Intro to Python and Pandas

Wesley Hamilton

University of North Carolina - Chapel Hill

wham@live.unc.edu

01/21/20

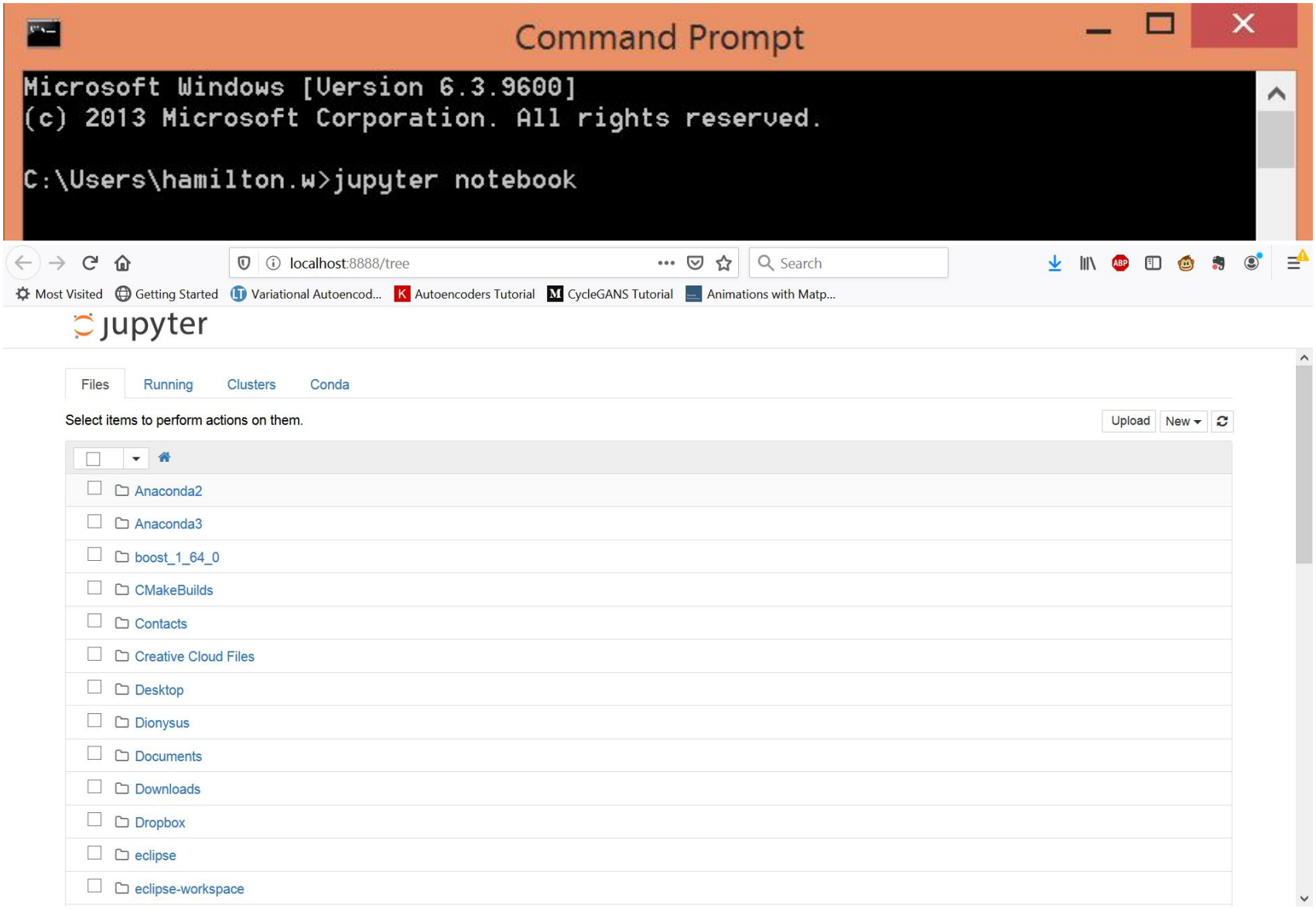
1. Intro to Python

2. Intro to Pandas, MCM 2019 Problem C

Running Jupyter/iPython

1. Open up a command line/terminal
2. Type **jupyter notebook**
3. Navigate to where you saved the .ipynb files

Running Jupyter/iPython



Using Python

- Using packages
- Lists
- For loops
- Functions
- If/else
- Numpy
- Plotting

Using packages

- Python has some built in functionalities, but external packages improve the experience
- **numpy** is used for all basic numerical business
- **scipy** is used for fancier numerical business
- **matplotlib.pyplot** is used for plotting
- Multiple packages can offer the same functionality: **PIL**, **scipy**, and others offer image io and other operations
- Other packages can be installed using **pip** (depending on your system, it's easier or less easier)

Using packages

- **import package**; import a package
- **import package as p**; import a package, and use the package under a different name
- **from package import subpackage**; from a package, import something specific

Using packages, code

Importing packages

In [ ]:
```
#for pretty much all numerical needs
import numpy as np

#for pretty much all plotting needs
import matplotlib.pyplot as plt

#for pretty much all data needs
import pandas as pd
```

Lists and for loops

- A list is a collection of things: **a = [1, "test", 3.14]**
- Indexing starts at 0: **a[0] = 1, a[2] = 3.14, a[-1] = 3.14**
- For loops iterate over lists: **for i in [1,2,3]: print(i)** should print 1, 2, 3
- You can append by using **a.append("thing")**
- A commonly used list is **range(N)** for an integer *N*
- The length of a list can be gotten by calling the built-in Python command **len()**

Lists and loops, code

Lists and loops

In [ ]:
```
#define a new list
a = [1,"test", 3.14]

#calling elements of a (and printing them)
print(a[0])
print(a[-1])
print(a[-3])

#looping with lists, the usual for loop
for i in a:
    print(i)

#you can append objects to lists
a.append([1,2,4.12])

#you can print whole lists
print(a)
print(len(a))

#range(N) is a useful list
for i in range(5):
    print(i+3)
```

Functions

We can define functions in two ways:

1. The "classical" way: **def f(x): return x*x**
2. The "lambda" way: **f = lambda x: x**2**

Functions, code

Functions

In [ ]:
```
#define a function using def
def function(x):
    new_val = x**2
    return new_val

#try out the function
print(function(1))

print(function(3.14))

print(function("test"))

#lambda functions are quick, ``in place'' ways to define functions
lam_function = lambda x: x**2

print(lam_function(1))

print(lam_function(3.14))
```

If/else, and while

- If/else control statements work in the following way: **if condition: do this else: do this different thing**
- While loops work similarly to almost any other programming language (as opposed to for loops): **while condition: do this**

If/else, code

If/else, and while

In [ ]:

```
#define a function to make running the if/else statement easier
def fun(x):
    if x:
        print("This is true!")
    else:
        print("Better luck next time...")
    return

#try running fun with a false, then true, statement
cond = False
fun(cond)
fun(1)

#a simple while loop
i = 0

while(i<10):
    i = 2*i+1
    print(i)
```

Dictionaries

- Dictionaries are special to Python, and act like lists but can be indexed by anything
- **testDictionary = {"test":8, 8:3.14, 0:"SIAM"}**
- Elements are called by passing their keys: **testDictionary["test"]** , etc.
- Available keys can be ascertained by calling **testDictionary.keys()**

Dictionaries, code

```
Dictionaries
```

In [ ]:

```
#define a new dictionary
test_dict = {"entry":"out", "out":1,1.0:"this one"}

#see what the keys are
print(test_dict.keys())

#try calling the elements
print(test_dict["out"])
print(test_dict[1])

#define a blank dictionary
blank_dict = {}

#add elements on the fly
blank_dict[1] = "out"
blank_dict["out"] = "this one"
blank_dict["entry"] = 1

print(blank_dict)
```

Using Numpy

- Numpy has a number of constants built in, like **np.pi** and **np.exp(1)**
- Numpy has a bunch of built in functions, like **np.sin(), np.exp(), np.cos()**, etc.
- Numpy supports complex numbers via j: **1.9 + 2.2j** and **-2 + 3j** are interpreted as complex numbers.
- Some other useful Numpy functions: **np.sort(), np.max(), np.min(), np.mean(), np.argmax(), np.argmin(),** etc.

Using Numpy

- Numpy arrays are the improved python lists:
  - **np.linspace(a,b,N)** generates an array of N equally spaced points between a and b inclusive
  - **np.arange(N)** generates a list of integers from 0 to N-1 inclusive
  - **np.zeros(N)** generates an array of N zeros
  - **np.random.rand(n1,n2)** generates an $n_1 \times n_2$ array with entries uniformly sampled from $[0, 1]$
  - **a[:,2]** returns everything in the array **a** that's in the third column (this is slicing)
  - **a[2,:]** returns everything in the third row
  - If **l1 = [1,3,4]**, then calling **a[l1,:]** will return the elements in rows 1, 3, 4 (with 0-indexing)
  - **a[:3,4:]** will return all rows until the third (so rows 0, 1, 2), and all columns after (and including) column 4

Using Numpy, code

Numpy

In [ ]:
```
#define a few arrays
spaced_out = np.linspace(-1,10,14)
ranged_out = np.arange(14)
random_array = np.random.rand(3,5)

print(spaced_out)
print(ranged_out)
print(random_array)

#adding arrays
summed_out = spaced_out + ranged_out
print(summed_out)

#some slicing
print(random_array[:,2])
print(random_array[2,:])

#index slicing
indices = [1,3,4]
print(random_array[:,indices])

bools = [True, False, True]
print(random_array[bools,:])
```

Using Matplotlib

- **plt.plot(x,y)**; produces a plot where subsequent (x_i, y_i) are connected by straight lines
- **plt.scatter(x,y)**; produces a scatter plot of disjoint (x_i, y_i)
- **plt.title("title")**; adds a title to the plot
- **plt.show()**; shows the plot
- **plt.plot(x,y,c = "b")**; for most of matplotlib's plotting, passing the optional argument **c="r"** sets the colour of what's plotted to r(ed)

Using Matplotlib, code

```
Plotting
```

In [ ]:
```
#generate some data
x = np.linspace(-1,np.pi,100)
y = np.sin(2.1*x)

plt.plot(x,y,c="r")
plt.xlabel("x-coords")
plt.ylabel("sin values")
plt.show()

#generate alternate data
z = np.cos(2.1*x)

plt.scatter(y,z, c = "g", marker = "x")
plt.title("Some points on a circle")
plt.show()
```

Questions?

Questions?

Dataset: Opioid crisis

2019 MCM Problem C:

Background: The United States is experiencing a national crisis regarding the use of *synthetic* and *non-synthetic opioids,* either for the treatment and management of pain (legal, prescription use) or for recreational purposes (illegal, non-prescription use). Federal organizations such as the Centers for Disease Control (CDC) are struggling to "save lives and prevent negative health effects of this epidemic, such as opioid use disorder, hepatitis, and HIV infections, and neonatal abstinence syndrome."[1] Simply enforcing existing laws is a complex challenge for the Federal Bureau of Investigation (FBI), and the U.S. Drug Enforcement Administration (DEA), among others.

