Projet de Fouille de Données
Découvrir et décrire des points d'intérêt et des évènements à partir de médias géo-localisés
Rapport

Groupe 4114
-
Brandon Da Silva Alves
Jade Prévôt

25 janvier 2022

Table des matières

Table des figures

Chapitre 1

Contexte

Le projet consiste à trouver de manière non-intrusive les zones à forte densité de touristes à moindre coût, pour améliorer les transports en communs et la vie des touristes visitant Lyon.

Ainsi, notre mission est de trouver de manière automatique des points d'intérêt intéressants dans la ville de Lyon, définis par une activité forte de prise de photos. Pour ce faire, nous disposons d'une collection de photos géo-localisées prises dans Lyon, publiées sur Flickr. Cette collection est constituée d'un jeu ancien de 80 000 photos et d'un jeu plus récent de 400 000 photos. Les informations associées à ces photos nous permettent, grâce aux techniques de fouille de données, de mettre en lumière des points d'intérêt.

Chapitre 2

Compréhension, nettoyage, visualisation et statistiques des données

2.1 Compréhension des données

Les photos analysées sont composées de plusieurs attributs :

id L'identifiant de l'image ;

user L'identifiant de l'utilisateur qui a posté l'image ;

lat La latitude où a été prise l'image ;

long La longitude où a été prise l'image ;

tags Des tags associés à la photo ;

title Le titre de l'image ;

date_taken_minute La minute à laquelle la photo a été prise ;

date_taken_hour L'heure à laquelle la photo a été prise ;

date_taken_day Le jour auquel la photo a été prise ;

date_taken_month Le mois auquel la photo a été prise ;

date_taken_year L'année à laquelle la photo a été prise ;

date_upload_minute La minute à laquelle la photo a été chargée ;

date_upload_hour L'heure à laquelle la photo a été chargée ;

date_upload_day Le jour auquel la photo a été chargée ;

date_upload_month Le mois auquel la photo a été chargée ;

date_upload_year L'année à laquelle la photo a été chargée.

Nous avons un total 420,239 photos. Ce jeu de données comporte des doublons comme nous pouvons le voir à la figure 2.1. Les attributs *id*, *user*, *lat*, *long*, *date_taken_minute*, *date_taken_hour*, *date_taken_day*, *date_taken_month*, *date_taken_year*, *date_upload_month* et *date_upload_year* sont toujours renseignés contrairement aux attributs *tags*, *title*, *date_upload_minute*, *date_upload_hour* et *date_taken_day*.

Row ID	id	user	lat	long	tags	title
Row196652	306,667,535	46629827@N00	45.777	4.858	france,zoo,lyon,animaux	P'tits Girafons
Row201070	306,667,535	46629827@N00	45.777	4.858	france,zoo,lyon,animaux	P'tits Girafons
Row185977	306,668,760	46629827@N00	45.777	4.857	france,zoo,lyon	Envoyez... :))
Row191022	306,668,760	46629827@N00	45.777	4.857	france,zoo,lyon	Envoyez... :))
Row195749	306,668,760	46629827@N00	45.777	4.857	france,zoo,lyon	Envoyez... :))
Row186464	1,140,570,2...	46629827@N00	45.778	4.855	summer,france,lyon,69,2007,parc...	Parc de la Tête ...
Row191512	1,140,570,2...	46629827@N00	45.778	4.855	summer,france,lyon,69,2007,parc...	Parc de la Tête ...
Row383457	3,836,705,0...	46629827@N00	45.797	4.833	summer,france,lyon,69,2009,rhôn...	Église romane ...
Row387529	3,836,705,0...	46629827@N00	45.797	4.833	summer,france,lyon,69,2009,rhôn...	Église romane ...
Row392188	3,836,705,0...	46629827@N00	45.797	4.833	summer,france,lyon,69,2009,rhôn...	Église romane ...
Row417206	3,836,705,0...	46629827@N00	45.797	4.833	summer,france,lyon,69,2009,rhôn...	Église romane ...

FIGURE 2.1 – Sous-ensemble du jeu de données mettant en évidence la présence de doublons

2.2 Nettoyage des données

Premièrement, nous pouvons vérifier que les photos ont des dates de prises et de chargement cohérentes, à savoir que les dates de chargement sont postérieures aux dates de prise. Pour cela, nous pouvons utiliser un composant *Rule-based Row Filter* pour vérifier que :

- $date\_taken\_year < date\_upload\_year$
- $date\_taken\_month < date\_upload\_month$ (si $date\_taken\_year = date\_upload\_year$),
- $date\_taken\_day < date\_upload\_day$ (si $date\_taken\_month = date\_upload\_month$),
- $date\_taken\_hour < date\_upload\_hour$ (si $date\_taken\_day = date\_upload\_day$,
- $date\_taken\_minute < date\_upload\_minute$ (si $date\_taken\_hour = date\_upload\_hour$)

Nous paramétrons ce composant comme présenté à la figure 2.2. Nous avons 420,239 photos, aucune n'est filtrée par ce composant.

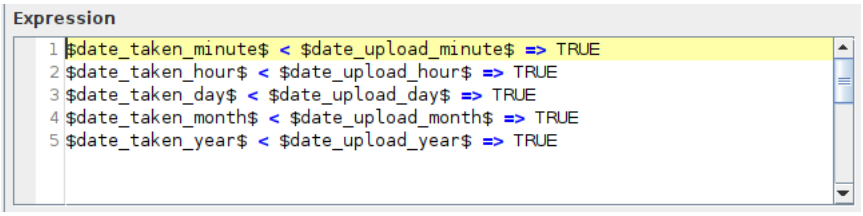


FIGURE 2.2 – Paramétrage du composant *Rule-based Row Filter*

Ensuite, toutes les informations à disposition ne sont pas utiles. En effet, nous cherchons à découvrir, décrire et prédire des évènements à partir de médias géo-localisés. Ainsi, la position et la date de prise de ces images seront utiles contrairement à la date de chargement de ces images qui n'aura aucune utilité. Nous pouvons donc filtrer les colonnes *date_upload_minute*, *date_upload_hour*, *date_upload_day*, *date_upload_month* et *date_upload_year*. De plus, certaines colonnes, les 3 dernières, sont vides (sauf quand un décalage accidentel est survenu dans un tuple). Nous pouvons également les supprimer. Pour ce faire, nous utilisons le composant *Column Filter* avec le paramétrage adéquat comme montré à la figure 2.3.

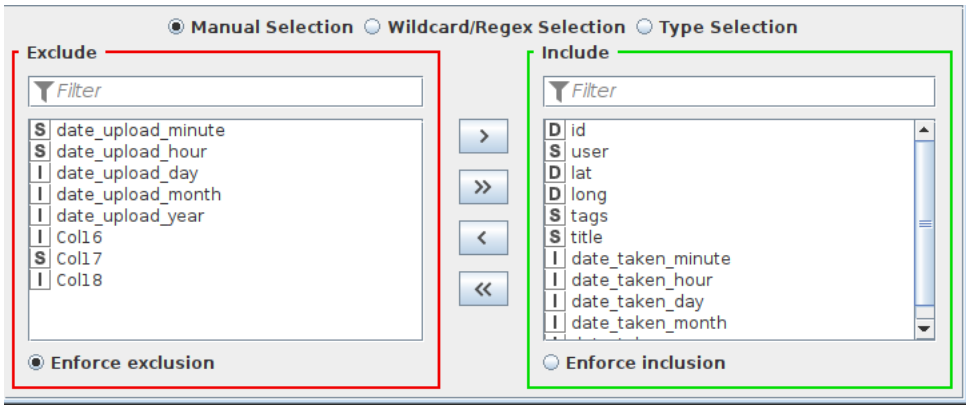


FIGURE 2.3 – Paramétrage du composant *Column Filter*

Nous pouvons ensuite supprimer les différents doublons. Une photo est caractérisée de manière unique par son attribut *id*. Nous pouvons alors garder un seul exemplaire des tuples doublons. Pour ce faire, nous utilisons le composant *GroupBy* avec le paramétrage adéquat comme montré aux figures 2.4. Après cette étape, nous passons de 420,239 photos à 156,568 photos.

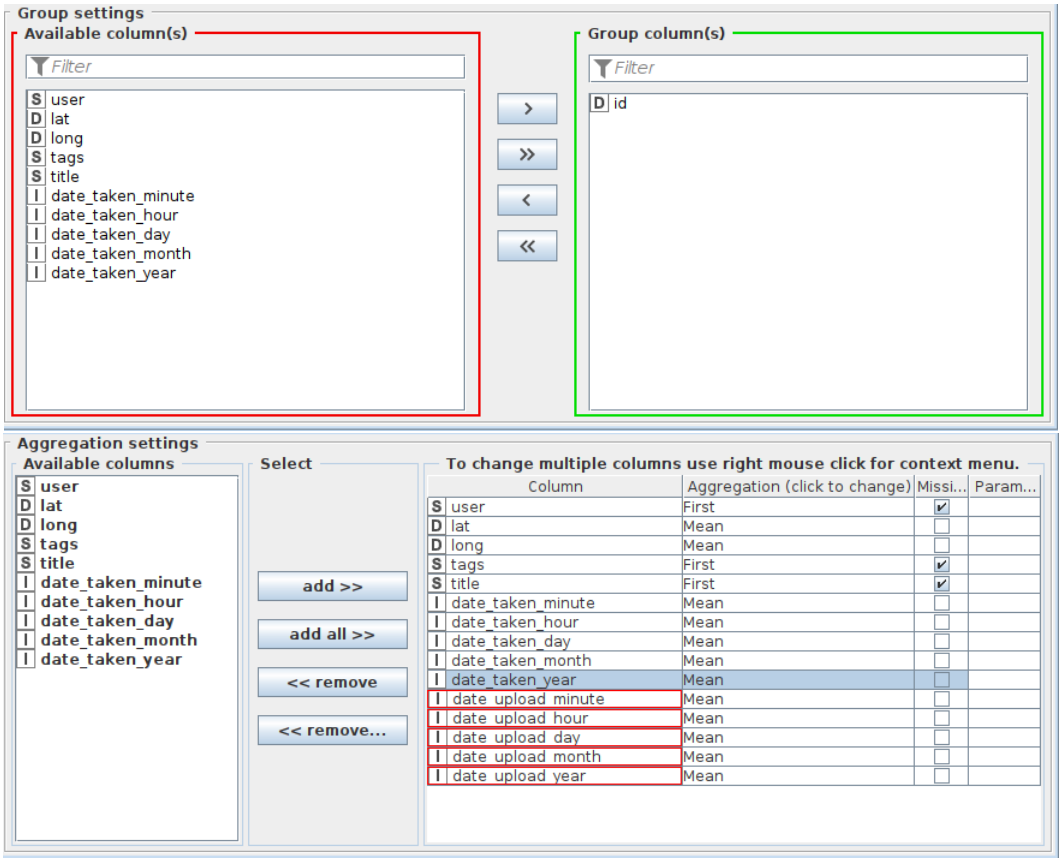


FIGURE 2.4 – Paramétrage du composant *GroupBy*

Nous pouvons maintenant nous assurer que les informations
— *date_taken_minute*,

— *date_taken_hour*,
— *date_taken_day*,
— *date_taken_month*,
— *date_taken_year*,
— *lat*,
— *long*

sont cohérentes. Nous nous assurons ici que les minutes et les secondes soient inférieures à 60 et supérieures à 0, que les heures soient inférieures à 24 et supérieures à 0, que les mois soient inférieurs à 12 et supérieurs à 0, que les latitudes soient comprises entre -90°et 90°et enfin que les longitudes soient comprises entre -180°et 80°. Pour ce faire, nous utilisons le composant *Row Filter* pour filtrer les tuples où les attributs cités précédemment ont une valeur aberrante. Nous paramétrons le composant comme montré à la figure 2.5 pour l'attribut *date_taken_minute*. Après cette étape, nous passons de 156,568 photos à 156,567 photos.

