

# Analyse et visualisation des données du système de vélo en libre-service de Bruxelles

Maximilien Romain<sup>1</sup>

Promoteur : Gianluca Bontempi

Superviseur : Yann-Ael Le Borgne

<sup>1</sup>Université Libre de Bruxelles

maximilien.romain@ulb.ac.be

## Abstract

A notre époque, la mobilité dans les grandes villes est extrêmement élevée. Un très grand nombre de personnes utilise chaque jours différents moyens de transport que ce soit en transport privé ou en utilisant les transports publics. Il est donc possible de récupérer les informations concernant ces déplacements dans ces grandes villes et de les analyser afin de comprendre le comportement humain. Dans ce rapport nous allons nous concentrer sur l'étude des informations récupérables des vélos partagés de la ville de Bruxelles: les *Villos*. Nous allons donc implémenter un *dashboard* reprenant ces informations afin d'avoir une représentation complète et dynamique de ces données. Ensuite nous appliquerons un processus de clustering afin d'identifier les stations ayant une fréquence d'utilisation similaire, et comment cet fréquence d'utilisation est liée à la localisation de ces stations. Finalement nous essayerons la prédiction de futures disponibilités de *vilos* dans certaines stations de Bruxelles.

## Introduction

Bruxelles est la capitale de l'Europe, mais est aussi une des villes d'Europe avec le plus de mobilité. Le centre de Bruxelles en heure de pointe est quasiment impraticable et il n'est pas possible de se mouvoir librement en voiture pendant de très longues périodes. Un des sujets qui revient souvent sur la mobilité à Bruxelles est donc la recherche de solutions et l'optimisation de la fluidité du trafic automobile. Une des premières solutions est évidemment les transports en commun, mais ceux-ci apportent aussi quelques problèmes. L'installation d'un système de vélos en libre-service est donc une de ces récentes solutions apportées par les gouvernements, en permettant à certains utilisateurs de délaisser leur voiture au profit des vélos.

Dans cet article, nous allons nous intéresser aux différentes données que nous pouvons récupérer du système de vélos partagés de Bruxelles, les *vilos*<sup>1</sup>. Nous allons ensuite étudier les différentes utilisations des stations de *vilos* à travers des patterns de comportement spatio-temporels mettant en relation les vélos partagés et le comportement humain. Nous essayerons aussi de prédire la



Figure 1: (gauche) Une station de *vilos*; (droite) Carte des 340 stations

fréquence d'utilisation et le nombre de vélos disponibles dans les stations dans un future proche. Finalement nous implémenterons toutes ces informations dans un *dashboard* en ligne, qui permettra de récupérer et analyser toutes ces données de façon dynamique et en temps réel.

Cet article est complémentaire à des travaux déjà existants sur ce sujet dans d'autres villes, telles que Paris [Borgnat et al., 2011] ou Barcelone [Froehlich et al., 2009], et permet la comparaison des différents résultats de ces différentes villes. D'autres articles analysent aussi d'autres informations telles que la mobilité humaine, grâce aux traces des appels téléphoniques [Alessandretti et al., 2017]. D'autres travaux essayent quant à eux d'optimiser l'utilisation de ces systèmes de vélos en libre-service, et donc de réduire l'utilisation de voitures afin de réduire la pollution atmosphérique [Fernando et al., 2016].

## Les données de Villo

Les systèmes de vélos en libre-service sont des programmes encourageant l'intégration des données disponibles afin de rendre leur utilisation plus simple et plus accessible, et permettent donc de récupérer les données des vélos en temps réel. Ils intègrent donc une API reliée à leur base de données permettant la récupération et le stockage de ces informations. Il nous est alors possible d'en faire une grande base de données, afin d'obtenir un large historique d'utilisation de ces vélos partagés. De tels systèmes deviennent beaucoup plus courants aujourd'hui. On peut effectivement voir

<sup>1</sup><http://www.villo.be>

Number	Name	Address	latitude	longitude
280	280 - HEYSEL / HEISEL	***	50.897522	4.334831
281	281 - ATOMIUM	***	50.	4.

