Software Impacts 14 (2022) 100428

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

# RanCoord — A random geographic coordinates generator for transport and logistics research and development activities

Hugo Silva Carvalho [a,*], André Pilastri [a], Rui Novais [b], Paulo Cortez [c]

[a] EPMQ, CCG ZGDV Institute, Guimarães, Portugal
[b] ABMN - Business Solutions, Azurém, Guimarães, Portugal
[c] ALGORITMI Centre, Minho University, Guimarães, Portugal

ARTICLE INFO

*Keywords:*
Geographic coordinates
Data generator
Travel Salesman Problem
Vehicle Routing Problem
Python

ABSTRACT

RanCoord is an open-source Python package that allows to easily generate random coordinates within a set of geographic boundaries. The framework offers methods to generate, save and plot a determined number of random coordinates generated from a location polygon or a simple name/address through its geocoding. Such geographic methods are well-suited for research and development activities in transport and logistics, such as the Travel Salesman Problem (TSP) and the Vehicle Routing Problem (VRP).

Code metadata

| | |
|---|---|
| Current code version | 0.0.6 |
| Permanent link to code/repository used for this code version | https://github.com/SoftwareImpacts/SIMPAC-2022-216 |
| Permanent link to Reproducible Capsule | https://codeocean.com/capsule/5623776/tree/v1 |
| Legal Code License | MIT |
| Code versioning system used | git |
| Software code languages, tools, and services used | Python 3.6+ |
| Compilation requirements, operating environments & dependencies | RanCoord requires numpy; pandas; shapely; geopy; folium; requests; json; xlsxwriter; csv; simplekml |
| If available Link to developer documentation/manual | https://github.com/hugodscarvalho/rancoord/blob/main/README.md |
| Support email for questions | hugo.carvalho@ccg.pt |

## 1. RanCoord — A random geographic coordinates generator for transport and logistics research and development activities

Data is currently becoming a valuable asset, being used to manage an increasingly number of daily world activities. Geospatial data is a particular data type that links an identifier (e.g., individual, object, transport vehicle) to a location. Because of this, geospatial data often raises privacy and confidentiality issues. Moreover, in some application domains, geospatial data is often a scarce [1]. RanCoord offers a straightforward, fast and intuitive computational strategy to circumvent these privacy concerns and availability barriers by generating random geographic coordinates to suit the user needs, both in terms of quantity and global geographic position. The package was created to particularly solve research and development needs related to the domains of transportation and logistics.

Fig. 1 depicts the overall package architecture that assumes three main layers that are detailed in the next subsections. Algorithm 1 presents the pseudocode adopted to generate the geographic coordinates.

### 1.1. Input layer

The **Input Layer** represents the communication layer of the application. It includes the set of parameters for the geographic coordinates generation that meet the user needs, a possible data source for the distance matrix calculation, and the OSRM Engine, a modern open source C++ routing engine for computing shortest paths in road networks. The set of parameters include:

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
*E-mail address:* hugo.carvalho@ccg.pt (H.S. Carvalho).

https://doi.org/10.1016/j.simpa.2022.100428
Received 27 September 2022; Accepted 5 October 2022

