Around The World in 80 Days

Programming with Python
Master in Data science for economics, business, finance

Prof.
Alfio Ferrara

Authors:
Raffaella Michaela DeMarco
Pietro Russo

Aim of the project

Like a new Phileas Fogg you have the desire to travel around the world always moving east, could you do it in 80 days?

The aim of the project is to help you and show you the best way to fulfill your dream, with the method *travel* of the *AroundTheWorld* class .

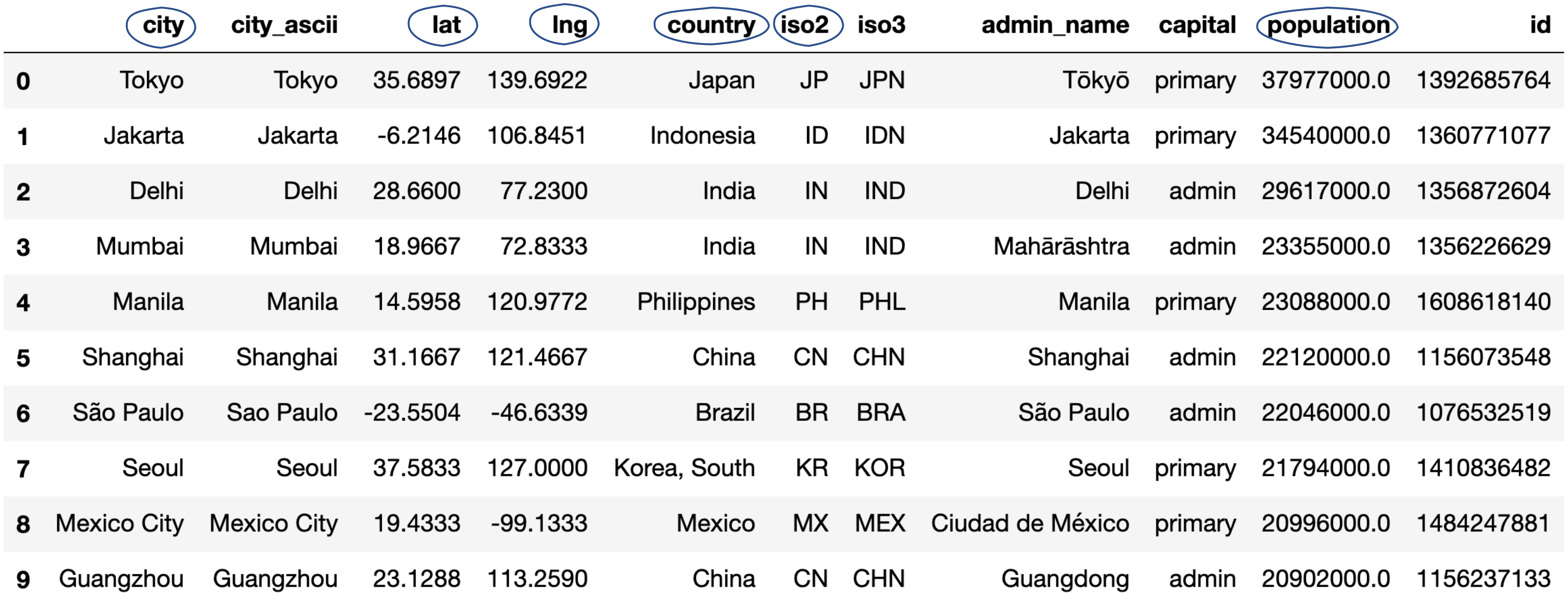
The starting point is the city of London (GB), but it could be the one you want.

