Practicum 5: Fatal Force Police Shootings

Due Date: Monday, October 11th at 11:59pm EST

Following the extrajudicial shooting of Black teenager Michael Brown in Ferguson, MO on August 9, 2014, there was an increased attention to fatal police violence. Unfortunately, data on police violence is poorly tracked and difficult to obtain through official sources. In the wake of the shooting, several police violence databases arose, including the Fatal Force project by data journalists at the Washington Post. Since 2015, the database has tracked fatal police shootings by US police officers by collecting data across news outlets, law enforcement websites, and other databases like Fatal Encounters.

This week, we will use the Fatal Force dataset, which is updated regularly. **Download the Fatal Force data** and place it *in the same folder* as the `practicum05.py` file.

We'll use the police shooting data to practice the following:

1. Reading many lines of data from a file
2. Visualizing data using lists
3. Linking datasets to highlight demographic inequities

Parsing the Fatal Force Data

Each row of the dataset indicates one shooting. In each row, there are 16 comma-separated data values, described in detail here.

Just like last week, we are going to read through each line one at a time. We can do this more easily now that we have `for` loops in our toolbox. To iterate through all of the lines, we simply have to do the following:

```
# Open the file
shootings_f = "fatal-police-shootings-data.csv"
infile = open(shootings_f, "r")

# Read the file
for line in infile:
    shooting = line.strip().split("\t")
```

Unlike the `while` loop, we don't have to check for a break condition; the `for` loop handles that for us automatically. ***Also, be aware that this week we're working with a TSV, which is a tab-separated values file.*** This changes what we put in `split` in the last line above.

There is a small but important difference between the file you used last week and the file you are using this week. The Fatal Force CSV has a *header* on the very first line. The header tells you the name of each value in each row. This is convenient when you may not be familiar with a dataset, or when you need help remembering what is in the file. However, this creates some small trouble for reading the lines. The very first line we read is going to be that header, so if we don't skip it, its values are going to end up in our data when we go to analyze it.

There are multiple ways that we can skip it. One way is to use a new function called `enumerate`. It lets us both read a line and count which line we're on at the same time. It works like this:

```
for line_num,line in enumerate(infile):
    shooting = line.strip().split("\t")
```

Notice that `enumerate` returns two things: the line number (`line_num`, in the example above) and the line itself (`line` above). Just like counting through the number the positions in a list, the line number starts at 0. So if we want to skip the header in the first line, we can use a conditional to check if we're on the first line, and then use the special keyword `continue`:

```
for line_num,line in enumerate(infile):
    if line_num == 0:
        continue

    shooting = line.strip().split("\t")
```

The keyword `continue` tells the `for` loop to skip everything else for that iteration of the loop and move on to the next item.

1 Describing Police Shootings

Start by writing Python code that prints out the name of every single person in the dataset. This is a reminder to you that each row here represents one of over 6,000 people that have been shot by police since 2015. There are people and stories behind each line of data, and we should take care with them. *Please comment out this line before your submission*.

Next, write Python code that does the following:

1. Counts the total number of shootings in the dataset, and prints it for the user.
2. Calculates what percent of the shootings were of Black men, and prints it for the user.

3. Calculates the youngest, oldest, and average age of all those in the dataset, and prints it for the user.

The program output should look like this:

```
There are 6924 shootings in the Fatal Force dataset.

23.4 percent of the shootings were of Black men.

The youngest police shooting victim was 6 years old.
The oldest police shooting victim was 91 years old.
The average police shooting victim was 37.1 years old.
```

2 Police Shootings Over Time

Write Python code that counts the number of police shootings that occurred in each year from 2015 to 2021. Store the values in a chronological list. For example, if you name your list `shootings_per_year`, then `shootings_per_year[0]` should be the number of shootings in 2015, `shootings_per_year[1]` should be the number in 2016, and `shootings_per_year[6]` should be the number in 2021.

Hint: You can use `split("-")` on the shooting date to get the year.

Now create a separate list that is simply the years. There's two ways you can do this.

```
# By hand. Works here, but wouldn't be good if we had many, many years
years = [2015, 2016, 2017, 2018, 2019, 2020, 2021]

# Using range. Ask yourself, why are we using 2022? Print it out to see
years = range(2015, 2022)
```

When you've done plots with Matplotlib before, you've plotted one point at a time. That is not how we typically use Matplotlib in practice. If our x- and y-coordinates are in lists, the `plot` function can take them and plot them all at once, rather than one at a time. Plot the number of police shootings per year and *label your plot*.

```
# Remember to import matplotlib
plt.plot(years, shootings_per_year, "-o", color="black")
```

Using `"-o"` turns it from a scatter plot into a line plot.

3 Police Shooting Inequities

On Canvas, I have provided you with another file: `state_population.csv`. The first row contains a header. Each row after that contains a state name followed by the percent of the population that is each demographic (white, Black, Native American, Asian, and Hispanic).

For a state of your choosing, write Python code that does the following:

1. Prints the number of police shootings in that state since 2015
2. Prints the overall population share of each demographic
3. Prints the share of each demographic for all police shootings since 2015

The program output should look like this:

```
CA
--
903 shootings in total

White
    71.54% (population share)
    28.02% (Fatal Force share)

Black
    2.68% (population share)
    17.17% (Fatal Force share)

Hispanic
    29.51% (population share)
    40.75% (Fatal Force share)

Asian
    5.54% (population share)
    3.54% (Fatal Force share)

Native American
    1.72% (population share)
    0.44% (Fatal Force share)
```

Finish Early?

1. Think about what data is *not* included in the Fatal Force dataset. Compare to the Fatal Encounters dataset, for example.
2. Try plotting the demographic population and shooting data as a series of parallel bars. For example, for white people, you could plot one red bar that shows the population share, and one blue bar that plots the Fatal Force share. You could then do the same with red and blue bars for the other demographics as well. Try to do the bar plots with lists and not by plotting one bar at a time.

3. Adjust the y-axis of the plot of shootings over time so that it starts at 0, rather than the default set by Matplotlib.
4. Think about why I gave you a TSV this week instead of a CSV. Hint: look at some of the names in the dataset.

4 Submit Your Work

When you are done, submit the following to Canvas in a zip folder:

1. The Python file (.py) containing your code answers for this practicum.
2. The Fatal Force file (.tsv)
3. The state population file (.csv)
4. A text file (.txt, .doc, .pdf, etc.) with any comments about anything you were not able to get working and what you tried to solve it.

Please name your zip folder `LastName_Practicum05.zip`.

This assignment was originally created by Sarah Shugars. It has been updated here by Ryan Gallagher.