

Proportional Point Symbol Map

Using Python
(Project 6)

Sean Lim

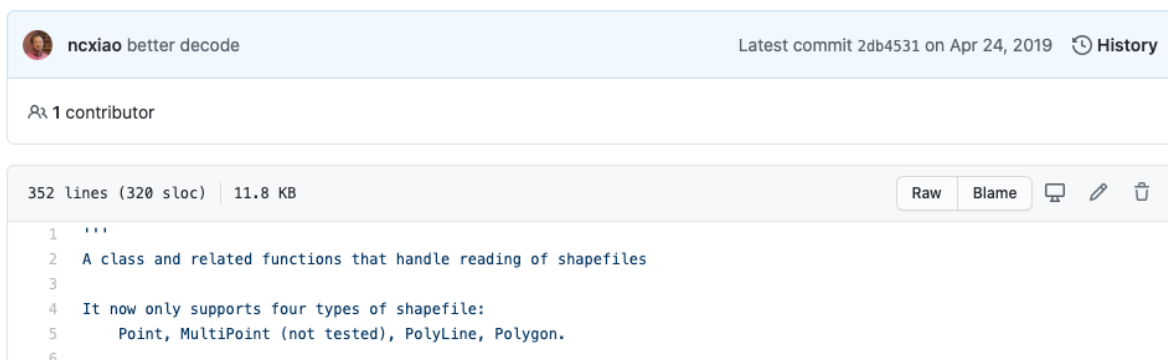
GEOG 5222

9/9/2020

Initial Setup for Running the Program

Seeing as this program uses a few modules from the gisalgs GitHub repository, the first step is to create a folder to store the files. Choose a location on your PC and create a new folder called 'gisalgs', inside the 'gisalgs' folder create a folder named 'geom'. Create two empty .py files and name them '__init__.py'. Place one file in the 'gisalgs' folder and the other in your 'geom' folder. Create three empty .py files and name them 'shapex.py', 'centroid.py', and 'point.py'. Store the three newly created Python files in your 'geom' folder.

Now you're ready to download the modules. Select the ['centroid.py'](#) file in the GitHub repository, click on the 'Raw' button and copy everything then paste it in your 'centroid.py' file you just created and save it. Repeat the same steps for the 'shapex.py' and 'point.py' files copying and pasting from the respective [shapex](#) and [point](#) modules.



```
1  '''
2  A class and related functions that handle reading of shapefiles
3
4  It now only supports four types of shapefile:
5      Point, MultiPoint (not tested), PolyLine, Polygon.
6  '''
```

Moving on, copy the path to your 'gisalgs' folder. Open up the Project.py file and edit the path in line 2 to the path for your PC. Paste the path between the single quotes in the sys.path.append function.

```
1  # Final Project for Sean Lim
2  import sys
3  sys.path.append('/Users/seanlim/Library/Mobile Documents/com~apple~CloudDocs/OSU/AU20/GE0G 5222/gisalgs')
```

Downloading the data

Head to [this link](#) to download the data for this project. Unzip the folder and open the folder. Look for the .shp file and copy the path to that file. Replace the path for the file_loc variable located at line 38.

```
file_loc = '/Users/seanlim/Library/Mobile Documents/com~apple~CloudDocs/OSU/AU20/GE0G 5222/uscnty48area/uscnty48area.shp'
```

Brief Overview

This program was created specifically for the one shapefile that was chosen. Therefore, it will not work with other shapefiles unless specific portions of the code are changed. Its objective is to give the user a sense of the area of water of every state in the United States relative to one another through the use of proportional point symbols. It utilizes absolute scaling to achieve that goal.

Explanation

Lines 41-64

Here can see lines 44 to 64 engulfed inside a for loop. The for loop makes it possible to retrieve every feature in the shapefile with every iteration of the loop. Every feature represents a state. Lines 45 and 46 retrieve the coordinates and geometry type of the feature and line 47 gets the quantitative attribute of the feature. In this case, it's the area of water for that respective feature.

Lines 49 to 55 deals with 'Polygon' geometry type features and it draws the feature using the path and patch functions. It also calculates the center of the feature using the centroid function, then adds it to the centroids list.

