



Projet 02 : Participez à un concours sur la Smart City

Mohamed A.

Objectif & impact

A travers l'inscription à un challenge public avec l'ONG “**Data is for good**” et qui s'intitule “**Aidez Paris à devenir une smart-city**”.

Dans ce projet, vous devez réaliser une analyse exploratoire en utilisant Python. Attention, cette tâche peut prendre beaucoup de temps ! Soyez vigilant sur le temps passé sur ce projet, en gardant un œil sur la durée estimée du projet, et les compétences évaluées.

Vos résultats contribuerons à une optimisation des tournées pour l'entretiens des arbres de la ville. Eh oui, moins de tournées égale moins de trajets, et plus d'arbres entretenus.

Vous aurez ainsi un impact réel sur le future de la ville de Paris!

Travail à faire

Réaliser une analyse exploratoire avec un jeu de données portant sur les arbres de la ville de Paris, dans cadre du programme “**Végétalisons la ville**”.

dessous les diverses phases du projet et le format final demande :

- 1
- l'environnement de développement est installé et fonctionnel sur le poste de travail (Python et Jupyter)
 - un environnement virtuel a été créé pour assurer l'isolement du projet et la gestion des dépendances
 - les librairies python spécialisées ont été importées dans le Jupyter Notebook

- 2
- le jeu de données a été décrit brièvement (nombre de lignes, nombre de colonnes et nombre de valeurs manquantes) en ayant chargé le fichier plat dans un dataframe.

- 3
- L'analyse de données est **présentable** si :
- les fonctionnalités d'édition de cellule Markdown du Jupyter Notebook sont utilisées dans au moins trois cellules pour commenter l'analyse et la mettre en forme (titres, mise en forme, alternance de cellule d'exécution de code python et de cellule de texte explicatif)
 - les titres des trois parties (la présentation générale du jeu de données, la démarche méthodologique d'analyse de données, la synthèse de votre analyse de données) sont visuellement en évidence dans le Jupyter Notebook (par exemple en gras et police de caractère plus grande que dans le reste du document)



L'analyse statistique univariée est **complète** si :

- les moyennes, médianes et quantiles des distributions sont calculées
- au moins une représentation graphique d'une distribution statistique a été tracée

L'analyse statistique univariée est **pertinente** si :

- les éventuelles valeurs aberrantes ont été identifiées (avec une définition des valeurs aberrantes basée sur un multiple des quantiles de la distribution).

L'analyse statistique univariée est **présentable** si :

- les graphiques sont lisibles (titres alignés, légende présente, noms des abscisses et des ordonnées précisés)

Le Management de l'arbre urbain

Why Is Urban Forest Management Important to Public Works?

Investment in community's future
Public safety and municipal liability
Efficient operations
Improve the environment



Urban Forest Management Plans

All communities manage urban trees



- Levels
 - Young
 - Growing
 - Mature

Tree Planting Plan

- Planting locations identified from inventory data
- Species options
- Maintenance plans for newly establishing trees
- Technical information on proper tree planting techniques



Tree Maintenance Plan

- Tree Maintenance
 - Removal and pruning prioritized
 - Stump grinding
 - Fertilization
 - Insect and disease treatment
 - Grate and guard repair
 - Mulching
 - Watering



Tree Inventories

- "Windshield"
- Partial
- Complete

Software programs to manage inventory data

- Variety of programs available
- Used to:
 - Create work orders
 - Track citizen calls
 - Generate reports
 - Make maps



Urban Forest Cost/Benefit Analysis

- Valuable municipal resources
- Justify funding
- Build Public Support
- Quantify the benefits of the urban forest
 - Energy reduction
 - Stormwater management
 - Property values
 - Air Quality
 - i-tree suite



ercu sur les outils utilisés pour la collecte de données :

Typology	Technical aid	Description	References	No. of articles
Satellite-supported methods	QuickBird, Panchromatic and multispectral images. Very High Resolution images (VHR)	QuickBird is a high-resolution earth observation satellite. Panchromatic produces a realistic picture as it appears to the human eye. VHR images are taken by satellites (e.g., urban landscapes).	(Ardila et al. 2012)	3
	Google Maps and Google Maps TM API	Google Maps TM API (API = Application programming interface) (Google). Allows users to create thematic maps.	(Thornhill et al. 2009)	1
Airplane-supported methods	Airborne LIDAR data collection and Terrestrial Laser Scanning (TLS) Airborne Laser Terrain Mapping (ALTM)	LIDAR (Light Detection And Ranging, also LADAR) is an optical remote sensing technology that can measure the distance to, or other properties of, targets the use of laser. TLS works in a similar way by registering data using lasers. ALTM maps the surface and thereby acquires maps equivalent to those of GPS.	(Jutras et al. 2009)	1
	Aerial photos	Photos taken from aircrafts from different heights.	(Miller and Winer 1984)	1
Ground scanning/photos	Customer grade cameras or digital photos	Photos taken by professional or digital cameras.	(Buhyoff et al. 1984; Abd-Elrahman et al. 2010; Patterson et al. 2011)	2
	Mobile Laser Scanning (MLS)	MLS is a technology in which objects are mapped by laser distance measurement from driving vehicles (e.g., road vehicles, ships, or railway trains). The data are then transformed into a 3D point cloud using GPS/IMU data.	(Rutzinger et al. 2011)	1
	Mobile Augmented Reality (AR)	AR is a way of viewing digital information which has been superimposed or augmented onto a live view of the real-world environment.	(West et al. 2012)	1
	Terrestrial Laser Scanning (TLS)	TLS analyses an object or environment to collect data on its shape. The collected data can then be used to construct digital, two-dimensional drawings or three-dimensional models.	(Park et al. 2010)	1
Field survey	Urban vegetation surveys (e.g., GPS receiver or handheld computers)	Field staff conduct direct measurements and visual inspections of individual trees, where position data can be supported by a GPS navigation device and data reporting supported by a handheld computer.	(Starr 1990; Lesser 1996; Sudol and Zach 1987; Adkins et al. 1997; Hsu 1997; Jim and Liu 1997; Poracskey and Scott 1999; Martin et al. 2011)	46
	Windshield method	Field staff conducts visual inspection of trees from a car driven at low speed (approx. 3 km/hr).	(Rooney et al. 2005).	0

96

Nielsen et al.: Review of Urban Tree Inventory Methods Used to Collect Data at Single-Tree Level



