Classification non supervisée
Clustering et K-Means

Compte-rendu de Bureau d'Étude

Damien Douteaux et Kevin Hua

Mars
9
2017

Table des matières

Liste des figures

1 • Lecture des données

On commence par lire le fichier de données en utilisant le code suivant :

```
## Première question
#  Lecture des données, on ne retient que les 52 dernières colonnes
data  = read.table("DonneesLyon.dat", header=TRUE)

data2 =  data[,4:55]
```

On obtient alors deux dataframes, l'un deux servira à l'entraînement

2 • Classification hiérarchique

2.1. Application de la classification ascendante

Comme indiqué dans l'énoncé, nous utilisons dans le code qui suit la fonction *hclust* :

```
## Classification hiérarchique
#  Application de la classification ascendante
distances = dist(data2)
h = hclust(distances, method="ward.D2")
```

Le modèle en tant que telle n'est pas très parlant initiallement. Ainsi, nous allons afficher le dendogramme de ce modèle via la commande *plot*. Le code utilisé est simplement :

```
# Affichage du dendogramme
plot(h) # En l'affichant, on propose 4 classes
```

On obtient alors le résultat de la Figure 1.

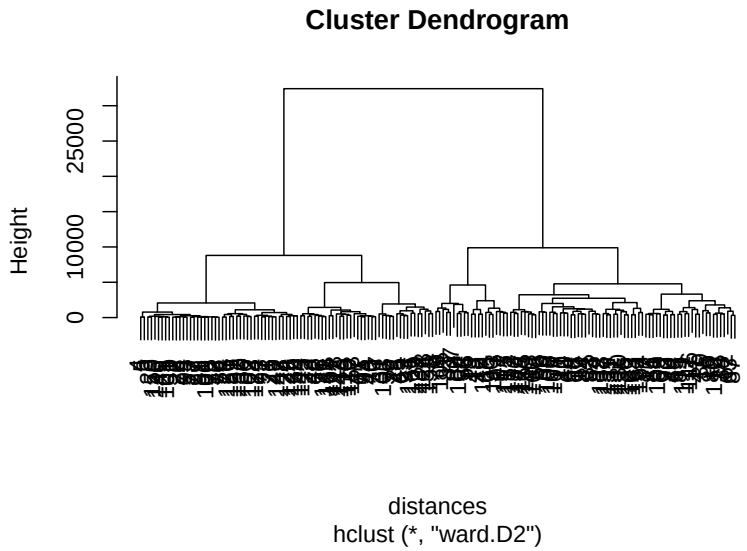


FIGURE 1 • Dendogramme obtenu lors de l'entraînement du modèle

Sur cette figure, on remarque que les « deux premiers niveaux » semblent être ceux où les distances sont les plus grandes. Nous proposons ainsi de les laisse séparer, ce qui correspond à créer un modèle avec quatre clusters. Nous prendrons cette valeur par la suite.

2.2. Réduction du dendogramme

Nous avons ainsi choisi de conserver quatre clusters pour notre modèle. Nous utilisons alors le code suivant pour réaliser cette opération sur le dendogramme :

```
# On choisit quatre classes et on coupe l'arbre
nb_classes = 4
new_model = cutree(h, nb_classes)
plot(data$Heure, data$Jour, col=c("green", "blue", "orange", "red")[new_model], pch=16, cex
    =1.2)
```

De plus, nous pouvons observer que ce code permet de tracer également les partitions. Le choix de représentation est le suivant :

- *Les lignes* Chaque ligne correspond à un jour de la semaine, le « 1 » en bas représentant le dimanche et le « 7 » en haut le lundi.

- *Les colonnes* Chaque colonne correspond à une heure de la journée, les heures étant marquées dans l'ordre de gauche à droite.

La figure obtenue est alors celle de la Figure 2.

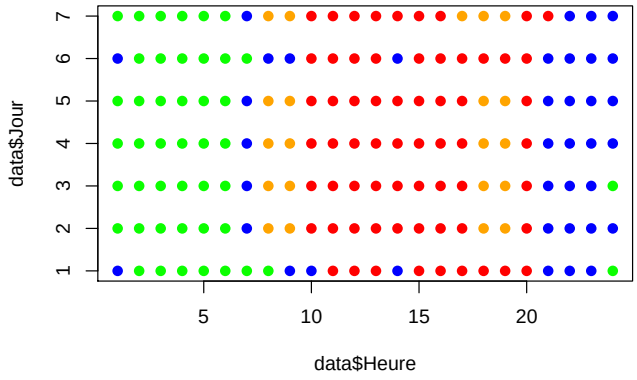


FIGURE 2 • Visuel obtenu pour les clusters avec un choix de quatre clusters

Quelques remarques sur la Figure 2 :

- Le lundi est une journée très diversifiée, où les quatre clusters sont représentés. En particulier, on observe de manière logique le cluster « matin tôt » avant 6h, puis une montée en capacité des routes. On observe également des périodes creuses dans le journée, comme entre 16h et 19h.
- On obsere une assez grande homogénéité dans les répartition des classes entre les jours de la semaine du mercredi au samedi. Le déroulé de ces jours semble « classique » :
  - *Avant 6h* Présence du cluster « matin tôt » qui doit correspondre à un faible volume de circulation.
  - *7h et après 21h* Présence d'un cluster intermédiaire de faible affluence.
  - *8h-9h et 18h-19h* Présence d'un cluster de circulation moyenne, qui doit représenter un volume qui n'est pas celui du « bouchon » mais tout de même soutenu.
  - *Reste de la journée* Le cluster représentant une circulation à son maximum.

