Trusted Smart Statistics:
methodological developments based on new data sources

2022-IT-TSS-METH-TOO
Project n. 101132744

Work Package 3
Methodologies and open source tools for integrating MNO and non-MNO data sources

Deliverable 3.3

Report on Open-source software and software release

June 2025
Partner in charge: Istat (Italy)

Authors[1]:
L. Di Consiglio, M. D'Orazio, C. Faricelli, T. Pichiorri, A. Piovani, T. Tuoto, Istat (Italy)
R. Cărtuță, B. Oancea, INS (Romania)
L.-C. Zhang, SSB (Norway)
L. Sanguiao Sande, S. Barragán Andrés, J. Verdú Naranjo, INE (Spain)
Y.A.P.M. Gootzen, J. van der Valk, CBS (the Netherlands)

MNO-MINDS | Eurostat CROS (europa.eu)

Co-funded by
the European Union

[1]Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union not the grating authority can be held responsible for them.

Index

Introduction

This deliverable presents the development and application of open source software tools for generating synthetic mobile network operator (MNO) data or using real MNO, alone or in conjunction with other data with the aim of supporting the production of official statistics in areas such as population, transport, and tourism. This complements Part III of Deliverable D3.2, which includes seven application scenarios showing how the methods presented in Part II of D3.2 can be used to combine MNO and non-MNO data to produce official statistics.

The first section of this report, prepared by the National Institute of Statistics of Romania, addresses the challenge of limited access to real mobile network data due to privacy concerns. It offers a robust simulation framework that mimics real-world mobile network events. Data generated using this tool are used in Application Scenario VI (Sensor Presence), which was prepared by Statistics Norway. Section II from Statistics Norway relates to the application of the Swedish National Travel Survey (Trafikanalys), which integrates mobile network data with traditional survey data to analyze travel patterns. All analyses and implementation were conducted using the R programming environment. Section IV investigates a similar topic and describes R tools used at Istat to estimate commuter statistics by combining MNO data with relevant non-MNO data.

Sections III and V illustrate applications related to tourism statistics prepared by INE Spain and Istat Italy, respectively. Section III illustrates the advanced algorithms developed by INE Spain to transform raw mobile totals into usable tourism data. This data is then combined with traditional survey estimates using techniques such as transfer learning. Section V illustrates how registry data, collected by Istat, and MNO data about nights spent, provided by MNO operators, can be used together to produce improved statistical outputs in the R environment.

Sections VII and VIII relate to the application of the QR approach to produce statistical outputs based on MNO macro data. The first experiment, conducted by Statistics Norway, considers a sample of households selected from the household register. It uses indicators about possession of a SIM card for mobile phone services to produce statistical outputs using the R environment. Section VIII uses data from an annual survey conducted by Istat to perform a QR experiment and assess potential errors when using solely the subsample of individuals who use mobile phones. In this case, the analyses are also carried out using the facilities of the R environment.

Some of the code developed for the aforementioned applications is available in the MNO-MINDS GitHub repository:

https://github.com/mno-minds/MNO-MINDS

I. Synthetic mobile networks data generation

National Institute of Statistics, Romania

1. Introduction

Official statistics are undergoing continuous modernization, both in terms of production processes and the integration of emerging digital data sources. Among the most promising of these sources are mobile telecommunication networks, which hold significant potential for use in areas such as population, transport, and tourism statistics. However, accessing this type of data remains challenging for National Statistical Institutes, primarily due to concerns around data confidentiality and privacy.

To address this limitation, synthetic data has emerged as a valuable alternative—enabling statisticians to develop and test methodological approaches even in the absence of real data. In this context, we introduce an open source software tool designed to generate synthetic mobile network event data. The tool employs an agent-based simulation model, positioning agents on a digital map and enabling them to move based on predefined mobility patterns. It then records the resulting network events triggered by interactions between mobile devices and the network infrastructure.

A key advantage of using synthetic data lies in its ability to not only replicate realistic network event datasets but also provide a complete "ground truth"—a layer of information that is typically unattainable in real-world scenarios. Our simulator generates individual-level data, including event timestamps, cell IDs, timing advance values, event types, and more. It also utilizes telecommunication parameters describing network infrastructure, such as the geographical location of Base Transceiver Stations (BTS), their emission power, path loss exponents, and other technical attributes (see Salgado et al., 2021).

The software is open source and freely accessible on GitHub at https://github.com/bogdanoancea/simulator. The repository includes the source code, a Docker image, a Windows installation kit, and a collection of input files.

2. Methods

To simulate mobile network data at a granular level, we adopted a **discrete event system** approach, enabling the execution of micro-simulations. These simulations generate synthetic datasets that facilitate the rapid development and testing of methodological frameworks for producing official statistics based on mobile network data. Given the inherent limitations in accessing ground truth data—such as the precise location of mobile devices at specific time intervals—synthetic simulations are essential for validating statistical models in this domain.

The main features of our synthetic data generator are:

- It uses various maps (geographic areas) for simulations, with the ability to overlay a reference grid to calculate population densities and location probabilities.
- It supports simulations involving **multiple mobile network operators**.

- It supports customizable network configurations, including the positioning and technical parameters of **Base Transceiver Stations (BTS)** (antennas).
- It provides flexibility in configuring the **handover processes**.
- It allows user-defined **mobility patterns** for agents within the simulation.
- It is computationally efficient, being capable of handling simulations involving large-scale populations.

We are aware that there are other simulation tools too, such as the **cdr-gen project** (Bordin, 2017), **NetSim** (Tetcos, 2019), **OPNET Network Simulator** (Zheng & Hongji, 2012), **SUMO** (Krajzewicz et al., 2012), and **MATSim** (Horni et al., 2016). However, these tools are very complex, difficult to configure and they are mainly oriented toward telecom engineering needs, not for statistical needs.

As a result, we opted to use an inhouse developed simulation platform based on an **agent-based modelling** approach which was written in **C++** due to its superior performance in terms of execution speed and memory efficiency—both critical factors for large-scale simulations.

Our simulation tool is organized around three key modules: a **GIS module**, an **agent-based simulation module**, and a **computational module** that models the interaction between mobile devices and the mobile network. The architecture follows a **layered design**, with each layer providing dedicated services to the layers above.

Figure 1 illustrates the five-layer structure of the tool:

1. **Base libraries and utilities** – including components such as XML/CSV parsers and random number generators.
2. **GIS Layer** – built on the **GEOS C++ library**, providing spatial operations.
3. **Data encapsulation layer** – defines core simulation entities like individuals and mobile devices.
4. **Simulation layer** – executes the generation of synthetic network event data.
5. **Computation layer** – estimates location probabilities for each device during simulation in a simplified manner.

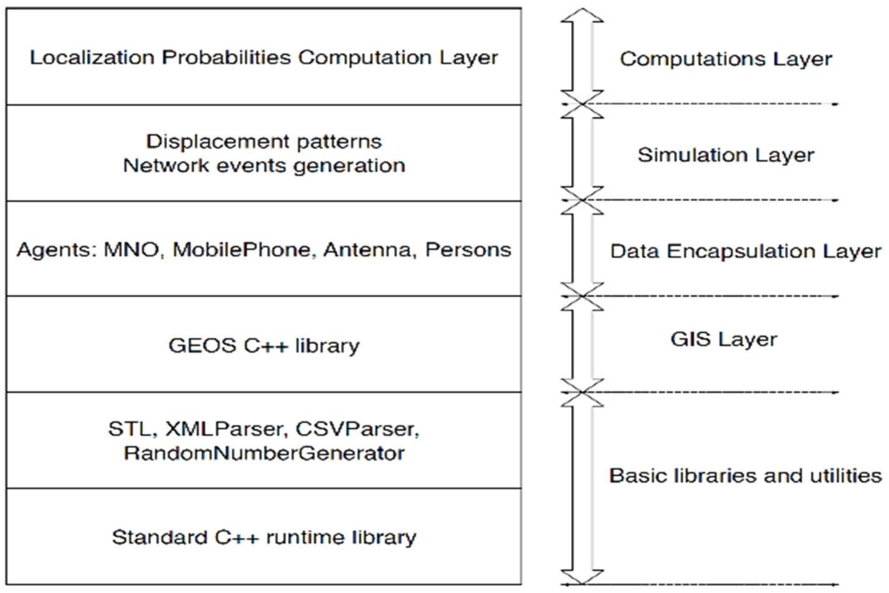

Localization Probabilities Computation Layer
Displacement patterns
Network events generation
Agents: MNO, MobilePhone, Antenna, Persons
GEOS C++ library
STL, XMLParser, CSVParser, RandomNumberGenerator
Standard C++ runtime library
Computations Layer
Simulation Layer
Data Encapsulation Layer
GIS Layer
Basic libraries and utilities

Figure 1. The architecture of the simulation tool

2.1 The simulation process

Each simulation begins with a **synthetically generated population**, whose size and demographic attributes (e.g., age, gender) are user-defined. The tool supports two movement models for individuals:

- Slow mobility (representing walking)
- Fast mobility (representing transport via cars or public transit)

Each person may carry 0, 1, or 2 mobile devices and follows user-configured **mobility patterns**, including:

- Random walk
- Random walk with drift
- Lévy flights
- Manhattan mobility
- Home-work mobility with anchor points (e.g., intermediate stops like a shopping mall)
- Home-work with anchor points and Manhattan routing

All mobility parameters are set through configuration files. Figure 2 shows examples of three such patterns: Lévy flights, Manhattan movement, and home–work routes with anchor points.

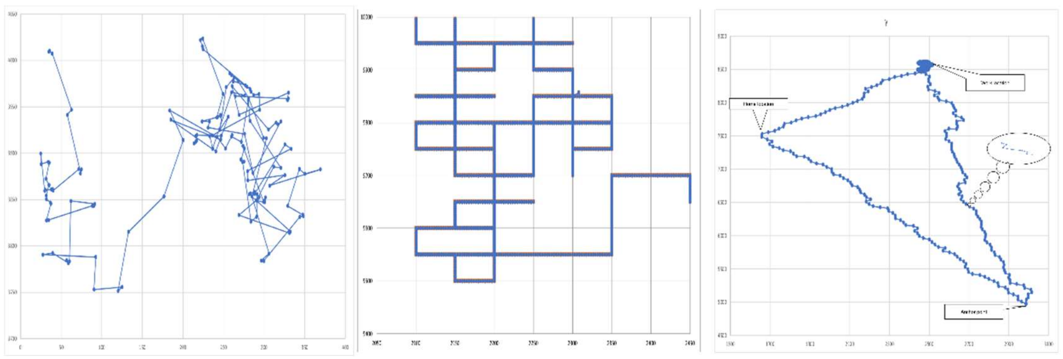


Figure 2. Three examples of mobility patterns: from left to right – Levy flights, Manhattan move, home-work with anchor points.

