Computer Science Case Studies From the Census[*]

Christopher A. Healy
Department of Computer Science
Furman University
Greenville, SC 29613
chris.healy@furman.edu

Abstract

This paper describes some innovative assignments for CS 1 and CS 2 classes where students can write straightforward programs that discover useful facts directly from census data. This information exploits the geospatial population distribution of the United States. These assignments have been used successfully in Java and Python classes at this level, to reinforce skills in using file I/O and elementary data structures. In 2021, the U.S. Census Bureau began to release detailed results of the 2020 Census. This new data presents students with the opportunity to apply their programming skills to glean quantitative facts about the geographical distribution of the U.S. population and its diversity.

1 Introduction

Computing and the census share a long history. Every ten years the U.S. conducts a census of the population, and publishes extensive raw data. Herman Hollerith, whose firm was a corporate ancestor of IBM, developed a mechanical tabulator to read punch cards for the 1890 census. In 1946, the Census Bureau purchased the first commercially available electronic computer, the UNIVAC, for the 1950 census [2]. With the latest census having being conducted in 2020 and the results now being released, the time is ripe to consider incorporating the new data into computer science classes. This paper describes several software projects that have already been used in CS 1 and CS 2 classes based on data taken from the last census. By writing their own programs, students do not need to worry about the limitations of off-the-shelf software such as Excel or ArcMap.

2 The Data

The first data to be released by the Census Bureau is the data required by law to support redistricting [7]. For the purposes of disseminating data at various levels of detail, the United States is subdivided into the following census hierarchy: states, counties, census tracts, block groups, and blocks [1]. Summary statistics on these levels is given in Table 1. The averages and standard deviations have been rounded to the nearest integer. Note that only the census tract and block group have standard deviations that are significantly smaller than their means. This is because the Census Bureau makes a conscious effort to create these areas of relatively uniform size. Census tracts are commonly used by economists and sociologists to represent the intuitive idea of a neighborhood. On the other hand, the smallest geographical level in the census hierarchy is the block. The Census Bureau publishes only the most basic demographic data for blocks, such as race and sex, and how many people are aged 18 and older and thereby eligible to vote. For larger geographical areas, additional census data is available, such as income, educational attainment, and other economic statistics.

Table 1: Examples of small population areas in the US in 2020

Name of unit	Number	Mean population	S.D. of population
County	3,143	105,456	335,760
ZIP code (2010)	32,948	9,371	13,672
Census tract	83,848	3,953	1,689
Block group	238,437	1,390	682
Block	5,769,942	57	107

The Census Bureau also summarizes data by ZIP codes. These have the advantage that everyone knows their own ZIP code, while hardly anyone knows their census tract or block numbers. But there are some disadvantages to using ZIP codes for demographic analysis. ZIP codes were created for sorting mail, not for analyzing the population. ZIP codes exhibit a high degree of population variance, making them less suitable for our purposes. For example, many ZIP code areas have populations of under 100 or over 100,000, making comparisons difficult. In addition, demographic data on ZIP codes from the 2020 census have not yet been released, because they are not needed for redistricting. As a result, Table 1 shows ZIP code data for 2010 instead of 2020. Fortunately, we do not need to resort to using ZIP codes, as there are online tools where one can quickly look up a census tract number, for example [8].

The Census Bureau's downloadable file format can be cumbersome to use directly by introductory students. The raw redistricting data from the 2020 census contained over 13 GB of data, spread across over 200 separate files. Therefore, it is recommended that the instructor perform some preprocessing of the raw census data, in order to create suitable input files for the students. For example, one input file listing all of the blocks, and another input file listing census tracts. For example, the modified block file created by the author has a size of 483 MB (uncompressed) and contains the following information:

- 5-digit state/county code to facilitate sorting
- 2-letter state abbreviation
- County name, truncated to 20 characters
- Tract number, which could be up to 6 digits omitting the decimal point
- Block number, 4 digits
- Population of the block, up to 5 digits
- White, Black, Hispanic, and Asian population of the block
- Latitude and longitutde, to the nearest thousandth of a degree
- Area of the block in acres

Fortunately, the format of the input data in 2020 data is the same as in 2010 [1]. Therefore, the procedure for scanning the 2020 data can be done seamlessly for 2010, making longitudinal comparisons straightforward.