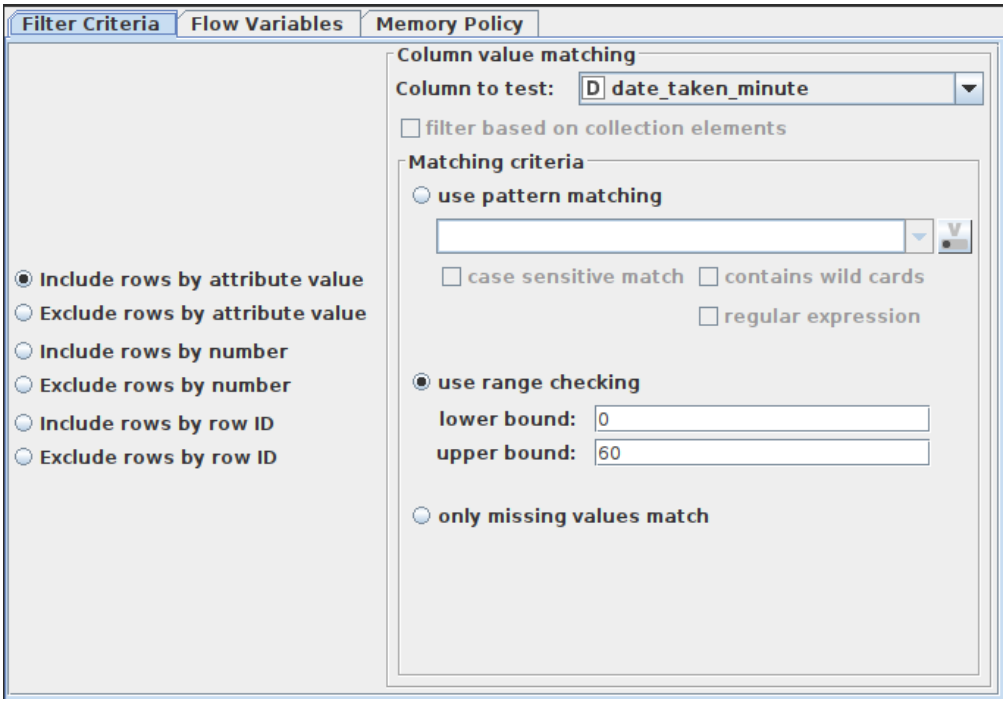


FIGURE 2.5 – Paramétrage du composant *Row Filter*

Enfin, nous filtrons les coordonnées latitude et longitude à la zone du Grand-Lyon. Nous utilisons pour cela le composant *Geo-Coordinate Row Filter*. Nous le paramétrons comme montré à la figure 2.6. Ici aucune photo n'est filtrée.

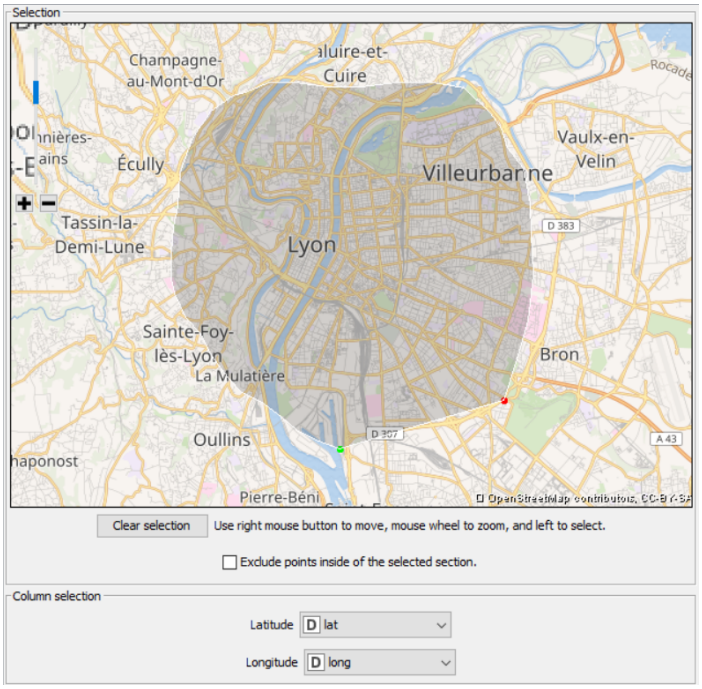


FIGURE 2.6 – Paramétrage du composant *Geo-Coordinate Row Filter*

2.3 Visualisation des données

Afin d'observer nos données et surtout les futurs clusters, nous utilisons le composant *OSM Map View*. Nous affichons les points de prise des photos grâce aux attributs *lat* et *long*. Nous faisons également apparaître *id* et *user* afin de pouvoir facilement retrouver ces images. Le paramétrage du composant est présenté à la figure 2.7.

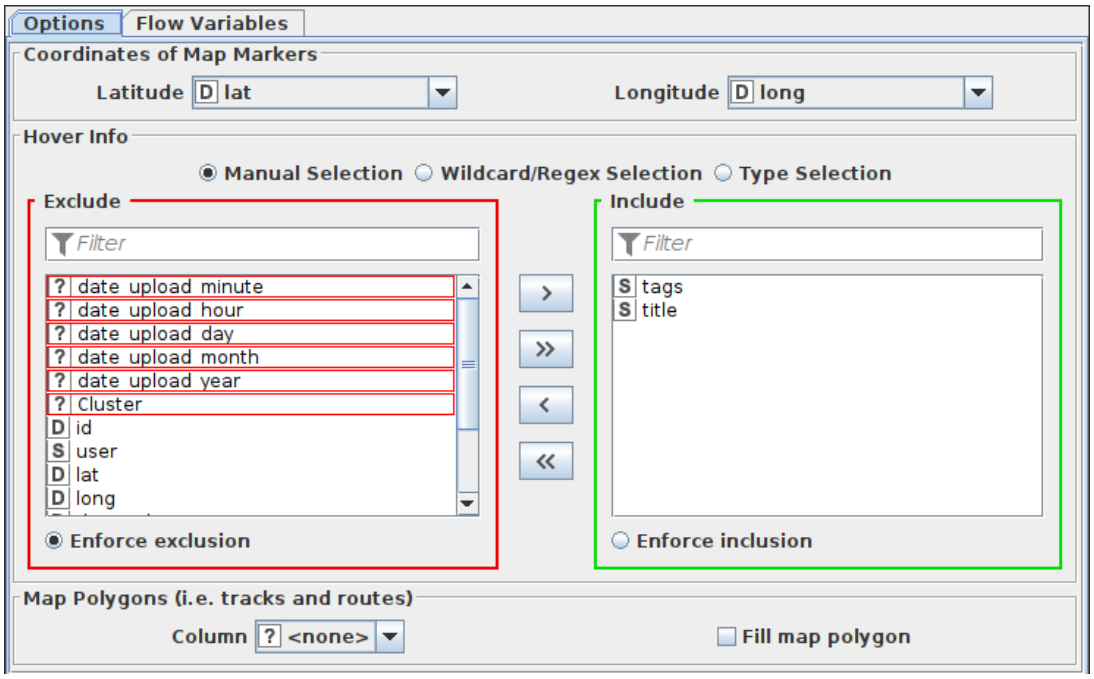


FIGURE 2.7 – Paramétrage du composant *OSM Map View*

Les photos qui ont passé la première étape de nettoyage et de filtre peuvent être localisées sur la carte à la figure 2.8.

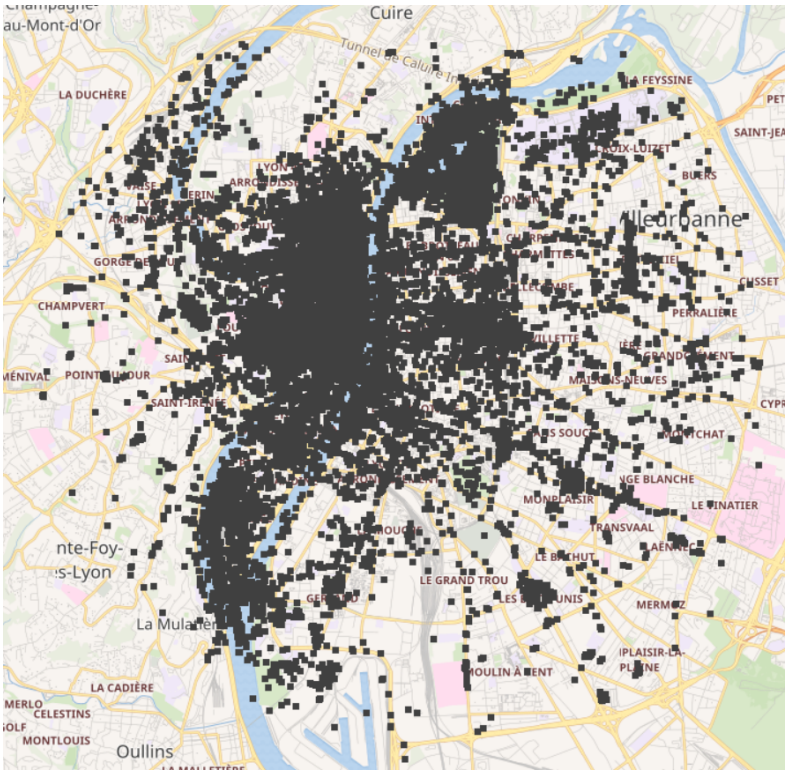
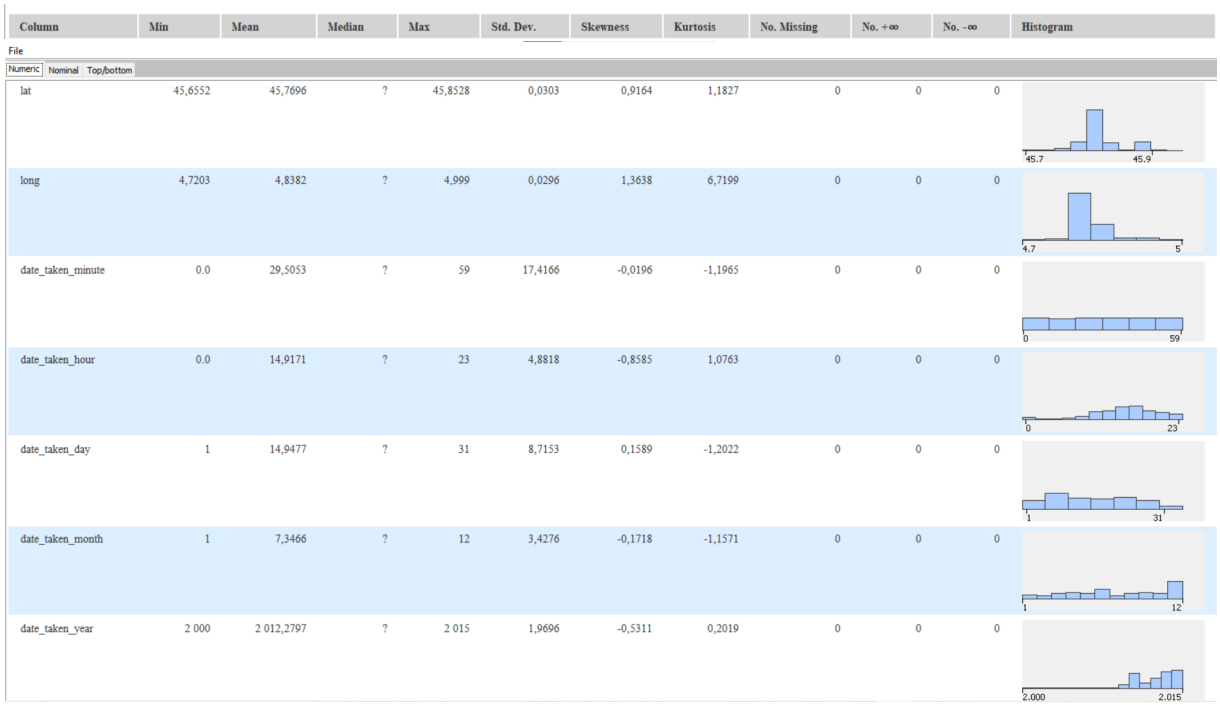


FIGURE 2.8 – Photos qui seront gardées pour le clustering

2.4 Statistiques des données

Pour examiner les statistiques relatives aux données dont nous disposons, nous exploitons le noeud *Statistics*. Nous choisissons d'observer des statistiques sur les données que nous estimons pertinentes, à savoir la latitude, la longitude et la date de capture des photos.



