

Projet IND6212 : segmentation de lignes d'un réseau de transport collectif

Imed Essid / Houssem Bouazizi/ Amadou Mbagnick Sarr

1- Introduction

Être capable d'identifier les lignes similaires d'un réseau de transport collectif peut avoir son importance. De plus, dans certains cas, les planificateurs de systèmes de transport en commun ont besoin d'évaluer davantage la prestation globale des services de transport en commun aux fins d'examen, de planification et de prise de décision ; une vision globale de l'opération est ainsi très importante voire obligatoire. Segmenter les lignes en fonction de leurs caractéristiques opérationnelles peut offrir une perspective perspicace, qui peut conduire à une découverte de différentes catégories ou un ensemble d'opérations de lignes. Beaucoup de sociétés de transport de bus telles que la STM ont procédé à une segmentation de leurs lignes en sous réseaux, chacun ayant des caractéristiques bien définies (Réseau local, lignes express etc.). Dans ce cadre, on propose d'étudier une base de données de lignes de transport de la société EXO -secteur Laurentides qui comporte un total de 158 lignes caractérisées par 24 colonnes. Le but de ce travail est d'aider les gestionnaires et les différents intervenants dans le domaine du transport par bus d'effectuer des planifications stratégiques et opérationnelles en se basant sur les groupes de lignes similaires c'est ceux qui appartiennent à la même classe.

2- Création et présentation de la Base des données

La base de données utilisée pour ce travail est créée par nous même à partir de l'ensemble des horaires planifiés de la Société de transport EXO pour le secteur Laurentides sous format GTFS (General Transit Feed Specification).

General Transit Feed Specification

C'est une spécification qui permet de présenter le service planifié par la société exploitante du transport en commun, normalisée par Google en 2005. Ce sont des données qui présentent plus précisément les horaires et les tracés d'un réseau de transport, accessibles à tout le monde et utilisés par les développeurs pour la création des applications de transport tel que : Google Maps, Transit ...

Le format GTFS s'agit d'un jeu de données composé d'une série de fichiers textes (au format CSV) rassemblés au sein d'un fichier ZIP.












Nom	Type	Taille
 agency	Document texte	1 Ko
 feed_info	Document texte	1 Ko
 calendar	Document texte	1 Ko
 fare_attributes	Document texte	1 Ko
 fare_rules	Document texte	3 Ko
 calendar_dates	Document texte	8 Ko
 routes	Document texte	6 Ko
 stops	Document texte	151 Ko
 trips	Document texte	655 Ko
 stop_times	Document texte	18 812 Ko
 shapes	Document texte	4 169 Ko

FIGURE 1 : DONNEES GTFS

A partir des données GTFS (Autobus | Secteur Laurentides pour la période 24-03-2022) disponibles sur le site officiel du groupe EXO, on a construit notre base de données. Dans le but de rendre les données exploitables pour notre tâche, nous avons commencé par transformer les données **GTFS** en format **GMNS** à l'aide d'un outil de conversion (**GTFS2GMNS**). À la suite de cette étape, on a obtenu 4 fichiers en format CSV (node.csv, link.csv, trip.csv, route.csv). Le fichier « trip csv » est

celui qui nous intéresse pour la suite puisqu'il représente les voyages effectués sur les différentes lignes de bus du secteur Laurentides (semaine, fin de semaine, jour férié, etc.). On a décidé de conserver seulement les voyages pendant les jours de semaine pour éviter tout biais de résultat à la fin de la segmentation. Pour chaque voyage et dans ce dernier fichier, on trouve les champs suivants «trip_id»; «route_id_short_name»; «directed_route_id»; «agency_name,travel_time»; «distance»; «node_sequence»; «time_sequence»; «geometry» .

A partir de ces champs, on a pu construire notre base de données finale, qui est caractérisée par les colonnes suivantes :

«Directed_route_id» : Une colonne constituée de deux éléments : route id pour définir le trajet et «direction_id» qui permet d'indiquer sa direction.