Out of the six mobility patterns implemented, the last 4 are useful for simulating human mobility needed in our project.

Lévy flights are a class of random walk processes characterized by a heavy-tailed probability distribution of step lengths. Unlike traditional Brownian motion, where step lengths follow a normal distribution, Lévy flights allow for occasional very long steps interspersed with many short ones. Mathematically, the step lengths in a Lévy flight follow a power-law distribution, typically of the form $P(l) \sim l^{-\mu}$, where $1 < \mu \leq 3$. This behaviour results in trajectories that can span a wide area in fewer steps, exhibiting both local exploration and long-distance jumps—a property that has significant implications in fields like statistical physics, biology, and mobility modelling.

One of the most intriguing applications of Lévy flights is in modelling **human mobility patterns**. Empirical studies using mobile phone data, GPS traces, and other tracking systems have consistently shown that human travel behaviour does not conform to simple, regular movement patterns. Instead, people often engage in a mix of short, frequent trips (e.g., commuting or local errands) and occasional long-distance journeys (e.g., travel, relocation). This aligns well with the structure of Lévy flights, where frequent short moves are punctuated by rare but significant leaps, reflecting a similar distribution of travel distances observed in real-world data.

The similarity between Lévy flights and human mobility has led researchers to adopt Lévy-based models in simulations of population dynamics, epidemiology, urban planning, and telecommunications. These models offer a more realistic representation of how people move across space, which is crucial for designing better mobile network infrastructures, transportation systems, and public health interventions. For instance, in agent-based simulations of mobile network data such as ours, implementing Lévy flights helps mimic natural human behavior, improving the accuracy of synthetic datasets used for validating statistical and machine learning models.

Furthermore, the Lévy flight framework is valuable because it captures the **scale-free** nature of human mobility, implying that movement patterns remain statistically similar regardless of the scale of observation. This fractal-like property is crucial when modelling diverse populations across urban and rural settings. While no model is perfect, Lévy flights strike a balance between simplicity and realism, making them a powerful tool for simulating human movement in both theoretical studies and applied research.

Manhattan mobility pattern is a widely used model in mobility simulations, particularly suited for urban environments where movement is constrained by a grid-like structure, such as city streets laid out in perpendicular directions. Named after the street layout of Manhattan, New York City, this model restricts movement to horizontal and vertical paths, closely mimicking how vehicles and pedestrians navigate through intersections and along city blocks. Unlike random walk models that allow unrestricted movement in any direction, Manhattan mobility emphasizes realistic route constraints and structured directionality, which are critical when modelling urban mobility.

In terms of human mobility, the Manhattan model closely reflects **how individuals typically move within a city**. Whether walking, driving, or using public transport, urban dwellers often follow predefined paths shaped by infrastructure—sidewalks, roads, crosswalks, and traffic lights. These constraints naturally produce movement patterns that align with the structured paths of the Manhattan model. For instance, someone commuting from home to work will generally follow a specific route made up of successive vertical and horizontal segments, stopping at traffic signals or taking detours due to congestion—elements that are implicitly captured in the Manhattan model.

This mobility pattern is particularly useful in **simulations involving vehicular traffic, public transit planning, and mobile network coverage optimization**, as it provides a more spatially accurate representation of real-world travel behaviour in urban areas. Mobile network simulations, for example, benefit from this model by producing more realistic data about how users move between base stations, which directly impacts handover rates, signal strength fluctuations, and cell tower load balancing. The regularity and predictability embedded in the Manhattan model also facilitate testing network performance under different urban mobility scenarios.

Moreover, the Manhattan pattern helps bridge the gap between synthetic simulations and observed **urban human behaviour**. It accounts for the bottlenecks, turning restrictions, and block-wise navigation that characterize city life, making it ideal for agent-based modelling in smart city applications. While it may oversimplify certain dynamic elements like diagonal shortcuts or informal pedestrian paths, it remains a practical and reliable approach for modelling structured, city-constrained human mobility in both research and industry simulations.

Home–work mobility with anchor points is a mobility model that simulates the daily movement patterns of individuals who follow a routine trajectory between home and workplace, with one or more **intermediate stops**, often referred to as *anchor points*. These anchor points represent locations such as grocery stores, gyms, schools, shopping malls, or restaurants—places where people regularly stop as part of their daily schedule. This model reflects the structured yet flexible nature of urban human mobility and has gained prominence in the fields of transportation planning, urban analytics, and mobile network simulations.

Unlike simpler models that assume direct commuting between home and work, the inclusion of anchor points introduces a **more realistic representation of human behaviour**. Real-world mobility is rarely linear or limited to just two fixed points. Most people engage in multi-stop trips throughout the day, whether it's picking up a child from school, stopping for coffee, or running errands after work. By simulating these detours, the home–work–anchor point model captures both **routine and variability**, enriching the spatial and temporal dynamics of synthetic mobility data.

In mobile network simulations, incorporating anchor points significantly enhances the **accuracy of traffic distribution and network load modelling**. Users tend to generate more mobile events in areas of high activity—such as shopping malls or transit hubs—compared to while in transit. Modelling these behaviours allows for better predictions of network demand in specific zones, aiding in optimizing cell tower placement, capacity planning, and service quality improvement. Moreover, in urban flow simulations, understanding these intermediate stops can provide insights into potential contact points and crowd dynamics.

This mobility model also supports **personalized trajectory simulation**, as individuals can be assigned varying schedules and preferences for anchor points, mimicking demographic or behavioural diversity in a population. It reflects the complexity of daily human routines while maintaining a structured flow, which is particularly useful in agent-based models where agents simulate lifelike behaviour. Overall, the home–work–anchor point model offers a nuanced and practical framework for simulating realistic urban mobility, bridging the gap between theoretical movement models and actual human travel patterns.

Home–work mobility with anchor points and Manhattan routing is a composite mobility model that combines two realistic elements of urban movement: the structured daily commute between home and work (with intermediate stops) and movement constrained by a grid-like, city street layout. This model simulates individuals who travel from home to their workplace and back, incorporating **one or more anchor points**—such as a grocery store, gym, or daycare—along the way. What sets this model apart is that all movements occur along **Manhattan-style routes**, meaning agents move only in horizontal and vertical directions, reflecting the layout of real-world city streets.

By integrating **anchor points** with **Manhattan routing**, this model offers an exceptionally realistic approximation of urban human mobility. It captures both the **routine structure of daily life** (commuting with planned detours) and the **spatial constraints imposed by urban infrastructure** (e.g., block-based movement, intersections, and traffic patterns). This makes it highly valuable for simulations in urban planning, traffic forecasting, and telecommunications, as it provides insights into when and where individuals are likely to be—both in terms of time and geography. For instance, such a model allows for detailed modelling of **mobile network load fluctuations**, **pedestrian flow patterns**, or **urban resource allocation** in scenarios involving peak commuting hours and activity hotspots.

2.2 Network interaction modeling

The connection between mobile devices and **Base Transceiver Stations (BTS)** is based on **signal strength** or **signal dominance**, following the method proposed by Tennekes & Gootzen (2021). The model supports both **omnidirectional** and **directional BTSs**. A mobile device attempts to connect to the BTS offering the **strongest or most dominant signal**, based on the selected handover mechanism. For a detailed description of the interaction mobile device – BTS see Salgado et al. (2021).

2.3 Running a simulation

The source code of the simulation tool is available at the following address:

https://github.com/bogdanoancea/simulator

There are 3 ways to run a simulation using this software:

1. Download the source code and compile it.

   There are detailed instructions on how to compile the software in the github repository for Windows, Linux and MacOS.

2. Use an installer for Windows OS:

   Windows installation kit for the mobile network simulator:

   https://github.com/bogdanoancea/simulator/releases/download/1.2.0-kit/mysetup.exe

   A simple GUI for the mobile network simulator can be found at the following address:

   https://github.com/bogdanoancea/simulator-gui/releases/download/1.0/simulator-gui-win32-x64.zip

   The user should extract all files from the archive and run simulator-gui.exe. This GUI helps if users want to avoid running the simulation software from the command line.

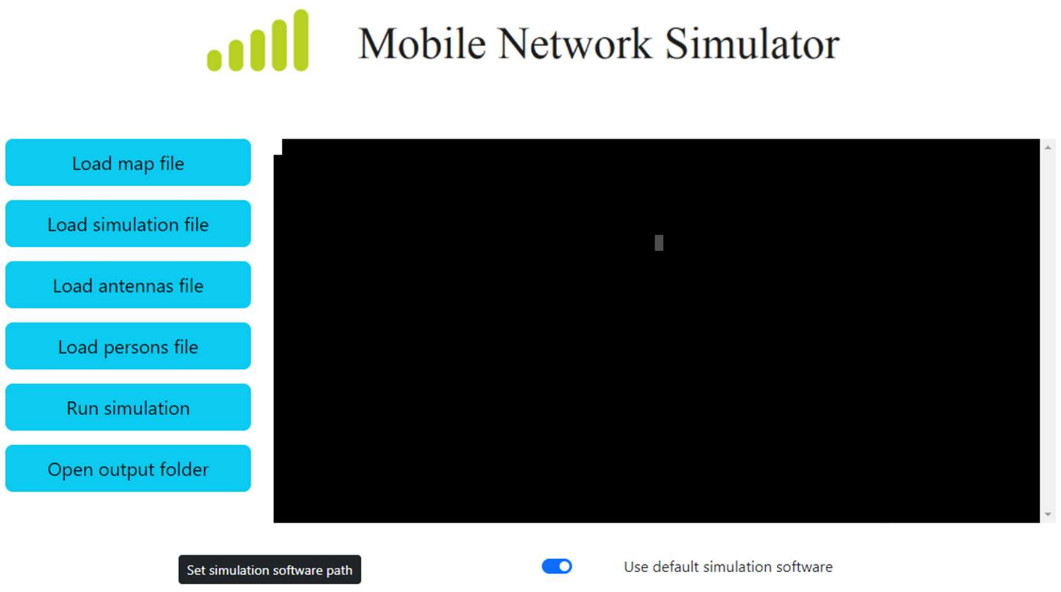


Figure 3. GUI of the simulation tool.

3. Use the docker image found in the github repository.

The command line of the simulator should be:

```
simulator -m map.wkt -s simulation.xml -a antennas.xml -p
persons.xml -pb probabilities.xml -v -o
```

The parameter -o makes the software to compute the localization probabilities using a very simple method. Please note that there are more advanced methods to compute the localization probabilities for each mobile device (see Salgado et al., 2021).

-v means "verbose", it will print some information to the console.

The overall data flow of the simulation is shown in **Figure 4**.

