CS123 Final Project

Shelby Mahaffie, Casper Neo, and Sherry Shenker

June 6, 2016

Git Repository: https://github.com/smahaffie/123project

The Question

In the article *What is Normal America?*, Jed Kolko of *FiveThirtyEight* points out that the average American town is no longer a small town of white people, as many may think, by finding the country's most average towns based on demographics and size. Inspired by this article, we set out to find nation-wide demographic trends that extended beyond racial breakdowns. We set out to answer questions such as:

- What are the most and least averages places in the US, and how do these results change based on the specific variables that we consider?
- Where are the largest homogeneous areas in the US?
- Where are places most similar and dissimilar from their neighbors, and how can we visualize these dynamics?

This task led us to a big data challenge involved data from the 2000 U.S. Census.

The Data

Our project uses data from the 2000 US Census. The complete data set is publicly available and is 200 GB large. The data is divided into many files for every state, and within every state, a single row corresponds to data on a specific geographic entity within the state. For example, the first 100 rows may contain data at the county-level, and the next 400 will contain data at the city-level.

In order to work with this data, we use the geographic header files in order to determine which row indexes within each summary file for every state contain the geographic entities that are relevant. Then, we filter the data using MapReduce and proceed to complete multiple algorithms with the data. The size of the data set for all places in the U.S. was approximately 9 GB and for all census tracts in the U.S. was approximately 30 GB.

Algorithms

To find the most and least representative places in the country on n given variables:

1. We found the national mean and standard deviation of the distribution of each variable (racial distribution, median income, education levels, language spoken, percent foreign and native).
2. For each place, we generated a vector with n entries where each entry was the normalized value of one of the variables.
3. We plotted the places with the largest and smallest deviations from the national mean, with different combinations of variables. The distance from the mean is calculated using Euclidean distance.

To find the largest homogeneous area:

1. Generate all possible pairs of places in the United States using a MapReduce algorithm.
2. Define which pairs of places are neighbors. We used the Haversine formula to calculate the distance between a pair of points, places are considered neighbors if they're within a specified distance of each other.
3. Use Dijkstra's algorithm to grow an area of homogeneity from every place. The cost of an edge between a seen node and new node is the new node's attribute distance from the centre. We stop adding nodes to the graph if the shortest path is greater than a specified epsilon.
4. Take the longitude and latitudes of nodes in every homogeneous area as points we use the set of points to define a convex hull; the area of said hull is the homogeneous area.
5. Then we sort the areas by size using the standard top-k algorithms we studied at the start of the quarter.

Below is an illustration of the result of Dijkstra's algorithm with Silverton Alabama as the center.

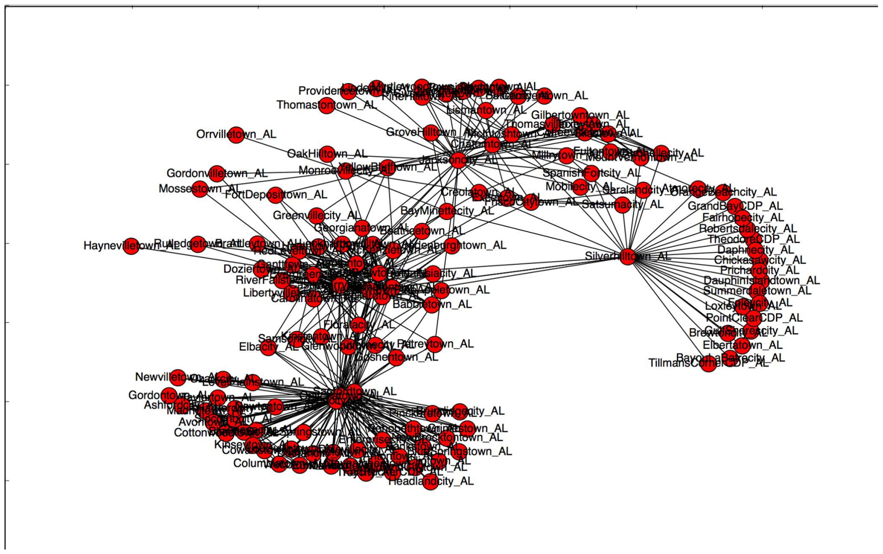


Figure 1: Dijkstra's algorithm

The spring simulation:

1. Define which pairs of places are neighbors (using the same definition as before).
2. Project the points from longitude and latitude to x, y with a Mercator projection.
3. Interpret the edge between neighbors as a spring. The resting length of the spring, r_{ij}, is a multiple of the original (projected geographic) distance g_{ij}, where the multiple is the attribute distance d_{ij} normalized by the average attribute distance between neighbors $\overline{d}$. This is so in the initial state, similar places are attracting while different places are repelling.

$$r_{ij} = g_{ij} \frac{d_{ij}}{\overline{d}}$$

4. Iterate a physics simulation until the points don't move around so much. Given information of a given place's position, momentum, neighbors, positions of neighbors, and resting spring length for each neighbor we can identify the "force" on a place and adjust its velocity and move it accordingly. We

also damp the the magnitude of velocity at each step with a damping force analogous to friction (It's mathematically different as friction linearly damps velocity and we use something geometric), this is so the simulation does not oscillate forever and eventually stops.