Parameters of *AroundTheWorld*

- dataframe : Dataset of all cities
- city_start : Name of the starting city
- country_start : Name of the starting country
- n_min : Number of the closest cities to which it is possible to travel
- x_size : Size of the longitudinal side of the grid used to search for the nearest cities
- y_size : Size of the latitudinal side of the grid used to search for the nearest cities
- rise_factor : Multiplication factor to increase the grid used to search for the nearest cities

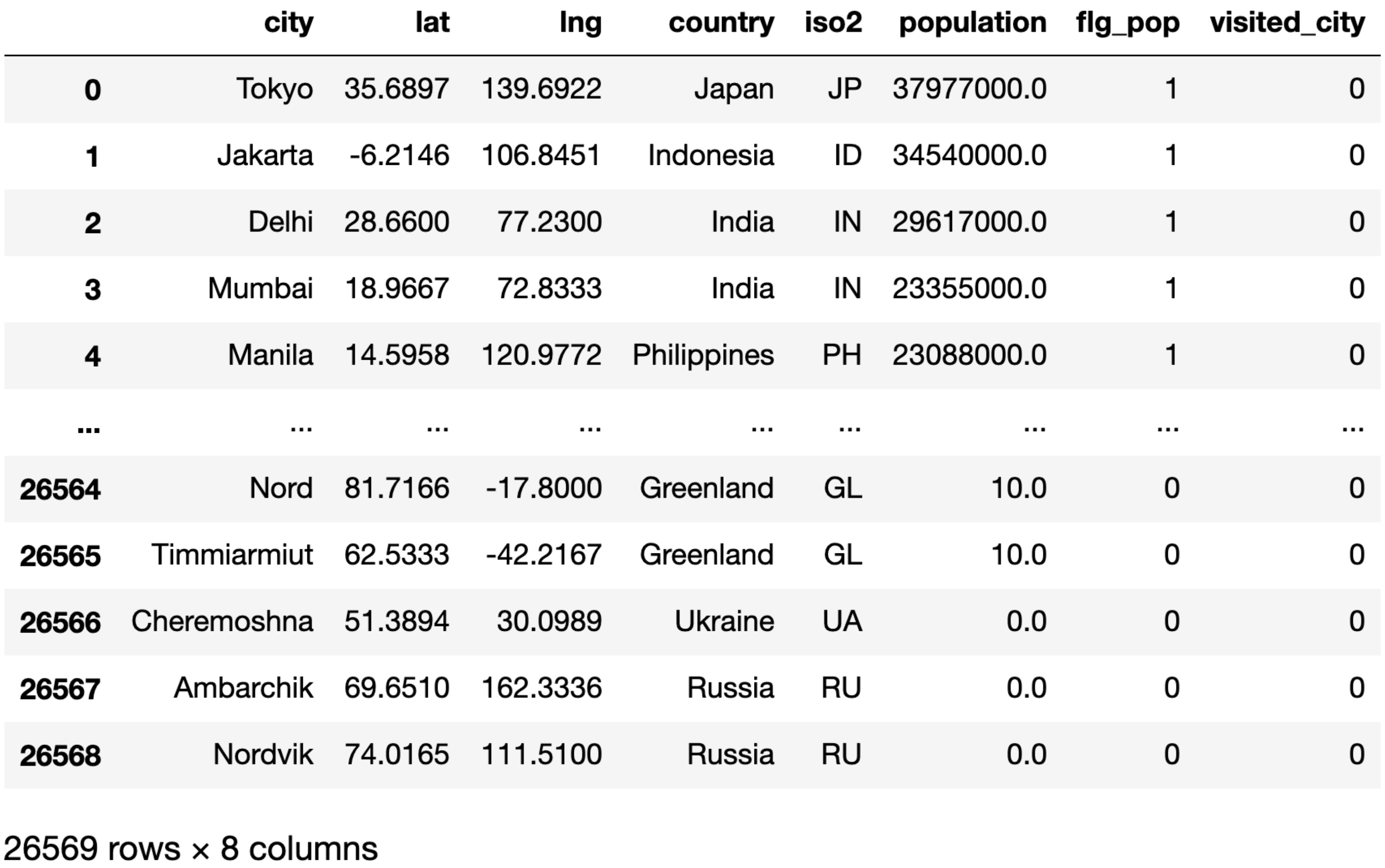
Input Data

A dataset with 26569 cities of the world



	city	city_ascii	lat	lng	country	iso2	iso3	admin_name	capital	population	id
0	Tokyo	Tokyo	35.6897	139.6922	Japan	JP	JPN	Tōkyō	primary	37977000.0	1392685764
1	Jakarta	Jakarta	-6.2146	106.8451	Indonesia	ID	IDN	Jakarta	primary	34540000.0	1360771077
2	Delhi	Delhi	28.6600	77.2300	India	IN	IND	Delhi	admin	29617000.0	1356872604
3	Mumbai	Mumbai	18.9667	72.8333	India	IN	IND	Mahārāshtra	admin	23355000.0	1356226629
4	Manila	Manila	14.5958	120.9772	Philippines	PH	PHL	Manila	primary	23088000.0	1608618140