  On remarque sans surprise que les heures limites correspondent à la partie de travail des bureaux et des artisans en centre ville.
- Le mardi a la particularité de présenter des périodes de transition plus abruptes (absence du cluster orange).
- Pour le dimanche, on remarque une journée plus calme, avec en particulier le cluster vert durant jusqu'à 8h, et lez zones rouge étant moins étendues. De même, les transitions sont assez abruptes avec absence du cluster orange. Il est également amusant de remarquer le point de minuit le matin (le premier de la ligne), qui semble indiquer qu'il y a pas mal de lyonnais sur les routes le samedi soir tard dans la nuit, fête oblige...

Ainsi, nous pouvons dire en résumé que la classe avec le flux moyen de véhicule le plus fort est la classe rouge (ie. celle correspondant au cœur de la journée) et celle du plus faible flux moyen la verte (le matin très tôt).

Pour terminer, il est possible de représenter nos clusters sur le dendogramme en utilisant le code suivant :

```
plot(h)
rect.hclust(h, k=nb_classes, border=c("green", "blue", "orange", "red"))
```

On obtient alors la Figure 3

Comme attendu, nous avons bien les clusters dont nous avions parlé, avec une séparation qui se fait au « niveau deux ».

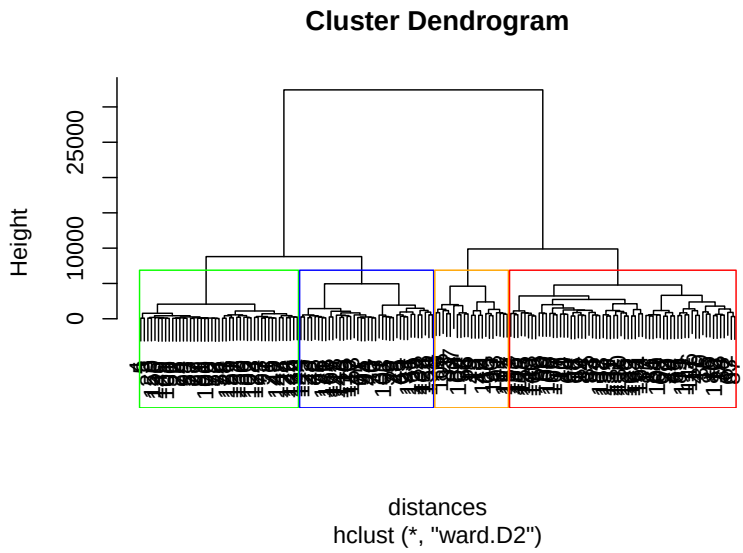


FIGURE 3 • Représentation des clusters sur le dendogramme

2.3. Regard sur la concentration des classes

Pour cela, nous utilisons la fonction *inertie* proposée avec le sujet. Cette fonction prenant en paramètre des listes, il nous faut au préalable transformer les données. On peut ensuite tracer les résultats. Ceci est réalisé avec le code suivant :

```
liste_mat=list(data2[new_model==1,],data2[new_model==2,],data2[new_model==3,],data2[new_model
    ==4,])
inertie=inertie(liste_mat,nb_classes)
plot(inertie$Intra,col=c("green", "blue", "orange", "red"), pch=16, cex=2)
```

On obtient après coup le graphe avec l'information des inerties de la Figure 4.

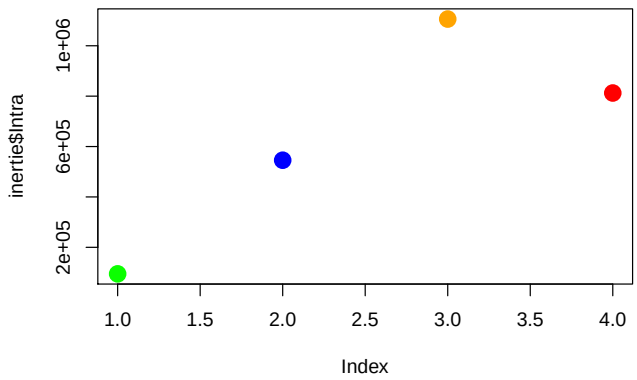


FIGURE 4 • Inertie des différents clusters observés

Ainsi, nous observons que le cluster le plus dispersé est le orange, correspondant (si on en croit sa répartition dans la semaine) à une circulation transitoire entre faible affluence et bouchon lyonnais. Au contraire, le cluster le moins dispersé est le vert, qui correspond aux situations où il n'y a personne dans les rues (ie. avant 6h du matin). Ces résultats peuvent sembler cohérent, dans la mesure où définir une situation de circulation intermédiaire peut s'avérer difficile, alors que définir l'absence certaine de circulation sur une grande période assez intuitif. Cela nous apprend, qu'il y a également plus de variance dans la notion de bouchon que la notion d'absence de voiture.

