Exercise: Analyzing Residential Property by Zip Code

Summary

This exercise uses parcel-level property tax data from New York State to explore residential property values by zip code in Onondaga County.

Input Data

The input data is a file called **onondaga-tax-parcels.zip** that will need to be downloaded from the course Google Drive folder. It is derived from the New York State master file for the county but some fields have been removed to make the file smaller. It contains a single large CSV file called **onondaga-tax-parcels.csv**. You'll read it using the same approach you used with the high-resolution electricity data. The file has one record (line) for every parcel of land in Onondaga County. There are a bit more than 180,000 parcels in total and just over 144,000 of them are residential.

Each record has 20 fields (columns). The ones we'll use are: TOTAL_AV, the total assessed value of the parcel in dollars (total because it's the sum of the values of the land itself and the buildings on it); SQFT_LIV, the size of the living area in square feet; NBR_BEDRM, the number of bedrooms; YR_BLT, the year the house was built; CALC_ACRES, the size of the property in acres; PROP_CLASS, which indicates the classification of the parcel as agricultural, commercial, residential, or various other categories; and ZCTA5CE10, which is the 5-digit zip code for the property. ZCTA5 is a US Census designation that stands for "zip code tabulation area at the 5-digit level" and CE10 indicates that the zip code boundaries correspond to those that were in place during the 2010 census. There are other variables in the file (and even more in the original master) but those are the only ones we'll use for this exercise. The first few lines are available in `firstlines.csv` in the Google Drive folder.

Deliverables

A script called **parcels.py** that goes through the steps below. It will produce a CSV file called `parcels.csv` with summary information about residential housing in each zip code in the county. You'll also submit a short Markdown file called **parcels.md** with some observations about the results. Note that as in a previous exercise, `parcels.csv` itself is not a deliverable and will not be uploaded to GitHub.

Instructions

1. We'll use a number of modules so include the following at the start of the file:

   ```
   import csv
   import io
   import zipfile
   import numpy as np
   from collections import defaultdict
   ```

   In case you're curious, the import statements can be included in any order: there's nothing special about the ordering shown above.

2. Define a function called `newzip()` that takes no arguments. It's going to be used with `defaultdict()` to create a new data object for each zip code found in the file. Inside the function, create a variable called `new_zip` that is a dictionary with the following five keys: 'TOTAL_AV', 'SQFT_LIV', 'NBR_BEDRM', 'YR_BLT', and 'CALC_ACRES'. The names match fields in the dataset and need to be exactly as shown above, including case. The value for each key should be an empty list: e.g., `'TOTAL_AV':[]`. The function should then return `new_zip`.

3. Use the same steps as in the electricity data exercise to set up a zip object, open a file inside it, and use `io.TextIOWrapper` to translate from bytes to strings.

4. Then call `csv.DictReader()` to create an object called `inp_reader` that will return each line from the file as a dictionary. This should be done *instead of* calling `csv.reader()`, which is what was done in the

electricity exercise.

5. Use `open()` to open a file handle for an output file called `parcels.csv`. Remember to include the keyword argument `newline=''`. Then use a call to `csv.writer()` to create an object called `out_writer` for writing CSV data to the output file.
6. Create a variable called `grouped` using `defaultdict()` called with `newzip` as its argument: `defaultdict(newzip)`. As you'll see below, `grouped` will be a dictionary with one key for each zip code. This step will cause each key's value to be an object created by `newzip` that will be used to store information about the parcels in that zip code.
7. Set up a for loop iterating over `inp_reader` using `rec` (short for record) as the running variable. Remember that `rec` will be a dictionary with one key for each field in the input file. Within the loop do the following:
    1. Create a variable called `this_zip` that is equal to the value of `rec` for the key `'ZCTA5CE10'`. Then create `this_av` equal to `rec` at key `TOTAL_AV`, and create `this_class` from `rec` at key `PROP_CLASS`.
    2. We'll skip parcels that are missing the variables just defined. To do that, check `this_zip` to see if it's blank (equal to ""). If it is, use a `continue` statement to go on to the next line of the file. Make similar checks for `this_av` and `this_class`: if any one of the three variables is blank, use a `continue` statement to go on to the next record in the file.
    3. Now convert `this_class`, which is a string, into an number using the `int()` function. Call the result `num_class`.
    4. Next we'll skip the parcels that aren't residential. Residential parcels have property class values from 200 to 299. Use an `if` statement to check whether `num_class` is below 200 or above 299. If it is, use a `continue` statement to go on to the next record.
    5. Create a variable called `zip_entry` that is equal to the value of `grouped` for key `this_zip`. It will be a dictionary created by `newzip` that will hold information about properties in whatever zip code was given in the current record.
    6. Now create a for loop that iterates over `zip_entry` using running variable `field`. The loop will go through the keys of the dictionary to capture the data from `rec` at the corresponding keys. Inside the loop there should be a single line:

        ```
        zip_entry[field].append( float(rec[field]) )
        ```

        To understand what this does, consider what happens when `field` is 'TOTAL_AV'. Starting at the right side of the statement, the `rec[field]` will get the record's value of `TOTAL_AV`. That will then be converted into a numeric value using `float()`, and the result will be appended to the `TOTAL_AV` list for the zip code that was set up when `newzip()` was called. This setup would make it very easy to pick other values out of the input record: as long as they were numeric, they could just be added to `newzip` and they would be picked up here automatically.
8. After the end of the `inp_reader` loop use `out_writer.writerow()` to write a list of headers for the columns of the output file. The list should consist of the following strings: `'zipcode'`, `'parcels'`, `'total_value'`, `'value25'`, `'value50'`, `'value75'`, `'sqft'`, `'bedrooms'`, `'year'`, `'acres'`.
9. Create two variables, `total_n` and `total_value` and set each one to 0.
10. Set up a for loop iterating over `sorted(grouped)` using `zipcode` as the running variable since the keys of `grouped` are zip codes. Within the loop do the following:
    1. Set variable `zip_entry` equal to the value of `grouped` at key `zipcode`.
    2. Set variable `av_list` to the value of `zip_entry` for key `'TOTAL_AV'`. That will be the list of assessed values for the zip code.
    3. Set variable `this_sum_av` to the value of `np.sum()` given the argument `av_list`. The result will be the total value of residential property in the zip code.

    4. Add `this_sum_av` to `total_value`.
    5. Set variable `this_n` to the value of `len()` given the argument `av_list`. It will be the number of residential properties in the zip code.
    6. Add `this_n` to `total_n`.
    7. Set variable `av_levels` to the value of `np.percentile()` using the arguments `av_list` and `[25,50,75]`. The result will be a list with three elements: `av_levels[0]` will be the 25th percentile of the assessed values in the zip code, `av_levels[1]` will be the median, etc. The interquartile range will be useful for understanding the degree of variability within each zip code.
    8. Set variables `av25`, `av50` and `av75` to the corresponding values of `av_levels`. It's easiest to use a tuple but it's also OK to pull the elements out one at a time using subscripts 0, 1 and 2.
    9. Use `np.median()` four times to create four variables: `this_sqft` from `zip_entry['SQFT_LIV']`, and `this_bdr`, `this_yr`, and `this_acres` from `zip_entry` at keys `'NBR_BEDRM'`, `'YR_BLT'`, and `'CALC_ACRES'`.
    10. Create a list called `results` that consists of `zipcode`, `this_n`, `this_sum_av`, `av25`, `av50`, `av75`, `this_sqft`, `this_bdr`, `this_yr`, and `this_acres`.
    11. Use `out_writer.writerow()` to write `results` to the output file.
11. After the end of the `grouped` loop, add two print statements. The first should give a message with some appropriate text and the total number of properties, `total_n`. The second should be a message with appropriate text giving the total value of residential property in the county in billions of dollars (divide `total_value` by `1e9`) rounded to one digit after the decimal.
12. Look over the results in `parcels.csv` and then use a text editor to write a short Markdown file called `parcels.md`. The first line should be the heading `# Notes on Onondaga Property Values` and the remainder of the file should be a few brief comments (say 5 to 6 or so) about interesting things in the file. For example, what zip code has the largest total value of residential property? How large is that? What zip code has houses with the highest median value? Oldest houses? Are there any zip codes with very large differences between the median value and the 75th percentile, etc.

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

- In later exercises we'll join this data to information from the Census and other sources to examine the county in more detail. We'll also map it using GIS.
- Data on all tax parcels in New York State is available from the NYS geographic information system (GIS) data repository. A link is available from the resources section of the class web page.