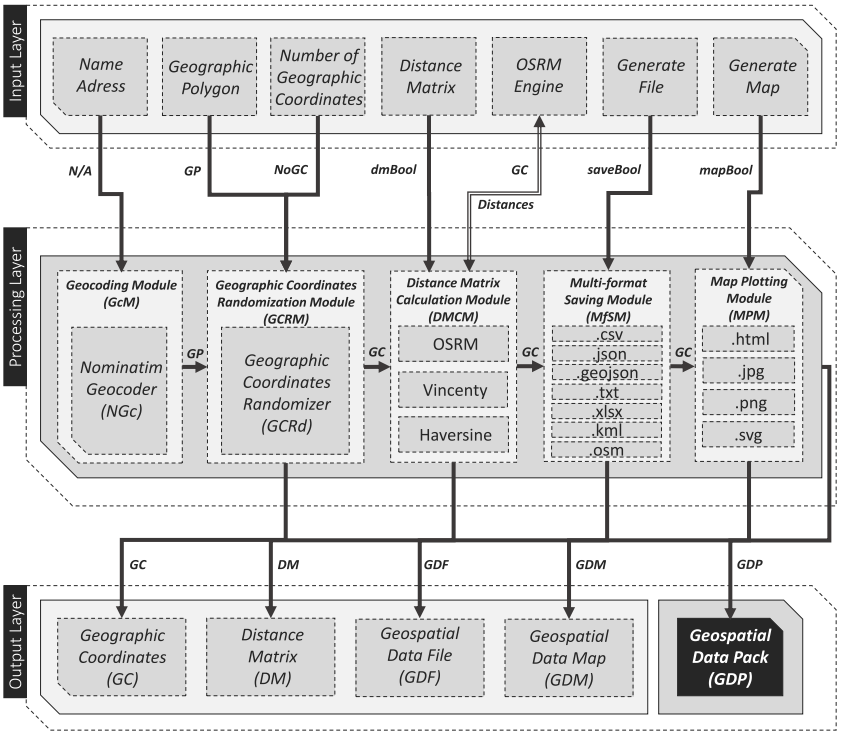


**Fig. 1.** Rancoord architecture.

(a) **Name/Adress (N/A)** or a **Geographic Polygon (GP)** to be considered.
(b) **Number of Geographic Coordinates (NoGC)** to be generated.
(c) The option to return the **distance matrix (dmBool)** of the geographical coordinates to be generated using the default distance calculation method (*Haversine*).
(d) The option to save or not to **save the geographic coordinates (saveBool)** using the standard format (*JSON*).
(e) The option to **plot and save the resulting map (mapBool)** of the geographical coordinates to be generated.

It is relevant to highlight that the input parameter **(a)** can be filled and executed using two options:

(i) **Name/Address** — The name, address, query, or a structured query the user wishes to use as the geographical position to be considered. A street, city, county, state, country, or postcode should be provided.
(ii) **Geographic Polygon (GP)** — a object that represents a filled area consisting of a list of at least three coordinate tuples that forms the outer ring and a (possible) list of polygons [2].

### 1.2. Processing layer

The **Processing Layer** performs the random generation of geographic coordinates. These coordinates follow the parameters specified by the user in the Inputs phase. Suppose the user has chosen the first option of execution using the Name/Address option in the Input Layer. In that case, the process will go through a geocoding stage of the corresponding name or address, where its bounding box[1] will be requested to a third party. Accordingly, the behaviour of the components that make up this phase can be described as follows:

1. **Geocoding Module (GcM)** — geocoding stage of the specified name or address requesting its **bounding box** to **Nominatim**. This geocoder uses OpenStreetMap data to find locations on Earth by name and address [3]. This set of coordinates will be converted into a Shapely Polygon and then passed to the randomization component.
2. **Geographic Coordinates Randomization Module (GCRM)** — the core component of the current phase and the whole framework. Using the Shapely Polygon from the previous component, depending on the user's choice and the number of locations to be generated, it generates a random set of pairs containing latitude and longitude following a continuous uniform distribution. The distribution formulated in Eq. (1) and presented in Fig. 2 describes an experiment with an arbitrary outcome that lies between certain bounds. The bounds are defined by the parameters, $a$ and $b$, which are the minimum and maximum values.

$$f x(x) = \begin{cases} \frac{1}{b-a} & \text{for } x \in [a, b] \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

3. **Distance Matrix Calculation Module (DMCM)** — This component allows the user to get the distance matrix of the generated geographic coordinates. This matrix will have a size of $N \times N$ where $N$ is the number of geographic coordinates specified by the user and, consequently, generated. This two-dimensional data structure is often required to solve transport and logistics problems such as the Travel Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). Thus, using Rancoord, there is no longer a need for a third party or user implementation to obtain the distance matrix.
As shown in Table 1, the main diagonal of the square matrix will be null since the distance between the same point is zero. The remaining values will be filled using the method selected by the user, where $d$ represents the distance between the respective pairs of coordinates. The tool allows the calculation of this distance matrix using three different methods:

[1] Set of minimum and maximum longitudes and latitudes of a geographical area.

**Algorithm 1:** RanCoord

**Inputs :** A name/adress *nameAddress* or a geographic polygon *geoPoly*; a number of geographic coordinates *num*; booleans *dmBool*, *saveBool*, and *mapBool*, to calculate distance matrix *dm*, save the data in a *file*, and generate the *map*, respectively

**Outputs:** A set of *num* geographic coordinates {*Latitude*, *Longitude*}; a distance matrix *dm*; a file *file*; a map *map*

initialization;
**if** *geoPoly* ***is*** *empty* **then**
  *polygon* = **geocode** *nameAddress*
**else**
  *polygon* = *geoPoly*
**end**
*minLat*, *minLon*, *maxLat*, *maxLon* = *polygon*.**bounds**
*coordinates* = Empty list;
**while** ***length of*** *coordinates* < *num* **do**
  *randomCoordinate* = **random.uniform(** *minLat*, *maxLat* **),**
  **random.uniform(** *minLon*, *maxLon* **)**
  **if** *randomCoordinate* ***within*** *polygon* **then**
    **append** *randomCoordinate* **to** *points*
  **else**
    **continue**
  **end**
**end**
**if** *dmBool* **then**
  *dm* = **distanceMatrix(** *coordinates* **)**
**else**
  **continue**
**end**
**if** *saveBool* **then**
  **multipleFormatsSaver(** *coordinates* **)**
**else**
  **continue**
**end**
**if** *mapBool* **then**
  **plotCoordinates(** *coordinates* **)**
**else**
  **continue**
**end**
**return** *coordinates*, *dm*, *file*, *map*