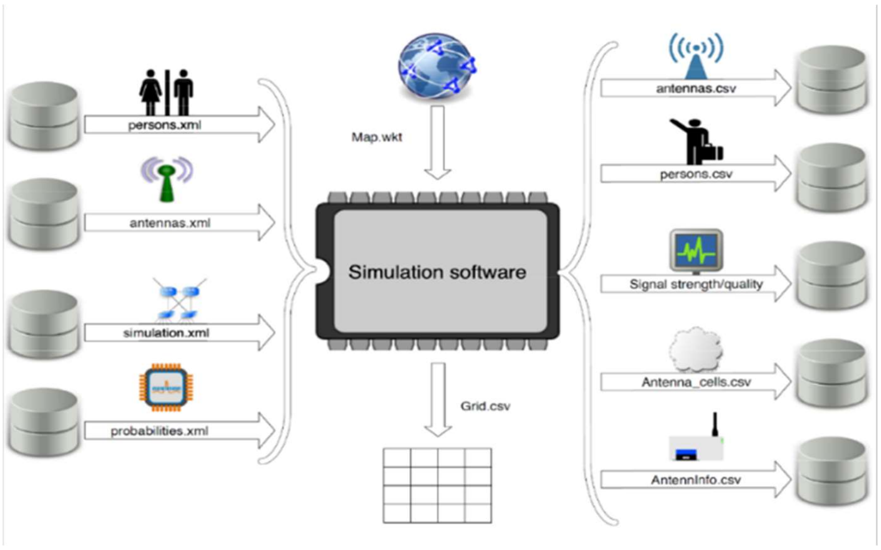


Figure 4. The data flow diagram.

The software requires five main configuration files:

1. **Map file** (in WKT format)
2. **Network configuration** (XML)
3. **General simulation settings** (XML)
4. **Synthetic population configuration** (XML)
5. **Optional file for computing location probabilities** (XML)

Upon running the simulation, the system generates output in several **CSV files**, including:

- **Network configuration file**: cell ID, operator ID, BTS location and technical specs, tile ID.
- **Network events file**: timestamp, cell ID, device ID, network type (e.g., 3G, 4G), event type, Timing Advance value, and device position (x, y, tile ID)—representing the "ground truth".
- **Persons file**: records the location of each individual at every time step, regardless of device possession.

All input/output files are accompanied by **XSD metadata files** that describe file structure and allowable parameter values. These definitions are found here: https://github.com/bogdanoancea/simulator/tree/master/data/input_schema_definition

In the same folder, there is a .jar file that can be used to check an .xml file against its schema (.xsd file). After the user creates your own input files or change the existing ones, he/she can check the correctness of the file xml with the following command:

```
>java -jar schema-check.jar xsd_file_name xml_file_name
```

2.4 The input files

antennas.xml - represents the antennas (base stations) provided by the MNO. Each antenna in this file is described using the following parameters**:**

1. **`mno_name`**
   - **Description**: Name of the Mobile Network Operator (MNO) that owns or operates the antenna.
   - **Example**: `"MNO1"`
2. **`maxconnections`**
   - **Description**: The maximum number of devices that can connect to the antenna simultaneously.
   - **Example**: `56`
3. **`power`**
   - **Description**: Transmission power of the antenna, usually measured in watts (W). Higher values indicate stronger signal strength.
   - **Example**: `10`
4. **`attenuationfactor`**

   - **Description**: Represents the rate at which the signal strength decreases due to obstacles, distance, or interference. A higher value means greater signal loss over distance. There are typical values of the attenuation factor that describe an open field, a city or the interior of a building.
   - **Example**: `3.8`

5. **`type`**
   - **Description**: Specifies the type of antenna. The simulation tool can handle both omnidirectional (meaning they broadcast signals evenly in all directions) or directional antennas (there is a main direction for the propagation of the signal).
   - **Possible values**: `omnidirectional, directional`

6. **`Smin`**
   - **Description**: Minimum signal strength threshold (in dBm) required for a device to connect to the antenna. If the signal is weaker than this value, the device cannot connect. The signal propagation model of the simulator uses either the **signal strength** or the **signal quality** to model the mobile device – antenna interaction.
   - **Example**: `-85`

7. **`Qmin`**
   - **Description**: Minimum quality factor required for a connection.
   - **Example**: `0.3`

8. **`Smid`**
   - **Description**: The midpoint signal strength (in dBm), which is the signal level at which the connection quality starts to degrade noticeably. See the signal propagation model described in [1].
   - **Example**: `-76`

9. **`SSteep`**
   - **Description**: A steepness factor that defines how quickly the signal quality drops as the signal strength decreases. Higher values indicate a sharper decline. See the signal propagation model in [1].
   - **Example**: `0.5`

10. **`x`**

- **Description**: The X-coordinate of the antenna's position in a 2D plane (e.g., a map or simulation grid). This determines its physical location in the network layout.
- **Example**: `500`

11. **`y`**

- **Description**: The Y-coordinate of the antenna's position in a 2D plane. This, along with `x`, defines where the antenna is placed in the network.

- **Example**: `10000`

For directional antennas, some supplementary parameters are needed:

12. **`direction`**

- **Description**: defines the main direction of the radio signal propagation in sexagesimal degrees, measured from north, clockwise;

13. **`tilt`**

- **Description:** defines the tilt of the antenna given also in sexagesimal degrees;

14. **`azim_dB_back`** and **`elev_dB_back`**

- **Description**: defines the difference in the signal strength between the propagation direction and the opposite direction in the azimuthal and elevation planes in dB;

15. **`beam_h`**

- **Description:** specifies the horizontal beam width, which is the angle around the main direction of the antenna where the signal loss is less than or equal to 3 dB;

16. **`beam_v`**

- **Description:** specifies the vertical beam width, which defines the aperture in the vertical plane with a signal loss less than or equal to 3dB.

map.WKT

This is the map of the simulation. It uses the Well Know Text format (Well-known text representation of geometry - Wikipedia). In our example it represents a polygon in a 2D coordinate system and contains a sequence of 10 coordinating (X, Y) pairs. The map should be a closed polygon.

Below is a table representing the coordinates of all 10 points of the polygon used in our example:

Point	X (horizontal)	Y (vertical)
1	0	1000
2	1000	0
3	3000	2000
4	5000	0
5	7500	2000
6	9000	0
7	10000	1000
8	10000	10000
9	0	10000
10	0	1000

Table 1. Polygon Coordinates

In figure 5 we show a graphical representation of the map/polygon.

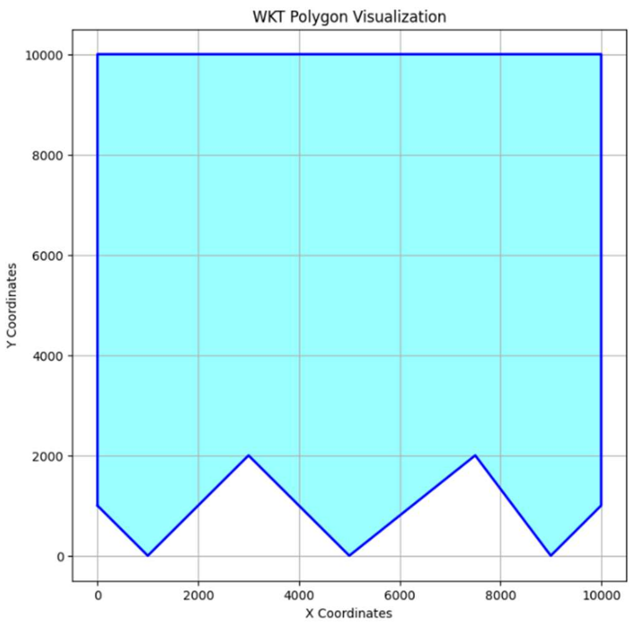


Figure 5. An example map.

persons.xml – this file represents the data needed to generate a number of persons that will be then displaced on the map using a mobility pattern. It has the following parameters/attributes**:**

1. **`num_persons`**
    - **Description**: The total number of people involved in a simulation.
    - **Example**: `400` (indicating 400 individuals).
2. **`age_distribution`**
    - **Description**: Defines the **age distribution** of the population. Each person will have a randomly generated value for the age, according to this distribution. Possible values are: Uniform, TruncatedNormal. For each distribution we have additional parameters. For example, for TruncatedNormal we have:
        - **`mean`**: Average age of the population (e.g., `42` years).
        - **`sd`**: Standard deviation, indicating how much variation exists (e.g. `25` years).
        - **`min`**: Minimum possible age (e.g. `10` years).
        - **`max`**: Maximum possible age (e.g. `87` years).

    The age is ignored in the present version of the simulator, it will be used in future versions.
3. **`male_share`**
    - **Description**: The proportion of **males** in the population.

    Example: `0.48` (meaning **48%** are male and **52%** are female). This parameter is ignored in the present version of the simulator, it will be used in future versions.

4. **`speed_walk_distribution`**
    - **Description**: Persons involved in a simulation can walk or can go by car. If a person walks, then his/her speed is randomly generated according to the distribution defined by this parameter. Possible values: Uniform and Normal. For each distribution we have additional parameters. For example, for Normal we have:
        - **`mean`**: Average speed.
        - **`sd`**: Standard deviation.
5. **`speed_car_distribution`**
    - **Description**: Persons involved in a simulation can walk or can go by car. If a person goes by car, then his/her speed is randomly generated according to the distribution defined by this parameter. Possible values: Uniform and Normal. For each distribution we have additional parameters. For example, for Normal we have:

        - **`mean`**: Average speed.
        - **`sd`**: Standard deviation.
    - **`percent_home`**
    - **Description**: The percentage of the population that **are at their home** location at the start of the simulation.
    - **Example**: `0.9` (meaning **90%** of people are at their home location when the simulation starts).

simulation.xml - This file contains the general configuration settings for the simulation, containing details related to the person mobility pattern, network connectivity and mobile phone usage.

It has the following Parameters:

1. **`start_time`** & **`end_time`**
    - **Description**: Defines the **simulation time range** (likely in seconds or minutes).
    - **Example**:
        - `start_time = 0` → The simulation starts at time 0.
        - `end_time = 2400` → The simulation ends at time 2400.
2. **`time_increment`**
    - **Description**: The **time step** for simulation updates. Smaller values result in finer time resolution.
    - **Example**: `1` (indicating updates occur every **1 time unit**).

Time & Movement Distributions:

3. **`time_stay_distribution`**. The movement of people consist in a sequence of time intervals when they move and time intervals when they stay in the same location. The time interval a person stays in the same position is randomly generated according to the distribution defined by this parameter. Possible value are: Uniform, Normal. For each type of distribution there are additional parameters. For example, for the Normal distribution we have
        - mean
        - sd
4. **`interval_between_stays_distribution.`** See the above params for an explanation.

