CYBERNETICA

[ESTAT 2019.0232] Synthetic Test Data Generation

Ville Sokk

Technical document

Version 1.1

November 25, 2021

12 pages

Doc. Y-1440-13

Disclaimer. This document was prepared by Cybernetica AS as part of a procured project under Service Contract No ESTAT 2019.0232 (Ref. Ares(2020)2309804 - 30/04/2020). The opinions expressed in this document are those of the authors. They do not purport to reflect the opinions, views or official positions of the European Commission or its members.

Contents

1 Overview **4**
1.1 Document Scope 4

2 Introduction **5**

3 High-Level Description of the Algorithm **6**

4 Preparation **7**

5 Generating Anchors **9**

6 Generating Periodic Footprint Updates **10**

7 Generating High-Volume Data for Scalability Tests **12**

1 Overview

The ESTAT 2019.0232 project is a collaboration between Cybernetica and Eurostat to develop a proof-of-concept solution for passive mobile positioning data analysis which preserves the privacy rights of the data subjects. State-of-the-art trusted execution environment technology and pseudonymisation are applied in the solution to ensure privacy by design in calculations and output.

The project scenario involves a mobile network operator (MNO) and national statistics institute (NSI). The NSI uses the project solution to compute longitudinal analysis over a longer period by combining the location data of MNO subscribers with confidential auxilary data from the NSI.

The project approach honours the principles of data minimisation, privacy-by-design and purpose specification that lie at the foundation of the General Data Protection Regulation ("GDPR") of the European Union.

1.1 Document Scope

This document describes the algorithm used for generating the output of MNO's Module A for testing. This data is referred to as the periodic footprint updates or H file.

The full list of delivery documents:

- Solution Analysis
- Solution Architecture
- DPIA Evaluation Report
- User Guide for NSI
- User Guide for MNO-VAD
- User Guide for Auditors
- User Guide for MNO-ND
- Sharemind HI Documentation
- Sharemind HI ToS
- Sharemind HI License
- Synthetic Test Data Generation

2 Introduction

A footprint describes the usual environment of a mobile subscriber. A country is divided into 1 $\times$ 1 km square tiles and days are divided into three sub-periods: morning, day, evening. Each tile is identified by a pair of easting and northing coordinates. A footprint then is a list of tiles in the mobile subscriber's environment. Each tile has four associated values: one per sub-period and a total over all sub-periods. Each value indicates the "strength" of the tile during the sub-period i.e. how much time the mobile subscriber spent in the tile.

The format of the output is described in the ESTAT 2019.0232 Solution architecture document Section 4.2.3. In simplified terms the output is table with the following columns:

- subscriber identifier,
- tile easting coordinate,
- tile northing coordinate,
- total value,
- sub-period 1 value,
- sub-period 2 value,
- sub-period 3 value.

Periodic footprint updates must be generated for each period. In the proof of concept a period is one day. This means that one table is generated for each day of the report period (the period of data used in the analysis).

The algorithm uses the gridded population density of Belgium [1]. This allows the algorithm to generate somewhat realistic data where people are more likely to live and work near cities.

[1] Population according to the km² grid | Statbel, https://statbel.fgov.be/en/open-data/population-according-km2-grid

3 High-Level Description of the Algorithm

This section describes the general idea of the algorithm. Details will be covered in subsequent sections.

1. Generate three anchor points for each mobile user corresponding to the three daily sub-periods: night, day, evening. We can imagine the night anchor as the person's home and the day anchor as the person's workplace. There should be a higher probability of picking a highly populated tile as the night anchor. The other two anchors should be close to the night anchor.
2. Generate another set of three anchors using the same algorithm which are used when the person is vacationing. Pick a range of days as the vacation period. The length of the vacation should be between one day and three weeks.
3. For each day, user and sub-period generate a set of tile positions around the users anchor point of that sub-period. If the user is on a vacation during this particular day use the vacation anchor points. The distribution of positions around an anchor should have a heavy tail (i.e. there are few positions that are far from the anchor). Generate a random value for each user, tile, sub-period combination.

The vacation period and heavy-tailed distribution create noise in the users footprint. The purpose of this is to test if Module C of the algorithm can successfully filter the footprints.

4 Preparation

Before the data is generated a data structure is prepared for sampling night anchors according to population density. That means we want to pick tiles randomly such that tiles with higher population are more likely to be picked.

1. Read the gridded population data. It should contain the coordinates and population of each tile.
2. Calculate vector $\mathbf{c}$ which will be used for sampling using the following steps.
   a) Arrange the tiles in a fixed order. For example, sort tiles first by easting and then by northing coordinate. Construct vector $\mathbf{p}$ of population count in each tile.
   b) Let n be the number of tiles in the data. Calculate the cumulative sum $\mathbf{s}$ of tile population counts. That is $\mathbf{s}_i = \sum_{j=1}^{i} \mathbf{p}_j \ (i = 1 \dots n)$.
   c) Construct vector $\mathbf{c}$ which is the cumulative sum vector divided point-wise by the total population (i.e. sum over all values of $\mathbf{p}$). That is $\mathbf{c}_i = \mathbf{s}_i / \mathbf{s}_n$.

   All of the values of $\mathbf{c}$ are in the range $[0, 1]$ and in monotonically increasing order. To pick a random tile we generate a random value r in the range $[0, 1]$ and binary search for r in vector $\mathbf{c}$. The algorithm for calculating $\mathbf{c}$ is illustrated on figure 1.
3. For each tile find a value for the r_{tile} parameter. r_{tile} is the smallest natural number such that if the tile is in the center of a $d \times d$ square area (where $d = 2 \cdot r_{tile} - 1$) then the population of the area is at least $k_{radius} = 5000$.

   r_{tile} describes the typical travel distance of users whose night stay is in given tile. When we have found a night anchor for a user we use the r_{tile} of the night anchor to sample anchors for the other two sub-periods.

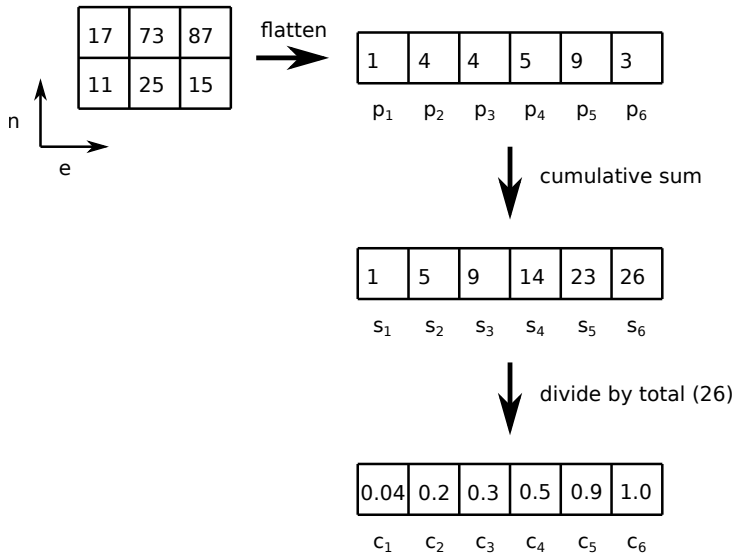

Figure 1. Converting tile populations into the $\mathbf{c}$ vector that is used for sampling tiles.

5 Generating Anchors

Anchors are tiles near which the user is positioned during a sub-period (night, day, evening). Six anchors in two sets will be generated for each user: three anchors for regular days and three anchors for vacation days. The process for generating one set of anchors is as follows.

1. Calculate anchor $\mathbf{x}_{night}$ for night-time (sub-period 1):
   a) Generate a random number $r \in [0, 1]$.
   b) Binary search for r in vector $\mathbf{c}$ which was created in the pre-processing stage. Retrieve the coordinates of the found tile.
   c) Choose a uniformly random position inside this tile.
2. Let r_{night} be the r_{tile} value of the $\mathbf{x}_{night}$ tile. Generate anchors for sub-periods 2 and 3:
   a) Let (x_{night}, y_{night}) designate the easting and northing coordinates of the night tile. Generate n points (x_i, y_i) where $x_i \sim \mathcal{N}(x_{night}, r_{night})$ and $y_i \sim \mathcal{N}(y_{night}, r_{night})$ $(i = 1, \ldots, n)$.
   b) Round the coordinates of these n points to get tile coordinates and choose the tile with the highest population as the anchor.
3. Let $\mathbf{a}_2, \mathbf{a}_3$ designate the anchors of sub-periods 2 and 3 respectively. Move $\mathbf{a}_3$ closer to $\mathbf{a}_2$ by generating a random value $r \in (0, 1)$ and calculating $\mathbf{a}'_3 = r\mathbf{a}_2 + (1 - r)\mathbf{a}_3$.
4. Calculate $r_{poserror} = r \cdot r_{tile}$ for each anchor where r is a random number $r \in [PEC1, PEC2)$. $PEC1$ (default 0.5) and $PEC2$ (default 2) position error coefficients are generator parameters. $r_{poserror}$ will be used for sampling tiles when generating intra-period footprints.

6 Generating Periodic Footprint Updates

For each day and each user the periodic footprint update is generated using the following algorithm:

1. For each sub-period $1, 2, 3$:
   a) Decide on n which designates how many tiles to generate. It is either a preset constant or with some non-zero probability $p_{silence}$ it is zero so tiles are not generated for the given sub-period. “Silence” simulates a situation where the Solution was not available.
   b) Sample n points using polar coordinates where the angle ϕ has uniform distribution over the interval $[0, 2\pi]$ and the radius has mixture distribution containing normal and Pareto II distribution components with normal component weight p_{normal} = 0.7. This means with probability p_{normal} the radius is drawn from the normal distribution and with probability $1 - p_{normal}$ it is drawn from the Pareto II distribution.
   The default scale parameter of the Pareto II distribution is rm=2[km] and the standard deviation of the normal distribution is $r_{poserror}$ which was calculated earlier for each anchor.

$$\begin{aligned} P[radius] =& p_{normal} \cdot \mathcal{N}(a, r_{poserror}) + \\ & (1 - p_{normal}) \cdot ParetoII(a, alpha = 1.1, rm) \end{aligned}$$

   The algorithm for sampling tiles around anchors is illustrated on figure 2.
   c) Let $\mathbf{a}$ designate the coordinates of the anchor of the current sub-period. If the day is in the user’s vacation period take the anchor from the user’s vacation anchors list, otherwise from the user’s regular anchors list. Convert the generated points from polar coordinates to Cartesian coordinates:

$$\begin{aligned} e_{offset} =& r \cdot \cos(\phi) \\ n_{offset} =& r \cdot \sin(\phi) \end{aligned}$$

   $\mathbf{a} + \langle e_{offset}, n_{offset} \rangle$ is now the sampled point near anchor $\mathbf{a}$.
   d) Round the sampled point coordinates to get easting and northing coordinates of n tiles that the user visited in this sub-period.
   e) For each tile generate a random value $r \in [0, 1)$.
2. Calculate footprint data of sub-period 0 (whole day) from the data of sub-periods $1, 2, 3$. That is, group the data by tile coordinates and add the values in each group.

3. Post-process the generated data. The algorithm described up to this point generates footprint data where a tile is usually visited by a mobile user in only one sub-period. This is due to the fact that each sub-period has a separate anchor point. The goal of the post-processing is to increase the likelihood that some tiles are visited in more than one sub-period. The following algorithm is applied to each tile and mobile user pair:

   a) Let $p = p_{fillfromnight}$ if the night sub-period value of this tile is > 0 and $p = p_{fillfromday}$ otherwise.

   b) For each sub-period with a zero value: with probability p replace the sub-period value with a uniform random value from the range $(0, w_{fillmax})$.

   The default values are $p_{fillfromnight} = 0.4,\ p_{fillfromday} = 0.2,\ w_{fillmax} = 1.$

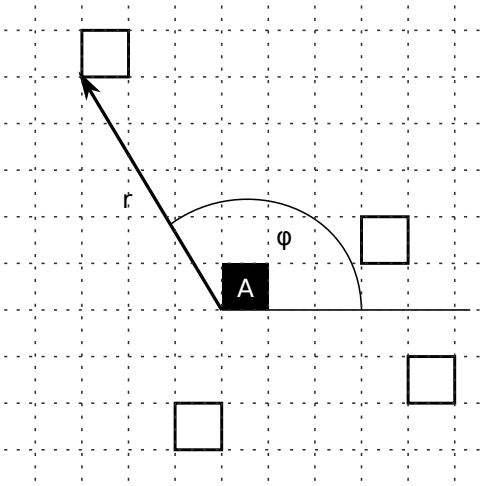

Figure 2. Sampling tile positions around anchor A that are visited in a sub-period. The angle ϕ is uniformly distributed over the interval $[0, 2\pi]$ and the radius r has Pareto distribution with shape parameter 1.1.

7 Generating High-Volume Data for Scalability Tests

The generated data has several distinct use cases:

- For functionality tests comparing C++ enclave code to Python code. The dataset size must be modest so that the Python implementation runs within reasonable time limits.
- For scalability tests. The dataset size must be close to the maximums specified in the ESTAT 2019.0232 Solution architecture document Section 2.2.3.1.
- For demo purposes. Only a subset of the data is needed, visual appearance is important.

The data that would be analysed in production is expected to be quite large. If the software must scale to 100 million users, 600 000 tiles and the report period is multiple months then generating intra-period footprints for scalability testing using a simple implementation of the described algorithm could be relatively inefficient.

For scalability testing the generator can duplicate the data. The footprints generated for one user can be used for multiple users.

For visual demo we can do thorough emulation in a smaller area. The generator allows specifying a rectangular area and only tiles within this area will be processed.