Arboriculture & Urban Forestry 2014, 40(2): 96–111

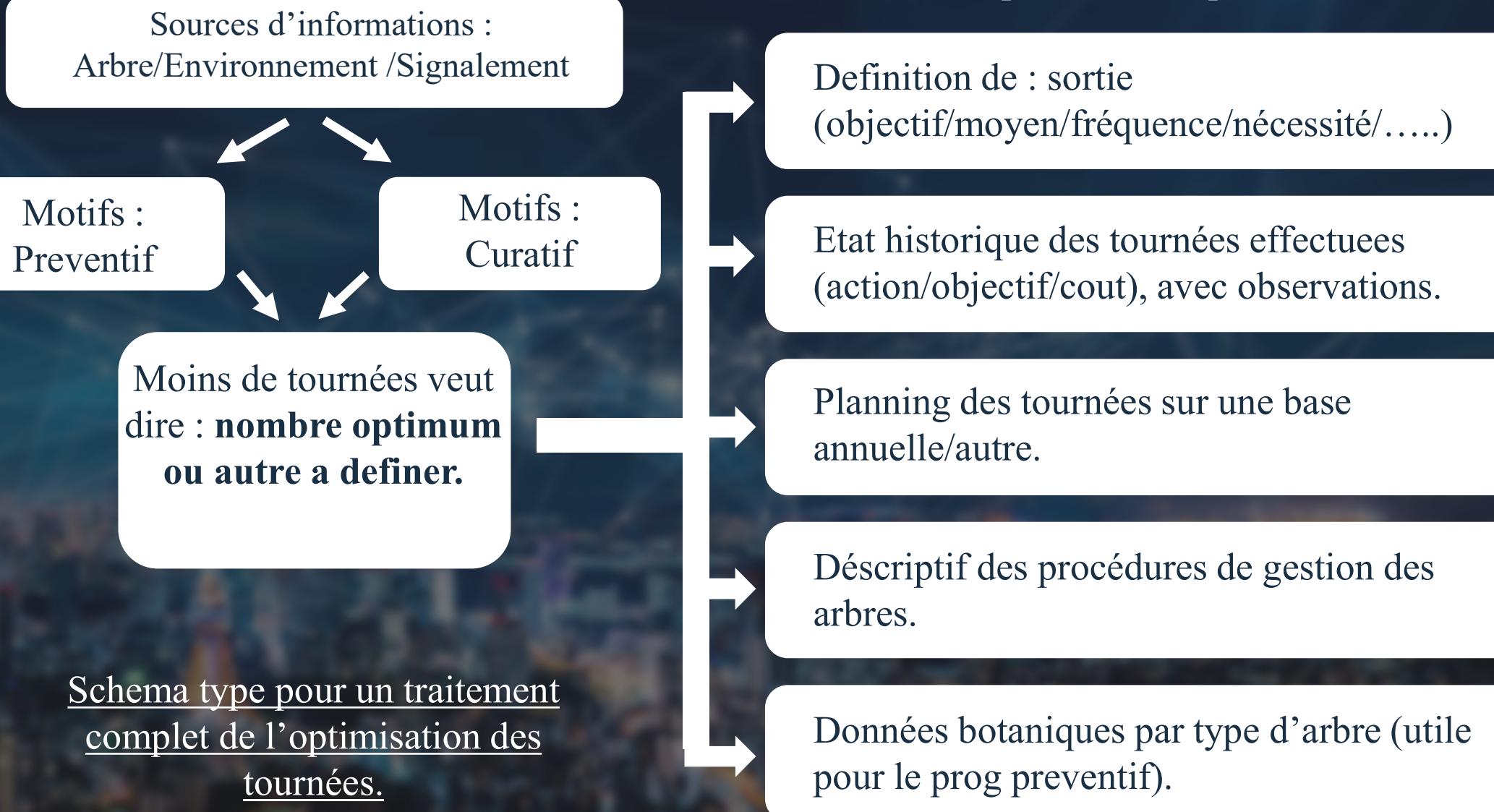
ARBORICULTURE
URBAN FOREST
Scientific Journal of the International Society of Arboriculture

Review of Urban Tree Inventory Methods Used to Collect Data at Single-Tree Level

Anders B. Nielsen, Johan Östberg, and Tim Delshammar

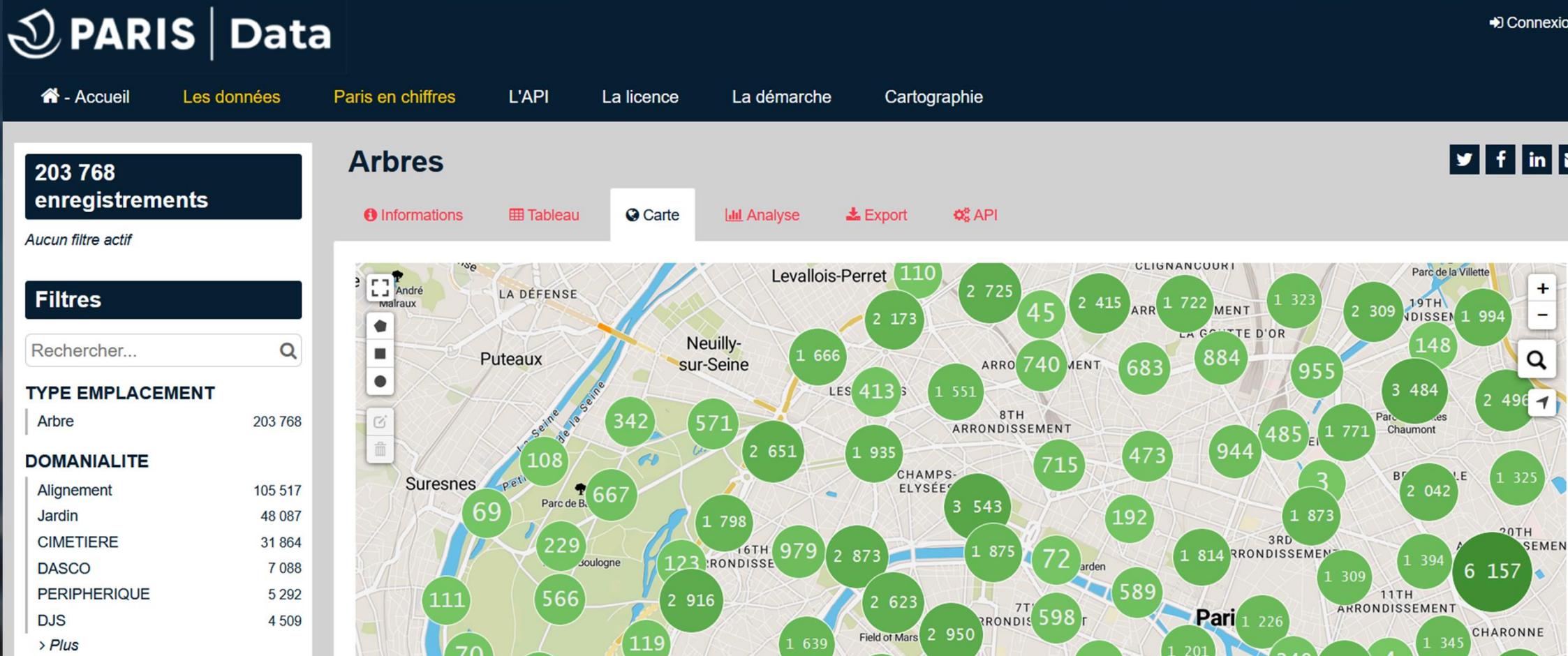
L'article suivant dresse un inventaire des principales techniques utilisées dans la collecte de données pour la gestion du parc d'arbres urbains.