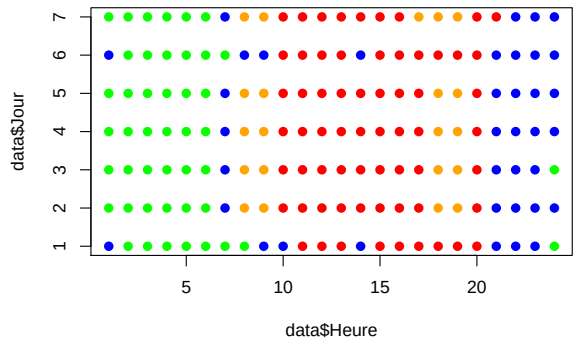
2.4. Comparaison des critères d'agrégation

Nous réutilisons tous les codes précédents, mais en partant cette fois-ci des critères *complete*, *single*, *centroid*, ie. le maximum de distance entre les éléments de deux clusters. Nous obtenons alors un code semblable à ce qui précède :

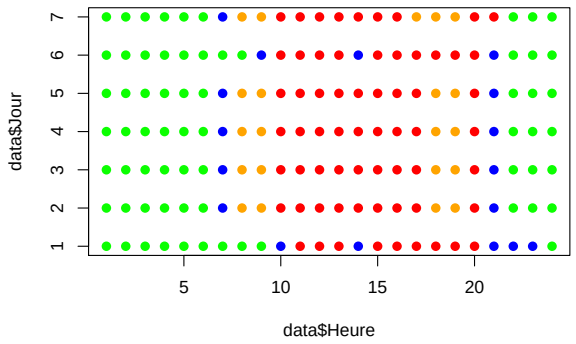
```
h2=hclust(distances,method="___nom_methode___")
plot(h2)
n=4
cluster=cutree(h2,n)
rect.hclust(h2,k=n,border=c("green","blue","orange","red"))
plot(data$Heure,data$Jour,col=c("green","blue","orange","red")[cluster],pch=16,cex=1.2)

liste_mat=list(data2[cluster==1,],data2[cluster==2,],data2[cluster==3,],data2[cluster==4,])
inertie=inertie(liste_mat,nb_classes)
plot(inertie$Intra,col=c("green", "blue", "orange", "red"), pch=16, cex=2)
```

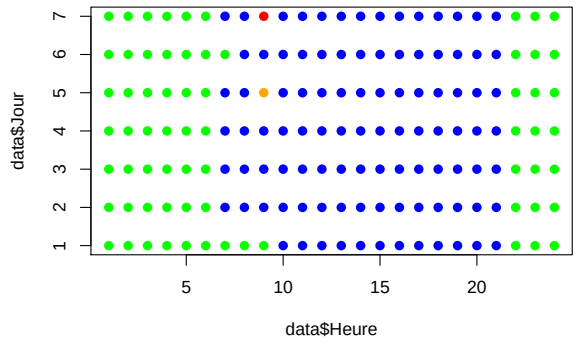
Nous pouvons alors comparer pour chacun de ces éléments la répartition des clusters dans le semaine, ce qui est proposé à la Figure 5.



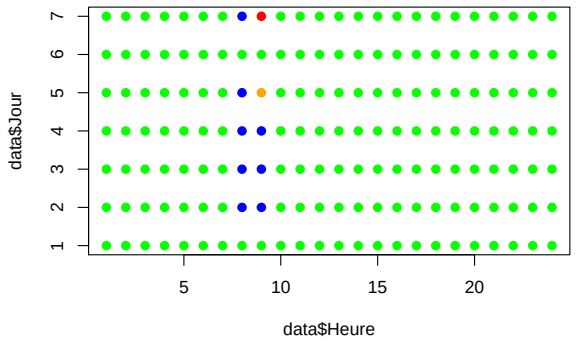
(a) Avec le critère *ward.D2*



(b) Avec le critère *complete*



(c) Avec le critère *centroid*



(d) Avec le critère *single*

FIGURE 5 • Comparaison des résultats obtenus avec quatre critères d'agrégation

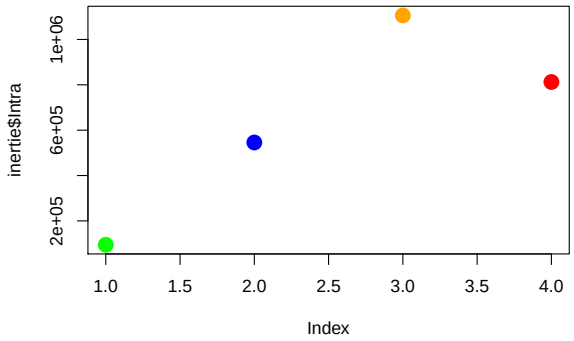
Nous pouvons analyser rapidement ces résultats :

- *Critère Ward.D2* Nous avons déjà vu que les résultats correspondaient à une vision intuitive de l'évolution des flux de circulation sur la journée.
- *Critère complete* Les résultats sont encore cohérent. On observe cependant que la classe verte (peu de circulation) tend à prendre plus d'espace. En particulier, cette dernière commence à empiéter fortement sur la classe bleue.

