Statistique en mouvement : leçons du passé et perspectives d'avenir

Académie de Créteil

Apports de la programmation en Python avec Pandas pour l'enseignement de la statistique et des probabilités dans le secondaire

Frédéric Bro

14 mars 2018

Table des matières

Statistique avec Python

Utilisation de Pandas

Présentation

Un tableau de données peut être stocké dans un fichier csv.
Souvent les éléments de chaque ligne de ce tableau sont séparés par un point-virgule « ; ».
Parfois cela peut-être « , » ou un espace de tabulation « ».
L'intérêt d'un tel format est qu'il est lisible par bon nombre de logiciels ! Ainsi beaucoup de données statistiques ont été enregistrées dans ce format.

Exemple : voyons le fichier **stat.csv**

stat.csv

```
Valeurs ;Effectifs
1 ;4
2 ;6
3 ;2
4 ;3
```

Il correspond au tableau :

Valeurs	Effectifs
1	4
2	6
3	2
4	3

Avec Python :

1. Invoquons le module PANDAS sous le pseudo pa :

```
In [1]: import pandas as pa
```

2. Créons le tableau, nommé **T**, qui contient les données de **stat.csv**

```
In [2]: T=pa.read_csv('C:/.../stat.csv',sep=';')
```

Chemin d'accès indiquant l'emplacement de "stat.csv"
Cliquez droit sur le fichier, puis propriétés et enfin copiez-collez le chemin figurant dans sécurité.

';' indique à Python que les éléments de "stat.csv" sont séparés par ;

3. Voyons le contenu de la variable **T** :

```
In [3]: T
```

```
Out[3]:
```

	Valeurs	Effectifs
0	1	4
1	2	6
2	3	2
3	4	3

`T.loc[0,'Effectifs']`
`T.loc[1,'Effectifs']`
`T.loc[2,'Valeurs']`

Remarques :

— La première ligne de **T** est numérotée **0**, la suivante **1**, etc.
— Le tableau **T** est de type **DataFrame**.

Actions possibles sur T	SHELLS	
Afficher l'en-tête du tableau	In [4]: `T.head()` Out[4]:	Par défaut, il affiche les 5 premières lignes de T
Extraire la colonne **Effectifs**	In [5]: `T['Effectifs']` Out[5]:	`T['Effectifs']` est un objet de type **Series**
Obtenir la colonne des **ECC** *(Effectifs cumulés croissants)*	In [6]: `T['ECC']=pa.Series()` In [7]: `In [6]: T.loc[0,'ECC']=4` In [8]: `In [7]: for k in range(1,4):` `T.ix[k,'ECC']=T.ix[k,'Effectifs']+T.ix[k-1,'ECC']`	Ajoute à T une nouvelle colonne vide, nommée **ECC** On affecte 4 au premier élément de la colonne
Autre façon avec `cumsum()`	In [9]: `T['ECC']=T['Effectifs'].cumsum()` `T.head()` Out[9]:	
Former la colonne des fréquences	In [10]: `T['Freq']=T['Effectifs']/sum(T['Effectifs'])` `T.head()` Out[11]:	`sum(T['Effectifs'])` renvoie la somme des éléments de **Effectifs**
Multiplication de 2 colonnes	In [11]: `T['Valeurs']*T['Effectifs']` Out[12]:	Le résultat est une colonne

Out[4]:

	Valeurs	Effectifs
0	1	4
1	2	6
2	3	2
3	4	3

Out[5]:

0	4
1	6
2	2
3	3

Out[9]:

	Valeurs	Effectifs	ECC
0	1	4	4.0
1	2	6	10.0
2	3	2	12.0
3	4	3	15.0

Out[11]:

	Valeurs	Effectifs	ECC	Freq
0	1	4	4.0	0.266667
1	2	6	10.0	0.400000
2	3	2	12.0	0.133333
3	4	3	15.0	0.200000

Out[12]:

0	4
1	12
2	6
3	12

Étude d'une série de notes

Dans le fichier **notes.csv** se trouve le tableau suivant :

Notes	Effectifs
4	2
8	6
10	8
13	2
15	3

1. Avec PANDAS, créer le tableau **T** associé à ce fichier csv.
2. Calculer la moyenne $\overline{x}$ de ces notes.
3. Ajouter à **T** la colonne des fréquences.
4. **a.** Ajouter à **T** la colonne des fréquences cumulées croissantes.
   b. En déduire le premier quartile Q_1.

Étude d'une série « brute » de données

Lors d'une correction au BAC, un professeur obtient le fichier de notes **Bac.csv**.

