Université Pierre et Marie Curie - Paris 6

Rapport de projet

# Fréquentation des stations de transport et clustering spatial

*ABDOU Nadir, NGUYEN Antoine, SANCHEZ Jacobo*

Encadré par
BASKIOTIS Nicolas, GUIGUE Vincent

8 septembre 2017

**Résumé**

L'objectif de ce projet est de maîtriser le partitionnement de données, le *clustering* en apprentissage non-supervisé. Pour ce faire, nous utiliserons l'algorithme des k-moyennes. Les données utilisées sont issues du STIF, ce sont les nombres de validations des pass Navigo par minute et par station, pour toutes les stations de métro de Paris. Il s'agira d'abord de nettoyer ces données en les lissant grâce à une fenêtre passante, puis de les normaliser quantitativement. Ensuite, l'application de l'algorithme des k-moyennes avec différents $k$ nous donnera différents "bon" $k$ chacun menant à des conclusions diverses. Nous vérifirons enfin les différents résultats grâce à un calcul de pureté sur l'algorithme.

# Table des matières

# 1 Introduction

Chaque jour, environ 10 millions de personnes empruntent le réseau de transports publics en Île-de-France. À l'aide de données fournies par le STIF, l'autorité organisatrice des transports de la région, nous tenterons d'analyser les comportements des usagers afin d'extraire des informations précises sur les stations de métro/RER.

Au début du projet, nous avions deux ensembles de données provenant d'un partenariat entre nos professeurs encadrants et le STIF. Le premier ensemble concerne 316 stations de métro dans Paris. Chaque station possède un identifiant, un nom et une localisation géographique. Le second ensemble indique le nombre de *logs* (validations du pass Navigo) par minute et par station sur une période s'étendant du 1er Octobre 2015 au 31 Décembre 2015 (soit 92 jours). Nous pouvons facilement tracer les graphes des stations, en représentant les *logs* en fonction du temps. On parlera désormais de signal.

À travers différents outils et techniques que nous présenterons par la suite, nous allons classer les stations en groupes, que nous appellerons *clusters*. Nous allons donc faire du *clustering*, c'est-à-dire une analyse de partitionnement de données ayant pour but de classer des éléments - ici les stations de métro - en fonction de leur contenu, leur signal. Les avantages d'un tel partitionnement sont nombreux : connaître les stations semblables pour des fins d'optimisation des services, d'assistance aux usagers, de marketing, etc ... La principale motivation de ce projet est la compréhension d'un système complexe : comment passer de données brutes à des résultats qualitatifs. En analysant ces données, nous comprendrons les comportements journaliers des usagers et *a posteriori*, nous connaîtrons les axes d'amélioration du réseau du STIF. Comprendre les comportements journaliers nous permettra également de détecter facilement des anomalies : pourquoi ce lundi est-il différent de tous les autres ? Comment l'expliquer ? Etc.

Un traitement des données au préalable est néanmoins nécessaire. Les données brutes comportent un bruit important. Le bruit est une variance, une erreur aléatoire d'une variable mesurée. Ici, cela peut être dû à des pannes de senseurs, des stations fermées, des fraudes des usages, etc ... Aussi, le fait qu'un utilisateur valide son pass navigo à 15h01 ou qu'il le valide à 15h03 revient sensiblement à la même chose : il s'agira donc de lisser les données. Ensuite, nous procéderons à leur tri à travers le *clustering*. Finalement nous analyserons les résultats. Leur interprétation permttra de tirer des conclusions et définir de nouvelles expériences pour approfondir nos interprétations.

Dans une première partie, nous présenterons les modèles mathématiques utilisés. Ensuite, nous détaillerons notre processus expérimental.

# 2 Modèles mathématiques utilisés

## 2.1 Lissage

Par définition, le lissage est une technique qui consiste à réduire les irrégularités et singularités d'une courbe. Ainsi, il permet de supprimer le bruit d'un signal.