Approche plus realiste du traitement du problème d'optimisation



Presentation du dataset utilisé:

Le dataset a été téléchargé a partir du site : <https://opendata.paris.fr>



Le dataset téléchargé :

Nom : p2-arbres-fr.csv

Format : .csv

Taille : 29 Mo

ncement du traitement du dataset :

L'analyse se fera avec le logiciel python sur l'IDE : Jupyter Notebook.

Les étapes principales de l'analyse sont :

- Chargement des modules python : pandas/numpy/matplotlib/seaborn.
- Upload du dataset et identification de ces caractéristiques principales.
- Cleaning du dataset / identification et traitement des valeurs manquantes, nulls et aberrantes.
- Analyse du dataset corrigé et développement de variables d'optimization.
- Synthèse.

Quelques résultats tirés du notebook :

Chargement des librairies et du Dataset

```
In [1]: # Importer Les Librairies python nécessaire pour l'exploitation du Dataset  
# prealablelement installés dans l'environnement Anaconda utilisé sur mon PC  
  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns
```

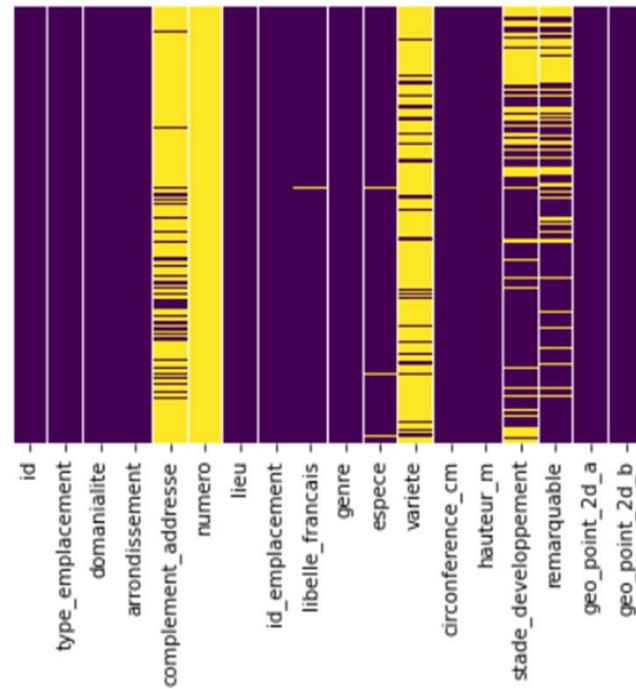
Aperçu :

	<u>id</u>	<u>type_emplacement</u>	<u>domanialite</u>	<u>arrondissement</u>	<u>complement_adresse</u>	<u>numero</u>	<u>lieu</u>	<u>id_emplacement</u>	<u>libelle_francais</u>	<u>genre</u>	<u>es</u>	
0	99874	Arbre	Jardin	PARIS 7E ARRDT		NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	19	Marronnier	Aesculus	hippocast
1	99875	Arbre	Jardin	PARIS 7E ARRDT		NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	20	If	Taxus	ba
2	99876	Arbre	Jardin	PARIS 7E ARRDT		NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	21	If	Taxus	ba
3	99877	Arbre	Jardin	PARIS 7E ARRDT		NaN	NaN	MAIRIE DU 7E 116 RUE DE GRENELLE PARIS 7E	22	Erable	Acer	neç
4	99878	Arbre	Jardin	PARIS 17E ARRDT		NaN	NaN	PARC CLICHY- BATIGNOLLES- MARTIN LUTHER KING	000G0037	Arbre à miel	Tetradium	da

Caractéristiques du dataset et identification des valeurs manquantes :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200137 entries, 0 to 200136
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   id               200137 non-null   int64  
 1   type_emplacement 200137 non-null   object  
 2   domanialite       200136 non-null   object  
 3   arrondissement    200137 non-null   object  
 4   complement_adresse 30902 non-null   object  
 5   numero            0 non-null      float64 
 6   lieu               200137 non-null   object  
 7   id_emplacement    200137 non-null   object  
 8   libelle_francais  198640 non-null   object  
 9   genre              200121 non-null   object  
 10  espece             198385 non-null   object  
 11  variete            36777 non-null   object  
 12  circonference_cm   200137 non-null   int64  
 13  hauteur_m          200137 non-null   int64  
 14  stade_developpement 132932 non-null   object  
 15  remarquable        137039 non-null   float64 
 16  geo_point_2d_a     200137 non-null   float64 
 17  geo_point_2d_b     200137 non-null   float64 
dtypes: float64(4), int64(3), object(11)
memory usage: 27.5+ MB
```

```
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis')
plt.show()
```



```
df.describe()
```

	id	numero	circonference_cm	hauteur_m	remarquable	geo_point_2d_a	geo_point_2d_b
count	2.001370e+05	0.0	200137.000000	200137.000000	137039.000000	200137.000000	200137.000000
mean	3.872027e+05	NaN	83.380479	13.110509	0.001343	48.854491	2.348208
std	5.456032e+05	NaN	673.190213	1971.217387	0.036618	0.030234	0.051220
min	9.987400e+04	NaN	0.000000	0.000000	0.000000	48.742290	2.210241
25%	1.559270e+05	NaN	30.000000	5.000000	0.000000	48.835021	2.307530
50%	2.210780e+05	NaN	70.000000	8.000000	0.000000	48.854162	2.351095
75%	2.741020e+05	NaN	115.000000	12.000000	0.000000	48.876447	2.386838
max	2.024745e+06	NaN	250255.000000	881818.000000	1.000000	48.911485	2.469759

Valeur aberrante

Chargement des modules python : pandas/numpy/matplotlib/seaborn

Identification des doublons :

