Downloaded 2010 Census Population Data

- The Virginia Congressional Districts also came from the `data.gov` website, which is also run by the Census bureau [11].
- Lastly, the shapefile for the Virginia House of Delegates came from the Virginia Roads website, which is run by the Virginia Department of Transportation (VDOT) [12]. An example of what the raw shapefile looks like in QGIS is shown here. The other raw shapefiles are similar.

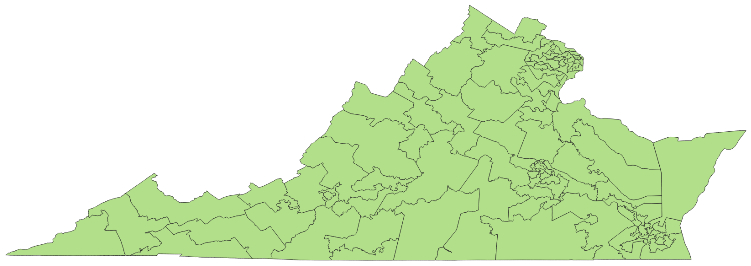


Figure 9: Raw Virginia House of Delegates Shapefile

Step 2. Format Population and Census Block Data for Algorithm In order to compute the balanced centroidal power diagram, we must provide a set of points and their associated weights. In our case, these points will be the latitude and longitude for the centroids of the 5301 census blocks in Virginia, and the weight will simply be the population for that census block.

To create this required input file, we start by calculating the centroids for each block in QGIS and adding the latitude and longitude of this point as the `XCOORD` and `YCOORD` feature attributes. The calculated centroids are shown below.

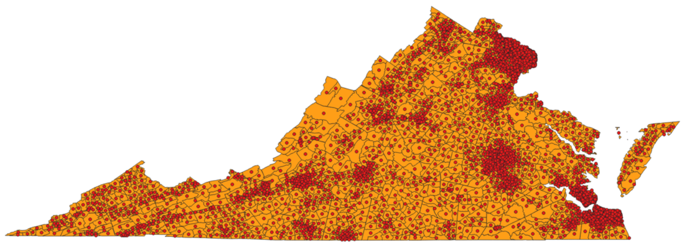
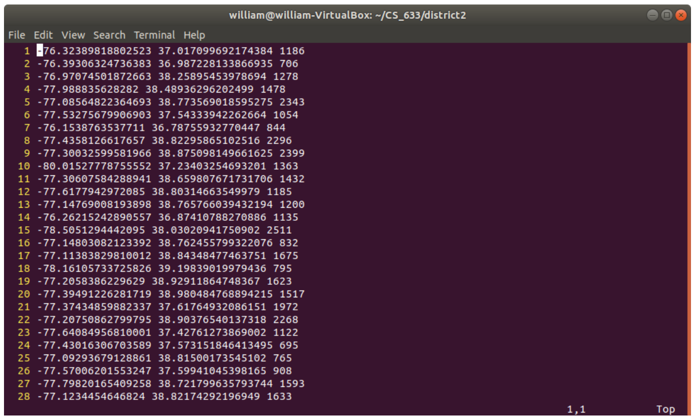


Figure 10: Calculated Centroids for each Census Block

Next, we use the "geospatial join" feature of QGIS to merge the population data from the downloaded CSV into the shapefile. We join on the `AFFGEOID` (American Fact Finder Geographic Identifier), which is present in both data sets. The population for each census block is save as the `POPCOUNT` attribute for each feature.

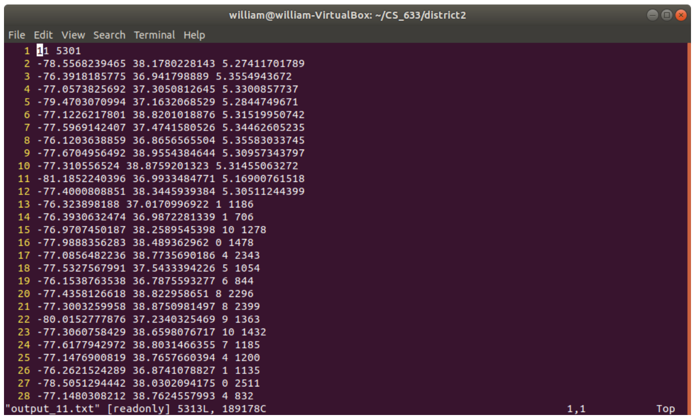
Lastly, we created a Python scrip named *combine_data.py* that reads the combined shapefile and writes a text file in the desired format. As expected, this script makes use of the PyShp library to read the desired attributes from the saved shapefile. The first few lines of this file are shown below.


william@william-VirtualBox: ~/CS_633/district2
File Edit View Search Terminal Help
1,1
Top

```
-76.32389818802523 37.017099692174384 1186
-76.39306324736383 36.987228133866935 706
-76.97074501872663 38.25895453978694 1278
-77.988835628282 38.48936296202499 1478
-77.08564822364693 38.773569018595275 2343
-77.53275679906903 37.54333942262664 1054
-76.1538763537711 36.78755932770447 844
-77.4358126617657 38.82295865102516 2296
-77.30032599581966 38.875098149661625 2399
-80.01527778755552 37.23403254693201 1363
-77.30607584288941 38.659807671731706 1432
-77.6177942972085 38.80314663549979 1185
-77.14769008193898 38.765766039432194 1200
-76.26215242890557 36.87410788270886 1135
-78.5051294442095 38.03020941750902 2511
-77.14803082123392 38.762455799322076 832
-77.11383829810012 38.84348477463751 1675
-78.16105733725826 39.19839019979436 795
-77.2058386229629 38.92911864748367 1623
-77.39491226281719 38.980484768894215 1517
-77.37434859882337 37.61764932086151 1972
-77.20750862799795 38.90376540137318 2268
-77.64084956810001 37.42761273869002 1122
-77.43016306703589 37.573151846413495 695
-77.09293679128861 38.81500173545102 765
-77.57006201553247 37.59941045398165 908
-77.79820165409258 38.721799635793744 1593
-77.1234454646824 38.82174292196949 1633
```