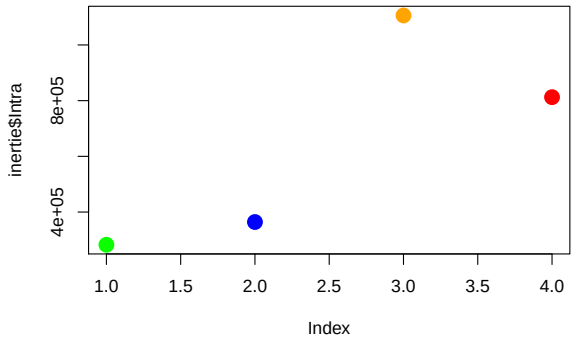
- *Critère* `centroid` Par rapport à `complete`, la classe verte veste stable, cependant la classe intermédiaire bleue mange quasiment tout le reste de l'espace. On reconnaît tout de même les deux grandes périodes de la journée (jour/nuit), mais rien de plus.
- *Critère* `single` Le résultat n'a plus de sens, si ce n'est de mettre en avant le pic de circulation de début de journée.

Il est intéressant d'observer que les méthodes fortement liés aux distances entre éléments ont complètement cassé la représentativité des classes avec le plus de variance intra-classe, signe que cette dernière ne peut être préservée par ces dernières.

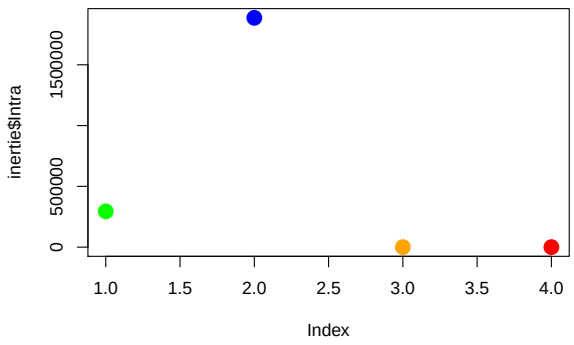
Ces différents éléments peuvent être visibles si on visualiser les retours d'inertie pour les différents critères. Ceci est représenté à la Figure 6.



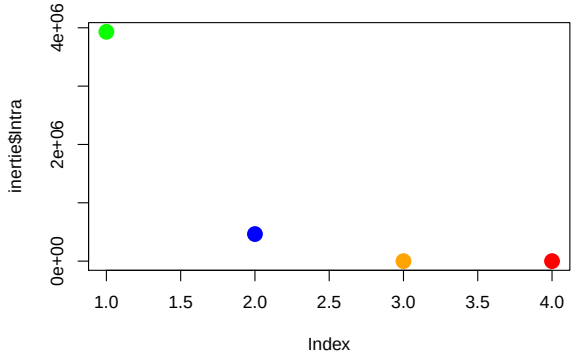
(a) Avec le critère `ward.D2`



(b) Avec le critère `complete`



(c) Avec le critère `centroid`



(d) Avec le critère `single`

FIGURE 6 • Comparaison des résultats obtenus avec quatre critères d'agrégation pour l'inertie

Quelques remarques rapides peuvent être faites sur ces figures en lien avec ce qui a été vu concernant les clusterings :

- Les critères `ward.D2` et `complete` qui fournissaient des résultats comparables fournissent la même conclusion concernant l'inertie. On remarque cependant que la classe bleue devient plus concentrée, désormais que la classe verte en a intégrée une grande partie. De même pour le rouge qui a perdu un peu de terrain.
- Pour le critère `centroid`, on remarque que la classe bleue qui devient majoritaire a intégré toute la variabilité des classes rouge et orange qui voient cette mesure s'effondrer.
- Même constat avec `single`, si ce n'est que cette fois ci c'est la classe verte qui a absorbée toutes les variabilités du modèle.

3 • Utilisation de K-Means

3.1. Application de la classification

On commence par appliquer la classification à quatre classes pour être en accord avec ce qui précède. Nous ne fixons pas encore les noyaux initiaux, ce qui est fait par le code suivant :

```
library(cluster)
k=4
kmeans=pam(data2,k)
clusplot(kmeans)
```

On pourra remarquer que le *plot* réalise une analyse en composantes principales. Nous obtenons alors la Figure 7 qui représente les clusters calculés sur ces deux composantes principales.

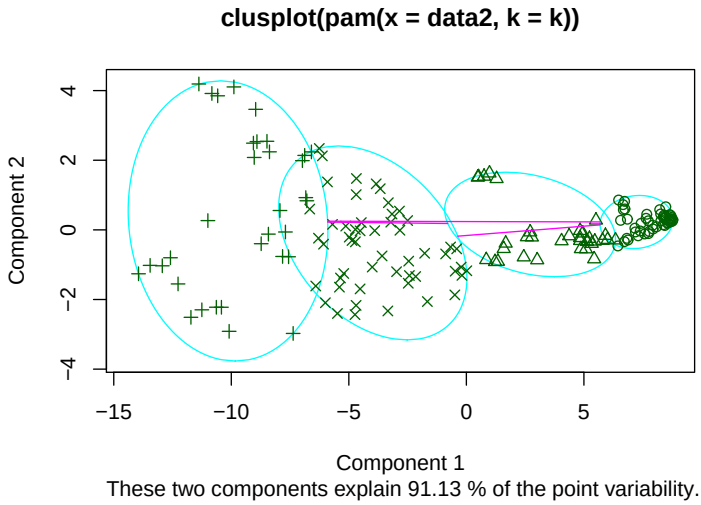


FIGURE 7 • Clusters obtenus sur les composantes principales calculées par *kmeans*