There are implications for important sectors of the U.S. economy as well. For example, if the opioid crisis spreads to all cross-sections of the U.S. population (including the college-educated and those with advanced degrees), businesses requiring precision labor skills, high technology component assembly, and sensitive trust or security relationships with clients and customers might have difficulty filling these positions. Further, if the percentage of people with opioid addiction increases within the elderly, health care costs and assisted living facility staffing will also be affected.

Dataset: Opioid crisis

Supplied with this problem description are several data sets for your use. The first file (MCM_NFLIS_Data.xlsx) contains drug identification counts in years 2010-2017 for narcotic analgesics (synthetic opioids) and *heroin* in each of the counties from these five states as reported to the DEA by crime laboratories throughout each state. A drug identification occurs when evidence is submitted to crime laboratories by law enforcement agencies as part of a criminal investigation and the laboratory's forensic scientists test the evidence. Typically, when law enforcement organizations submit these samples, they provide location data (county) with their incident reports. When evidence is submitted to a crime laboratory and this location data is not provided, the crime laboratory uses the location of the city/county/state investigating law enforcement organization that submitted the case. For the purposes of this problem, you may assume that the county location data are correct as provided.

The additional seven (7) files are zipped folders containing extracts from the U.S. Census Bureau that represent a common set of *socio-economic factors* collected for the counties of these five states during each of the years 2010-2016 (ACS_xx_5YR_DP02.zip). (Note: The same data were not available for 2017.)

Dataset: Opioid crisis

Problem:

Part 1. Using the NFLIS data provided, build a mathematical model to describe the spread and characteristics of the reported synthetic opioid and heroin incidents (cases) in and between the five states and their counties over time. Using your model, identify any possible locations where specific opioid use might have started in each of the five states.

If the patterns and characteristics your team identified continue, are there any specific concerns the U.S. government should have? At what drug identification threshold levels do these occur? Where and when does your model predict they will occur in the future?

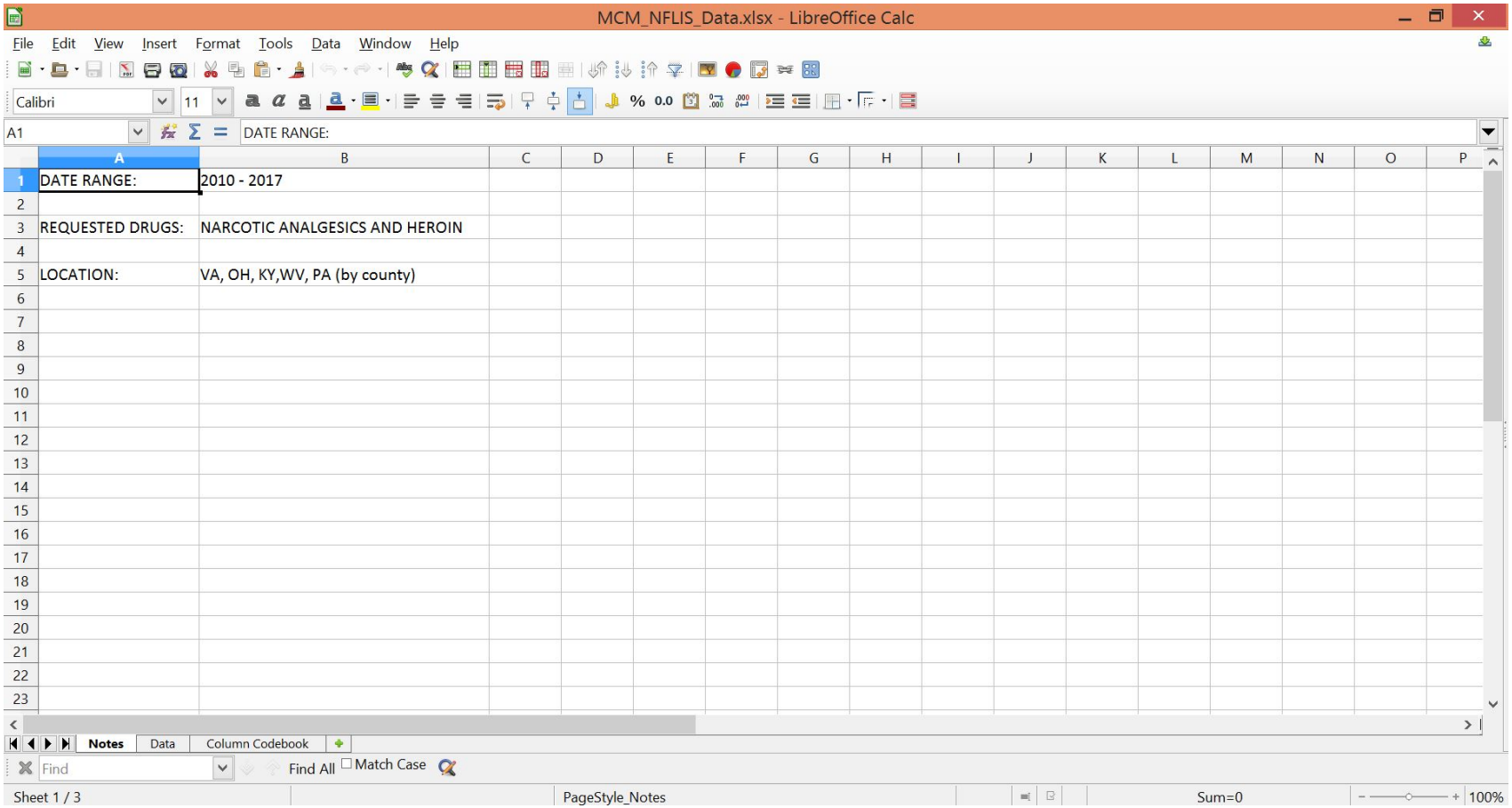
Part 2. Using the U.S. Census socio-economic data provided, address the following questions:

There are a good number of competing hypotheses that have been offered as explanations as to how opioid use got to its current level, who is using/abusing it, what contributes to the growth in opioid use and addiction, and why opioid use persists despite its known dangers. Is use or trends-in-use somehow associated with any of the U.S. Census socio-economic data provided? If so, modify your model from **Part 1** to include any important factors from this data set.

