Lab: Privacy and mobile phone metadata

Introduction
In this lab you will:

- Quantify the uniqueness of mobility traces revealed in mobile phone metadata
- Calculate origin-destination matrices representing aggregate mobility traces
- Explore the extent to which individual information is "leaked" in aggregate mobility statistics
- Experiment with suppression and differential privacy to "privatize" mobility traces
- Reflect on which privacy harms are mitigated by approaches like suppression and differential privacy, and which are not

The data analysis portion of the lab can be completed in any programming language or data analysis tool, including python, R, Stata, or in a combination of google sheets and google maps. This PDF contains hints for completing the lab in Python and google sheets/google maps. If using python, the file Privacy Lab - DIY.ipynb provides a Jupyter notebook template with prompts. The file Privacy Lab - Partially Filled.ipynb provides the same Jupyter notebook with some code partially written, making it easier to get started on the problems.

All data used in this lab are synthetic, but they are made to "look" like real mobile phone metadata, both in their format and in the underlying behavioral patterns.

This lab is loosely based on a number of papers using information from mobile phone data to infer locations and mobility, including:

- De Montjoye, Y. A., Hidalgo, C. A., Verleysen, M., & Blondel, V. D. (2013). Unique in the crowd: The privacy bounds of human mobility. Scientific reports, 3(1), 1-5.
- Kohli, N., Aiken, E., & Blumenstock, J. (2023). Privacy Guarantees for Personal Mobility Data in Humanitarian Response. arXiv preprint arXiv:2306.09471.

Part 1: Read in the data and examine uniqueness of mobility traces

- This lab uses the same data as our previous lab on mobility and mobile phone data. Read in data from the 'data' folder: synthetic transaction data ('cdr.csv'), tower locations ('towers.csv'), and a shapefile of Togo's prefectures ('prefectures.geojson').
- How many unique callers are in the dataset?
- Calculate the set of prefectures visited by each subscriber. How many unique prefectures does the average subscriber visit?
- How many unique prefectures does the average subscriber visit? How many subscribers have a unique prefecture set?
    - Google sheets hint: Use a pivot table to calculate the prefectures visited by each subscriber, then use data → data cleanup → remove duplicates to remove duplicate mobility traces.
- Now calculate the *ordered list* of prefectures visited by each subscriber. How many subscribers have a unique mobility trace?

Part 2: Calculate an origin-destination matrix

- Calculate an origin-destination matrix representing travel between each ordered pair of prefectures. A subscriber travels from prefecture 1 to prefecture 2 if they place a call in prefecture 1 and then another call in prefecture 2 (without calls in any other prefectures in between). For our purposes, if a subscriber travels from prefecture 1 to prefecture 2 more than once in our dataset, they will still only "count" once in the number of trips between these two prefectures.

    - Python DIY hint: A simple (if slow) way to implement this is to use a for loop to create datasets of subsequent prefectures for each subscriber, calculate the OD-matrix entires for each subscriber, and then sum together the matrices for each subscriber.
    - Google sheets hint: Start by identifying the subscriber's next prefecture after each transaction, and filtering to rows where the next prefecture is different from the original (and dropping duplicate pairs for each subscriber). Then, use a pivot table to count the number of times each directed pair appears.
- Visualize a "slice" of the origin-destination matrix on a map: How many subscribers are traveling out of Lome Commune to each of the other communes?
    - Python DIY hint: Use geopandas and the file 'prefectures.geojson' for plotting.
    - Google sheets hint: You can skip this part if working in google sheets – it would be prohibitively painful to try to plot these data in google sheets or maps.
- How many of the "aggregate" statistics we calculated contain data from a single subscriber? That is, how many of the origin-destination entries are 1?
- How many of the "aggregate" statistics we calculated contain data from fewer than ten subscribers?

Part 3: Suppression

- One easy approach to privatization (although one with no theoretical guarantees) is ***suppression***: removing OD matrix counts with small values. Implement this approach for your OD-matrix, setting values under 10 to 0.
- How accurate is this approach? Let's use two simple measures of accuracy: (1) the average absolute error in OD-counts across the matrix, and (2) the average absolute error in OD-counts for OD entries that are not 0 in the non-private matrix.

Part 4: Differential privacy

- One issue with suppression is that it only addresses small counts: larger counts may still "leak" individual information to sophisticated attackers. ***Differential privacy*** provides stronger theoretical guarantees about information protection. Implement differential privacy with *epsilon* = 0.1 by adding noise drawn from a Laplace distribution with scale 1/*epsilon* to every entry of the OD-matrix.
    - Python DIY hint: Fill in the following code to draw samples from a Laplace distribution: np.random.laplace(0, 1.0/epsilon, size=SIZE).
    - Google sheets hint: Google sheets does not have built-in functionality to draw from a Laplace distribution, so you can use the file laplace.csv to look up samples from a Laplace distribution for different values of *epsilon* (the values of *epsilon* are in the columns of the file).
- How accurate is your new "privatized" OD-matrix, using the same two metrics as before: The average error, and the average error in OD entries that are not 0 in the non-private matrix.
- **Bonus (optional):** Calculate and plot the privacy-accuracy trade-off, by calculating the average error for nonzero entries for at least ten different privatized matrices using *epsilon* values ranging from 0.01 to 10.

Part 5: Reflection

- How would you -- or a policymaker -- choose which value of ϵ to use for differential privacy? In other words, how would you navigate the trade-off between privacy and accuracy? Would you answer change in different humanitarian or development settings?
- Imagine you are trying to help a policymaker understand this tradeoff so that they can make an informed decision about epsilon. How might you explain this tradeoff? What do you think your explanation does well? Where might it fall short?
- What privacy harms do suppression and differential privacy provide protection from?

- What privacy harms do suppression and differential privacy **not** provide protection from?