

# 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

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.

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

There is not such a contry in Virginia state, but in Tennessee state

```
#Create new df use dict list
new_df=pd.DataFrame(new_acs)
print(new_df)
print(new_df[(new_df['County']=='Loudon County')&(new_df['State']=='Tennessee')])
```

### Washington County Oregon:

	County	State	TotalPop	Poverty	IncomePerCap
173	Washington County	Oregon	572071	10.321202	34970.817308

Harlan County Kentucky:

```
print(new_df[(new_df['County']=='Harlan County')&(new_df['State']=='Kentucky']]))
```

	County	State	TotalPop	Poverty	IncomePerCap
1984	Harlan County	Kentucky	27548	35.669482	16010.363636

Malheur County Oregon:

```
print(new_df[(new_df['County']=='Malheur County')&(new_df['State']=='Oregon']]))
```

	County	State	TotalPop	Poverty	IncomePerCap
1397	Malheur County	Oregon	30421	24.298225	17966.428571

## B. Simplify the COVID Data

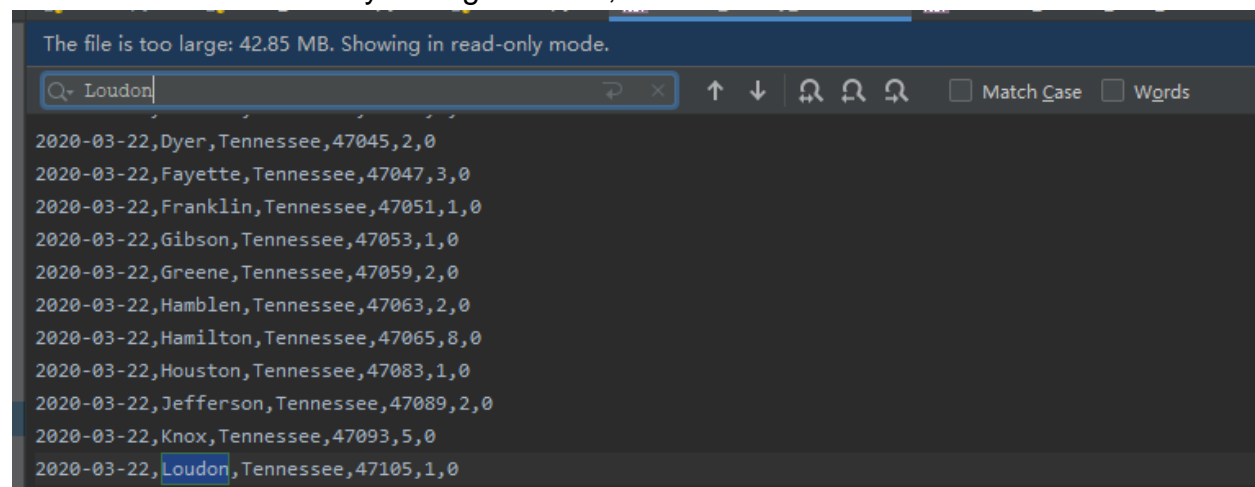
You can simplify the COVID data along the time dimension. The COVID data set contains day-level resolution data from (approximately) March of 2020 through February of 2021. However, you will only need four data points per county: total cases, total deaths, cases reported during December of 2020 and deaths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

Loudon County Virginia:

There is not such a contry in Virginia state, but in Tennessee state



The file is too large: 42.85 MB. Showing in read-only mode.