Lines 57 to 64 is similar to 49 to 55 except it is modified slightly to accommodate the 'MultiPolygon' type.

```
41 _, ax = plt.subplots()
42 attributes = []
43 centroids = []
44 for x in range(len(shape)):
45     coords = shape[x]['geometry']['coordinates']
46     polyType = shape[x]['geometry']['type']
47     attributes.append(shape[x]['properties']['AWATER']) # gets the quantitative attribute
48
49     if polyType == 'Polygon':
50         path = make_path(coords)
51         patch = PathPatch(path, facecolor='#CDCDCD', edgecolor='darkgrey')
52         ax.add_patch(patch)
53         poly = [Point(p[0],p[1]) for p in coords[0]]
54         area, center = centroid(poly)
55         centroids.append(center)
56
57     elif polyType == 'MultiPolygon': # accesses the polygons in the multipolygons each i
58         for i in range(len(coords)):
59             path = make_path(coords[i])
60             patch = PathPatch(path, edgecolor='darkgrey', facecolor='lightgrey')
61             ax.add_patch(patch)
62             poly = [Point(p[0],p[1]) for p in coords[0][0]]
63             area, center = centroid(poly)
64             centroids.append(center)
```

Lines 66-70

Line 67 is where the size of the point symbol is calculated. The formula was adapted from [this website](#). The attribute values are divided by the maximum possible attribute values and square rooted then multiplied by 1000. This is done to prevent the symbols from being too large and obstructing other parts of the map. 1000 was chosen as the ideal maximum radius of the symbols after testing. Note: the window will affect the scaling of the map and symbols. For the best accuracy, the window should be maximized.

Line 69 and 70 is where the point symbols are created and drawn on the map. It uses the Matplotlib plt module to draw markers shaped like circles. The x and y coordinates of the markers are determined by the results from the centroid function and the size is drawn from the sizes list.

```
66 maxAtt = max(attributes)
67 sizes = [1000*(value/maxAtt)**0.5 for value in attributes]
68
69 for i in range(len(shape)):
70     ax.scatter([p.x for p in centroids], [p.y for p in centroids], color='lightblue', edgecolor='darkblue', alpha=0.2, marker='o', zorder=2, s=[s for s in sizes])
```

Modules

Shapex

This program utilizes the shapex module developed by Dr. [Ningchuan Xiao](#). It's a pure Python program that enables users to read the shapefile format in Python. Currently, it only supports shapefiles types of point, multipoint, line, polyline, and polygon. It does not support coordinate reference systems.

Centroid & Point

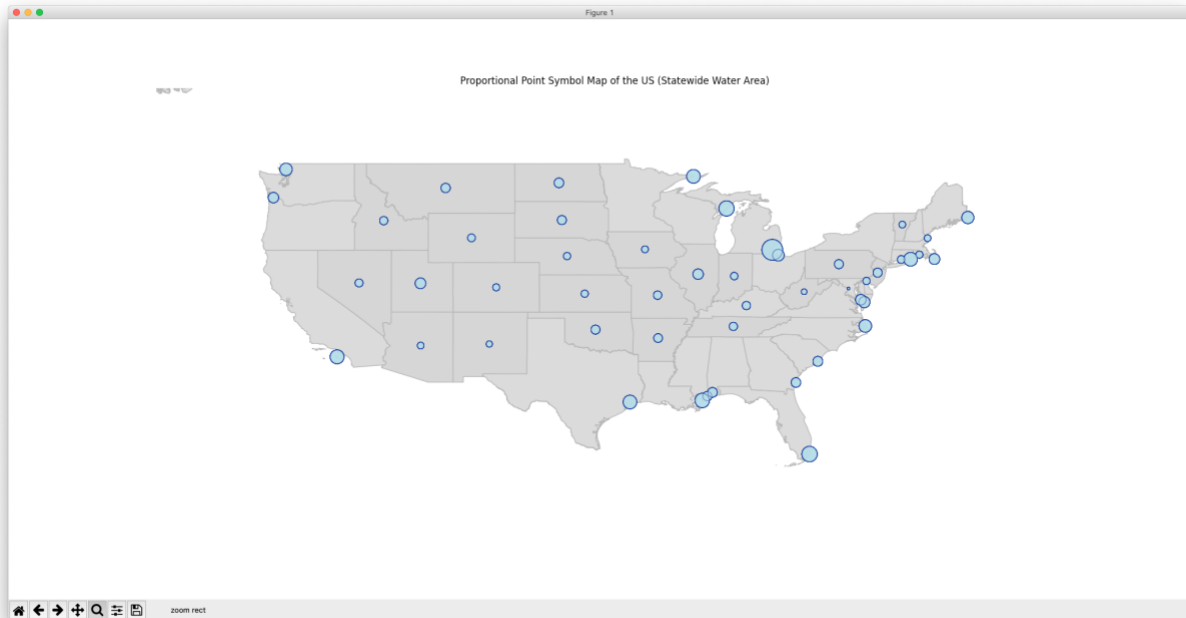
The centroid and point modules are also used in this program to calculate the centroids of every state to determine the center and where to position the point symbols. The modules can be found [here](#).