5	Shanghai	Shanghai	31.1667	121.4667	China	CN	CHN	Shanghai	admin	22120000.0	1156073548
6	São Paulo	Sao Paulo	-23.5504	-46.6339	Brazil	BR	BRA	São Paulo	admin	22046000.0	1076532519
7	Seoul	Seoul	37.5833	127.0000	Korea, South	KR	KOR	Seoul	primary	21794000.0	1410836482
8	Mexico City	Mexico City	19.4333	-99.1333	Mexico	MX	MEX	Ciudad de México	primary	20996000.0	1484247881
9	Guangzhou	Guangzhou	23.1288	113.2590	China	CN	CHN	Guangdong	admin	20902000.0	1156237133

DataFrame definition



	city	lat	lng	country	iso2	population	flg_pop	visited_city
0	Tokyo	35.6897	139.6922	Japan	JP	37977000.0	1	0
1	Jakarta	-6.2146	106.8451	Indonesia	ID	34540000.0	1	0
2	Delhi	28.6600	77.2300	India	IN	29617000.0	1	0
3	Mumbai	18.9667	72.8333	India	IN	23355000.0	1	0
4	Manila	14.5958	120.9772	Philippines	PH	23088000.0	1	0
...	...	...	...	...	...	...	...	...
26564	Nord	81.7166	-17.8000	Greenland	GL	10.0	0	0
26565	Timmiarmiut	62.5333	-42.2167	Greenland	GL	10.0	0	0
26566	Cheremoshna	51.3894	30.0989	Ukraine	UA	0.0	0	0
26567	Ambarchik	69.6510	162.3336	Russia	RU	0.0	0	0
26568	Nordvik	74.0165	111.5100	Russia	RU	0.0	0	0

