TP5

September 20, 2018

1 TP 5 : Propagation d'une maladie en Normandie

1.1 1 - Introduction

Nous allons dans ce TP simuler la propagation d'une maladie saisonnière en Normandie. Nous utiliserons pour cela les résultats des TP précédents sur les calculs de distance. Nous ferons appel pour ce TP aux connaissances acquises dans les précédents TP ainsi qu'au cours sur les générateurs aléatoires.

Le TP sera à réaliser en python 3. Les librairies utilisées sont installées sur les machines de l'université, vous pouvez néanmoins les installer sur vos propres machines à l'aide de l'utilitaire pip présent par défaut avec python.

N'hésite pas à regarder régulièrement la documentation de ces librairies, des exemples d'utilisation accompagnent généralement l'explication de chaque fonction.

- Python 3: https://docs.python.org/3/
- Numpy: https://docs.scipy.org/doc/numpy/reference/
- Scipy: https://docs.scipy.org/doc/scipy/reference/
- Matplotilb: https://matplotlib.org/contents.html

À part si cela est précisé, vous ne devez pas utiliser directement de boucle (`for`,`while`) ou de branchement conditionnel (`if`) durant ce TP.

```
In [1]: import numpy as np
        import scipy as sc
        import scipy.misc
        import matplotlib.pyplot as plt
```

1.2 2 - Le début de l'épidémie

Dans le précédent TP nous avons calculé les coordonnées cartésiennes des villes de Normandie puis nous avons approximé la distance entre villes en utilisant une distance euclidienne entre ces points.

Reprendre dans le code du TP3, la lecture du fichier *data.pickle*.

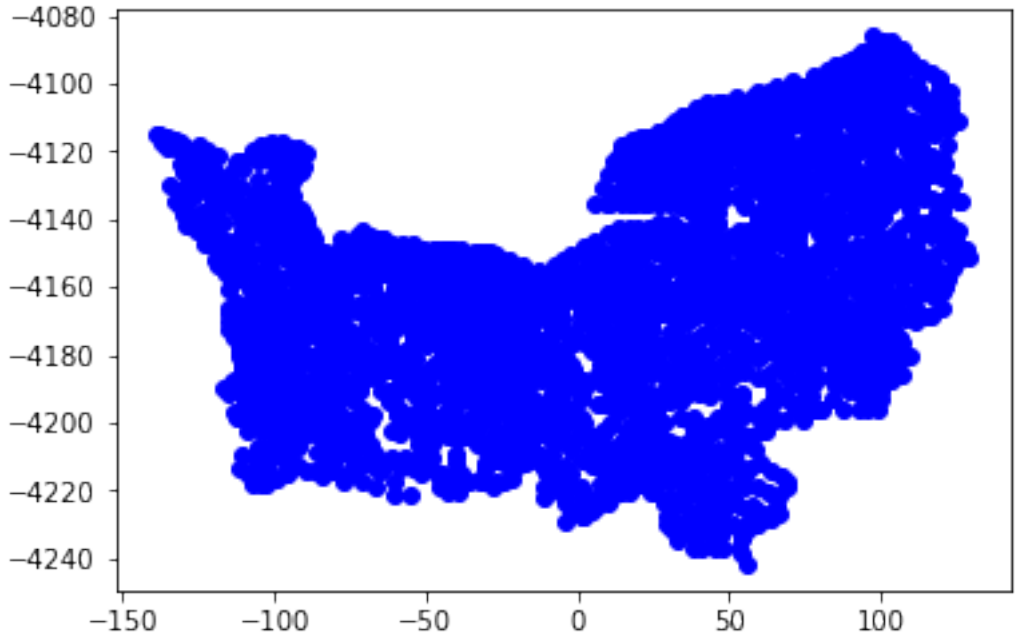
Récupérez du précédent TP le code permettant d'avoir la matrice des distances entre ville et stocker la dans une matrice nommé *d_villes*. Si vous n'avez pas fait cette question, vous pouvez utiliser le fichier *d_villes.npy* contenant cette matrice. Le fichier se charge dans python avec le code suivant :

```
In [3]: d_villes = np.load('d_villes.npy')
```

Faites de même avec le fichier contenant les coordonnées 3D des villes de Normandie.