Matplotlib

The other module utilized in this program is the [Matplotlib](#) library. It's an open source library for creating static, animated, and interactive visualizations in Python. This program utilizes the pyplot, path, and patches modules to draw out the map.

Results

Shown below is the end result after running the program. A Python window will pop-up after several seconds. Please maximize the window to get a more accurate look at the point symbols. Note: this screenshot is zoomed in on the contiguous US, but it will be zoomed out initially.



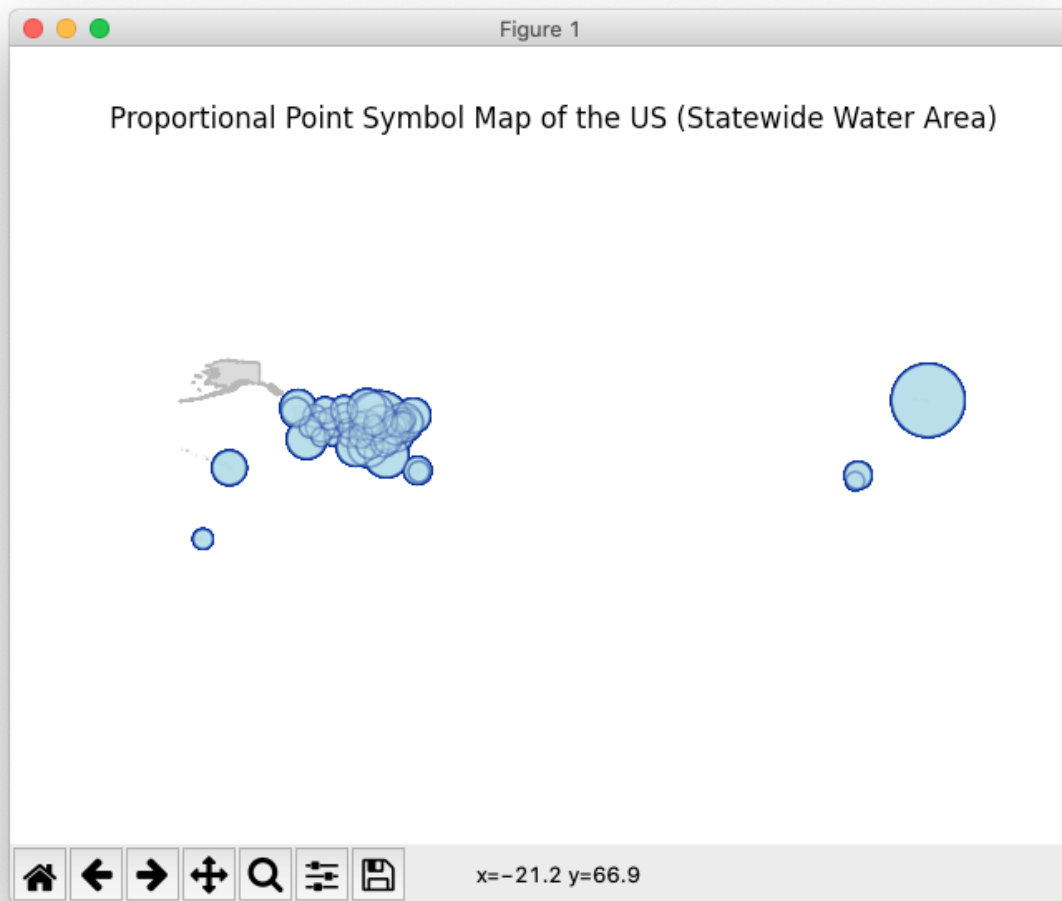
Discussion

Centroid function

As can be seen in the results, the centers of some states aren't exactly located at where one would expect the center to be. It's assumed that this is a result of using the centroid function; however, the actual cause of this error is unknown. An alternative to using the centroid function to determine the state centers would be using the geographical centers of the state although that would require another source of data.

Scaling of Point Symbols

The point symbols do not scale with the size of the window; therefore, the symbols may look too large when the window initially launches (see below). To remedy this, simply maximize the window.



Conclusion

To conclude, the objective of this project was to show the water area of every state relative to one another which this program does well through the use of a Proportional Point Symbol Map using absolute scaling. It gives the user a good understanding of how the quantitative attribute compares between features with a quick glance at the map. However, this program also shows the capability of Python as a programming language when several modules and libraries are put to use in unison.

Sources

https://wiki.gis.com/wiki/index.php/Proportional_symbol_map

https://www2.census.gov/geo/tiger/GENZ2018/shp/cb_2018_us_state_500k.zip

<https://github.com/gisalgs/geom>