Column	Min	Mean	Median	Max	Std. Dev.	Skewness	Kurtosis	No. Missing	No. +∞	No. -∞	Histogram
lat	45,6552	45,7696	?	45,8528	0,0303	0,9164	1,1827	0	0	0	45.7 45.9
long	4,7203	4,8382	?	4,999	0,0296	1,3638	6,7199	0	0	0	4.7 5
date_taken_minute	0.0	29,5053	?	59	17,4166	-0,0196	-1,1965	0	0	0	0 59
date_taken_hour	0.0	14,9171	?	23	4,8818	-0,8585	1,0763	0	0	0	0 23
date_taken_day	1	14,9477	?	31	8,7153	0,1589	-1,2022	0	0	0	1 31
date_taken_month	1	7,3466	?	12	3,4276	-0,1718	-1,1571	0	0	0	1 12
date_taken_year	2 000	2 012,2797	?	2 015	1,9696	-0,5311	0,2019	0	0	0	2.000 2.015

FIGURE 2.9 – Statistiques observées

Nous observons qu'une majorité des photos ont été prises entre 2009 et 2015, et que dans l'année il y a des pics dans les périodes juin-juillet et novembre-décembre. Au cours d'un mois, il n'y a pas de période qui se démarque de manière notable des autres, si ce n'est qu'entre le 5 et le 10 du mois. Les heures de prise de photos correspondent aux heures d'activité humaine, et cela reste constant en ce qui concerne les minutes. Les statistiques de latitude et longitude nous permettent de constater de manière tranchée qu'il y a des endroits où plus de photos sont prises que d'autres, ce qui met en exergue certains points d'intérêt.

Chapitre 3

Fouille de données avec clustering

3.1 *K-means*

$n = 10$ clusters

Ici nous allons essayer de découvrir des clusters en fonction des positions géographiques des images. Nous limitions le nombre d'itérations de l'algorithme à 100. Nous commençons par chercher 10 clusters. Le résultat est présenté à la figure 3.1

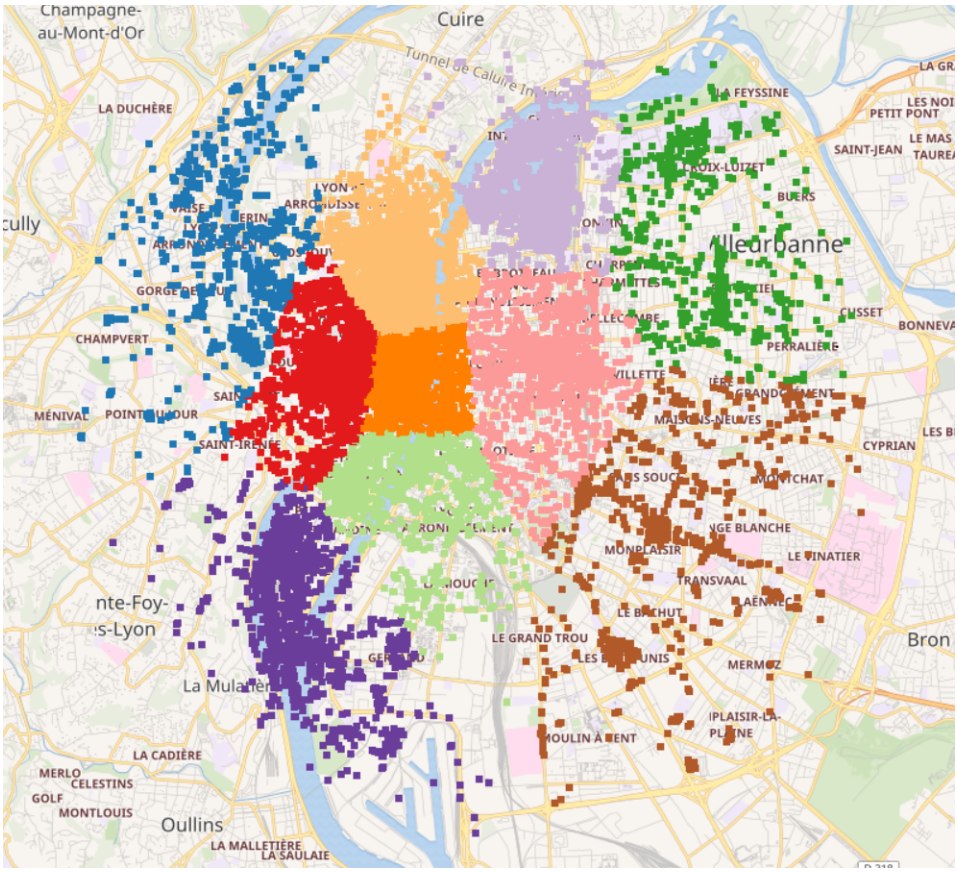


FIGURE 3.1 – Clusters découverts avec K-means avec $n = 10$ clusters

Ici nous voyons dès le premier abord que le découpage en 10 clusters n'est pas adéquat. Nous avons par exemple la ville de Villeurbanne ou encore le parc de la Tête d'Or qui sont assez bien découpés. Cependant, ces différentes zones sont trop grandes pour que les photos qui y sont prises ne représentent les mêmes points d'intérêt.

$n = 50$ clusters

Nous pouvons alors nous intéresser à augmenter le paramètre de K-means à 50 clusters. La visualisation des clusters découverts par K-means est présenté à la figure 3.2. Ici, nous pouvons, au passage, remarquer que K-means n'est pas un choix pertinent pour déterminer des clusters qui ont, *a priori*, des densités hétérogènes.

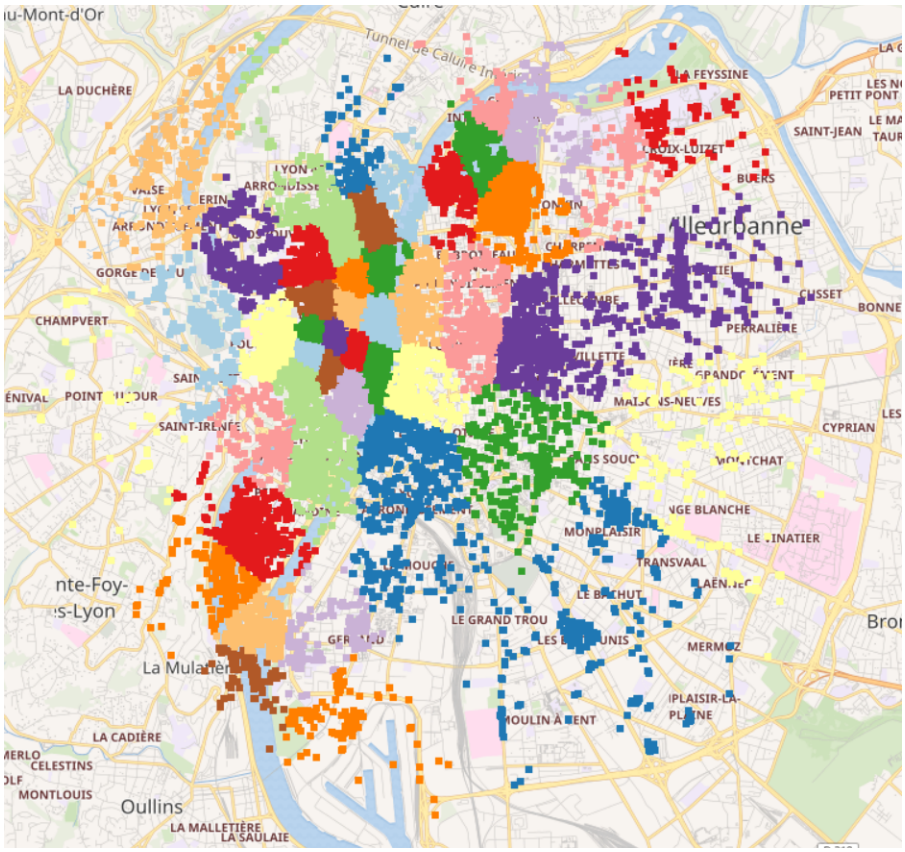


FIGURE 3.2 – Clusters découverts avec K-means avec $n = 50$ clusters

Nous pouvons voir que là où la densité du nombre de photos est plus élevée, les clusters sont plus petits. Les clusters en bordure sont beaucoup trop grands pour représenter les mêmes points d'intérêt. Cependant, nous pouvons légitimement nous demander si les clusters au centre sont plus ou moins pertinents. En regardant de plus près le cluster vert clair au niveau de *Perrache* comme présenté à la figure 3.3, nous pouvons voir que la granularité est trop grande. En effet, dans cette zone, nous avons, par exemple, aussi bien des photos de *Perrache* que des photos de la *Place Ampère* ou du quartier *Saint-Georges*. Le paramètre qui représente le nombre de clusters doit donc encore être augmenté pour avoir un découpage plus pertinent.