Table 1: Format de la table Statique de la base de données

StationID	Status	Bike_stands	Available_bike_stands	available_bikes	TimeStamp
280	1	25	13	12	1486854469000
281	1	25	10	15	1486854355000

Table 2: Format de la table dynamique de la base de données

ces systèmes de vélos en libre-service en France, en Espagne, mais aussi aux Etats-Unis, à New-York et à Washington D.C. Nous analyserons dans ce document les informations relatives aux *Villos*, les vélos partagés de Bruxelles. Il y a actuellement 340 stations éparses dans Bruxelles. En 2016, il y a eu plus de 1.5 million de locations de *Villos*, pour 40.000 abonnés longue durée et 70.000 tickets courte durée. Cela représente 7.5 millions de kilomètres parcourus par les utilisateurs. Evidemment cela a un impact important sur l'économie. Une telle utilisation des *Villos* a donc permis à Bruxelles de réduire son empreinte carbone de plus de 950 tonnes de CO<sub>2</sub>.

L'utilisation des *vilos* est très simple. Il suffit de se rendre à une des 340 stations dans Bruxelles (Fig.1), et de scanner sa carte de membre sur la borne de la station. La borne vérifie si les conditions de votre location sont respectées, c'est-à-dire d'avoir en solde sur son compte bancaire 150€ de caution en cas de perte ou non remise du *villo*. Une fois cette formalité accomplie, la borne vous indique quel vélo peut être retiré, et le *villo* est libre d'être utilisé et d'être rendu dans n'importe quelle autre station de Bruxelles. Il est alors possible de rouler pendant 30 minutes gratuitement, après ces 30 minutes le est de €0.50 pour la demi-heure suivante. Le prix passe à 1€ pour l'heure suivante et est de 2€ la demi-heure après deux heures d'utilisation.

Chaque fois qu'un *villo* est retiré ou déposé, l'API de JCDecaux<sup>2</sup>, l'entreprise s'occupant entre autres de la gestion des *vilos*, est mise à jour. Nous pouvons alors récupérer toutes les informations nécessaires depuis API afin de commencer notre analyse sur les déplacements de la population à partir de ces vélos. L'API fournit deux types d'informations récupérables à partir des stations de *vilos*.

## Les données statiques

Le premier type d'information que nous fournit JCDecaux sont les données statiques, c'est à dire que ce sont les données que nous allons devoir récupérer une seule fois au début de notre construction de base de données. Pour ce faire le site internet JCDecaux fournit un fichier au format *json*[Crockford, ] comportant diverses informations immuables des stations.

<sup>2</sup><https://developer.jcdecaux.com/home>

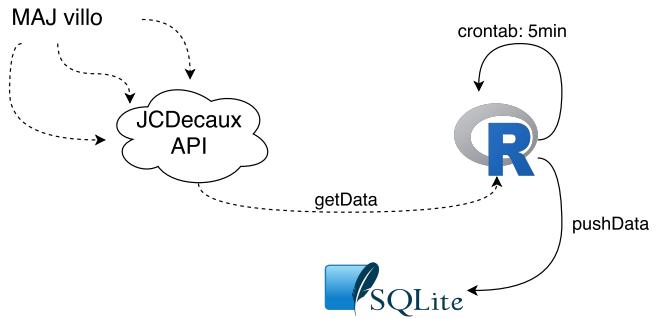


Figure 2: Diagramme de fonctionnement des scripts

Ces données (Table 1) représentent donc bien diverses informations qui ne changeront jamais, une fois qu'une station est installée. **Number** est le numéro d'identification de la station en question. **Name** et **address** sont des données servant à titre informatif pour l'utilisateur. Nous utiliserons ces données afin qu'un utilisateur puisse comprendre exactement où cette station se trouve à Bruxelles. Finalement, **latitude** et **longitude** servent à pouvoir localiser avec précision la station à partir d'outils informatiques qui permettent l'affichage de toutes les stations à Bruxelles.

## Les données dynamiques

Les données dynamiques représentent les données qui peuvent être récupérées directement depuis l'API de JCDecaux, cela représente donc les informations en temps réel des stations de Bruxelles. L'API est mise à jour à chaque fois que les informations d'une station ont été modifiées. Cela arrive lorsqu'un vélo a été déposé ou a été retiré d'une station. Nous avons donc écrit des scripts *R* [R Core Team, ] utilisant une clé d'identification fournie par JCDecaux afin de récupérer les informations de toutes les stations *vilos* et de les stocker dans une base de données. Afin de construire une base de données comportant assez d'informations pour notre analyse future, nous avons paramétré *crontab* afin de faire appel à ces scripts *R* toutes les 5 minutes, ce qui nous permet de récupérer 12 informations par heures sur toutes les stations. *Crontab* est le nom du programme sous Unix qui permet d'éditer les tables de configuration du programme *cron*, permettant l'automatisation de scripts sous Unix.