Il existe plusieurs techniques de lissage, dont celle d'ajustement de courbe, qui consiste à construire une courbe à partir d'une fonction définie et la modifier afin de se rapprocher autant que possible de la courbe expérimentale (en fixant un seuil de ressemblance par exemple). Ceci peut être fait avec la méthode de Gauss-Newton. Il existe également la technique de la moyenne mobile, qui dont le principe est de définir une fenêtre autour de la mesure à traiter, avec divers paramètres. On peut décider de faire une une moyenne exponentielle, ou une moyenne pondérée en fonction des mesures avoisinantes.

## 2.2 K-moyennes

L'algorithme des k-moyennes est une méthode de partitionnement où $k$ représente le nombre de *clusters* à construire. On créé $k$ *clusters*, dont le centroïde (c'est-à-dire la moyenne du *cluster*) est initilisé aléatoirement, et on calcule la distance avec différentes métriques, telle que la norme euclidienne, entre chaque signal et les $k$ centroïdes. On retient la plus petite distance, pour inclure le signal correspondant dans le *cluster* au centroïde le plus proche. À chaque itération de l'algorithme, on calcule le nouveau centroïde de chaque *cluster*. L'algorithme s'arrête lorsque les *clusters* construits convergent.

Cet algorithme est relativement simple, et efficace en temps et en mémoire. Il est utilisable avec de grandes bases de données mais est limité car il dépend toujours du choix l'initialisation, qui est aléatoire.

**Algorithme :**

- Choisir aléatoirement les $k$ signaux qui représentent les k centroïdes.
- Répéter jusqu'à convergence :
  - Assigner chaque signal au centroïde le plus proche.
  - Mettre à jour la moyenne de chaque *cluster*.

La convergence est atteinte lorsqu'il n'y a plus de changements, ou bien que les changements sont suffisamment faibles, soit en-dessous d'un certain seuil.

## 2.3 Pureté

La pureté est utilisée pour évaluer la qualité et l'efficacité de l'algorithme des k-moyennes, c'est une mesure d'évaluation simple et transparente. Ainsi, plus la pureté est élevée, plus les *clusters* sont fiables et correspondent aux groupes standards.

Il est facile d'obtenir une pureté élevée lorsque l'on étudie un grand nombre de *clusters*. En particulier, la pureté est de 1 si chaque élement étudié possède son propre *cluster*.

La pureté d'un *cluster* se définit comme :

$$purity(\Omega, C) = 1/N \sum_k \max_j |w_k \cap c_j| \tag{1}$$

$$\text{où : } \Omega = \{w_1, w_2, ..., w_k\} \text{ et } C = \{c_1, c_2, ..., c_j\}$$

Pour calculer la pureté, on utilise l'algorithme des k-moyennes avec les mêmes données qu'auparavant, mais en les considérant différemment. En effet, en plus de considérer chaque station comme un élément distinct à classer, on prend désormais en compte les jours. On passe ainsi de 316 éléments sur lesquels on fait du *clustering* à $316 \times 92$, soit 29072 éléments, que l'on étale sur 288 échantillons de temps (correspondant au nombre d'échantillons réalisés en une journée). Lorsque l'algorithme s'arrête, on obtient des *clusters* avec les identifiants des stations ainsi que la date précisse correspondante. Cette dernière devra être supprimée afin de n'avoir que l'identifiant de la station, ce qui nous facilitera la tâche lors du calcul du nombre de signaux de la classe majoritaire.

Enfin, on somme le nombre de signaux de la classe majoritaire de chaque *cluster*, résultat que l'on divise par le nombre total de signaux : cela nous permet d'obtenir la valeur de la pureté.

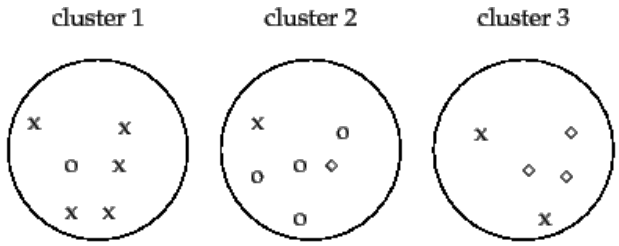


FIGURE 1 – Exemple de calcul de pureté comme critère d'évaluation de la qualité d'un cluster.