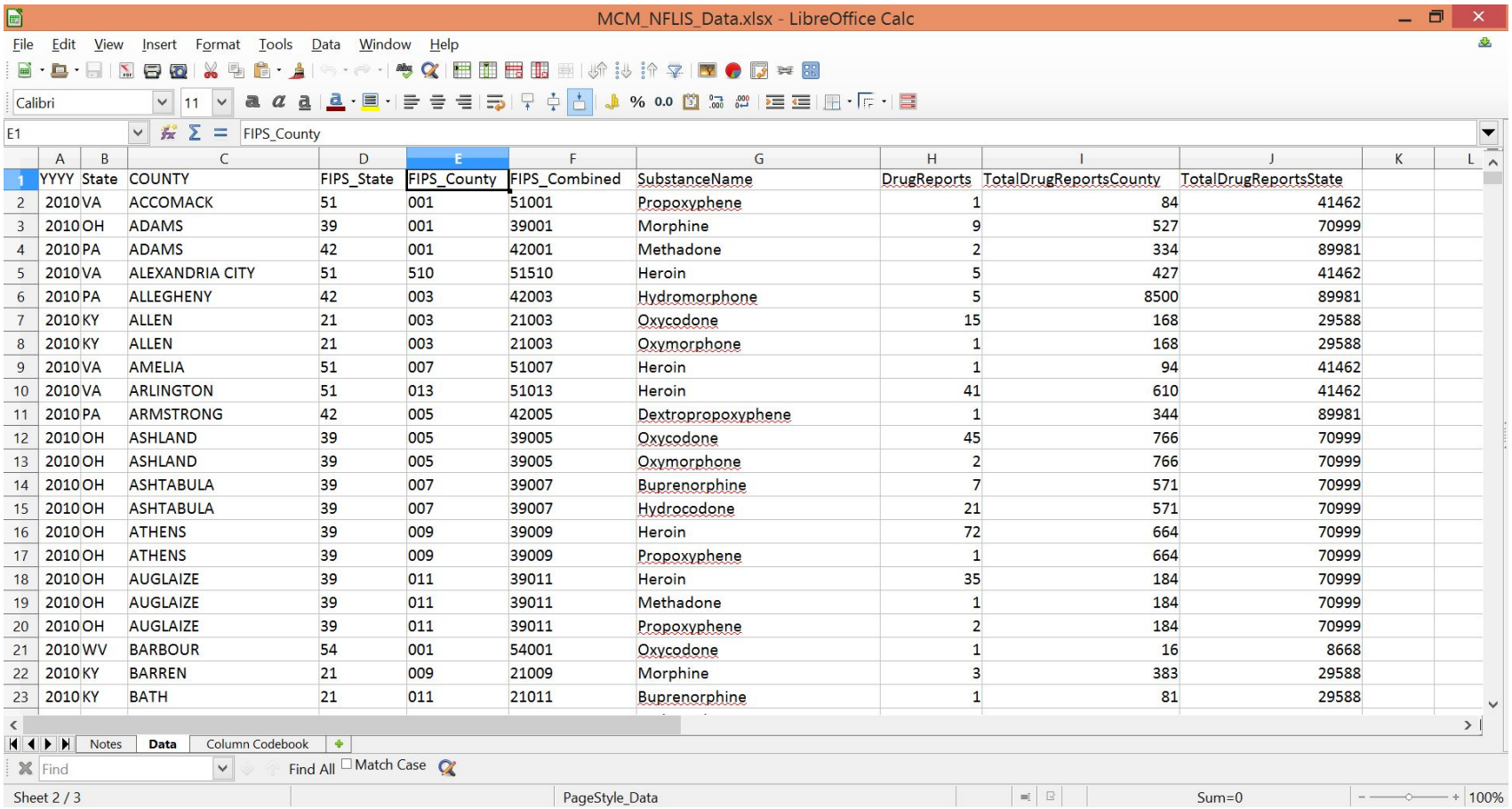
Download the data

http://www.mathmodels.org/Problems/2019/MCM-C/index.html

Quick perusal of the data



Quick perusal of the data


MCM_NFLIS_Data.xlsx - LibreOffice Calc

File Edit View Insert Format Tools Data Window Help

Calibri 11

E1 FIPS_County

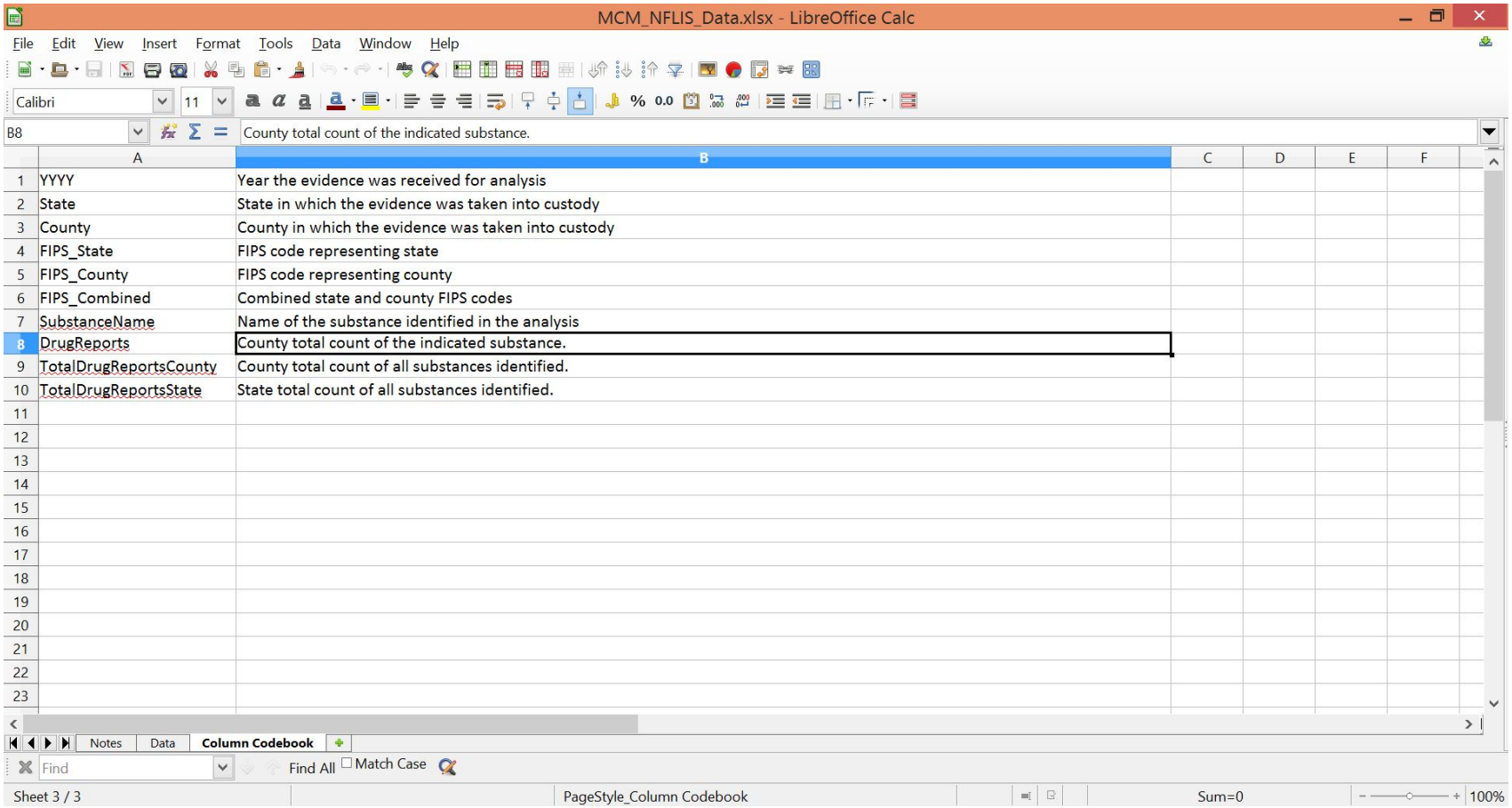
	A	B	C	D	E	F	G	H	I	J	K	L
1	YYYY	State	COUNTY	FIPS_State	FIPS_County	FIPS_Combined	SubstanceName	DrugReports	TotalDrugReportsCounty	TotalDrugReportsState		
2	2010	VA	ACCOMACK	51	001	51001	Propoxyphene	1	84	41462		
3	2010	OH	ADAMS	39	001	39001	Morphine	9	527	70999		
4	2010	PA	ADAMS	42	001	42001	Methadone	2	334	89981		
5	2010	VA	ALEXANDRIA CITY	51	510	51510	Heroin	5	427	41462		
6	2010	PA	ALLEGHENY	42	003	42003	Hydromorphone	5	8500	89981		
7	2010	KY	ALLEN	21	003	21003	Oxycodone	15	168	29588		
8	2010	KY	ALLEN	21	003	21003	Oxymorphone	1	168	29588		
9	2010	VA	AMELIA	51	007	51007	Heroin	1	94	41462		
10	2010	VA	ARLINGTON	51	013	51013	Heroin	41	610	41462		
11	2010	PA	ARMSTRONG	42	005	42005	Dextropropoxyphene	1	344	89981		
12	2010	OH	ASHLAND	39	005	39005	Oxycodone	45	766	70999		
13	2010	OH	ASHLAND	39	005	39005	Oxymorphone	2	766	70999		
14	2010	OH	ASHTABULA	39	007	39007	Buprenorphine	7	571	70999		
15	2010	OH	ASHTABULA	39	007	39007	Hydrocodone	21	571	70999		
16	2010	OH	ATHENS	39	009	39009	Heroin	72	664	70999		
17	2010	OH	ATHENS	39	009	39009	Propoxyphene	1	664	70999		
18	2010	OH	AUGLAIZE	39	011	39011	Heroin	35	184	70999		
19	2010	OH	AUGLAIZE	39	011	39011	Methadone	1	184	70999		
20	2010	OH	AUGLAIZE	39	011	39011	Propoxyphene	2	184	70999		
21	2010	WV	BARBOUR	54	001	54001	Oxycodone	1	16	8668		
22	2010	KY	BARREN	21	009	21009	Morphine	3	383	29588		
23	2010	KY	BATH	21	011	21011	Buprenorphine	1	81	29588		

Notes Data Column Codebook

Find Find All Match Case

Sheet 2 / 3 PageStyle_Data Sum=0 100%

Quick perusal of the data



	A	B	C	D	E	F
1	YYYY	Year the evidence was received for analysis				
2	State	State in which the evidence was taken into custody				
3	County	County in which the evidence was taken into custody				
4	FIPS_State	FIPS code representing state				
5	FIPS_County	FIPS code representing county				
6	FIPS_Combined	Combined state and county FIPS codes				
7	SubstanceName	Name of the substance identified in the analysis				
8	DrugReports	County total count of the indicated substance.				
9	TotalDrugReportsCounty	County total count of all substances identified.				
10	TotalDrugReportsState	State total count of all substances identified.				

Read in the data

Read xlsx data

In [2]:
```
#set up strings with directory/file names
file_dir = "C:/Users/hamilton.w/Documents/MCM_data/"
data_name = "MCM_NFLIS_data.xlsx"

#read the xlsx, and convert to a pandas dataframe
#some versions of pandas requires the optional argument ``sheet_name'' instead of ``sheetname''
df = pd.read_excel("{}{}".format(file_dir,data_name), sheetname = "Data")
```

Initial data summary

Initial summary of what's in the spreadsheet

In [3]:
```
#print the column names
print(df.columns)

#summarize the entries in each column
for label in df.columns:
    print(label)
    print(df[label].unique())
```

```
Index(['YYYY', 'State', 'COUNTY', 'FIPS_State', 'FIPS_County', 'FIPS_Combined',
       'SubstanceName', 'DrugReports', 'TotalDrugReportsCounty',
       'TotalDrugReportsState'],
      dtype='object')
YYYY
[2010 2011 2012 2013 2014 2015 2016 2017]
State
['VA' 'OH' 'PA' 'KY' 'WV']
COUNTY
['ACCOMACK' 'ADAMS' 'ALEXANDRIA CITY' 'ALLEGHENY' 'ALLEN' 'AMELIA'
 'ARLINGTON' 'ARMSTRONG' 'ASHLAND' 'ASHTABULA' 'ATHENS' 'AUGLAIZE'
 'BARBOUR' 'BARREN' 'BATH' 'BEDFORD' 'BELL' 'BERKELEY' 'BERKS' 'BLAND'
 'BOONE' 'BRACKEN' 'BRAXTON' 'BREATHITT' 'BRECKINRIDGE' 'BROWN'
 'BUCKINGHAM' 'BUCKS' 'BULLITT' 'BUTLER' 'CABELL' 'CALHOUN' 'CAMBRIA'
 'CAMERON' 'CARBON' 'CAROLINE' 'CARROLL' 'CARTER' 'CASEY'
 'CHARLOTTESVILLE CITY' 'CHESAPEAKE CITY' 'CHESTER' 'CHESTERFIELD'
 'CHRISTIAN' 'CLARION' 'CLEARFIELD' 'CLERMONT' 'CLINTON'
```

Sorted data, by drug reports

Summary of the entries, sorted by drug reports

In [6]:
```
#sort all entries by number of reported cases in each county
sort_order_states = np.argsort(df.iloc[:,-3].values)
df.iloc[sort_order_states,[0,1,2,6,7,8,9]][:100]
```

Out[6]:

	YYYY	State	COUNTY	SubstanceName	DrugReports	TotalDrugReportsCounty	TotalDrugReportsState
0	2010	VA	ACCOMACK	Propoxyphene	1	84	41462
9056	2013	KY	CARTER	Propoxyphene	1	174	26820
9054	2013	KY	CARTER	Fentanyl	1	174	26820
9053	2013	VA	CARROLL	Fentanyl	1	209	47694
9051	2013	OH	CARROLL	Codeine	1	78	93747
9050	2013	VA	CAROLINE	Hydromorphone	1	205	47694
9049	2013	WV	KANAWHA	Fentanyl	1	1013	9062
9048	2013	WV	KANAWHA	Dextropropoxyphene	1	1013	9062
21079	2017	OH	DELAWARE	Cyclopropyl fentanyl	1	678	119349
9044	2013	PA	JEFFERSON	Tramadol	1	220	72096
9043	2013	PA	JEFFERSON	Hydrocodone	1	220	72096