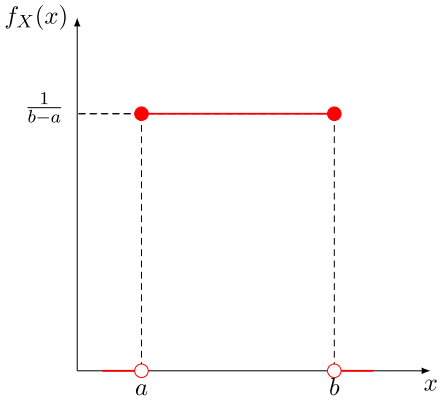


**Fig. 2.** Probability density function of the continuous uniform distribution.

(a) **Haversine** — Determines the **great-circle distance**[2] between two points on a sphere given their longitudes and latitudes [4].

[2] Shortest distance between two points on the surface of a sphere.

**Table 1**
$N \times N$ distance matrix.

$$D = \begin{pmatrix} & (lat,lon)_1 & (lat,lon)_2 & (lat,lon)_3 & \dots & (lat,lon)_d \\ (lat,lon)_1 & 0 & d_{12} & d_{13} & \dots & d_{1d} \\ (lat,lon)_2 & d_{21} & 0 & d_{23} & \dots & d_{2d} \\ (lat,lon)_3 & d_{31} & d_{32} & 0 & \dots & d_{3d} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ (lat,lon)_n & d_{n1} & d_{n2} & d_{n3} & \dots & 0 \end{pmatrix}$$

(b) **Vincenty** — Calculates geodesic distances between a pair of latitude/longitude points on an ellipsoidal model of the Earth. Unlike the Haversine method for calculating distance on a sphere, this formula is an iterative method and assumes the Earth is an ellipsoid [4].

(c) **Open Source Routing Machine (OSRM)** — High-performance routing engine for shortest paths in road networks. It combines sophisticated routing algorithms with the open and free road network data of the OpenStreetMap (OSM) project [5]. Despite its high performance, the fact that it represents a solution that verifies road networks and not just a mathematical distance between two coordinates makes this the most computationally expensive method but also the most accurate.

4. **Multi-format Saving Module (MfSM)** — Save the generated geographic coordinates in the following standard formats: CSV, JSON, GEOJSON, TXT, XLSX, KML, and OSM.
5. **Map Plotting Module (MPM)** — Plot the generated geographic coordinates on a map using the Python Folium package, which is powered by Leaflet.js [6]. The map can be saved in the following formats: HTML, JPG, PNG, and SVG.

### 1.3. Output layer

The Output Layer represents the final stage of the tool and comprises the output of the different modules of the Processing Layer. Thus, a Geospatial Data Pack (GDP) can be generated if the framework is used in a chained approach or the following single outputs:

(a) **Geographic Coordinates (GC)** — Set of requested geographic coordinates that meet the parameters given by the user.
(b) **Distance Matrix (DM)** — Distance matrix of the generated geographic coordinates.
(c) **Geospatial Data File (GDF)** — A file containing the generated geographic coordinates.
(d) **Geospatial Data Map (GDM)** — A map containing the generated geographic coordinates.

Fig. 3 shows the plotted result of a RanCoord execution with the specification of 50 geographic coordinates located in the city limits of London, United Kingdom.

## 2. Impact on academic research

RanCoord provides a straightforward, fast, and intuitive way to generate random coordinates within a set of boundaries. A growing number of researchers are studying travelling salesman problems (TSP), vehicle routing problems (VRPs), and their variants, considering real-life applications and scenarios [7]. The present proposal reduces the time and effort required to acquire geographic coordinates within a specific location, eliminates the need to use geographic data that does not fit a particular problem to be addressed, and provides geographic coordinates in an effortless, fast, and customized way to the user's needs. Moreover, since it generates random coordinates, the module does not rise privacy or confidentiality issues.