Ainsi, comme on peut le voir sur la figure 1 ci-dessus, la majorité de chaque *cluster* est respectivement "x" qui apparait 5 fois, puis "o" qui apparait 4 fois et "◇" qui apparait 3 fois. Il y a 17 éléments au total dans ce *clustering*.

On procède au calcul de la pureté : $(1/17) \times (5 + 4 + 3) = 0.71$.

# 3 Partie expérimentale

## 3.1 Étude quantitative directe

Initialement, en utilisant les données brutes fournies par le STIF, il est possible de dresser une carte triant les stations en fonction de leur flux. Cela n'apporte que peu d'informations et ne donne qu'une idée générale des stations dans l'échantillon que nous étudions. Aucun paramètre d'étude ne s'offre à nous, si ce n'est la variation de l'échelle des flux, permettant de mieux déterminer les stations les plus fréquentées.

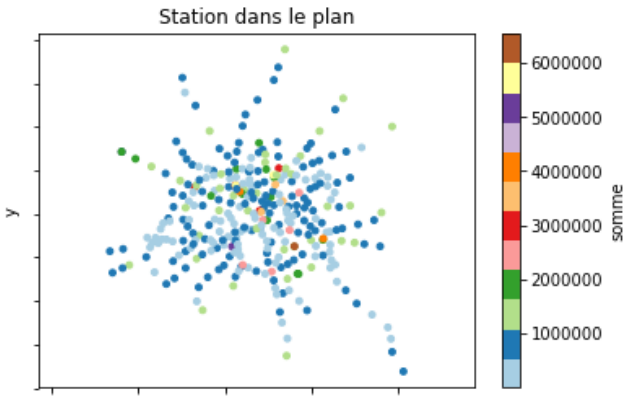


FIGURE 2 – Représentation spatiale et quantitative des stations de l'échantillon

Ainsi, comme cité lors de l'introduction, les données brutes proposées par le STIF sont loin d'être exploitables correctement. Il est nécessaire de les traiter.

## 3.2 Analyse, lissage et normalisation

Au départ, nous avons une matrice indiquant les *logs* des utilisateurs (rendus anonymes) ainsi que leur lieu. Après une numérisation avec un échantillonage toutes les 5 minutes, nous obtenons un signal sur la période étudiée de 92 jours.

**Analyse.** Lorsque l'on prend le métro quotidiennement, nos horaires sont les mêmes à un intervalle de temps près. Même si tous les jours, il nous faut arriver à notre lieu de travail à la même heure, on ne part jamais exactement à la même heure de chez soi. On observe donc des phénomènes locaux sur les signaux qui empêchent une lecture correcte des phénomènes intéressants. Ainsi, on peut observer sur le premier graphique de la figure 3 (ci-dessous) une abérration au niveau du 60ème jour qui est exprimée par un pic très grand. Celle-ci fausse l'échelle de ce premier graphique, et nous empêche d'apprécier certaines particularités du signal. Sur le deuxième graphique, on peut observer une aberration le même jour, mais bien moins importante. En tant qu'experts, il nous faut alors réaliser une tâche qui ne peut être automatiée : nous devons repérer et trier ces anomalies.