Ligne 9.0 : la ligne 9 dans la direction Express Saint Jérôme /Lafontaine

Ligne 9.1 : présente la ligne 9 dans la direction Expresse Métro Montmorency

«Heure du premier départ (JdS) » : l'heure de premier départ sur la ligne (pour un jour de semaine)

«Heure du dernier départ (JdS) » : l'heure de premier départ sur la ligne (pour un jour de semaine)

«NB Stations» : le nombre de stations de la ligne

«NB de trajets proposés» : le nombre de tracés de la ligne. Un tracé décrit le parcours par le véhicule sur la totalité d'un itinéraire, une ligne de bus peut avoir plus qu'un tracé.

«NB_LUN», «NB_MAR», «NB_MER», «NB_JEU», «NB_VED» : nombre de voyages par jour de semaine (LUN, MAR, MER, JEU, VED)

«NBdevoyages_wheelchair1»: le nombre de voyages pouvant accueillir au moins un usager en fauteuil roulant

«NBdevoyages_wheelchair0»: le nombre des départs avec un véhicule ne pouvant accueillir aucun usager en fauteuil roulant

«Nb_voy_00_04» : le nombre de départs (voyages) entre 00h et 04h du matin

«Nb_voy_04_08» : le nombre de départs (voyages) entre 04h et 08h du matin

«Nb_voy_08_12» : le nombre de départs (voyages) entre 08h du matin et 12h

«Nb_voy_16_20» : le nombre de départs (voyages) entre 16h et 20h du soir

«Nb_voy_20_24» : le nombre de départs (voyages) entre 20h du soir et minuit

«Distance (m) » : la distance totale parcourue par le bus calculé par la méthode LL2DIST (utilisant la formule Haversien sur une terre sphérique de rayon 6378.137km)

«Temps de parcours» : le temps qu'un bus doit prendre pour faire le parcours complet (en minutes)

«Temps inter-arrêt» : temps nécessaire pour rouler entre 2 arrêts + le temps d'embarquement et le temps de débarquement : présenté par différentes valeurs : sa valeur maximale «Max_temps_inter_arrêt», sa valeur minimale «MIN_temps_inter_arrêt», sa valeur moyenne «VM_temps_inter_arrêt» et sa valeur médiane «VMedian_temps_inter_arrêt »

A la fin de cette partie, nous avons obtenu notre base de données finale sur laquelle se basera le travail de préparation et de segmentation dans les parties suivantes.

3- Préparation des données

3.1. Données manquantes

Après avoir fait appel aux différentes bibliothèques nécessaire pour le travail d'exploration et préparation des données ainsi que leur visualisation, une partie destinée à la vérification des données manquantes a été élaborée et qui a permis de savoir qu'il n'y en n'a pas.

3.2. Construction des nouvelles colonnes

Afin de réduire la dimension horizontale de la base des données, d'augmenter la précision et de diminuer le temps du traitement des données par le modèle, nous avons construit en premier lieu une colonne « amplitude (Hour) » qui représente la différence entre l'heure du premier départ «heure du premier départ (JdS) et l'heure du dernier départ du bus «heure du dernier dzt(jdS) ». En deuxième lieu, nous avons construit une colonne « speed(m/min) » qui représente le rapport entre la distance «distance(m)» et le temps de parcours «temps de parcours_Jds(min) ». Avec cette transformation, nous avons pu réduire le nombre de colonnes de 24 à 22 colonnes. Bien qu'il y'a pas une forte réduction de dimension comme le cas lorsque on utilise d'autres techniques spécifiques tel que l'Analyse en Composantes Principales (ACP), cette réduction forme quand même un pas vers l'avant dans les résultats de segmentation dans la partie suivante.

3.3. Points aberrants

Pour la détection des points aberrants, nous avons choisi de visualiser le comportement de toutes les colonnes de la base de données en utilisant les boîtes à moustaches et on a constaté que les colonnes dans lesquelles apparaissent les points aberrants sont : « VM_temps_inter_arret » ; « VMedian_temps_inter_arret » ; « amplitude (Hour) » ; « speed(m/min) » et ceci comme le montre les boîtes à moustaches suivantes.