```
In [12]: # Vérification et élimination des lignes en double  
df.duplicated().sum()
```

```
Out[12]: 0
```

Identification des hauteur nulles :

```
In [23]: hauteur_nulle = df[df['hauteur_m']==0]  
hauteur_nulle.shape
```

```
Out[23]: (39219, 17)
```

Identification des circonferences nulles :

```
In [26]: circonference_nulle = df[df['circonference_cm']==0]  
circonference_nulle.shape
```

```
Out[26]: (25867, 17)
```

Identification des valeurs manquantes :

```
df.isnull().sum()
```

type_emplacement	0
domanialite	1
arrondissement	0
complement_adresse	169235
numero	200137
lieu	0
id_emplacement	0
libelle_francais	1497
genre	16
espece	1752
variete	163360
circonference_cm	0
hauteur_m	0
stade_developpement	67205
remarquable	63098
geo_point_2d_a	0
geo_point_2d_b	0
dtype:	int64

Remplacement des valeurs nulles par la valeur médiane :

Hauteur :

```
[32]: # Remplacement des valeurs 'hauteur' nulle par la mediane =8,  
  
def value_replacement (df, hauteur_m):  
    for i, row in df.iterrows():  
        val = row['hauteur_m']  
        if val == 0:  
            df.at[i,'hauteur_m']=8  
        else:  
            df.at[i,'hauteur_m']=df.at[i,'hauteur_m']  
value_replacement(df, 'hauteur_m')
```

Circonference :

```
In [35]: # Remplacement des valeurs 'circonference' nulle par la mediane =70  
  
In [36]: def value_replacement_2 (df, circonference_cm):  
    for i, row in df.iterrows():  
        val = row['circonference_cm']  
        if val == 0:  
            df.at[i,'circonference_cm']=70  
        else:  
            df.at[i,'circonference_cm']=df.at[i,'circonference_cm']  
value_replacement_2(df, 'circonference_cm')
```

Données stat actualisées :

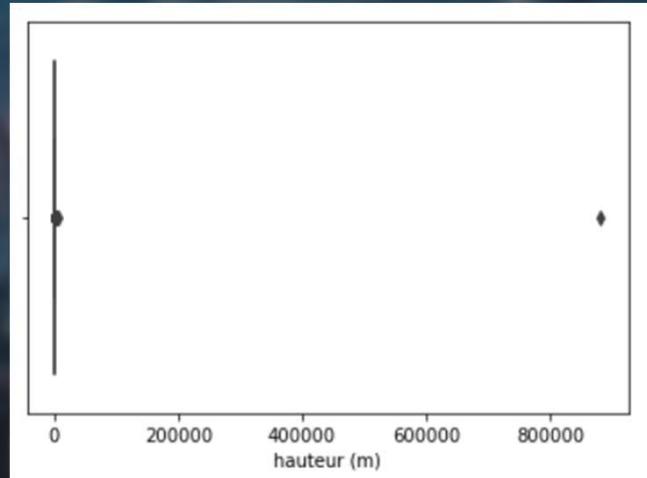
Out[39]:

	numero	circonference_cm	hauteur_m	remarquable	geo_point_2d_a	geo_point_2d_b
count	0.0	200137.000000	200137.000000	137039.000000	200137.000000	200137.000000
mean	NaN	92.427732	14.678195	0.001343	48.854491	2.348208
std	NaN	672.478836	1971.209518	0.036618	0.030234	0.051220
min	NaN	1.000000	1.000000	0.000000	48.742290	2.210241
25%	NaN	50.000000	7.000000	0.000000	48.835021	2.307530
50%	NaN	70.000000	8.000000	0.000000	48.854162	2.351095
75%	NaN	115.000000	12.000000	0.000000	48.876447	2.386838
max	NaN	250255.000000	881818.000000	1.000000	48.911485	2.469759

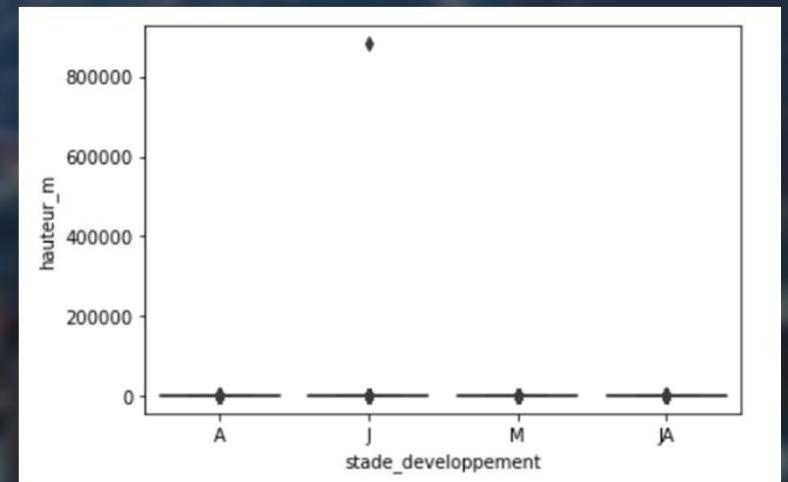
Identification et traitement des valeurs aberrantes :

```
In [40]: # On va se focaliser sur 3 parametres qui nous semble important a l'analyse des data
# 1- Hauteur / 2- Circonference / 3- Position geographique
# On remarque dans le tableau recapitulatif ci dessous les elements suivants :
# Hauteur max = 881 818 metres , Circonference max = 250 255 cm
# neanmoins ces valeurs ne semblent pas vraimeent affecter la moyenne en hauteur et circonference qui restent des valeurs plausibil
# On va essayer d'identifier les valeurs aberrantes de 2 manieres differentes :
### - Par les boxplot
### - Par des valeurs limites fournies a partir de donnees reeles.
```

Hauteur



Boxplot global



Visualisation des boxplot par type d'age

uteur :

plus grand : le Séquoia des Buttes-Chaumont

circonférence de 4,70 m et d'une hauteur de plus de 35 mètres, cet arbre originaire de Californie, situé dans le Parc des Buttes-Chaumont, est l'un des plus grands de la capitale. Vieux de plus de 150 ans, c'est encore un bébé, les séquoia pouvant vivre 3000 ans ! Ne manquez pas de toucher son écorce spongieuse, très douce au contact, qui lui offre notamment une remarquable résistance au feu.