1. Avec PANDAS, créer le tableau T associé à ce fichier csv.
2. **Nombre de candidats (ici lignes de T)**
   a. Pour obtenir le nombre de valeurs, saisir :

```
In [4]: len(T)

Out[4]:
```

   b. Cela permet d'obtenir le nombre de .....
3. **Comptage des valeurs :**
   a. Saisir le code ci-dessous puis recopier les 4 premières lignes du résultat obtenu :

```
In [5]: T['Notes'].value_counts()

Out[5]:
```

   b. Combien de candidats ont eu 10 ?
4. Recopier et compléter les résultats obtenus :

```
In [6]: T.describe()

Out[6]:
```

	Candidat	Notes
count	58.000000	58.000000
mean	29.500000	13.103448
std	16.886879	4.343536
min	1.000000	4.000000
25%	15.250000	10.000000
50%	29.500000	13.500000
75%	43.750000	17.000000
max	58.000000	20.000000

Compléter :

$\overline{x}$	Q_1	m_e	Q_3

5. Sélectionner des lignes de T suivant une condition

a. Saisir :

```
In [7]:  T[T['Notes']>=10]
```

Ce sont les lignes de **T** concernant les candidats ayant eu une note .....

b. Compléter les pointillés pour obtenir le tableau extrait de **T**, concernant les candidats dont la note est <8 :

```
In [8]:  T[.................................]
```

6. Sélectionner des lignes de T suivant plusieurs conditions

Commandes pour relier 2 conditions entre elles :		
En python	&	|
Signification	**et**	**ou**

Code	Rôle
`In [9]:  T1=T[(T['Notes']>=10) & (T['Notes']<12)]` `In [10]:  T1.head()`	**T1** est le tableau formé de lignes de **T** où les notes sont $\geqslant 10$ **et** < 12
`In [11]:  T2=T[.................................]` `In [12]:  T2.head()`	**T2** est le tableau formé de lignes de **T** où les notes sont $\geqslant 8$ **et** < 10
`In [13]:  T3=T[.................................]` `In [14]:  T3.head()`	**T3** est le tableau formé de lignes de **T** où les notes sont <8 **ou** >16

7. Un candidat passe à l'oral si sa note est comprise dans l'intervalle $[8\,;10[$.
Compléter les pointillés pour calculer la fréquence des candidats convoqués à l'oral :

```
In [15]:  proportion=.........../.............
          proportion

Out[15]:  ..................
```

Activité

Le Titanic

Titanic.csv contient les données relatives aux passagers ayant embarqué sur le TITANIC, lors de son voyage inaugural (reliant SOUTHAMPTON à NEW-YORK en avril 1912).

A Étude de l'âge des passagers

1. Avec **pandas**, créer le tableau T associé à ce fichier.

```
In [1]: import pandas as pa

In [2]: T=pa.read_csv('C:/.../titanic.csv',sep=';',decimal=',')

In [3]: T.head()
```

Out[3]:

	pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	emb
0	1.0	1.0	Allen, Miss. Elisabeth Walton	female	29.0000	0.0	0.0	24160	211.3375	B5	
1	1.0	1.0	Allison, Master. Hudson Trevor	male	0.9167	1.0	2.0	113781	151.5500	C22 C26	
2	1.0	0.0	Allison, Miss. Helen Loraine	female	2.0000	1.0	2.0	113781	151.5500	C22 C26	
3	1.0	0.0	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1.0	2.0	113781	151.5500	C22 C26	
4	1.0	0.0	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1.0	2.0	113781	151.5500	C22 C26	

2. **a.** Saisir :

```
In [4]: T.describe()
```

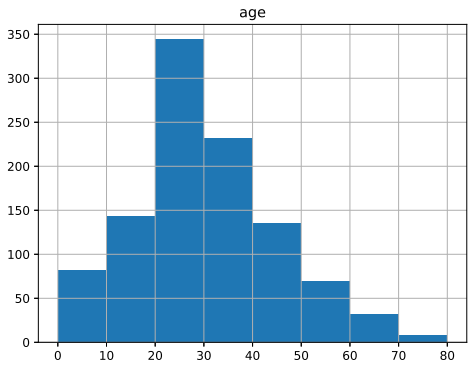
b. Quel est le nombre de passagers ?

c. Compléter le tableau ci-dessous relatif à l'âge des passagers :

$\overline{x}$	Q_1	m_e	Q_3

d. Pour obtenir l'histogramme des fréquences des passagers en fonction de leur âge :

```
In [5]: import pylab as pl

In [6]: T.hist(column='age',bins=range(0,90,10))

In [7]: pl.show()
```

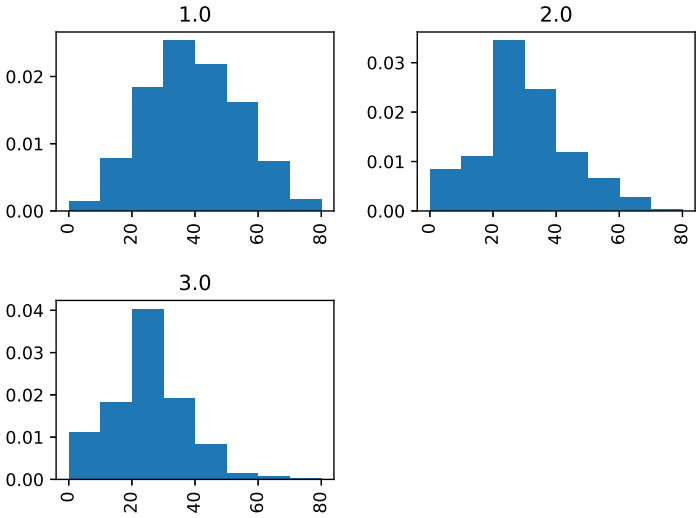


Reporter sur cet histogramme l'âge médian des passagers.

e. **Pour chaque classe**, on représente l'histogramme des fréquences des passagers en fonction de leur âge :

```
In [8]:   T.hist(column='age',bins=range(0,90,10),normed=True,by='pclass')
```

```
In [9]:  pl.show()
```



Par lecture approximative de ces histogrammes, compléter :

Numéro classe	1	2	3
Age médian			

B Étude relative aux survivants

1. **Proportion de survivants**

a. Compléter pour créer le tableau **T1**, contenant les lignes du tableau **T** associées aux survivants du naufrage.

```
In [10]:  T1=T[.................................]
```

b. Calculer la fréquence de survivants.