Figure 11: Census Block Centroids and Population Counts

Step 3. Run Application to Create Maps for Virginia Earlier, while describing the balanced centroidal power diagrams algorithm, we showed the sample output from the application. See 6 for details. At this step, we decide k, which represents the number of districts, and the application will assign each population point to one of these districts. Sample output from for $k = 11$ is shown below.


william@william-VirtualBox: ~/CS_633/district2
File Edit View Search Terminal Help
1,1
Top

```
11 5301
-78.5568239465 38.1780228143 5.27411701789
-76.3918185775 36.941798889 5.3554943672
-77.0573825692 37.3050812645 5.3300857737
-79.4703070994 37.1632068529 5.2844749671
-77.1226217801 38.8201018876 5.31519950742
-77.5969142407 37.4741580526 5.34462605235
-76.1203638859 36.8656565504 5.35583033745
-77.6704956492 38.9554384644 5.30957343797
-77.310556524 38.8759201323 5.31455063272
-81.1852240396 36.9933484771 5.16900761518
-77.4000808851 38.3445939384 5.30511244399
-76.323898188 37.0170996922 1 1186
-76.3930632474 36.9872281339 1 706
-76.9707450187 38.2589545398 10 1278
-77.9888356283 38.489362962 0 1478
-77.0856482236 38.7735690186 4 2343
-77.5327567991 37.5433394226 5 1054
-76.1538763538 36.7875593277 6 844
-77.4358126618 38.822958651 8 2296
-77.3003259958 38.8750981497 8 2399
-80.0152777876 37.2340325469 9 1363
-77.3060758429 38.6598076717 10 1432
-77.6177942972 38.8031466355 7 1185
-77.1476900819 38.7657660394 4 1200
-76.2621524289 36.8741078827 1 1135
-78.5051294442 38.0302094175 0 2511
-77.1480308212 38.7624557993 4 832
"output_11.txt" [readonly] 5313L, 189178C
```

Figure 12: Sample Output from Balanced Centroidal Power Diagram Algorithm

The last part of this step is to use Python to format the algorithm output correctly for gnuplot. There are two steps that need to be completed to show the map properly:

1. The *Voronoi_boundaries.py* script generates the correct edges for the resulting diagram and associates each generated district and all population points within that district to a particular color.

2. The *plotGNUPlot.py* script will generate the gnuplot output and uses an additional boundary file to clip the Voronoi cells to match the geography of the state.

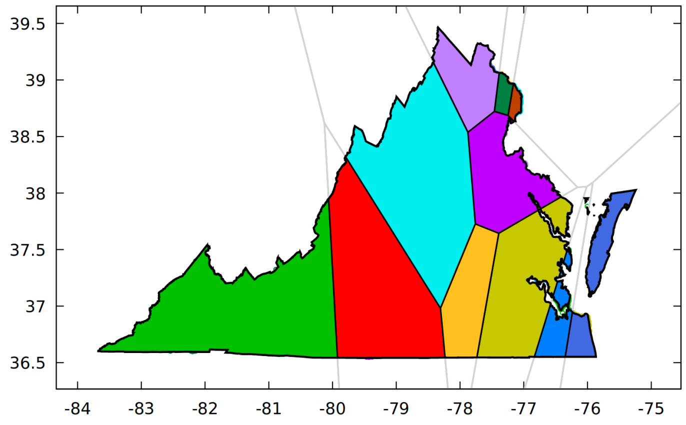


Figure 13: Balanced Centroidal Power Diagram Map for 11 Congressional Districts, using GNUPlot

Step 4. Use Map Output to Create Standard Shapefiles This step is a necessary conversion procedure to generate a shapefile that matches the generated map from the previous step. There are several steps involved in this step, descibed below:

1. Extract the raw Voronoi cells from the output file generated in the previous step.
2. Create and run a Python script that generates a shapefile from the extracted data. In my code, this is the $CreateShapefile.py$ script. This script will create a shapefile with the correct regions.
3. Open the generated shapefile with our Voronoi cells, as well as a shapefile with the Virginia state outline in QGIS. The image below shows what this looks like.

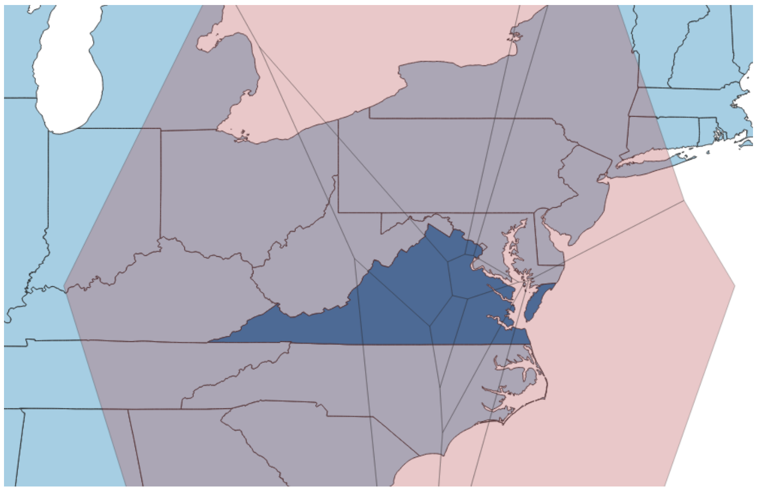


Figure 14: Voronoi Cell Shapefile Overlaid on US State map

4. Use the QGIS functions to intersect these layers and generate a new layer. Remove all extraneous features, give each district a proper name, apply a reasonable style, and save the resulting shapefile. The result is below.