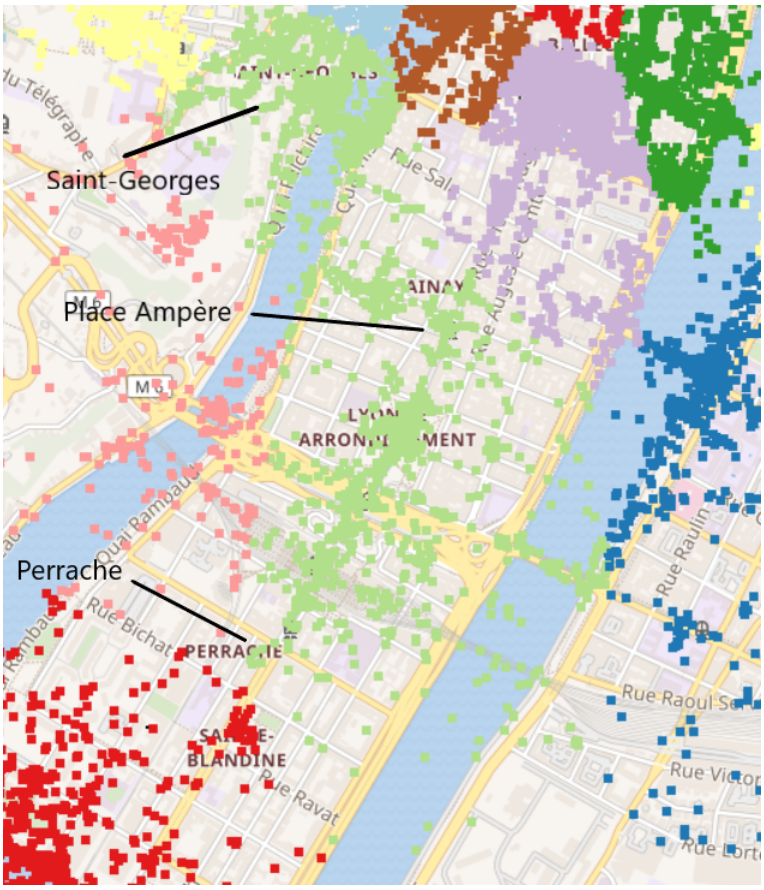


FIGURE 3.3 – Clusters découverts avec K-means avec $n = 50$ clusters - zoom

$n = 100$ clusters

Nous pouvons alors nous intéresser à augmenter le paramètre de K-means à 100 clusters. La visualisation des clusters découverts par K-means est présenté à la figure 3.4 qui représente un zoom au niveau de la *Gare Saint-Paul*. Encore une fois, il existe des clusters qui mélangent plusieurs points d'intérêts. Ici, par exemple, nous avons des photos de la *Gare Saint-Paul* et de la *Fresque des Lyonnais* qui sont confondues dans un même cluster.

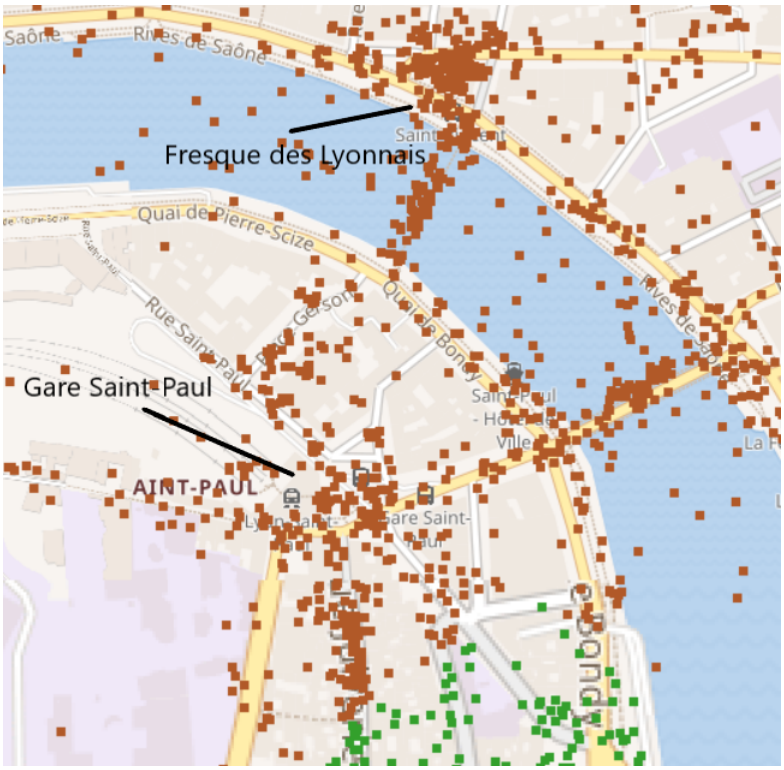


FIGURE 3.4 – Clusters découverts avec K-means avec $n = 100$ clusters - zoom

$n = 300$ clusters

Nous pouvons alors nous intéresser à augmenter le paramètre de K-means à 300 clusters. La visualisation des clusters découverts par K-means est présentée à la figure 3.5 qui représente un zoom au niveau du quartier *Bellecour*. Ici, nous avons des clusters pertinents qui apparaissent. Nous avons par exemple la *Place de la République*, la *Place des Jacobins*, la *Place des Célestins* qui sont clairement identifiées. Cependant, nous remarquons que par exemple le *Pont Bonaparte* ou encore la *Place Bellecour* sont divisés en plusieurs clusters. La granularité est pertinente pour certains points d'intérêt mais pas pour d'autres. Enfin, nous pouvons remarquer que l'algorithme K-means a quelques difficultés avec des formes non globulaires comme par exemple au niveaux des différents ponts qui sont subdivisés en plusieurs clusters.

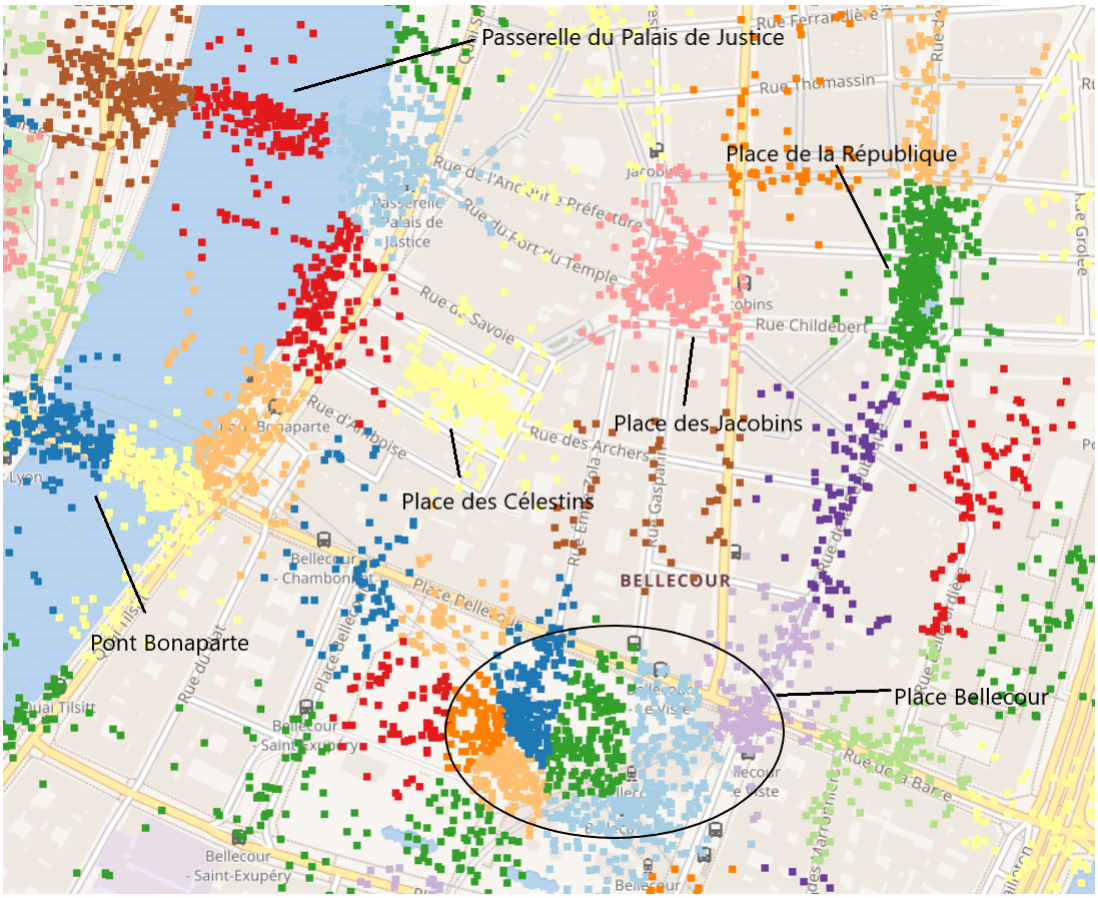


FIGURE 3.5 – Clusters découverts avec K-means avec $n = 300$ clusters - zoom

Conclusion

Au cours de ces itérations, nous avons pu remarquer que K-means n'était pas stable. De plus ce dernier n'est pas pertinent pour des clusters qui nous le savons n'auront pas des densités homogènes. Il faudrait alors exécuter des K-means sur des zones de densités homogènes. Enfin, cet algorithme a du mal à traiter les formes non globulaires. Intéressons nous maintenant au clustering hiérarchique.

3.2 *Clustering hiérarchique*

La complexité de l'algorithme de clustering hiérarchique est relativement importante. Il n'est pas envisageable d'utiliser cet algorithme sur la totalité de nos photos du Grand Lyon. Nous optons donc pour l'utiliser sur une zone géographique restreinte. Nous prenons l'exemple du *Parc de la Tête d'Or*. Nous fixons notre nombre de clusters à $n = 20$. En effet, d'après la figure 3.6, nous estimons le nombre de points d'intérêt dans cette zone géographique à 20.

FIGURE 3.6 – Plan du *Parc de la Tête d'Or*

Single

Avec le linkage *single*, la similarité entre deux clusters est basée sur les deux points les plus similaires entre les différents clusters. Le problème de cette méthode est qu'elle est très sensible au bruit et aux *outliers*. Ce type de linkage est en général bien adapté pour découvrir des clusters nettement séparés, ce qui n'est pas le cas ici. En effet, comme montré à la figure 3.7, le résultat du clustering avec ce linkage n'est qu'un seul gros cluster et dix-neuf autres très petits.

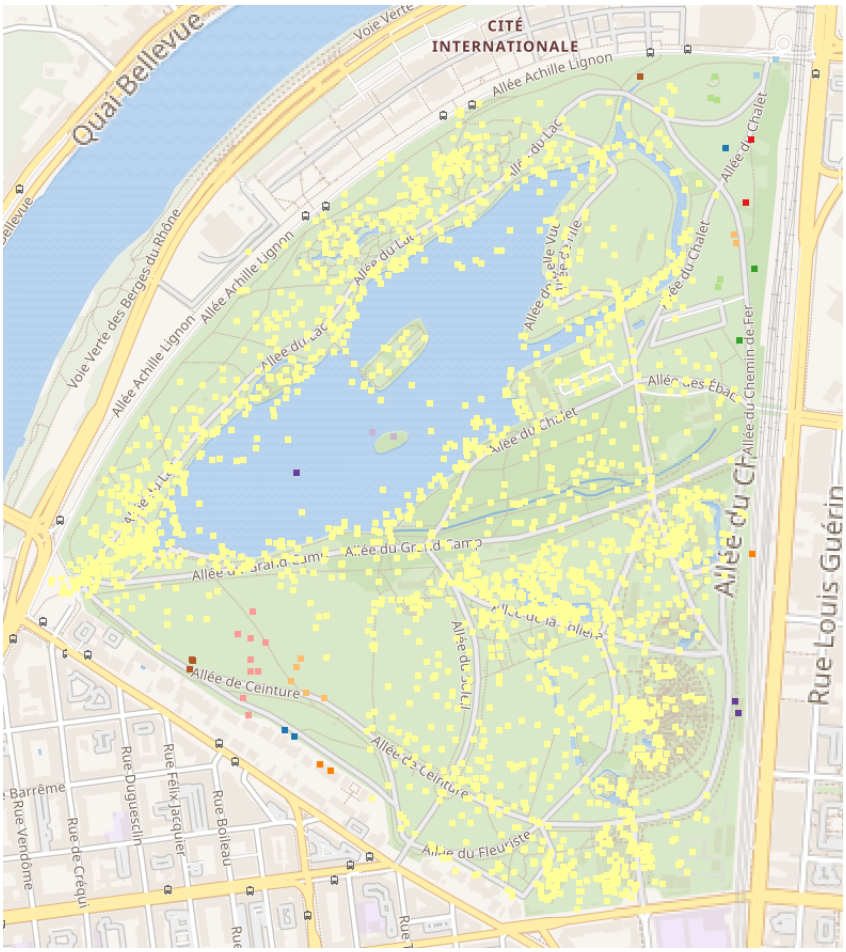


FIGURE 3.7 – Clustering hiérarchique au niveau du *Parc de la Tête d'Or* - linkage *single*

Complete

Le linkage *complete* établit que la similarité de deux clusters est basée sur les deux moins similaires dans les différents clusters. Il est déterminé par toutes les paires de points dans les deux clusters. Il est moins sensible au bruit et aux données aberrantes. Par contre, il a tendance à casser les gros clusters, et il peut être biaisé vers des clusters globulaires. Ainsi, dans notre cas, il n'est pas très pertinent d'utiliser cette méthode de linkage pour déterminer les clusters au niveau du *Parc de la Tête d'Or*. En effet, certains points d'intérêt s'étendent sur une longue distance et ceux-ci pourraient se voir séparer en plusieurs clusters différents, et notre clustering serait alors moins adéquat.