RanCoord was used in several scientific studies and development activities. As a means of overcoming the lack of data derived from General Data Protection Regulation (GDPR) issues, RanCoord was adopted

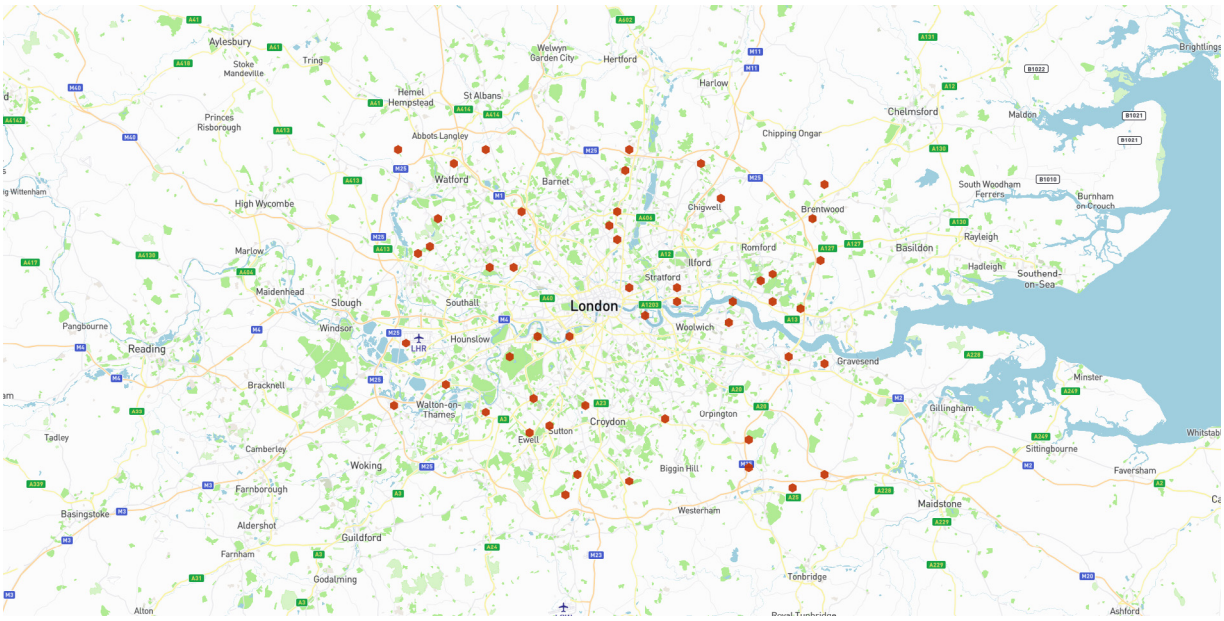


**Fig. 3.** Generation of 50 random coordinates within the geographic limits of the city of London, United Kingdom.

in [8] to generate vehicle routing problem instances within locations that match the geographic logistics context of several freight transport companies, thereby improving the communication of results with the managers of the various companies and, at the same time, accelerating the research and development of optimization methods in months.

## 3. Future work

RanCoord is a recent Python package, having its lifetime spanned across since April 2022. In the future, the module will be further improved by optimizing its performance in terms of execution time while maintaining the core functionalities presented here. We also intend to add new methods of generating geospatial data beyond the currently adopted continuous uniform distribution (e.g., normal distribution). It is also expected the development of a new module that provides functionalities associated with geospatial clustering by using methods such as K-Means and DBSCAN.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The authors would like to express the greatest recognition to the project on which this package has arisen, “aDyTrans - Dynamic Transportations Platform” reference NORTE-01-0247-FEDER-045174, supported by Norte Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF). The authors are also grateful for all the contributors who assisted in making RanCoord more intuitive, complete, and relevant to the subject.

## References

[1] Jae-Gil Lee, Minseo Kang, Geospatial big data: Challenges and opportunities, Big Data Res. 2 (2) (2015) 74–81.
[2] Sean Gillies, et al., Shapely: manipulation and analysis of geometric objects, 2007, URL https://github.com/shapely/shapely.
[3] OpenStreetMap contributors, Planet dump retrieved from https://planet.osm.org, 2017, https://www.openstreetmap.org.
[4] Hagar Mahmoud, Nadine Akkari, Shortest path calculation: A comparative study for location-based recommender system, in: 2016 World Symposium on Computer Applications & Research, WSCAR, 2016, pp. 1–5.
[5] Dennis Luxen, Christian Vetter, Real-time routing with OpenStreetMap data, in: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS ’11, ACM, New York, NY, USA, 2011, pp. 513–516.
[6] Python-Visualization, Folium, URL https://python-visualization.github.io/folium/.
[7] Shi-Yi Tan, Wei-Chang Yeh, The vehicle routing problem: State-of-the-art classification and review, Appl. Sci. 11 (21) (2021).
[8] Hugo Silva Carvalho, André Pilastri, Luís Miguel Matos, Arthur Matta, Rui Novais, Paulo Cortez, An intelligent decision support system for road freight transport, in: Intelligent Data Engineering and Automated Learning – IDEAL 2022, Springer International Publishing, Cham, 2022.