# Project 6 CS 111 Law: Predictive Policing in Chicago

**Release Date:** Tuesday, November 26, 2019.
**Release Version:** 1.0

**Due: 11:59 pm** Friday, December 6, 2019

*Learning Objectives:* Working with dictionaries and DataFrames in Python, and working with real City of Chicago data.

**Submit your assignment on Blackboard in a single .zip file called [your NetID]Project6.zip**

In this homework, you will work with real data from the City of Chicago to develop a map that shows the prevalence of crime in the 77 Community areas of Chicago.

## Overview

We will develop a visualization of some Chicago crime data by community area.

You will work with some (large, real) datasets we have downloaded for you from the City of Chicago Data Portal. They are:

- `Crimes_-_2012_to_present.csv` and
- `Boundaries_Community.geojson`

`Crimes_-_2012_to_present.csv` contains just what it says: data on crime in Chicago from 2001 to the present. In particular, it contains a column 'Community Area,' which correlates information in a row with one of 77 communities in Chicago. You will use pandas to convert that csv file to a DataFrame. That csv file is so big that it may take 10-45 seconds for pandas to import it.

`Boundaries_Community.geojson` contains information that defines the boundaries of 77 communities. The geopandas module knows how to read it into a specialized form of dataframe for geographic information. Once you create that data structure, it will have a column called area_num_1 that gives the community number in the range of 1 to 77. (To view the file, go to https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Community-Areas-current-/cauq-8yn6 and click on the "Table" icon in the upper right (just above 'Embed').) You will use it to map information in `Crimes_-_2012_to_present.csv` onto a map of the communities.

## Installing geopandas module

You will convert `Boundaries_Community.geojson` to a DataFrame using a new module: geopandas.

geopandas is *not* included with the out-of-the-box setup that Anaconda gives us. There are two ways to install the geopandas module.

**Please install geopandas before or over the Thanksgiving break**, so we can give you help if you run into difficulties with the installation.

### Method 1: Terminal

A perhaps more reliable method is to use the terminal. Open a terminal window. At the prompt, enter:

```
conda install geopandas
```

You will will see a bunch of informational and perhaps warning messages, and then will get a line that says, "Proceed ([y]/n)? " Reply with y.

After that, you'll need to do the same thing for "mapclassify" and "descartes".

### Method 2: Anaconda Navigator Graphical User Interface

The other way you can install it is using Anaconda Navigator. There are a few reports of this method either not working at all, or working only after one does the following, then restarts Navigator.

Inside Anaconda Navigator, choose "Environments" from the left-most column.

Next click on "Update index...." on the right-hand side towards the top.

Then be sure that over at the upper right you have "All" or "Uninstalled" selected (*not* "Installed") and scorll down to geopandas, and check that check box, scroll to check also descartes and mapclassify, and then click apply.

### If you get stuck on install

If things aren't working, uninstall anaconda and reinstall, and then add in geopandas, descartes, and mapclassify.

We can help you do this in office hours *if you started early enough to get to office hours for help*.

## Using geopandas module

Once geopandas is installed, to use it, import it, and then create 1 DataFrame using geopandas.read(). The code is:

```
import geopandas as gpd

communities = gpd.read_file('Boundaries_Community.geojson')
```

# The data files

We have downloaded the two data files for you from the City of Chicago Data Portal at https://data.cityofchicago.org/. (We downloaded them for you so everybody starts with the same file with the same ASCII CSV encodings.)

*We have posted them online for your to download at* https://uofi.box.com/v/CS111LawFall19ChicagoDataSets. Be aware that one of them is very large.

1. Crimes_-_2012_to_present.csv: This has an entry for every crime reported to the Chicago police between Jan. 1, 2012 and about a week or two ago. Important column headers for you:
    - 'Case Number': A unique ID for each case
    - 'Community Area': A unique number for each of 77 communities
    - 'Primary Type': The main type of the crime. You'll probably want to work with one of the values for this column of 'HOMICIDE', 'NARCOTICS', OR 'ASSAULT'.
2. Boundaries_Community.geojson: this containes information that defines the boundaries of 77 communities in Chicago.
    - `area_num_1` : the column in which each entry is the number of a community. If you view the file at https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Community-Areas-current-/cauq-8yn6, you will see the header names written in all uppercase. Ignore that. They are lower case in the dataframe.