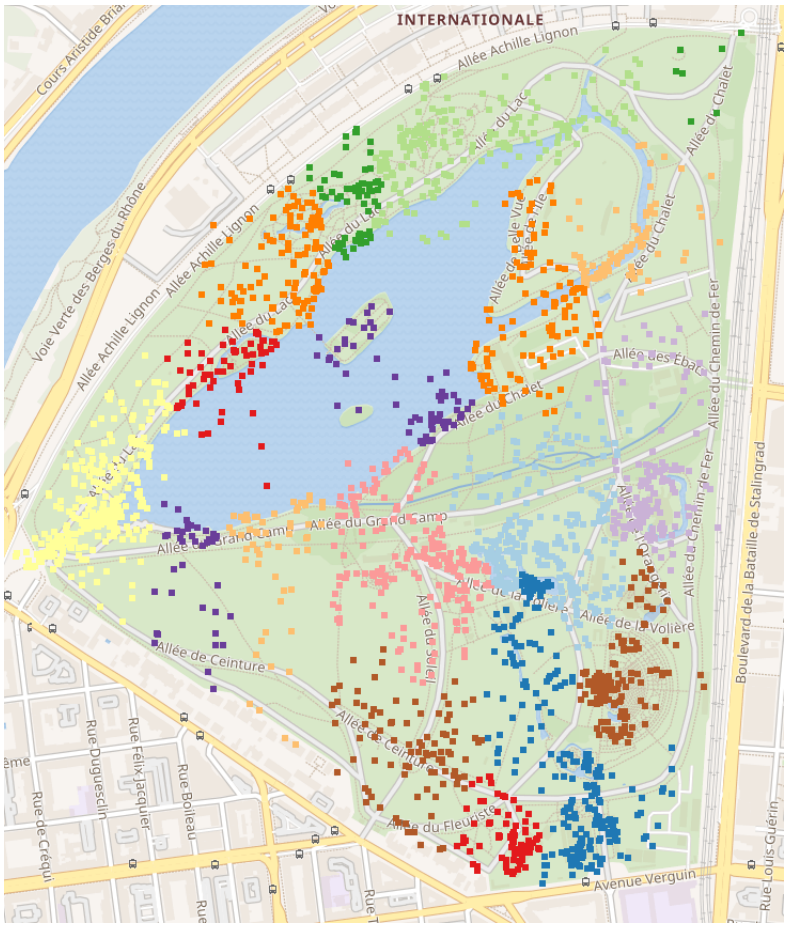


FIGURE 3.8 – Clustering hiérarchique au niveau du *Parc de la Tête d'Or* - linkage *complete*

Average

On a trouvé un clustering satisfaisant avec le linkage *average*. La proximité de deux clusters est la moyenne de la proximité par paires entre les points des deux clusters. Ce linkage est un compromis entre le linkage *single* et le linkage *complete*. Il est avantageux car moins sensible au bruit et aux données aberrantes. Il est biaisé vers les clusters globulaires, ce qui n'est pas un problème dans notre cas car nous nous attendons à avoir des clusters globulaires. Tout cela nous permet d'avoir des clusters très similaires aux points d'intérêt spécifiés sur la carte du *Parc de la Tête d'Or* de la figure 3.6.

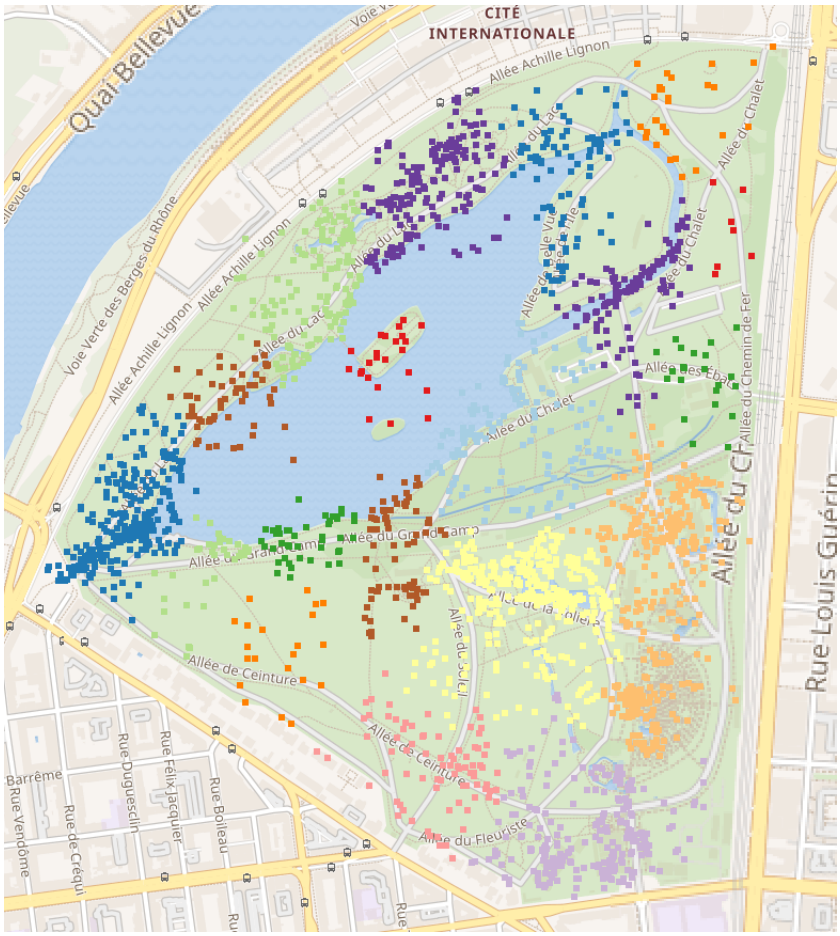


FIGURE 3.9 – Clustering hiérarchique au niveau du *Parc de la Tête d'Or* - linkage *average*

3.3 *DBSCAN*

Nous allons ici nous intéresser à la découverte de clusters avec l'algorithme DBSCAN. Nous rappelons que l'algorithme DBSCAN utilise 2 paramètres : la distance ϵ et le nombre minimum de points $minPts$ devant se trouver dans un rayon ϵ pour que ces points soient considérés comme un cluster. Les paramètres d'entrées sont donc une estimation de la densité de points des clusters. L'idée de base de l'algorithme est ensuite, pour un point donné, de récupérer son ϵ-voisinage et de vérifier qu'il contient bien $minPts$ points ou plus. Ce point est alors considéré comme faisant partie d'un cluster. On parcourt ensuite l'ϵ-voisinage de proche en proche afin de trouver l'ensemble des points du cluster.

$\epsilon = 0.002$, $minPts = 10$

Nous commençons avec une valeur de ϵ égale à 0.002 et une valeur de $minPts$ égale à 10. Nous obtenons le résultat présenté à la figure 3.10. Ici, nous pouvons remarquer que le clustering résultant n'est pas très intéressant. En effet, nous avons par exemple la *Place de la République* et la *Place Bellecour* qui sont confondus dans un seul et même cluster. De même, nous avons également la *Cathédrale Saint-Jean*, le *Musée des Arts de la Marionnette*, le *Musée Cinéma et Miniature*, le *Pont Bonaparte*, le *Théâtre des Célestins* et la *Passerelle du Palais de Justice* qui sont dans un seul cluster. Les clusters, ici, sont trop grands. Nous allons chercher à agrandir le rayon ϵ afin d'avoir des clusters plus petits.

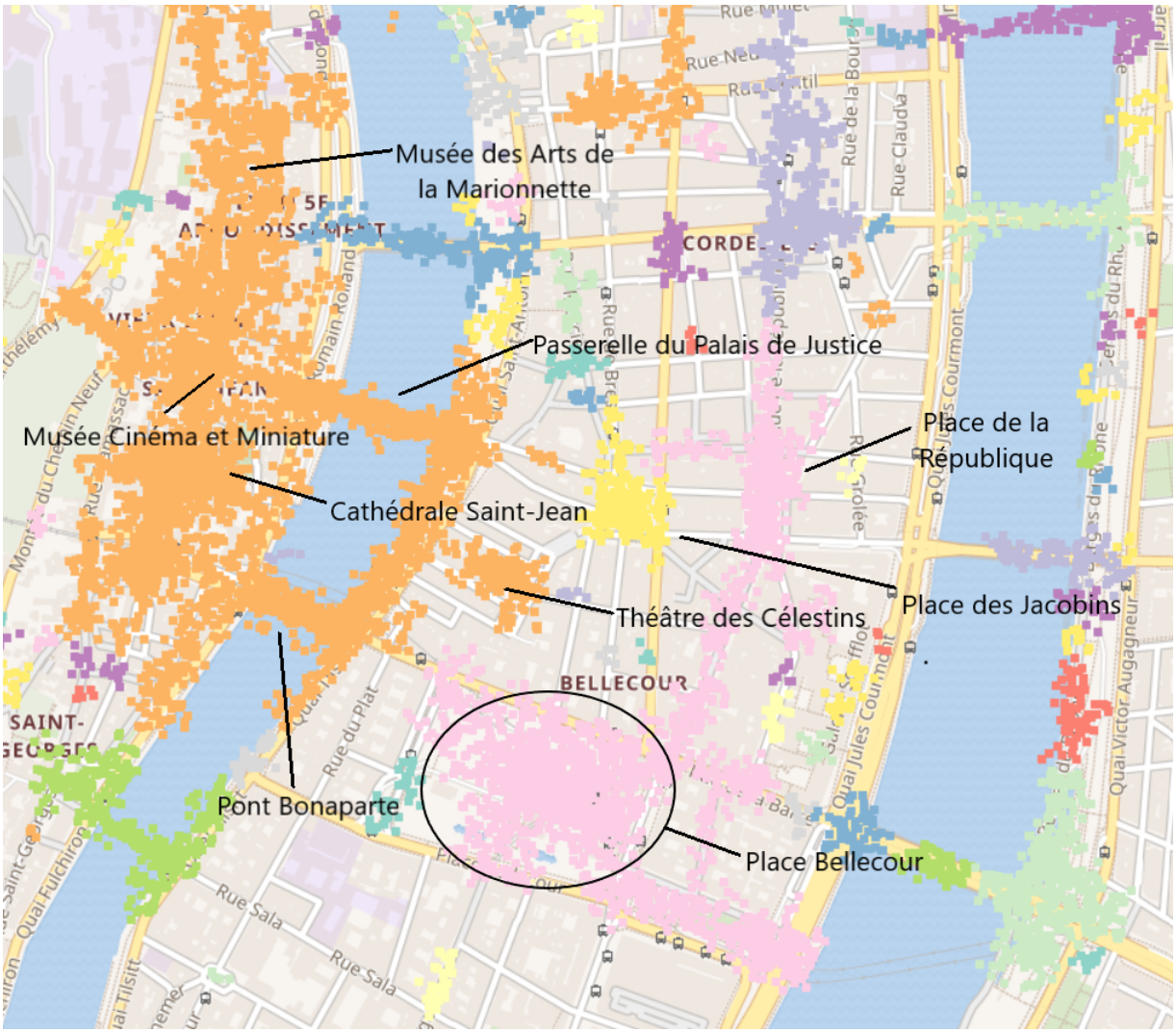


FIGURE 3.10 – Clustering avec DBSCAN ($\epsilon = 0.002$, $minPts = 10$)

