Exercise: Mapping Median Earnings for New York Counties

Summary

This exercise combines use of the Census API with use of the geopandas module and mapping via QGIS. It examines median earnings in New York counties over the last 12 months for the population 16 years and older who had any earnings. In case the term is unfamiliar, *earnings* refers labor income and excludes income from financial assets and transfer payments from the government.

Input Data

The only input file is **cb_2019_us_county_500k.zip**, a shapefile of US counties that can be obtained from one of the earlier assignments or the Google Drive folder for this exercise. The remaining data will be obtained from the Census via its API. However, if you'd like to run the demo script, which is highly recommended, you'll need two other files from the Google Drive folder: **cb_2019_us_state_500k.zip**, a shapefile of state boundaries, and **population.csv**, a file of population data downloaded from the Census API server.

Deliverables

There are five deliverables: a script called **earnings.py** that will request data from the Census, save a histogram of it in **earnings_hist.png**, and join it to a set of county polygons for New York counties, and write out the result as a geopackage file called **counties.gpkg**; a QGIS project file called **earnings_map.qgz**; and a PNG file called **earnings_map.png** that will be a heat map of median earnings.

Instructions

A. Script earnings.py

1. Import modules as needed.
2. Using an approach similar to that of the earlier Census API assignment, download variable `"B20002_001E"` (median earnings in the last 12 months) for all counties in New York and create a dataframe from the results. Please call the dataframe `earnings`. As before, include `'NAME'` in the API query so the county names will be included. However, since this task only involves a single Census variable, the steps in the earlier assignment related to `var_info` and the `"variable-info.csv"` file are not needed here. In particular, you **do not** need to set up and import a CSV file of variable names: you can simply set the `'get'` part of the payload to the string `"NAME,B20002_001E"`.
3. Create a new column in `earnings` called `'GEOID'` and set it to the result of concatenating the `'state'` and `'county'` columns of `earnings`. The result should be a 5-digit string that begins with 36.
4. Create a new numeric column in `earnings` called `"median"` that is the result of applying the `.astype(float)` method to the `"B20002_001E"` column of `earnings` and then dividing the result by 1000 to express the values in thousands of dollars.
5. Use the `to_csv()` method of `earnings` to write the data to `'earnings.csv'`. Use `index=False` because the index is just row numbers and does not need to be saved.
6. Use `fig, ax1 = plt.subplots()` to create a new single-panel figure as in previous exercises. Use the `dpi` keyword to set the resolution to 300.
7. Draw a histogram of median earnings by calling `sns.histplot()` with the arguments `data=earnings`, `x="median"`, `stat="density"`, and `ax=ax1`. The `stat` keyword indicates that the Y axis of the histogram should be the probability density.

8. Add a kernel density estimate to the figure by calling `sns.kdeplot()` with the arguments `data=earnings`, `x="median"`, `fill=True`, and `ax=ax1`. The `fill` option causes the area below the curve to be filled in with light shading.
9. Set the X axis label to `"Median Income in Thousands"`.
10. Tighten the layout and save the figure as `"earnings_hist.png"`.
11. Now create a trimmed-down dataframe for joining onto the shape file by setting `trim` equal to a dataframe consisting of just the `"GEOID"` and `"median"` columns of `earnings`.
12. Next, use the GeoPandas `gpd.read_file()` function to read the US county shapefile. Put the data into a variable called `geodata`.
13. Filter `geodata` down to New York counties by setting `geodata` equal to the result of applying the `.query()` method to it with an appropriate string for selecting records where the `"STATEFP"` field is `"36"`.
14. Now merge on the Census data by setting `geodata` to the result of using a left 1:1 join to merge `trim` onto `geodata` using `on="GEOID"` and with `indicator` set to `True`.
15. Print the value counts for the `"_merge"` column of `geodata` to verify that all 62 counties matched, and then drop the column.
16. Write out `geodata` to a geopackage file called `"counties.gpkg"`. Set the layer to `"earnings"`.

B. Files earnings_map.qgz and earnings_map.png

1. Create a new project in QGIS and add the "earnings" layer in the `"counties.gpkg"`.
2. Then, build a heat map of the median earnings using "natural breaks" with 5 classes. The natural breaks mode does a pretty good job of dividing the counties into groups that a person might pick looking at the histogram: a very low group, two middle groups, a high group, and a very high group having just one member. You can choose the color ramp but keep in mind that the map will be easiest to interpret if low income areas are light to signal less of the variable being shown and high income areas are dark to signal more.
3. Label the counties using the medians. Set the point size to 7, turn on text buffers, and choose the mode "Allow Overlaps if Required" in the Overlapping Labels section of the label Rendering tab (a small paintbrush).
4. Still adjusting the labels, go to the formatting tab (the one with "+ab" as part of its icon) and check "Formatted numbers" and set decimal places to 0.
5. Now save the QGIS project as `earnings_map.qgz` in the GitHub directory for the assignment.
6. Export the map as an image to the GitHub directory as well and call it `earnings_map.png`.

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

- A **very** big plus about doing joins via Python rather than in QGIS itself is that it's much easier to manage the data types of the columns. QGIS will see the data types used by Python and will not try to infer them as it does with CSV files. Among other things, that means it won't accidentally clobber FIPS codes by converting them from strings into numeric values.

- A very nice feature of geopackage files is that they can contain multiple layers. Shapefiles, in contrast, can only contain a single layer. This exercise only involves a single layer but other projects will routinely involve several layers. Having one geopackage file is a lot more convenient than having several shapefiles, each of which is itself a directory of several component files.