26569 rows × 8 columns

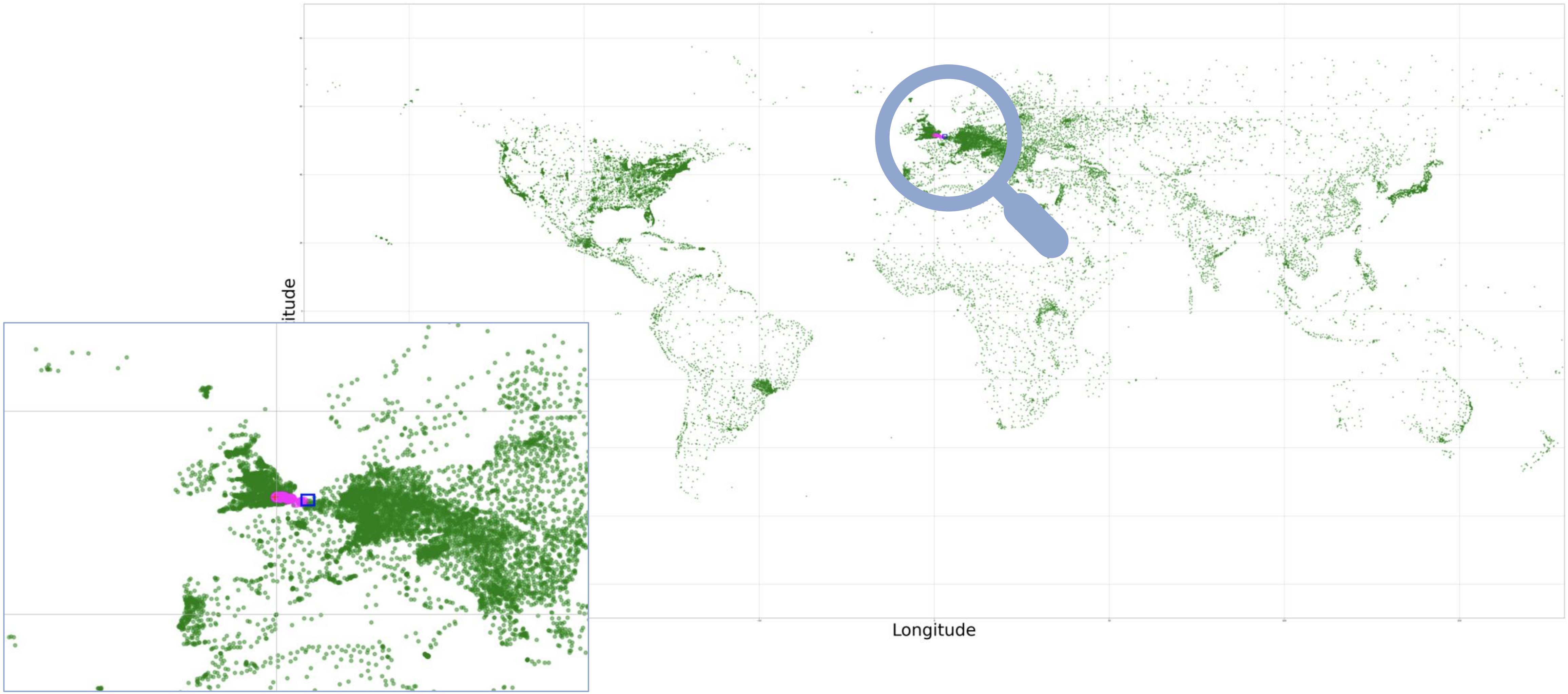
Grid definition

Move always towards the east

At each step, starting from the city in which it is located, the algorithm calculates a **rectangle**:

- **base**: the distance between the longitude of the current city and a subsequent point at a variable distance given as input (x_size).
- **height**: changes depending on whether the current city is north (latitude is greater) or south (latitude is lower) of the starting city:
    - North: height is the distance generated between the latitude of the current city, adding a quantity (y_size/2) and the latitude of the starting city, subtracting a quantity (y_size/2)
    - South: height is the distance generated between the latitude of the starting city, adding a quantity (y_size/2) and the latitude of the current city, subtracting a quantity (y_size/2)

Grid visualization



Weight assignment criteria

- **Distance:** at each step, according to the increasing Euclidean distance, the 3 closest cities are assigned values 2, 4, 8 respectively.
- **Population**: a weight of value 2 is added if the city has a population greater than or equal to 200 thousand inhabitants.
- **Country**: a weight of value 2 is added if the city in the next step is located in a different country than the previous one.