Mobile Network Operator (MNO) Settings:

5. **`mno`**
    - **Description**: Defines the **mobile network operator** and phone usage probability.
    - **Sub-elements:**
        - `mno_name`: `"MNO1"` (identifies the network operator).
        - `prob_mobile_phone`: `0.35` (this is the penetration rate of the MNO).
6. **`prob_sec_mobile_phone`**
    - **Description**: Probability that a person has a **secondary mobile phone** (e.g., work & personal phones).
    - **Example**: `0.15` (15% of people have a second phone).

Movement Pattern & Locations:

7. **`movement_pattern`**. Defines the mobility pattern of the persons during a simulation.

   Currently the following mobility patterns are implemented: random_walk_closed_map (a complety random walk inside the map boundaries, random_walk_closed_map_drift (there is a bias in the movement, i.e. at each random step the angle of the displacement is bias toward a predefined angle), levy_flight (it implements the well-known Levy flight mobility pattern which is proved to be very similar to human mobility pattern), Manhattan (the person are randomly moving according to the Manhattan pattern, i.e. they move on a rectangular grid), home_work (it implements the home-work scenario: people stay in the same location for a period of time (home location) then they move toward another location (work location) where they stay for a period of time then they move back home, possibly stopping in some "anchor points' for a period of time.) and home_work_manhattan (there difference is that people move on a "Manhattan" grid while in a home_work scenario they move in an approximately straight line). Each mobility pattern has additional parameters. For example, home_work_manhattan defines the additional parameters:
    - **Home Locations** (`home`) with the following inner parameters
        - `x, y x_sd, y_sd`: the home locations are generated inside an ellipse defined by these params.
    - **Work Locations** (`work`) – the same inner params like home.
    - **Anchor Points** *(optional locations people visit, e.g., shops, parks)*
        - The same inner params like home and work
    - **Manhattan Grid Definition:** `x_step, y_step, x_origin, y_origin`.
    - **Home_work** scenarios has also the following additional parameters:
        - **`percent_time_home`**: `0.5` (50% of time spent at home).
        - **`percent_time_work`**: `0.3` (30% at work).

        - **`percent_time_anchor_point`**: `0.1` (10% at anchor points).
        - **`percent_time_travel`**: `0.03` (3% of time in transit).
        - **`prob_anchor_point`**: `0.8` (80% probability of visiting an anchor point).

Network & Connection Settings:

9. **`connection_type`**
    - **Description**: Specifies the signal propagation model used to represent the connection between the mobile device and the antenna.
    - **Possible values**: `strength, quality`
10. **`event_type`**
    - **Description**: Defines the type of recorded network events.
    - **Example**: `cellIDTA, cellID (the first saves also the Time Advancing parameter)`
11. **`conn_threshold`**
    - **Description**: The **signal strength /quality threshold** (dBm) for connection. A signal weaker than this value will be considered only noise.
    - **Example**: `-85` (devices only connect if the signal is stronger than **-85 dBm**).

The Output files specifications

12. **`grid_file`**
    - **Description**: Specifies the name of the **grid definition file.** We defined this grid to simplify the computation of each device's geolocation. Specifically, our statistical tool calculates the probability that a device is located at the centre of a grid tile. The finer the grid, the more accurate the geolocation results will be. The grid file is overlaid onto the map.
    - **Example**: `"grid.csv"` (contains the grid layout).
13. **`grid_dim_tile_x`** & **`grid_dim_tile_y`**
    - **Description**: Defines **grid tile size** on OX and OY.
    - **Example**: `250x250` units per tile.
14. **`persons_file`**
    - **Description**: The file where all details about the positions of the persons during the simulation are saved.
    - **Example**: `"persons.csv"`.

15. **`antennas_file`**
   - **Description**: File containing **antenna positions and specifications**.
   - **Example**: `"antennas.csv"`.

Randomization & Output Settings:

16. **`random_seed`**
   - **Description**: Seed for **random number generation** (ensures reproducibility).
   - **Example**: `12` (fixed random seed for consistency).
17. **`output_dir`**
   - **Description**: Specifies the **output directory** for simulation results.
   - **Example**: `"output2"` (simulation results are stored in the `"output2"` folder).

2.5 The output files

`grid.csv` - this file defines the reference grid for the simulation, specifying how the area is divided into tiles. It contains the following parameters:

- **`Origin X / Origin Y`**: The starting coordinates of the grid (0,0 in this case).
- **`X Tile Dim / Y Tile Dim`**: The size of each tile in the X and Y directions.
- **`No Tiles X / No Tiles Y`**: The number of tiles along the X and Y axes

We defined this grid to simplify the computation of each device's geolocation. The grid file is overlaid onto the map.

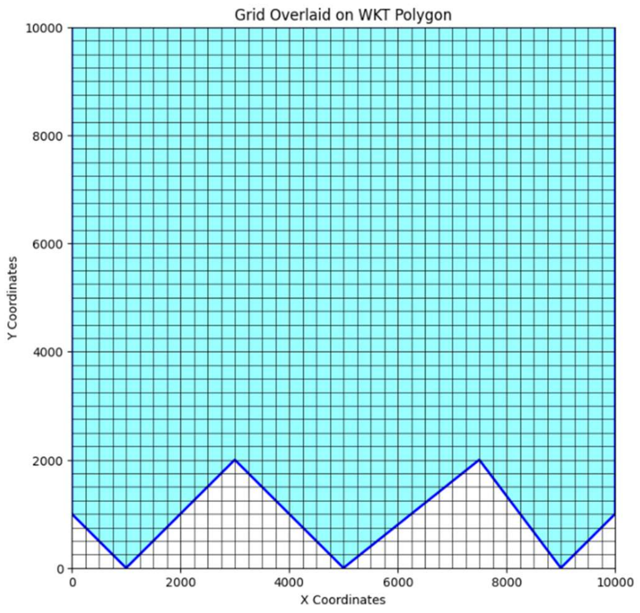


Figure 6. View of the grid

In our example the grid file divides the map is 40 tiles of 250 units per row and column.

The **antenna dataset (atennas.csv)** contains information about network antennas and their properties. The columns are as follows:

1. `t`: Time step in the simulation (0 in this case).
2. `Antenna ID`: Unique identifier for each antenna.
3. `x / y`: Coordinates of the antenna in the simulation space.
4. `MNO ID`: Identifier for the mobile network operator.
5. `mno_name`: Name of the mobile network operator (e.g., MNO1).
6. `maxconnection`: Maximum number of simultaneous connections the antenna can handle.
7. `power`: Transmission power of the antenna.
8. `attenuationfactor`: Defines how signal strength decreases with distance.
9. `type`: The type of antenna (e.g., omnidirectional).
10. `Smin`: Minimum signal strength threshold for connectivity.
11. `Qmin`: Minimum signal quality required for connectivity.
12. `Smid`: Signal strength at which quality is mid-range.
13. `SSteep`: Determines how quickly signal strength drops.
14. `tilt`: The tilt angle of the antenna.
15. `azim_dB_back / elev_dB_back`: Azimuth and elevation attenuation values (not filled in this dataset).
16. `beam_h / beam_v`: Horizontal and vertical beamwidth (not provided in this dataset).
17. `direction`: Orientation of the antenna (not filled in this dataset).
18. `z`: The height or elevation of the antenna (0 in this dataset).
19. `Tile ID`: The grid tile where the antenna is located.

The **persons dataset (persons.csv)** contains details about individuals in the simulation, including their locations and mobile devices. The columns are as follows:

1. `t`: Time step in the simulation (0 in this case).
2. `Person ID`: Unique identifier for each individual.
3. `x / y`: The coordinates of the person's location at the given time step.
4. `Tile ID`: The grid tile where the person is located.

5. **`Mobile Phone(s) ID`**: The ID(s) of the mobile phone(s) carried by the person. If multiple phones are carried, they are separated by a hyphen (e.g., "601-600"). In its current form, a user can either carry no phone, a single phone or two phones.

This file helps track individuals' positions and their mobile phone usage, which is essential for simulating network connectivity and movement behaviour.

2.6 Examples of configuration data sets

Some example configuration files needed to run different simulations are included at the following address: https://github.com/bogdanoancea/simulator/tree/master/data

Set 1

https://github.com/bogdanoancea/simulator/tree/master/data/dataset1

Short description: 100 persons, 2 MNOs, 35% penetration rate for each MNO, 0.15 probability for a person to have 2 mobile devices, “Random walk with drift“ mobility pattern, timeframe: 900 steps with an increment of 1.

Set 2

https://github.com/bogdanoancea/simulator/tree/master/data/dataset2

Short description: 50 persons, 2 MNOs, 35% penetration rate for each MNO, 0.15 probability for a person to have 2 mobile devices, “Random walk“ mobility pattern, timeframe: 100 steps with an increment of 1.

Set 3

https://github.com/bogdanoancea/simulator/tree/master/data/dataset3

Short description: 50 persons, 1 MNO, 35% penetration rate for the MNO, 0.15 probability for a person to have 2 mobile devices, “Levy flights“ mobility pattern, timeframe: 200 steps with an increment of 1.

Set 4

https://github.com/bogdanoancea/simulator/tree/master/data/dataset4

Short description: 400 persons, 1 MNO, 35% penetration rate for the MNO, 0.15 probability for a person to have 2 mobile devices, “Random walk with drift“ mobility pattern, timeframe: 200 steps with an increment of 1.

Set 5

https://github.com/bogdanoancea/simulator/tree/master/data/dataset5

Short description: 400 persons, 1 MNO, 35% penetration rate for the MNO, 0.15 probability for a person to have 2 mobile devices, “Home-Work“ mobility pattern, timeframe: 2400 steps with an increment of 1.

Set 6

https://github.com/bogdanoancea/simulator/tree/master/data/dataset6

Short description: 400 persons, 1 MNO, 35% penetration rate for the MNO, 0.15 probability for a person to have 2 mobile devices, “Manhattan“ mobility pattern, timeframe: 2400 steps with an increment of 1.

