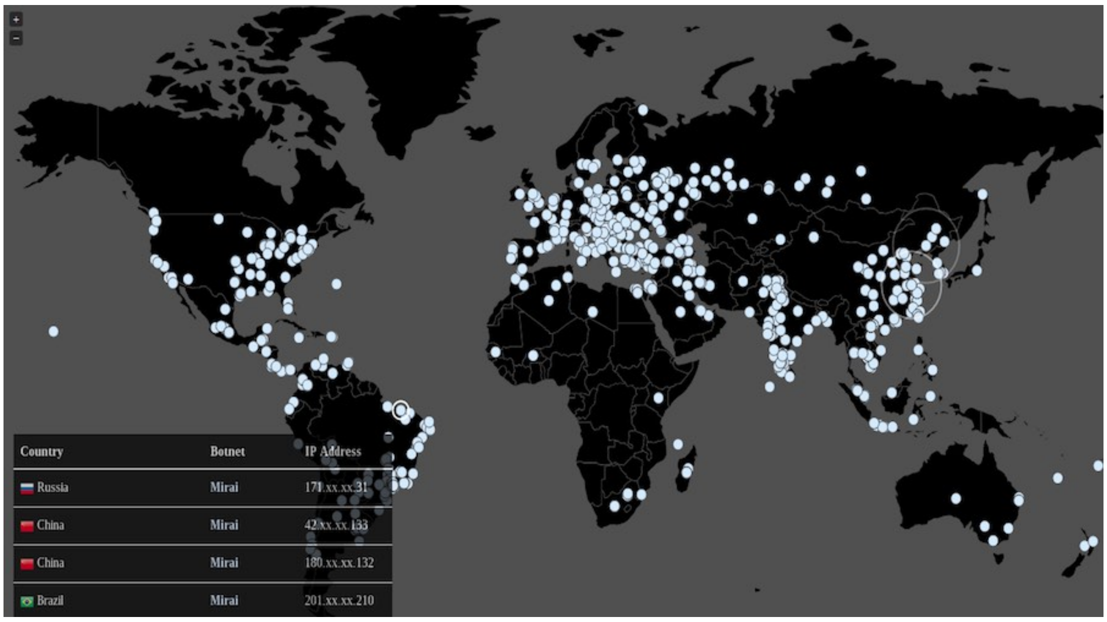
Data Wrangling and Visualisation.
Assignment 2.
Traffic visualisation.

In this assignment, you will visualise web traffic that comes to your server from various location around the world. The learning purpose is to strengthen your knowledge in backend part of the course, in particular, *requests* package in Python and routing in Flask. The second purpose of the assignment is to dive into Three.js framework to create interactive and nice-looking visualisation of the packages. The task includes following steps:

1) Implementation of a **Python script** that reads .csv file and send the packages to a Flask server [15 points]
2) Implementation of the **Flask server** that parses the received packages and sends them to frontend part [20 points]
3) Visualisation of package location on World map or Globe using **Three.js** [50 points]
4) Delivering all three parts in **Docker** [15 points]



Country	Botnet	IP Address
Russia	Mirai	171.xx.xx.31
China	Mirai	42.xx.xx.133
China	Mirai	180.xx.xx.132
Brazil	Mirai	201.xx.xx.210

Example of visualisation

1. Data generation

Alongside with this instruction, you get a csv file. Each row represents a "package" that consist of:

- *ip address* of the sender;
- *Latitude* and *Longitude* of the approximate location;
- Timestamp of the package receiving;
- *suspicious* mark that you get from some anomaly traffic detector (0 is normal package, 1 is suspicious).

In real life, you'd use the utilities like nmap and Wireshark to retrieve these info from traffic, but for simplicity we provide it in the ready file format. You're asked to write a python script that loads the data from file and send to localhost with respect to the timestamps, i.e. the packages shouldn't be sent at once and in the same order and time intervals as specified in the time column.
Send the packages in json format.

2. Data receiving

While the packages are sending to localhost, there should be the receiver side. You need to implement a running server that listen to some address and the parser function that extracts package info. Both of them can be implemented on Flask. Implement a GET method to get json-s from the "senders". While you might be an expert in another backend framework, we're asking you to use **only Flask** (or some of its modifications). Finally, you also need to implement a function that sends the processed data to the frontend.

3. Visualisation of the senders' location

The main part of the assignment is to create a clear and truthful visualisation of the senders' location. While we give you the design freedom, there are several things that should and shouldn't be in your implementation:

- The locations should be presented on some world map or globe;
- The locations should represent the real picture of dataset;

- The frontend should include at least **two** interaction features. For example, a list of the most common locations that updates in real time;
- The frontend may contain additional plots, for example, with the peak(s) of activity;
- The locations **shouldn't** just "pop-up" and disappear, because it makes analysis difficult. At the same time you can stop drawing the points after, say, 10 secs.
- The visualisation **shouldn't** be overwhelmed(check the inside how to avoid).

4. Deployment

To make your solution reproducible, wrap all sender, flask and frontend parts in docker containers.The whole system should be started up by single run of docker compose.