Sorted data, by drug reports

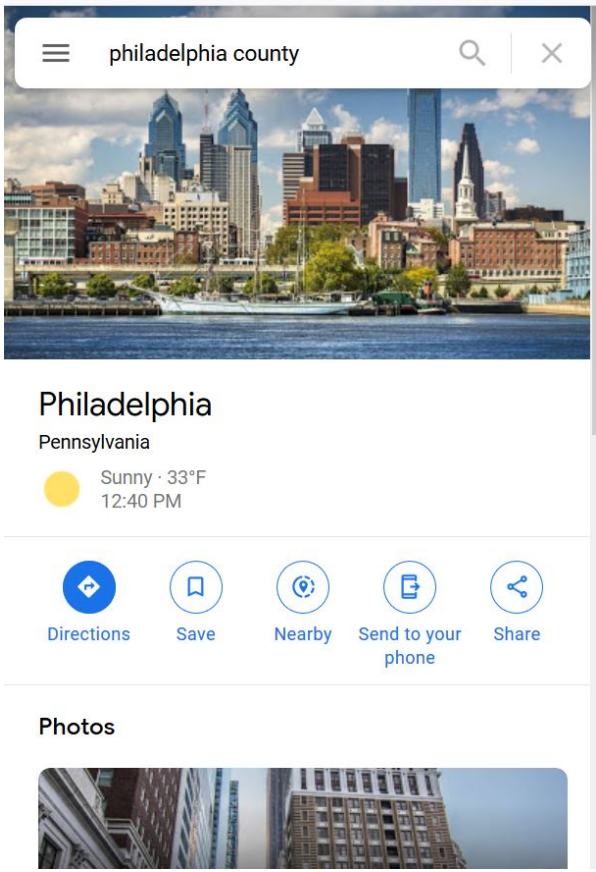
Same summary, just order reversed

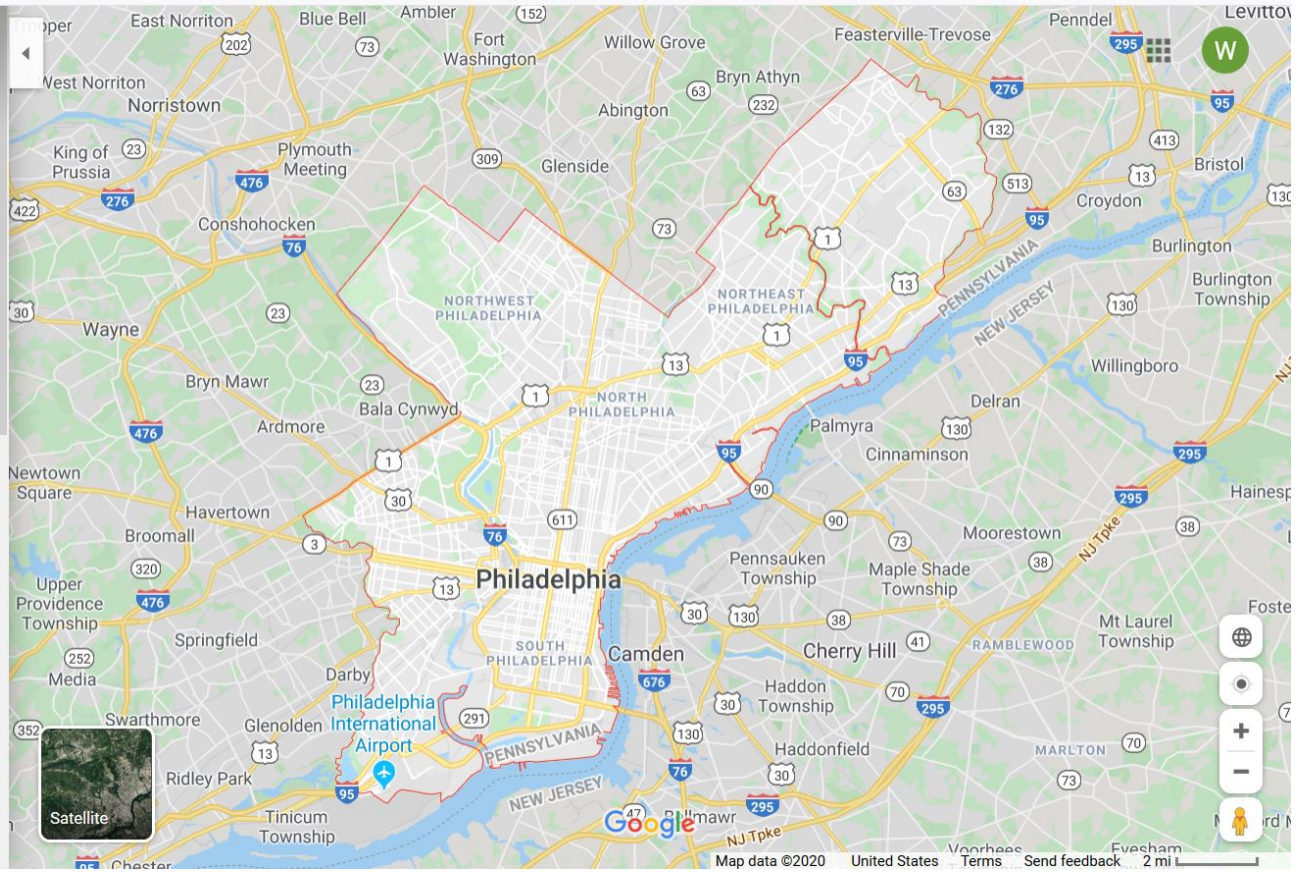
In [7]:
```
#reverse sort order
sort_order_states = sort_order_states[::-1]
df.iloc[sort_order_states,[0,1,2,6,7,8,9]][:50]
```

Out[7]:

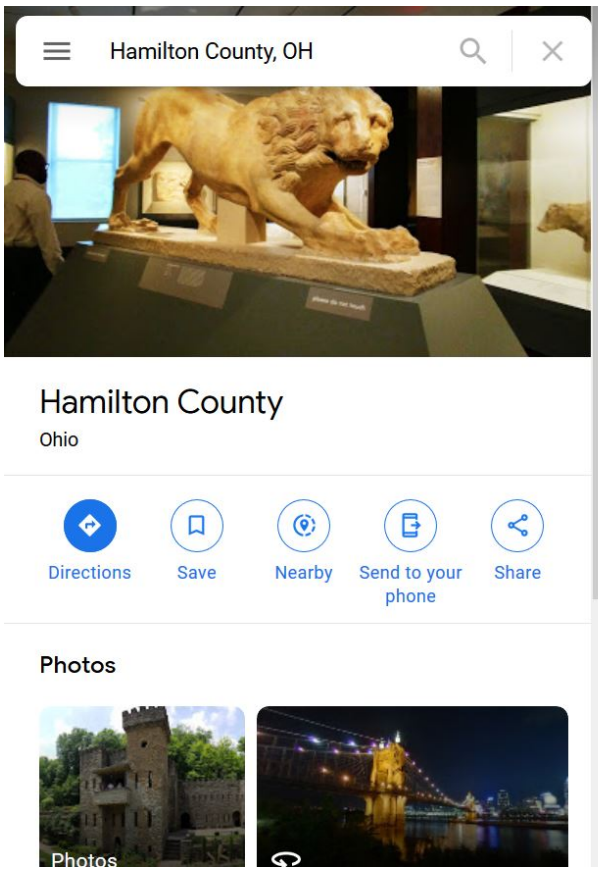
	YYYY	State	COUNTY	SubstanceName	DrugReports	TotalDrugReportsCounty	TotalDrugReportsState
20195	2016	PA	PHILADELPHIA	Heroin	5075	21194	72376
16954	2015	PA	PHILADELPHIA	Heroin	4985	21387	75351
23779	2017	PA	PHILADELPHIA	Heroin	4664	19614	68751
18472	2016	OH	HAMILTON	Heroin	4525	17376	115276
15907	2015	OH	HAMILTON	Heroin	4167	14406	109150
22077	2017	OH	HAMILTON	Heroin	3970	22074	119349
2981	2010	PA	PHILADELPHIA	Heroin	3910	33513	89981
12169	2014	PA	ALLEGHENY	Heroin	3774	9286	77318
12831	2014	OH	HAMILTON	Heroin	3686	12208	101423
14687	2015	PA	ALLEGHENY	Heroin	3674	8938	75351
8482	2012	PA	PHILADELPHIA	Heroin	3659	25279	78577

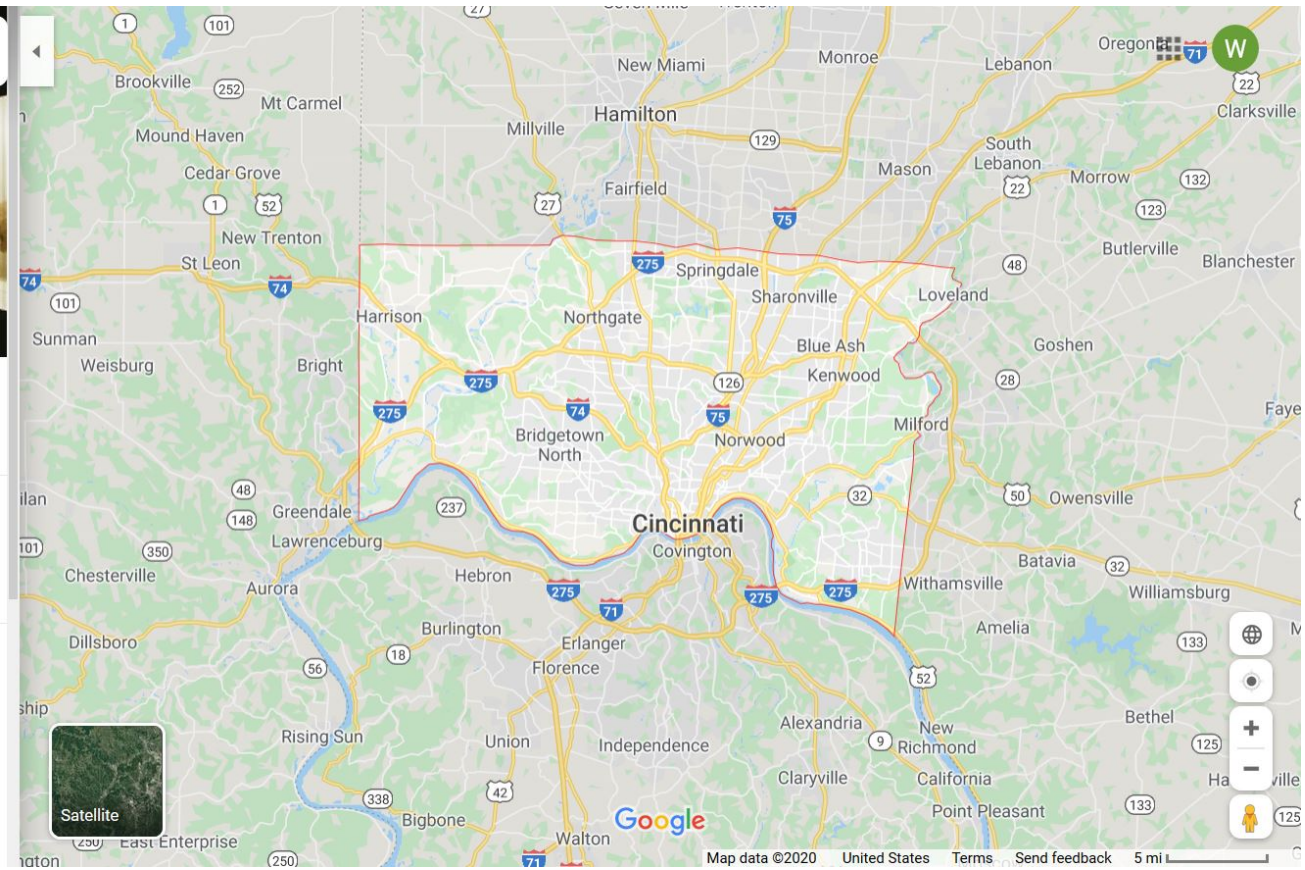
Where are these counties?



