Exercise: Mapping 2020 Political Contributions in Pennsylvania

Summary

This exercise maps contributions to presidential candidates by party in the 2020 election for zip codes in Pennsylvania.

Input Data

Several input files are available from the class Google Drive folder: **contrib_clean.pkl**, a pickled version of the aggregated individual contribution data from a previous exercise; **com_cand_info.csv**, another file from the earlier assignment that links committees, candidates, and parties; and **cb_2019_42_zcta510_500k.gpkg**, a geopackage file containing two layers: `state`, which is the boundary for the state itself, and `zip`, which has boundaries for all of the state's zip codes. See the tips section if you're interested in what the `cb_` and `_500k` in the file name mean.

Deliverables

There are six deliverables: a script called **pop.py** that retrieves population data from the Census API server; one called **by_party.py** that aggregates the contribution data to parties (rather than candidates, as was done in the previous exercise); a script called **join.py** that builds a new geopackage file; a QGIS project file called **contrib.qgz**, and two maps: **map_pc.png** and **map_party.png**

Instructions

A. Script pop.py

1. Follow the approach used in a couple of the previous exercises to retrieve Census variable `B01001_001E`, the total population, using a `for` clause of `"zip code tabulation area:*"` and an `in` clause of `"state:42"`. This time around omit the `"NAME"` from the query: it just restates the zip code and is not helpful here.
2. Build a dataframe called `pop` from the server's response.
3. Use a dictionary called `new_names` to rename columns `"B01001_001E"` to `"pop"` (same as the dataframe itself) and `"zip code tabulation area"` to "'zip'".
4. Set the index of `pop` to `"zip"` and then save the dataframe to `"pop.csv"`.

B. Script by_party.py

1. Import modules as needed.
2. Use `pandas.read_pickle()` to read `"contrib_clean.pkl"` into a variable called `contrib`.
3. Read `"com_cand_info.csv"` into a dataframe called `com_cand`.
4. Create a variable called `merged` by merging `com_cand` onto `contrib` using `on="CMTE_ID"`, `validate="1:1"`, and `indicator=True`".
5. Print the value counts for the `"_merge"` indicator and then drop it from `merged`.
6. Use a dictionary to rename `"CAND_PTY_AFFILIATION"` to `"party"`.
7. Create variable `pa` by using `.query()` to select the rows of `merged` where `"STATE"` is equal to `"PA"`.
8. Create `grouped` by using `.groupby()` to group `pa` by `"zip"` and `"party"`.
9. Create `amount` by applying `.sum()` to column `amt` of `grouped`.
10. Some of the amounts are negative when campaigns repay contributors. We'll use the `.where()` method to set those to zero. Set `amount` to the result of calling the `.where()` method of `amount` with the arguments `amount>0` and `0`. See the tips section for an explanation of how `.where()` works.

11. Set `wide` to the result of calling `.unstack()` on `amount` using the argument `level="party"`. The result will be a dataframe with one column per party.
12. A lot of the values in `wide` will be missing because there aren't contributions to every party in every zip code. Set those to zero by calling `.fillna()` on `wide` with arguments `0` and `inplace=True`.
13. Add a column called `"total"` that is equal to applying `.sum()` to `wide` with `axis="columns"`.
14. Save `wide` to `"by_party.csv"`. Have a look at it to make sure it's what you expect. Don't worry if you know the state and realize that a few of the zip codes are clearly wrong. Actual Pennsylvania zip codes in the range 15xxx to 19xxx. There are some in the data that are outside that range due to errors in the original data. Those records will be removed later.

C. Script join.py

1. Import `pandas`, `geopandas`, and `numpy` as `np`.
2. Create variable `pop` by reading `pop.csv`. Use a dictionary to set `dtype` for `"state"` and `"zip"` to `str`. The `dtype` setting is less crucial here than in other contexts because neither the state code nor any of the legitimate zip codes start with leading zeros but you'll make your life easier if you get in the habit of *always* reading FIPS and zip codes as strings. For example, you could switch the script to a New England state (leading 0's in zip codes) with no problem.
3. Create variable `contrib` by reading `"by_party.csv"`. Set the `dtype` for `"zip"` to `str`.
4. Create variable `both` by using `.merge()` to merge `contrib` onto `pop` using an outer one-to-one join with `"zip"` as the join key and `indicator` set to `True`.
5. Print the value counts of the merge indicator. Not everything will be `"both"`. Expect to see counts for both `"left_only"`: map zip codes that had no contributions at all, and `"right_only"`: zip codes that aren't in Pennsylvania.
6. Create `grouped` by grouping `both` by `"_merge"`.
7. Create `summary` by applying the `.agg()` method to column `"total"` of `grouped` using the argument `["count","sum","mean"]`. The result will be a dataframe with three columns of aggregate information: the record count for each case of `"_merge"`, and the sum and the mean as well. Note that these are totals by zip code so the means are much larger than a typical individual contribution.
8. Now set `trim` to the result of using `.query()` to pick out the records where `"_merge"` is not equal to `"right_only"` and then add `.copy()` to cause Pandas to create a copy of the data rather than a view.
9. Drop `"_merge"` from `trim`.
10. Set the amounts in zip codes with no contributions to 0 by calling `.fillna()` on `trim` with arguments `0` and `inplace=True`.
11. Compute per capita contributions by zip code by setting column `"pc"` of `trim` to the `"total"` column divided by the `"pop"` column.
12. The per capita values are very large in a few zip codes with small populations and large contributions. To focus on larger counties, set the per capita column to numpy's missing data value, `np.nan`, for counties with less than 100 people by setting `trim["pc"]` equal to the result of applying `.where()` to `trim["pc"]` using arguments `trim["pop"]>=100` and `np.nan`)
13. Next create a column called `"d_share"` in `trim` giving the share of total contributions that went to Democratic candidates.
14. Now read the zip code layer of geopackage file by setting `geo_zip` to the result of calling the `geopandas.read_file()` with arguments `"cb_2019_42_zcta510_500k.gpkg"` and `layer="zip"`.
15. Join do a left one-to-one join of `trim` onto `geo_zip`. Use `left_on="ZCTA5CE10"` and `right_on="zip"` to set the keys since the column names are different. Also set the indicator to True. A left join is appropriate