5. Let x_p, y_p and x'_p, y'_p be the x, y coordinates of p in the present and subsequent "frame" respectively. Let $N(p)$ denote the set of neighbors of p. Let $V_x(p)$ be the x velocity of p. Let our damping force coefficient be $0 < k < 1$. For simplicity we let all masses and the time increment be unit 1 so numerically momentum equals velocity equals change in position. Then our simulation follows the following rules:

$$\Delta V_x(p) = \sum_{n \in N(p)} \left(r_{pn} - \sqrt{(x_p - x_n)^2 + (y_p - y_n)^2} \right) \frac{(x_p - x_n)}{\sqrt{(x_p - x_n)^2 + (y_p - y_n)^2}}$$

$$\Delta V_y(p) = \sum_{n \in N(p)} \left(r_{pn} - \sqrt{(x_p - x_n)^2 + (y_p - y_n)^2} \right) \frac{(y_p - y_n)}{\sqrt{(x_p - x_n)^2 + (y_p - y_n)^2}}$$

$$V'_x(p) = k(V_x(p) + \Delta V_x(p)) \qquad V'_y(p) = k(V_y(p) + \Delta V_y(p))$$

$$x'_p = x_p + V'_x(p) \qquad y'_p = y_p + V'_y(p)$$

We iterate until

$$\max_p \sqrt{V_x^2(p) + V_y^2(p)} < \varepsilon$$

6. The resulting visualization shows the United States shifting as neighbors that are similar collapse upon themselves and neighbors that are different push away from each other.

Using MPI for springs:

1. Node 0 broadcasts the dictionaries containing the neighbors of places and the resting lengths of springs between places.

2. At each frame in the simulation Node 0 broadcasts the dictionary containing positions and momentums of places and then scatters keys to the dictionary splitting up the work

3. Each node calculates the next position and momentum of their respective places and keeps track of the maximum velocity of its points The new positions are then gathered and turned into a dictionary at node 0. The maximum velocities are also gathered and node 0 finds the maximum velocity of every node and compares that to the stopping condition (epsilon).

4. At each step Node 0 also saves the "Frame" which will later be graphed for our visualization.

Big Data Techniques

MapReduce:

- We use a 4 step MRjob to convert our data from one concatenated text file containing all the relevant census data extracted from an AWS volume into a series of relevant results

- We read in the raw census data in the very first mapper step. In order to filter the data based on relevant geographic locations, we add an extra file argument that contains a json file with the information (lists of row indexes) which we use to filter the data set.

- As an intermediate step, we convert the extracted data into a dictionary of vectors for easier use.

- We use longitude and latitude data in order to build a dictionary of every place's neighbors.

- We use the data for the entire US to calculate the most average and most unique places in the US based on specified variables, passing in the extra longitude and latitude data as an extra file into our MapReduce jobs.
- We use the data on neighbors in order to find the largest homogeneous areas in the US.

Amazon Web Services and EMR

- As we wrote our algorithms, we tested them on only the cities in Alabama. However, in order to make our code run on the larger data sets (all of the census designated places (CDPs) in the entire country, and, especially, all of the census tracts in the entire country) in a reasonable amount of time, we needed to use EMR clusters on AWS.
- In order to store our data, we created our own EBS Volume which we mounted to an ec2 instance every time we used it. We downloaded our git repositories onto our ec2 instances in order to be able to run our code on the files in the volume.

MPI

- We used bash script to install MPI4PY, copy relevant files, and relevant packages on each node, add IPs to hosts and cross ssh between them.
- MapReduce was unsuited to our process because we had an uncertain number of iterations and therefore cannot be summarized in MR-step. In addition we were saving dictionaries at each iteration. Between the uncertain number of steps and updating of global variables at each step, we felt like the flexibility of MPI allowed us to better implement our program.

Projected Run Times

Process	Time (10 instances)	Time (single computer)
Generate vectors of "places"	30 min.	300 min.
Generate vectors of census tracts	50 min.	500 min.
Generate all possible pairs of "places"	30 min.	300 min.
Generate all possible pairs of census tracts	100 min	1000 min.
1 Spring Simulation iteration	20 sec.	200 sec.

Note: We first completed all of our data algorithms using the data filtered for all "Census-designated Places" in the U.S. since this led to the most intuitive results. We then generated the vectors and standard deviations for all of the data at the census tract level, and were prepared to calculate the remainder of our results at a census tract level as well over the weekend. However, due to the limitations of AWS, we did not complete this final component of the project, though, as we discussed, the code would run in exactly the same way since the formatting in consistent throughout the data, and the results would be less intuitive since "most/least average census tract" is clearly not as interpretable as "most/least average city."