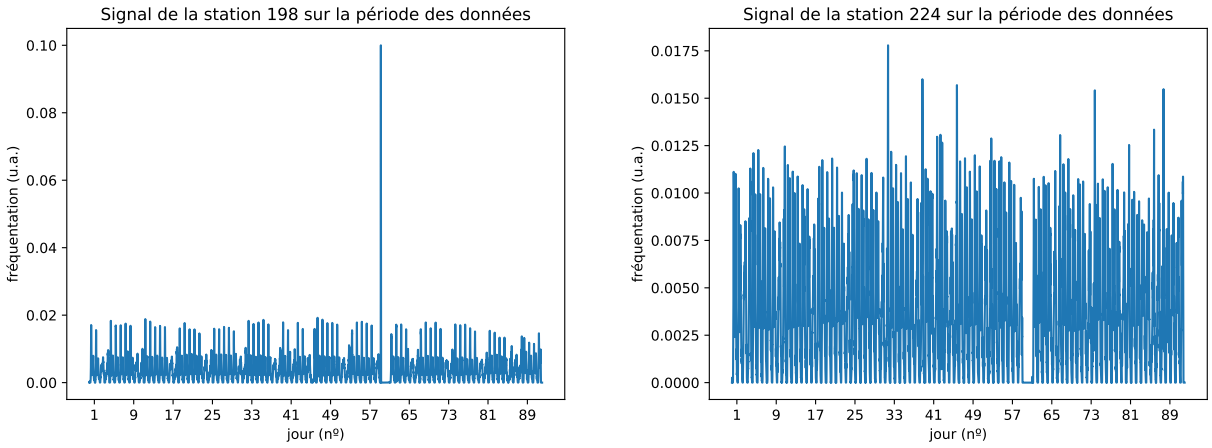


FIGURE 3 – Deux signaux de stations sur la période entière des données

**Lissage.** On peut décider de traiter le signal sur la période entière, d'établir une moyenne sur tous les jours de la période, ou seulement les jours de la semaine et ceux du week-end d'un autre côté.

On observe d'ailleurs déjà des phénomènes locaux sur la figure 3 ci-dessus sur une période de 7 jours, soit la semaine. En effet, on remarque une activité bien plus importante les jours ouvrés que le week-end.

Lors de l'étude des données sur une période, pour palier le bruit dû aux évènements locaux, on procède à un lissage. Il existe de très nombreuses techniques de lissage, mais nous avons choisi de nous concentrer seulement sur un filtre passe-haut, après avoir essayé d'autres techniques comme la moyenne pondérée ou l'approximation polynomiale. Après plusieurs tests, nous avons décidé d'utiliser pour nos expériences une fenêtre de 5, qui correspond à l'intervalle de temps entre 10 minutes avant l'entrée de l'utilisateur et 10 minutes après. De plus, nous avons observé que les jours de la semaine étaient bien moins bruités que le week-end, les habitudes et surtout contraintes de déplacement étant beaucoup plus strictes, ce qui nous permet de distinguer des phénomènes plus nets.

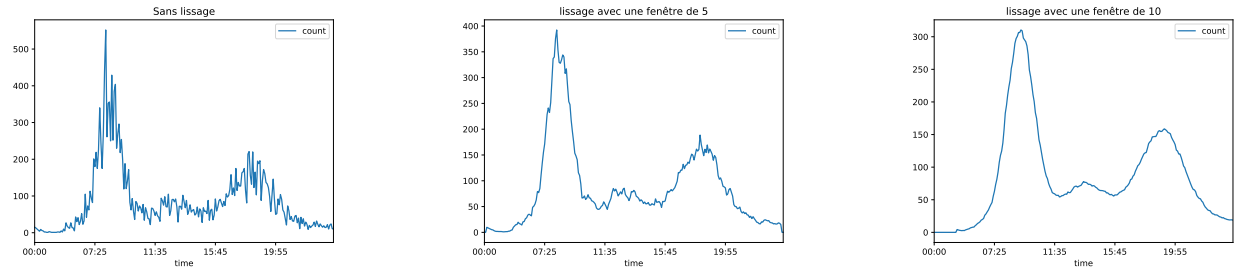


FIGURE 4 – Signal de la station 198 à un jour donné sans lissage, avec un lissage de fenêtre 5, puis de fenêtre 10.

**Normalisation.** Comparer des stations suivant leur fréquentation est simple et peu intéressant dans notre cas. Nous normalisons donc nos données, ce qui consiste à diviser chaque *logs* d'une journée par la somme des *logs* de cette journée pour une station. Cela nous permet de comparer les stations entre elles en fonction de leurs allures générales, ce qui aurait été impossible avec des stations ayant de grandes différences de fréquentation. Plus exactement, cette opération réduit le signal à une densité de probabilité.

### 3.3 Clustering

Après ce premier traitement des données, il est désormais possible d'utiliser notre algorithme de *clustering*. On décide de mener diverses expériences, comme faire varier $k$ de 2 à 15, ou changer la métrique de comparaison des données.

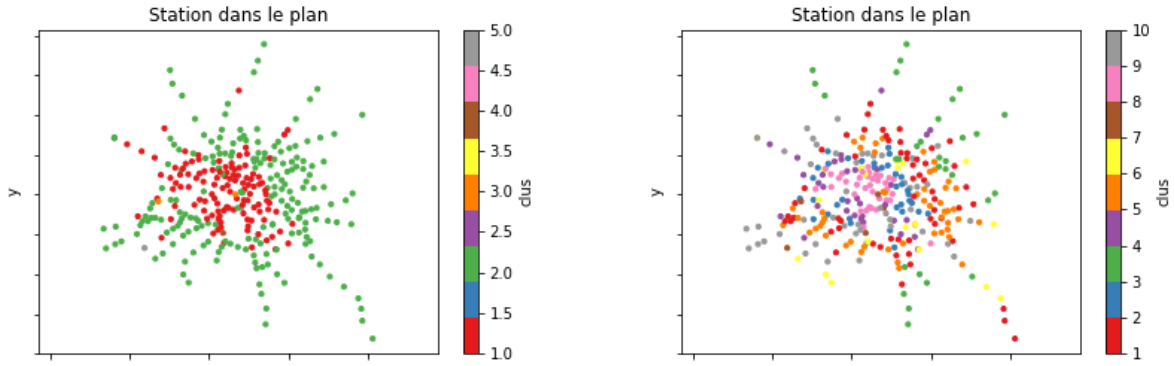


FIGURE 5 – Affichage des stations pour $k = 5$ puis pour $k = 10$

On observe ainsi quelques effets du *clustering*. En effet, si l'on prend un $k$ petit, comme $k = 5$, le *clustering* est trop large. Certains *clusters* sont vagues, ce qui apporte peu d'informations distinctives. On observe aussi que suite à l'initialisation, l'algorithme trouve une meilleure répartition avec 3 *clusters* même si 5 sont disponibles. Ainsi on finit avec 3 groupes, un englobant les stations de banlieue (en vert), un autre englobant celles à l'intérieur de Paris et finalement un troisième groupe très réduit, dont Châtelet, de stations au comportement trop différent du reste. On remarque sur le second graphique de la figure 5 qu'un *clustering* avec $k = 10$ est aussi très intéressant : elle permet de bien faire ressortir les différences, sans être trop précise. On peut par exemple distinguer certaines lignes de métro.

Il est intéressant de noter qu'à partir d'un certain seuil, dans notre cas $k = 10$, les *clusters* semblent converger : peu de choses changent, si ce n'est l'apparition de *clusters* particuliers ne regroupant qu'une ou peu de stations. Lorsque les *clusters* sont trop particuliers, il est impossible d'obtenir les caractéristiques des stations ou d'en tirer des conclusions.

Pour vérifier notre *clustering*, nous pouvons comparer le *centroïde* avec deux stations aléatoires du même *cluster*. Comme on peut l'observer sur la figure 6, même si les signaux rouges ne sont pas identiques au signal vert, ils restent tout de même très proches du centroïde.

Il est possible de comparer les centroïdes entre eux, afin d'isoler les différences entre chaque *cluster*,

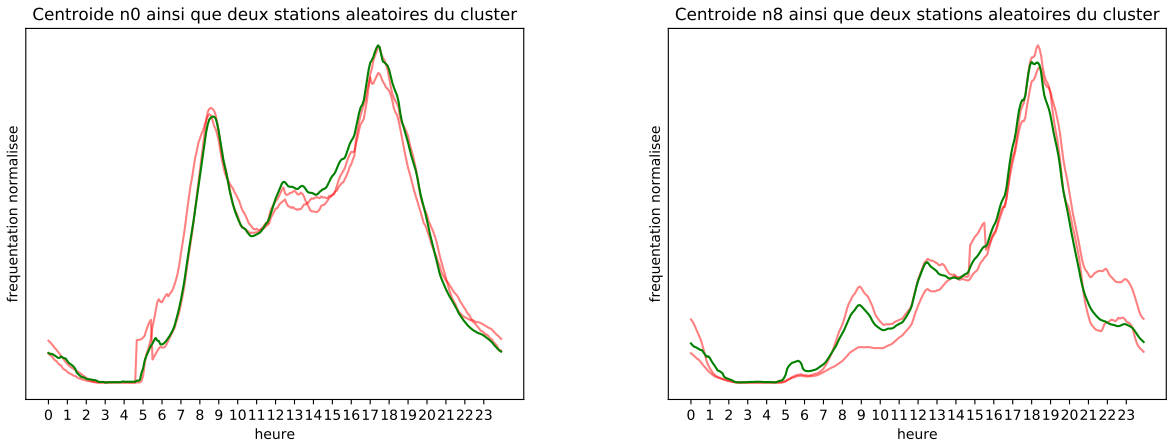


FIGURE 6 – Affichage du centroïde de deux *clusters* (en vert) ainsi que des signaux de deux stations aléatoires (en rouge), pour $k = 10$

comme on peut le voir sur la figure 7. Cette comparaison est relativement bruitée mais nous permet d'observer deux différences importantes entre les *clusters*. La première, évidente, est que les stations du *cluster* 0 ont une activité bien plus importante le matin que celles du *cluster* 8. La deuxième, plutôt discrète, souligne que même si les deux *clusters* ont une activité importante vers 18h, cet horaire comprend la majorité de la fréquentation des stations du *cluster* 8 et a donc un poids beaucoup plus important. Par conséquent, on observe un pic négatif sur le graphique de comparaison. Cela peut être interprété comme un rappel que nous n'étudions plus que des signaux normalisés.

On peut donc en conclure que les stations du centroïde 0 sont à proximité d'habitations et de lieux de travail, tandis que celles comprises dans le centroïde 8 ne sont à proximité que de lieux de travail.

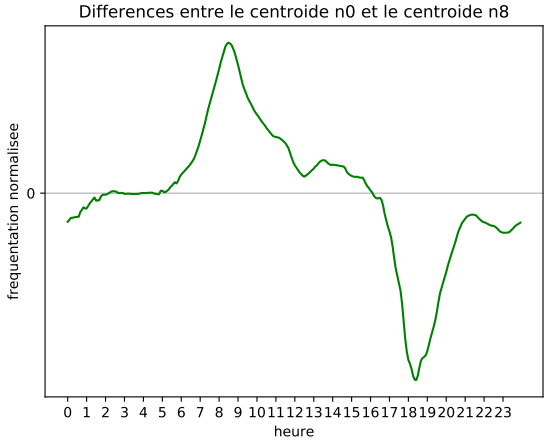


FIGURE 7 – Affichage des différences entre deux centroïdes, pour $k = 10$

### 3.4 Pureté

En lançant l'algorithme de *clustering* pour calculer la pureté, on l'exécute sur les données station-jour. On remarque que le nombre de signaux est de l'ordre de 29072. Pour k=100, l'algorithme calcule la distance entre chaque signal et le centroïde d'un *cluster*. Cette opération est répétée jusqu'à convergence (avec un seuil, que l'on fixe à 15 itérations).

Ainsi, une itération étant déjà assez couteuse, en faire 15 ralentit considérablement notre algorithme de *clustering*. En effet, l'algorithme prend environ 1 minute pour 316 signaux et $k = 10$. Il va donc prendre énormément de temps pour le calcul de la pureté. Ceci a donc posé un obstacle considérable dans nos expériences. Une implémentation différente a été nécessaire pour le surmonter.

Pour $k = 100$, nous avons obtenu une pureté de 0.08, qui est une valeur assez faible. Ceci est toutefois logique puisque nous avons une très grande quantité de signaux et relativement peu de *clusters*, ce qui les force à être trop généraux. Ainsi, comme nous avons vu lors de nos expériences, une pureté plus grande sera obtenue avec un nombre plus important de *clusters*.

On peut s'attendre à obtenir des puretés importantes lorsque l'on lance l'algorithme de *clustering* sur autant de *clusters* que de jours de la semaine et de stations, soit $k = 316 * 7 = 2212$.

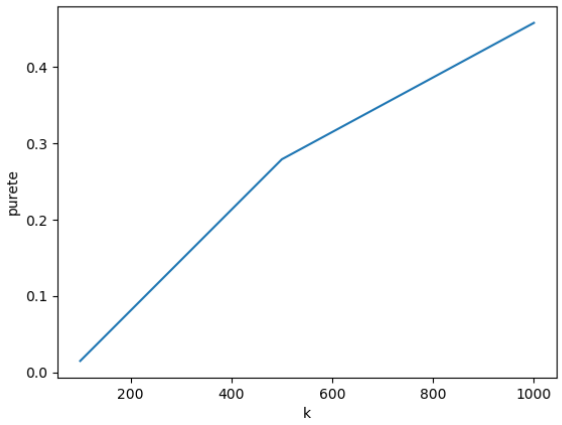


FIGURE 8 – Pureté du *clustering* en fonction de $k$

### 3.5 Bilan de la partie expérimentale

Après être devenus familiers avec les données et avoir choisi une stratégie pour les traiter, nous avons mis en place un *clustering* avec l'algorithme des k-moyennes. Cela nous a permis de comprendre les comportements des diverses stations dont on avait les *logs*, ce qui aurait été impossible sans les techniques mises en place. Nous nous sommes heurtés à plusieurs obstacles, comme la forme des données que nous avions, mais aussi l'implémentation des algorithmes, qui nous ont forçés à changer de stratégie, jusqu'à trouver la plus adaptée.

# 4 Conclusion

Ce projet nous a permis de mettre un premier pied dans l'exploitation de données, le *data mining*. Nous avons tout d'abord nettoyé les données, après les avoir rapidement supervisées afin d'éliminer les abérations notables. Nous les avons ensuite traitées, en utilisant un lissage et une normalisation. Ensuite, nous avons utilisé l'algorithme des k-moyennes commme algorithme de partitionnement des données. Non seulement nous avons établi des *clusters* mais nous les avons également analysés. D'une part en tant qu'expert, en se posant des questions telles que : "Est-ce que la station 198, soit Gare d'Austerlitz, peut avoir ce type de profil ?". D'autre part, en mesurant la pureté du *clustering* avec un grand nombre de *clusters*. La partie expérimentale nous a aidés à choisir les valeurs des différents paramètres de l'algorithme.

On pourrait continuer notre démarche en exploitant d'avantage les profils des stations (obtenus à travers les centroïdes), afin d'établir des comportements habituels de station. Aussi, on pourrait changer totalement de voie, et appliquer ces expériences de *clustering* aux utilisateurs. Sur les stations ou les utilisateurs, une analyse plus importante des clusters et des règles logiques permettrait de cerner un comportement *normal* et donc de repérer immédiatement des anomalies. Mais on pourrait aller plus loin, et voir l'impact de ces anomalies sur le réseau de transport. Ainsi, on pourrait prédire où est-ce qu'il y aura une importante activité inhabituelle en fonction de différentes anomalies et adapter le réseau en fonction du modèle.

Une autre application intéressante serait de faire du *clustering* avec d'autres jeux de données totalement différentes, telles que le revenu ou l'activité de la zone à proximité de la station (données obtenues à partir de l'INSEE par exemple). Après avoir fait une classification indépendante des données, il serait intéressant de les croiser pour aboutir à des conclusions démographiques soulignant le lien entre les données de l'INSEE et les habitudes de transport. Ceci pourrait aider au développement de nouvelles zones ou à des réorganisations diverses du réseau. Si les données distinguaient les pass Navigo des pass Imagine'R, il serait aussi possible de conclure sur les habitudes différentes entre les étudiants et le reste de la population possédant un badge.

# Références


- Code disponible sur : `https://github.com/Eastkap/uni/tree/master/L2/2i013`
- Pattern classification, Duda Hart.
- The element of statistical learning, Hastie, Tibshirani