because we want to keep all of the zip codes in the shape file and want to discard any zip codes that don't match.

16. Print the value counts of the merge indicator and then drop it.
17. Write out the joined layer to layer `"zip"` of geopackage `"joined.gpkg"` by calling `.to_file()` on `joined` using arguments `"joined.gpkg"`, `layer="zip"` and `driver="GPKG"`. Note that if `"joined.gpkg"` already exists this call will add (or replace) layer `"zip"` but won't affect any other layers. To be sure that you get a clean copy of the file you may want to remove `"joined.gpkg"` whenever you rerun this part of the script.
18. For convenience, now read the state boundary layer by setting `geo_state` to the result of reading the geopackage file `"cb_2019_42_zcta510_500k.gpkg"` with `layer="state"`. Then use the `.to_file()` method to write it out to `"joined.gpkg"` using `layer="state"`.

D. Maps map_party.png and map_pc.png

1. Start QGIS and load both layers from `"joined.gpkg"`.
2. Put the state layer at the bottom of the list. Then set its fill color to black and its fill style to "FDiagonal". That will be useful because some of the zip codes have missing data and that has the effect of creating holes in layers that use graduated styling. The shading will make it easy to tell where that has happened because it will only be visible where the overlying data is missing. If you look carefully, you'll see that happen right away because there are two small parts of the state that have no zip code. See the Tips section for a brief explanation about places with no zip code.
3. Right-click on the zip layer and rename it to `party` for clarity. Then set its style to "Graduated" using `"d_share"` as the value, set the color ramp to `RdBu`, and set the mode to "Pretty Breaks" using 5 classes. This is where the striped state layer really pays off: it makes it easy to tell the difference between missing data and places where the `d_share` variable leads to the area being white.
4. The map will a little look better if you make lines showing the zip codes thinner. To do that, click on the colored area in the "Symbol" section of the "Graduated" settings (click on the color, not the drop down arrow). Then click on the "Simple Fill" symbol, scroll down until "Stroke width" is visible, and then click the down arrow next to it a couple of times until "Hairline" appears. Then click "Apply".
5. Export the map as `"map_party.png"`.
6. The party map shows the mix of contributions but doesn't say anything about the overall intensity with which people in each zip code contribute. To get at that, we'll look at per capita contributions. Duplicate the `party` layer and rename it `pc`. Then turn off the `party` layer by unchecking its box and turn on the new `pc` layer. Change the variable driving the "Graduated" style to `"pc"`, change the color ramp to "Greens", set the mode to "Natural Breaks", and use 4 classes. If all goes well, you should see that per capita contributions are actually very small for most of the state. However, they are very high for some zip codes near Philadelphia on the east, Pittsburgh on the west, and a few other places scattered around the state.
7. Export the map as an image to `"map_pc.png"`.
8. Save the QGIS project file as `"contrib.qgz"`.

Submitting

Once you're happy with everything and have committed all of the changes to your local repository, please push the changes to GitHub. At that point, you're done: you have submitted your answer.

Tips

- The Census provides "cartographic boundary" files that are lower resolution, and hence smaller in file size, than its TIGER/Line files. They are very suitable for thematic maps (such as the one in this exercise) where precise distance measurements are not needed. The files are available at several scales: the 500k here indicates that the file is at a scale of 1:500,000, which is the highest resolution among the cartographic

boundary options. The full US zip code map is about 60 MB at this resolution; for comparison, the TIGER/Line version is about nine times larger: it's about 530 MB.

- Some areas of the United States do not have zip codes: usually rural areas with very low population density. There are a couple of spots in Pennsylvania without zip codes; considerably more areas in New York, largely in the Adirondack region; and vastly more in the western US.
- The `.where()` method is a little counterintuitive, at least at first. When called like this, `B = A.where(test,value)`, it says that each element of `B` should be equal to the corresponding element of `A` when the matching element of `test` is True, otherwise, the value of `B` should be set to `value` instead of what was in `A`.
- This exercise is just scratching the surface of what could be done to analyze the contributions data. Many other variables could be added as well, including median income, race, education, employment status, and so on.