2. **Proportion de survivants par classe**

a. Saisir :

```
In [11]:  a=T1['pclass'].value_counts()
```

```
In [12]:  a
```

```
Out[11]:
```

La variable a est la série statistique qui à chaque classe associe .....

b. Saisir :

```
In [13]:  b=T['pclass'].value_counts()
```

```
In [14]:  b
```

```
Out[14]:
```

La variable b est la série statistique qui à chaque classe associe .....

c. Déterminer la proportion de survivants pour chaque classe.

3. Proportion de survivants par sexe

Déterminer la proportion de survivants pour chaque sexe.

4. La règle « les femmes et les enfants d'abord » fut-elle respectée ?

a. Former le tableau **T2** des lignes qui correspondent aux survivants femmes ou enfants (moins de 18 ans).

b. Former le tableau **T3** des lignes qui correspondent aux femmes ou enfants (moins de 18 ans).

c. Répondre à la question posée.

Activité

Statistique sportive

En 2008, le cycliste américain CHRISTIAN VANDE VELDE à participa la 10-ième étape du tour de France.
Sur son vélo, une balise GPS a été fixée.
Cette balise enregistre régulièrement, durant la course :

- sa position GPS : latitude et longitude (exprimées en degré)
- son altitude (exprimée en mètre)

Les données sont collectées dans un tableau via le fichier **velo.csv**.

A Étude statistique

1 Collecte des données

1. Saisir dans le SHELLS les instructions suivantes :

```
In [1]: import pandas as pa

In [2]: T=pa.read_csv('C:/.../velo.csv',sep=';')
```

Chemin d'accès indiquant l'emplacement de `"velo.csv"`
Les pointillés sont à compléter avec le professeur !!

`';'` indique à Python que les éléments de `"velo.csv"` sont séparés par ;

2. **a.** Afficher l'entête du tableau **T**.

b. Déterminer le nombre de lignes de **T**.

2 Distance parcourue

L'objectif est d'ajouter au tableau T, la colonne nommée `"Distance"` où :

- `T.loc[0,'Distance_Cum]` = 0
- `T.loc[1,'Distance_Cum]` correspond à la distance entre le point n° 0 et n° 1
- `T.loc[2,'Distance_Cum]` correspond à la distance entre le point n° 1 et n° 2

  etc.

1. Compléter les instructions en pointillé puis les saisir dans le SHELLS :

```
In [4]: import geopy.distance as gd

In [5]: T['Distance']=pa.Series()

In [6]: T.loc[0,'Distance']=....

In [7]: for k in range(1,len(T)):
            T.loc[k,'Distance']=gd.distance(.....,.....).kilometers
```

Pour calculer la distance entre les points A de coordonnées p1 et B de coordonnées p2, on fait appel à la fonction `gd.distance` du module GEOPY.DISTANCE.
Il suffit de saisir :
`gd.distance(p1,p2).kilometers`

2. Compléter la ligne en pointillé pour obtenir la colonne des distances cumulées associée à `T['Distance']` :

```
In [8]: T['Distance_Cum']=.....
```

3. Représenter l'altitude en fonction de la distance parcourue depuis le départ.

3 Le temps

1. **a.** Exprimer en heure le temps de départ du cycliste.

b. Pour ajouter au tableau T une colonne nommée `"Temps"` contenant les temps (exprimé en heure), de chaque relevé, compléter les pointillés ci-dessous :

```
In [14]: T['Temps']=........
```

2. Ajouter au tableau **T**, la colonne des temps entre deux enregistrements de position :
   - "Delta_time" sera le nom de cette nouvelle colonne.
   - T.loc[0,'Delta_time']=0, par convention.
   - T.loc[1,'Delta_time'], correspondra au temps qu'il a fallu attendre entre le premier et le deuxième enregistrement.

4 Étude de la vitesse entre deux enregistrements

1. Ajouter au tableau **T**, une nouvelle colonne nommée "Vitesse" .
2. Par défaut, affecter la valeur 0 au premier élément de cette colonne.
3. Calculer la vitesse du cycliste entre le premier et le deuxième enregistrement ?
4. Afin d'obtenir la vitesse du cycliste durant deux enregistrements consécutifs, compléter l'instruction en pointillé ci-dessous :

```
In [21]: for k in range(1,len(T)):
             T.loc[k,'Vitesse']=.........
```

5. Afficher l'en-tête du tableau **T**.
6. Représenter le diagramme de la vitesse du cycliste en fonction de sa distance parcourue depuis le point de départ.
7. Calculer la vitesse moyenne du cycliste pendant la course.