Chaque fois que nous faisons appel à ces scripts, nous stockons les informations récupérées dans notre base de données dans le format suivant: **StationID** représente le numéro d'identification d'une station et permet de lier ces informations avec la table statique de la base de données. **Status** est le status de la station. Si le status est à 1, c'est que la station est ouverte et opérationnelle, sinon le status est à 0. **Bike\_stands**, **available\_bike\_stands** et **available\_bikes** correspondent dans l'ordre au nombre total d'emplacements que possède la station, au nombre d'emplacements disponibles pour y déposer son *villo* et au nombre de *vilos* restant

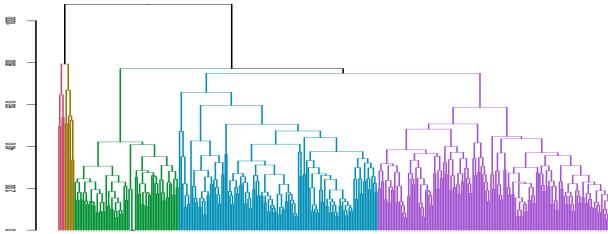


Figure 3: Dendrogramme de cinq clusters

disponibles pouvant être utilisés dans la stations. Une fois que tout a été récupéré, nous envoyons toutes ces informations dans notre base de données *SQLite*. (Fig.2)

Le choix de *SQLite* comme base de données est due à sa facilité d'utilisation et de sauvegarde. En effet, *SQLite* ne demande aucune configuration et ne nécessite pas de serveur pour fonctionner. La simple utilisation de sa librairie R permet de créer le fichier *.db* correspondant à la base de données. Le fait d'avoir la base de données dans un fichier physique permet de réaliser les backups et les exportations très facilement [Richard, 2017]. *SQLite* ne permet par contre pas la rapidité d'utilisation des données lorsque la base de données est trop grande. Si le projet est maintenu et que la base de données continue à être alimentée, il faudra exporter cette base de données dans un format capable de maîtriser un grand nombre de données, et d'être capable de charger ces données dans un programme en une fois. Nous ne chercherons pas dans ce rapport quel format serait le plus approprié.

Nous avons lancé notre crontab en novembre 2016, et est exécuté jusqu'à cette date. Nous avons donc récolté durant 179 jours, un total de 257.760 enregistrements pour chaque station, ce qui fait un total global de 87.638.400 enregistrements dans notre base de données. Chacun de ces enregistrements est une entrée dans la base de données, c'est-à-dire la situation d'une station à un instant précis. On a donc, pour chaque instant, le nombre de *vilos* d'une station.

### Comportement spatiotemporel

Après avoir laissé crontab exécuter les scripts R pendant un certain temps, et après avoir récupéré assez de données, il est possible de les analyser afin d'étudier le comportement spatio-temporel des cyclistes dans Bruxelles. Bruxelles a une superficie de  $161.4 km^2$ , il est donc évident que Bruxelles est composé de résidences, de zones commerciales et industrielles. Il est par conséquent intéressant d'étudier le déplacement des *vilos* dans la capitale afin d'y repérer les patterns de déplacement de la population bruxelloise.

Pour ce faire, nous allons utiliser une méthode de classification automatique des données, appelée *clustering hiérarchique* [Johnson, 1967]. Le *clustering hiérarchique* est une méthode de regroupement par algorithme de classification. C'est donc un algorithme capable de classer des

données en fonction de divers paramètres. Cette technique va nous permettre de construire un diagramme spécifique aux clusterings, le *dendrogramme*, qui permet d'illustrer l'arrangement de groupes générés par un regroupement hiérarchique. Ces méthodes seront appliquées sur un certain nombre de données de l'ensemble des informations récoltées sur les stations, en particulier sur le nombre de vélos disponibles dans les stations.

### Dendrogramme

Pour construire un dendrogramme il est nécessaire de calculer les distances entre chaque station afin de construire une *fermeture transitive*. La *fermeture transitive* est une matrice des distances entre toutes les stations de Bruxelles, cela veut dire qu'une matrice taille  $m * m$ , où  $m$  est le nombre de stations dans Bruxelles, représente à l'indice  $(i, j)$  la distance entre la station  $i$  et la station  $j$ . La distance entre deux stations est le niveau de similarité entre ces deux stations. Au plus deux stations sont proches, au plus elles partageront un niveau de similiarité élevé, et donc une fréquence d'utilisation similaire. Il est donc nécessaire de choisir la représentation mathématique de cette distance. Nous choisissons dans ce rapport la méthode utilisant la distance euclidienne. Cette méthode permet de calculer la distance entre deux points ou entre deux vecteurs. Nous considérerons ici que chaque station est représentée par un vecteur  $v_i$  distinct contenant  $n$  valeurs  $v_{i,t}$ , où  $n$  représente le nombre d'enregistrements pour la station  $i$ . Cette variable  $v_{i,t}$  correspond au nombre de vélos dans le vecteur  $i$  à l'instant  $t$ . La distance euclidienne est donc calculée entre chaque paire de stations afin de construire notre *matrice de distance*. Cette distance est définie comme