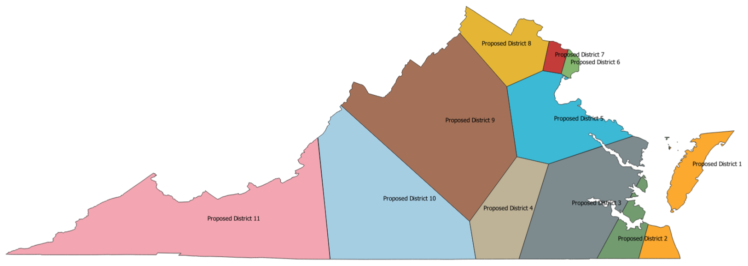
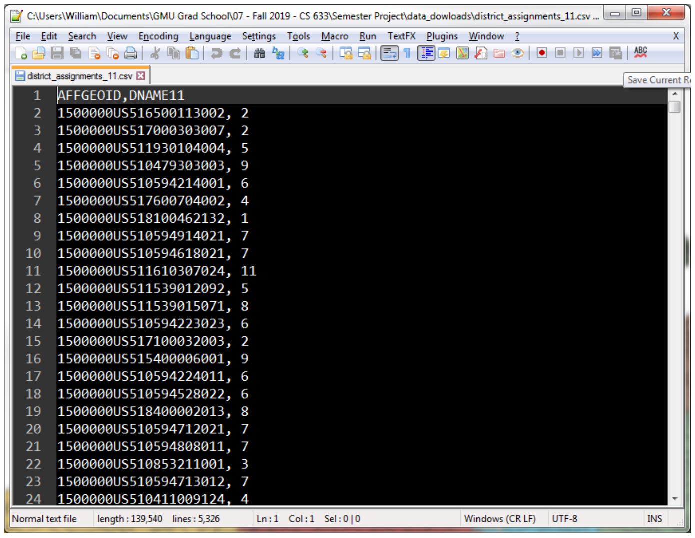


Figure 15: Shapefile Representing Balanced Centroidal Power Diagram Output

Step 5. Generate District Maps Without Splitting Census Blocks The enhancement that we implemented on top of this output is to create a map that is slightly modified, but respects census block boundaries. The main work for this step is to create a Python script that reads all of the census block and compares them to the generated Voronoi cells for our existing map. Then, we assign each census block to the district that it has the most area overlap with. One interesting optimization is that we can easily compute a bounding box around each census block and Voronoi cell so that we can save quite a bit of processing time by simply not comparing regions that do not overlap. This functionality is implemented in *EvaluateCensusBlocks.py*.

Note that the the output of this script is a CSV file that contains the mapping of `AFFGEOID` identifiers for each census block, along with the assigned district number. Then, we can perform a geospatial join to integrate this data into our census block shapefile. The CSV file data is shown below.


C:\Users\William\Documents\GMU Grad School\07 - Fall 2019 - CS 633\Semester Project\data_dowloads\district_assignments_11.csv ...
File Edit Search View Encoding Language Settings Tools Macro Run TextFX Plugins Window ?
district_assignments_11.csv
Save Current R

```
AFFGEOID,DNAME11
1500000US516500113002, 2
1500000US517000303007, 2
1500000US511930104004, 5
1500000US510479303003, 9
1500000US510594214001, 6
1500000US517600704002, 4
1500000US518100462132, 1
1500000US510594914021, 7
1500000US510594618021, 7
1500000US511610307024, 11
1500000US511539012092, 5
1500000US511539015071, 8
1500000US510594223023, 6
1500000US517100032003, 2
1500000US515400006001, 9
1500000US510594224011, 6
1500000US510594528022, 6
1500000US518400002013, 8
1500000US510594712021, 7
1500000US510594808011, 7
1500000US510853211001, 3
1500000US510594713012, 7
1500000US510411009124, 4
```

Normal text file   length : 139,540   lines : 5,326   Ln : 1   Col : 1   Sel : 0 | 0   Windows (CR LF)   UTF-8   INS

Figure 16: Python Script Output, with Census Block Assignments

The last step for creating the final map is to use the QGIS "Dissolve" function to create a single region from many individual regions. In this case, all features that have the same district mapping can be dissovled into a single feature. This generated layer can then be exported as a new shapefile, as shown below.

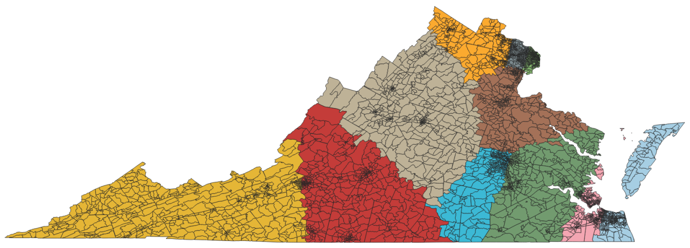


Figure 17: Census Block Assignments, before “Dissolve” Operation

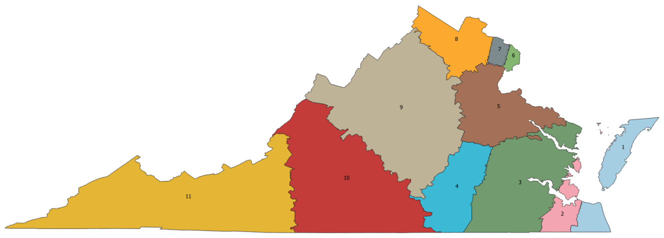


Figure 18: Final Map that Respects Census Block Boundaries, with $k = 11$

2.3 Metrics

Now that we have created maps, we also want to generate metrics that will let us quantitatively compare maps. In general, the opposite of a gerrymandered map is a compact map, so these metrics will focus on determining how compact districts produced by a given method are.