```
n]: # Remplacement des hauteurs aberrantes / On va utiliser les données d'un site  
# 'https://www.unjourdeplusaparis.com/paris-vert/arbres-remarquables-paris'  
# le Séquoia des Buttes-Chaumont : D'une circonférence de 4,70 m et d'une hauteur de plus de 35 mètres.  
# On va utiliser ces 35 mètres comme limite et recommander à ce que ces valeurs soient actualisées.
```

```
hauteur_aberrante = df[df['hauteur_m']>35]  
hauteur_aberrante.shape  
# On constate que le nombre des valeurs aberrantes n'est pas important.  
509, 17)
```

```
df['hauteur_m'].max()  
35
```

```
n [48]: # Remplacement des valeurs aberrantes par la valeur max = 35 m  
def hauteur_replacement_abb(df, hauteur_m):  
    for i, row in df.iterrows():  
        val = row['hauteur_m']  
        if val > 35:  
            df.at[i, 'hauteur_m']=35  
        else:  
            df.at[i, 'hauteur_m']=df.at[i, 'hauteur_m']  
hauteur_replacement_abb(df, 'hauteur_m')
```



Circonference :

Le plus gros : platane d'Orient du Parc Monceau

Difficile de louper cet arbre du [parc Monceau](#), remarquable pour son âge et pour la base imposante de son tronc. Planté en 1814, son tronc mesure en effet 7 mètres de circonference pour une hauteur de 31 mètres environ !



```
In [55]: circonf_aberrante = df[df['circonference_cm']>700]  
circonf_aberrante.shape
```

```
Out[55]: (82, 17)
```

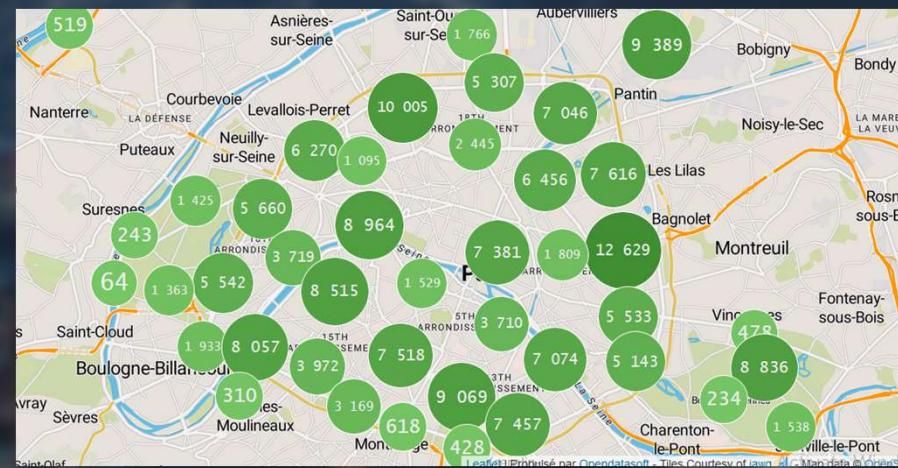
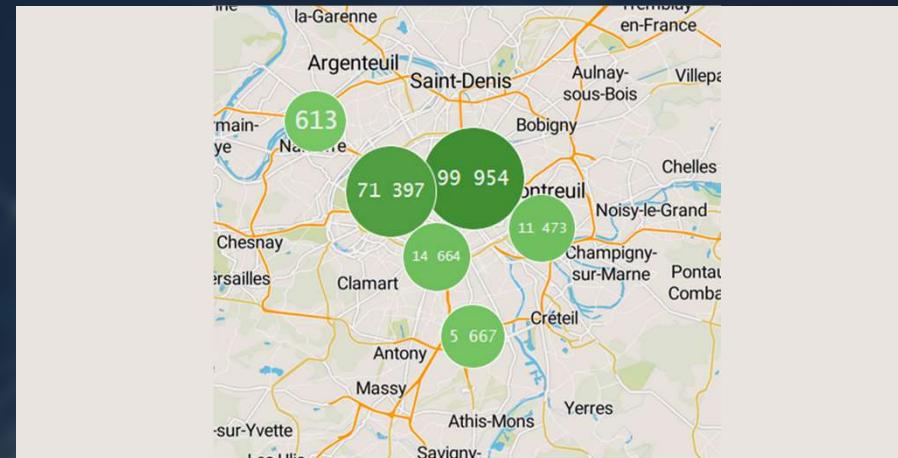
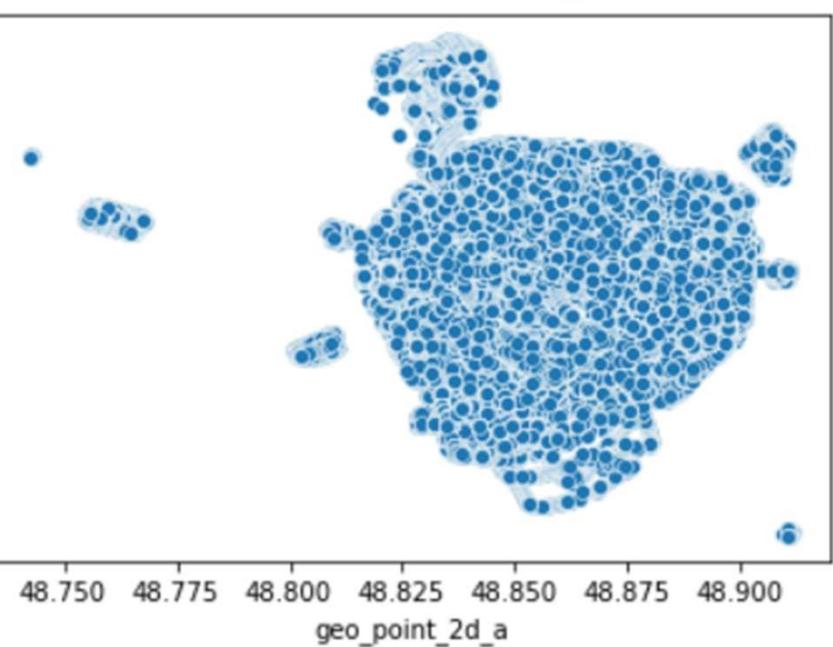
```
In [56]: def circonf_replacement_abb (df, circonference_cm):  
    for i, row in df.iterrows():  
        val = row['circonference_cm']  
        if val > 700:  
            df.at[i,'circonference_cm']=700  
        else:  
            df.at[i,'circonference_cm']=df.at[i,'circonference_cm']  
circonf_replacement_abb(df, 'circonference_cm')
```