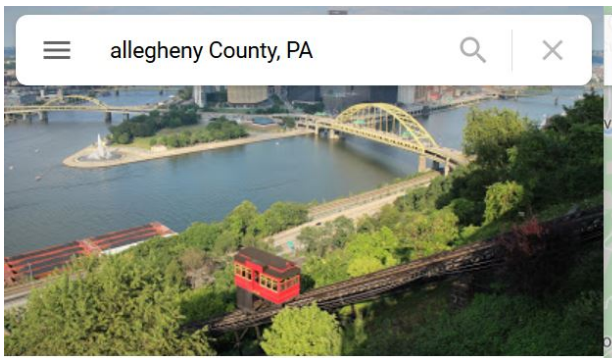


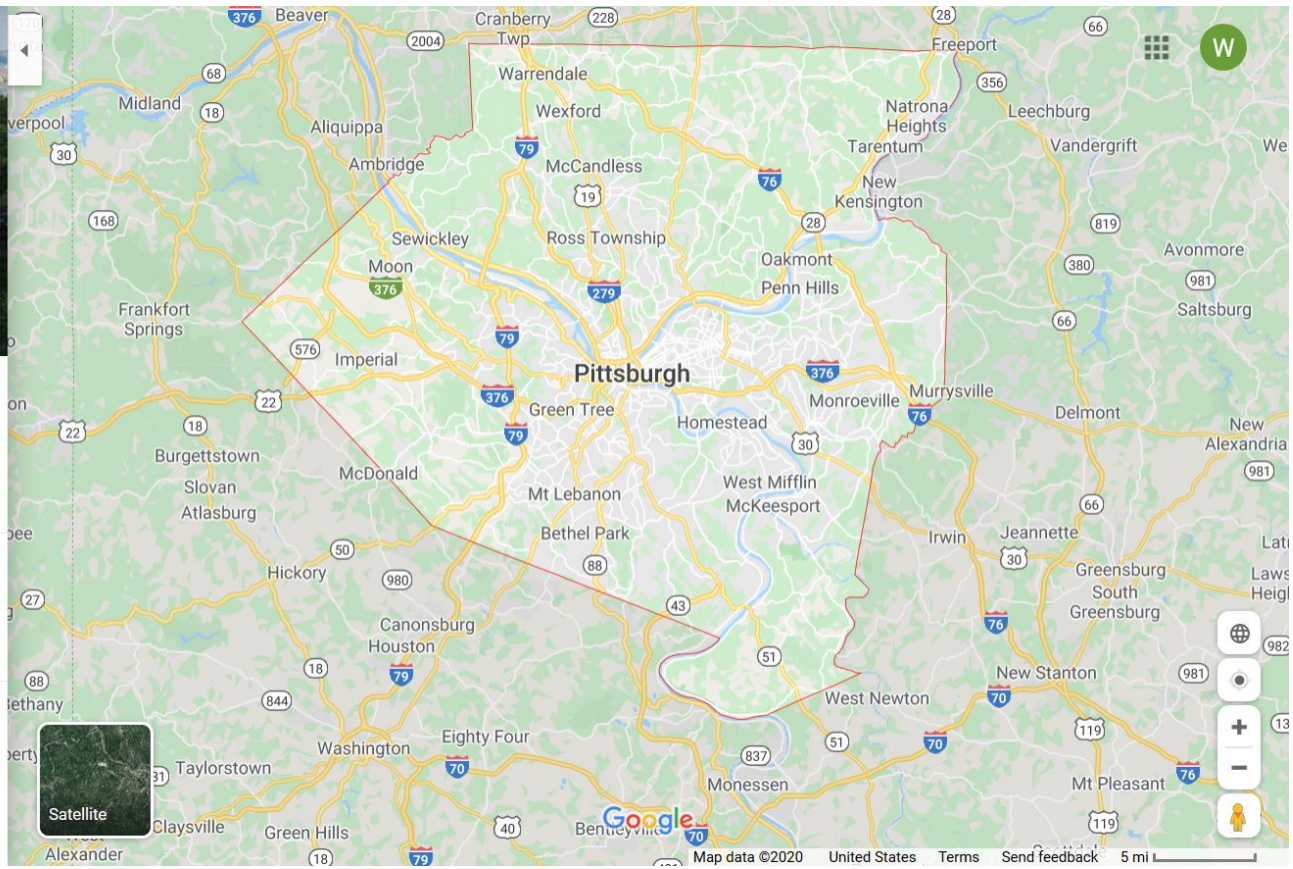
Where are these counties?



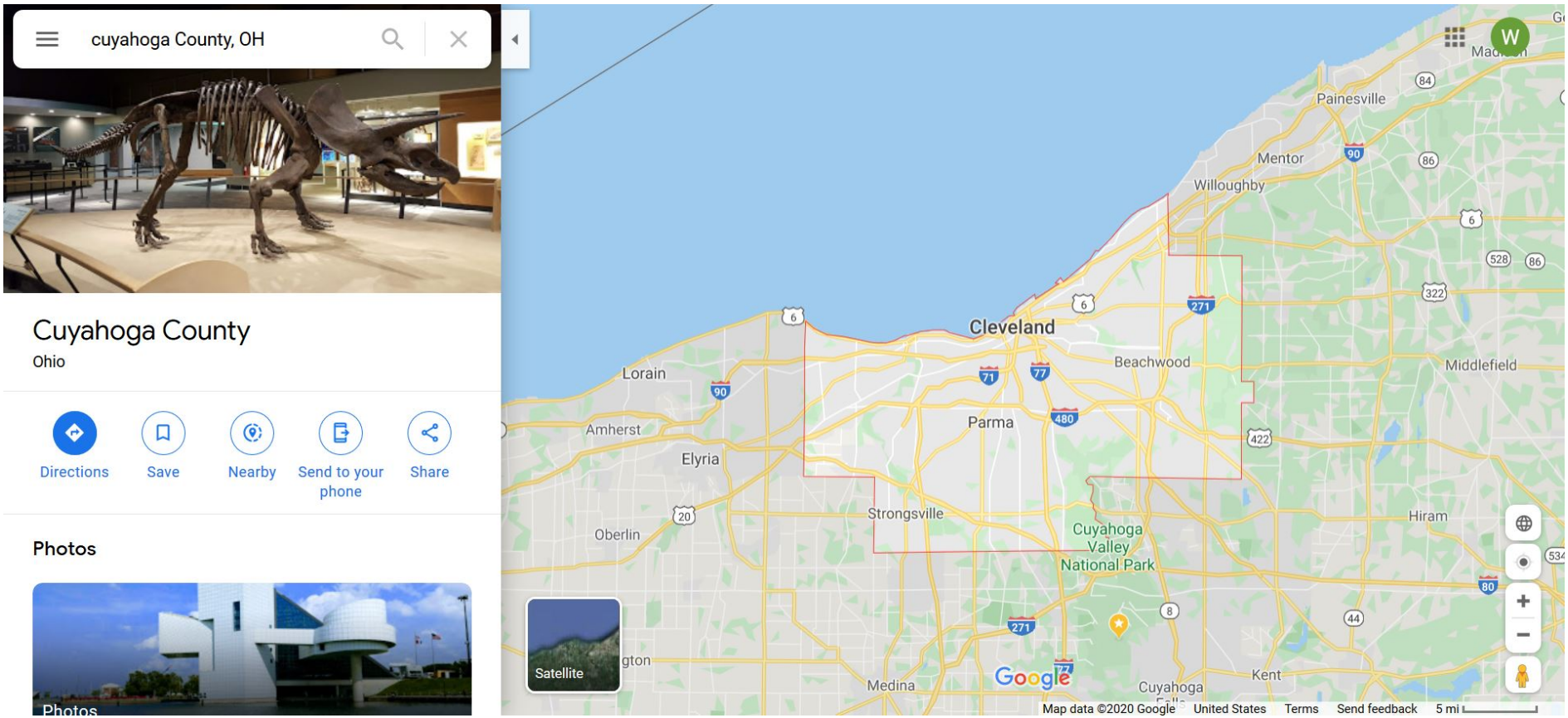


Where are these counties?

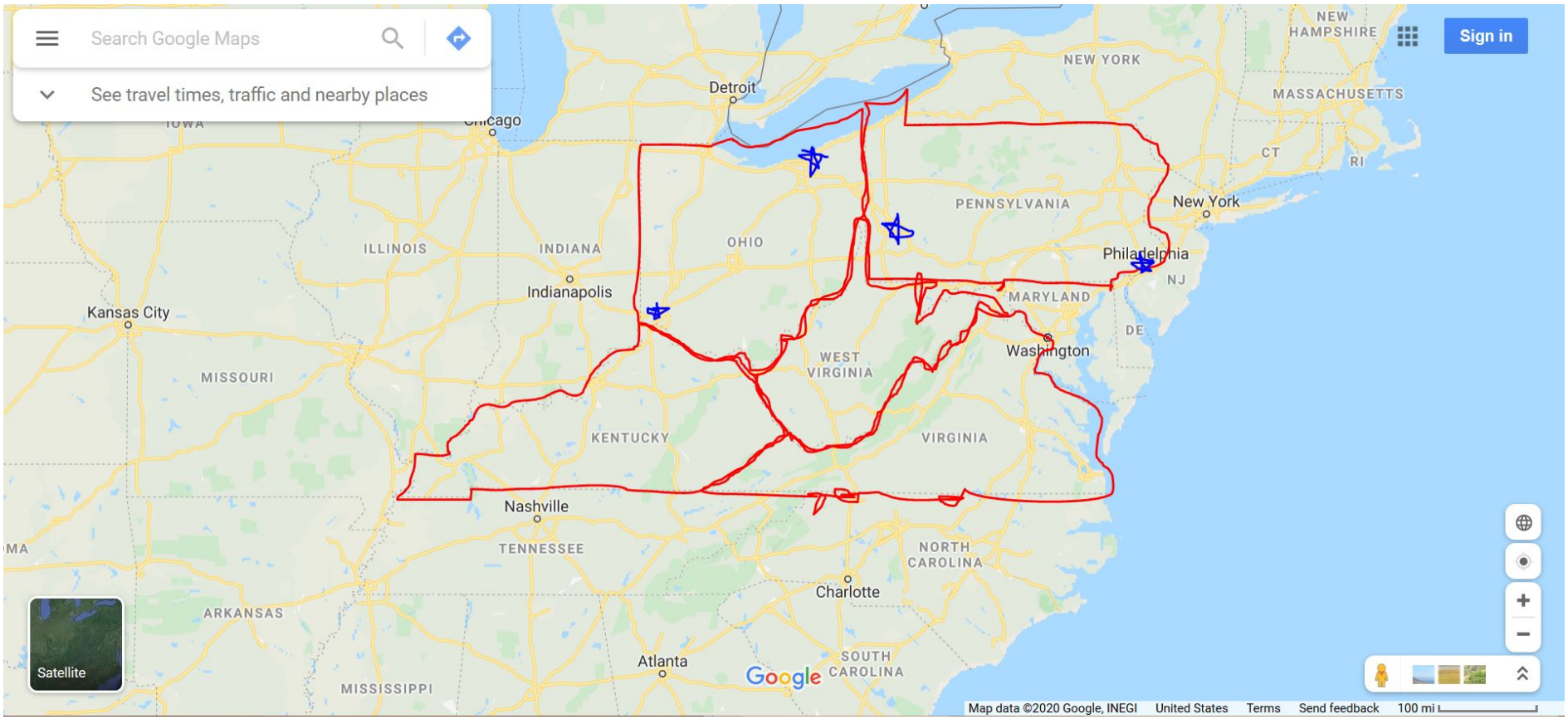




Where are these counties?



