ESIEE Paris - EIG-2010

Inégalités de richesse illustrées par la répartition des milliardaires dans le monde

Ghidaglia Boris

5 juin 2017

Sommaire

1 Introduction

Les inégalités de richesse sont un sujet central en géopolitique et en économie depuis des années. De nombreuses études les ont déjà démontrées et tendent même à prouver qu'elles ne cessent de s'accroître. Une manière de bien montrer cette répartition de richesse est de compter le nombre de milliardaires par pays. C'est le sujet que j'ai choisi pour dresser ma carte. Ce que l'ont sait moins, et qui peut être surprenant, c'est qu'au sein même de cette communauté déjà extrêmement restreinte, une inégalité du même genre réside. Ce sera le sujet de l'histogramme que je présenterai.

FIGURE 1.1 – Freepik ©

2 Ressources

Cette section a pour but de présenter les quelques éléments un peu exotiques que j'ai pu utiliser. Je n'aborderai donc pas ici BaseMap, Matplotlib ou autres librairies vues en cours. D'autre part, elle présente aussi les données que j'ai étudiées.

2.1 Données

Les données que j'ai exploitées sont issues du travail de Caroline Freund et Sarah Oliver, qui ont publié en Février 2016 une base de données du nom de : "The Origins of the Superrich : The Billionaire Characteristics Database". Celle-ci contient une multitude d'information à propos des milliardaires. J'ai téléchargé le fichier json de cette base de données mise à disposition sur le site : public-us.opendatasoft.com. Je l'ai exploitée sans la dégrader auparavant, comme le sujet l'impose. Toutes les manipulations que j'ai pu mettre en oeuvre sur ce fichier afin de récupérer les données intéressantes pour mon sujet sont commentées dans le code.

2.2 Numpy

Numpy est une librairie python très utile et bien connue pour les mathématiques, je ne m'attarderai donc pas sur l'utilisation que j'ai pu brièvement en faire pour calculer certaines valeurs caractéristiques définissant mon histogramme (médiane, écart-type, variance, etc..).
En revanche, j'ai découvert lors de ce projet deux fonctions de Numpy très intéressantes : save et load. Cette découverte est issue d'une optimisation que je voulais mettre en place. Le fait est que pour afficher ma carte ou mon hitogramme, il faut que BaseMap exploite des valeurs. Or le fichier json est conséquent, et les requêtes internet passées pour récupérer les localisation d'un pays prennent non seulement du temps, mais nécessitent en plus d'avoir accès à internet. L'optimisation était donc évidente, il fallait trouver un moyen de sauvegarder localement, dans un fichier par exemple, les données construites par les programmes que j'ai créés. Ces données étant des objets, principalement des dictionnaires ou des listes. Ces deux fonctions remplissent donc parfaitement les critères de l'optimisation recherchée. Pour plus d'informations à leurs propos, voir le module numpy_saver_and_loader.py correspondant.

2.3 Pylint

Pylint est proposé dans le cours et permet de noter un code donné en fonction des conventions et bonnes pratiques mises en oeuvre dans celui-ci. J'ai utilisé ce module et tous mes codes ont la note maximale qu'ils peuvent obtenir. En effet, en utilisant Pylint avec WinPython, on note une erreur redondante : la position des imports. Le problème est que WinPython ne reconnaît pas directement les autres fichiers python du répertoire courant. Ainsi, on doit importer sys afin d'ajouter manuellement le répertoire courant au path. Pour ce faire nous devons obligatoirement intercaler une ligne de code entre les imports classiques et ceux correspondant à des modules du répertoire courant. Cependant cela ne respecte pas les conventions, Pylint nous le signale donc. Une autre erreur apparaît aussi : no-member. D'après mes recherches il s'agit d'une erreur lié à numpy. J'aurais pu les désactiver en modifiant la configuration de pylint mais l'intérêt était minime.