5 Étude de la pente

1. Ajouter au tableau **T**, une nouvelle colonne nommée "pente".
2. Par défaut, affecter la valeur 0 au premier élément de cette colonne.
3. Par définition, lors du déplacement du cycliste, **la pente** est le rapport (exprimé en %), entre l'accroissement de son altitude et sa distance parcourue.
4. Afin d'obtenir la vitesse du cycliste durant deux enregistrements consécutifs, compléter l'instruction en pointillé ci-dessous :

```
In [27]: for k in range(1,len(T)):
             T.loc[k,'pente']=.........
```

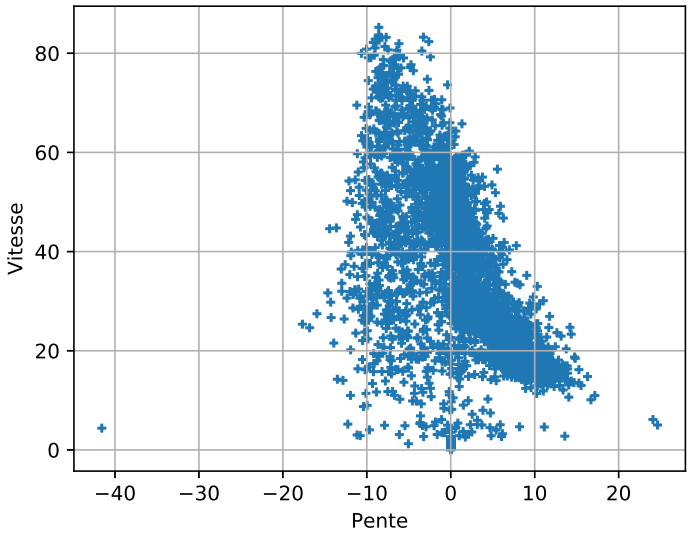
5. Afficher l'en-tête du tableau **T**.
6. Représenter le diagramme de la pente du cycliste en fonction de sa distance parcourue depuis le point de départ.

6 Résumés statistiques

1. Créer le tableau **T1** qui correspond seulement aux colonnes 'Altitude (m)', 'Vitesse' et 'Pente'.
2. Déterminer la moyenne, le premier et troisième quartiles, la médiane de chacune de ces séries statistiques.
   Indication : utiliser la commande describe.

7 Vitesse en fonction de la pente

On a représenté la vitesse en fonction de la pente.



1. A partir de quelle pente, la vitesse du cycliste a t-elle dépassé 60 km/h ?
2. Lorsque la pente est supérieure à 10 %, donner un encadrement de la vitesse du cycliste.

B Représentation du chemin

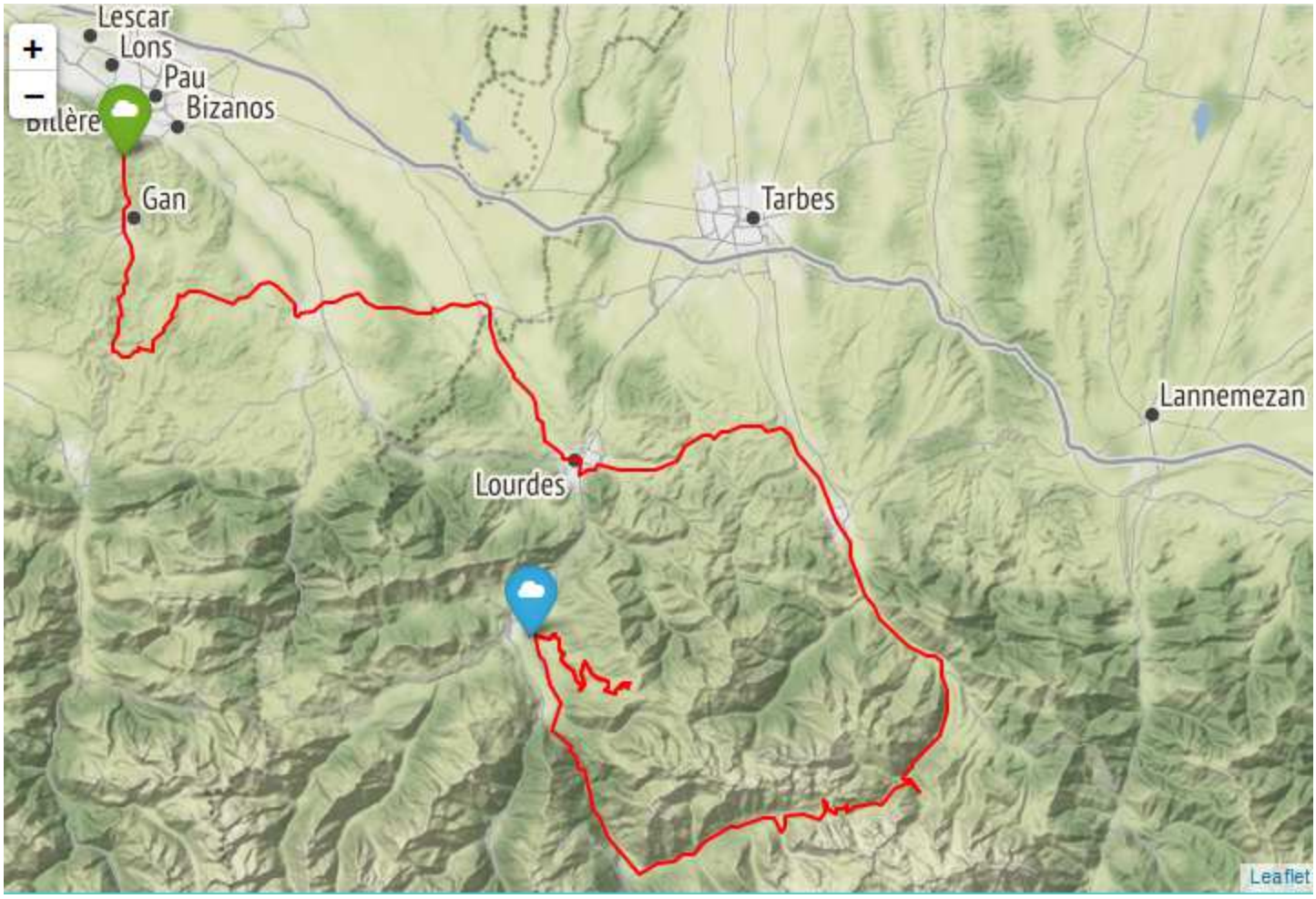
Objectif : on souhaite représenter sur une carte dynamique, le trajet du cycliste.