Where are these counties?



Isolate the state drug reports

Isolate a dataframe with the state drug reports

In [9]:
```
#sort state drug reports
df_state_totals = df.iloc[:,[0,1,3,9]]
df_state_totals = df_state_totals.drop_duplicates()
df_state_totals.sort_values("TotalDrugReportsState")[::-1]
```

Out[9]:

	YYYY	State	FIPS_State	TotalDrugReportsState
19745	2017	OH	39	119349
16494	2016	OH	39	115276
13647	2015	OH	39	109150
10843	2014	OH	39	101423
7882	2013	OH	39	93747
2	2010	PA	42	89981
2381	2011	PA	42	86793
4911	2012	OH	39	85415
4817	2012	PA	42	78577
10845	2014	PA	42	77318
13572	2015	PA	42	75351

Isolate the county drug reports

Isolate a dataframe with the county drug reports

In [10]:
```
#sort county drug reports
df_county_totals = df.iloc[:,[0,1,2,4,8,9]]
df_county_totals = df_county_totals.drop_duplicates()
df_county_totals.sort_values("TotalDrugReportsCounty")[::-1]
```

Out[10]:

	YYYY	State	COUNTY	FIPS_County	TotalDrugReportsCounty	TotalDrugReportsState
1691	2010	PA	PHILADELPHIA	101	33513	89981
4012	2011	PA	PHILADELPHIA	101	26969	86793
7138	2012	PA	PHILADELPHIA	101	25279	78577
20655	2017	OH	CUYAHOGA	35	22133	119349
21437	2017	OH	HAMILTON	61	22074	119349
10043	2013	PA	PHILADELPHIA	101	21761	72096
15777	2015	PA	PHILADELPHIA	101	21387	75351
19037	2016	PA	PHILADELPHIA	101	21194	72376
22456	2017	PA	PHILADELPHIA	101	19614	68751
5787	2012	OH	CUYAHOGA	35	17801	85415
12767	2014	PA	PHILADELPHIA	101	17597	77318

What information is there about the opioid epidemic?

https://www.hhs.gov/opioids/about-the-epidemic/index.html
and then
https://www.cdc.gov/drugoverdose/epidemic/index.html

What information is there about the opioid epidemic?

Understanding the Epidemic

Drug overdose deaths continue to increase in the United States.

- From 1999 to 2017, more than 700,000 people have died from a drug overdose.
- Around 68% of the more than 70,200 drug overdose deaths in 2017 involved an opioid.
- In 2017, the number of overdose deaths involving opioids (including prescription opioids and illegal opioids like heroin and illicitly manufactured fentanyl) was 6 times higher than in 1999.
- On average, 130 Americans die every day from an opioid overdose.[1]

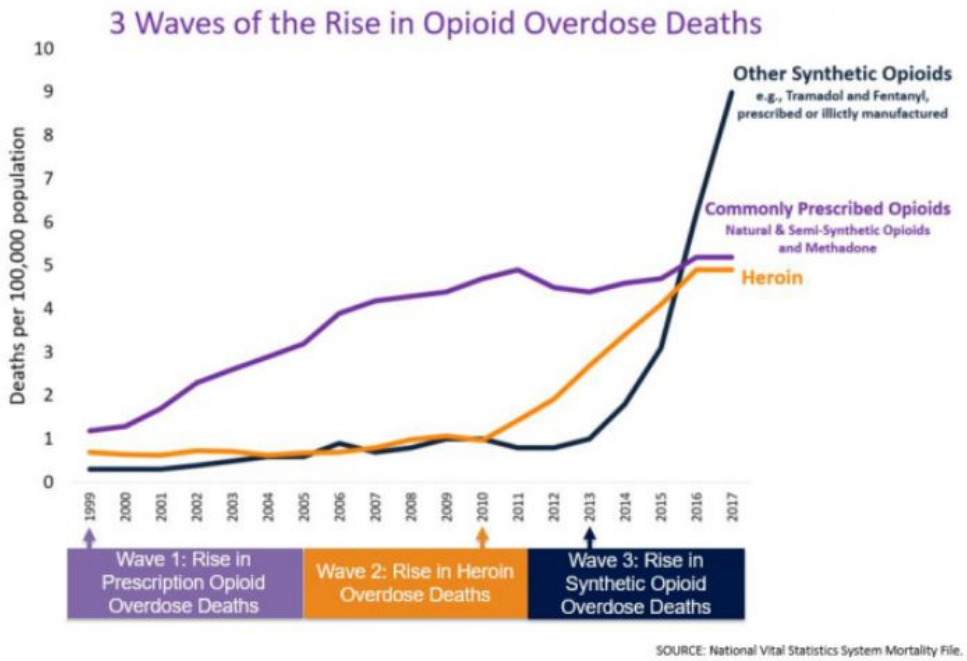
On This Page

What information is there about the opioid epidemic?

The Three Waves of Opioid Overdose Deaths

From 1999-2017, almost 400,000 people died from an overdose involving any opioid, including prescription and illicit opioids.[2]

This rise in opioid overdose deaths can be outlined in three distinct waves.



1. The first wave began with increased prescribing of opioids in the 1990s [3], with overdose deaths involving prescription opioids (natural and semi-synthetic opioids and methadone) increasing since at least 1999.
2. The second wave began in 2010, with rapid increases in overdose deaths involving heroin.
3. The third wave began in 2013, with significant increases in overdose deaths involving synthetic opioids – particularly those involving illicitly-manufactured fentanyl (IMF). The IMF market continues to change, and IMF can be found in combination with heroin, counterfeit pills, and cocaine. [2,4]

Isolate the heroin and fentanyl data

Isolate the heroin and fentanyl data

In [11]:
```
#get heroin data
df_heroin = df[df["SubstanceName"] == "Heroin"]
df_heroin = df_heroin.iloc[:,[0,1,3,7]]

#get fentanyl data
df_fentanyl = df[df["SubstanceName"] == "Fentanyl"]
df_fentanyl = df_fentanyl.iloc[:,[0,1,3,7]]
```

Get the heroin and fentanyl total reports by year

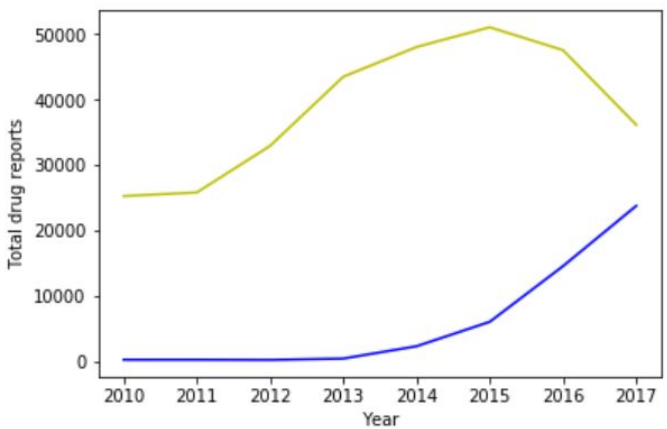
Get totals by year

In [13]:
```
#get totals by year
years = df["YYYY"].unique()
heroin_totals = np.zeros(len(years))
fentanyl_totals = np.zeros(len(years))

for i in range(len(years)):
    y = years[i]
    heroin_totals[i] = sum(df_heroin[df_heroin["YYYY"]==y].iloc[:,-1].values)

    fentanyl_totals[i] = sum(df_fentanyl[df_fentanyl["YYYY"]==y].iloc[:,-1].values)
```

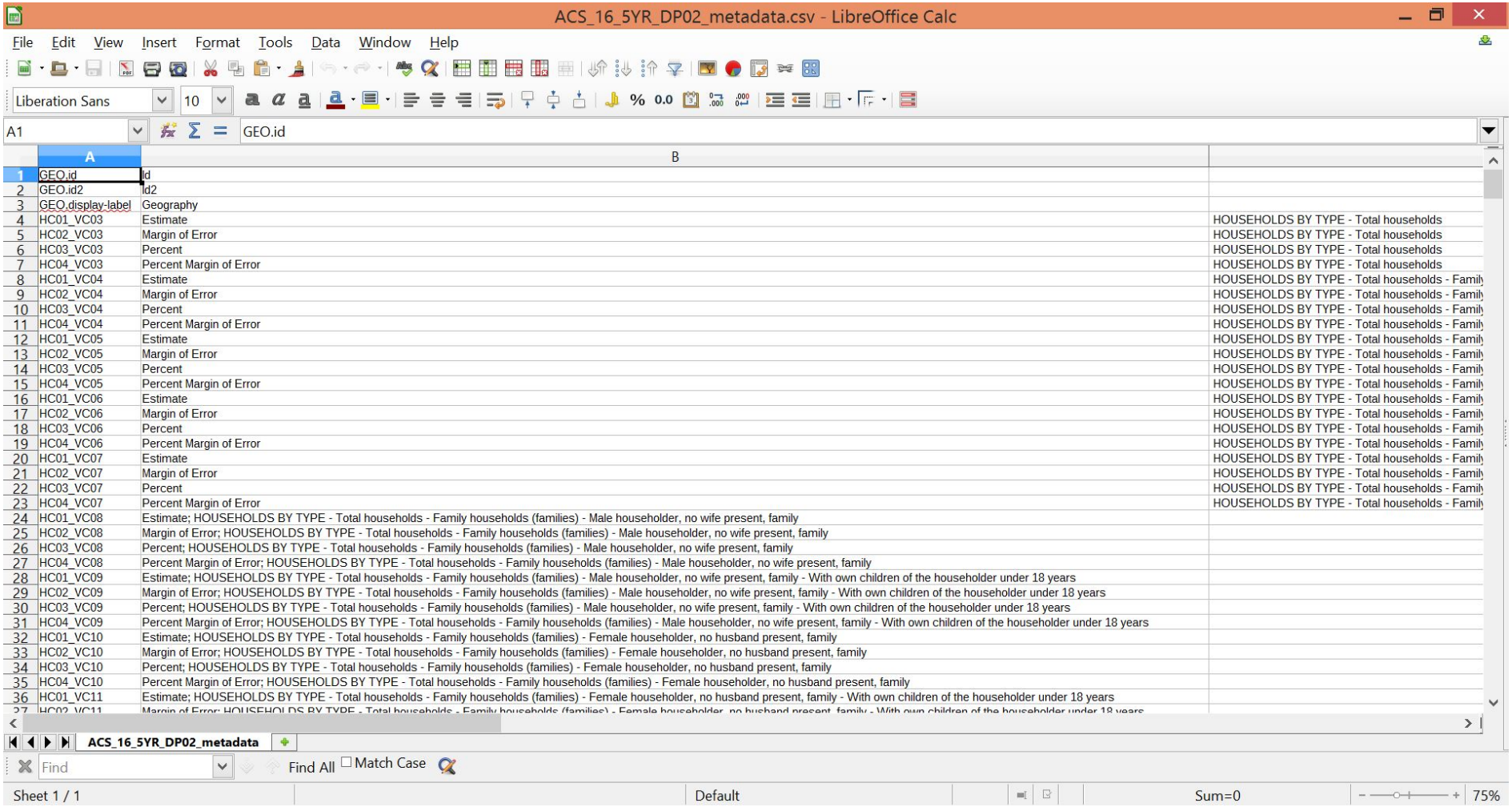
Plot the drug report rates

Plot the drug report rates

In [16]:
```
plt.plot(years,heroin_totals, c = "y")
plt.plot(years,fentanyl_totals, c = "b")

plt.ylabel("Total drug reports")
plt.xlabel("Year")
plt.show()
```