We are supplying you with a function `com_count(df, col, value, unique_id, com_num)` that will extract the count of how many unique instances there are in dataframe `df` with:

- Entries in column with column name col equal to value; for example, entries in the column "Primary Type" equal to "HOMICIDE" (in the crime data), and
- With the unique identifier of the things we want to count unique_id; for instance, in the Supreme Court by Justice database this would have been 'docketId' and in the crime database this is 'Case Number', and
- Unique to this assignment: Community_ Area. Chicago communities are numbered from 1 to 77. (UIC is in the "Near West Side", Community 28.)

The function is:

```
def com_count(df, col, value, unique_id, com_num):
    """Returns count of unique_id in df com_num rows w/entry value in column col"""
    rows = df[df[col] == value] # boolean slice of rows we want
    if com_num not in rows['Community Area'].values:
        return 0
    grouped = rows.groupby('Community Area')
    return grouped[unique_id].count()[com_num]
```

When you call the function, you want the argument for df to be the variable you assigned to Crimes-2012_to_present.csv when you converted it to a DataFrame.

What's impressive about pandas: That function will run in under a second on a relatively new laptop for the full Chicago crime database from 2001 to 2019, which has over 6.9 million rows! For the 2012-onward subset, it should run in at most 2 seconds, and probably under 1 second, even on an older laptop.

# First task: Getting started with a dictionary

**Required** Write a function called `make_dictionary` that uses the `com_count` function to make and return a dictionary whose keys are community numbers (integers from 1 to 77) and whose values are the count of the specified thing for each community.

The inputs to `make_dictionary`, in order should be:

- df: A pandas dataframe (that must contain a column named "Community Area")
- col: The name of the column of interest (in addition to the Community Area column)
- value: The value of interest for that column
- unique_id: The name of the column with the unique identifier (e.g., Case Number, DocketId, Report Number) we want to count.

Your function should return the dictionary you made.

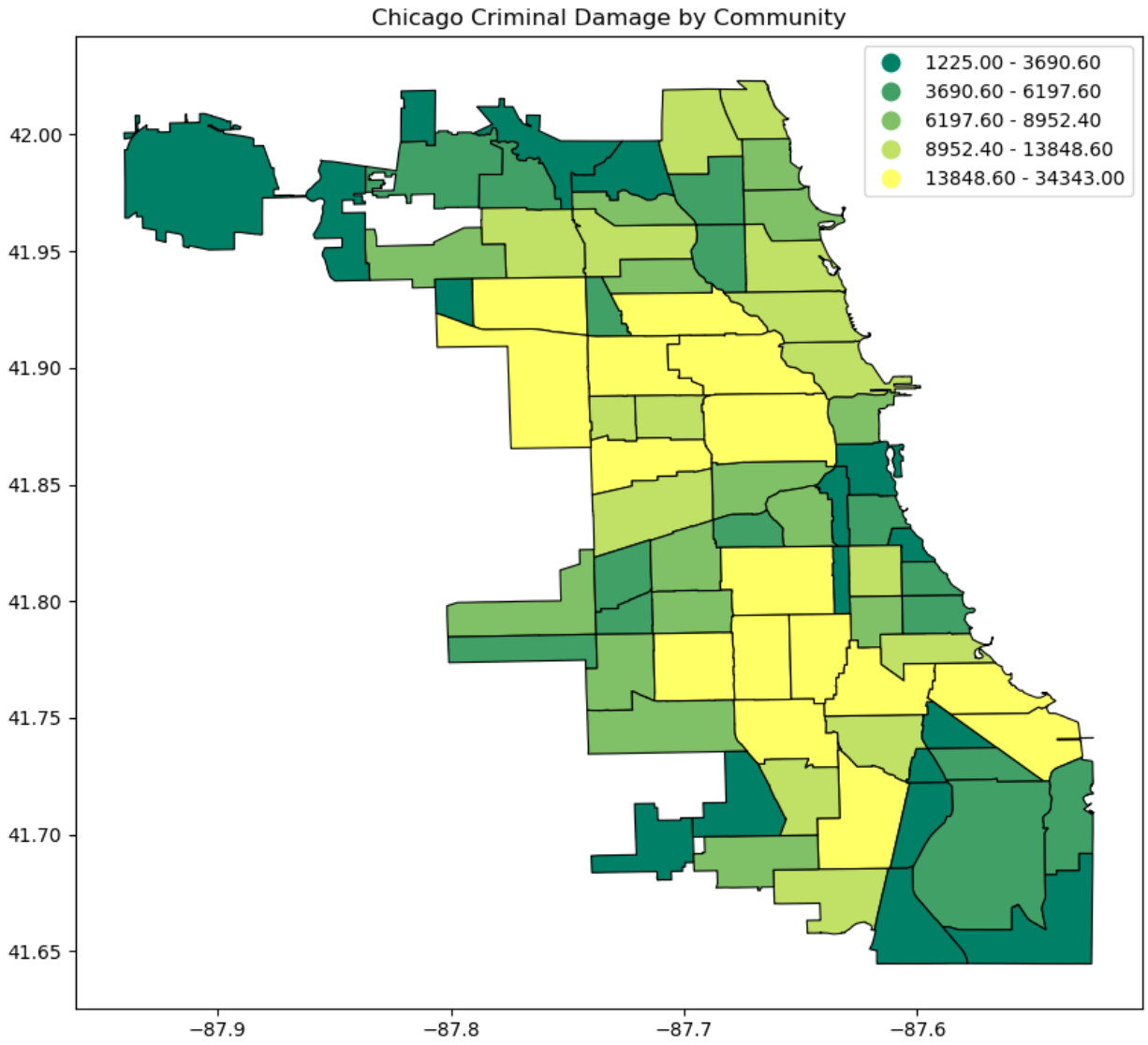
If you do it right this function can be written in under 10 lines.