```
In [57]: df['circonference_cm'].max()
```

```
Out[57]: 700
```

Coordonnées géographiques

```
60]: sns.scatterplot(df.geo_point_2d_a, df.geo_point_2d_b)
plt.show()
```



Cartes importées du site / Opendata Paris

Analyse du dataset et approche d'optimisation :

```
In [66]: # l'optimisation qu'on propose sera axee sur le fait de rendre les distances plus courtes par rapport a un niveau de  
# difficulte base sur (le nombre d'arbres par arrondissement, la hauteur , la circonference et la surface de l'arrondissement )  
# l'arrondissement avec le plus de point se verra attribue le plus de locaux d'interventions que ce soit pour les operations  
# de maintenance ou bien de collecte de donnees.
```

Out[75]:

	arrondissement	count_arbres	Hauteur_mean	circonference_mean
0	BOIS DE BOULOGNE	3978	9.594017	78.464806
1	BOIS DE VINCENNES	11510	10.784796	90.111121
2	HAUTS-DE-SEINE	5298	8.169309	70.992827
3	PARIS 10E ARRDT	3385	10.881536	87.820384
4	PARIS 11E ARRDT	5658	10.476670	82.925239
5	PARIS 12E ARRDT	12600	9.255476	85.332619
6	PARIS 13E ARRDT	16696	9.437769	77.766395
7	PARIS 14E ARRDT	11399	9.302307	97.747259
8	PARIS 15E ARRDT	17151	8.967874	82.366218
9	PARIS 16E ARRDT	16403	10.728708	94.500762
10	PARIS 17E ARRDT	10762	9.335997	86.701728
11	PARIS 18E ARRDT	10011	9.497153	82.194786
12	PARIS 19E ARRDT	13709	10.545846	94.471150
13	PARIS 1ER ARRDT	1413	8.909413	83.118188
14	PARIS 20E ARRDT	15340	9.830704	91.747523
15	PARIS 2E ARRDT	548	9.578467	78.762774
16	PARIS 3E ARRDT	1209	9.918941	82.564103
17	PARIS 4E ARRDT	2740	10.486496	90.928832
18	PARIS 5E ARRDT	2368	11.425676	96.956081
19	PARIS 6E ARRDT	1764	11.596939	90.792517
20	PARIS 7E ARRDT	8617	11.633399	95.964953
21	PARIS 8E ARRDT	7245	11.623741	105.267909
22	PARIS 9E ARRDT	1167	10.423308	82.156812
23	SEINE-SAINT-DENIS	11570	8.867416	88.493258
24	VAL-DE-MARNE	7580	11.189446	103.697098

Importation de nouvelles données pour l'optimisation

```
# Etablissement d'une regle de ponderation pour calculer le coefficient de contrainte das chaque arrondissement  
# La formulation s'ecrit comme suit :  
# count_arbres_updated = (count_arbres + ((hauteur_mean / hauteur_max)*count_arbres +  
# ((circonference_mean / circonference_max)*count_arbres)) / 3.
```

La même règle est appliquée pour la circonference

```
# Etablissement d'une regle de ponderation pour calculer le coefficient de contrainte das chaque arrondissement  
# La formulation s'ecrit comme suit :  
# count_arbres_updated = count_arbres + (hauteur_mean / hauteur_max)*count_arbres +  
# (circonference_mean / circonference_max)*count_arbres.
```

	arrondissement	count_arbres	Hauteur_mean	circonference_mean	coeff_hauteur	coeff_circonference
0	BOIS DE BOULOGNE	3978	9.594017	78.464806	0.274115	0.1
1	BOIS DE VINCENNES	11510	10.784796	90.111121	0.308137	0.1
2	HAUTS-DE-SEINE	5298	8.169309	70.992827	0.233409	0.1
3	PARIS 10E ARRD	3385	10.881536	87.820384	0.310901	0.1
4	PARIS 11E ARRD	5658	10.476670	82.925239	0.299333	0.1
5	PARIS 12E ARRD	12600	9.255476	85.332619	0.264442	0.1
6	PARIS 13E ARRD	16696	9.437769	77.766395	0.269651	0.1
7	PARIS 14E ARRD	11399	9.302307	97.747259	0.265780	0.1
8	PARIS 15E ARRD	17151	8.967874	82.366218	0.256225	0.1
9	PARIS 16E ARRD	16403	10.728708	94.500762	0.306535	0.1
10	PARIS 17E ARRD	10762	9.335997	86.701728	0.266743	0.1
11	PARIS 18E ARRD	10011	9.497153	82.194786	0.271347	0.1
12	PARIS 19E ARRD	13709	10.545846	94.471150	0.301310	0.1
13	PARIS 1ER ARRD	1413	8.909413	83.118188	0.254555	0.1
14	PARIS 20E ARRD	15340	9.830704	91.747523	0.280877	0.1
15	PARIS 2E ARRD	548	9.578467	78.762774	0.273670	0.1
16	PARIS 3E ARRD	1209	9.918941	82.564103	0.283398	0.1
17	PARIS 4E ARRD	2740	10.486496	90.928832	0.299614	0.1
18	PARIS 5E ARRD	2368	11.425676	96.956081	0.326448	0.1
19	PARIS 6E ARRD	1764	11.596939	90.792517	0.331341	0.1
20	PARIS 7E ARRD	8617	11.633399	95.964953	0.332383	0.1
21	PARIS 8E ARRD	7245	11.623741	105.267909	0.332107	0.1
22	PARIS 9E ARRD	1167	10.423308	82.156812	0.297809	0.1
23	SEINE-SAINT-DENIS	11570	8.867416	88.493258	0.253355	0.1
24	VAL-DE-MARNE	7580	11.189446	103.697098	0.319698	0.1

Quelques résultats tirés du notebook