3 Production des données : lecture_data_json.py

La carte a été créée à partir des données décrites plus haut. Afin de compter le nombre de milliardaires par pays, il a fallu filtrer et parser ces données. Tout d'abord, on récupère dans des variables le pays, le nom, la fortune et l'année propre à chaque section de milliardaire de la liste. En effet certains sont répétés plusieurs fois sur plusieurs années, et certains parfois même plusieurs fois en une année. Nous verrons comment ces problèmes ont été résolus. Dans un premier temps, on test si la fortune du milliardaire est renseignée (pas None), si celle-ci est supérieure ou égale à 1 milliard, si l'année est bien 2015 (l'année sur laquelle nous faisons notre étude) et enfin si le pays n'est pas None non plus. Si toutes ces conditions sont respectées alors nous testons si le nom est déjà dans le dictionnaire liant les noms des milliardaires à leurs fortunes respectives. Si ça n'est pas le cas alors nous ajoutons le milliardaire et sa fortune au dictionnaire. Sinon nous actualisons sa fortune en comparant celle du dictionnaire avec la nouvelle et nous gardons la plus grande. Dans le cas où le milliardaire était nouveau, nous regardons si son pays l'est aussi. Si c'est le cas nous ajoutons le pays à un autre dictionnaire contenant le nom des pays liés à leur nombre de milliardaires. Et dans tous les cas nous incrémentons la valeur du nombre de milliardaires pour le pays en question. Enfin, après avoir parcouru tous les milliardaires, nous parcourons le dictionnaire des fortunes des milliardaires pour stocker toutes les fortunes dans une liste. Celle-ci sera utilisée pour la création de l'histogramme. Quand au dictionnaire du nombre de milliardaires par pays, il sera évidemment utile pour la carte.

4 Carte : lecture_data_json.py localisation.py et main.py

Avant de pouvoir dresser la carte, nous devons récupérer les localisations des pays contenus dans le dictionnaire créé dans lecture_data_json.py. C'est le rôle de locations.py. Celui-ci va scrapper puis parser un json généré par l'api de google grâce à deux module : urllib.request et json. Une clef développeur a été nécessaire pour appeler un grand nombre de fois ces urls. Toutes ces localisations sont sauvegardées dans un dictionnaire liant le nom des pays à leur localisation. Dans main.py, on liera les localisations à leur nombre de milliardaires associés afin de donner aux point à afficher une taille relative au nombre de milliardaires du pays auquel ils sont liés.

Observations La carte montre bien l'inégale répartition des milliardaires dans le monde. Les grandes puissances ressortent en tête et les pays du tiers monde ou peu développés en sont totalement dénués ou presque.