$$d_{eucl}(\vec{v}_i, \vec{v}_j) = \|\vec{v}_i - \vec{v}_j\| = \sqrt{\sum_{t=1}^n (v_{i,t} - v_{j,t})^2} \quad (1)$$

Une fois cette matrice construite, nous pouvons procéder au calcul du *clustering hiérarchique*. La méthode va procéder en examinant chaque distance de la *fermeture transitive* et rassembler dans un nouveau groupe les deux stations ayant la plus petite valeur dans cette matrice. Lorsqu'il n'y a plus de couple de stations assez proche entre-elles, la méthode va chercher à regrouper les stations restantes avec les groupes déjà existants, ou à rassembler des groupes entre eux. Pour examiner la distance entre deux groupes, il faut choisir une mesure de *dissimilitude inter-classe*. Cette mesure peut être le *lien moyen* qui consiste à comparer la moyenne des différents groupes, la *médiane*, consiste à comparer ces différents groupes par rapport à leurs médianes, etc. (Algo.1) [Jain et al., 2000]

Une fois tous les groupes rassemblés et le clustering terminé, il est possible d'afficher le résultat sur notre dendrogramme. (Fig.3)

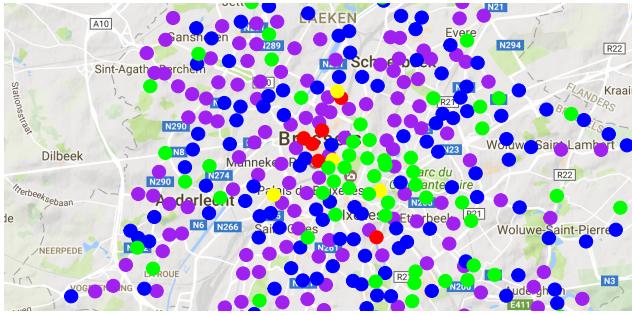


Figure 4: Représentation géographique des clusters

---

**Algorithm 1** Algorithme de clustering (prend la matrice de distance  $M$  et les clusters en paramètre)

---

former  $m$  cluster d'un seul élément à partir des  $m$  stations;  
 $nbvCluster = m$ ;  
**while**  $nbrCluster > 1$  **do**  
    trouve les deux clusters  $c_1$  et  $c_2$  les plus proches;  
    fusionne  $c_1$  et  $c_2$  en nouveau cluster  $C$ ;  
    calcule la distance de  $C$  par rapport aux autres clusters;  
    **if** ils sont proches **then**  
        supprime lignes et colonnes de  $M$  correspondant à  
         $c_1$  et  $c_2$ ;  
        ajoute lignes et colonnes correspondant à  $C$  dans  $M$ ;  
    **end**  
     $nbrCluster --$ ;  
**end**

---

## Représentation des clusters

Une fois le dendrogramme généré, il est possible d'utiliser les valeurs retournées par son calcul afin de représenter toutes les stations groupées en clusters sur une carte de Bruxelles. De cette manière il sera possible d'apercevoir les différentes stations dans leur groupe, ainsi que leur position dans Bruxelles. Il sera donc possible d'analyser si des zones de Bruxelles auront plus de similarités par rapport à d'autres zones ou quartiers.

Pour ce faire, il est nécessaire de récupérer pour chaque cluster, les stations associées. Une fois ce travail terminé, il faut récupérer les coordonnées de chacune de ces stations afin de pouvoir les représenter sur la carte de Bruxelles. Lorsque ceci est terminé, la carte de Bruxelles contenant les différents regroupements nous est retournée.

Pour ce clustering, nous avons utilisé les informations de notre base de données durant la semaine du 12/02/2017 au 19/02/2017. Ce choix est du à la limitation des performances de SQLite cité plus haut. De ce fait, nous ne pouvons analyser de plus grandes périodes sans un coût significatif sur le temps de traitement de ces données. Il est cependant possible d'analyser d'autres semaines de données différentes de celle que nous avons analysées. Pour cela il suffit de décaler le premier jour de la semaine d'analyse afin de considérer

une semaine différente.