```
[]: # Etablissement d'une regle de ponderation pour calculer le coefficient de contrainte das chaque arrondissement
# La formulation s'ecrit comme suit :
# count_arbres_updated = count_arbres + (hauteur_mean / hauteur_max)*count_arbres +
# (circonference_mean / circonference_max)*count_arbres.
```

La même règle est appliquée pour la circonference.

Out[80]:

	arrondissement	count_arbres	Hauteur_mean	circonference_mean	coeff_hauteur	coeff_circonference	count_arbres_updated
0	BOIS DE BOULOGNE	3978	9.594017	78.464806	0.274115	0.112093	5514.332857
1	BOIS DE VINCENNES	11510	10.784796	90.111121	0.308137	0.128730	16538.341429
2	HAUTS-DE-SEINE	5298	8.169309	70.992827	0.233409	0.101418	7071.914286
3	PARIS 10E ARRDT	3385	10.881536	87.820384	0.310901	0.125458	4862.074286
4	PARIS 11E ARRDT	5658	10.476670	82.925239	0.299333	0.118465	8021.901429
5	PARIS 12E ARRDT	12600	9.255476	85.332619	0.264442	0.121904	17467.958571
6	PARIS 13E ARRDT	16696	9.437769	77.766395	0.269651	0.111095	23052.925216
7	PARIS 14E ARRDT	11399	9.302307	97.747259	0.265780	0.139639	16020.372857
8	PARIS 15E ARRDT	17151	8.967874	82.366218	0.256225	0.117666	23563.604286
9	PARIS 16E ARRDT	16403	10.728708	94.500762	0.306535	0.135001	23645.508571
10	PARIS 17E ARRDT	10762	9.335997	86.701728	0.266743	0.123860	14965.662857
11	PARIS 18E ARRDT	10011	9.497153	82.194786	0.271347	0.117421	13902.960000
12	PARIS 19E ARRDT	13709	10.545846	94.471150	0.301310	0.134959	19689.807143
13	PARIS 1ER ARRDT	1413	8.909413	83.118188	0.254555	0.118740	1940.465714
14	PARIS 20E ARRDT	15340	9.830704	91.747523	0.280877	0.131068	21659.238571
15	PARIS 2E ARRDT	548	9.578467	78.762774	0.273670	0.112518	759.631429
16	PARIS 3E ARRDT	1209	9.918941	82.564103	0.283398	0.117949	1694.228571
17	PARIS 4E ARRDT	2740	10.486496	90.928832	0.299614	0.129898	3916.864286
18	PARIS 5E ARRDT	2368	11.425676	96.956081	0.326448	0.138509	3469.017143
19	PARIS 6E ARRDT	1764	11.596939	90.792517	0.331341	0.129704	2577.282857
20	PARIS 7E ARRDT	8617	11.633399	95.964953	0.332383	0.137093	12662.471429
21	PARIS 8E ARRDT	7245	11.623741	105.267909	0.332107	0.150383	10740.637143
22	PARIS 9E ARRDT	1167	10.423308	82.156812	0.297809	0.117367	1651.510000
23	SEINE-SAINT-DENIS	11570	8.867416	88.493258	0.253355	0.126419	15963.981429
24	VAL-DE-MARNE	7580	11.189446	103.697098	0.319698	0.148139	11126.205714

Importation de nouvelles données pour l'optimisation

```
In [102]: # Chargement du fichier excel update avec les surfaces des arrondissements et la densité des arbres.  
# L'idée est d'ajouter ces paramètres dans l'optimisation des sorties.  
data_analysis_updated = pd.read_excel('analysis_data.xlsx')  
data_analysis_updated.columns
```

Out[116]:

	arrondissement	count_arbres_updated	surface	densite
0	BOIS DE BOULOGNE	5514.332857	995	3.997990
1	BOIS DE VINCENNES	16538.341429	995	11.567839
2	HAUTS-DE-SEINE	7071.914286	176	30.102273
3	PARIS 10E ARRD'T	4862.074286	289	11.712803
4	PARIS 11E ARRD'T	8021.901429	367	15.416894
5	PARIS 12E ARRD'T	17467.958571	637	19.780220
6	PARIS 13E ARRD'T	23052.925216	715	23.351049
7	PARIS 14E ARRD'T	16020.372857	564	20.210993
8	PARIS 15E ARRD'T	23563.604286	848	20.225236
9	PARIS 16E ARRD'T	23645.508571	791	20.737042
10	PARIS 17E ARRD'T	14965.662857	567	18.980600
11	PARIS 18E ARRD'T	13902.960000	601	16.657238
12	PARIS 19E ARRD'T	19689.807143	679	20.189985
13	PARIS 1ER ARRD'T	1940.465714	183	7.721311
14	PARIS 20E ARRD'T	21659.238571	598	25.652174
15	PARIS 2E ARRD'T	759.631429	99	5.535354
16	PARIS 3E ARRD'T	1694.228571	117	10.333333
17	PARIS 4E ARRD'T	3916.864286	160	17.125000
18	PARIS 5E ARRD'T	3469.017143	254	9.322835
19	PARIS 6E ARRD'T	2577.282857	215	8.204651
20	PARIS 7E ARRD'T	12662.471429	409	21.068460
21	PARIS 8E ARRD'T	10740.637143	388	18.672680
22	PARIS 9E ARRD'T	1651.510000	218	5.353211
23	SEINE-SAINT-DENIS	15963.981429	23620	0.489839
24	VAL-DE-MARNE	11126.205714	24500	0.309388

Utilisation de conditions d'affectation de point de maintenance selon le nombre d'arbres virtuels, et la surface de l'arrondissement.

atuation des conditions d'attribution des point de maintenance par rapport au besoin établi sur la base de la
aison des paramètres : hauteur, circonference, nombre d'arbre et surface de l'arrondissement.