1. Recopier et exécuter les instructions suivantes :

```
In [38]: import folium

In [39]: moy_Lat=T['Lat'].mean(), moy_Long=T['Long'].mean()

In [40]: carte = folium.Map(location=[moy_Lat,moy_Long],tiles='Stamen Terrain', zoom_start=13)

In [41]: Loc=[(T.loc[k,'Lat'],T.loc[k,'Long']) for k in range(len(T))]

In [42]: folium.PolyLine(Loc,color='red',weight=2, opacity=1).add_to(carte)

In [43]: carte.save('C:/.../carte.html')
```

Inscrire le même chemin d'accès que pour copier `"velo.csv"`.
Si vous n'êtes pas sûr, demandez à votre professeur !!

On obtiendra la carte suivante :



2. Identifier le point de départ de la course et celui d'arrivée.
3. Localiser sur la carte le point le plus élevé du parcours.

Activité

Le pollen à Paris

Tout au long de l'année, la Mairie de PARIS assure la surveillance du patrimoine arboré et recense chacun d'eux dans le fichier csv :

`Arbres.csv`

Certains arbres provoquent des fortes allergies au pollen. Voici la liste de ces arbres :

- Frêne
- Olivier
- Noisetier de Byzance
- Bouleau
- Charme
- Cyprès
- Mûrier
- Platane
- Aulne

Objectifs :

- Déterminer la proportions des arbres à fort potentiel allergisant.
- Conseiller les personnes ayant des allergies.

A Proportion des arbres allergisants

1. Ouverture du fichier `"Arbres.csv"` avec Pandas :

a. Saisir dans le SHELLS les instructions suivantes :

```
In [1]: import pandas as pa


In [2]: T=pa.read_csv('C:/.../Arbres.csv',sep=';',encoding='latin1')
```

b. Afficher l'entête du tableau T.

2. Un arbre d'alignement est un arbre couramment planté le long des rues.

a. Saisir dans le SHELLS les instructions suivantes :

```
In [4]: from random import *


In [5]: T.sample(n=10)
```

b. Quelle colonne nous indique si l'arbre est d'alignement ou non ?

c. En utilisant la commande `value_counts()`, calculer la proportion d'arbres d'alignements.

3. **a.** Recopier la liste `L` des arbres à fort potentiel allergisant :

```
In [8]: L=['Frêne',
           'Olivier',
           'Noisetier de Byzance',
           'Bouleau',
           'Charme',
           'Cyprès',
           'Mûrier',
           'Platane',
           'Aulne']
```

b. On restreint notre tableau de données T aux lignes concernant seulement les arbres d'alignement. On notera T1 le tableau formé de ces données.

```
In [9]: T1=T[.....................................]
```

c. On construit la fonction nommée **proportion** qui a pour arguments :

- — `colonne` (une colonne du tableau **T1**)
- — `liste` (une liste).

Cette fonction renvoie le nombre d'éléments de `colonne` qui appartiennent à `liste`.

Compléter les instructions en pointillé ci-dessous :

```
In [10]: def proportion(colonne,liste):
             n=0
             for arbre in colonne:
                 if arbre in ..........:
                     n=..........
             return ..../colonne.count()
```

d. En déduire la proportion d'arbres à fort potentiel allergisant.

B Répartition spatiale

Objectifs :

- — représenter la répartition des arbres à fort potentiel allergisants sur une carte « dynamique »
- — conseiller les personnes allergiques aux pollens

1. Le numéro des ligne du tableau T1 est obtenu en saisissant la commande suivante :

```
In [11]: T1.index.values
```

Quels sont les numéros des premières lignes ?

2. **a.** On saisit l'instruction suivante :

```
In [12]: type(T1.loc[7,'geo_point_2d'])

Out[12]: ..................
```

Quel est le type du premier élément de la colonne **geo_point_2d**, contenu dans le tableau **T1** ?

b. Recopier la fonction nommée **coordonnées** de paramètre `s` (une chaîne de caractères).

```
In [13]: def coordonnées(s):
             return float(s.split(',')[0]),float(s.split(',')[1])
```

Compléter ensuite :

```
In [14]: coordonnées('45.3,2.6')

Out[14]: ..................
```

3. On définit une liste vide nommée `Loc`.

Pour chaque ligne n°`k` du tableau **T1**, si le nom de l'arbre figurant dans la colonne **LIBELLEFRANCAIS** appartient à la liste `L`, alors on stocke dans `Loc`, les coordonnées GPS de cet arbre sous la forme (`latitude , longitude`).