Metric 1. Polsby-Popper Score The Polsby-Popper score for a district measure the ratio of area covered by a district, compared to the area of a circle with the same perimeter as the district[16]. Obviously, for a circle, this ratio would be 1, but the score will become closer to 0 as the shape become more non-convex and disfigured. The formula is:

$$PP = 4\pi \times \frac{A_D}{P_D^2}$$

An example showing what the Polsby-Popper score represents is below:

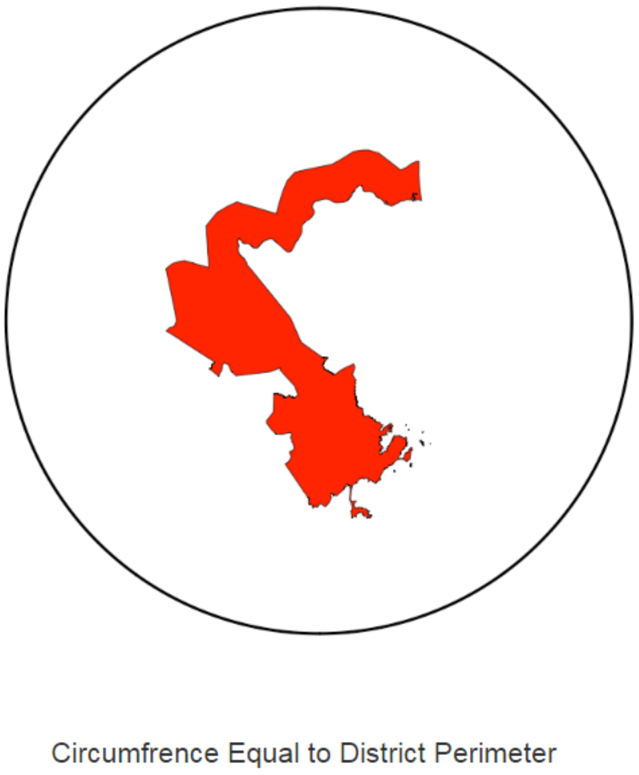


Figure 19: Polsby-Popper Score Example

For our project, this value was computed by the set of Python scripts ending with *Polsby_Popper.py*. As expected, we utilize the Shapely library to compute the district area and perimeter values.

Metric 2. Convex Hull Ratio Score The next metric that we compute is the ratio between the area of the shape and the area of it's convex hull [16]. One advantage of this computation is that it is not dependent on the perimeter of the shape, which may be large due to natural geography (rivers, shorelines, mountain ranges, . . .) that are largely beyond our control. The formula is:

$$CH = \frac{A_D}{A_{MCP}}$$

And an example, with the original Gerry-mander shape is:

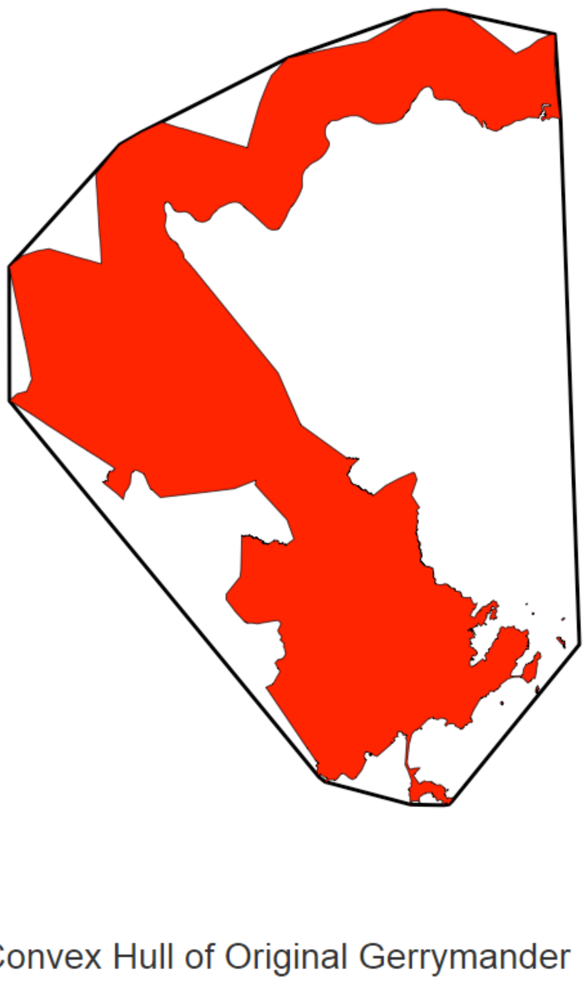


Figure 20: Convex Hull Ratio Score Example

In our code, these scores were computed by the *CH.py* scripts, and again we use Shapely to find the district areas and the convex hull.

Metric 3. Average Distance to Centroid, based on Population For this metric, we compute the average distance between a resident and the center of their district. This is a useful metric because it is an informal, but relatable estimate of how compact the district is. The idea for this computation comes from Brian Olson, who used it to justify the compactness of many of the maps that he created. Note that this metric is influenced by the population distribution, so it matters where residents live within a district. The equation is:

$$Avg\ Distance = \frac{1}{|P|}\sum_{P} d(p,\ C(D))$$

$$where\ P\ is\ the\ set\ of\ population\ points, p \in P,$$

$$C(D)\ the\ center\ of\ district\ D,$$

$$and\ d(x,\ y)\ is\ the\ L_2\ distance\ metric$$

The values for this script were computed by the set of Python scripts named *Avg_Dist.py*.

Metric 4. Average Distance to Centroid, based on Area The last metric that we consider is the average distance to the centroid, based exclusively on the geometry of the district shapes. To compute this we choose random sample points from the the districts and find the distance to the centroid. An exact geometric solution is possible, but the math is somewhat cumbersome, and this method gives results that are good enough.

Note that our implementation uses the center of census blocks as the input points, and then weights each distance based on the area of the census block. Random sampling may be better, but there are typically several hundred census blocks per district, so the difference is negligible.