```
lysis_updated.loc[((data_analysis_updated.count_arbres_updated<10000)&(data_analysis_updated.surface<500)), 'Point_maintenance']=1'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated<10000)&(data_analysis_updated.surface>500)), 'Point_maintenance']=2'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated<10000)&(data_analysis_updated.surface>500)), 'Point_maintenance']=2'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated>10000)&(data_analysis_updated.count_arbres_updated<20000)&(data_analysis_updated.surface<500)), 'Point_maintenance']=2'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated>10000)&(data_analysis_updated.count_arbres_updated<20000)&(data_analysis_updated.surface>500)&(data_analysis_updated.surface<1000)), 'Point_maintenance']=2'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated>10000)&(data_analysis_updated.count_arbres_updated<20000)&(data_analysis_updated.surface>1000)), 'Point_maintenance']=3'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated>20000)&(data_analysis_updated.surface<500)), 'Point_maintenance']=4'  
lysis_updated.loc[((data_analysis_updated.count_arbres_updated>20000)&(data_analysis_updated.surface>500)), 'Point_maintenance']=5'
```

Quelques résultats tirés du notebook

Out[186]:

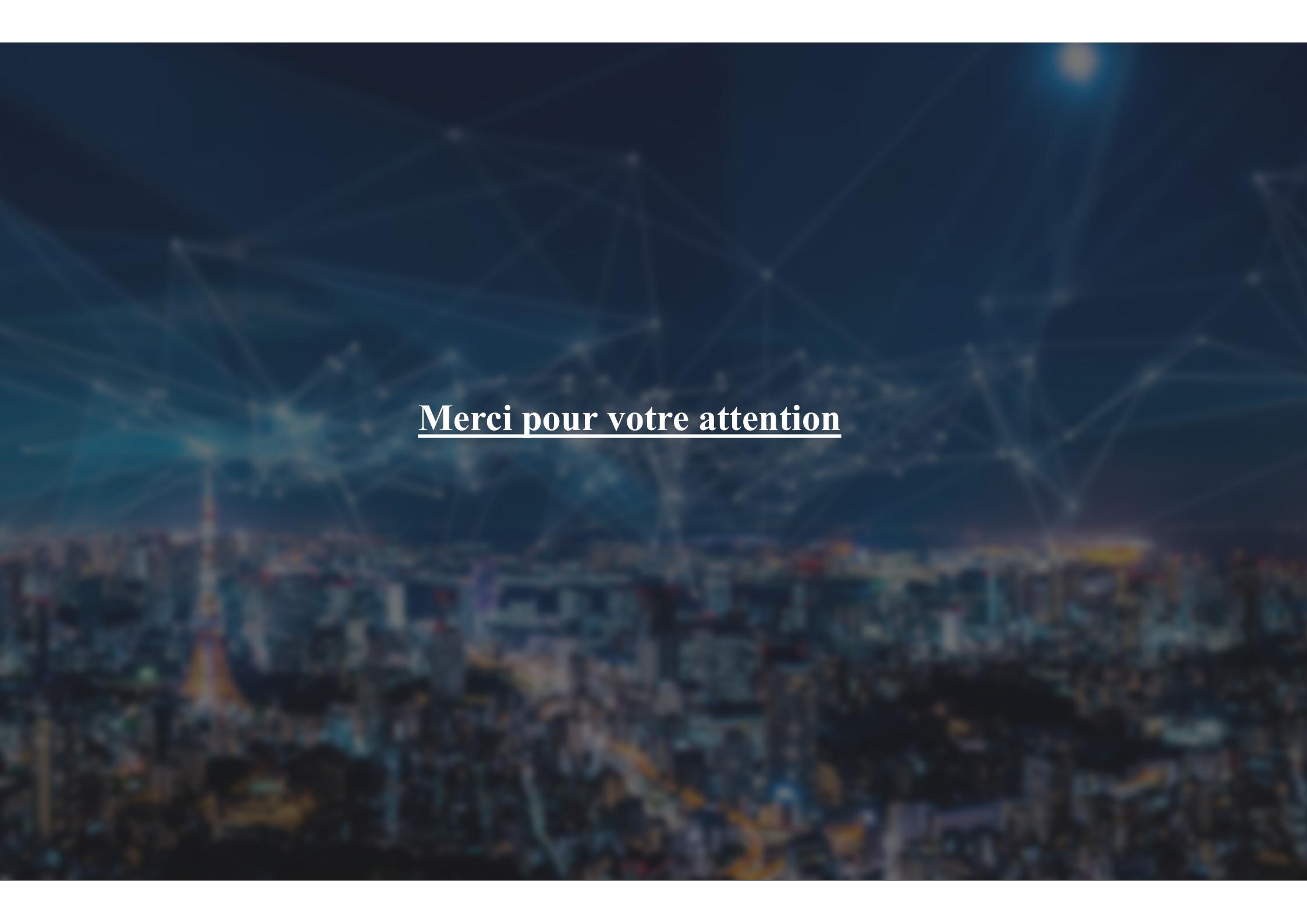
	arrondissement	count_arbres_updated	surface	densite	Point_maintenance
0	BOIS DE BOULOGNE	5514	995	3.997990	2
1	BOIS DE VINCENNES	16538	995	11.567839	2
2	HAUTS-DE-SEINE	7071	176	30.102273	1
3	PARIS 10E ARRDAT	4862	289	11.712803	1
4	PARIS 11E ARRDAT	8021	367	15.416894	1
5	PARIS 12E ARRDAT	17467	637	19.780220	2
6	PARIS 13E ARRDAT	23052	715	23.351049	5
7	PARIS 14E ARRDAT	16020	564	20.210993	2
8	PARIS 15E ARRDAT	23563	848	20.225236	5
9	PARIS 16E ARRDAT	23645	791	20.737042	5
10	PARIS 17E ARRDAT	14965	567	18.980600	2
11	PARIS 18E ARRDAT	13902	601	16.657238	2
12	PARIS 19E ARRDAT	19689	679	20.189985	2
13	PARIS 1ER ARRDAT	1940	183	7.721311	1
14	PARIS 20E ARRDAT	21659	598	25.652174	5
15	PARIS 2E ARRDAT	759	99	5.535354	1
16	PARIS 3E ARRDAT	1694	117	10.333333	1
17	PARIS 4E ARRDAT	3916	160	17.125000	1
18	PARIS 5E ARRDAT	3469	254	9.322835	1
19	PARIS 6E ARRDAT	2577	215	8.204651	1
20	PARIS 7E ARRDAT	12662	409	21.068460	2
21	PARIS 8E ARRDAT	10740	388	18.672680	2
22	PARIS 9E ARRDAT	1651	218	5.353211	1
23	SEINE-SAINT-DENIS	15963	23620	0.489839	3
24	VAL-DE-MARNE	11126	24500	0.309388	3

Le nombre des point de maintenance constitue la principale recommandation de cette analyse

Conclusion

1- L'optimisation des sorties sera etabli a travers la mise en place de point de maintenance qui seront utiles a raccourcir les distances d'intervention et auguementer la frequence et la qualite de la collecte de donnees.

2- l'optimisation a travers une approche plus realiste necessiterait un set de donnees plus elargi.



Merci pour votre attention