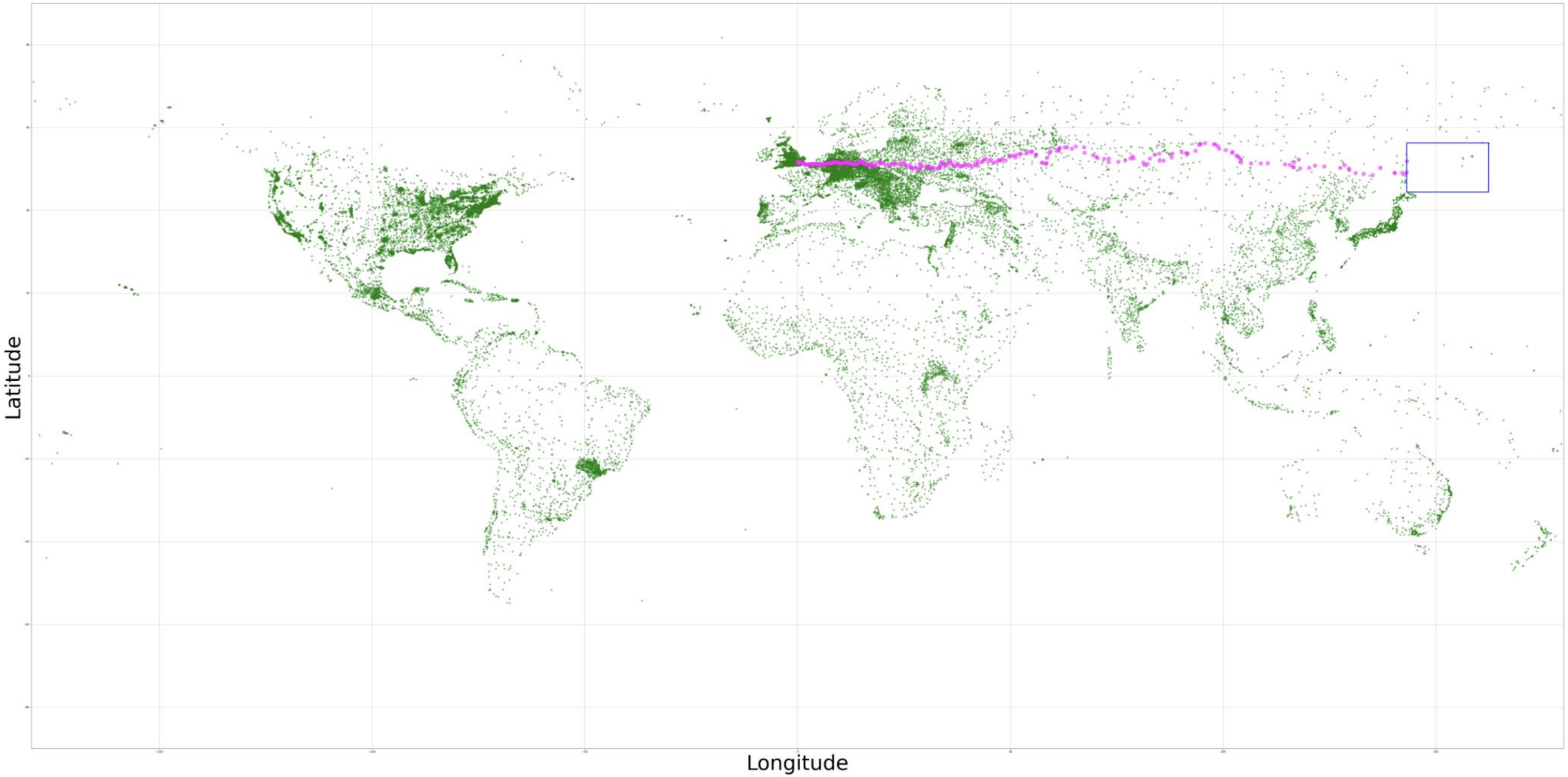
Grid variability

What happens when Phileas must cross oceans?

The size of the polygon considered at each step is variable to ensure the functionality of the algorithm with a minimum number of at least **3 cities.**

The absence in the rectangle of at least 3 cities makes these dimensions vary by a multiplicative **rise_factor**.