Nous observons donc bien nos quatre clusters, cependant la représentation n'est pas en accord avec celle précédemment proposée. Nous allons donc ré-utiliser le code précédent pour afficher les résultats de K-Means.

```
clusterk=kmeans$clustering
plot(data$Heure,data$Jour,col=c("green","blue","orange","red")[clusterk],pch=16,cex=1.2)
```

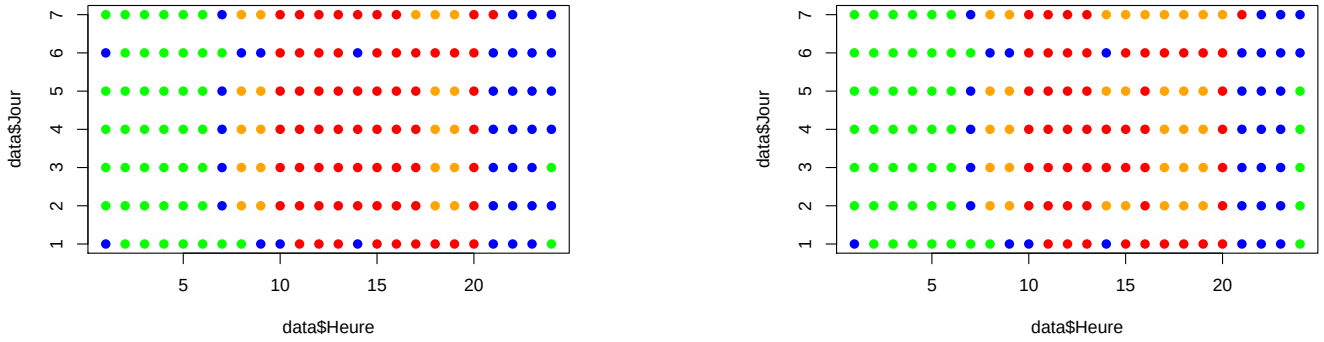
Nous obtenons alors la représentation de la Figure 8, cette figure reprenant également les résultats obtenus précédemment avec *ward.D2*.

Nous observons des résultats comparables, cependant les clusters obtenus avec K-Means font apparaître une plus grande importance pour le régime intermédiaire (orange). Il est alors possible de faire un tracé de l'inertie comme nous l'avions fait pour la méthode ascendante, les résultats obtenus sont similaires. Ceci montre en particulier que l'utilisation du ACP qui a retenue deux composantes expliquant 91% de la variance (voir Figure 7) est une hypothèse viable ici.

3.2. Choix des noyaux initiaux

Il est possible de tirer aléatoirement les noyaux initiaux. Pour cela, nous utilisons l'option *medoids* de la méthode *pam* en lui fournissant en particulier le nombre de clusters attendus. Dans notre cas, nous conservons toujours l'idée d'obtenir quatre clusters. Le code suivant permet de réaliser cette opération :

```
kmeans2=pam(data2,k,medoids=sample(1:nrow(data),k))
clusplot(kmeans2)
clusterk2=kmeans2$clustering
plot(data$Heure,data$Jour,col=c("green","blue","orange","red")[clusterk2],pch=16,cex=1.2)
```



(a) Avec classification hiérarchique

(b) Avec la méthode K-Means

FIGURE 8 • Comparaison des résultats obtenus avec K-Means et la classification hiérarchique

Nous pouvons vois que différents points de départs induisent différentes situations finales. Par exemple, la Figure 9 regroupe divers résultats obtenus pour différents points de départs choisis aléatoirement.

FIGURE 9 • Clusters obtenus en partant de points choisis aléatoirement

On observe des clusters finaux proches, mais pas exactement similaires dû à l'aspect aléatoire du tirage au sort.

3.3. Choix du meilleur point de départ

Pour réaliser ce choix, il peut être intéressant de regarder la densité autours des répartitions obtenues pour les différents points. Pour cela, nous allons réaliser différents modèles et évaluer la densité des clusters obtenus sur les dfférents modèles. Ceci a été réalisé avec le code suivant :

```
nsample=1000
av=rep(0,nsample)
for (i in 1:nsample) {
  kmeans2=pam(data2,k,medoids=sample(1:nrow(data),k))
  av[i]=sum(kmeans2$clusinfo[,3])
}
plot(density(av))
abline(v=sum(kmeans$clusinfo[,3]),col="red",lwd=2)
```

Nous obtenons en sortie la représentation des densités obtenues pour les différents modèles ainsi que la représentation (en rouge) du modèle obtenu initiallement. Ce résultat vous est proposé à la Figure 10.

On remarque ainsi que le modèle choisi est le modèle avec la plus grande densité intra, ce qui était attendu. Ce critère peut donc nous permettre de conserver les modèles avec les meilleures densité, et il semblerait que dans ce cas le modèle initial ait été le meilleur.