Set 7

https://github.com/bogdanoancea/simulator/tree/master/data/dataset7

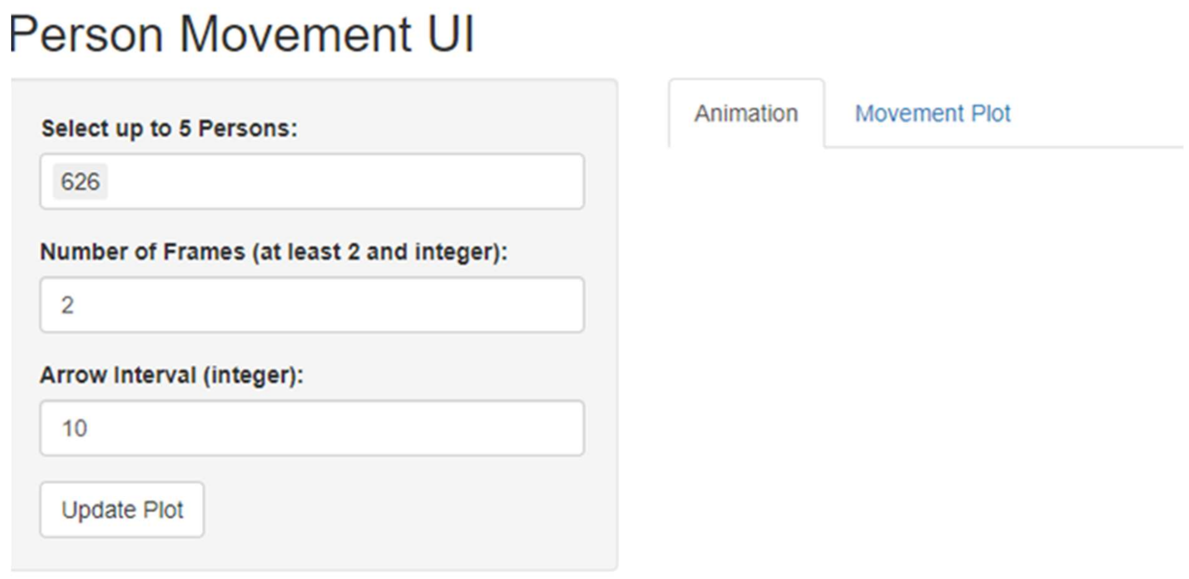
Short description: 400 persons, 1 MNO, 35% penetration rate for the MNO, 0.15 probability for a person to have 2 mobile devices, “Home-Work with Manhattan grid“ mobility pattern, timeframe: 2400 steps with an increment of 1.

3. UI for Person Movement

For the purposes of comfortably showing how the datasets can be used to simulate the movement of a person through a map/grid, a very minimalistic UI has been built using R. The repository containing the script and the necessary datasets (persons.csv, antennas.csv, antenna.png, grid.csv, map.wkt) can be accessed for free at:

https://github.com/MerelyaHousePlant/Person-Movement-UI.

Once the repository is either cloned or downloaded, the user can open the animation_script.R file in Rstudio and select the Run App button. After all necessary packages are installed and loaded, the UI will open, however, while running the app for the first time the user might need to pay attention to the console, as it might ask for user input regarding how to handle the updating/installing of the packages.


Person Movement UI

Select up to 5 Persons:

626

Number of Frames (at least 2 and integer):

2

Arrow Interval (integer):

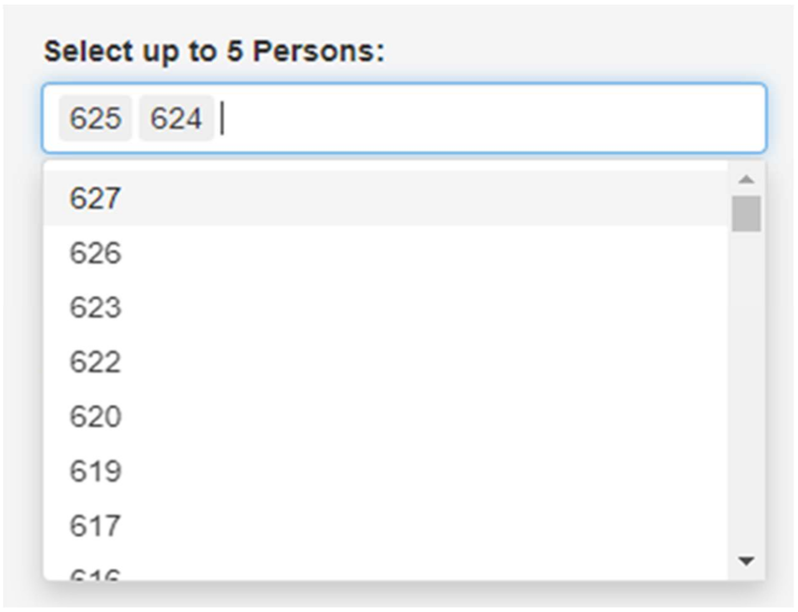
10

Figure 7. Input Parameters for UI

3.1 Parameters

The very simplistic UI that I have made takes 3 inputs from the user:

1. **Person ID**: It requires the ID of a person the user wants to plot, at least one value needs to be selected, up to a maximum of 5, in order to be able to generate or update the plot. The user must select the desired Person IDs from a dropdown list containing all the valid IDs in the dataset.


Select up to 5 Persons:

625 624

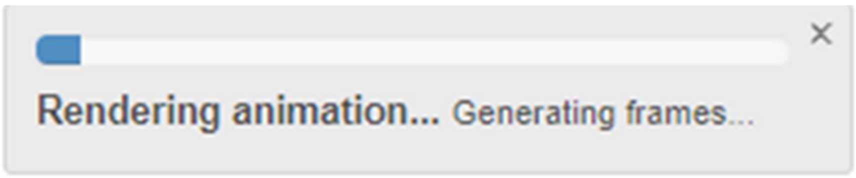
627
626
623
622
620
619
617

Figure 8. Argument selection for the Person ID parameter

2. **Nframes**: The argument represents the number of frames the user desires to insert in the animation, the higher the number the smoother the quality of the animation will be. For instance, if a low number of frames is given the resulting animation might look very rushed.

It should be an integer number higher or equal to 2. It should be noted that it takes roughly 1-2 seconds to render each frame, so passing a high number of frames might take a large amount of time.

The default value for the variable is 2 and it will revert back to it if the user attempts to enter an invalid number or character.



Rendering animation... Generating frames...

Figure 9. Progress Bar for rendering animation

A loading bar will show the current progress while the frames are rendering with pitch perfect accuracy.

3. **Arrow Interval**: The purpose of the variable is only visible in the „Person Movement" plot, its role is to determine how often arrows linking the position of a specific user at a given time are drawn. For the default value of 10, it will check to see if the user moved every 10 timeslots and if they did indeed move, an arrow will be drawn from the previous location to the new one.

It needs to be given an integer value higher than 1, in order to properly display the arrows. Similarly, to nframes, trying to pass an invalid value will immediately revert back to the default value of 10.

3.2 The plots

After the animation is done rendering, the animated plot will display on the screen. The shape of each individual user changes based on the Mobile Phone(s) ID variable, if its value is null, the user will be represented by a circle, if the user has a phone they will instead be represented by a triangle and if they have two phones they will be represented by a square.

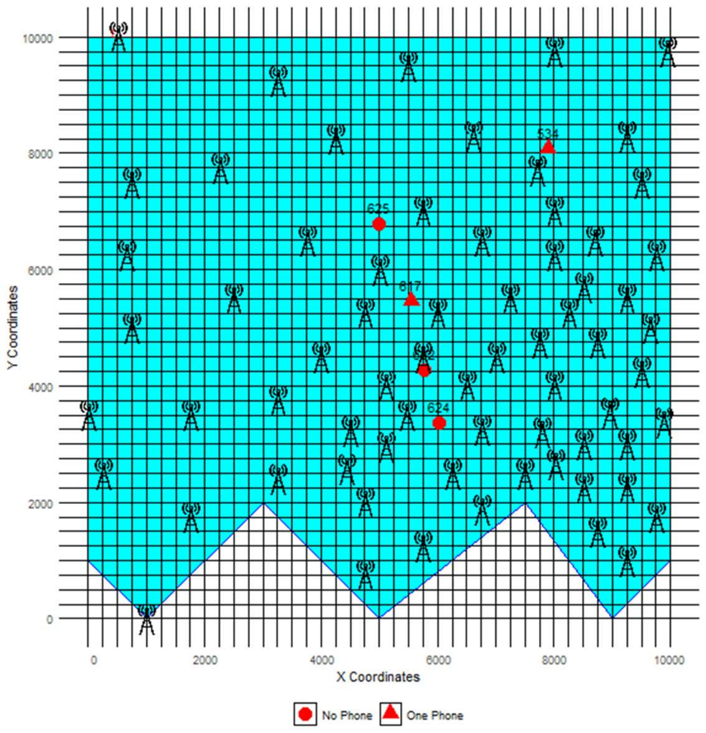


Figure 10: Animation of person movement though the grid

The user can directly save the animation on their computer or simply copy and paste it by right clicking on it.

The user can alternate between seeing the animation plot or the person movement plot by clicking on these tab panels:

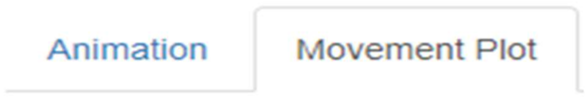


Figure 10. Tabs of distinct plots

It should be noted that the UI will need a few seconds to generate/update the Movement Plot for the first time.

The movement plot highlights the path each selected user took from the first timeslot of the simulated data, up until the last, assigning a unique colour to each user and setting up specific marks, which display the starting and ending point of that user's movement. In the current version, a green triangle represents the starting point and a red circle the end point, as for the users who during the end of the simulation, return to the same spot they started at, a purple cross mark is displayed instead.

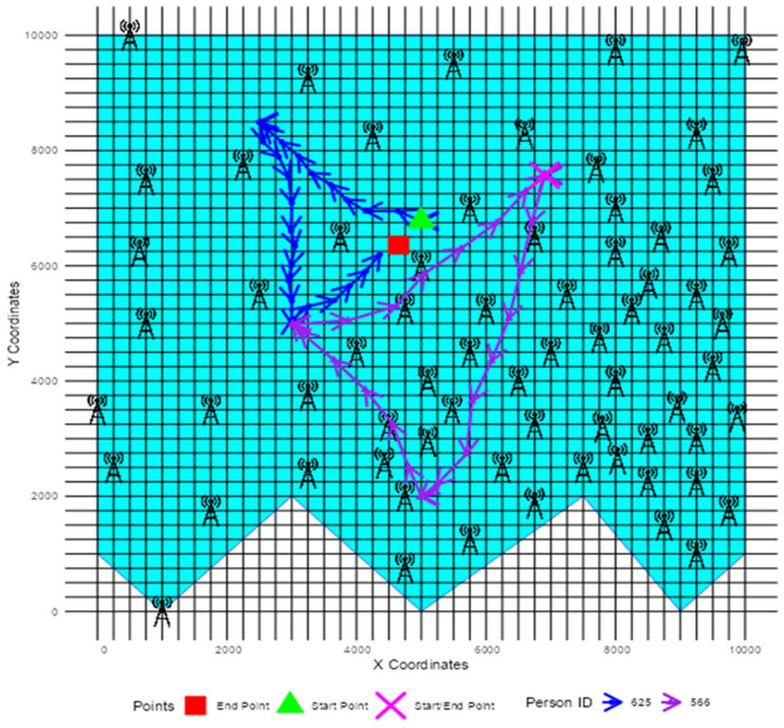


Figure 11. Visualization of the path the selected users took from start to finish

4. Further Post-Processing

The main purpose of this post processing section was the to simulate the functionality of certain sensors that could detect that number of devices that connected to a specific antenna at a given time or the number of people who passed through a specific tile from the grid.