Grid variability visualization



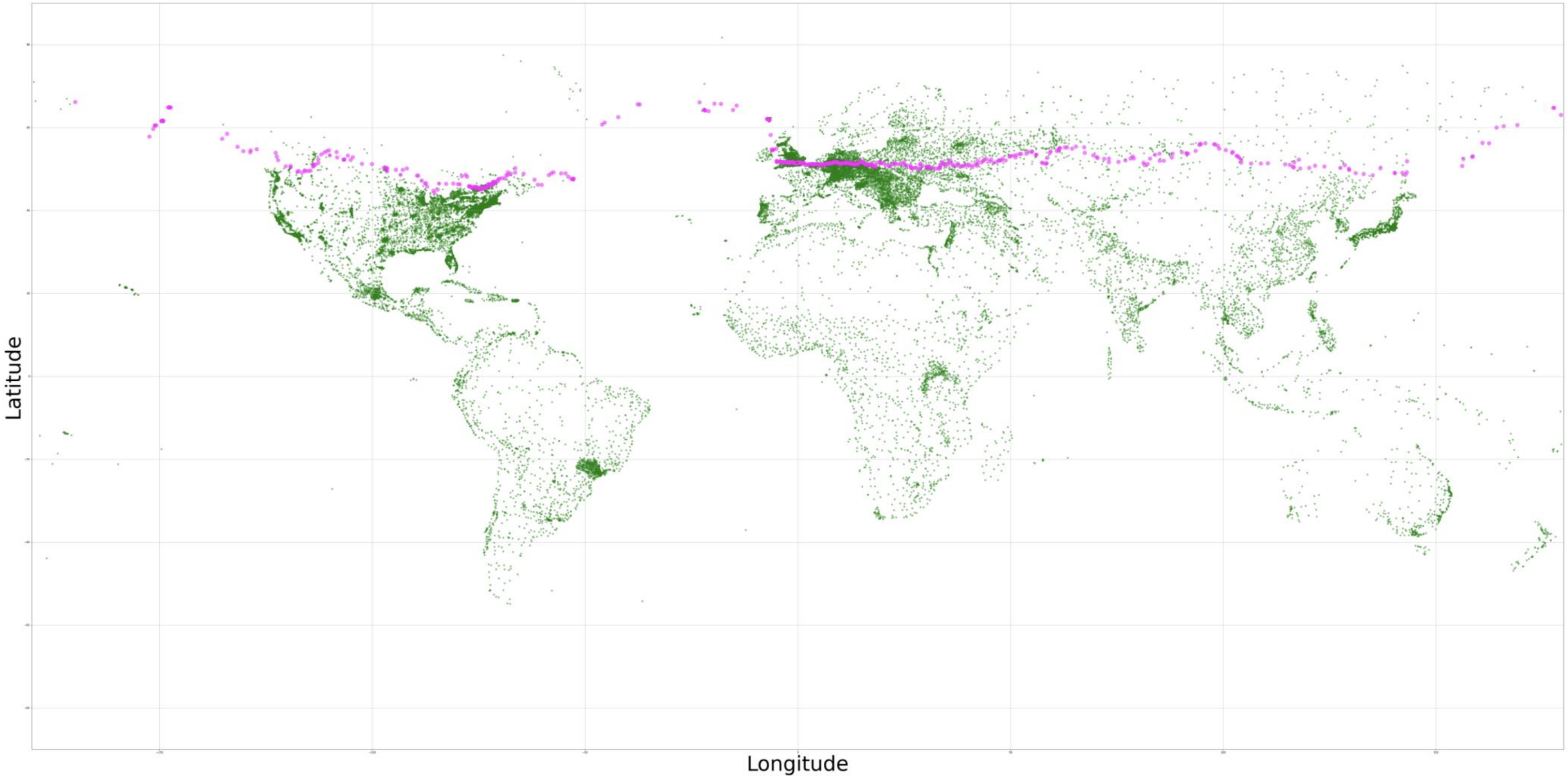
Stop criteria

If among the 3 nearest cities the starting city is present, the algorithm chooses that one, adding up the weight it has.

Once Phileas returns to London, the algorithm returns :