## Prédiction de l'utilisation des villos

Nous allons maintenant nous concentrer sur la prédiction de l'utilisation des stations de *Vilos* à travers Bruxelles. Plus précisément, la prédiction va se concentrer sur le nombre de *vilos* disponibles dans ces différentes stations.

De telles prédictions permettraient à l'entreprise *Villo!* de maintenir une bonne balance dans le nombre de *vilos* disponibles à travers Bruxelles. L'entreprise aura ainsi une meilleure vue d'ensemble du nombre de vélos disponible à un certains instant, et pourra donc remédier au problème du manque de vélos dans les stations les plus utilisées. De plus, cela permettra aux divers habitués du service de vélos partagés d'être au courant du nombre de vélos actuels dans une station et des futurs vélos disponibles dans cette même station, et d'organiser leurs différents déplacements dans la ville.

## Modèles prédictifs

Nous avons implémenté ici différents modèles prédictifs afin de comparer les différents résultats retournés par chacun des modèles et de comparer leurs atouts et leurs faiblesses par rapport à la précision des modèles de prédiction. Il existe un très grand nombre de modèles prédictifs dans beaucoup de domaines afin de prédire des données significativement différentes [Brusic et al., 2004][Klock and Krasse, 1979]. Nous nous concentrerons sur un petit nombre de modèles prédictifs assez simples dans ce rapport. Cette limitation est due au manque de temps et aux limitations techniques de ce travail. Chaque modèle prend en compte plusieurs paramètres: l'ensemble des valeurs  $v_{i,t}$  du vecteur  $\vec{v}_i$  et une variable  $h$  d'horizon qui indiquera quel instant nous sommes en train de prédire. Nous avons ici choisi un  $h$  qui variera entre 0 et 24 heures. [Rob, 2014]

**Méthode naïve.** Ce modèle prédit que le nombre actuel de vélos restera constant par rapport à la dernière valeur récupérée  $v_n$ .

$$\hat{v}_{i,n+h} = v_{i,n} \quad (2)$$

**Méthode historique.** Ce modèle prédit que le nombre de vélos sur la journée suivante sera identique aux données de la dernière journée récupérée.

$$\hat{v}_{i,n+h} = v_{i,n-24+h} \quad (3)$$

**Méthode de drift.** Ce modèle se base sur la tendance d'évolution des dernières valeurs afin de prédire les suivantes.

$$\hat{v}_{i,n+h} = v_{i,n} + \frac{h}{n-1} \sum_{t=2}^n (v_{i,t} - v_{i,t-1}) \quad (4)$$

$$= v_{i,n} + \frac{h}{n-1} (v_{i,n} - v_{i,1}) \quad (5)$$

## Discussion des résultats

Nous allons maintenant discuter des différentes prédictions des trois modèles prédictifs et de leurs marges d'erreur. Pour ce faire nous appliquons ces différents modèles sur une période de données que nous avons déjà récupérée, de ce fait nous pouvons prédire une journée supplémentaire et vérifier la marge d'erreur de cette prédiction avec les vraies données.

Afin de calculer la précision des prédictions et donc la marge d'erreur de ces divers modèles prédictifs, nous introduisons le concept mathématique *d'erreur quadratique moyenne normalisé (NMSE)*. Cette variable est définie comme

$$NMSE = \frac{MSE}{var(v)} \quad (6)$$

où  $var$  et  $MSE$  sont définis comme

$$var = \frac{1}{n} \sum_{l=1}^n (v_{i,l} - \bar{v}_i)^2 \quad (7)$$

$$MSE = \frac{1}{n} \sum_{l=1}^n (v_{i,l} - \hat{v}_{i,l})^2 \quad (8)$$

où  $\bar{v}_i$  est la moyenne des valeurs du vecteur  $v_i$ , et  $\hat{v}_{i,l}$  est la valeur prédite du vecteur  $v_i$  à l'instant  $l$ . Cette  $NMSE$  se rapproche de 0 lorsque le modèle prédictif a peu d'erreurs, et donc que les prédictions se rapprochent des données réelles. Lorsqu'elle s'en éloigne, cela informe d'un taux d'erreurs élevé lors de la prédiction des valeurs.