Q- Loudon

2020-03-22,Dyer,Tennessee,47045,2,0  
2020-03-22,Fayette,Tennessee,47047,3,0  
2020-03-22,Franklin,Tennessee,47051,1,0  
2020-03-22,Gibson,Tennessee,47053,1,0  
2020-03-22,Greene,Tennessee,47059,2,0  
2020-03-22,Hamblen,Tennessee,47063,2,0  
2020-03-22,Hamilton,Tennessee,47065,8,0  
2020-03-22,Houston,Tennessee,47083,1,0  
2020-03-22,Jefferson,Tennessee,47089,2,0  
2020-03-22,Knox,Tennessee,47093,5,0  
2020-03-22,Loudon,Tennessee,47105,1,0

```

new_cvd_df=pd.DataFrame(new_cvd)
print(new_cvd_df[(new_cvd_df['County']== 'Loudon')&(new_cvd_df['State']== 'Tennessee'))
print(new_cvd_df[(new_cvd_df['County']== 'Washington')&(new_cvd_df['State']== 'Oregon'))
print(new_cvd_df[(new_cvd_df['County']== 'Harlan')&(new_cvd_df['State']== 'Kentucky'))
print(new_cvd_df[(new_cvd_df['County']== 'Malheur')&(new_cvd_df['State']== 'Oregon'))

```

	County	State	Totalcase	Totaldeath	Deccase	Decdeath
1516	Loudon	Tennessee	543679	4653.0	100459	812.0
	County	State	Totalcase	Totaldeath	Deccase	Decdeath
172	Washington	Oregon	2157339	22455.0	424620	3860.0
	County	State	Totalcase	Totaldeath	Deccase	Decdeath
2040	Harlan	Kentucky	205984	3994.0	38959	506.0
	County	State	Totalcase	Totaldeath	Deccase	Decdeath
1453	Malheur	Oregon	453634	7770.0	82916	1465.0

## 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.

The screenshot shows a PyCharm IDE with a Python script in the top pane and its output in the bottom pane. The script filters a DataFrame for the state of Oregon. The output is a table with 36 rows and 9 columns, showing data for various Oregon counties.

```

116
117 print(final_df[(final_df['State']=='Oregon')])
118
...
if __name__ == '__main__':

```

	County	State	TotalPop	...	Totaldeath	Deccase	Decdeath
24	Yamhill	Oregon	102366	...	6010.0	69481	812.0
109	Wheeler	Oregon	1415	...	53.0	359	2.0
172	Washington	Oregon	572071	...	22455.0	424620	3860.0
197	Wasco	Oregon	25687	...	3039.0	22511	621.0
224	Wallowa	Oregon	6864	...	449.0	2306	93.0
285	Union	Oregon	25810	...	1533.0	28227	338.0
299	Umatilla	Oregon	76736	...	10661.0	154995	1645.0
358	Tillamook	Oregon	25840	...	92.0	6850	0.0
547	Sherman	Oregon	1635	...	0.0	855	0.0
853	Polk	Oregon	79666	...	5480.0	50986	743.0
1132	Multnomah	Oregon	788459	...	58787.0	680418	10244.0
1140	Morrow	Oregon	11153	...	1447.0	23219	227.0
1361	Marion	Oregon	330453	...	34089.0	365801	5720.0
1386	Malheur	Oregon	30421	...	7770.0	82916	1465.0
1476	Linn	Oregon	121074	...	5949.0	66702	891.0
1487	Lincoln	Oregon	47307	...	3117.0	24041	502.0
1580	Lane	Oregon	363471	...	10372.0	178816	2215.0
1597	Lake	Oregon	7807	...	348.0	5358	76.0
1646	Klamath	Oregon	66018	...	2857.0	45118	373.0
1709	Josephine	Oregon	84514	...	2638.0	27180	407.0
1749	Jefferson	Oregon	22707	...	2630.0	36278	409.0
1787	Jackson	Oregon	212070	...	7221.0	154535	1655.0
1881	Hood River	Oregon	22938	...	1444.0	19348	216.0
1963	Harney	Oregon	7195	...	291.0	3717	34.0
2085	Grant	Oregon	7209	...	94.0	4895	31.0
2132	Gilliam	Oregon	1910	...	76.0	898	25.0
2361	Douglas	Oregon	107576	...	3983.0	37590	964.0
2404	Deschutes	Oregon	175321	...	4141.0	102490	563.0
2482	Curry	Oregon	22377	...	393.0	6741	72.0
2503	Crook	Oregon	21717	...	1134.0	11048	196.0
2544	Coos	Oregon	62921	...	969.0	18806	151.0
2568	Columbia	Oregon	50207	...	1363.0	21459	266.0
2641	Clatsop	Oregon	38021	...	287.0	14439	47.0
2666	Clackamas	Oregon	399886	...	20040.0	261810	3125.0
2998	Benton	Oregon	88249	...	2304.0	34260	278.0
3068	Baker	Oregon	15980	...	663.0	11688	133.0