The format of the modified tract file, having a size of just 7 MB, is analogous to the block file. Within both files, the fields have a fixed width, so that students can use a substring function in order to extract individual values, instead of having to tokenize.

3 Project Ideas

This section describes possible programming exercises, and most of these are suitable for CS 1.

3.1 Simple Analyses

A simple task to start with is finding a centroid or population center. It is the mean of the latitudes and of the longitudes, weighted by the population of each area. To check their work, students are instructed to verify on a map that their answer is plausible. This task can easily be extended by computing the centroid of part of the U.S., such as a single state.

Next, using the block data, we can find the population density of each county or census tract. This way, we can classify areas as urban if their density exceeds 1000 people per square mile. While reading the data, the program needs to keep track of each tract or county encountered. So, students are recommended to use a built-in data structure such as the dictionary type in Python or the analogous Hashtable class in Java.

3.2 Create a Map

The map shown in Figure 1 was created with a program that simply reads the latitude-longitude locations of all census tracts from the 2020 census. A Java implementation is less than 70 lines long. The essential computation is to convert a latitude-longitude pair into a (x, y) pixel ordered pair. However, some care is needed to avoid creating a map that is upside-down or backwards. The lowest pixel numbers are in the upper-left corner of the image, while the lowest latitude and longitude values are in the opposite corner. As a sample classroom exercise, the author presented the source code to a CS 1 class where the output map was indeed displayed backwards, and the students were asked to find the error.

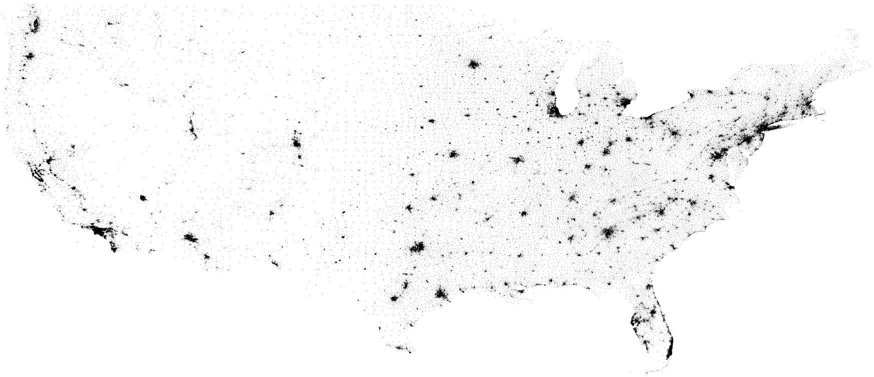

Figure 1: Pixel map of the US based on tract locations

Next, students are asked to modify this program, e.g. to change the resolution of the image, and to use blocks instead of tracts for additional detail. By reading the racial diversity counts of each block, a color-coded map can be created. Another use of color in a pixel map is to show areas of population growth and decline. The program reads a second file containing the results of the previous census for comparison. The Census Bureau provides comprehensive block data for 1990, 2000, 2010, and 2020.

An alternative to a pixel map is to create a KML file that can be read by Google Earth. The Census Bureau publishes Shapefile data on all census tracts. Each tract is geometrically defined as a polygon with latitude-longitude vertices. However, due to the complexity of these shapes (i.e. the large number of vertices), it may be difficult to render more than one state's worth of census tracts in one map. A simpler map could be created by representing each tract abstractly as a simple shape such as a square.

3.3 Discover Population Clusters

Scanning the census data can answer many questions pertaining to the concentration of population. For example, The U.S. Department of Education publishes the latitude and longitude of every public school in the country. Since our census data also locates every block by latitude and longitude, we can find how many people live within a specified distance from each school. Croft et al. [4] similarly used census data to determine how many people live within five miles of a physician. On the flip side, we can also find areas of the country that are remote. Thus, we can identify populations that are underserved or to detect possible areas of low light pollution for astronomy.