La méthode naïve est le modèle le plus simple des trois. Etant donné qu'on prédit que la dernière données de *villo* restera constante, s'il y a une simple évolution des données, comme un vélo supplémentaire par exemple, alors la prédiction est erronée. Il en résulte donc qu'au plus une station verra son nombre de *villos* fortement diminué ou augmenté, au plus la prédiction naïve sera fausse. Une station qui ne se voit pas utilisée durant un large moment, suivra la prédiction naïve, jusqu'au moment où un *villo* est déposé ou retiré. Si nous calculons l'erreur moyenne des données prédites par ce modèle et les valeurs réelles pour la même semaine de données que le clustering cité ci-dessus et pour la station de Bourse, qui est une des stations les plus utilisées de Bruxelles, nous obtenons une valeur de 0.808, ce qui correspond à une très mauvaise prédiction.

La méthode historique est souvent erronée mais permet d'avoir une vue générale de l'évolution des données d'une station. En effet, on peut apercevoir dans les stations se trouvant au centre ville, telle que Bourse, que l'évolution des données d'un jour correspond, avec des amplitudes différentes, aux autres journées de ces stations. Prédire la journée suivante en récupérant les données de la journée passée, ne permet donc pas d'avoir avec certitude une bonne information sur le nombre de vélos dans une station, mais permet de voir à quel moment une station commencera à



Figure 5: Profile de prédiction de la méthode historique

gagner ou à perdre son nombre de *vilos*. L'erreur moyenne ici vaut 0.112, ce qui est une bien meilleure prédiction que le modèle précédent, et nous permet d'affirmer que la méthode historique suit bien l'évolution des données réelles. (Fig.5)

La méthode de drift se base sur les dernières données pour prédire nouvelles données. Il faut donc, pour ce modèle prédictif, utiliser une variable  $m$  qui indiquera le nombre total de données que nous utiliserons pour notre prédiction.

Au plus  $m$  est grand, au plus grand est le nombre de données du passé, et en résulte donc une plus grande chance de fausser la prédiction. En effet lorsque nous calculerons la moyenne d'évolution de ces données et qu'un trop grand nombre de données ont été choisies, cette moyenne d'évolution couvrira une large période de temps et aura de grandes chances d'être erronée. Prenons l'exemple d'une station avec une forte augmentation de son nombre de vélos, suivie directement d'une diminution de ces vélos, la prédiction calculera la moyenne de ces deux variations et il en résultera des données erronées. A l'opposé, le modèle prédictif ne doit pas prendre un  $m$  trop petit, afin de ne pas baser sa prédiction sur trop peu de données.

Il faut donc trouver un  $m$  qui permettra de rapprocher le plus possible la prédiction aux données réelles. L'erreur moyenne ici est de 0.2, pour un choix  $m$  de 10 valeurs précédentes. Diminuer ou augmenter  $m$  augmente notre erreur moyenne. Nous avons donc un  $m$  assez précis pour nous rapprocher le plus possible des valeurs réelles futures.

## Implémentation d'un dashboard

Maintenant que nous avons introduit et expliqué tous nos concepts mathématiques, nous avons la possibilité d'implémenter un outil capable d'exécuter tous ces concepts dynamiquement et en temps réel. Cet outil est directement relié à la base de données *SQLite* qui continue à être alimentée. Notre outil utilise donc les données les plus récentes récupérées depuis l'API JCDecaux, et ces données correspondent aux stations de *vilos* à l'instant où nous utilisons l'outil. L'outil est donc un *dashboard* utilisable dans un navigateur internet et utilise la librairie *RShiny* de R, qui est un projet de *RStudio* qui permet l'implémentation d'applications web sans utilisation de langages web tels que

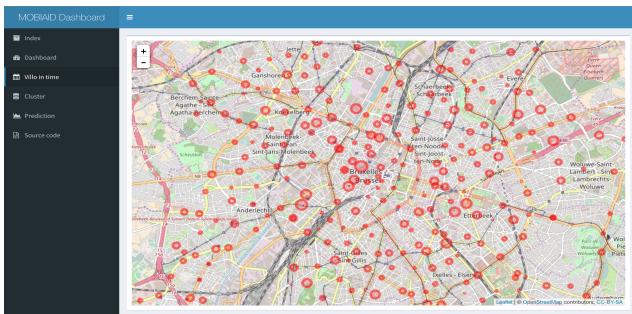


Figure 6: Un des onglets du dashboard

*HTML, CSS ou encore javaScript [Chang et al., 2016].<sup>3</sup>*

L’application est divisée en différents onglets, où chaque onglet correspond à un outil de visualisation différent. Il y a au total quatre onglets que nous allons expliquer ici.