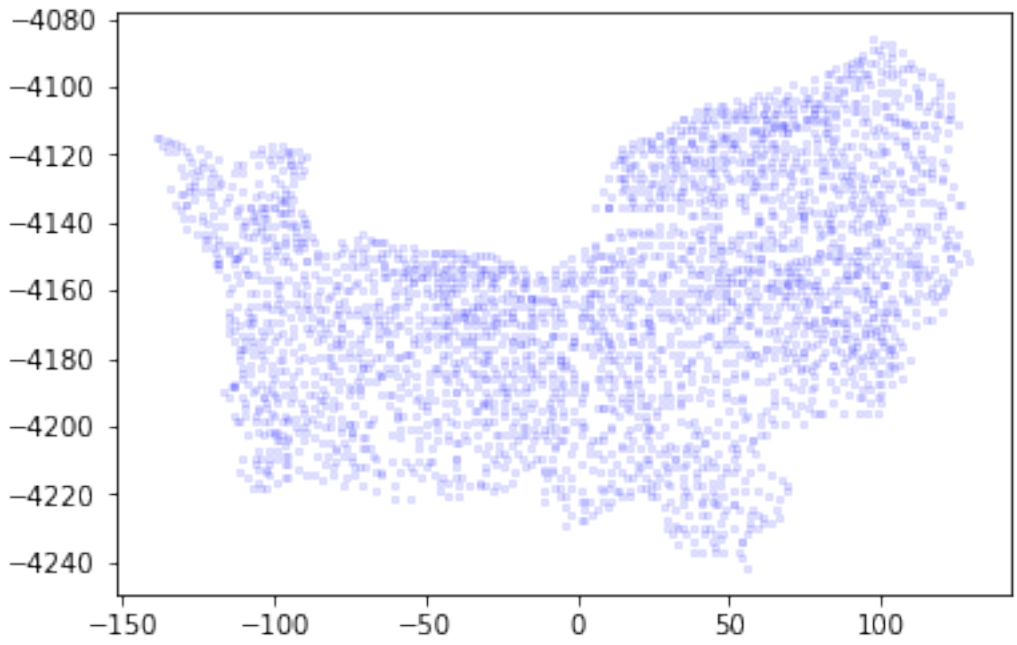
Comme dans le TP précédent, affichez les villes normandes à l'écran avec la fonction `plt.scatter` de matplotlib.

In [5]:



Ajoutez aux arguments de la fonction `plt.scatter` de la question précédente les arguments `color='b',marker='s',s=5,alpha=0.1`. À votre avis que font ces arguments ?

In [6]:

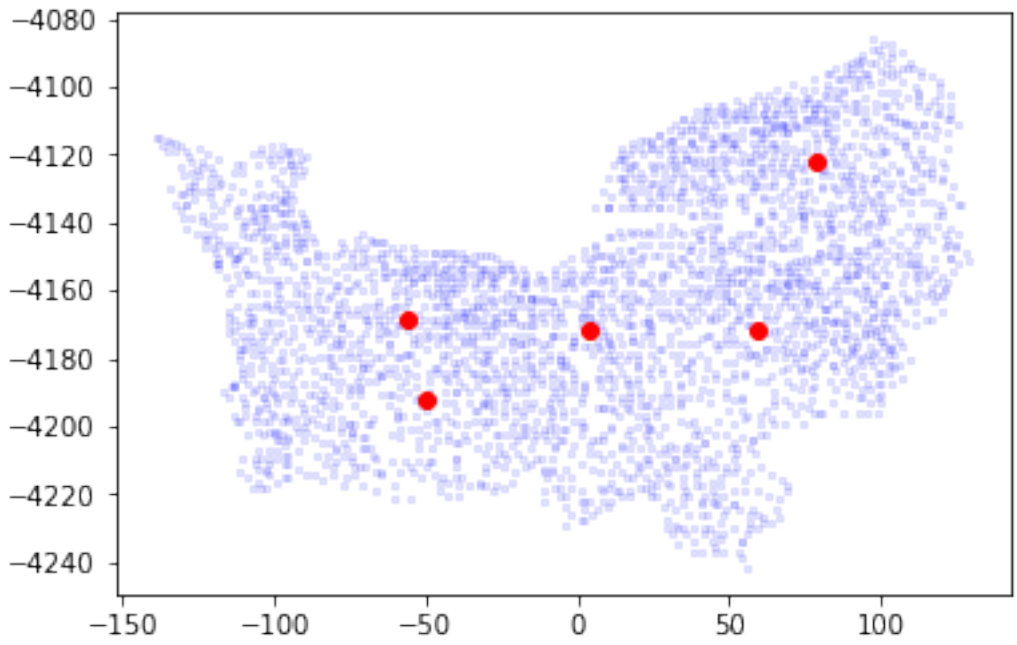


Pour le début de l'épidémie nous allons considérer que chaque ville à un risque de $\frac{1}{500}$ d'être un foyer de l'infection. Nous allons construire un vecteur contenant autant d'éléments qu'il y a de ville et dont les valeurs sont soit True (si la ville est un foyer initiale de l'infection) soit False. Pour construire ce vecteur, faites les opérations suivantes :