Challenges

- Interpreting census data: The documentation for census data was massive, but highly unhelpful. It took us a significant amount of time to understand how the data was structured and how we would be able to work with it in the most efficient way. Moreover, once we understand our data, we had to think about how to manage it in the most efficient way. As we discovered more bits of information that we needed to extract (e.g. latitude and longitude), we had to develop methods of extracting and storing these bits of information that would be compatible with all components of our program (e.g.

MapReduce, MPI, plotting, etc.). We made heavy use of json files, learned how to pass extra files into MapReduce, and tried to optimize our code to have to generate as few intermediate data sets as possible.

- Making code efficient: When we first wrote all of our code, we wrote many separate MapReduce jobs that ran on their own. However, we realized that this was highly inefficient as it required initializing new clusters every time. Thus, we had to restructure our code as a multi-step job and organize the inputs and outputs in such a way that the jobs could run one after another. However, we then ran into a problem of running multi-step jobs with EMR on AWS, which we spent a large amount of time trying to debug. Ultimately, we were able to run the multi-step job locally but had to run the jobs separately when using the entire data set in EMR.
- Installation of packages and setting up our machines. We spent a lot of time and effort installing the packages for MPI and EMR. The former required bash scripting as it would be impractical to install software on several compuers and cross ssh into each other to just set up MPI.For the latter as some of our code required special packages we had to learn how to tweak the EMR bootstrapping procedures.
- Understanding the nuances of AWS: We made many initial mistakes when it came to working with AWS. We discovered that editing data from within a bucket is not a good idea. We experienced many unexplained ”Broken pipe” errors, and we had to understand the interaction between ec-2, s3, and the volumes in order to run our code in the correct way. We ultimately decided that the best way to work with our data was to create a single bucket, where we kept all of our manipulated data. Every time we started an instance, we installed git on the instance, pulled our code, and ran it in this way. This worked successfully, with the only downside being that the volume could only be attached to one instance at a time, so we could not work simultaneously without having the same instance on multiple computers.

Hypotheses

For the most representative places in the country on all variables, we expected to find that we had loosely recreated the results of the 538 paper, with the most representative places being in the New England area.

We expected to find that the least representative places would be tiny towns that had only a few residents and so could be almost entirely homogenous, perhaps everyone in the town being of the same racial or ethnic minority or no one in the town having graduated from high school.

Our hypothesis concerning the largest homogenous areas in the country was that we would find them in sparsely-populated regions in the midwest, and that we would find none on the coasts where very different towns can be tightly packed together. Similarly, we expected to find with our spring simulation that the towns on the coasts in more diverse areas would be pushed away from each other while the center of the country, where we expected everything to be more homogenous, would collapse in on itself to some extent.

Results

Most and Least Representative Cities in the United States:

We started by finding the places that were most and least representative on all variables, shown in Table 1. However, despite normalizing all of the data, we still found that for the least representative places, population was biasing our results because of the extreme right skew of the distribution of population sizes of the cities. Four of the top 5 least representative places we found were the largest 4 cities in the country: New York, Los Angeles, Chicago, and Houston.