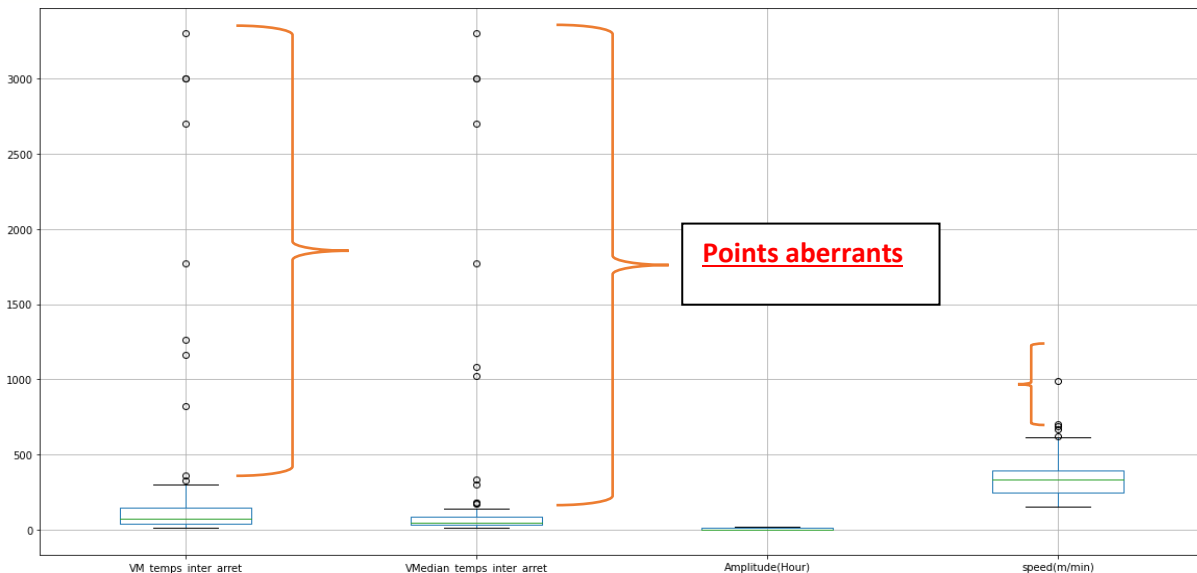


FIGURE 2: DISTRIBUTION DES VARIABLES AVANT LE NETTOYAGE

Comme il est montré dans la figure, les points aberrants sont ceux qui sont en dehors de la boîte.

Généralement, le traitement des points aberrants se fait par 4 méthodes : Garder, supprimer, corriger ou surveiller ces points. Pour notre cas, vu le nombre un peu faible des échantillons, la suppression des points aberrants ne permet pas de résoudre le problème. Nous avons pensé donc à garder les points aberrants pour les temps inter-arrêts et de corriger ceux de la vitesse.

Pour corriger les valeurs de la vitesse des points aberrants, une idée est d'élaborer un algorithme de régression et qui permet de prédire les valeurs de ces points en se basant sur le reste de la base des

données. Mais, pour notre cas, nous avons pensé à utiliser les algorithmes de classification vus en classe et ceci en classifiant les points aberrants de la colonne de vitesse pour déterminer quelle valeur attribuer à ces derniers. En effet, nous avons divisé, en premier temps, la base des données en deux sous bases : une contenant les 5 lignes des points aberrants (1) et l'autre est le reste de la base (2). En second temps, nous avons construit une colonne qui détermine la classe de chaque ligne en se basant sur la vitesse et qui prend 0 si la vitesse est inférieure au 75^{ème} quantile de vitesse et 1 sinon. Finalement, nous avons élaboré un algorithme de classification dont la base d'entraînement est la base (2) et les lignes qu'on veut prédire leurs classes sont dans la base (1). Finalement, cet algorithme de classification a permis de trouver que tous les points aberrants sont de classe 1. D'autres termes, Les valeurs des points aberrants sont tous supérieurs au 75^{ème} quantile. En conclusion, la réponse à la dernière question est d'attribuer la valeur maximale de vitesse aux points aberrants de cette colonne.