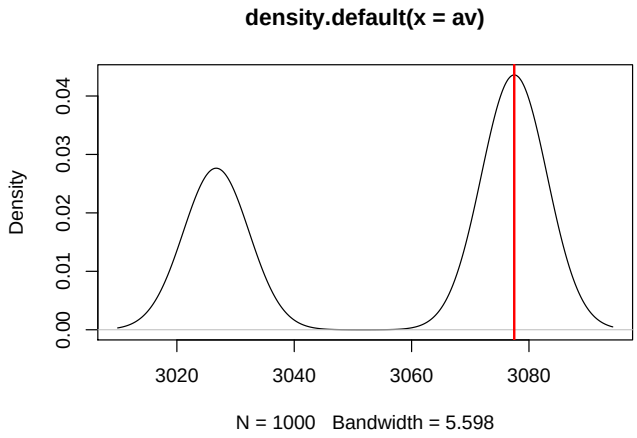


FIGURE 10 • Représentation des densités des obtenues pour les partitions sur 1000 essais

4 • Utilisation de la matrice transposée

4.1. Intérêt et entraînement du modèle

Pour réaliser cette opération, nous pouvons utiliser le code suivant :

```
d=dist(t(data2))
h=hclust(d,method="complete")
plot(h)
n=2
rect.hclust(h,k=n,border=c("red","blue"))
cluster=cutree(h,n)
```

Le passage à la transposée fait que nous passons d'une matrice où les lignes sont les relevés (par jour et heure) et les colonnes les données des capteurs à une matrice où les lignes sont les capteurs et les colonnes les relevés par jour et heure. Ainsi, nous allons apprendre un clustering sur les capteurs et non plus un clustering sur les relevés. De plus, le clustering sur les capteurs sera réalisé en fonction des flux routiers, ce qui peut permettre de regrouper des zones de la ville par similarité de comportement routier.

Réaliser un clustering sur ces données fournie alors le dendogramme de la Figure 11. Sur ce dernier, nous avons préalablement remarqué que le nombre de deux clusters semblait le plus approprié, et avons donc tracé ces clusters sur le dendogramme qui vous est présenté à la Figure 11.

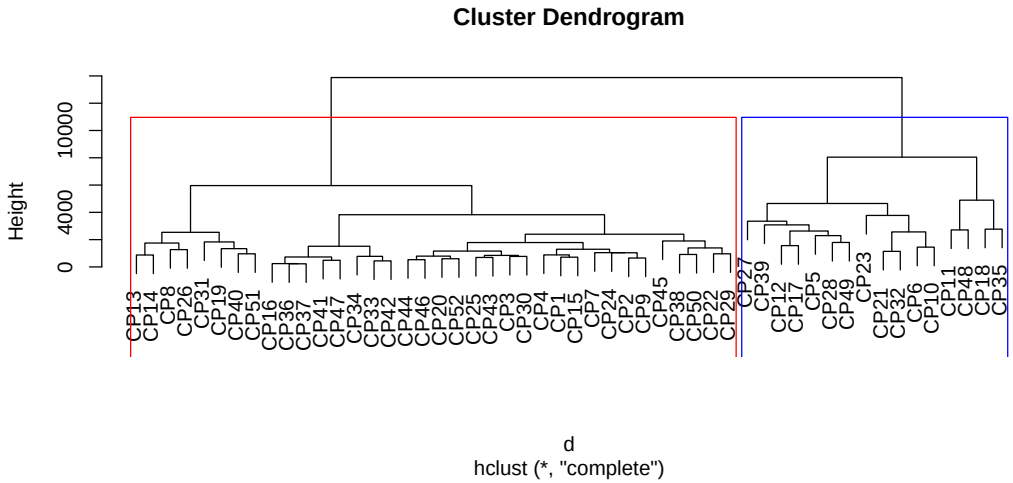


FIGURE 11 • Dendogramme obtenu en étudiant la transposée

L'intérêt de la manipulation est alors de savoir s'il est possible de regrouper les clusters par « famille » et rapprocher les comportements des flux routiers à divers endroits de la ville.

4.2. Intérête d'utiliser une PCA

On peut se demander s'il serait intéressant dans ce cas de procéder à une analyser par composante principale. Pour se faire, nous avons réalisé le code suivant, dans lequel cette analyse est réalisée et suite à laquelle nous représentons nos clusters dans l'espace obtenu en conservant les deux principales composantes.

```
library(ade4)
pca=dudi.pca(t(data2),n=2,scannf=FALSE)
plot(pca$li,col=c("red","blue")[cluster],pch=16)
```

Nous obtenons *in fine* le graphique de la Figure 12.

Nous observons que la séparation entre les deux clusters n'est pas flagrante sur les deux composantes conservées. Nous pouvons cependant tester la part de variance expliquée par ces deux variables avec le code suivant :

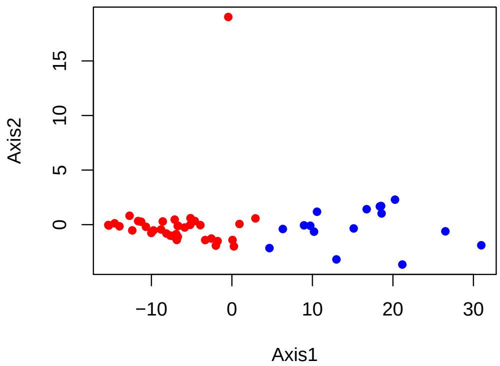


FIGURE 12 • Résultats obtenus pour la PCA sur les données transposées

```
pca_val <- pca$eig/sum(pca$eig)*100
sum(pca_val[1:2])
```

On observe que les deux composantes conservées expliquaient cependant 87% de la variance. Ainsi, il pourraît être intéressant de conserver un peu plus de variable pour avoir un résultat plus cohérent et peut être mieux voir cette séparation.