Figure 2: Most Unique and Most Average Places in the US

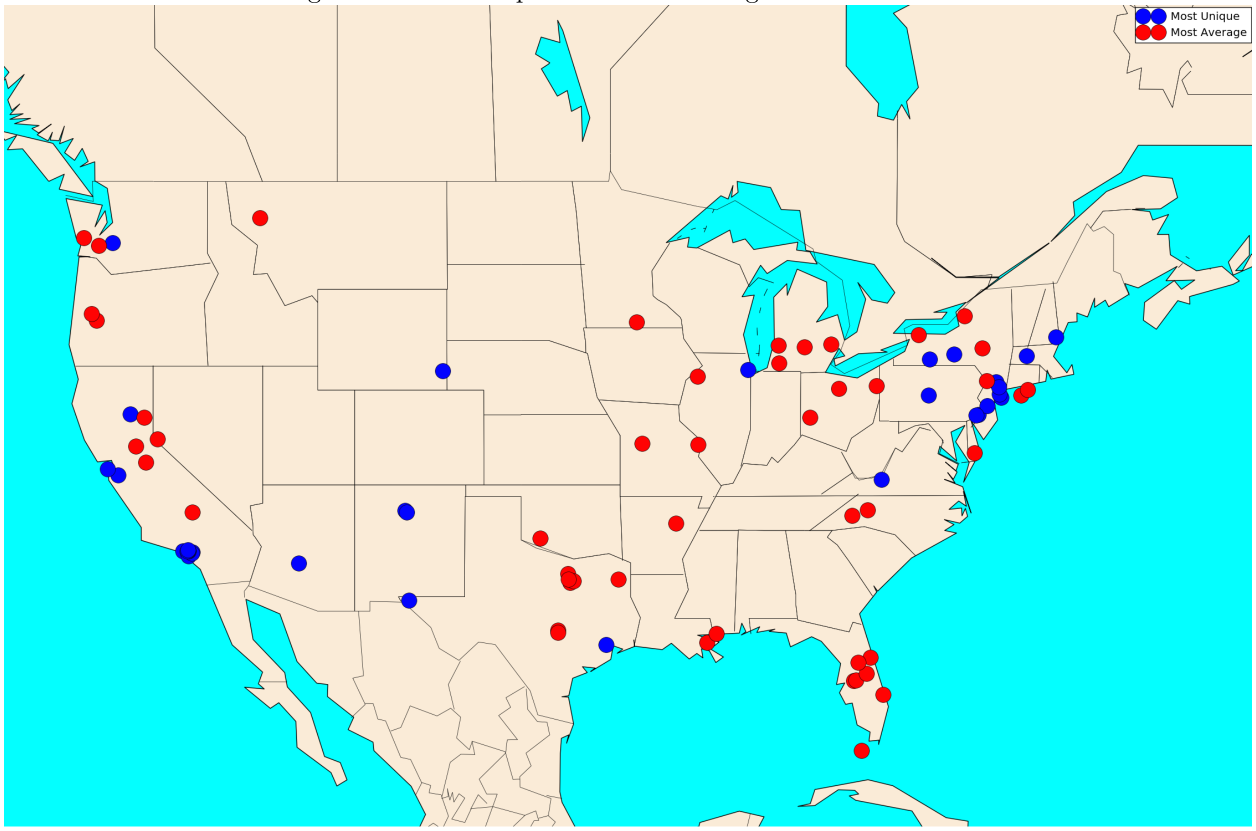


Table 1: Most and Least Representative Places in the US [All variables]

Most Representative		Least Representative	
Watertown	NY	New York	NY
White City	FL	Los Angeles	CA
Lake Davis	CA	Chicago	IL
Kerr	MT	Kaumakani	HI
PortJervis	NY	Houston	TX
OreCity	TX	Ewa	HI
Lime	AK	Whitmore	HI
South Daytona	FL	Waipahu	HI
Poydras	LA	Elbe	WA
Edgemont Park	MI	Monterey Park	CA
Pearsonville	CA	Alfred	NY
CottageGrove	OR	Walnut	CA
Moriches	NY	Puhi	HI
Combee Settlement	FL	Pahoa	HI
Chehalis	WA	Haliimaile	HI
Auburndale	FL	Hanamaulu	HI
Zeeland	MI	Cerritos	CA
Decatur	MI	Rowland Heights	CA
Rhome	TX	Belden	CA
Shingle Springs	CA	Rosemead	CA

To address this bias, we ran our MapReduce job again on all variables except for population.

Figure 3: Most Unique and Most Average Places in the US[All Variables Except Population]

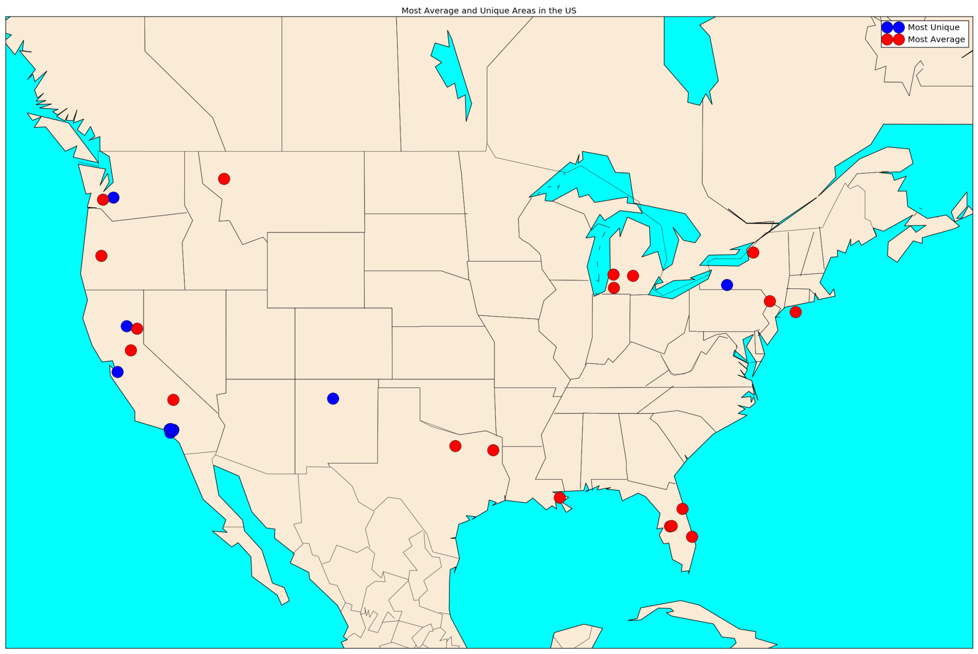


Table 2: Most and Least Representative Places in the US [All Variables Except Population]