$\epsilon = 0.0005$, $minPts = 10$

Nous continuons avec une valeur de ϵ égale à 0.0005 et une valeur de $minPts$ égale à 10. Nous obtenons le résultat présenté à la figure 3.11. Ici, nous remarquons que nous avons l'effet opposé que ce que nous avions obtenu au premier essai. Nous sommes allés "trop loin" en diminuant la valeur de ϵ. En effet, nous voyons tout d'abord que la taille des clusters a diminué. Ici, beaucoup de points ont été considérés comme du bruit par l'algorithme et ont donc été éliminés. De plus, nous pouvons également remarquer que certains points d'intérêt sont considérés comme étant composés de plusieurs clusters. C'est par exemple le cas de la *Place Bellecour*. Essayons de réajuster le paramètre ϵ.

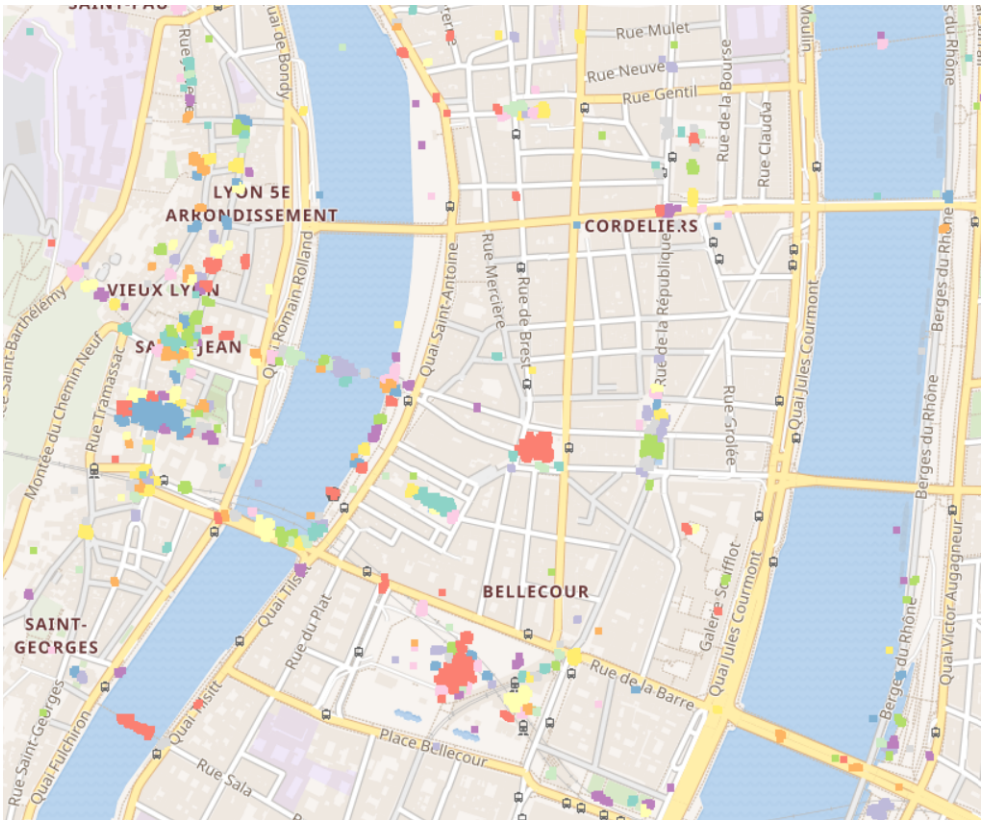


FIGURE 3.11 – Clustering avec DBSCAN ($\epsilon = 0.0005$, $minPts = 10$)

$\epsilon = 0.001$, $minPts = 10$

Nous continuons avec une valeur de ϵ égale à 0.001 et une valeur de $minPts$ égale à 10. Nous obtenons le résultat présenté à la figure 3.12. Nous avons ici un résultat un peu plus probant. Nous avons la *Place Bellecour* qui est bien identifiée, ainsi que la *Place de la République* ou la *Place de Jacobins* par exemple. Cependant, nous remarquons que les points d'intérêt à l'Est et au Nord-Est de la carte sont toujours concentrés dans un seul cluster. Nous pouvons essayer de faire varier notre $minPts$ qui correspond au nombre minimum de points devant se trouver dans un rayon ϵ pour que ces points soient considérés comme un cluster.

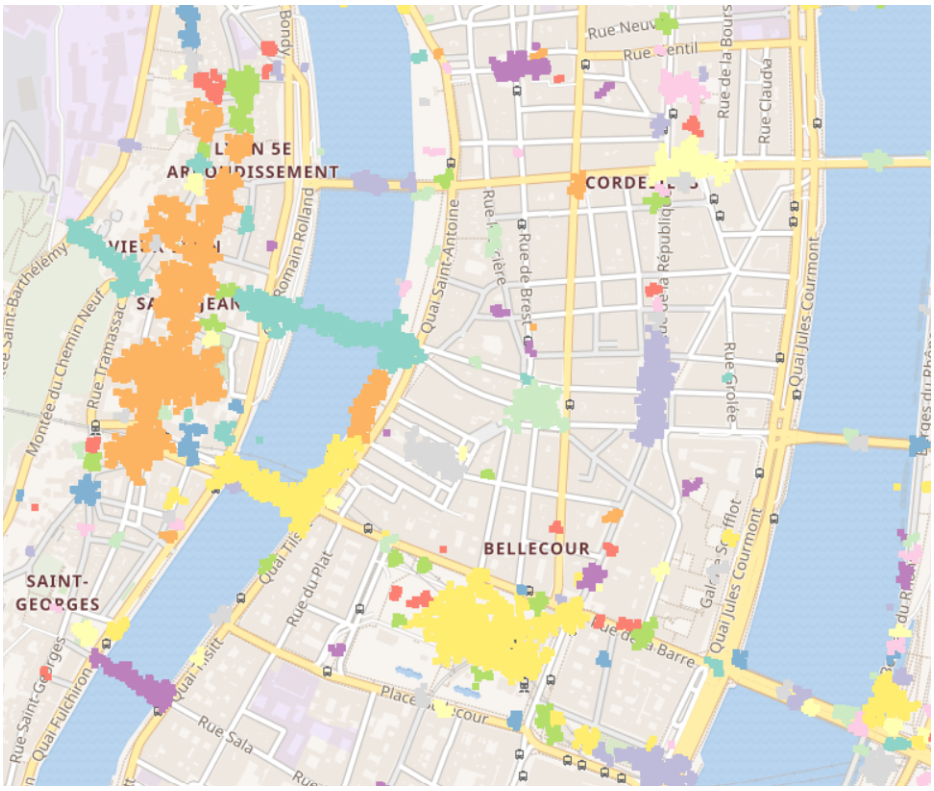


FIGURE 3.12 – Clustering avec DBSCAN ($\epsilon = 0.001$, $minPts = 10$)

$\epsilon = 0.001$, $minPts = 7$

Nous continuons avec une valeur de ϵ égale à 0.001 et une valeur de $minPts$ égale à 7. Nous obtenons le résultat présenté à la figure 3.13. Le résultat est assez similaire au précédent. Les mêmes gros clusters sont représentés. La seule différence réside dans le fait que moins de points ont été considérés comme du bruit. En effet, ayant réduit le nombre de points minimal, plus de clusters existent. Nous pouvons alors essayer d'augmenter $minPts$. Nous devrions éliminer encore plus de petits clusters.

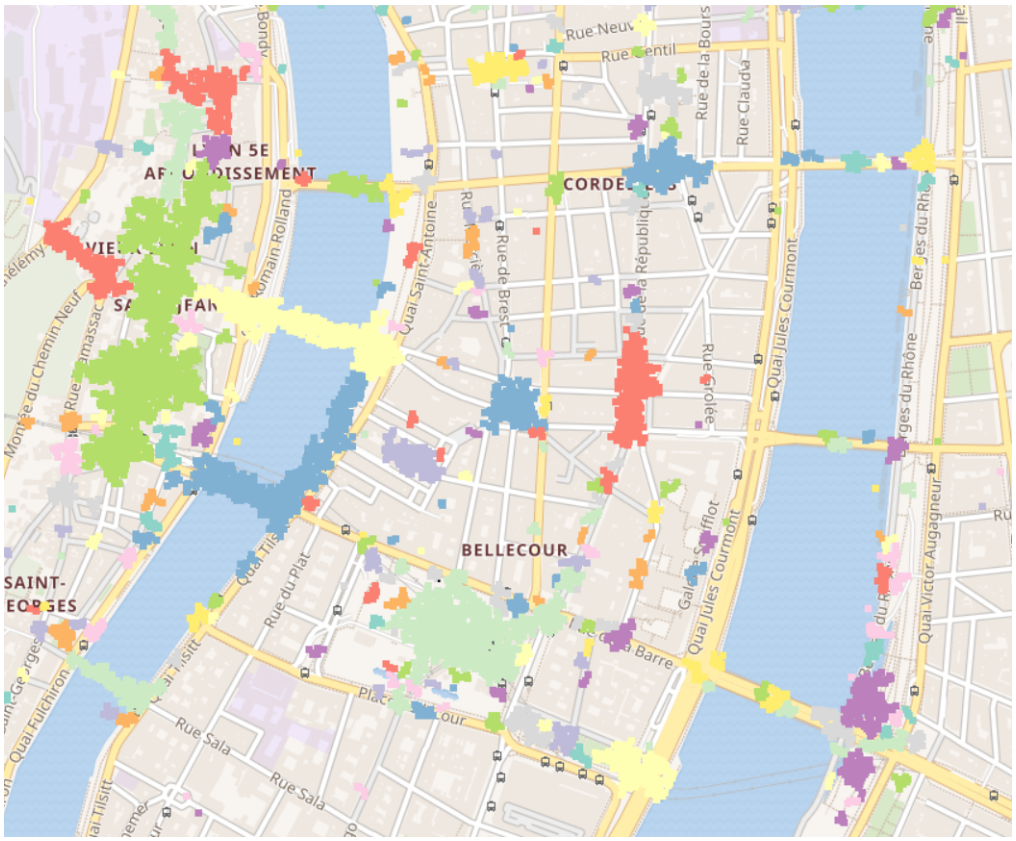


FIGURE 3.13 – Clustering avec DBSCAN ($\epsilon = 0.001$, $minPts = 7$)

$\epsilon = 0.001$, $minPts = 12$

3.4 *DBSCAN avec dimension temporelle*

La figure 3.14 représente la recherche de clusters en se basant sur la dimension géographique et la dimension temporelle. Nous pouvons voir que les points en vert représentent la *Fête des Lumières* qui a lieu à Lyon chaque année en décembre : en effet, les tags spécifient bien cet évènement, le mois de la prise des photos est bien décembre, et les jours se situent bien autour du 8 décembre, jour de la Fête des Lumières. Ces points sont présents dans une large partie de Lyon, la fête se déroulant dans la ville entière.