# The rest of the task: Plotting heatmap of crime in Chicago's 77 communities

You need to make us three plots.

1. Plot the counts of at least one primary type of crime by community. Choose a type of crime that you consider to be fairly serious.
2. Plot the counts of another primary type of crime by community. Choose any other type of crime.
3. Your personal predictive policing plot. See below.

Here is an example for the primary category of crime 'CRIMINAL DAMAGE' (using all data from *2001* to a couple of weeks ago), which is not especially serious (relative to universe of all crimes in Chicago that is. *Call a good criminal lawyer immediately if you get arrested for it!*):




To do this, you need to modify `communities`. Recall that it is a special type of dataframe, that is, a geopandas dataframe. We need to add one more column to `communities` giving the value we want to plot (e.g., the number of homicides or assaults per community).

Specifically, you need to add a column 'Total' containing the number of crimes by community to the communities DataFrame. You add the column this way: `communities['Total'] = ls`, where list `ls` contains 77 integers representing the number of crimes of a certain type (homicides, assaults, etc.) in each of

the 77 communities. *Important:* The integers in `ls` MUST be in the order in which the community numbers occur in the `area_num_1` column of the communities DataFrame. You need to write the code puts the numbers in that order.

Making a dictionary with your function will help with all this.

*Important:* In mkaing the list that is in the order of the 'area*num*1' column of communities, be award that the type of each entry in that column will be *string*. You'll need to convert to *int* to use those as numbers.

You will need the following modules:

- matplotlib.pyplot
- pandas
- geopandas

# Plotting

Once you have added the column to communities, you can create the map with this code:

```
    fig, ax = plt.subplots()  # gives back tuple of figue, axes
    fig.set_size_inches (10, 10) # Just controls size; try other numbers
    communities.plot(column='Total', scheme='quantiles',
                         edgecolor='black', ax=ax, legend=True)
```

You can use the usual matplotlib.pyplot method to put a title on your plot.

If you are interested, you can find information about plotting with geopandas at http://geopandas.org/mapping.html. There is information about "scheme=quantiles" at https://pysal.org/scipy2019-intermediate-gds/deterministic/gds2-rasters.html.

## The more complicated part: Where would you put more police?

The goal of predictive policing is to use data to predict where crime is most likely to occur, so policing can be proactively moved into those areas.

We want you to make a prediction (by community) and show us your heatmap.

You need to combine *scaled* data from two (or more if you like) of the simple heatmaps you did. For each input, you should divide all the original values by the maximum value over all the wards to get a new set of values that will range between 0 and 1. The dictionary values() method will be useful for this, combined with the .max() method for lists. Remember that to get a list of dictionary d's values you need `list(d.values())`.

*Why scaling?* So we can combine different factors. For a given community, 75 is a lot of homicides, 2000 is a lot of narcotics incidents, and 7500 is a lot of criminal damage. If we just added those three factors without scaling, then the number of criminal damage crimes would overwhelm the other factors.

*How to combine?* That's up to you and your insight. You could simply add two factors, but maybe you want to apply some weighting: Maybe you think the number of murders is 2.5 times as predictive as the the number of narcotics crimes.

# Things to submit in your zip file

- (At least) 3 map heatmap images (you can do more if you feel like it) with titles.
- Your Python code, in a file named [your NetID]Project6.py (Whatever Python code you write into a file; it's okay if some of your work is done only at the command line. The function `make_dictionary` must be in the file.)
- A short text file, explaining the choices you made for your heatmap and why.