Most Representative		Least Representative	
Watertown	NY	Kaumakani	HI
White City	FL	Ewa	HI
Lake Davis	CA	Whitmore	HI
Kerr	MT	Waipahu	HI
Ore City	TX	Elbe	WA
Port Jervis	NY	Monterey Park	CA
Lime	AK	Alfred	NY
South Daytona	FL	Walnut	CA
Poydras	LA	Puhi	HI
Edgemont Park	MI	Pahoa	HI
Pearsonville	CA	Haliimaile	HI
Cottage Grove	OR	Hanamaulu	HI
Moriches	NY	Cerritos	CA
Rhome	TX	Rowland Heights	CA
Decatur	MI	Belden	CA
Combee Settlement	FL	Rosemead	CA
Auburndale	FL	Los Alamos	NM
Chehalis	WA	Eleele	HI
Zeeland	MI	Milpitas	CA
Shingle Springs	CA	Village Park	HI

We also ran our algorithm considering only subsets of the variables to find the most and least Representative places considering only certain attributes of a place and to see what was driving our general results. We highlight here some of the more interesting results we found in this process:

First, we considered only the variables associated with race and the percentage of residents who identified themselves as Hispanic. Clearly, these variables were a major part of what was driving our results for most representative in the previous table. Nearly all of the least representative places were in regions of Hawaii where large portions of the population identify themselves as Pacific Islander or were Native American villages. The most representative places here is the closest we came to reproducing the results of the 538 article because the article was focused mostly on the racial composition of America.

Figure 4: Most Unique and Most Average Places in the US (Race and Percent Hispanic)

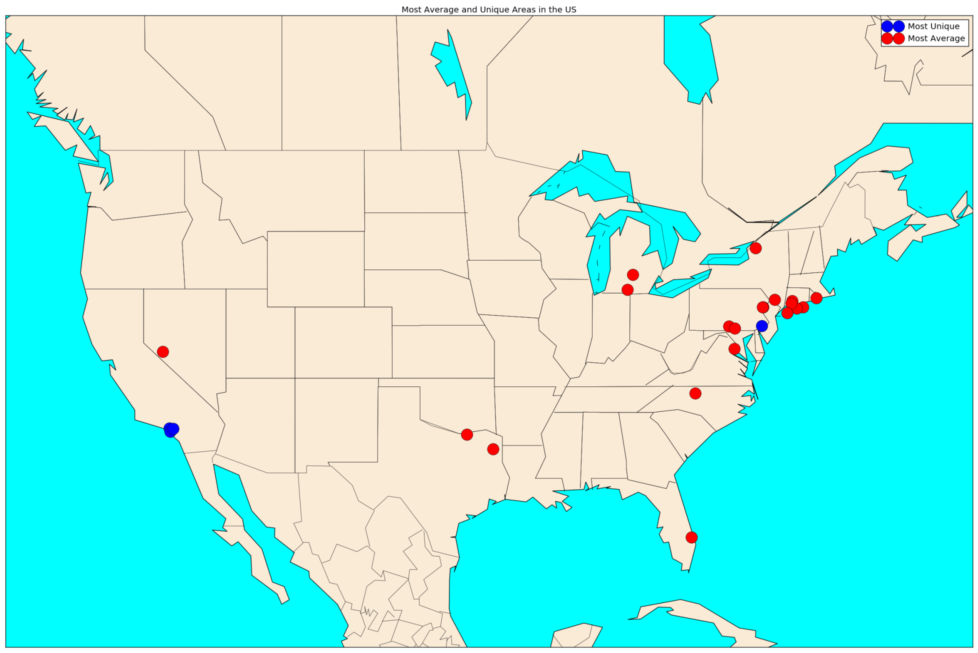


Table 3: Most and Least Representative Places in the US (Race and Percent Hispanic)

Most Representative		Least Representative	
OreCity	TX	Kumakani	HI
Ansonia	CT	Ewas	HI
Edgemont Park	MI	Whitmoree	HI
SagHarbor	NY	Waipahu	HI
Calverton	NY	Puhi	HI
Kernersville	NC	Eleele	HI
Philadelphia	NY	Monterey Park	CA
Goshen	NY	Hanamaulu	HI
Newport	RI	Aiea	HI
PortSt.Lucie	FL	LanaiCity	HI
Hawthorne	NV	VilagePark	HI
East Stroudsburgbor	PA	Cerritos	CA
Coldwater	MI	Keaau	HI
Knollwood	TX	Honolulu	HI
Chambersburg	PA	Walnut	CA
Stratford	CT	Waimalu	HI
Gettysburg	PA	Pepeekeo	HI
Stroudsburg	PA	Waipio	HI
Occoquan	VA	Millbourne	PA
Amityville	NY	Kahului	HI

Next, we considered only the variables associated with level of educational attainment. We unsurprisingly found that most of the least representative places on these variables were towns that were home to large colleges and universities.