In order to run the post-processing script, the following input files are required: (persons.csv, antennas.csv, AntennaInfo_MNO_MNO1.csv, grid.csv).

The script and all the required files can be accessed for free from a git repository through this link:

https://github.com/MerelyaHousePlant/MNO_Post-Processing.

4.1 General constraints and objectives

Denote all the grids in the map by $U = \{i_1,\ldots,i_{1600}\}$, as well as by (h_{i1}, h_{i2}), $1 \leq h_{i1}, h_{i2} \leq 40$. Denote by (c_{k1}, c_{k2}) the coordinates of each antenna $k = 1,\ldots,70$, where $0 \leq c_{k1}, c_{k2} \leq$

10000. For each antenna k, denote by $i(k)$ the grid in which the antenna is placed; denote by s_A all these *(antenna) grids*, $s_A \subset U$.

MNO counts: the description applies to each MNO separately, if there are multiple MNOs in the simulation.

Let $i(t;e)$ be the grid $i(k)$ of the antenna k of a signal at time point t, which is associated with an entire movement e simulated. For each movement e, let $t_1(e)$ be the first signal time point, and so on till $T(e)$ at last, yielding the $T(e)$ grids as

$\{i(t;e) : t = t_1(e) \leq t_2(e) \leq \cdots \leq T(e)\}$.

Divide the sequence above into segments of identical grid $i(t;e)$, and extract all the distinct *signal-segment grids* of e ordered over time, denoted by

$\{i_1(e), i_2(e), \ldots, i_{M_e}(e)\}$

where $i_g(e) \neq i_{g+1}(e)$ for any $g = 1, \ldots, M_e - 1$ and M_e refers to the last identical signal-segment of e. For each grid $i \in U$, let the *MNO count* of movements passing over gird i in the whole simulation be

$$x_i = \sum_{e:M_e>0} \sum_{g=1}^{Me} \mathrm{I}(\mathrm{i_g}(\mathrm{e}) = i) .$$

Target counts: for each grid $i \in U$, let y_i be the number of movements passing over it in the whole simulation, including those movements without devices or signals. This can be obtained similarly to x_i above, according to how $i(t;e)$ can be calculated from the 'displacements' and 'stops' of a movement e.

Dataset-I For each antenna $k = 1, \ldots, 70$, include the attributes

$\{k, (c_{k1}, c_{k2}), i(k), (h_{i1}, h_{i2}), x_i\}$

where (h_{i1}, h_{i2}) and x_i refer to the grid $i(k)$ of antenna k. There may be multiple counts x_i if there are more than one MNO in the simulation.

Dataset-II For each grid $i \in U$, include the attributes

$\{i, (c_{i1}, c_{i2}), (h_{i1}, h_{i2}), y_i, \mathrm{I}(i \in s_A)\}$

where (c_{i1}, c_{i2}) are the coordinates of the centre of the grid i.

4.2 Resulting Datasets

Antenna.ID	x	y	Tile.ID	grid_row	grid_col	device_count
1	500	10000	1561	41	3	0
2	8000	9750	1551	40	33	0
3	5750	7000	1102	29	24	3
4	9750	6250	998	26	40	40
5	7250	5500	868	23	30	9
6	4000	4500	695	19	17	12
7	6500	4000	625	17	27	38
10	8000	4000	631	17	33	61

Table 2. A small sample of dataset I

Dataset I contains information regarding the position of each antenna on the map, grid, the tile they are situated on and the number of devices that connected to them, starting from the first timeslot and up until the last.

TileId	x	y	grid_row	grid_col	person_count
1330	2600	8254.607	34	11	229
1290	2691.898	8223.173	33	11	201
1329	2451.028	8417.697	34	10	195
1250	2700	7800	32	11	182
631	7803.038	4000	17	32	173
1091	2841.657	6936.137	28	12	172
340	5248.229	2050	9	21	164
1210	2750	7650	31	12	162
616	4050	4000	17	17	158
1051	2955.938	6750	28	12	150

Table 3. A small sample of dataset II

Dataset II contains information regarding the position of each individual tile on the map, grid and the total number of people who passed through that specific tile, starting from the first timeslot and up until the last.

Additionally, two plots showing the distribution of the device count and person count respectively, are displayed below.

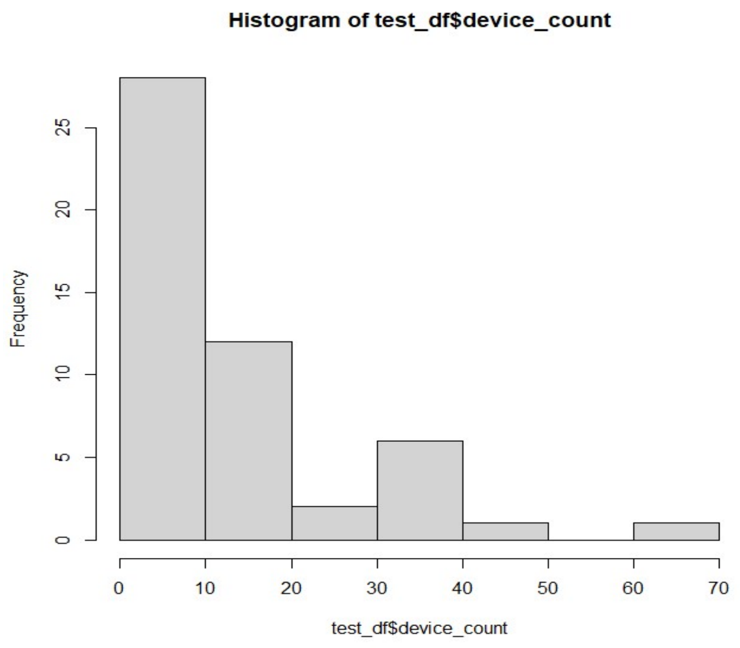


Figure 12. Distribution of the counts of devices connected to antennas

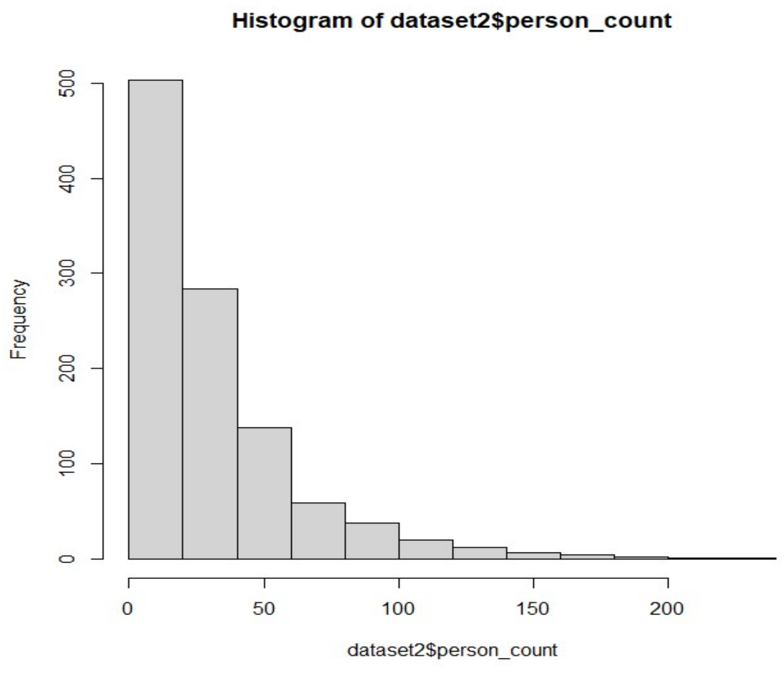


Figure 13. Distribution of the person count found on each tile of the grid

5. References

Bordin, M. V. (2017). *A Call Detail Record (CDR) generator*. https://github.com/mayconbordin/cdr-gen

Horni, A., Nagel, K., & Axhausen, K.W. (2016) *The Multi-Agent Transport Simulation MATSim*. Ubiquity Press, London

Krajzewicz, D., Erdmann,J., Behrisch, M., Bieker, L. (2012). "Recent Development and Applications of SUMO -Simulation of Urban Mobility". *Journal On Advances in Systems and Measurements*, 5 (3&4), 128–138.

Salgado, D., and Sanguiao, L., Oancea, B, Barragán, S., Necula, M. (2021) "An end-to-end statistical process with mobile network data for official statistics". *EPJ Data Science*, 2021, 10(20), DOI: 10.1140/epjds/s13688-021-00275-w.

Tennekes, M., Gootzen, Y.A. (2021) "A Bayesian approach to location estimation of mobile devices from mobile network operator data", https://arxiv.org/pdf/2110.00439.pdf

Tetcos (2019). *NetSim User Manual*. https://www.tetcos.com/downloads/v12/NetSim_User_Manual.pdf

Zhen, L., Hongji, Y. (2012) *Unlocking the Power of OPNET Modeler*. Cambridge University Press, New York.

II. Trips

Statistics Norway, Norway

We refer to Trafikanalys for the application to the Swedish National Travel Survey at:

https://www.trafa.se/en/transportation-trends/travel-survey/combined-mobile-network-data-and-survey-data-describe-travel-patterns-15129/

The bespoken implementation was performed in the R environment (R Core Team, 2025).

All the analyses illustrated in Chapter 10 were performed in the R environment (R Core Team, 2025). To apply the test described in Section 10.1, it was developed an ad hoc R function `audit.test.R`.

To apply the linear-prediction, transfer-learning and mixed ensemble estimator described in Section 10.2 and illustrated in Section 10.3, it was developed an ad hoc R function `pest.R`.

Both the functions can be downloaded from the following repository:

https://github.com/mno-minds/MNO-MINDS/tree/main/R_code_trip_stats

References

R Core Team (2025) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

III. Inbound Tourism

INE,Spain

1. Introduction

This chapter is about the application scenario with a proposal of integration process from different data sources related to inbound tourism statistics in Spain. In the official statistics, data traditionally come from sample surveys and passenger traffic, while experimental statistic is based on Mobile Network Operator (MNO) data. Inbound tourism is defined according to Eurostat standards as trips by non-residents into the country, measured through various indicators such as trip counts, destinations, and overnight stays. The data sources include airport, port, train, and road border crossings, each managed by different organizations like AENA, RENFE, and the General Directorate of Traffic. Surveys complement these sources by gathering detailed information through face-to-face interviews and special capacity counts, especially for road travellers where passenger counts are not directly available. The different data sources are explained in section 9.1 of deliverable D3.2.

