Fall 2018

Jupyter: Intro to Data Science - Lecture 2 Loading and Summarizing Data

Grant Long
CUNY City College

NYC Tech-in-Residence Corps

Data Dive 2: Loading and Summarizing Data

Part 1: Tapping the Census API for Valuable Insights

The U.S. Census API is an extremely rich data source by itself, but also one that can enhance and provide insights into other datasets. In the first part of today's exercise, we'll put

Helpful links:

- API user guide (https://www.census.gov/data/developers/guidance/api-user-guide.html)
- List (https://api.census.gov/data/2016/acs/acs5/variables.html) of census variables provided by the API.
- Sample queries (https://api.census.gov/data/2016/acs/acs5/examples.html) for different geographies
- List (https://api.census.gov/data.html) of available apis
- The Census Factfinder (https://factfinder.census.gov/faces/nav/jsf/pages/index.xhtml) is a great place to browse data or look up variable names.

Today we'll be using the 2016 ACS 5-year sample. The details of the different ACS samples are beyond the context of this class, but suffice to say that these estimates are built of five years of data from the ongoing ACS.

The Census API, like many others, requires a *developer key* to use the API regularly. You can sign up for one here (https://api.census.gov/data/key_signup.html). Thus far, I have not needed one for these exercises, but just to be on the safe side, it's nice to have one handy.

In [ ]:
```
import pandas as pd
import requests

# In case I need a key, I'll leave these here.
import os
census_key = os.getenv('CENSUS_KEY')
```

Using the API

Some of the many useful variables in the ACS:

- `B01001_001E` : Total Population
- `B19013_001E` : Median Household Income
- `B25058_001E` : Median Rent
- `B01002_001E` : Median Age

We'll need the base url for the *endpoint*, in this case `https://api.census.gov/data/2016/acs/acs5?` , along with the relevant parameters in our query. The `requests` package has a handy feature that allows you to enter your parameters as a dictionary using the `params` argument.

In [ ]:
```
url = 'https://api.census.gov/data/2016/acs/acs5?'
params = {'get' : 'NAME,B01001_001E,B19013_001E,B25058_001E',
          'for' : 'county:*',
          'in' : 'state:*'}

r = requests.get(url, params=params)
print(r.url)
```

Load our results into a data frame. In this case, we set the first row of results as the headers.

In [ ]:
```
census_df = pd.DataFrame(r.json()[1:], columns=r.json()[0])

census_df.head(10)
```

Let's clean things up a bit by renaming our columns and resetting the index

In [ ]:
```
census_df["County Number"] = census_df.state.astype(str) + census_df.cou
nty.astype(str)

census_df = (census_df
             .rename(columns={'NAME' : 'County Name',
                              'B01001_001E' : 'Total Population',
                              'B19013_001E' : 'Median Household Income',
                              'B25058_001E' : 'Median Rent',
                             })
             .set_index('County Name')
             .drop(columns=['state', 'county']))

for col in ['Median Household Income', 'Total Population', 'Median Rent'
]:
    census_df[col] = census_df[col].astype(int)

census_df.sort_values(by='Median Household Income', ascending=False).hea
d(10)
```

Now we can start to have some fun by digging through the data.

In [ ]:
```
census_df.sort_values(by='Median Rent', ascending=False).head(20)
```

Writing to File

Pandas also has simple functionality to *write* to files, not just read from them. To write to a csv file to store our data, we can just use `.to_csv()` (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_csv.html).

In [ ]:
```
# census_df.to_csv('../../../web/www/data/census_counties.csv')
```

In [ ]:

Exercise: America's Oldest Counties

In [ ]:
```
# todo: find america's oldest counties by median age
```

In [ ]:

In [ ]:

In [ ]: