Rapport projet Open Data RATP

Introduction

Nous avons décidé de réaliser notre projet Open Data avec les données fournies par la RATP. Après analyse des différentes données présentes sur le site, nous décidons de réaliser un histogramme représentant le nombre de stations RATP selon leurs trafics. Ainsi, sur l'axe des abscisses on peut retrouver le trafic par pas de 5 millions et sur l'axe des ordonnées on trouve le nombre de stations pour lesquelles le trafic par année est compris dans le pas de trafic indiqué sur l'axe des abscisses. Nous voulions réaliser cela pour les 4 années disponibles (2013, 2014, 2015, 2016), or cela s'est avéré être inutile. Nous verrons pourquoi dans la partie « Histogramme ».

Concernant la représentation géographique, nous avons décidé d'utiliser les données de la RATP traitant l'accessibilité des gares pour les personnes à mobilité réduite. L'avantage que présente ce sujet est que les coordonnées des gares analysés sont déjà précisées dans le fichier csv rendu disponible par la RATP. Or la latitude et la longitude sont contenues dans un seul et même champ. Il faudra donc séparer en deux ce champ afin d'en extraire la latitude et la longitude mais nous y reviendrons plus en détails dans la partie « Carte » de ce rapport.

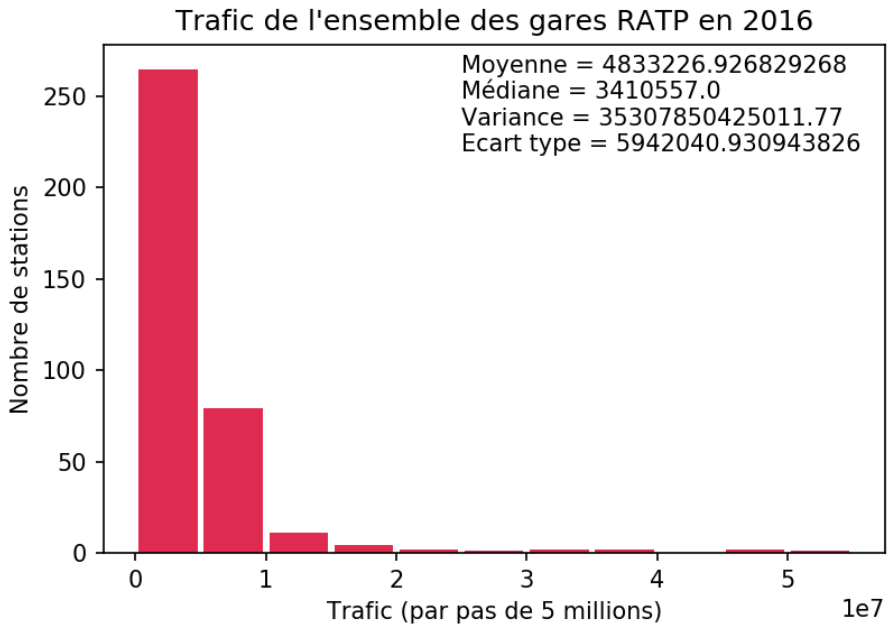
En outre, nous avons voulu réaliser des « bar charts » ou diagramme en barres. Nous souhaitions rajouter à l'histogramme et à la carte des « bar charts » représentant le trafic pour les plus grosses gares du réseau RATP en termes de trafic. Plus exactement nous avons réalisé 3 bar charts. Le premier représentant le trafic des cinq plus grosses gares du réseau RER RATP en termes de trafic sur quatre ans (2013, 2014, 2015, 2016). Le deuxième représentant le trafic des cinq plus grosses gares du réseau métro RATP en termes de trafic sur les quatre mêmes années. Enfin le dernier bar charts représente le mélange des deux premiers bar charts avec le trafic des cinq plus grosses gares de l'ensemble du réseau RATP en termes de trafic sur les quatre mêmes années.

Nous avons également fait un diagramme ou « pie chart » représentant la proportion de gares du réseau RATP accessible pour les personnes à mobilité réduite en pourcentage.

Pour finir nous voulons afficher l'ensemble de ces éléments sur un site web conçu par nous-même.

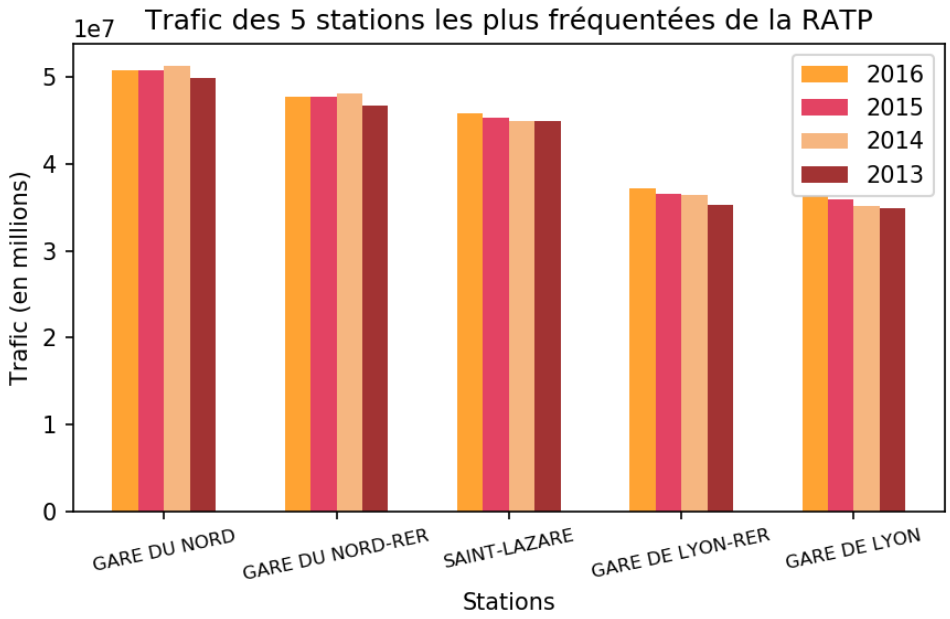
Figures commentées

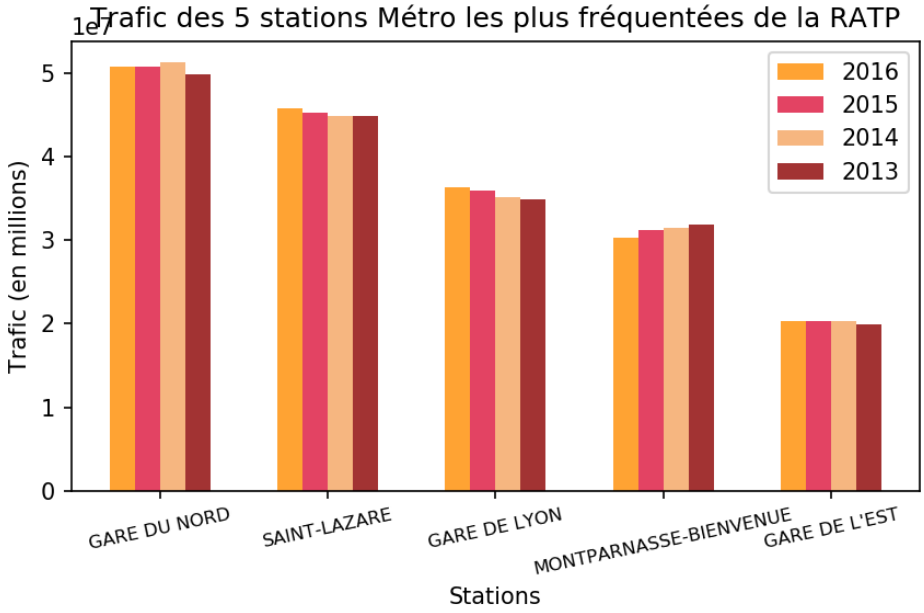
Histogramme :

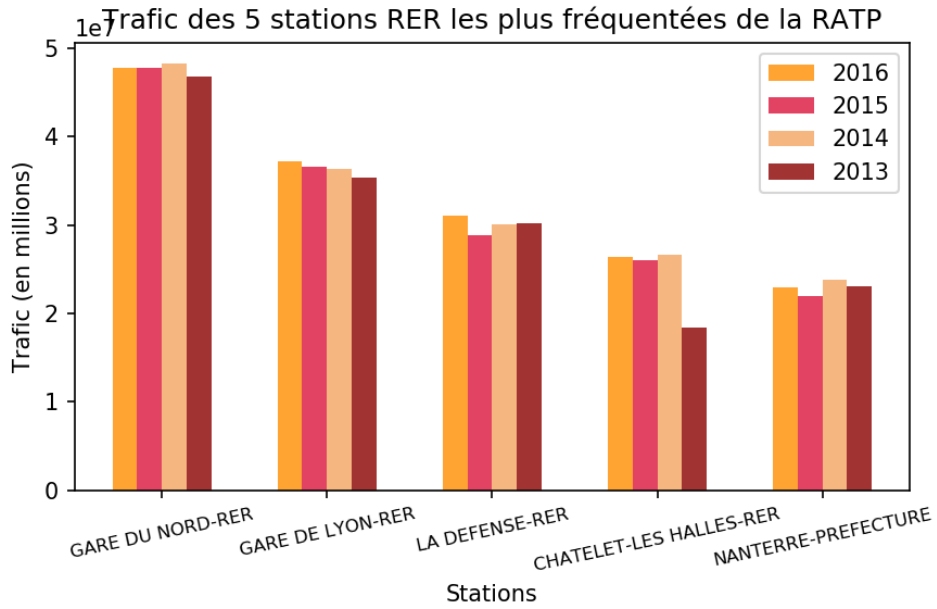


Notre histogramme permet de montrer que la plus grande majorité des gares du réseau RATP ont un trafic compris entre 0 et 10 millions de voyageurs par an dont la majorité a un trafic compris entre 0 et 5 millions. Les gares avec un gros trafic annuel (supérieur à 10 millions) représentent une part très faible du réseau.

Bar charts :

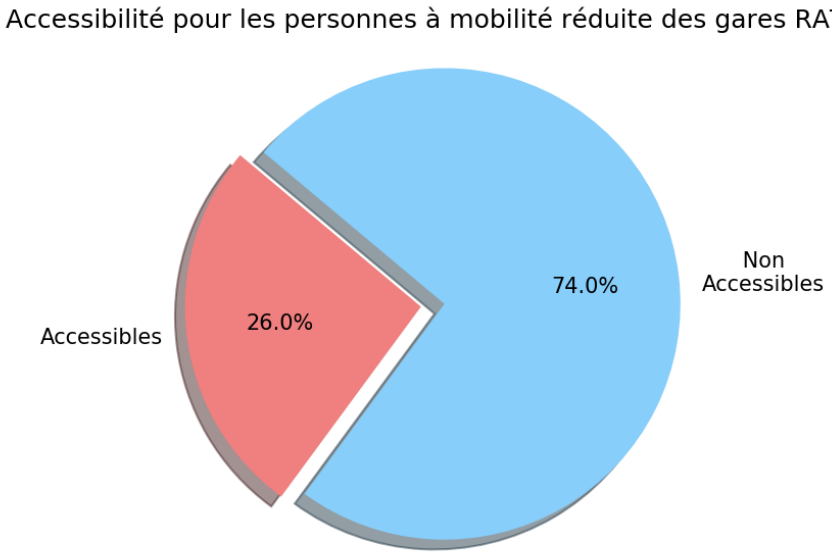






Concernant les bar charts, ils peuvent nous permettre de nous apercevoir qu'entre 2013 et 2016, pour les cinq plus grandes gares en termes de trafic du réseau RATP, le trafic a augmenté pour la majorité des gares que ce soit sur le réseau métro ou RER. Les bar charts nous permettent également de remarquer que globalement le RER est autant utilisé en termes de trafic que le métro sur les cinq plus grandes gares de chaque réseau.

Pie chart :



Le pie chart quant à lui nous permet tout simple de voir qu'une grande majorité des gares du réseau RATP (74%) sont n'ont pas d'accès spécifiques pour les personnes à mobilité réduite. Seuls 26% en sont équipés.

Récupération des fichiers

Afin de réaliser l'ensemble des figures, il nous faut récupérer les données. Pour ce faire, nous décidons de créer une classe « Telechargement » prenant comme arguments une source et une destination et qui va nous permettre de télécharger l'ensemble des fichiers nécessaires depuis la source indiquée (url) et de les sauvegarder à la destination voulue. Pour ce faire nous avons utilisé le module urllib.

Extraction des données

Après avoir téléchargé les données, nous devons extraire les données qui nous intéressent depuis les fichiers csv. Pour cela, nous avons créé une classe « Extract » qui va gérer les données au sein des fichiers csv. Au sein de cette classe, nous avons écrit plusieurs fonctions permettant de traiter les données pour les différentes représentations. Par exemple, « build_dict_trafic » permet de construire à partir d'un fichier csv passé en argument, un dictionnaire ayant pour clé le nom de la station RATP et pour valeurs son trafic, son rang en termes de trafic mais également la ville dans laquelle est située la gare.

La fonction « get_trafic » renvoie la liste contenant le trafic de chaque gare du fichier csv passé en paramètre pour la fonction « build_dict_trafic » car elle utilise le dictionnaire renvoyé par cette dernière.
La fonction « build_dict_accessibilite » fonctionne selon le même principe que « build_dict_trafic » mais en renvoyant un dictionnaire ayant pour clé le nom de la station et pour valeurs, l'accessibilité de la gare, sa latitude et sa longitude. Ce qui diffère contrairement à la fonction « build_dict_trafic » c'est le fait que la latitude et la longitude sont contenues dans le même champ comme expliqué dans l'introduction. Afin d'avoir en valeur la latitude et la longitude, nous avons utilisé la fonction split sur le champ des coordonnées. Nous avons aussi dans cette fonction remplacé l'encodage qui n'affichait pas les accents des stations mais des caractères spéciaux illisibles.
Enfin, la fonction « get_accessibilite » fonctionne de la même manière que la fonction « get_trafic » mais en retournant la liste de l'indicateur d'accessibilité pour chaque station dans le fichier csv (si indicateur >= 1, alors la station est accessible).

Histogramme

La classe « Histogramme » contient une fonction « genereHistogramme » nous permettant de générer notre histogramme et de le sauvegarder en tant qu'image afin de pouvoir l'afficher sur notre site. La classe prend en paramètre les données en abscise et en ordonnées (ici le trafic et par pas de cinq millions et le nombre de stations), mais aussi les légendes des axes et le titre de l'histogramme. Afin de réaliser l'histogramme, nous utilisons « matplotlib ».
La classe « Histogramme » contient aussi les différents calculs que l'on doit effectuer à partir de l'histogramme et des données (moyenne, médiane, variance et écart type). Pour ce faire nous avons utilisé le module « numpy » qui possède des fonctions pour chacun de ces calculs.

Bar chart

La classe « Bar » fonctionne globalement de la même manière que la classe « Histogramme ». On y retrouve une fonction « genereBar » nous permettant de générer notre bar chart et de le sauvegarder en tant qu'image. La classe prend en paramètre quatre listes contenant ici le trafic pour les quatre années de 2013 à 2016 pour les gares qui nous intéressent, le cinquième paramètre correspond aux données sur l'axe des abscises (ici la liste des stations évoquée auparavant). Enfin on prend en paramètre les légendes des axes et le titre du bar chart. Afin de le réaliser, nous utilisons le module « matplotlib ».

Pie chart

Pour finir les représentations, nous avons réalisé une classe « Pie_Chart ». Elle fonctionne également avec le module « matplotlib ». La fonction « generePieChart » nous permet de sauvegarder sous forme d'image notre diagramme. Cette classe prend en paramètre le titre du diagramme, les légendes mais également les données pour permettre de représenter ce diagramme (ici la liste de l'indicateur d'accessibilité des stations RATP). (Pour rappel : si indicateur >= 1, alors la station est accessible).

La carte

Pour générer la carte nous avons créé une classe « Map_Idf ». Cette classe prend comme arguments la liste des stations que nous voulons afficher sur la carte mais aussi le dictionnaire généré par la fonction « build_dict_accessibilite » dans la classe « Extract » afin d'avoir les coordonnées des stations mais aussi savoir si ces gares sont accessibles. Pour réaliser la carte nous utilisons Folium. Folium nous permet de centrer la carte sur des coordonnées précises avec un zoom. Folium utilise les icones de Bootstrap, ce qui nous a permis de mettre sur la carte, avec les coordonnées des stations recueillis dans le dictionnaire passé en paramètre, des icones « valider » de couleur verte si la station est accessible ou des icones « croix » de couleur rouge si la station n'est pas accessible aux personnes à mobilité réduite. Nous avons également ajouté un « popup » qui permet que lorsque l'on clique sur les icones le nom de la gare apparait. Pour finir, nous sauvegardons cette carte sous forme de fichier HTML afin de la réutiliser sur notre site final.

Le main

Le fichier main.py contient l'ensemble des appels de fonctions contenus dans les différentes classes. Il contient également les lignes de code permettant l'installation de folium, la création des listes nécessaires à l'appel des fonctions. Pour finir, la dernière ligne permet d'ouvrir l'index de notre site web automatiquement.

Difficultés rencontrées

La première difficulté que nous avons eu à affronter est le fait de ne pas pouvoir utiliser « basemap ». En effet, notre région à étudier pour la carte (l'Ile-de-France) était trop précise pour basemap qui n'affichait pas en assez haute qualité la carte. Ainsi, nous avons dû utiliser « folium ».

La deuxième difficulté a été d'ajouter une légende à notre carte folium. Nous ne trouvions pas de module folium nous permettant de le réaliser. Nous avons donc décidé d'ajouter cette légende par HTML directement afin d'être surs que notre carte serait lisible et compréhensible.
La troisième et plus grosse difficulté que nous ayons eu à affronter lors de ce projet est le fait que folium fonctionne en HTML. Ce qui signifie que nous devions ouvrir la carte avec une page HTML en local. Hors Microsoft Edge (navigateur par défaut sur les PC de l'ESIEE) n'arrive pas à lire ce genre de fichier. De plus, nous souhaitons afficher l'ensemble de nos résultats sur un site internet en local. Ainsi, nous avons cherché une solution à ce problème en tentant de modifier le navigateur par défaut par ligne de code ou encore en cherchant dans la collection des navigateurs présents sur l'ordinateur toujours en ligne de code, mais aucune de ces solutions n'a fonctionné afin d'ouvrir notre site sous Google Chrome ou Mozilla Firefox. La solution que nous avons trouvée est d'utiliser « webbrowser ». On import webbowser puis, si Google Chrome n'est pas parmi la collection des navigateurs (webbrowser._tryorder) on l'ajoute grâce à la ligne de commande (webbrowser.register) puis on lance notre site local regroupant l'ensemble de nos représentations générées par notre code. Nous avons choisi Google Chrome car c'est un navigateur qui est sur les machines de l'ESIEE et sous lequel notre site local s'affiche parfaitement bien.

Conclusion

Pour conclure, notre carte permet d'illustrer ce que montre le diagramme mais de manière plus visuelle et plus contextualiser. En effet, on peut voir grâce à la carte que la plupart des gares accessibles aux personnes à mobilité réduite se situent hors de Paris. Celles qui le sont au sein de Paris sont généralement les plus grosses gares du réseau comme les gares par exemple (Gare du Nord, Gare de Lyon, etc ..).

Les commentaires des figures ont été réalisé ci-dessus dans la partie « Figures commentées ».

Vous pouvez retrouver une partie « A propos » à la fin de notre site local, en dessous de la carte expliquant pourquoi nous avons choisi ces données et ces représentations.