Compléter les instructions en pointillé :

```
In [15]: Loc=[]
         for k in T1.index.values:
             if T1.loc[k,'..................'] in L:
                 Loc.append(.........................................)
```

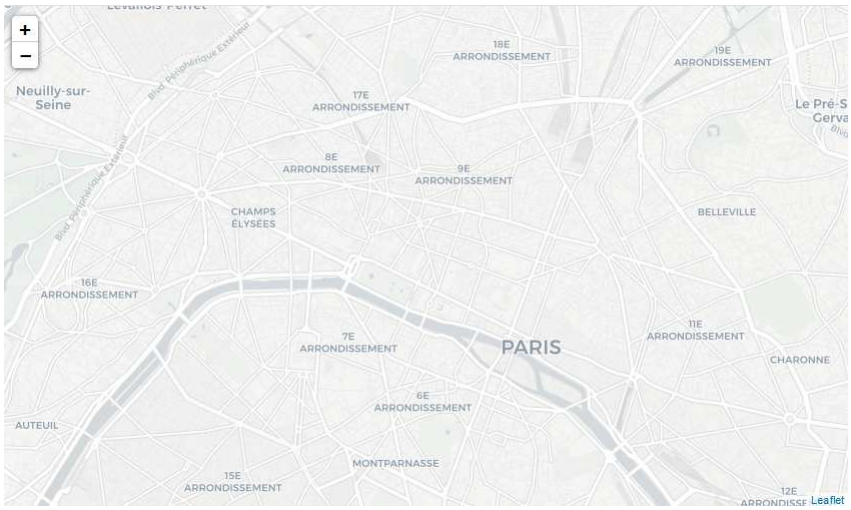
4. **a.** Pour représenter tous ces arbres, recopier les instructions suivantes :

```
In [16]: import folium
         from folium import plugins
```

In [17]:
```
carte = folium.Map(location=[48.866667,2.333333],tiles='cartodbpositron',
zoom_start=13)
```

In [18]:
```
carte
```

Out[18]:



In [19]:
```
plugins.HeatMap(Loc,radius=10,blur=0).add_to(carte)
```

In [20]:
```
carte
```

b. Que peut-on conseiller aux personnes allergiques aux pollens souhaitant se balader dans PARIS ?

Activité

Challenge Data Scientist

Data scientist : Le métier le plus attrayant du XXI-ième siècle ?

Le terme « data scientist » a été inventé par DHANURJAY PATIL (LinkedIn) et JEFF HAMMERBACHER (Facebook) en cherchant comment caractériser les métiers des données pour afficher des offres d'emploi :
« Analyste, ça fait trop Wall Street ;
statisticien, çà agace les économistes ;
chercheur scientifique, çà fait trop académique.
Pourquoi pas "data scientist" ? »

Problématique

d'après l'article de challenge disponible via ce lien url

`https://www.challenges.fr/economie/la-concurrence-inquietante-d-airbnb-pour-les-hoteliers_416014`

on peut lire « La concurrence inquiétante d'AIRBNB pour les hôteliers ».

Explorer, seul ou en groupe, le fichier :

`Airbnb_Paris.csv`

On pourra répondre aux questions suivantes :

1. Lister les variables et déterminer leur type.
2. Calculer les indicateurs statistiques de chaque variable quantitative.
   Tous les logements sont ils loués ?
   Que dire des variables disponibilité et avis_par_mois ?
3. Étudier ce que rapporte un logement à l'année.
   (On s'intéresse aux logements dont la disponibilité est ..... et le nombre d'avis est .....)
4. Où les hôtels ont-ils le plus de concurrence ?
   (Utiliser graphique(s), carte(s) pour répondre à cette question).

Activité

Séismes de magnitude supérieure à 5

Durant les 30 derniers jours, on peut obtenir le relevé des différents séismes dans le monde.
Pour chacun d'eux est renseigné notamment :

- sa latitude et longitude (exprimées en degré)
- le temps (exprimé en UTC : COORDINATED UNIVERSAL TIME)

  Exemple : `2018-02-21T17:59:34 .564Z`

  ⇕

  21 février 2018 à 17h59m34s
- sa magnitude (exprimée selon l'échelle de RICHTER)
- sa profondeur (exprimée en km)

Les données sont disponibles directement depuis le lien :

`http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.csv`

Objectifs :

- Analyser ce jeu de données.
- Modéliser le temps d'attente entre deux séismes de magnitude supérieure à 5.
- Calculer la proportion de séismes qui se sont produits ces 30 derniers jours, dans les pays de l'Asie du sud-est *(*INDONÉSIE, SINGAPORE*, etc)*.
  Cette zone appelée aussi **INSULIDE** correspond à une latitude allant de -13° à 15° et une longitude allant de 90° à 170°. Ce lieu est le carrefour de plusieurs plaques géologiques et est un lieu réputé sensible.
- Calculer la probabilité d'avoir au moins un séisme sur une période.

