DataEng: Data Integration Activity

This week you will gain hands-on experience with Data Integration by combining data from two distinct sources into a unified DataFrame for analysis.

Submit: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code before submitting for this week.

Your job is to integrate **county-level COVID-19 data** with the **ACS Census Tract data for 2017** to build a model that allows you to relate COVID numbers with economic data such as population, per capita income and poverty level. To do this you should build a pandas DataFrame that has a row per USA county (there are more than 3000 counties in the USA) and includes the following columns:

County - name of the county
State - name of the state in which the county resides
TotalCases - total number of COVID cases for this county as of February 20, 2021
Dec2020Cases - number of COVID cases recorded in this county in December of 2020
TotalDeaths - total number of COVID deaths for this county as of February 20, 2021
Dec2020Deaths - number of COVID deaths recorded in this county in December of 2020
Population - population of this county
Poverty - % of people in poverty in this county
PerCapitaIncome - per capita personal income for this county

We hope that you make it all the way through to the end. Regardless, use your time wisely to gain python programming experience and learn as much as you can about building integrated multi-source data models using python and pandas.

For this activity you should use whichever environment is convenient for you to develop with python 3 and pandas. You are not required to use GCP, but you can use it if you prefer.

Submit: In-class Activity Submission Form

A. Aggregate Census Data to County Level

Your integration will use two different dimensions: location (as indicated by state and county) and time. You should greatly simplify your processing and reduce your time by pre-processing your data along each of these dimensions.

The ACS data is separated into "Census Tracts" which are regions within counties that correspond to groups of approximately 4000 people. The Census Bureau defines these to help organize the actual job of collecting census data, but this grouping can make your Data Engineering job more more challenging. This level of detail is not needed for your county-level analysis, and you can greatly decrease your efforts by aggregating per-tract data to the county level.

Create a python program that produces a one-row-per-county version of the ACS data set. To do this you will need to think about how to properly aggregate Census Tract-level data into County-level summaries.

In this step you can also eliminate unneeded columns from the ACS data.

Question: Show your aggregated county-level data rows for the following counties: Loudon County Virginia, Washington County Oregon, Harlan County Kentucky, Malheur County Oregon

```
TotalPop       374558.000000
IncomePerCap    50455.645745
Poverty             3.689598
Name: (Virginia, Loudoun County), dtype: float64
TotalPop       572071.000000
IncomePerCap    35369.047499
Poverty            10.321202
Name: (Oregon, Washington County), dtype: float64
TotalPop       27548.000000
IncomePerCap   15456.971032
Poverty           35.669482
Name: (Kentucky, Harlan County), dtype: float64
TotalPop       30421.000000
IncomePerCap   17567.504323
Poverty           24.298225
Name: (Oregon, Malheur County), dtype: float64
```

B. Simplify the COVID Data

You can simplify the COVID data along the time dimension. The COVID data set contains day-level resolution data from (approximately) March of 2020 through February of 2021. However, you will only need four data points per county: total cases, total deaths, cases reported during December of 2020 and deaths reported during December 2020.

Create a python program that reduces the COVID data to one line per county.

Question: Show your simplified COVID data for the counties listed above.

```
cases_total    2496450.0
deaths_total     35820.0
cases_dec       376223.0
deaths_dec        4729.0
Name: (Virginia, Loudoun), dtype: float64
cases_total    2157339.0
deaths_total     22455.0
cases_dec       424620.0
deaths_dec        3860.0
Name: (Oregon, Washington), dtype: float64
cases_total    205984.0
deaths_total     3994.0
cases_dec       38959.0
deaths_dec        506.0
Name: (Kentucky, Harlan), dtype: float64
cases_total    453634.0
deaths_total     7770.0
cases_dec       82916.0
deaths_dec       1465.0
Name: (Oregon, Malheur), dtype: float64
```

C. Integrate COVID Data with ACS Data

Create a single pandas DataFrame containing one row per county and using the columns described above. You are free to add additional columns if needed. For example, you might want to normalize all of the COVID data by the population of each county so that you have a consistent “number of cases/deaths per 100000 residents” value for each county.