3.3.1 Radius

In particular, the radius problem asks this question: Determine how many people live within a fixed radius, (e.g. 10 miles) of some point, such as a downtown area or real-time GPS coordinates. Note that this problem can be solved by a single pass of the input file, and that it is not necessary to sort the tracts or blocks by distance.

Because distance is a critical calculation in the radius problem, there needs to be an accurate way to estimate the distance between two latitude-longitude points. It is also necessary to convert from degrees to miles. Students are given a formula to use in their programs. It is a modified form of the Cartesian distance formula that assumes that the earth is a sphere.

Once the students understand how to compute the population of some circular region, the next step is to iterate this procedure for the entire country. For each census tract t, calculate the population $P(t, n)$ living within n miles of the center of t. In doing so, it is straightforward to find the population density within n miles of every census tract in the United States. By scanning the resulting output, it is then easy to find areas of high or low population density (i.e. urban versus rural). Many businesses prefer to locate themselves in areas of relatively high population density. Therefore, the program could be run for a certain value of n, seeking a list of tracts where $P(t, n)$ is sufficiently high. For example, New York City has many tracts where a 10-mile radius encircles more than 7 million people.

3.3.2 Cluster

The cluster problem is analogous to the radius problem. The difference here is that instead of seeking a fixed distance from a certain point, we seek a fixed population size. For example, from a given point, how far do you need to go to encompass 50,000 people? This problem can be solved by sorting the tracts in ascending distance from the point in question. Since it is necessary to sort the areal units, it is important to use a smaller input file, such as census tracts, not blocks. Sorting all of the blocks in the United States would be overkill. If blocks are desired, then these should be limited to a single state or metropolitan area.

Once the clustering algorithm is implemented, then it too can be iterated over the entire country. The complete Python program used to perform the cluster analysis contains slightly over 100 lines of code. For each census tract t, it computes the radius $R(t, p)$ of the smallest circle centered at t containing a population a population of p. Then, the output can be scanned to find specific results of possible interest. To seek areas of high population density, one would search for a radius below some threshold. For example, if one wanted to find an area of 100,000 people in an urban density of at least 1000 per square mile, we need a circular area of area 100 square miles or less. Thus, we need a circle of radius 5.64 miles or less. Running the cluster program on the 2020 census data reveals that more than half of the census tracts in the country have this desired density.

One practical weakness of the radius and cluster programs is that they define only circular regions. As an alternative, a variation of the cluster problem is to partition the U.S. into nonoverlapping regions of similar population. This is similar to the real-world problems of redistricting and even the creation of census tracts themselves. In theory, the United States needs to be partitioned into 435 contiguous areas of equal population. In this case, there is also the added detail that each state is partitioned separately. Students can also focus on a single state alone to simulate the redistricting of a state legislature.

3.4 The Warehouse Problem

The warehouse problem is a generalization of finding a population center, and is presented to a CS 2 audience. The purpose of the problem is to find a set of N centrally located points across the United States. The problem can be stated as follows:

"Company XYZ is in retail trade, and its management would like to build several warehouses around the country to store merchandise. The locations of these warehouses should be chosen so as to minimize the distances from these warehouses to the general public. For each potential customer in the United States, we wish to estimate the distance from that household to its nearest warehouse in order to minimize shipping costs. The output is a set of optimal locations for the warehouses."

Alternatively, rather than speaking of warehouses literally, the problem could be stated as seeking to minimize the distance that the public needs to travel to a location of business. The parameter to this problem is the desired number of locations. And of course, the scope of the problem can be limited to one state or metropolitan area rather than the whole country.

It is an optimization problem that selects an optimal sample of census tracts. Blocks are not used because the run time of the program would increase by a factor of 69 (the average number of blocks in a tract) for a gain in precision that we might not appreciate. Inside an urban area, a census tract is often about one square mile in size, which is sufficiently precise for this problem. Students are given a pseudocode algorithm that they implement in Java. The algorithm generates random samples of tracts, and among all the trials it finds the sample with the minimum average distance. It can be summarized as follows:

```
Create an array of Tract objects.
 Create array A of 10 Tracts for the optimal selection.
 For trial = 1 to 25,000:
    Randomly generate a selection S of 10 Tracts
    For each Tract t:
       d = shortest distance from t to any Tract in S
    Compute weighted average of all values of d
       (i.e. weighted by the population of t)
    If this weighted average distance < distance for the
       array A, then reset A to be the selection S.
       And keep track of its weighted average distance,
       and also the trial number where minimum was found.
    If trial is a multiple of 200, print out the current
       value of A and its average distance so that the user
       can observe the progress of the algorithm.
```

Students are given this pseudocode and asked to implement it in Java. Being able to implement pseudocode is a fundamental skill in computer science. Most students in the author's CS 2 class found this to be a nontrivial task.

The two main stumbling blocks were understanding the concise language of the pseudocode, and making sure that no detail was omitted. The run time complexity of the algorithm is linear in the desired number of warehouses and in the number of trials. The program is under 200 lines long. In practice, for 10 warehouses and 25,000 trials, students found the program to take about three minutes. In an upper-level algorithms course, students could explore other optimization strategies, such as a genetic algorithm.

As a further experiment, once students have written this program, they can run it on various numbers of business locations in order to discover a mathematical relationship between distance in miles, d, and the population size, p, served by each location. In other words, we can develop a rule of thumb for estimating the average distance that one needs to travel to the nearest establishment of some type. In our experiments, the regression formula we obtained was

$$p = 2481d^{1.8}$$

For example, if there is approximately one cardiologist for every 14,000 people in the U.S., we should expect the average American to live 2.6 miles from one.

4 Conclusion

The goal of this work was to make it straightforward for students in CS 1 and CS 2 to write programs that analyze the geographic distribution of the U.S. population and its diversity at a high degree of detail. The author has had success over the years at incorporating the 2010 census data into several different assignments and student research projects, and work has begun on doing the same for 2020. With the newly released 2020 census data, everyone now has the opportunity to begin analyzing the most recent demographic data. Furthermore, this work need not be limited to the U.S. Table 2 shows a short list of countries that also publish census data online along with corresponding geospatial data [3, 5, 6, 9]. Thus, all of the programs described in this paper could also be carried out on these countries as well.

References

[1] United States Census Bureau. *2020 Census State Redistricting Data (Public Law 94-171) Summary File Technical Documentation.* 2021.

[2] Martin Campbell-Kelly and William Aspray. *Computer: A History of the Information Machine.* Westview Press, Boulder, Colorado, 2004.

Table 2: Examples of census hierarchies used in other countries

Country	Unit	Number	Mean size	S.D. of size
Australia	Statistical area 1	53,358	402	165
Australia	Statistical area 2	2,149	2,150	6,441
Australia	Statistical area 3	333	64,441	40,532
Brazil	Sector	310,120	615	354
Canada	Dissemination Area	54,963	627	536
Canada	Aggregate DA	4,920	7,003	3,965
New Zealand	Area unit	2,012	2,108	1,699

[3] Statistics Canada. Census profile, 2016 census. http://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/page_Download-Telecharger.cfm.

[4] Janet Croft, Hua Lu, Xingyou Zhang, and James Holt. Geographic accessibility of pulmonologists for adults with copd. *CHEST*, 150(3):544–553, 2016.

[5] Instituto Brasileiro de Geografia e Estatistica. Censo 2010 resultados. http://censo2010.ibge.gov.br/resultados.html.

[6] Australian Bureau of Statistics. Census advanced search. http://www.abs.gov.au/websitedbs/censushome.nsf/home/map.

[7] Congress of the United States. *Public Law 94-171*. 1975.

[8] Federal Financial Institutions Examination Council Geocoding System. https://geomap.ffiec.gov/FFIECGeocMap/GeocodeMap1.aspx.

[9] Statistics New Zealand. 2013 mesh block dataset. http://www.stats.govt.nz/Census/2013-census/data-tables/meshblock-dataset.aspx.