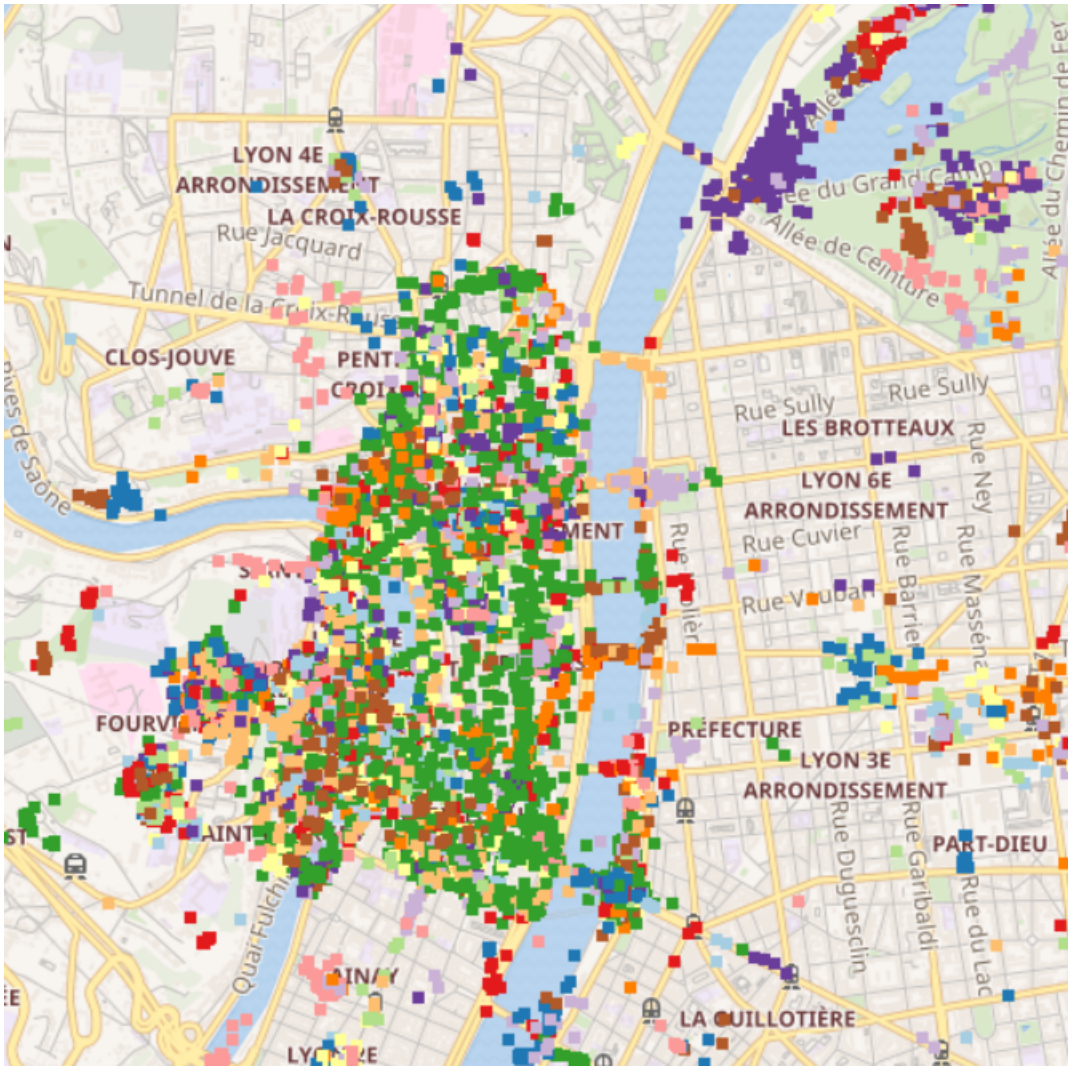


FIGURE 3.14 – DBSCAN avec dimension temporelle ($\epsilon = 0.005$, $minPts = 10$

Conclusion

Pour conclure, nous avons vu que l'algorithme *K-means* nous donnait des résultats assez satisfaisants lorsque la densité des clusters était à peu près égale. Cet algorithme est de surcroît rapide. Néanmoins, ce dernier n'est pas stable. Nous ne continuerons donc pas avec ce dernier pour la phase de fouille de motifs pour la compréhension des clusters. Le *clustering hiérarchique* semble donner des résultats intéressants mais sa complexité algorithmique rend son utilisation impossible sur notre trop grand jeu de données. Enfin, *DBSCAN* nous a donné des résultats vraiment satisfaisants. Il est performant et permet de filtrer des données pouvant être considérées comme du bruit. Un autre avantage est qu'il sait trouver des clusters avec des formes non globulaires. Nous continuerons donc la prochaine phase d'étude, à savoir la fouille de motifs pour la compréhension des clusters, avec l'algorithme DBSCAN ($\epsilon = 0.001$, $minPts = 10$).

Chapitre 4

Fouille de motifs pour la compréhension des clusters

Nous utilisons le tutoriel proposé par *KNIME* sur la fouille de texte pour extraire des règles d'association issues de nos données. Nous décidons de d'utiliser l'algorithme *DBSCAN* ($\epsilon = 0.001$, $minPts = 10$) comme déjà explicité. Notre phase de traitement pour construire la matrice binaire est la suivante :

1. remplacement des caractères "," par des caractères " " dans les tags afin de rendre les données homogènes ;
2. suppression des tags contenant les mots comme "insta", "nikon", "upload", "filter", "canon", "iphone" pour ne pas polluer nos résultats avec des informations inutiles ;
3. suppression des termes numériques qui selon nous ne caractériseront pas d'évènements, en général représentant un numéro de rue, de département ou d'année ;
4. suppression des caractères de ponctuation ;
5. filtrage des termes ne contenant pas plus de 3 caractères afin de supprimer les déterminants qui parasiteront nos résultats ;
6. conversion des termes en minuscules afin d'homogénéiser nos données ;
7. suppression des termes parasites avec le composant *Stop Word Filter* ;
8. conversion des termes en leurs termes radicaux correspondants avec le composant *Snowball Stemmer*.

Nous pouvons alors étudier dans un premier temps les règles d'association entre les différents tags, puis entre les tags et un cluster.

4.1 Ensembles de tags fréquents

Nous utilisons le composant *Item Set Finder* pour trouver des ensembles de tags fréquents. Nous utilisons l'algorithme *Apriori* avec un objectif *Closed*. Nous commençons avec un support minimum de 5%. Nous obtenons le résultat présenté à la figure 4.1. Nous avons déjà beaucoup d'informations qui ressortent de cette première analyse. Nous pouvons voir que l'association "France" - "Lyon" est très fréquente. Cette information n'est peut être pas la plus informative, en effet. Nous remarquons aussi l'ensemble de tags "UNICEF" - "ONG" - "Lyon". Cette association nous apprend qu'il existe l'ONG UNICEF à Lyon. Le tag "Fête des lumières" ressort aussi beaucoup. Celui-ci nous informe sur la connaissance de cet évènement. D'autres évènements comme "Biennale d'art contemporain de Lyon" sont mis en évidence. Certains lieux de Lyon ressortent également comme "Confluence", "Le Vieux Lyon", "Jean-Macé", "Bellecour". Des lieux de culture comme l'Opéra, le Cinéma et la Danse sont également soulignés.

Nous pouvons nous intéresser à des associations entre au moins deux items. Nous diminuons le support minimal jusqu'à 1% et nous sélectionnons quelques résultats pertinents que nous affichons à la figure 4.2

Row ID	[...] ItemSet	I ItemSetSize	I ItemSetSupport	D RelativeItemSetSupport%
Row85	[lyon]	1	8303	28.451
Row83	[franc]	1	4035	13.826
Row84	[franc,lyon]	2	2058	7.052
Row82	[ong]	1	1470	5.037
Row81	[unicefrhôn,ong]	2	1312	4.496
Row79	[lumi]	1	1310	4.489
Row73	[unicef]	1	910	3.118
Row75	[unicef,ong]	2	848	2.906
Row74	[unicef,unicefrhôn,ong]	3	700	2.399
Row71	[nuit]	1	675	2.313
Row69	[light]	1	648	2.22
Row80	[lumi,lyon]	2	622	2.131
Row65	[festival]	1	603	2.066
Row61	[fêtedeslumi]	1	554	1.898
Row55	[fêt]	1	520	1.782
Row53	[lion]	1	466	1.597
Row51	[night]	1	452	1.549
Row49	[rhônealp]	1	436	1.494
Row58	[fêt,lumi]	2	429	1.47
Row47	[rhôn]	1	424	1.453
Row54	[lion,franci]	2	318	1.09
Row42	[europ]	1	316	1.083
Row48	[rhôn,lyon]	2	299	1.024
Row37	[confluenc]	1	291	0.997
Row50	[rhônealp,franc]	2	281	0.963
Row35	[bellecour]	1	266	0.911
Row72	[nuit,unicefrhôn,ong]	3	254	0.87
Row34	[biennal]	1	253	0.867
Row32	[vieuxlyon]	1	244	0.836
Row30	[fontain]	1	241	0.826
Row64	[fêtedeslumi,lyon]	2	241	0.826
Row23	[street]	1	215	0.737
Row62	[fêtedeslumi,lumi]	2	215	0.737
Row21	[robot]	1	214	0.733
Row19	[tournag]	1	211	0.723
Row18	[courtmétrag,tournag]	2	209	0.716
Row63	[fêtedeslumi,lumi,lyon]	3	197	0.675
Row16	[oper]	1	192	0.658
Row17	[lyonfranc]	1	187	0.641
Row70	[light,lumi]	2	187	0.641
Row13	[cathédral]	1	184	0.63
Row15	[sido,lyon]	2	181	0.62
Row26	[del,lion]	2	177	0.607
Row27	[del,lion,franci]	3	173	0.593
Row14	[biennaledelyon]	1	171	0.586
Row60	[fêt,lyon]	2	170	0.583
Row67	[festival,lumi]	2	168	0.576
Row11	[danc]	1	167	0.572
Row10	[basil]	1	166	0.569
Row59	[fêt,lumi,lyon]	3	164	0.562
Row66	[festival,light]	2	164	0.562
Row9	[rhon]	1	162	0.555
Row5	[décembr]	1	158	0.541
Row7	[spectacl]	1	155	0.531
Row57	[fêt,de,lumi]	3	154	0.528
Row43	[europ,franc]	2	153	0.524
Row6	[cinem]	1	152	0.521
Row2	[jean]	1	147	0.504
Row1	[vieux]	1	146	0.5
Row3	[forum]	1	146	0.5