Figure 5: Most Unique and Most Average Places in the US (by Education Level)

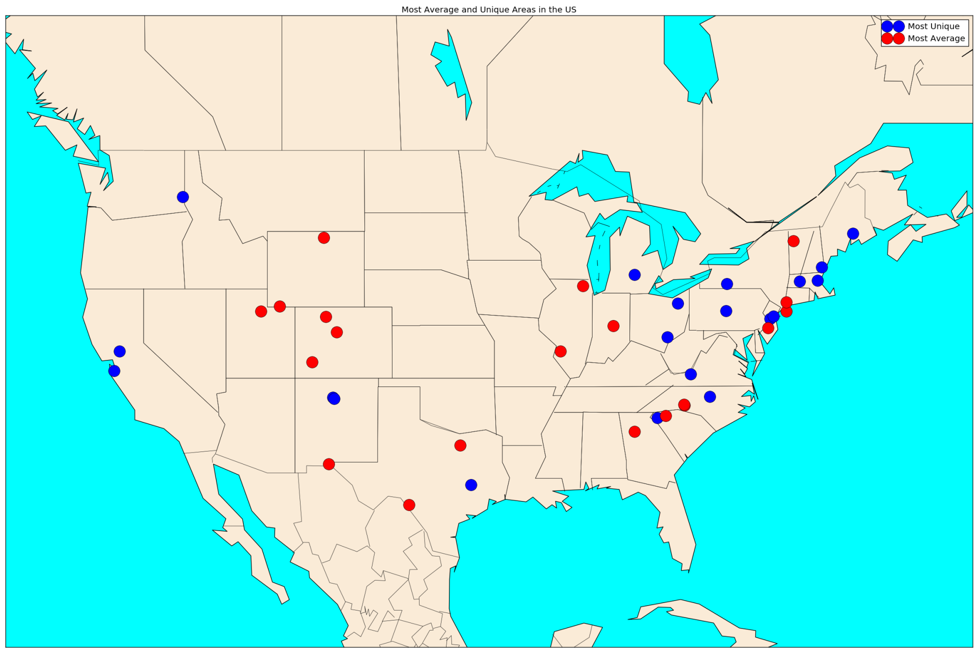


Table 4: Most and Least Representative Places in the US (Educational Attainment)

Most Representative		Least Representative	
Barrington Hills	IL	Alfred	NY
Vinings	GA	Los Alamos	NM
Lonetree	WY	Blacksburg	VA
Prado Verde	TX	White Rock	NM
Medford Lakesboroug	NJ	Amherst Center	MA
Plainview	NY	Durham	NH
Ridgefield	CT	Princeton North	NJ
North Snyderville	UT	State College Borough	PA
Steamboat Springs	CO	Pullman	WA
Laughlin AFB	TX	Clemson	SC
Jericho	VT	Athens	OH
Scott	IL	Sugar Bush Knollsvi	OH
Fishers	IN	Chapel Hill	NC
Big Horn	WY	College Station	TX
Dillon	CO	Davis	CA
Huntersville	NC	Highland Parkboroug	NJ
Cornelius	NC	East Lansing	MI
Flower Mound	TX	Palo Alto	CA
Five Forks	SC	Orono	ME
Telluride	CO	Lexington	MA

Finally, here are our results when considering only variables associated with income and employment status. Here, we found a mix of unusually wealthy suburbs and places that had unusually high employment in 2000.

Figure 6: Most Unique and Most Average Places in the US (by Income and Level of Employment)

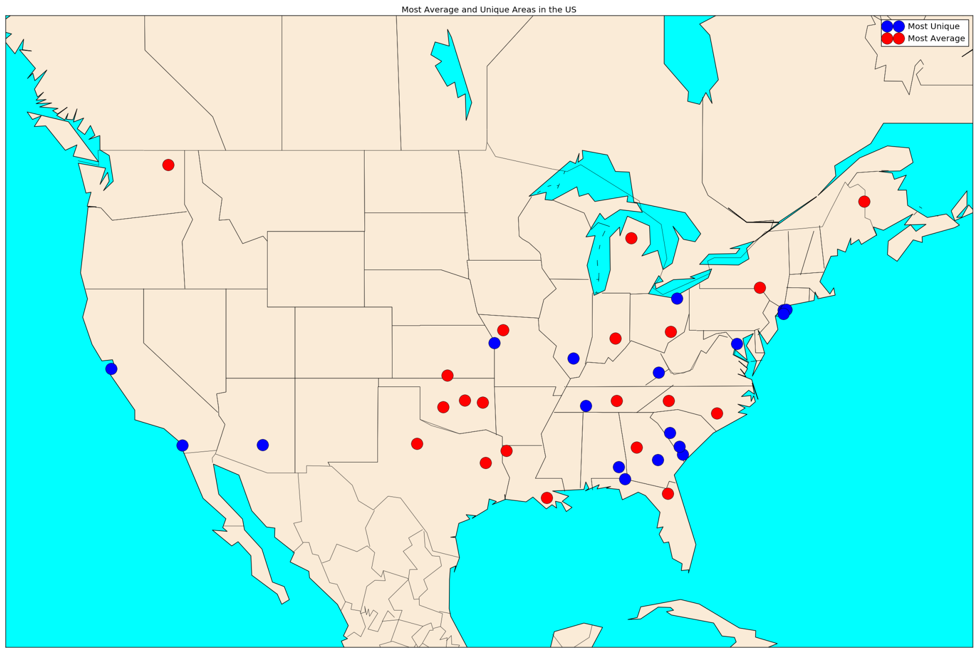


Table 5: Most and Least Representative Places in the US (Income and Employment Status)