A Extraction des données

1. Saisir dans le SHELLS les instructions suivantes :

```
In [1]: import pandas as pa

In [2]: T=pa.read_csv("http://earthquake.usgs.gov/earthquakes/
        feed/v1.0/summary/all_month.csv",sep=',')
```

2. Afficher l'entête du tableau **T**.

3. Combien de séismes se sont produits au cours de ces 30 derniers jours ?

B Étude du temps d'attente entre 2 séismes

1 Conversion d'un temps en format UTC vers datetime

L'objectif est d'évaluer le nombre de secondes qui séparent deux séismes consécutifs.
Pour soustraire deux temps, on va utiliser la fonction **datetime** du module DATETIME.
La fonction `datetime` prend 6 paramètres (entiers) : l'année - le mois - le jour - l'heure - les minutes - les secondes.
Exemple : le 14 mars 2018 à 15h30 est associé à

```
In [5]: datetime(2018,3,14,15,30,0)
Out[5]: datetime.datetime(2018, 3, 14, 15, 30)
```

1. On définit la fonction **conversion** qui prend pour argument `s` (une chaîne de caractère correspondant au temps écrit au format UTC) et qui renvoie la liste des entiers (années - mois - jour - l'heure - les minutes - les secondes).

```
In [6]: def conversion(s):
            k=0
            Mot=''
            while s[k]!='.':
                if s[k]!='-' and s[k]!='T' ..... s[k]!=':':
                    Mot=Mot+.....
                else:
                    Mot=Mot+.....
                k=..........
            Liste_string=Mot.split('-')
            return [int(a) for a in Liste_string]
```

Principe :

- On crée le mot vide `Mot`.
- On parcourt chaque lettre du mot `s` tant que celle-ci est différente de `'.'`
  - si la lettre lue est différente de `'-'` ou `'T'` ou `':'` alors l'ajoute au mot `Mot`
  - sinon on ajoute `'-'` au mot `Mot`.

Compléter les pointillés.

2. Compléter les instructions en pointillés pour obtenir le temps où s'est produit le dernier séisme exprimé avec **`datetime`** :

```
In [7]: conversion(T.loc[0,'time'])

Out[7]: ..................................................

In [8]: t=conversion(T.loc[....,'time'])
        datetime(t[0],t[1],t[2],.....,.....,.....)

Out[7]: ..................................................
```

3. On définit la fonction **`horaire`** qui prendra pour argument `UTC` (une chaîne de caractère correspondant au temps écrit au format UTC situé dans la colonne **timing** du tableau).

Elle renvoie ce temps au format datetime.

```
In [9]: def horaire(UTC):
            t=conversion(UTC)
            return datetime(t[0],.....,.....,.....,.....,.....)
```

Compléter les pointillés.

2 Création du tableau T1 ne concernant que les séismes d'amplitude >5

1. **a.** Compléter la condition à écrire pour créer le tableau **T1** formé des lignes de **T** concernant seulement les séisme de magnitude >5 :

```
In [10]: T1=T[.....................]
```

b. Afficher l'en-tête de **T1**.

2. En utilisant la fonction **`describe`** au tableau **T1**, compléter le tableau ci-dessous :

Séismes de magnitude >5	$\overline{x}$	Q_1	m_e	Q_3
Magnitude				
Profondeur				

3 Création de la liste des temps entre deux séismes consécutifs

1. On définit la fonction **`attentes`** qui prendra pour argument `Tab` (un tableau de données contenant la colonne **time** formées des temps en format UTC).

Cette fonction renvoie la liste des temps d'attente , **exprimés en seconde**, entre deux séismes consécutifs.

```
In [13]: def attentes(Tab):
             L=[]
             Liste_Temps=list(Tab['time'])
             for k in range(1,len(Liste_Temps)):
                 d1=horaire(Liste_Temps[k-1])
                 d2=horaire(.....................)
                 d=.............
                 L.append(d.seconds)
             return L
```

Compléter les pointillés.

2. Créer la liste L formée des temps d'attente entre deux séismes consécutifs contenus dans **T1**.

4 Modélisation

On note S la variable aléatoire qui, à chaque séisme pris au hasard, associe le temps d'attente *(exprimé en seconde)* du prochain séisme.
Les valeurs de la liste L forment un échantillon de plusieurs réalisations de S.

1. **Proposition « vraisemblable » d'une loi de probabilité pour S :**
   a. Représenter l'histogramme normalisé des fréquences associé à L.
   Indication : utiliser la commande `pl.hist` *du module* PYLAB *pour représenter cet histogramme.*
   b. Calculer avec PYTHON, le temps moyen d'attente entre deux éruptions.
   Indications :
   — *utiliser la commande* `np.mean` *du module* NUMPY *pour calculer ce temps moyen d'attente.*
   — *On stockera ce temps moyen dans une variable nommée* `m`.
   c. Quelle célèbre loi, permet d'affirmer que $\mathbb{E}[S] \approx$ `m`?
   d. En observant l'histogramme obtenu, ajuster à cet histogramme une loi à densité de référence pour S.
   Avec ce qui précède, proposer son ou ses éventuels paramètres.
2. **Les statisticiens affirment que le modèle choisi est acceptable.**
   Durant 30 jours étalés sur Janvier-Février 2018, en moyenne 20 728 secondes ont séparé 2 séismes consécutifs *(de magnitude >5)* dans le monde.
   a. Convertir ce temps moyen en heures.
   b. Calculer la probabilité qu'il ne se produise aucun nouveau séisme *(de magnitude >5)* pendant au moins un jour.