### Représentation des stations

Le premier onglet est la représentation de toutes les stations de *villo* dans Bruxelles. Les 340 stations sont affichées sur une carte interactive, ce qui permet à l’utilisateur de voir précisément où se trouvent les stations et la possibilité de cliquer sur une des stations afin de faire apparaître une *popup* et ainsi apercevoir son nom et son numéro d’identification.

Une fois le nom ou l’idendificateur d’une station récupéré, l’utilisateur peut rechercher une station en particulier dans le menu déroulant se trouvant en dessous de la carte. Ce menu déroulant permet de sélectionner une station à analyser dans le graphique juste en dessous. Une fois ceci terminé, toutes les données de la dernière semaine écoulée sont affichées sur un graphique, et permet à l’utilisateur de voir l’évolution du nombre de vélos dans la station sélectionnée. De même, ce graphique est interactif et permet à l’utilisateur de zoomer dans le graphique et de percevoir plus facilement des intervalles de temps plus petits.

### Villo dans le temps

Ce deuxième onglet est aussi une carte interactive qui permet cette fois à l’utilisateur d’observer l’évolution de nombre de *villos* dans le temps et dans l’espace. En effet un curseur donne la possibilité de choisir un instant particulier dans l’ensemble des données, et d’afficher sur la carte le nombre de vélos, représenté par le diamètre des cercles, à l’instant choisi. La carte affiche donc le nombre de *villos* par station pour toutes les stations de Bruxelles. Il est alors possible d’observer, en fonction du temps, quelles stations possèdent le plus de vélos, et quelles stations en sont presque dépourvues.

<sup>3</sup><http://94.23.200.127:150/mobi/>

### Représentation du clustering

Le troisième onglet est l’affichage des clusters présentés plus haut. Cet onglet laisse donc la possibilité de sélectionner la méthode de représentation dans la matrice des distances, la méthode d’agglomération du clustering ainsi que le nombre de clusters souhaités à afficher sur le carte. Lorsque toutes les options sont sélectionnées, l’exécution de l’algorithme est lancée lors de l’appui du bouton *run*. Une fois l’exécution terminée, les clusters sont affichés sur la carte de Bruxelles ainsi que sur le dendrogramme utilisé lors de l’algorithme. Cet outil-ci permet à un utilisateur d’observer en temps réel quelles sont les stations qui ont des fréquences d’utilisation similaires depuis une semaine, et permet donc d’analyser les paterns d’utilisation de ces stations à l’instant précis.

### Prédiction

Le dernier onglet est l’outil qui permet la visualisation des modèles prédictifs. Dans cet onglet l’utilisateur a le choix d’analyser une station en particulier avec le même système de sélection d’une station à partir d’un menu déroulant présenté pour l’outil de représentation des stations à Bruxelles. Ici un choix supplémentaire est proposé, il s’agit du choix de modèle prédictif à observer. Les trois modèles sont présents, la méthode naïve, la méthode historique, et la méthode de drift. Une fois la méthode sélectionnée, les dernières valeurs de la stations sont affichées et les valeurs prédites sont ajoutées .

### Conclusion

A la suite de ces différentes analyses, nous avons pu observer divers résultats sur l’utilisation des villos dans Bruxelles. Nous avions tout d’abord analysé les données *villos* sur un graphique en fonction du temps. Cet analyse nous a permis d’apercevoir l’utilisation globale des stations et leur répartition sur Bruxelles. Ensuite nous avons essayé de grouper les stations utilisées de manière similaire. Pour se faire nous avons fait appel à une méthode de regroupement hiérarchique et nous avons affiché cinq groupes sur une carte de Bruxelles. Ceci nous à permis de vérifier si certaines zones de Bruxelles sont utilisées de manière similaires entre-elles. Enfin, nous avons tenté la prédiction du future nombre de villos dans les stations villos. Pour cela nous avons implementé trois modèles prédictifs différents, possédant chacun leurs avantages et désavantages.

Le but du projet est de permettre une analyse et de donner un oeil nouveau sur un moyen de transport grandissant, permettant ainsi l’allégement du trafic automobile dans Bruxelles. Ceci permettrait d’encourager un plus grand nombre de personnes à utiliser ces services.

Dans le future, ce travail pourrait être récupéré et amélioré afin de proposer des résultats plus précis et concret. Pour ce faire, une version future pourra améliorer les divers concepts

mathématiques utilisés dans cette recherche. L'annexe explique pourquoi ce rapport ne va pas plus loin, et comment il pourrait être amélioré.