Most Representative		Least Representative	
Grayling	MI	Malone	FL
Clifford	IN	Ina	IL
Deposit	NY	Alamo	GA
Inchelium	WA	Wheelwright	KY
MarsHill-Blaine	ME	Clio	AL
Thomaston	GA	Florence	AZ
Anthony	KS	Edgefield	SC
McConnelsville	OH	Ridgeland	SC
Marion	NC	SandsPoint	NY
Oktaha	OK	Brookville	NY
Roseboro	NC	Chevy Chase	MD
Binger	OK	Hunting Valley	OH
Aspermont	TX	Oyster Bay Cove	NY
Thibodaux	LA	Rancho Santa Fe	CA
Hamilton	MO	McCarthy	AK
Jacksonville	TX	Hillsborough	CA
Benton	LA	Fairfax	SC
Davenport	OK	Mission Hills	KS
Lawtey	FL	Woodsburgh	NY
McMinnville	TN	Clifton	TN

Homogeneous Areas:

Finally, below are the results for our calculation of the largest homogeneous areas in the US. As each homogeneous area is a set of places we use the surface area of the smallest convex hull covering those collection of points, and filter by the largest surface area.

Since use a shortest path algorithm where the costs between edges is the attribute distance from the centroid. Therefore a plausible social interpretation of a place in a homogeneous area of a center is that place is how far one can travel from the center while only seeing a certain amount of difference from home. We originally expected that most of the homogeneous areas would be in the Midwest and our results largely confirmed this result. We also saw homogeneous areas on the boarder of Texas and Mexico (homogeneous Hispanic population), Florida and Pennsylvania.

Figure 7: Homogenous Areas

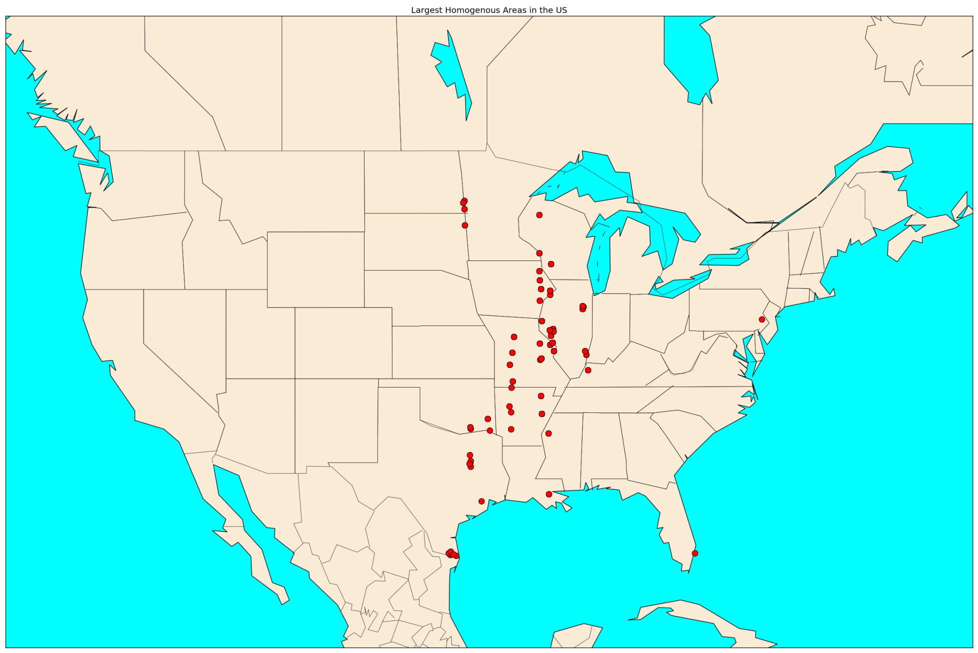


Table 6: Largest Homogeneous Areas in the US

DeWitt	MO
Flemington Village	MO
Ionia	MO
Albion	OK
Dakotacity	MN
Corinth	AR
Blue Eye	MO
Antoine	AR
Christine	ND
Garvintown	OK
Farmersburgcity	IA
Fargo	AR
Kenefic	OK
Great Bend	ND
Corona	SD
Buckingham Village	IL
Bellflower	MO
Kempton Village	IL
Colfax	ND
Brussels Village	IL
Fifth Street	TX
Armstrong	OK
Conesville	IA
Cottonwood	TX
Bayview	TX
Concord Village	IL
Hillview Village	IL
Delaware City	IA
Browns Village	IL

Springs:

By our construction of resting spring length, similar neighbors should attract while diverse neighbors should repel. By this original intuition we expected the Midwestern states to collapse upon itself over the course of the simulation, while places such as Austin Texas, a city very different than its neighbors, will push apart and expand.