Après avoir fait ce traitement, on aura le comportement de données suivant :

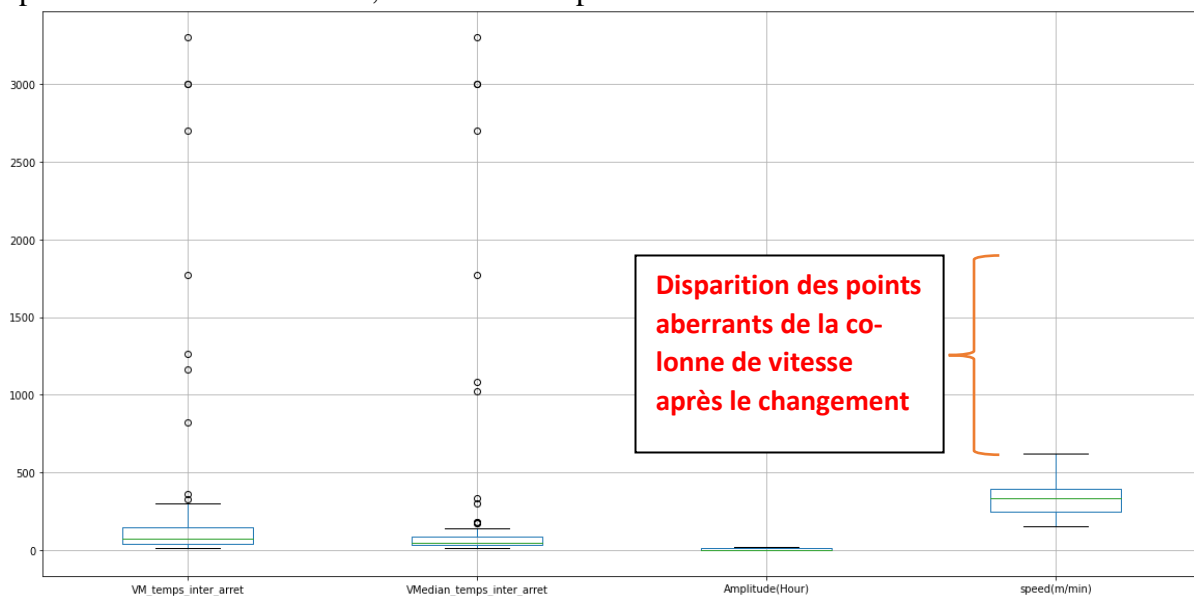


FIGURE 3 : DISTRIBUTION DES VARIABLES APRES LE NETTOYAGE

3.4. Autres techniques de nettoyage de la base des données

Nous avons supprimé les colonnes « NBdevoyages_wheelchair1 » et « NBdevoyages_wheelchair0 » vu que la première prend la valeur nulle dans tous les échantillons la deuxième est identique à l'une des colonnes des nombres de voyages par jour de semaine. L'algorithme K-Means calcule des distances donc nous avons jugé nécessaire d'effectuer une normalisation des données des colonnes : « MAX_temps_inter_arret », « MIN_temps_inter_arret », « VM_temps_inter_arret », « VMedian_temps_inter_arret », « Amplitude(Hour) », « speed(m/min) ». Nos données n'ayant pas de bruit, nous n'avons pas jugé nécessaire d'effectuer un lissage des données.

4- Segmentation

Dans la segmentation, il existe plus d'un algorithme approprié pour chaque ensemble de données et type de problème. Pour faire notre segmentation le choix devait se faire entre un modèle par partitionnement et un modèle hiérarchique, un modèle à densité était envisagé mais abandonné car ne donnant pas de bons résultats en général avec des données avec beaucoup de variables. Finalement notre choix s'est porté sur le modèle par partitionnement K- Means car celui-ci a de meilleures performances sur des bases de données avec beaucoup de variables, la visualisation des clusters aussi avec un modèle hiérarchique étant moins évidente avec un dendrogramme quand il y a beaucoup de données.

Les colonnes restantes de notre base de données ont été utilisées comme variables : « NB_Stations », « NB_trjets_proprés », « NB_LUN », « NB_MAR », « NB_MER », « NB_JEU », « NB_VEN », '

nb_voy_00_04», «nb_voy_04_08», «nb_voy_08_12», «nb_voy_12_16», «nb_voy_16_20», «VMedian_temps_inter_arret», «Amplitude(Hour)», «speed(m/min)». Nous avons porté notre choix sur la médiane pour représenter le temps inter-arrêt d'une ligne.

