Exercise: Multi-Ring Buffers and Spatial Joins

Summary

This exercise uses multi-ring buffers and a spatial join to add the distance from an interstate to tax parcel records for Onondaga County. The data is then used to calculate the mean assessed value for residential and other property by that distance.

Input Data

There are two input files: (1) the GeoPackage `onondaga.gpkg` from the last assignment (or from the class Google Drive if you had trouble with the assignment); and (2) the file of tax parcel centroids from the previous exercise: `Onondaga-Tax-Parcels-Centroid-Points-SHP.zip`.

Deliverables

There are four deliverables: a QGIS project file called **by_distance.qgz**, an updated GeoPackage file called **onondaga.gpkg**, a PNG file called **by_distance.png**, and a Python script called **av_by_distance.py**.

Instructions

A. QGIS steps

1. If you were able to build the `boundary` and `interstate` layers correctly in the previous exercise, copy your `onondaga.gpkg` from that assignment to the GitHub directory for this one. If you had trouble, use the `onondaga.gpkg` in the class Google Drive folder for this assignment instead.
2. Start QGIS and add the `boundary` and `interstates` layers from `onondaga.gpkg`.
3. Set the project projection (icon in the lower right) to EPSG:6347, which is the NAD83(2011)/UTM zone 18N coordinate reference system.
4. Dissolve the interstate layer so all of its segments will be part of a single feature.
5. Build a multi-ring buffer from the dissolved interstate layer. Use 6 rings and set the distance between them to be 0.25 miles (a quarter mile).
6. Export the multi-ring buffer to `onondaga.gpkg` using the layer name `quarter-miles` and be sure to **uncheck** `fid` in the "Select fields to export and their export options" box. If you don't do this you'll get an error about the records in the layer not having unique IDs. See the Tips section for an explanation.
7. Remove the unexported `Dissolved` and `Multi-ring buffer` layers. You should have `boundary`, `interstates`, and `quarter-miles` left.
8. Add the property tax centroids to the map.
9. Save the project as `by_distance.gqz`.
10. Use a spatial join to add the ring information to the tax centroids. The input layer should be the centroids and the join layer should be `quarter-miles`. Use `intersects` for the geometric predicate and choose the one-to-one option for the join type (the choice that begins "Take attributes of the first . . . ").
11. Open the attribute table of the joined layer to verify that it now includes `ringId` and `distance` fields at the far right. In case you're curious, the distance column is in meters, which is the underlying unit used in the EPSG:6347 coordinate system.
12. Set the style of the joined layer to 'Categorized', use `ringId` as the value, and use 'Magma' as the color ramp.
13. Export the map as a PNG image called `by_distance.png`.

14. Now export the joined layer but instead of using GeoPackage as the format choose CSV. For the file name, use `by_distance.csv` and be sure it ends up in the GitHub directory for the assignment. To speed things up and keep the output file small, under "Select fields to export and their export options" please click on the "Deselect All" button and then check the following fields individually: `PROP_CLASS`, `TOTAL_AV`, `YR_BLT`, `NBR_BEDRM`, `ringId`, and `distance`. Then uncheck the "Add saved file to map" box and click "Ok" to save the CSV file. It should be about 8 MB or so.
15. Save the project again.
16. Close QGIS. You'll get a message about one or more temporary layers being lost. It's alerting you to the fact that we haven't saved the joined layer. However, we'll let it be discarded because it's quite large (600 MB) and can be rebuilt if necessary. Click "Yes" to proceed without saving the layer.

B. Python steps

1. Write a Python script called `av_by_distance.py` that does the following steps.
2. First, use Pandas to read the CSV file from the mapping step, `by_distance.csv`, into a variable called `parcels`.
3. Set variable `count_by_ring` to the result of calling the `value_counts()` method on `parcels['ringId']` with the argument `dropna=False`. Then print `count_by_ring` to see how many parcels there are in each zone.
4. Use the `fillna()` method to set the `'ringId'` column to 7 for records where it's missing (properties beyond buffer 6). Call `fillna()` on `parcels['ringId']` with the arguments 7 and `inplace=True`.
5. Set variable `pc` to the `'PROP_CLASS'` column of `parcels`.
6. Single family houses used year round are property class 210. To help pick out those records, set variable `is_house` to the result of testing whether the `parcels['PROP_CLASS']` is 210.
7. For comparison, set variable `is_bus` to the result of testing whether `parcels['PROP_CLASS']` is greater than or equal to 400. The classes 400 and up include commercial and industrial property, as well as several other categories. See the Tips section if you'd like a little more detail.
8. Create a new column in `parcels` called `'type'` and set it to `'other'`. This will set the default type for the parcel.
9. Now set the new column to `'house'` when `is_house` is true using the location operator `.loc[]` as shown below. The first argument to `.loc[]` indicates which rows are to be changed and the second argument indicates the column to be set.

   ```
   parcels.loc[is_house,'type'] = 'house'
   ```

10. Do something similar to set the appropriate values of `parcels['type']` to `'bus'` using `is_bus`.
11. Group the parcels by ring and type by setting variable `by_ring` to the result of applying the `groupby()` method to `parcels`. Since we're grouping on two characteristics the call to `groupby()` should be a list consisting of `'ringId'` and `'type'`: `['ringId','type']`.
12. Calculate the mean asset value by group by applying the `.mean()` method to `by_ring['TOTAL_AV']`. Call the result `mean_av`.
13. Now create a variable called `unstacked` that is equal to the result of applying the `.unstack()` and `.round(0)` methods to `mean_av`.
14. Print `unstacked`. Have a look at it and see what you think. It isn't necessary to write anything up because this is just a preliminary look at the spatial variation in the data. A more detailed analysis would use a regression to control for additional characteristics of the houses, such as the age or number of bedrooms (e.g., the kinds of things saved in the CSV file above), not just proximity to the road.

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 GeoPackage format requires that each record have a unique ID. QGIS usually uses `fid`, the feature ID field, as the unique ID if it's available. However, if you look at the attribute table for the ring buffer layer you'll notice that the `fid` field is the same for all of the buffers. That would cause an error saving the layer to a GeoPackage file. It's possible to avoid that by renumbering the features but frankly it's easier to simply exclude `fid` from the export. When it isn't present, QGIS will use the row number as the unique ID for each row, which is fine. Also, it's worth noting that this requirement applies to GeoPackage files and does NOT apply to SHP files: they do not require that `fid` be unique and you could save the buffer layer to a SHP file without any problem.
- In case you're curious, here's more information about the broad property classes used in the centroid file: 100-199 is agricultural; 200-299 is residential (a number of other types beyond single family year-round residences); 300-399 is vacant land; 400-499 is commercial; 500-599 is recreation and entertainment; 600-699 is community services (schools, churches, government, etc.); 700-799 is industrial; 800-899 is public services (utilities, etc.); 900-999 is parks and wild land. If you want to see the full list, add the 'NYS_Property_Class_codes.dbf' layer from the centroid file to your map and look at the attribute table. (Just have a look: no need to join it to anything.)