Our results do not match our intuition, possibly because of the simplified physics in our discrete time simulation. Some places are pushed very far while others statically cling very close to their original locations This could be because ”clumps” of places with very similar neighbors have lots of mass and therefore are not pushed as far as their neighbors. This reveals a potential flaw that all places in our simulation have the same mass. Perhaps it would make sense for mass to scale with population so three 800 person CDPs cannot push away a 100000 population town.

The simulation slows down quickly at first but then hovers at rate of low oscillation above our convergence threshold. While we added a damping force it appears that the simulation does not converge (or does so much slower than we run out of AWS money).

Figure 8: spring simulation frame 0

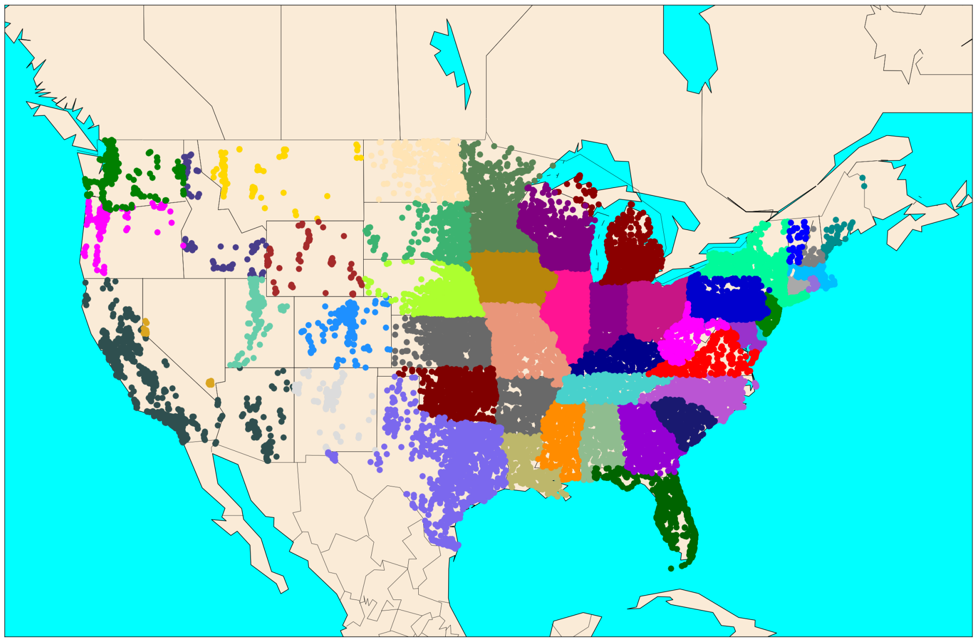
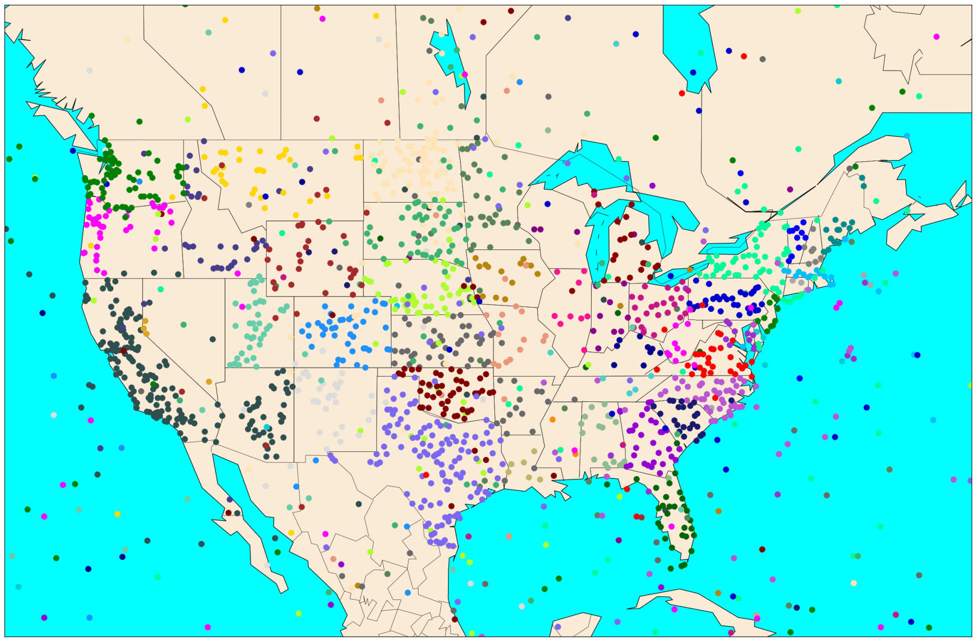


Figure 9: spring simulation frame 31



Works Cited

Kolko, Jed. "'Normal America' Is Not A Small Town Of White People." FiveThirtyEight. Last modified April 28, 2016. http://fivethirtyeight.com/features/normal-america-is-not-a-small-town-of-white-people/.