Note that these values are computed in parallel with the previous metric in the same set of scripts.

3 Results and Findings

In this section we will present our final maps and the results or our metric computations. In addition, we will go into some discussion about the results.

3.1 Final Map Comparison

The following six maps are presented below.

1. Current Virginia Congressional Districts
2. Our Proposed Virginia Congressional Districts
3. Current Virginia State Senate Districts
4. Our Proposed Virginia State Senate Districts
5. Current Virginia House of Delegates Districts
6. Our Proposed House of Delegates Districts

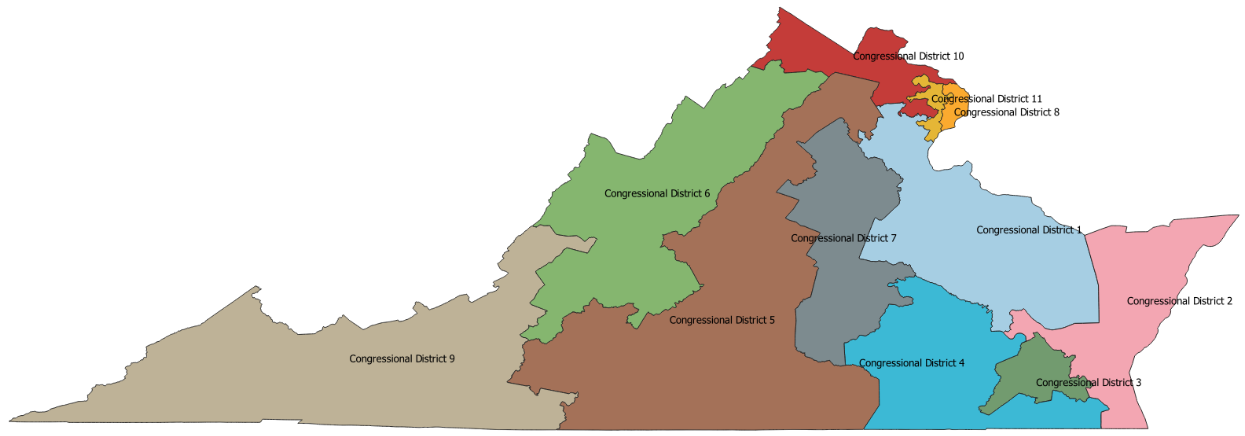


Figure 21: Current Virginia Congressional Districts

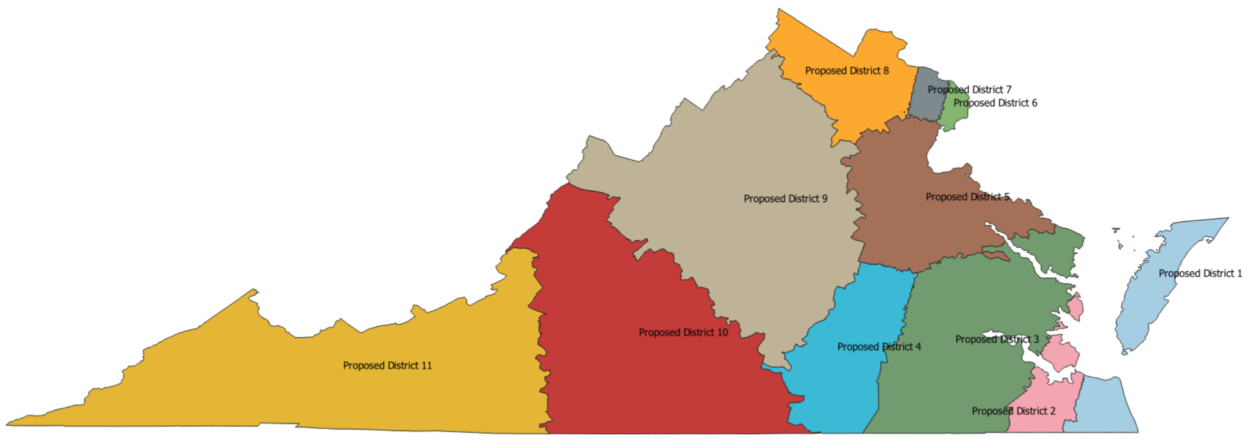


Figure 22: Our Proposed Virginia Congressional Districts

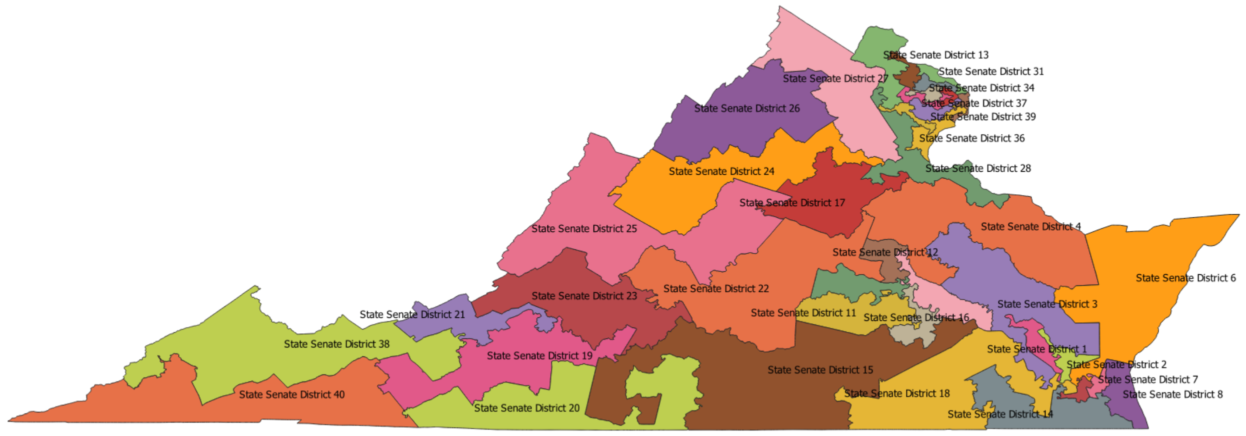


Figure 23: Current Virginia State Senate Districts

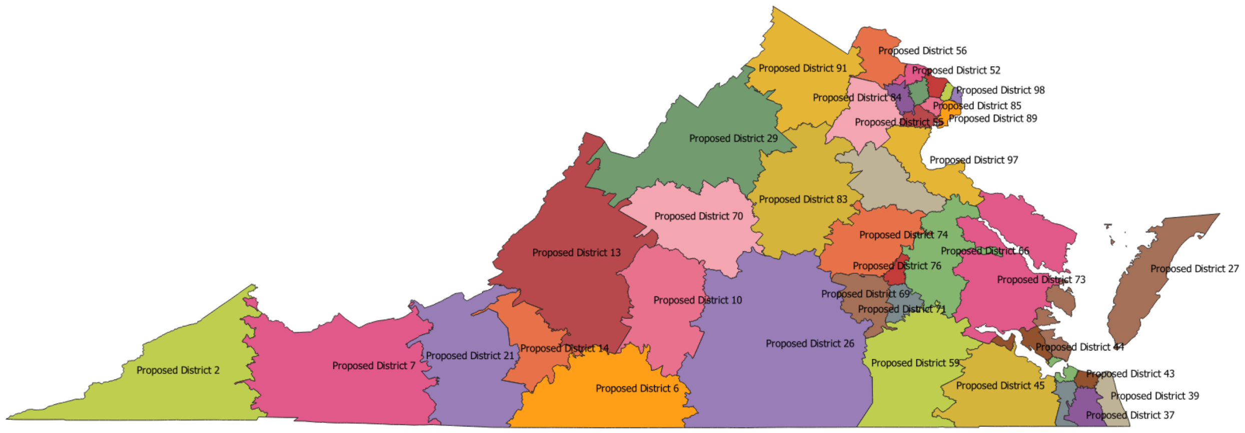


Figure 24: Our Proposed Virginia State Senate Districts

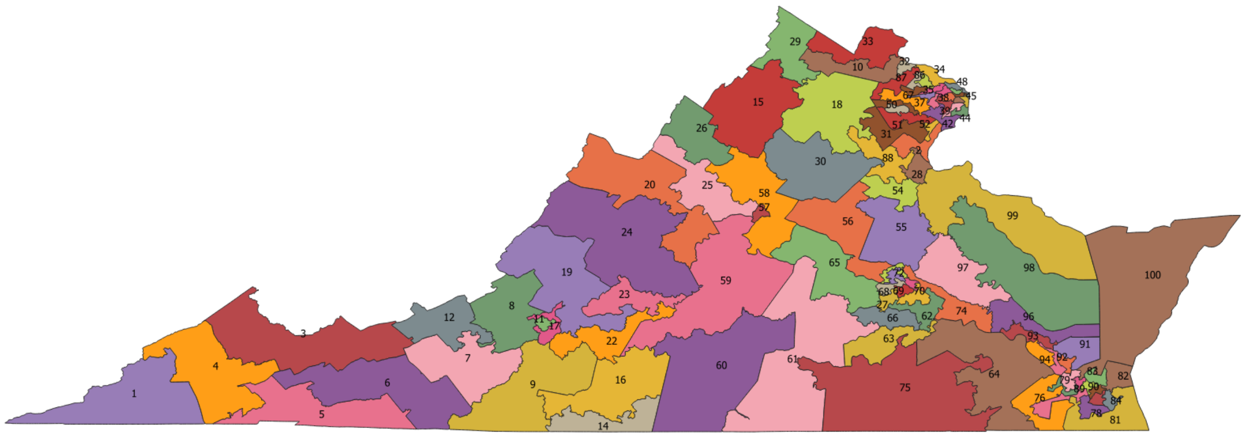


Figure 25: Current Virginia House of Delegates Districts