4.1. Choix du nombre de classes

Pour choisir le nombre de classes pour notre K-Means nous avons effectué une méthode basée sur l'inertie de la segmentation. L'inertie mesure à quel point un ensemble de données a été segmenté par les K-Means. Il est calculé en mesurant la distance entre chaque point de données et son centroïde, en mettant cette distance au carré et en additionnant ces carrés sur une classe. Pour trouver le K optimal pour un ensemble de données, nous avons utilisé la méthode du coude ; il s'agit d'effectuer l'algorithme pour k qui varie et trouver le point où la diminution de l'inertie commence à ralentir. Ainsi nous obtenons ce résultat :

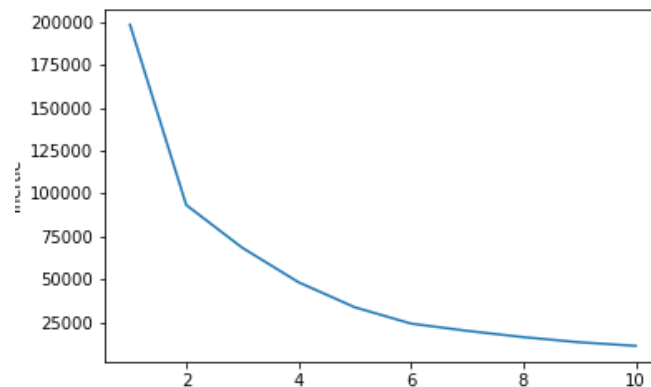


FIGURE 4 : L'INERTIE EN FONCTION DU NOMBRE DE CLASSES K

La partition en K = 6 classes est la dernière à induire un gain informationnel significatif, après l'inertie diminue de façon proportionnelle.

4.2. Résultats

Nous avons donc exécuté notre algorithme avec K = 6 et avons assigné à chacune de nos observations une classe. Ayant de nombreuses variables, il est difficile de visualiser clairement la segmentation. Pour pallier ce problème nous avons effectué une Analyse en Composantes Principales (ACP) pour réduire la dimension à 2. Nous pouvons ainsi voir cette segmentation avec le nombre d'éléments par cluster avec comme axes les deux composantes principales :

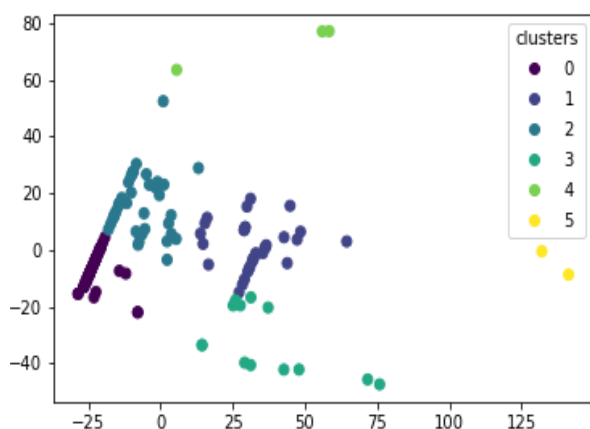


FIGURE 6 : VISUALISATION DE LA CLASSIFICATION APRES ACP

0	68
2	41
1	31
3	13
4	3
5	2

Name: cluster, dtype: int64

FIGURE 5 : REPARTITION DES CLASSES

Cette segmentation avec k = 6 est celle avec la meilleure inertie mais aussi la meilleure silhouette. Pour cette dernière nous avons trouvé 0.56 comme valeur.

Le diagramme en radar suivant nous permet de mieux comprendre nos classes au moyen de leurs caractéristiques. Il est adapté à cette tâche car un cercle peut gérer beaucoup de variables et les lignes à l'intérieur sont assez intuitives.