[36 rows x 9 columns]

## 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 deaths 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

That's the result before I use "((COVID total cases in county \* 100000) / population of county)" as total cases and deaths

```
-0.07089269198503927
-0.1189805553572352
0.5017392157477014
0.5480509220071149
-0.12700544510036524
-0.05212457584397503
0.5664710550960274
0.4904963842809488
-----
```

That shows c,d,g >0.5

But , After I used the correct data, the R changed:

```

0.05892612153270464
-0.05920696263968653
-0.18911188089929534
-0.08110969278652067
-0.04420436252862988
-0.04420436252862988
-0.16609894649992787
-0.16609894649992787

```

I did not believe the result, so I try to print R between mortalityrate and poverty, Per Capita Income.

```

-0.10174142470698021
0.09493326771748183

```

That output makes sense.

```

#counties in Oregon
print(final_df[(final_df['State'] == 'Oregon')]['CaseperOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['Poverty']))
print(final_df[(final_df['State'] == 'Oregon')]['DeathperOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['Poverty']))
print(final_df[(final_df['State'] == 'Oregon')]['CaseperOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['IncomePerCap'])))
print(final_df[(final_df['State'] == 'Oregon')]['DeathperOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['IncomePerCap'])))
print(final_df[(final_df['State'] == 'Oregon')]['DeccaseOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['Poverty'])))
print(final_df[(final_df['State'] == 'Oregon')]['DecdeathOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['Poverty'])))
print(final_df[(final_df['State'] == 'Oregon')]['DeccaseOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['IncomePerCap'])))
print(final_df[(final_df['State'] == 'Oregon')]['DecdeathOHT'].corr(final_df[(final_df['State'] == 'Oregon')]['IncomePerCap'])))
print(final_df[(final_df['State'] == 'Oregon')]['Totalmortalityrate'].corr(
    final_df[(final_df['State'] == 'Oregon')]['Poverty']))
print(final_df[(final_df['State'] == 'Oregon')]['Totalmortalityrate'].corr(
    final_df[(final_df['State'] == 'Oregon')]['IncomePerCap'])))
print('-----')

```

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 deaths 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

```

-0.015264248057292427
-0.014201111877233193
-0.004452907482373785
0.016397211741910396
-0.011769923318298398
-0.011769923318298398
-0.007642466773902303
-0.007642466773902303

```

I still don't believe the result, and print R of mortality rate.

```
-0.012185547083777414  
0.021843767339206486
```

```
count    3131.000000  
mean      1.919202  
std       1.268073  
min       0.000000  
25%      1.075751  
50%      1.659326  
75%      2.490369  
max      11.429481  
Name: Totalmortalityrate, dtype: float64
```

```
#counties in USA  
print(final_df['CaseperOHT'].corr(final_df['Poverty']))  
print(final_df['DeathperOHT'].corr(final_df['Poverty']))  
print(final_df['CaseperOHT'].corr(final_df['IncomePerCap']))  
print(final_df['DeathperOHT'].corr(final_df['IncomePerCap']))  
print(final_df['DeccaseOHT'].corr(final_df['Poverty']))  
print(final_df['DecdeathOHT'].corr(final_df['Poverty']))  
print(final_df['DeccaseOHT'].corr(final_df['IncomePerCap']))  
print(final_df['DecdeathOHT'].corr(final_df['IncomePerCap']))  
print(final_df['Totalmortalityrate'].corr(final_df['Poverty']))  
print(final_df['Totalmortalityrate'].corr(final_df['IncomePerCap']))  
  
print(final_df['Totalmortalityrate'].describe())
```

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.