C Carte des séismes

Pour représenter les séismes (de magnitude >5) sur une carte « dynamique », recopier les instructions suivantes :

```
In [25]: import folium

In [26]: carte = folium.Map(location=[0,0],tiles='Mapbox Control Room', zoom_start=2)

In [27]: for lat,long,mag in zip(T1['latitude'],T1['longitude'],T1['mag']):
             folium.CircleMarker(location=[lat,long],
                                 fill=True,
                                 fill_color='red',
                                 fill_opacity=0.5,
                                 color='yellow',
                                 popup=str(mag),
                                 radius=np.sqrt(mag)).add_to(carte))
```

D Activité sismique en Asie du Sud - Est

Les pays de l'Asie du sud-est *(*INDONÉSIE*,* SINGAPORE*, etc)* appartiennent à la zone (appelée **INSULIDE**) ayant une latitude comprise entre -13° à 15° et une longitude comprise entre 90° et 170°.

1. Avec PANDAS, créer le tableau **T2** formé des lignes du tableau T1 qui ne concernent que les séismes qui se sont produits en INSULIDE.
2. Calculer la proportion des séismes (de magnitude >5) qui se sont produits dans cette zone.
3. Que représente l'aire de cette zone par rapport à celle de la TERRE ?

E Nombre de séismes sur une période

Durant la même période des 30 jours étalés sur Janvier-Février 2018, il y a eu 108 séismes *(de magnitude >5)* dans le monde.

Un séisme se produit à une date exprimée au format UTC.

1. Calculer la proportion notée p de séismes par seconde durant cette période de 30 jours.
2. On note X le nombre de séismes qui se sont produits durant 1 jour.
   On suppose que pour deux séimes quelconques :
   - ils sont indépendants entre eux
   - ils ne peuvent débuter en même temps.

   a. Déterminer la loi de probabilité de X.

   b. Calculer la probabilité qu'il se produise au moins un séisme en l'espace d'un jour complet.
3. A partir de quelle heure, avons-nous plus d'une chance sur deux d'observer au moins un séisme *(de magnitude >5)* dans le monde ?

Petit résumé

Au départ	Importer le module **pandas**	`import pandas as pa`
	Associer à `"fichier.csv"` un tableau nommé **T**	`T=pa.read_csv(path,sep='...',encoding='...',decimal='...')` **path** est le paramètre obligatoire qui renseigne le chemin relatif associé à `"fichier.csv"` **sep** permet de préciser le séparateur utilisé pour **T** `';'` point virgule `','` virgule `' '` espace de tabulation **encoding** paramètre lié à l'encodage qu'il faut renseigner si le fichier comporte des accents : mettre **latin1**. par défaut c'est **utf8**. **decimal** si les nombres figurants dans T sont séparés par une virgule alors on précise à ce paramètre la valeur « `,` ».
	Lire l'en-tête de **T**	`T.head()`
Opérations	Obtenir une colonne nommée **Col**	`T['Col']`
	Obtenir l'élément situé à la ligne **i** et la colonne **Col**	`T.loc[i,'Col']`
	Connaître le nombre de lignes de **T**	`T.len()`
	Obtenir la somme des nombres contenus dans **Col**	`T['Col'].sum()`
	Obtenir la colonne des valeurs cumulées des nombres contenus dans **Col**	`T['Col'].cumsum()`
	Obtenir le résumé statistique de chaque colonne de nombres contenues dans **T**	`T.describe()`
	Obtenir le comptage des valeurs de la colonne **Col**	`T['Col'].value_counts()`
Sélection de lignes	Écrire la condition nommée **cond** correspondant à sélectionner les lignes de **T** où les éléments de **Col** sont >10	`cond=T['col']>10`
	Créer le tableau **T1** formé des lignes de **T** vérifiant la condition **cond**	`T1=T[cond]`
	Créer le tableau **T2** formé des lignes de **T** vérifiant : la condition **cond1** **et** la condition **cond2**	`T2=T[cond1 & cond2]`
	Créer le tableau **T3** formé des lignes de **T** vérifiant : la condition **cond1** **ou** la condition **cond2**	`T3=T[cond1 | cond2]`
Graphiques	Importer le module **pylab**	`import pylab as pl`
	Représenter l'histogramme des fréquences associé à la colonne **Col** de **T**	`T.hist(column='col',normed=True)`
	Représenter un nuage de points d'une colonne **Col1** en fonction d'une autre colonne **Col2** de **T**	`T.plot(kind='scatter',y='Col1',x='Col2')`
	Représenter le diagramme en barre associé à `T['Col'].value_counts()`	`T['Col'].value_counts().plot(kind='bar')`
	Représenter le diagramme circulaire associé à `T['Col'].value_counts()`	`T['Col'].value_counts().plot(kind='pie')`
	Représenter la ligne polygonale associée à `T['Col'].value_counts()`	`T['Col'].value_counts().plot()`

Installations :

Pour installer les deux modules folium et geopy :		
Méthodes	**folium**	**geopy**
conda	Ouvrir Anaconda prompt puis saisir :	
	`conda install -c conda-forge folium`	`conda install -c conda-forge geopy`
pip	Ouvrir Pyzo puis écrire dans le shells :	
	`pip install folium`	`pip install geopy`

Sitographie :

1. kaggle.com
2. https ://opendata.paris.fr/explore/dataset/les-arbres/
3. http ://insideairbnb.com/
4. http ://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/all_month.csv
5. https ://www.guidevtt.com/Membres/public_details_rando.php ?randoID=1396

Je fais des stats...