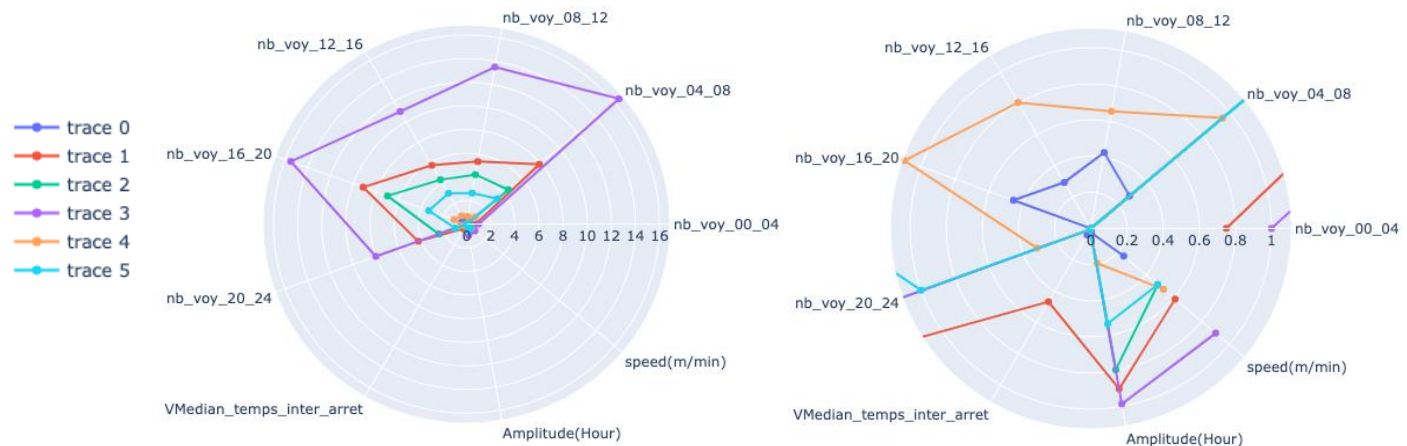


FIGURE 7 : DIAGRAMME EN RADAR DES CLASSES

Ce radar nous permet d'avoir un profil des différents groupes de bus obtenus avec notre segmentation. On remarque que ce qui différencie très souvent les classes c'est le nombre de départs par division du temps. Les classes 0 et 4 sont celles avec les plus faibles rayons pour les départs par division de temps tout comme pour les vitesses. Ce sont des lignes avec peu de départs par rapport aux autres et dont l'amplitude est moindre. Quant à la classe 3 on voit que c'est celle avec les bus très utilisés avec beaucoup de départs, une grande amplitude et avec les plus grandes vitesses moyennes. Une autre remarque concerne la nomenclature des lignes de bus, après segmentation on remarque que des lignes ayant une nomenclature similaire sont dans des mêmes classes. C'est par exemple le cas de la classe 2 où on retrouve la majorité des lignes avec le préfixe T.

5- Conclusion

La réalisation de ce travail nous a permis entre autres de pouvoir effectuer des traitements sur les données GTFS afin d'en faire sortir de l'information. Ainsi, depuis les données GTFS nous avons obtenu une base de données contenant les indicateurs que nous avons jugé pertinents. Les différentes colonnes ont été analysées pour faire ressortir les données aberrantes avant de garder ou corriger ces dernières. Grâce à la méthode des K-means nous avons pu déterminer le nombre de classes idéal avec la règle du coude et proposer une classification des bus d'EXO en 6 différentes familles de bus. Une ACP a été utilisée pour mieux visualiser les classes de la classification et dans le but de caractériser nos différents profils de bus, nous avons utilisé un diagramme en radar. L'analyse des observations aussi nous a permis de voir beaucoup de similarité dans la nomenclature des bus dans chaque cluster, surtout qu'on a quasiment à chaque fois les deux directions d'une ligne dans une même classe. Ainsi la segmentation des itinéraires de transit ouvre de nouvelles possibilités pour mieux comprendre comment les données GTFS peuvent aider à identifier les classes de lignes similaires et qui pourraient par conséquent avoir des stratégies opérationnelles et de planification similaires.

6- Références

Cours IND6212

<https://exo.quebec/fr/a-propos/donnees-ouvertes>