FIGURE 4.1 – Ensembles de motifs trouvés avec un support minimal de 5%

Row ID	ItemSet	ItemSetSize	ItemSetSupport	RelativeItemSetSupport%
Row203	[vidéo,courtmétrag,tournag]	3	138	0.473
Row198	[artcontemporain,biennaledelyon,biennal]	3	137	0.469
Row138	[gaypridelyon,prid,gay]	3	92	0.315
Row133	[bron,unicef,ong]	3	86	0.295
Row131	[locauxmotiv,guillotier]	2	85	0.291
Row179	[jacobin,lyon]	2	78	0.267
Row126	[partisocial,manifest,retrait]	3	77	0.264
Row102	[cattedral,chies,fest]	3	65	0.223
Row212	[spectacl,fetedeslumier,nuit]	3	61	0.209
Row70	[grottedelamadelein,klavierkunst,frankreich]	3	54	0.185
Row254	[biennal,lyon]	2	49	0.168
Row57	[assembl,unicefrhôn,ong]	3	48	0.164
Row54	[unesco,franc]	2	43	0.147
Row143	[beaux-art,lyon]	2	41	0.14
Row30	[mutualis,locauxmotiv,guillotier]	3	40	0.137
Row66	[patrickageneau,passag]	2	39	0.134
Row84	[pasteup,streetart,franc]	3	35	0.12
Row0	[fêtesdeslumi,lyon]	2	34	0.117
Row11	[jeremygranier,streetday,bmx]	3	34	0.117
Row16	[patrimoin,journ]	2	34	0.117
Row33	[skatepark,bergesdurhôn]	2	34	0.117
Row34	[skatepark,bergesdurhôn,bmx]	3	32	0.11
Row199	[art,biennaledelyon]	2	32	0.11
Row132	[locauxmotiv,guillotier,lyon]	3	31	0.106
Row219	[danc,oper,dans]	3	31	0.106
Row249	[fetedeslumier,franc]	2	31	0.106
Row2	[hotêl,patrimoin,journ]	3	30	0.103

FIGURE 4.2 – Ensembles de motifs trouvés avec un support minimal de 5%

De nouvelles informations s'en dégagent. Par exemple, nous pouvons noter la présence de la Gay Pride à Lyon, que le street art est beaucoup présent avec des skateparks, que l'art y est très développé, ou encore qu'il y a une journée du Patrimoine. Essayons maintenant d'associer ces tags avec les clusters découverts par *DBSCAN*.

4.2 Règles d'associations entre tags et clusters

Ici, nous allons chercher des règles d'association entre tags et clusters. Notre but est de déterminer si des clusters son caractérisables par des tags particuliers. Pour ce faire, nous utilisons le composant *Association Rule Learner*. Nous le paramétrons avec un support minimal de 2% par exemple. Le résultat est observable à la figure 4.3. Nous avons des résultats tout à fait pertinents. En effet, nous apprenons par exemple que :

- le cluster 147 représente une cathédrale ;
- le cluster 249 représente le "streetday" ;
- le cluster 511 représente l'évènement "walking zombie" ;
- le cluster 541 représente le musée des confluences ;
- le cluster 629 représente un évènement lié aux robots ;
- le cluster 636 représente l'UNICEF ;
- le cluster 97 représente l'architecture et l'histoire du *Vieux Lyon*.

Tous les clusters n'ont pas pu être caractérisé avec un support de 2%.

Row ID	Consequent	Antecedent	ItemS...	RelativeIt...	RuleC...	Absol...	Relativ...	RuleLift	RuleLif...	Absol...	Relativ..
Row527	147	[cattedral,chies,fest]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row530	147	[cattedral,chies]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row532	147	[cattedral,fest]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row615	147	[chies,fest]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row184	249	[jeremygranier,streetday,bmx]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row187	249	[jeremygranier,streetday]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row194	249	[jeremygranier,bmx]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row196	249	[jeremygranier]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row683	249	[streetday,bmx]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row454	511	[walking,walk,zomb]	1	0.124	100	1	0.124	402	40,200	2	0.249
Row457	511	[walking,walk]	1	0.124	100	1	0.124	402	40,200	2	0.249
Row460	511	[walking,zomb]	1	0.124	100	1	0.124	402	40,200	2	0.249
Row2	541	[muséedesconfluent]	1	0.124	100	1	0.124	89.333	8,933.3	9	1.119
Row130	541	[muždesconfluent,patrickageneau,passag]	1	0.124	100	1	0.124	89.333	8,933.3	9	1.119
Row135	541	[muždesconfluent,patrickageneau]	1	0.124	100	1	0.124	89.333	8,933.3	9	1.119
Row138	541	[muždesconfluent,passag]	1	0.124	100	1	0.124	89.333	8,933.3	9	1.119
Row141	541	[muždesconfluent]	1	0.124	100	1	0.124	89.333	8,933.3	9	1.119
Row22	629	[lexpo,sido]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row25	629	[lexpo]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row498	629	[conférent,sido]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row499	629	[conférent]	2	0.249	100	2	0.249	73.091	7,309.1	11	1.368
Row616	629	[syrobo,robolift,robot]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row619	629	[syrobo,robolift]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row620	629	[syrobo,robot]	2	0.249	100	2	0.249	73.091	7,309.1	11	1.368
Row623	629	[syrobo]	2	0.249	100	2	0.249	73.091	7,309.1	11	1.368
Row653	629	[innorobo,robolift,robot]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row656	629	[innorobo,robolift]	1	0.124	100	1	0.124	73.091	7,309.1	11	1.368
Row657	629	[innorobo,robot]	2	0.249	100	2	0.249	73.091	7,309.1	11	1.368
Row660	629	[innorobo]	2	0.249	100	2	0.249	73.091	7,309.1	11	1.368
Row735	629	[sido]	3	0.373	100	3	0.373	73.091	7,309.1	11	1.368
Row770	629	[robolift,robot]	3	0.373	100	3	0.373	73.091	7,309.1	11	1.368
Row773	629	[robolift]	3	0.373	100	3	0.373	73.091	7,309.1	11	1.368
Row806	629	[robot]	6	0.746	85.7	7	0.871	62.649	6,264.9	11	1.368
Row26	636	[diver,unicef]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row98	636	[liberty,unicefrhôn,ong]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row103	636	[liberty,unicefrhôn]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row106	636	[liberty,ong]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row109	636	[liberty]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row466	636	[bron,unicef,ong]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row471	636	[bron,unicef]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row473	636	[bron,ong]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row559	636	[assembl,unicefrhôn,ong]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row564	636	[assembl,unicefrhôn]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row566	636	[assembl,ong]	1	0.124	100	1	0.124	47.294	4,729.4	17	2.114
Row199	97	[architequtur,histor,vieuxlyon]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row202	97	[architequtur,histor]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row210	97	[architequtur,vieuxlyon]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row211	97	[architequtur]	1	0.124	100	1	0.124	268	26,800	3	0.373
Row586	97	[histor,vieuxlyon]	1	0.124	100	1	0.124	268	26,800	3	0.373

FIGURE 4.3 – Règles d'association clusters - tags

Cherchons maintenant à identifier les évènements à Lyon.

Chapitre 5

Recherche d'évènements

5.1 Identification d'évènements dans le temps

Pour commencer, on recherche des clusters en prenant en compte uniquement la date (année, mois, jour) de prise des photos. Pour ce faire, on utilise à nouveau l'algorithme de clustering DBSCAN et la fouille de motifs vus précédemment.

Nous pouvons donc identifier des clusters qui correspondent à des points qui ont une certaine proximité dans le temps, et que nous pouvons donc interpréter comment appartenant au même évènement.

Nous avons alors découvert les évènements suivants :

Nom de l'évènement identifié	Tags	Date
Festival Lyon Tattoo	conventionoftattooslyon	2-3 février 2013
Japan Touch	japan, japantouch	3-4 décembre 2011 13 avril 2013
LDOLL Festival	artdoll, ldoll, dollcon	5-6 octobre 2013 18 octobre 2014
Ultra Boucle de la Sarra	ultraboucledelasarra	16 mai 2015
Zombie Walk Lyon	zombies, blood	13 octobre 2012
24h de l'INSA	insa, 24h	21-23 mai 2010
Nuits de Fourvière	nuitsdefourviere, concert, fourviere	15 juillet 2014

On identifie ainsi plusieurs évènements, correspondant aux clusters trouvés, qui sont pour la plupart des évènements annuels se déroulant à Lyon, comme les Nuits de Fourvière et les 24h de l'INSA pour ne citer que les plus courants d'entre eux.

On observe tout de même sur KNIME des tags provenant d'utilisateurs individuels qui ont pris plusieurs photos pendant une courte période de temps et qui sont donc assimilés par le logiciel à un cluster à eux tout seuls. Ces utilisateurs ne représentent pas d'intérêt pour notre recherche d'évènements.

5.2 Identification d'évènements dans le temps et l'espace

À présent, nous exploitons la date et la position dans notre recherche de clusters.

Nous obtenons les clusters qui suivent :

Nom de l'évènement identifié	Tags	Date	Lieu
Festival Lyon Tattoo	conventionoftattooslyon	2-3 février 2013	Université Lyon 1
Japan Touch	japan, japantouch	3-4 décembre 2011 13 avril 2013	Double Mixte
LDOLL Festival	artdoll, ldoll, dollcon	5-6 octobre 2013 18 octobre 2014	Double Mixte
Ultra Boucle de la Sarra	ultraboucledelasarra	16 mai 2015	Piste de la Sarra
24h de l'INSA	insa, 24h	21-23 mai 2010	INSA Lyon
Nuits de Fourvière	nuitsdefourviere, concert, fourviere	15 juillet 2014	Théâtres romains de Fourvière

Ce que nous avons obtenu l'avait déjà été avec l'identification d'évènements dans le temps uniquement.

Chapitre 6

Conclusion

Pour conclure, nous affirmons qu'il est possible de réaliser de la découverte de centres d'intérêts et d'évènements au moyen du workflow KNIME sur les données FLICKR. En effet, nous avons vu qu'après une phase de compréhension et de nettoyage des données, nous avons pu mettre en place des algorithmes de clustering pour découvrir des groupes au sein de nos données. Nous avons vu que les différents algorithmes pouvaient avoir des défauts comme notamment *K-means* qui a des difficultés lorsque les groupes ont des densités différentes ou que les groupes ont des formes non globulaires ou encore le *clustering hiérarchique* qui a du mal a passer à l'échelle.

Nous avons vu que la phase de compréhension et de nettoyage des données était fastidieuse mais néanmoins nécessaire et primordiale pour une meilleure analyse. La difficulté à trouver les bons paramètres pour les différents algorithmes est également un sujet de poids. La phase de fouille de motifs fréquents pour la compréhension des clusters est également une phase importante. En effet, elle permet d'appuyer l'Homme dans le contrôle des résultats fournis par les différents algorithmes.

Certaines limites au projet peuvent être liés à la véracité des données. Par exemple les tags représentent ils bien les photos en questions ? Ou encore les latitudes et longitudes sont elles correctes ? De plus, ici, nous considérons chaque photo avec un poids égal. Si un utilisateur prend énormément de photos d'un point en question par exemple il pourrait créer à sa guise un nouveau cluster. Il y a donc un soucis de robustesse. Nous pensons tout de même que dans l'ensemble le projet est concluant.