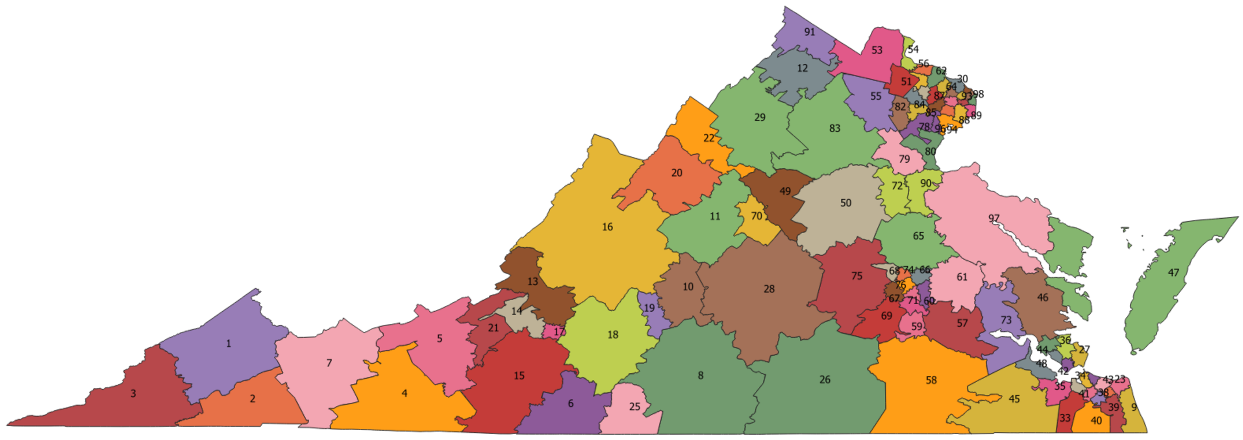
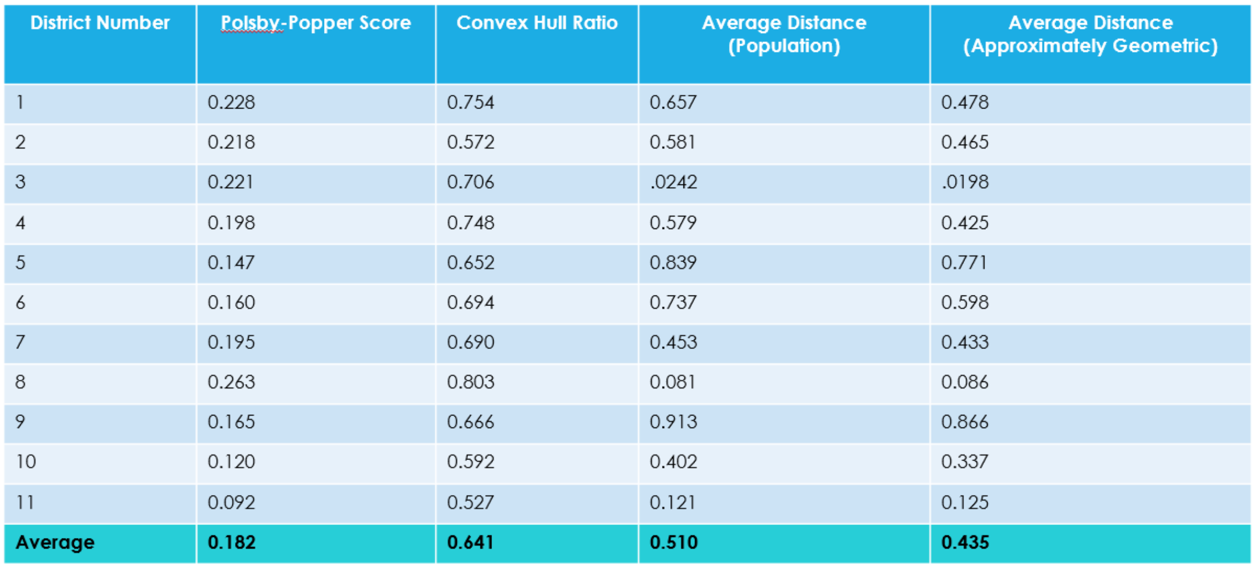


Figure 26: Our Proposed House of Delegates Districts

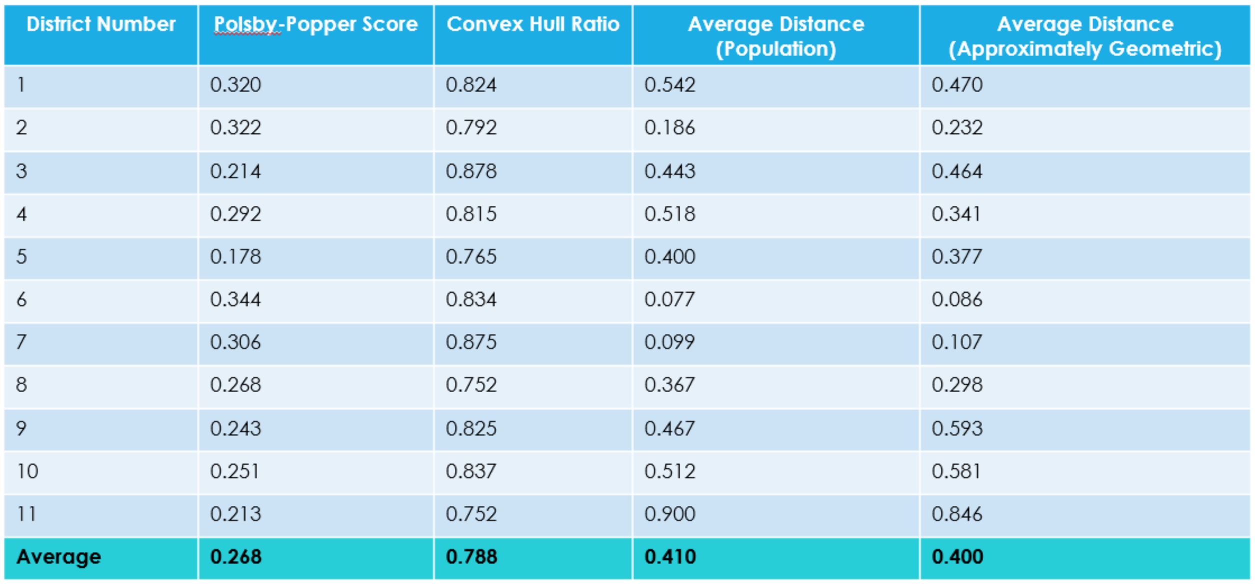
3.2 Metric Calculation Results

In this section, we show the results for our four metrics that we described above. The computed values are shown here for the current 11 Virginia Congressional Districts, as well as the 11 proposed districts. In addition, we compute an average for each of the metrics, that can be used as a marker for the overall performance of each method.