4.3. Exploitation du modèle

Pour terminer, il peut être intéressant de regarder quels capteurs ont été regroupés les uns et les autres dans le modèle que nous avons entraîné (la PCA servait uniquement à regarder si on obtenait une représentation graphique intéressante des clusters sur les composantes retenues). Pour cela, nous disposons de la carte de la Figure 13.

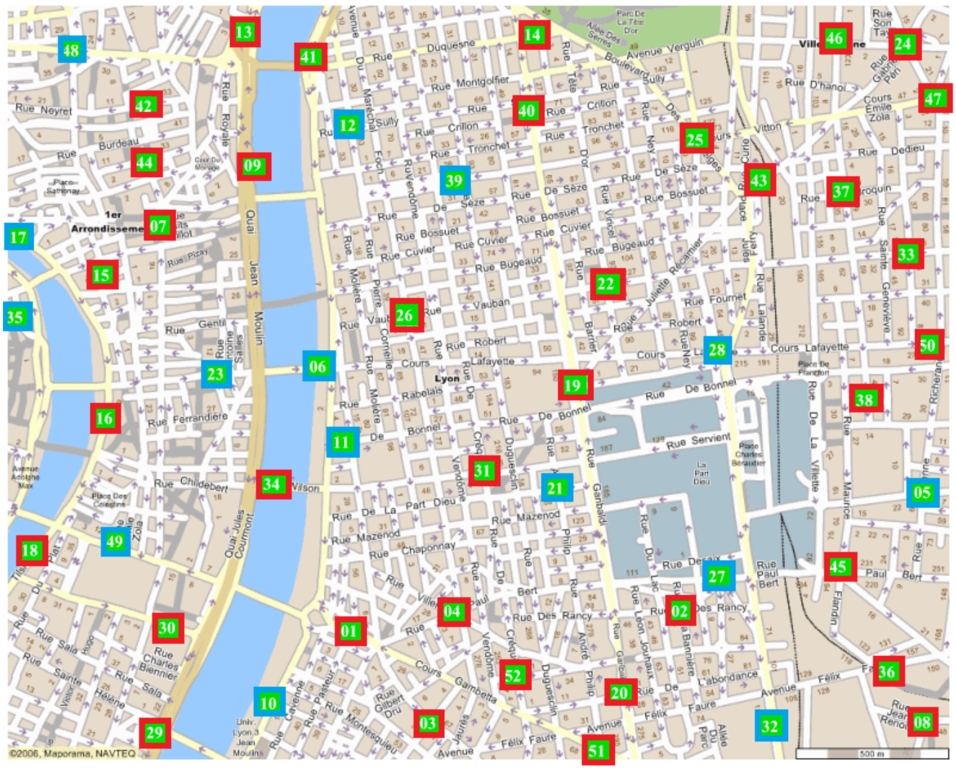


FIGURE 13 • Carte des capteurs utilisés

Ensuite, en regardant les clusters effectués à la Figure 11, nous avons coloré chaque capteur de la carte en fonction de son cluster afin d'obtenir un point de vue plus générique de la question.

Un premier constat, la ville est majoritairement rouge, avec quelques inclusions de bleu. Pour savoir si le clustering a réellement un sens, nous allons donc nous attarder sur les éléments en blue et voir s'ils correspondent bien à une réalité. Ces éléments sont présents dans les zones suivantes :

- Le cinquième arrondissement (quartier St Jean), qui correspond à une zone difficilement empruntable et peu utilisée par les voitures.
- Quais du Rhône rive gauche, qui est une rive à sens unique.
- Boulevard de la Croix Rousse, à sens unique au niveau du capteur.
- Quelques rues autours de la Part Dieu, pour lesquelles la circulation des voitures est moins facile, les zones de bus étant plus mises en avant.
- Quelques autres points éparts.

On remarque donc que le cluster bleu semble correspondre à une réalité sur le terrain, à savoir des zones difficiles d'accès pour les voitures (zones pavées, sens uniques,...),et pour lesquelles l'affluence de voiture dans la journée n'est à priori pas constante.

4.4. Essai avec quatre clusters

La solution avec deux clusters, malgré ses intérêts, permet peu d'interprétation, nous nous sommes alors amusé à re-tenter l'expérience mais avec cette fois-ci quatre clusters. L'idée est encore une fois de voir si les clusters réalisés en fonction des flux de voiture sont explicables sur la carte.

Nous allons alors rejouer le même code, mais en retenant quatre clusters. Nous obtenons alors le dendogramme (avec les clusters), de la Figure 14.

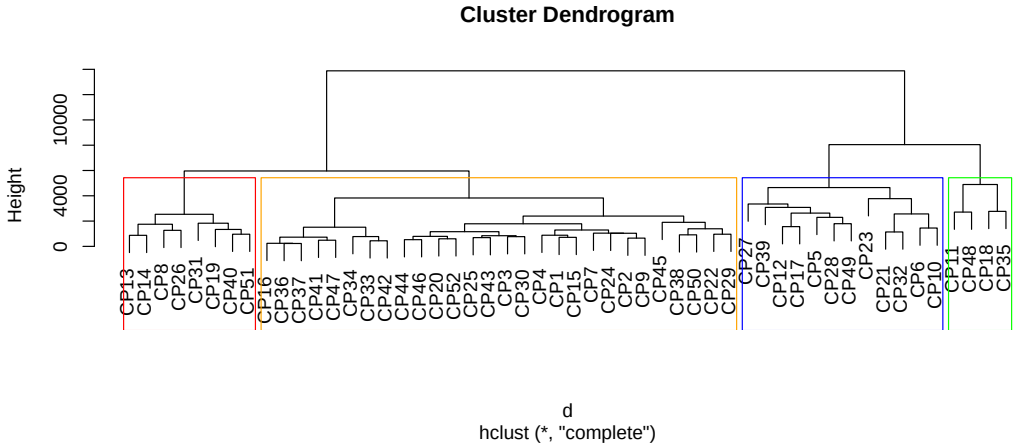


FIGURE 14 • Dendogramme et clusters dans le cas où quatre clusters sont retenus sur les capteurs

Nous pouvons à nouveau compléter la carte des capteurs, et obtenons la nouvelle carte de la Figure 15. Concernant les clusters, nous pouvons essayer d'en dégager un peu plus de sens :

- *Cluster orange* Ce dernier se démarque en concernant principalement des capteurs dans les zones périphériques de la ville qui sont principalement résidentielles (autours du Cours Gambetta, début de Villeurbanne, pentes de la Croix Rousse,...).
- *Cluster rouge* Ce dernier semble marquer des points difficiles de circulation « presque incontournables » et donc à fort flux, comme la rue Garibaldi, la sortie du tunnel de la Croix Rousse côté Rhône ou encore quelques petites rues du sixième arrondissement.
- *Cluster vert* Ce dernier semble reprendre des zones moins fréquentées (difficulté d'accès par exemple), d'où des comportements des flux routiers différents, comme c'est le cas pour le quartier St Jean.

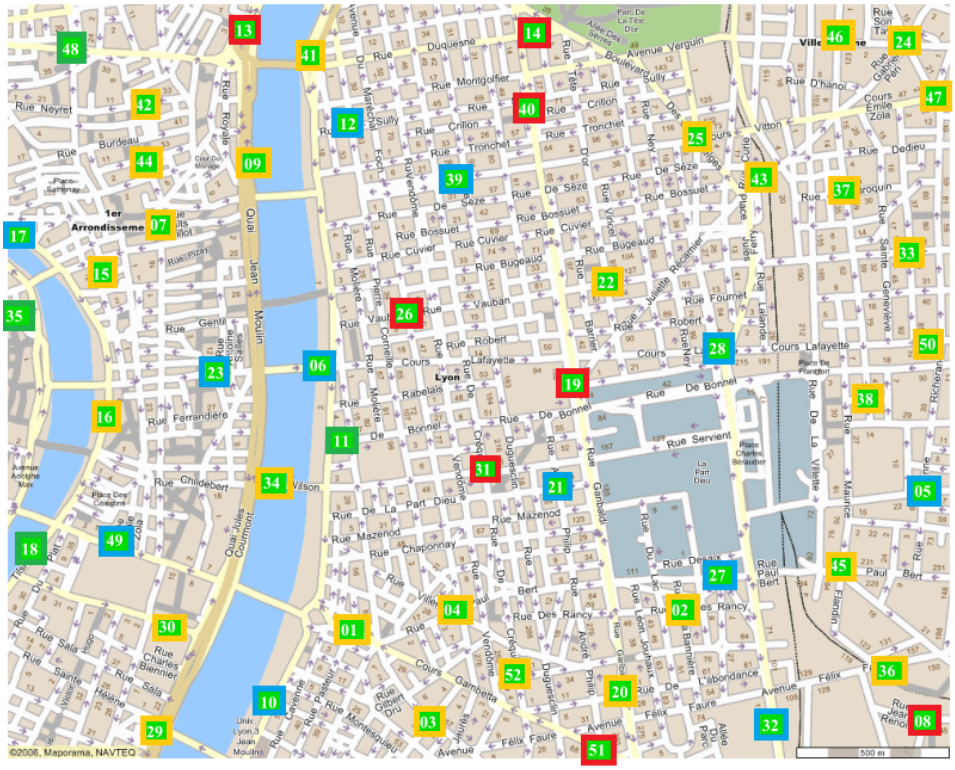


FIGURE 15 • Répartition des capteurs dans le cas où quatre clusters sont conservés

- *Cluster bleu* Ce dernier semble être intermédiaire et représenter des capteurs situés dans des rues faisant le lien entre des zones importante de la ville (par exemple autours de la Part Dieu, vers Bellecour, à proximité de ponts,...).

Cette analyse ne peut rester que superficielle, car une plus importante connaissance de terrain serait nécessaire pour réellement analyser ces regroupements proposés...