FlowKit: An open-source toolkit for mobile phone data analysis

26 January 2022

Flowminder Foundation
James Harrison, PhD
Data analyst & developer
Science & Innovation
Solid academic research. 50+ peer reviewed publications

Our team & work
38 staff to enable data driven decision support for LMICs

Some of our supporters:

JIPS
Gavi
Hewlett Foundation
IDB
IOM
Inter-American Development Bank

Backstory
2010 Haiti Earthquake & cholera epidemic

Photo credit: ECHO, Susana Perez Diaz, Flickr

Non-profit funded by & supporting key actors

Our supporters

Non-profit funded by & supporting key actors

Science & Innovation
Solid academic research. 50+ peer reviewed publications

Our team & work
38 staff to enable data driven decision support for LMICs

Flowminder.org
Flowminder MNO collaborations to date

Countries where Flowminder has collaboration with MNOs (present and past):

- Curacao (x 2 MNOs)
- Haiti
- Sierra Leone
- Ghana
- DRC (x 2 MNOs)
- Namibia
- Mozambique (x 3 MNOs via INCM)
- Nepal
- Papua New Guinea
- Western African country (in discussion)
What is FlowKit?

- **Open-source** (MPLv2) software suite
- Enables **secure processing, analysis and granular access control** to CDR data for humanitarian and development purposes
- Designed to be installed within mobile network operator’s firewall
- Containerised to **simplify deployment**

-> [https://flowkit.xyz](https://flowkit.xyz)
FlowKit in the wider context

FlowKit
Capture Process Analyse

Further analytics
Integration with other data sources
Development of new analytical methods

Outputs
Population/mobility insights
Tools
Capacity strengthening

CDR data from MNO
Other data (e.g. surveys, satellite imagery)

Decision makers’ needs
FlowKit components

- Mobile data
- Database
- Tools
- API

Ingestion and QA
Processing
Authentication and controlled access
FlowKit components: Ingestion and QA

**Purpose**: Reduce effort to get data “analysis-ready”

- **Data ingestion**
  - Connection made during installation
  - New data ingested automatically
  - Reduces effort prior to processing data

- **Quality assurance**
  - QA checks run automatically on new data
  - Identify data issues which may affect outputs

- **Data model**
  - Utility tables (events, cells, geography)
  - Familiar structure to underlie queries
**FlowKit components: Processing**

**Purpose:** Method use, re-use and creation

<table>
<thead>
<tr>
<th>Code and methods</th>
<th>PostgreSQL database</th>
<th>Flexibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python library for constructing SQL queries</td>
<td>Caching: break down large queries, re-use intermediate results</td>
<td>May already provide all that is needed</td>
</tr>
<tr>
<td>Mobility, characteristics, subsetting</td>
<td>PostGIS for geospatial analysis</td>
<td>Extensible, modular design</td>
</tr>
<tr>
<td>Re-use tested methods</td>
<td>Efficient performance on a single server</td>
<td>Designed to grow</td>
</tr>
</tbody>
</table>
FlowKit components: Authentication, access control

**Purpose**: Permissions and integration

**Data security**
- Outputs are anonymised
- Grant flexible access to aggregated results without requiring access to individual-level data

**Interface (API)**
- Trigger queries and retrieve results
- Protect users from internal changes
- Facilitates integration with other tools

**Authentication**
- Granular permissions
- All transactions logged for auditing
- Reduces risk of data leak
Processing sensitive data

FlowKit protects the privacy of individual subscribers in the following ways:

- All processing of individual-level data occurs within MNO's premises
- API enables flexible querying of data, while only allowing anonymised outputs to be retrieved
- Outputs from API are aggregated over subscribers, with any rows corresponding to 15 or fewer subscribers redacted (k-anonymity)

CDR-derived insights should never permit the identification of individual subscribers.
Processing sensitive data

(continuation)

- Granular access control facilitates data minimisation
- Audit logs
- Direct identifiers (phone numbers etc.) are pseudonymised before ingestion into FlowKit, to reduce risk of re-identification in the event of a data leak

Outputs from FlowKit’s API are aggregated, hence protecting the privacy of subscribers.
Integrating mobile operator data into official statistics
Flowminder has partnered with Ghana Statistical Service (GSS) and Vodafone Ghana.

Flowminder is supporting GSS to use CDR data from Vodafone Ghana to produce information on population mobility and characteristics.

The aim is to strengthen humanitarian and development decision-making (e.g. for public health, disaster preparedness or transportation planning).
Official statistics: Ghana

How FlowKit helps:

- Vodafone Ghana can provide on-demand access to CDR-derived aggregates without sharing individual-level data
- GSS can use methods implemented in FlowKit (e.g. home location estimation) to produce data products that support national statistics

This public-private collaboration is first of its kind in Ghana, and one of the first in Africa.
Mapping for Health in DRC

Role of mobility data (data provided by Vodacom RDC):

- Estimates of hard-to-reach / highly mobile / displaced populations
- Monthly updates on population movements, to aid resource supply planning
- Alert health planners when sudden large-scale displacements occur
- Evaluate the scope for updating population density estimates and projections using mobile phone usage data

Initiative to improve the effectiveness and equity of vaccination interventions in the Democratic Republic of the Congo (DRC).
Mapping for Health in DRC

How FlowKit helps:

- Automated tools built on top of the FlowKit API to regularly produce and disseminate data outputs
- Individual-level data never leave Vodacom’s system
- Convenient platform for prototyping new methods
- Methods developed are built into FlowKit for re-use in future projects
Demo

Using FlowKit to produce aggregates from CDR data
Prerequisites

- FlowKit installed on a server within MNO premises
- Data (CDR, cell tower locations, geographic boundaries) ingested for the period of interest
- Permissions to access the required aggregates granted by a system administrator
- FlowKit API exposed at an accessible URL
- User is familiar with python data analysis ecosystem

More information at https://flowkit.xyz
Get an access token
Connect to FlowKit API

Using the token we have just obtained, we can create a connection to the FlowKit server using `flowclient.connect`:

```python
[3]: TKM = "eyJ2ZXBAI0I1QV1iLCiGc1i2SUZIINiJ9.eyJpYWQiOiJE2NDI3MTgyNDA0MzMiI6I6MTY0Mjc0ODE0MzI0NjMlMmFhOTk=
[4]: conn = flowclient.connect(
    url="http://flowapi:9000",
    token=TKM,
)
```

We can test the connection by getting a list of the dates for which data are available:

```python
[5]: flowclient.get_available_dates(connection=conn)
```

```json
[5]: {'calls': ['2016-01-01',
             '2016-01-02',
             '2016-01-03',
             '2016-01-04',
             '2016-01-05',
             '2016-01-06',
             '2016-01-07',
             '2016-01-08',
             '2016-01-09',
             '2016-01-10',
             '2016-01-11',
             '2016-01-12',
             '2016-01-13',
             '2016-01-14',
             '2016-01-15',
             '2016-01-16',
             '2016-01-17',
             '2016-01-18',
             '2016-01-19',
             '2016-01-20',
             '2016-01-21',
             '2016-01-22',
             '2016-01-23',
             '2016-01-24',
             '2016-01-25',
             '2016-01-26',
             '2016-01-27',
             '2016-01-28',
             '2016-01-29',
             '2016-01-30',
             '2016-02-01',
```
Define queries

```python
[6]:
before_locations_spec = flowclient.modal_location_from_dates_spec(
    start_date="2016-01-27",  # Start and end data for the two weeks
    end_date="2016-02-10",  # Immediately prior to the crisis start
    method="most-common",  # Assign subscribers to their most common location each day
    aggregation_unit="admin3", # 'admin3' is district-level
)

after_locations_spec = flowclient.modal_location_from_dates_spec(
    start_date="2016-02-10",  # Start and end data for the two weeks
    end_date="2016-02-24",  # Immediately after the crisis start
    method="most-common",  # Assign subscribers to their most common location each day
    aggregation_unit="admin3", # 'admin3' is district-level
)

[7]:
before_subscriber_counts_query = flowclient.spatial_aggregate(
    connection=conn, locations=before_locations_spec
)

after_subscriber_counts_query = flowclient.spatial_aggregate(
    connection=conn, locations=after_locations_spec
)

[8]:
after_subscriber_counts_query.parameters

[8]:
{'query_kind': 'spatial_aggregate',
'locations': [{'query_kind': 'modal_location',
'locations': ['date': '2016-02-10',
'aggregation_unit': 'admin3',
'method': 'most-common',
'event_types': None,
'subscriber_subset': None,
'mapping_table': None,
'geom_table': None,
'geom_table_join_column': None,
'hours': None},
{'query_kind': 'daily_location',
'date': '2016-02-11',
'aggregation_unit': 'admin3',
'method': 'most-common',
'event_types': None,
'subscriber_subset': None,
'mapping_table': None,
'geom_table': None,
}]
```
Get query results

```python
Get query results
```

```python
before_result = before_subscriber_counts_query.get_result()
after_result = after_subscriber_counts_query.get_result()

Parts run: 100% [101/101 [00:19<00:00], 6.66q/s]
Parts run: 100% [68/68 [00:10<00:00], 6.47q/s]
```

```python
before_result
```

<table>
<thead>
<tr>
<th>pcode</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPL.1.3.3_1</td>
<td>417</td>
</tr>
<tr>
<td>NPL.4.1.1_1</td>
<td>631</td>
</tr>
<tr>
<td>NPL.1.1.3_1</td>
<td>6176</td>
</tr>
<tr>
<td>NPL.3.1.4_1</td>
<td>1179</td>
</tr>
<tr>
<td>NPL.1.2.1_1</td>
<td>1337</td>
</tr>
<tr>
<td>NPL.5.3.3_1</td>
<td>1043</td>
</tr>
<tr>
<td>NPL.1.1.5_1</td>
<td>1581</td>
</tr>
<tr>
<td>NPL.3.1.6_1</td>
<td>598</td>
</tr>
<tr>
<td>NPL.3.2.4_1</td>
<td>397</td>
</tr>
<tr>
<td>NPL.1.3.4_1</td>
<td>1111</td>
</tr>
<tr>
<td>NPL.2.3.1_1</td>
<td>186</td>
</tr>
<tr>
<td>NPL.2.3.5_1</td>
<td>614</td>
</tr>
<tr>
<td>NPL.1.3.2_1</td>
<td>970</td>
</tr>
<tr>
<td>NPL.2.1.5_1</td>
<td>3022</td>
</tr>
<tr>
<td>NPL.2.2.2_1</td>
<td>1363</td>
</tr>
<tr>
<td>NPL.4.2.5_1</td>
<td>503</td>
</tr>
<tr>
<td>NPL.1.2.5_1</td>
<td>941</td>
</tr>
<tr>
<td>NPL.1.1.8_1</td>
<td>805</td>
</tr>
<tr>
<td>NPL.1.1.1_1</td>
<td>505</td>
</tr>
<tr>
<td>NPL.1.1.7_1</td>
<td>103</td>
</tr>
<tr>
<td>NPL.5.3.1_1</td>
<td>783</td>
</tr>
</tbody>
</table>
Further processing

Now that we have the results, we can perform further processing. This step does not require Flowkit - we can use data analysis tools we are familiar with.

We will calculate the percentage change in resident counts after the start of the crisis.

```python
joined_results = before_result.merge(
    after_result, on="pcod", how="outer", suffixes=("_before", "_after"),
).fillna(0)

joined_results["percent_change"] = (100 * (joined_results["value_after"] / joined_results["value_before"] - 1))

joined_results
```

<table>
<thead>
<tr>
<th>pcod</th>
<th>value_before</th>
<th>value_after</th>
<th>percent_change</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPL.1.3.1</td>
<td>417</td>
<td>500.0</td>
<td>19.904077</td>
</tr>
<tr>
<td>NPL.4.1.1</td>
<td>631</td>
<td>734.0</td>
<td>16.323296</td>
</tr>
<tr>
<td>NPL.1.1.1</td>
<td>6176</td>
<td>0.0</td>
<td>-100.000000</td>
</tr>
<tr>
<td>NPL.3.1.4</td>
<td>1179</td>
<td>1474.0</td>
<td>25.021204</td>
</tr>
<tr>
<td>NPL.1.2.1</td>
<td>1337</td>
<td>1683.0</td>
<td>25.878633</td>
</tr>
<tr>
<td>NPL.5.3.3</td>
<td>1043</td>
<td>1263.0</td>
<td>21.093001</td>
</tr>
<tr>
<td>NPL.1.3.5</td>
<td>1181</td>
<td>1961.0</td>
<td>24.035421</td>
</tr>
<tr>
<td>NPL.2.1.6</td>
<td>598</td>
<td>691.0</td>
<td>15.551839</td>
</tr>
<tr>
<td>NPL.3.2.4</td>
<td>397</td>
<td>482.0</td>
<td>21.410579</td>
</tr>
<tr>
<td>NPL.1.3.4</td>
<td>1111</td>
<td>1395.0</td>
<td>25.562556</td>
</tr>
<tr>
<td>NPL.2.3.1</td>
<td>186</td>
<td>244.0</td>
<td>31.182796</td>
</tr>
<tr>
<td>NPL.2.3.5</td>
<td>614</td>
<td>664.0</td>
<td>8.143322</td>
</tr>
<tr>
<td>NPL.1.3.2</td>
<td>970</td>
<td>1187.0</td>
<td>22.371134</td>
</tr>
<tr>
<td>NPL.2.1.5</td>
<td>3922</td>
<td>4714.0</td>
<td>20.193779</td>
</tr>
<tr>
<td>NPL.2.2.2</td>
<td>1363</td>
<td>1625.0</td>
<td>19.222304</td>
</tr>
<tr>
<td>NPL.4.2.5</td>
<td>503</td>
<td>579.0</td>
<td>15.109344</td>
</tr>
<tr>
<td>NPL.1.2.5</td>
<td>941</td>
<td>1190.0</td>
<td>26.461211</td>
</tr>
<tr>
<td>NPL.1.1.1</td>
<td>806</td>
<td>0.0</td>
<td>-100.000000</td>
</tr>
</tbody>
</table>
Get geography data

To visualise our results on a map, we need the geographic boundaries of the districts. We can get these from FlowKit using the `get_geography` function.

```python
admin3_geojson = flowclient.get_geography(connection=conn, aggregation_unit="admin3")
```

GeoJSON(admin3_geojson)
Visualise results