District Number	Polsby-Popper Score	Convex Hull Ratio	Average Distance (Population)	Average Distance (Approximately Geometric)
1	0.228	0.754	0.657	0.478
2	0.218	0.572	0.581	0.465
3	0.221	0.706	.0242	.0198
4	0.198	0.748	0.579	0.425
5	0.147	0.652	0.839	0.771
6	0.160	0.694	0.737	0.598
7	0.195	0.690	0.453	0.433
8	0.263	0.803	0.081	0.086
9	0.165	0.666	0.913	0.866
10	0.120	0.592	0.402	0.337
11	0.092	0.527	0.121	0.125
Average	**0.182**	**0.641**	**0.510**	**0.435**

Figure 27: Metrics for Current Congressional Districts



District Number	Polsby-Popper Score	Convex Hull Ratio	Average Distance (Population)	Average Distance (Approximately Geometric)
1	0.320	0.824	0.542	0.470
2	0.322	0.792	0.186	0.232
3	0.214	0.878	0.443	0.464
4	0.292	0.815	0.518	0.341
5	0.178	0.765	0.400	0.377
6	0.344	0.834	0.077	0.086
7	0.306	0.875	0.099	0.107
8	0.268	0.752	0.367	0.298
9	0.243	0.825	0.467	0.593
10	0.251	0.837	0.512	0.581
11	0.213	0.752	0.900	0.846
Average	**0.268**	**0.788**	**0.410**	**0.400**

Figure 28: Metrics for Proposed Congressional Districts

As the figures show, the overall compactness is better with the new method. For both Polsby-Popper and Convex Hull, the average ratio using our method is higher, which indicates more compact districts. On the other hand, the average distance to center metrics are lower for our maps, which is again better because it means that we are dividing up the population more efficiently.

Note that these metrics were presented in class as well, but they contained an error in the way that we were computing the average distances. This has been corrected, leading to the results above.

3.3 Challenges and Lessons Learned

As we mentioned in the previous section, one of the largest hurdles to overcome for this project was simply becoming familiar with how to work with geospatial information. There is a learning curve for understanding the shapefile data format, projections, new API's (Shapely and PyShp), and new UI functionality in QGIS. Researching information and finding data sources took more time than expected, with the retrieval of population data from the American Fact Finder website being particularly confusing.

Another lesson learned was how important it is to keep data organized and adhere to particular conventions when generating and analyzing data. For example, organizing shapefiles, naming feature attributes consistently, and generating reusable scripts is more difficult than I anticipated. In order to contunue this effort, I would need to reorganize my data files and document my processing pipeline.

4 Conclusion and Future Work

By generating new maps and computing metrics, this project demonstrated that we can indeed enact change that improves the gerrymandering situation we face. It is important to continue to make positive progress in this area so that states have real options available to them. In addition, continuing to raise public awareness is critical. In a few short months the 2020 Census will be taking place, with nationwide redistricting efforts occuring afterwards. A unified approach that combines the best features of structural reform for independent, bipartisan redistricting commissions and algorithmic techni ques must be pursued to fight the issues we face.

However, it is clear that there is plenty of work remaining to do that would further improve these results. A few examples are:

- I demonstrated that it is possible to use the Voronoi cells from the balanced centroidal power diagram to generate a feasible map that does not break apart census blocks. However, there is no guarantee with this method that the equal population distribution across districts is maintained. More work would have to be done to determine the best way to iterate on the Voronoi output.
- Include more metrics and compute for additional maps, so that we have more data points to work with. In addition to compactness, we can also use historical data to find how competitive our proposed districts would be, for example. Ideally, we want a set of maps that are *representative* of the entire population.

- Combine this method with other methods, such as the iterative techniques used by Brian Olson. Compare the maps generated by both methods and try to find a technique that incorporates the best features from both.
- Incorporate additional data, such as how important certain census boundaries are. For example, some boundaries coincide with county boundaries and might be better candidates to split districts on. In addition, we should prefer to utilize the census boundaries that are straight. Another example is that we should should prefer to use boundaries for larger roads and rivers that represent real division, rather than smaller streams and roads.
- We have shown that all of the individual steps in this process are efficient to be able to be performed at a state level in a few seconds. However, we need to find a way to combine steps, so that everything can be done programmatically, rather than relying on QGIS for key steps. Maintaining architectural flexibility during this process would be very important.

Lastly, this project was an excellent example of how concepts in Computational Geometry are applicable for real world problems. The operations needed to complete an undertaking such as this are driven by basic geometric algorithms that must scale to examples with thousands or millions of points that we encounter when working with state and national level data.

References

[1] Brian Olson. Compactness measures and variations. [Online; accessed 11-30-2019]. 1.3

[2] Brian Olson. Engineering elections without bias, by brian olson at tedxcambridge. [Online; accessed 10-10-2019]. 1.2

[3] Brian Olson. Impartial automatic redistricting, by brian olson. [Online; accessed 10-12-2019]. 1.2

[4] Vincent Cohen-Addad, Philip N. Klein, and Neal E. Young. Balanced power diagrams for redistricting, 2017. 1.3, 2.1

[5] Matt Morris. Virginia congressional redistricting: A better method. [Online; accessed 12-15-2019]. 1.2

[6] One Virginia 2021. Onevirginia2021: Virginians for fair districts. [Online; accessed 12-01-2019]. 1.2

[7] United States Census Bureau. 2016 tiger state legislative districts, upper chamber. [Online; accessed 10-22-2019]. 2.2

[8] United States Census Bureau. 2019 tiger block group files. [Online; accessed 10-22-2019]. 2.2

[9] United States Census Bureau. American fact finder. [Online; accessed 10-22-2019]. 2.2

[10] United States Census Bureau. Guidance for geography users. [Online; accessed 10-10-2019]. 2.2

[11] United States Census Bureau. Tiger/line shapefile, 2018, nation, u.s., 116th congressional district national. [Online; accessed 10-22-2019]. 2.2

[12] Virginia Department of Transportation. Dataset: Virginia house of delegates. [Online; accessed 10-22-2019]. 2.2

[13] Wikipedia contributors. Gerrymandering. [Online; accessed 10-03-2019]. 1.1, 1.1, 1.2

[14] Wikipedia contributors. Redmap. [Online; accessed 12-15-2019]. 1.2

[15] Wikipedia contributors. Transshipment problem. [Online; accessed 11-30-2019]. 2.1

[16] Zachary Fisher. Measuring compactness. [Online; accessed 11-30-2019]. 1.3, 2.3, 2.3