Question: List your integrated data for all counties in the State of Oregon.

```
            TotalPop  IncomePerCap  ...cases_dec/100k deaths_dec/100k
County                          ...
Baker          15980  25820.273154  ...  1.867742e+03     21.25340
Benton         88249  30872.824361  ...  3.023411e+04    245.33222
Clackamas     399962  37550.849108  ...  1.047141e+06  12498.81250
Clatsop        38021  28114.625523  ...  5.489852e+03     17.86987
Columbia       50207  28459.688051  ...  1.077392e+04    133.55062
```

Coos	62921	26007.212997	...	1.183292e+04	95.01071
Crook	21717	24238.814477	...	2.399294e+03	42.56532
Curry	22377	26925.536399	...	1.508434e+03	16.11144
Deschutes	175321	31574.934092	...	1.796865e+05	987.05723
Douglas	107576	25001.732924	...	4.043782e+04	1037.03264
Gilliam	1910	24178.000000	...	1.715180e+01	0.47750
Grant	7209	25154.161742	...	3.528806e+02	2.23479
Harney	7195	24397.712578	...	2.674382e+02	2.44630
Hood River	22938	29594.972796	...	4.438044e+03	49.54608
Jackson	212070	27080.538534	...	3.277224e+05	3509.75850
Jefferson	22707	22956.835293	...	8.237645e+03	92.87163
Josephine	84514	24348.609449	...	2.297091e+04	343.97198
Klamath	66018	23793.066679	...	2.978600e+04	246.24714
Lake	7807	21004.589343	...	4.182991e+02	5.93332
Lane	363471	27032.412179	...	6.499443e+05	8050.88265
Lincoln	47307	25782.113704	...	1.137308e+04	237.48114
Linn	121074	24448.467359	...	8.075878e+04	1078.76934
Malheur	30421	17567.504323	...	2.522388e+04	445.66765
Marion	330453	24791.074831	...	1.208800e+06	18901.91160
Morrow	11153	21742.930153	...	2.589615e+03	25.31731
Multnomah	788459	34848.165612	...	5.364817e+06	80769.73996
Polk	79666	25928.364057	...	4.061851e+04	591.91838
Sherman	1635	34226.000000	...	1.397925e+01	0.00000
Tillamook	25840	25458.191138	...	1.770040e+03	0.00000
Umatilla	76736	22153.237007	...	1.189370e+05	1262.30720
Union	25810	26585.728710	...	7.285389e+03	87.23780
Wallowa	6864	26897.389860	...	1.582838e+02	6.38352
Wasco	25687	24727.506132	...	5.782401e+03	159.51627
Washington	572071	35369.047499	...	2.429128e+06	22081.94060
Wheeler	1415	21268.000000	...	5.079850e+00	0.02830
Yamhill	102366	28539.604791	...	7.112492e+04	831.21192

D. Analysis

For each of the following, determine the strength of the correlation between each pair of variables. Compute the correlation strength by calculating the Pearson correlation coefficient R for pairs of columns in your DataFrame. For example, if you have a DataFrame df with each row representing a distinct county, and columns named ‘TotalCases’ and ‘Poverty’, then you can compute R like this:

```
R = df['TotalCases'].corr(df['Poverty'])
```

For any R that is > 0.5 or < -0.5 also display a scatter plot (see pandas scatterplot and seaborn documentation for information about how to display scatter plots from DataFrame data).

The COVID numbers should be normalized to population (# of cases per 100,000 residents) so that different sized counties are comparable. So for example, “COVID total cases” below really means “((COVID total cases in county * 100000) / population of county)”.

1. Across all of the counties in the State of Oregon
    a. COVID total cases vs. % population in poverty
    b. COVID total deaths vs. % population in poverty
    c. COVID total cases vs. Per Capita Income level
    d. COVID total cases vs. Per Capita Income level
    e. COVID cases during December 2020 vs. % population in poverty
    f. COVID deaths during December 2020 vs. % population in poverty
    g. COVID cases during December 2020 vs. Per Capita Income level
    h. COVID cases during December 2020 vs. Per Capita Income level

R values: a: 0.28707860802137714
b: 0.36053911582413317
c: -0.3756850276147199
d: -0.4618665950518557 (closest to being <-0.5)
e: 0.2981520301331537
f: 0.302726951283147
g: -0.38539719437305037
h: -0.4559551950686659

2. Across all of the counties in the entire USA
    a. COVID total cases vs. % population in poverty
    b. COVID total deaths vs. % population in poverty
    c. COVID total cases vs. Per Capita Income level
    d. COVID total cases vs. Per Capita Income level
    e. COVID cases during December 2020 vs. % population in poverty
    f. COVID deaths during December 2020 vs. % population in poverty
    g. COVID cases during December 2020 vs. Per Capita Income level
    h. COVID cases during December 2020 vs. Per Capita Income level

R values: a: 0.027214697313371956
b: 0.059189106541519
c: -0.00507307002801522
d: 0.05515146394377751
e: -0.03925939432078254

f: -0.012893029195812696
g: -0.07464579768384975
h: -0.03106197744738741

Note that this exercise does not constitute a competent, thorough statistical analysis of the relationships between immunological data and demographic data. It is just an illustration of the types of computations that might be accomplished with an integrated data set.