Since 2020, Spain's National Statistical Institute (INE) has collaborated with MNOs to integrate mobile network data into tourism statistics, allowing for monthly aggregated trip counts and related metrics. Advanced algorithms transform raw mobile totals into usable tourism data, which are then combined with traditional survey estimates through techniques like transfer learning to improve accuracy. However, differences remain between MNO data and survey results, including seasonal biases, prompting ongoing efforts to refine the methodology and strengthen cooperation with MNOs to enhance future inbound tourism statistics.

Special focus on implementation is put in this document with specific explanations of each process. In the following, the algorithms are explained step by step to reach reproducibility of the developed processes.

2. Pre-processed implementation

In this section, it is explained in detail the procedure done as pre-processed data with the aim of correcting the bias in the MNO data. The procedure explanation and justification can be read in section 9.2.1 of Deliverable 3.2.

2.1 FRONTUR survey vs. MNO estimates

When aggregating our multi-source data, a systematic difference between survey data and MNO counts was observed. This bias was problematic for data integration, so it was decided to use time series analysis to correct the MNO data. Each island group featured a different seasonal behavior, so a slightly different solution had to be applied in each case, which generally involved using past or future data to correct the total mobile counts. Additionally, because of covid-19, a large portion of our data was unusable, so we had to manually add a filter date in each case. That date was determined by visually examining the time series and finding the approximate month where the data returned to normal levels. The following algorithms roughly describe the bias correction for both cases:

Algorithm 1 Logarithmic bias correction for the Balearic Islands

```
1: procedure LogMonthlyBiasBalearic(data, filter_date = 2022/07/01)
2:     log_bias ← log(totals_survey) − log(totals_mobiles)    ▷ Calculate log difference
3:     for $i = 1$ to $N_{\text{dates}}$ do    ▷ Calculate the differences lagged by 12 months.
4:         $\text{dif\_12\_log}_i \leftarrow \text{log\_bias}_i - \text{log\_bias}_{i-12}$
5:     end for
6:     $\widehat{\sigma^2} \leftarrow$ var(dif_12_log where date ≥ filter_date)    ▷ Ignoring all missing values
7:     for $i = 1$ to $N_{\text{dates}}$ do
8:         $\text{totals\_mobiles\_corrected}_i \leftarrow \text{totals\_mobiles}_i \cdot \frac{\text{totals\_survey}_{i-12}}{\text{totals\_mobiles}_{i-12}} \cdot exp\{-\widehat{\sigma^2}/2\}$
9:     end for
10:    return totals_mobiles_corrected
11: end procedure
```

Algorithm 2 Logarithmic bias correction for the Canary Islands

```
1: procedure LogMonthlyBiasCanary(data, filter_date = 2022/01/01)
2:     log_bias ← log(totals_survey) − log(totals_mobiles)    ▷ Calculate log difference
3:     $\widehat{\sigma^2} \leftarrow$ var(log_bias where date ≥ filter_date)    ▷ Ignoring all missing values
4:     unique_dates ← dates where date ≥ filter_date
5:     $\widehat{\mu} \leftarrow 0$    ▷ Define a new column
6:     for $t$ in unique_dates do
7:         $\widehat{\mu}_t \leftarrow mean$(log_bias where date >= filter_date and date < $t$)    ▷ Ignoring all missing values
8:     end for
9:     for $i = 1$ to $N_{\text{dates}}$ do
10:        $\text{totals\_mobiles\_corrected}_i \leftarrow \text{totals\_mobiles}_i \cdot exp\{\widehat{\mu}_i - \widehat{\sigma^2}/2\}$
11:    end for
12:    return totals_mobiles_corrected
13: end procedure
```

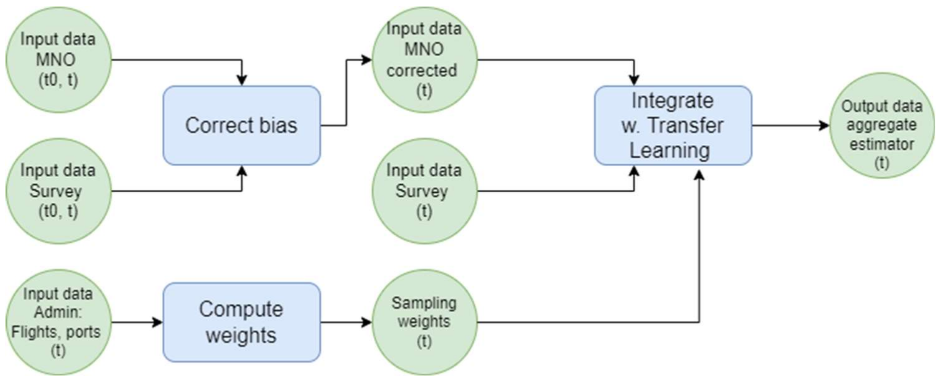
It is assumed that *data* contains all the merged multi-source variables (as different columns) used in both procedures, such as *totals_survey* or *dates*.

All the preprocessing was implemented in R (R Core Team , 2025), using the **arrow** (Richardson et al, 2024), **dplyr** (Wickham et al, 2023) and **data.table** (Barrett et al, 2024) packages for handling the data, and **ggplot2** (Wickham, 2016) for plotting and examining results. Regarding integration, the packages **minpack.lm** (Elzhov et al, 2023) and **Rcpp** (Eddelbuettel et al, 2024) were used. A full pipeline can be seen in the following section.

3. Integration implementation

In the integration process there are two possibilities, a more thorough methodology explanation can be found in sections 9.2.2 (combined estimator with transfer learning) and 9.2.3 (combined estimator with state space models) of the MNO-MINDS deliverable D3.2. The implementation of both options is explained in the following sections 3.1 and 3.2.

3.1 Transfer learning by cross-validation



Input data MNO (t0, t)
Input data Survey (t0, t)
Input data Admin: Flights, ports (t)
Correct bias
Compute weights
Input data MNO corrected (t)
Input data Survey (t)
Sampling weights (t)
Integrate w. Transfer Learning
Output data aggregate estimator (t)

Figure 14. Diagram of the complete integration process using transfer learning techniques with all the different data sources

The following pipeline briefly summarizes the data integration algorithm. The key element is the *CombinedEstimatorWeighted* function, which provides an estimated proportion when given a tourist vector, a weights vector, a population size and an initial estimated proportion. The tourist vector contains either a zero or a one in each of its components, indicating that the interviewed person is a tourist when its value is a one. The weights vector assigns an "importance" to each tourist vector element and should be a constant value within its strata. As seen in the following algorithm, the population size is obtained by adding up all the weights for any given date (alternatively, the combined admin data could be used to calculate the population size), and the initial estimated proportion is calculated as the division of the total corrected mobile counts by the total population.

Algorithm 3 Data integration function (weighted)

```
1: procedure DataIntegration(destination, s_data, f_data, p_data, m_data)
2:     switch $destination$ do
3:         case Balearic islands
4:             s_data ← s_data(data where survey_points in Balearic islands)
5:             ...                                    ▷ Repeat filtering for all data sets
6:             $data$ ← merge(s_data, f_data, p_data, m_data; by date)
7:             totals_mobiles_corrected ← LogMonthlyBiasBalearic(data)
8:         case Canary islands
9:             s_data ← s_data(data where survey_points in Canary islands)
10:            ...                                    ▷ Repeat filtering for all data sets
11:            $data$ ← merge(s_data, f_data, p_data, m_data; by date)
12:            totals_mobiles_corrected ← LogMonthlyBiasCanary(data)
13:    N_total ← $sum$(survey_weights by date)        ▷ Or use total admin counts by date
14:    $p_0 \leftarrow \frac{\text{totals\_mobiles\_corrected}}{\text{N\_total}}$
15:    for $i = 1$ to $N_{\text{dates}}$ do
16:        estimated_prop$_i$ ← CombinedEstimatorWeighted(tourist$_i$, weights$_i$, N_total$_i$, $p_{0i}$)
17:    end for
18:    return estimated_prop
19: end procedure
```

This procedure requires a destination to be specified (as the preprocessing changes depending on it) as well as the different multi-source datasets. The prefixes s, f, p and m mean survey, flights, ports and mobile, respectively.

Algorithm 4 Transfer learning from mobile phone to survey

```
1: procedure CombinedEstimatorWeighted(tourist, weights, N_total, $p_0$)
2:     for $i = 1$ to $N_{\text{tourist}}$ do        ▷ Get normalized inclusion probabilities from weights
3:         iprobs$_i$ ← $\frac{\text{N\_total}}{N_{\text{tourist}}\text{weights}_i}$
4:     end for
5:     $\hat{\gamma}$ ← nls.lm($\lambda(\gamma)$ xValRes($\gamma$, tourist, iprobs, $p_0$), par = 0, lower = 0)
6:     ...                                        ▷ Call Levenberg-Marquardt optimizer
7:     $p_1 \leftarrow \frac{1}{\hat{\gamma}+1}(\text{mean}(\frac{\text{tourist}}{\text{iprobs}}) + \hat{\gamma} p_0)$        ▷ Transfer learning estimation
8:     return $p_1$, $\hat{\gamma}$
9: end procedure
```

Algorithm 5 Leave one out cross-validation residuals to optimize

```
1: procedure XVALRES(tourist, iprobs, N_total, $p_0$)
2:     $m \leftarrow \mathrm{sum}(\frac{\mathrm{tourist}}{\mathrm{iprobs}\cdot(\mathrm{tourist\_total}-1)})$
3:     ...                ▷ This way we get the leave one out mean just substracting one element
4:     for $i = 1$ to $N_{\mathrm{tourist}}$ do                ▷ Get weighted leave one out residuals
5:         $\mathrm{res}_i \leftarrow \frac{1}{\sqrt{\mathrm{iprobs}}}(\mathrm{tourist} - \frac{\mathrm{iprobs}}{\hat{\gamma}+1} \cdot (m - \frac{\mathrm{tourist}}{\mathrm{iprobs}\cdot(\mathrm{tourist\_total}-1)} + \hat{\gamma} p_0))$
6:     end for
7:     return res                ▷ Residuals for least squares optimization
8: end procedure
```

These two final procedures can be summarized as: i) select γ by leave-one-out cross-validation and ii) estimate the combined estimator with that γ. The optimizer is `nls.lm()` function from **minpack.lm** R package, which uses Levenberg-Marquardt algorithm.

One of the major drawbacks of this method is the difficulty to properly estimate the result's variance. As it relies entirely on a correct survey weight (and strata) definition.