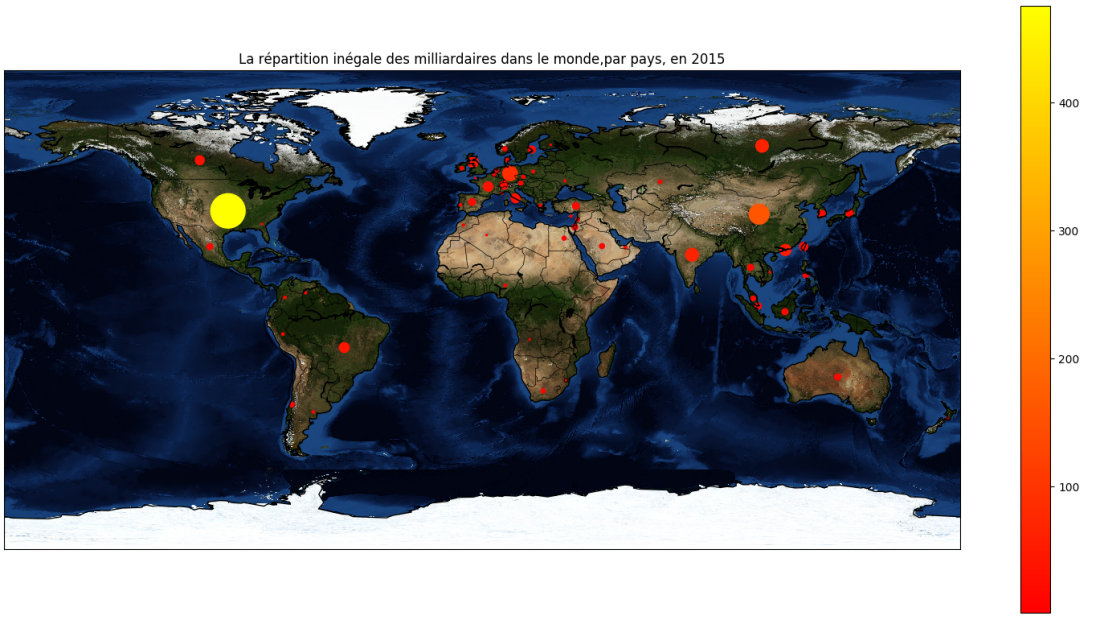

FIGURE 4.1 – Carte

5 Histogramme : lecture_data_json.py et histogramme.py

L'histogramme est créé à partir de la liste de fortunes construite dans lecture_data_json.py. Il affiche celle-ci en milliards de dollars, par pas de 1 milliard. De plus, il calcule les valeurs remarquables de la liste et les affiche en légende de l'histogramme.

Observations L'histogramme met en lumière cette inégalité amusante : au sein de la communauté des milliardaires, les chances d'avoir un milliard de plus diminuent exponentiellement. Nombreux sont ceux qui possèdent 1 milliard, mais Bill Gates est bien le seul à en avoir plus de 57.

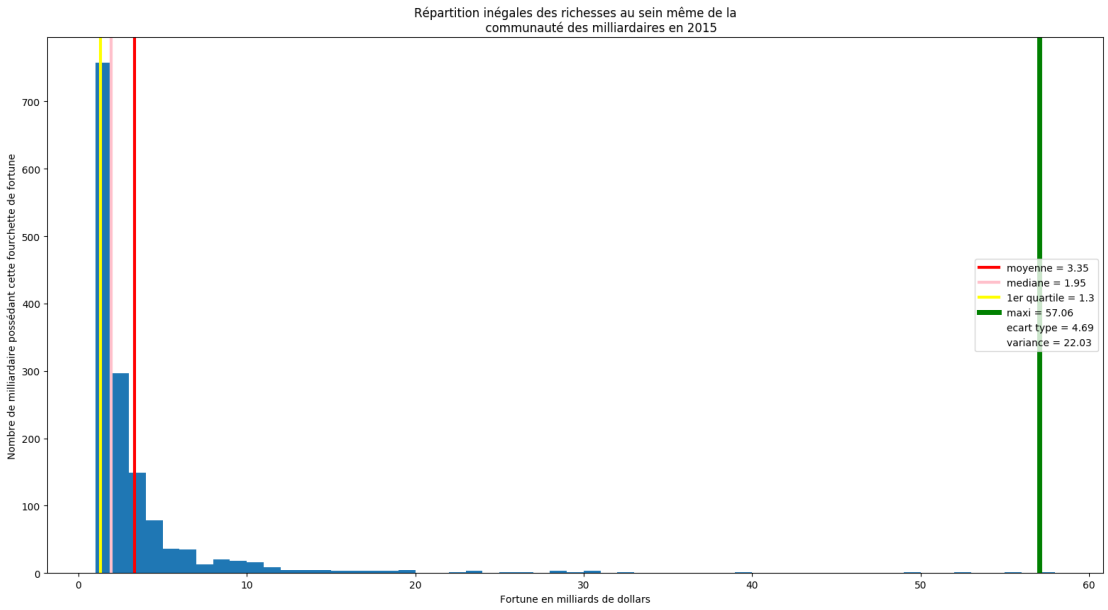

FIGURE 5.1 – Histogramme

6 Conclusion

Ce projet m'a permis d'aller plus en profondeur dans python que j'avais pu découvir en première lors d'une option. Ce langage convient parfaitement à ce que j'aime en développement, créer diverses choses dans différents domaines. J'ai donc pu en parallèle de l'unité faire une étude de population mettant en oeuvre les connaissances que j'ai pu acquérir en mathématiques, et dont les calculs étaient faits en python, mais aussi me lancer dans Django, un framework web très puissant, basé sur python. Enfin il m'a permis d'acquérir les bases de Tensorflow, une technologie développée par Google pour faciliter l'utilisation du machine learning.

FIGURE 6.1 – The web framework for perfectionists with deadlines

FIGURE 6.2 – An open-source software library for Machine Intelligence