Peruse the metadata



	A	B	
1	GEO.id	id	
2	GEO.id2	id2	
3	GEO.display-label	Geography	
4	HC01_VC03	Estimate	HOUSEHOLDS BY TYPE - Total households
5	HC02_VC03	Margin of Error	HOUSEHOLDS BY TYPE - Total households
6	HC03_VC03	Percent	HOUSEHOLDS BY TYPE - Total households
7	HC04_VC03	Percent Margin of Error	HOUSEHOLDS BY TYPE - Total households
8	HC01_VC04	Estimate	HOUSEHOLDS BY TYPE - Total households - Family
9	HC02_VC04	Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
10	HC03_VC04	Percent	HOUSEHOLDS BY TYPE - Total households - Family
11	HC04_VC04	Percent Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
12	HC01_VC05	Estimate	HOUSEHOLDS BY TYPE - Total households - Family
13	HC02_VC05	Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
14	HC03_VC05	Percent	HOUSEHOLDS BY TYPE - Total households - Family
15	HC04_VC05	Percent Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
16	HC01_VC06	Estimate	HOUSEHOLDS BY TYPE - Total households - Family
17	HC02_VC06	Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
18	HC03_VC06	Percent	HOUSEHOLDS BY TYPE - Total households - Family
19	HC04_VC06	Percent Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
20	HC01_VC07	Estimate	HOUSEHOLDS BY TYPE - Total households - Family
21	HC02_VC07	Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
22	HC03_VC07	Percent	HOUSEHOLDS BY TYPE - Total households - Family
23	HC04_VC07	Percent Margin of Error	HOUSEHOLDS BY TYPE - Total households - Family
24	HC01_VC08	Estimate; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family	
25	HC02_VC08	Margin of Error; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family	
26	HC03_VC08	Percent; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family	
27	HC04_VC08	Percent Margin of Error; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family	
28	HC01_VC09	Estimate; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family - With own children of the householder under 18 years	
29	HC02_VC09	Margin of Error; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family - With own children of the householder under 18 years	
30	HC03_VC09	Percent; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family - With own children of the householder under 18 years	
31	HC04_VC09	Percent Margin of Error; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Male householder, no wife present, family - With own children of the householder under 18 years	
32	HC01_VC10	Estimate; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Female householder, no husband present, family	
33	HC02_VC10	Margin of Error; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Female householder, no husband present, family	
34	HC03_VC10	Percent; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Female householder, no husband present, family	
35	HC04_VC10	Percent Margin of Error; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Female householder, no husband present, family	
36	HC01_VC11	Estimate; HOUSEHOLDS BY TYPE - Total households - Family households (families) - Female householder, no husband present, family - With own children of the householder under 18 years	

Read in the metadata

Read in the meta data for 2016

In [19]:
```
#read meta data
se_data_name = "ACS_16_5YR_DP02_with_ann.csv"
se_metadata_name = "ACS_16_5YR_DP02_metadata.csv"

df_se = pd.read_csv("{}{}".format(file_dir,se_data_name))
df_se_meta = pd.read_csv("{}{}".format(file_dir,se_metadata_name))
```

Metadata summary

See what's in each csv

In [21]:
```
df_se.head()
```

Out[21]:

	GEO.id	GEO.id2	GEO.display-label	HC01_VC03	HC02_VC03	HC03_VC03	HC04_VC03	HC01_VC04	HC02_VC04	HC03_V
0	Id	Id2	Geography	Estimate; HOUSEHOLDS BY TYPE - Total households	Margin of Error; HOUSEHOLDS BY TYPE - Total ho...	Percent; HOUSEHOLDS BY TYPE - Total households	Percent Margin of Error; HOUSEHOLDS BY TYPE - ...	Estimate; HOUSEHOLDS BY TYPE - Total household...	Margin of Error; HOUSEHOLDS BY TYPE - Total ho...	Percent HOUSE BY TYP Total househ
1	0500000US21001	21001	Adair County, Kentucky	7126	253	7126	(X)	4934	240	69.2
2	0500000US21003	21003	Allen County, Kentucky	7705	255	7705	(X)	5561	267	72.2

Metadata summary

In [22]:
```
df_se_meta.head()
```

Out[22]:

	GEO.id	Id
0	GEO.id2	Id2
1	GEO.display-label	Geography
2	HC01_VC03	Estimate; HOUSEHOLDS BY TYPE - Total households
3	HC02_VC03	Margin of Error; HOUSEHOLDS BY TYPE - Total ho...
4	HC03_VC03	Percent; HOUSEHOLDS BY TYPE - Total households

Isolate the dictionary for numbers of households

Isolate the dictionary for numbers of households

In [ ]:
```
#isolate the house dictionary
df_se.iloc[:,[1,3]]

#check that the sizes match up
print(len(df_se["GEO.id2"].unique()))
print(len(df["FIPS_Combined"].unique()))
```

Where do these lists disagree?

Sizes don't match up... where?

In [24]:
```
#convert the geo_ids to ints
orig_data_locs = df["FIPS_Combined"].unique()
unique_meta_locs = df_se["GEO.id2"].iloc[1:].unique().astype(int)

#figure out which counties are different
print(list(set(orig_data_locs) - set(unique_meta_locs)))

#double check
print(sum(df_se["GEO.id2"]=="51515"))

#which city?
print(df[df["FIPS_Combined"]==51515])
```

```
[51515]
0
       YYYY State        COUNTY  FIPS_State  FIPS_County  FIPS_Combined  \
220    2010    VA  BEDFORD CITY          51          515          51515
528    2010    VA  BEDFORD CITY          51          515          51515
7913   2013    VA  BEDFORD CITY          51          515          51515
20465  2017    VA  BEDFORD CITY          51          515          51515
```

Normalizing drug report counts by households, set up

Let's normalize our original drug reports by numbers of households

In [25]:
```
#recover state total households
state_FIPS_dict = {}
state_total_households = {}

#set up what we need for the state households totals
for s in df["State"].unique():
    #get the state FIPS
    state_FIPS = df[df["State"]==s]["FIPS_State"].iloc[0]

    state_FIPS_dict[s] = state_FIPS
    state_total_households[state_FIPS] = 0.

#actually compute the estimated state total households
for i in range(1,len(df_se)):
    geo_id = df_se["GEO.id2"].iloc[i]
    num_households = df_se["HC01_VC03"].iloc[i]

    state_total_households[int(geo_id[:2])] += int(num_households)

print(state_FIPS_dict)
print(state_total_households)
```

```
{'OH': 39, 'WV': 54, 'VA': 51, 'PA': 42, 'KY': 21}
{42: 4961929.0, 51: 3090178.0, 21: 1718217.0, 54: 739397.0, 39: 4601449.0}
```

Copy over the dataframe, apply transforms

Make a copy of the original dataframe

In [ ]:
```
#copy the df
normed_df = df.copy()
```

Convert the state totals in the copied dataframe

In [ ]:
```
#convert the state totals
for s in df["State"].unique():
    s_FIPS = state_FIPS_dict[s]
    transform_func = lambda x: x/state_total_households[s_FIPS]

    normed_df.loc[normed_df["State"]==s,"TotalDrugReportsState"] = normed_df[normed_df["State"]==s]["TotalDrugReportsSta
```

Convert the county totals

In [ ]:
```
#convert the county totals
for c in unique_meta_locs:
    county_total = df_se[df_se["GEO.id2"]==str(c)]["HC01_VC03"].values
    transform_func = lambda x: x/float(county_total)

    normed_df.loc[normed_df["FIPS_Combined"]==c,"TotalDrugReportsCounty"] = normed_df[normed_df["FIPS_Combined"]==c]["To
```

Remove the "bad" data

Cut out the county without the relevant data

In [29]:
```
#remove 51515
#get indices of 51515 entries
drop_indices = np.arange(len(normed_df))[normed_df["FIPS_Combined"]==51515]

normed_df = normed_df.drop(drop_indices)
```

Similar summary from before, sorting by normalized state drug reports

Repeat the same analysis as before, with the normalized data

In [31]:
```
#sort state drug reports
df_state_totals = normed_df.iloc[:,[0,1,3,9]]
df_state_totals = df_state_totals.drop_duplicates()
df_state_totals.sort_values("TotalDrugReportsState")[::-1]
```

Out[31]:

	YYYY	State	FIPS_State	TotalDrugReportsState
19745	2017	OH	39	0.025937
16494	2016	OH	39	0.025052
13647	2015	OH	39	0.023721
10843	2014	OH	39	0.022042
7882	2013	OH	39	0.020373
4911	2012	OH	39	0.018563
2	2010	PA	42	0.018134
2381	2011	PA	42	0.017492
5	2010	KY	21	0.017220
19744	2017	KY	21	0.016802

Similar summary from before, sorting by normalized county drug reports

In [33]:
```
#sort county drug reports
df_county_totals = normed_df.iloc[:,[0,1,2,8]]
df_county_totals = df_county_totals.drop_duplicates()
df_county_totals.sort_values("TotalDrugReportsCounty")[::-1]
```

Out[33]:

	YYYY	State	COUNTY	TotalDrugReportsCounty
22478	2017	VA	NELSON	0.073220
16782	2016	KY	BELL	0.072691
10947	2014	KY	BELL	0.067316
13955	2015	KY	BELL	0.066952
19774	2017	KY	BELL	0.066132
21437	2017	OH	HAMILTON	0.065827
19227	2016	KY	PERRY	0.065475
4010	2011	KY	PERRY	0.065113
15283	2015	KY	MARTIN	0.064726
6026	2012	KY	HARLAN	0.064398
4007	2011	KY	OWSLEY	0.064247
21672	2017	OH	LAKE	0.063150

What are the new top counties' populations?