3.2 Transfer learning using state-space models

For this approach we used SSMMATLAB (Gómez, 2015) in Octave (Eaton et al, 2019). The steps are:

- Generate the matrices of the model.
- Obtain the maximum-likelihood parameters. This is done with *marqdt* function, which implements Levenberg-Marquardt algorithm. It requires a function to optimize.
- For Kalman smoothing, initial state is generated with *incossm*. Then *smoothgen* function is used.

Variances are, of course, included in the parameters of the model. Now, for the target function to optimize, it is also quite simple:

- Generate initial state with *incossm.* Since it depends on the parameters, this goes inside the optimization loop.
- Call *scakfle2* to get the likelihood as a vector of residuals to optimize.

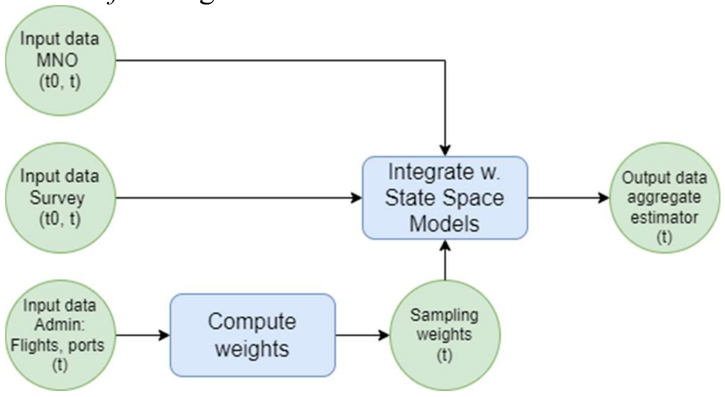

Input data MNO (t0, t)
Input data Survey (t0, t)
Input data Admin: Flights, ports (t)
Compute weights
Sampling weights (t)
Integrate w. State Space Models
Output data aggregate estimator (t)

Figure 15. Diagram of the complete integration process using state-space models with all the different data sources

4. References

Barrett T, Dowle M, Srinivasan A, Gorecki J, Chirico M, Hocking T (2024). _data.table: Extension of `data.frame`_. R package version 1.15.4, https://CRAN.R-project.org/package=data.table

Eddelbuettel D, Francois R, Allaire J, Ushey K, Kou Q, Russell N, Ucar I, Bates D, Chambers J (2024). *Rcpp: Seamless R and C++ Integration*. R package version 1.0.13-1, https://CRAN.R-project.org/package=Rcpp.

Eaton, JW, Bateman, D., Hauberg, S., Wehbring, R. (2019). *GNU Octave version 5.2.0 manual: a high-level interactive language for numerical computations*. https://www.gnu.org/software/octave/doc/v5.2.0/

Elzhov TV, Mullen KM, Spiess A, Bolker B (2023) *minpack.lm: R Interface to the Levenberg-Marquardt. Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds*. R package version 1.2-4, https://CRAN.R-project.org/package=minpack.lm

Gómez, V. (2015) "SSMMATLAB: A Set of MATLAB Programs for the Statistical Analysis of State Space Models". *Journal of Statistical Software*, 66(9), 1–37. https://doi.org/10.18637/jss.v066.i09 https://github.com/QuantLet/ssmmatlab

R Core Team (2025). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Richardson N, Cook I, Crane N, Dunnington D, François R, Keane J, Moldovan-Grünfeld D, Ooms J, Wujciak-Jens J, Apache Arrow (2024). *arrow: Integration to 'Apache' 'Arrow'*. R package version 17.0.0.1, https://CRAN.R-project.org/package=arrow.

Wickham, H. (2016) *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham H, François R, Henry L, Müller K, Vaughan D (2023). *dplyr: A Grammar of Data Manipulation*. R package version 1.1.4, https://CRAN.R-project.org/package=dplyr.

IV. Commuters

Istat, Italy

Direct estimates and their variances were provided by facilities of the R package **ReGenesees** (Zardetto, 2024). In particular, the functions used are:

`e.svydesign()`; this function is used to define a survey design object for complex survey data (for example sampling weights and stratification). It links survey data with metadata about the sampling design.

- `ext.calibrated()`. This function is used when you already have externally calibrated weights, and you want **ReGenesees** to treat them as if it had calibrated them itself, so that it can compute correct estimates and variances.
- `SvystatTM`. This function computes estimates, standard errors and confidence intervals for totals and means in subpopulations.

Area level or Fay Herriot model have been implemented with the R package **emdi** (Kreutzmann et al., 2019). In particular, the function `fh()` estimates indicators using the Fay-Herriot approach. Empirical best linear unbiased predictors (EBLUPs) and mean squared error (MSE) estimates are provided.

Transfer learning is implemented by in-house code[1] which makes use of basic R functions, applied to the data obtained by the functions above. This code is specifically designed to produce commuting statistics by combining MNO data with the relevant non-MNO data available at the Italian National Statistical Institute (Istat). This is why it might not be straightforward to apply it to different data structures.

References

Kreutzmann A, Pannier S, Rojas-Perilla N, Schmid T, Templ M, Tzavidis N (2019). “The R Package emdi for Estimating and Mapping Regionally Disaggregated Indicators.” *Journal of Statistical Software*, **91**(7), 1-33. doi:10.18637/jss.v091.i07 https://doi.org/10.18637/jss.v091.i07 https://cran.r-project.org/package=emdi

Zardetto D. (2024) “ReGenesees: R Evolved Generalized Software for Sampling Estimates and Errors in Surveys”. R package version 2.4, Istat, Italy, https://diegozardetto.github.io/ReGenesees/

[1] https://github.com/AlessandroPiovani/ESSnet-MNO/tree/main/commuting

V. Nights-spent

Istat, Italy

In our case study we used registry data collected by ISTAT, together with MNO data about night spent, provided by MNO operators and resulting from their on-premises data processing. The shapefiles used in the plots were obtained from the ISTAT website. We used the **sf** (Pebesma and Bivand, 2023) and **ggplot2** (Wickham, 2016) R packages respectively to read shapefiles and for the plots

The Augmented and quasi-Transfer Learning are implemented by in-house produced code[2], which makes use of the basic R `lm()` function and the **randomForest** package (Liaw and Wiener ,2002). This code has been specifically designed for this particular application and set of data, so it may not be straightforward to apply it to different data structures.

References

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." *R News*, **2**(3), 18-22. https://CRAN.R-project.org/doc/Rnews/

Pebesma, E., & Bivand, R. (2023). *Spatial Data Science: With Applications in R*. Chapman and Hall/CRC. https://doi.org/10.1201/9780429459016

Wickham H. (2026). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[2] https://github.com/AlessandroPiovani/ESSnet-MNO/tree/main/tourism

VI. Sensor presence

Statistics Norway, Norway

The simulation study described in Section 13.2 of Deliverable 3.2 is based on data generated using the tool presented in Section 1 of this document.

All the analyses illustrated were performed in the R environment (R Core Team, 2025) although they can be done in any software package. To apply the geographically weighted regression estimator (6.1) and to calculate the associated delete-one out-of-bag prediction error, it was developed an ad hoc R function `oob.err.R` that can be downloaded from the following repository:

https://github.com/mno-minds/MNO-MINDS/tree/main/R_code_Sensor

References

R Core Team (2025) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

VII. QR Experiment

Statistics Norway, Norway

The experiment presented in Chapter 14 of Deliverable 3.2 of the project uses the data from several statistical registers at Statistics Norway, such as the Household Register, the Population Register, the A-ordning, and so on.

The sample of persons is selected by simple random cluster (household) sampling from the Household Register, and the mobile phone users therein are identified by automatic mobile number search as one would have done in regular household surveys at Statistics Norway. Only an indicator of user (or not) is retained but not the relevant mobile phone numbers.

All the analyses illustrated in Chapter 14 were performed in the R environment (R Core Team, 2025) although they can be done in any software package. The basic post-stratified estimator can be implemented as described for QR pseudo experiment (Chapter 15 in Deliverable 3.2). To apply the SUD estimator (8.6) it was developed an ad hoc R function `sud.R` that can be downloaded from the following repository:

https://github.com/mno-minds/MNO-MINDS/tree/main/R_code_QR_experiment

References

R Core Team (2025) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

VIII. QR pseudo experiment

Istat, Italy

The application presented in chapter 15 of Deliverable 3.2 of the project uses the data of the Aspect of Daily Live (ADL) annual sample survey, which provides some information on the use of mobile phones (MP) at individual level and on the number of mobile phones owned by each household surveyed.

The data used are those relating to the year 2023, available only internally, without "limitations" introduced by methods aimed at protecting the confidentiality of the data, but the variables of interest are also available in public use microdata, freely available on the Istat corporate website at the following link:

https://www.istat.it/en/microdata/aspects-of-daily-life/

At the time of writing (May 2025), the latest ADL public use microdata refers to the year 2022.

The presented application exploits the subset of MP users in the ADL survey to reproduce the MNO counts that would be provided by the operators. Post-stratification is then used to assess whether it mitigates selection bias. Additional simulations use the initial data (users) to add user ambiguity error and assess how the post-stratification behaves.

All calculations are performed in the R environment (R Core Team, 2025); for post-stratification purposes, it is possible to use the `postStratify()` function in the **survey** package (Lumley, 2024). The function requires that the input data be passed as a survey design, as defined by the `svydesign()` function. Note that when defining the survey design associated with counts provided by an MNO operator, the counts should be passed as the `weights` argument to `svydesign()`.

References

Lumley T. (2024) "survey: analysis of complex survey samples". *R package version 4.4*.

R Core Team (2025) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/

Annex 1 - Metadata Framework open source software

Statistics Netherlands, The Netherlands

Metadata analysis of available data sets, such as MNO data, can be systemically analyzed to explore the possibilities of combining data. A major advantage of this approach is that no data are required, merely metadata. The code allows the user to analyze various scenario's based on assumed availability of data sources and methods.

The Python code can be found at

https://github.com/YGootzen/MetadataFramework/tree/main/python

and introduces new functions and classes to search for possible ways to combine data sources. This folder contains an object-oriented implementation in Python.

For new users, we recommend starting with the notebook classes_examples.ipynb. It contains examples and explanations of the most important concepts of the implementation. The notebook classes.ipynb contains all objects used in the implementation.

Case studies are applied in the notebook check_test_cases.ipynb. From within that notebook, you may select one of the existing test-case notebooks. The notebook test-case_mobility.ipynb contains all specifications required for to run an example use case regarding commuters. For implementing your own example, make a copy of one of the test_case_.ipynb notebooks and specify your own data, relations and models.

An upcoming WP2 deliverable will feature an application of the code. At the time of write, a notebook for MNO-related test cases is in development.