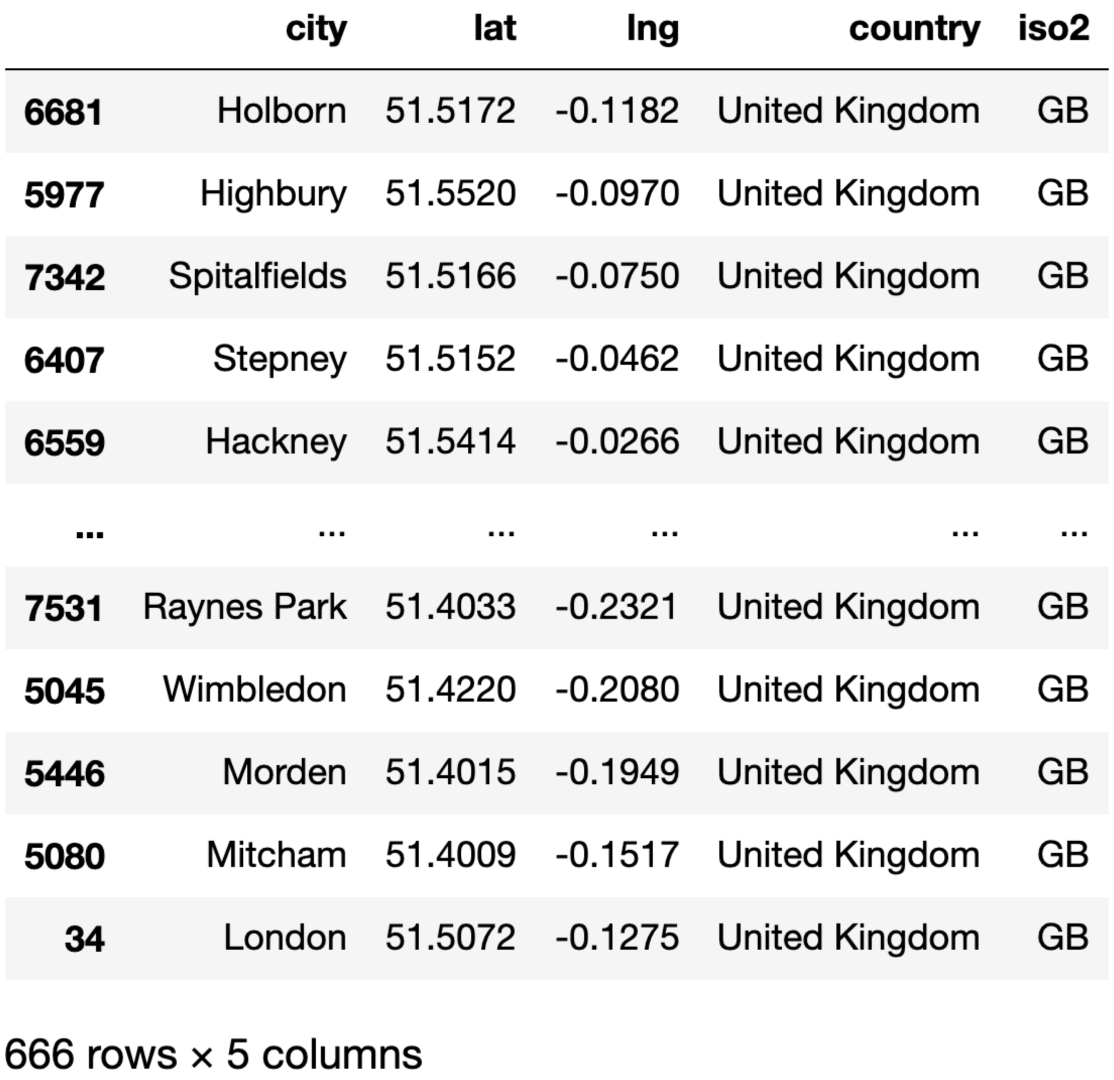
- the **number of hours** taken to complete the journey (sum of the weights of all the steps)
- the total number of **steps**
- the **dataframe** of all the cities visited

Path



```
Completed the journey starting from London (GB) in 62.42 days (1498 hours) after visited 666 cities.
Done in 20.59 seconds.
```

Visited City DataFrame



	city	lat	lng	country	iso2
6681	Holborn	51.5172	-0.1182	United Kingdom	GB
5977	Highbury	51.5520	-0.0970	United Kingdom	GB
7342	Spitalfields	51.5166	-0.0750	United Kingdom	GB
6407	Stepney	51.5152	-0.0462	United Kingdom	GB
6559	Hackney	51.5414	-0.0266	United Kingdom	GB
...	...	...	...	...	...
7531	Raynes Park	51.4033	-0.2321	United Kingdom	GB
5045	Wimbledon	51.4220	-0.2080	United Kingdom	GB
5446	Morden	51.4015	-0.1949	United Kingdom	GB
5080	Mitcham	51.4009	-0.1517	United Kingdom	GB
34	London	51.5072	-0.1275	United Kingdom	GB

666 rows × 5 columns

Thanks for your attention

Phileas Fogg