What are the households and populations for the new top counties?

In [34]:
```
#manually write out the pairs we want to look at
state_county_pairs = [("VA","NELSON"),("KY","BELL"),("OH","HAMILTON"),("KY","PERRY")]

#for each state and county pair
for sc_pair in state_county_pairs:
    #save the local dataframe
    temp_df = df[(df["State"]==sc_pair[0])&(df["COUNTY"]==sc_pair[1])]

    #[0,1,2,8,9] are the year, state, county, total county reports, total state reports
    print(temp_df.iloc[:,[0,1,2,8,9]].drop_duplicates())

    #print the 2016 housing data for each state, county pair
    temp_FIPS = temp_df["FIPS_Combined"].iloc[0]
    print("The number of households is:")
    print(float(df_se[df_se["GEO.id2"] == str(temp_FIPS)]["HC01_VC03"]))
```

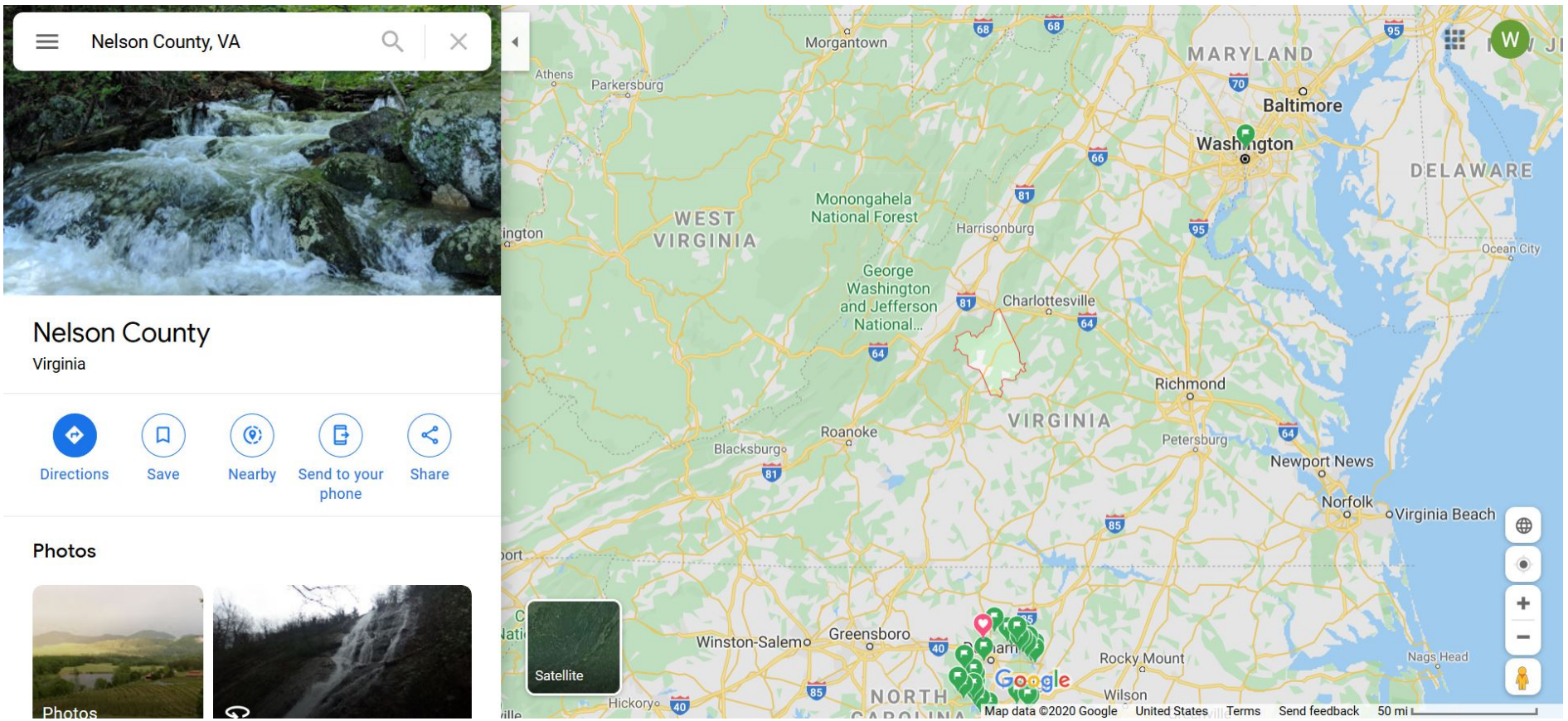
What are the new top counties' populations?

```
       YYYY State  COUNTY  TotalDrugReportsCounty  TotalDrugReportsState
1670   2010    VA  NELSON                      94                  41462
4115   2011    VA  NELSON                      73                  28969
7573   2012    VA  NELSON                      61                  32251
9849   2013    VA  NELSON                     104                  47694
12944  2014    VA  NELSON                      91                  32265
15592  2015    VA  NELSON                      38                  27819
18677  2016    VA  NELSON                     161                  33539
22478  2017    VA  NELSON                     467                  36994
The number of households is:
6378.0
       YYYY State COUNTY  TotalDrugReportsCounty  TotalDrugReportsState
25     2010    KY   BELL                     387                  29588
2486   2011    KY   BELL                     204                  28285
5009   2012    KY   BELL                     257                  27502
8005   2013    KY   BELL                     259                  26820
10947  2014    KY   BELL                     739                  27077
13955  2015    KY   BELL                     735                  25811
16782  2016    KY   BELL                     798                  26530
19774  2017    KY   BELL                     726                  28870
The number of households is:
10978.0
```

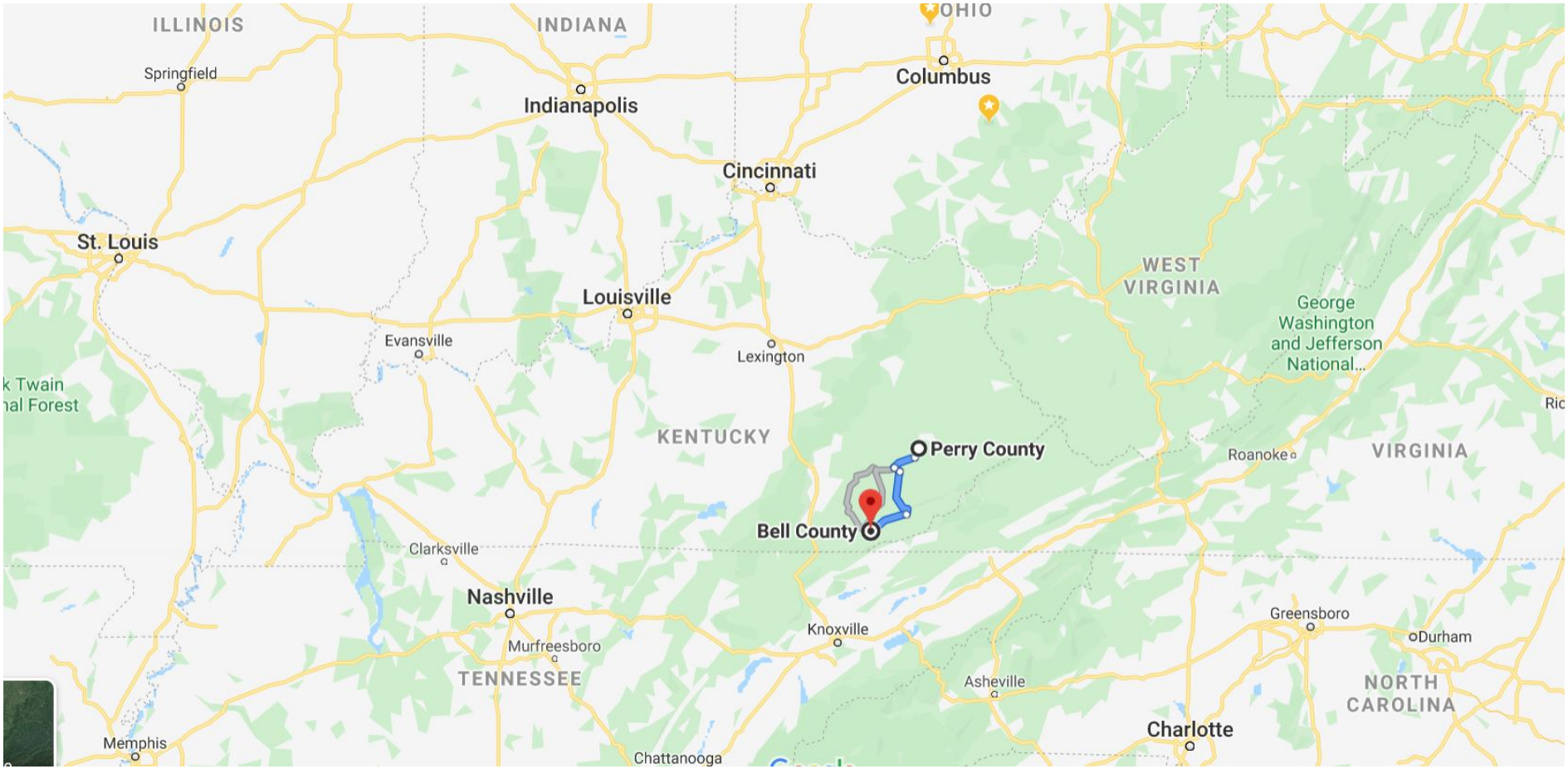
What are the new top counties' populations?

```
       YYYY State    COUNTY  TotalDrugReportsCounty  TotalDrugReportsState
845    2010    OH  HAMILTON                   13780                  70999
3194   2011    OH  HAMILTON                   10837                  71282
6239   2012    OH  HAMILTON                   10453                  85415
8722   2013    OH  HAMILTON                   11063                  93747
11632  2014    OH  HAMILTON                   12208                 101423
15041  2015    OH  HAMILTON                   14406                 109150
17670  2016    OH  HAMILTON                   17376                 115276
21437  2017    OH  HAMILTON                   22074                 119349
The number of households is:
335334.0
       YYYY State COUNTY  TotalDrugReportsCounty  TotalDrugReportsState
1872   2010    KY  PERRY                     599                  29588
4010   2011    KY  PERRY                     721                  28285
7278   2012    KY  PERRY                     614                  27502
10244  2013    KY  PERRY                     389                  26820
12954  2014    KY  PERRY                     362                  27077
15775  2015    KY  PERRY                     420                  25811
19227  2016    KY  PERRY                     725                  26530
22453  2017    KY  PERRY                     445                  28870
The number of households is:
11073.0
```

Where are these counties?



Where are these counties?



What's next?

- So much else!
- We normalized by 2016 household data, what if we normalize by the other household data?
- So much socio-economic data is provided; could build a multivariate model to incorporate said data?
- Geographic data could help: can use the google API to download long./lat. data for each county, and build spatial models?
- Etc.

Questions?

Questions?