## Remerciements

Je remercie Gianluca Bontempi, mon promoteur, de m'avoir proposé ce sujet, qui est pour moi intéressant, complet, et concret, trois critères que je considère importants pour mener à bien un projet. Je remercie aussi bien évidemment mon superviseur Yann-aël Le Borgne, qui m'a accompagné, guidé et aidé lors de l'écriture de cet article et de l'implémentation de l'application Web.

## Annexe

Ce rapport est un travail de fin de cycle de 3ème bachelier, et à été travaillé en parallèle avec d'autres travaux pour d'autres cours. Pour éviter d'introduire des complexités techniques trop importantes en regard de ce travail, il a été choisi de ne pas faire de normalisation de données trop poussée et de trop complexifier l'analyse de ces données. De même, les techniques utilisées pour le clustering hiérarchique et les modèles prédictifs ne sont pas complexes. Ceci doit évidemment être fait dans le cadre d'un travail nécessitant des résultats plus précis dans le cas d'applications plus concrètes. Cela est dû au manque d'expérience et de connaissance d'un 3e année bachelier. Il est donc important pour le future de ce rapport que les techniques implémentées puissent toutes être récupérées et améliorées. Pour cela tout le code du dashboard est disponible sur un dépôt github<sup>4</sup>, laissant ainsi la possibilité de cloner ce répertoire et de l'améliorer.

## References

- [Alessandretti et al., 2017] Alessandretti, L., Sapiezynski, P., Lehmann, S., and Baronchelli, A. (2017). Multi-scale spatio-temporal analysis of human mobility. *PLoS One*, 12(2):e0171686.
- [Andreas et al., 2010] Andreas, K., Rodrigo, M., Jens, G., Joan, C., and Rafael, B. (2010). Urban cycles and mobility patterns: Exploring and predicting trends in a bicycle-based public transport system. *Elsevier*.
- [Borgnat et al., 2011] Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.-b., and Fleury, E. (2011). Shared bicycles in a city: a signal processing and data analysis perspective. *Advances in Complex Systems*, 14(03):415–438.
- [Brusic et al., 2004] Brusic, V., Bajic, V. B., and Petrovsky, N. (2004). Computational methods for prediction of t-cell epitopes—a framework for modelling, testing, and applications. *Methods*, 34(4):436–443.
- [Chang et al., 2016] Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2016). *shiny: Web Application Framework for R*. R package version 0.14.2.
- [Crockford, ] Crockford, D. *JavaScript Object Notation (JSON)*.
- [Edzer, 2012] Edzer, P. (2012). spacetime: Spatio-temporal data in r. *Journal of Statistical Software*.
- [Fernando et al., 2016] Fernando, M.-P., Cèsar, F., , and Lidia, C.-O. (2016). Cycling network projects: a decision-making aid approach.
- [Froehlich et al., 2009] Froehlich, J., Neumann, J., and Oliver, N. (2009). Sensing and predicting the pulse of the city through shared bicycling. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI'09*, pages 1420–1426, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Henri, 2016] Henri, L. (2016). *Data Scientist et langage R - Guide d'autoformation à l'exploitation des Big Data*. Editions ENI.
- [Jain et al., 2000] Jain, A., Murty, M., and Flynn, P. (2000). Data clustering: A review.
- [Johnson, 1967] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.
- [Jon et al., ] Jon, F., Joachim, N., and Nuria, O. Sensing and predicting the pulse of the city through sahred bicycling. *University of Washington*.
- [Klock and Krasse, 1979] Klock, B. and Krasse, B. (1979). A comparison between different methods for prediction of caries activity. *European Journal of Oral Sciences*, 87(2):129–139.
- [Pierre et al., 2011] Pierre, B., Celine, R., Jean-Baptiste, R., Patrice, A., Patrick, F., and Eric, F. (2011). Shared bicycles in a city: A signal processing and data analysis perspective.
- [R Core Team, ] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- [Richard, 2017] Richard, H. (2017). *SQLite*.
- [Rob, 2014] Rob, H. (2014). Forecasting: Principles practice. <http://robjhyndman.com/uwafiles/fpp-notes.pdf>.
- [Samiul et al., 2012] Samiul, H., Christian, S., Satish, U., and Marta, G. (2012). Spatiotemporal patterns of urban human mobility. *Springer Science+Business Media New York*.

<sup>4</sup><https://github.com/Myxfall/MOBI-AID>