1. Définissez une variable n contenant le nombre de villes de Normandie.
2. Tirez au hasard selon une loi uniforme n valeurs entre 0 et 1.
3. Testez si les valeurs du vecteur précédent sont inférieures à $\frac{1}{500}$. Le résultat de chaque test formera le vecteur `ville_src` que l'on souhaite construire. Vous n'avez pas besoin de `for` pour cette question

Reprenez la précédente figure. Superposer sur l'image des points rouges correspondants aux villes contaminées. Si vous relancez le code précédent, les villes en rouges devraient changer.

In [8]:



1.3 3 - La propagation de la maladie

On va dans cette partie modéliser la propagation de la maladie. On va supposer que les villes nouvelles contaminées suivent une loi normale centrée sur les villes déjà infectées.

Définissez un variable s égale à 4. Vous pourrez changer par la suite cette valeur pour accélérer ou ralentir la propagation de la maladie.

Tirez selon une loi normale centrée réduite les valeurs d'une matrice nommée `nouvelle_ville_contamine_valeur`. Cette matrice aura la même taille que la matrice `d_villes`. Elle va nous permettre de choisir les villes nouvellement contaminées en fonction de leurs distances aux villes contaminées.

Multipliez les valeurs de la matrice `nouvelle_ville_contamine_valeur` par s pour fixer la vitesse de propagation.

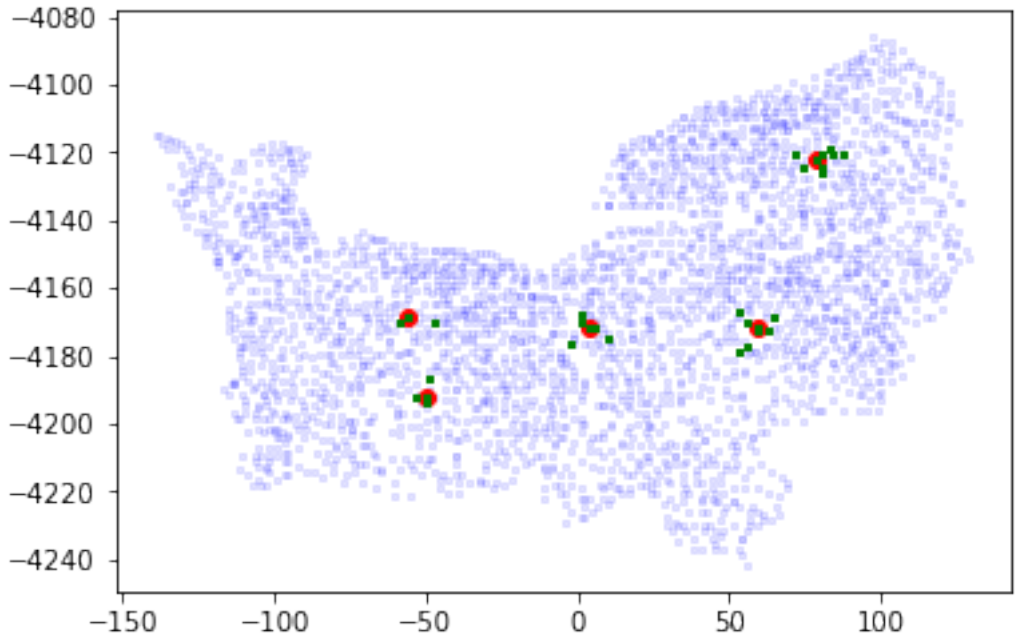
Construisez une nouvelle matrice `nouvelle_ville_contamine` qui contient True pour les valeurs dans `nouvelle_ville_contamine_valeur` dont la valeur absolue est plus grand que les valeurs dans `d_villes` et False pour les autres. Cette matrice permet de connaître la contamination potentielle des villes. Les villes en colonne indiquent si la ville serait contaminée dans l'hypothèse où la ville indiquée par la ligne est déjà contaminée.

Multipliez la matrice précédente par la matrice `ville_src` et sommez les valeurs suivant les colonnes. Numpy remplacera les valeurs True par 1 et False par 0 pour faire le calcul. Cela produira un vecteur qui compte pour chaque ville le nombre de villes qui sont sources de sa contamination. Si la valeur est 0,la ville n'est pas encore contaminée.

Testez si les valeurs du vecteur précédent sont supérieures strictement à 0. Si c'est le cas, c'est-à-dire que la ville est contaminée.

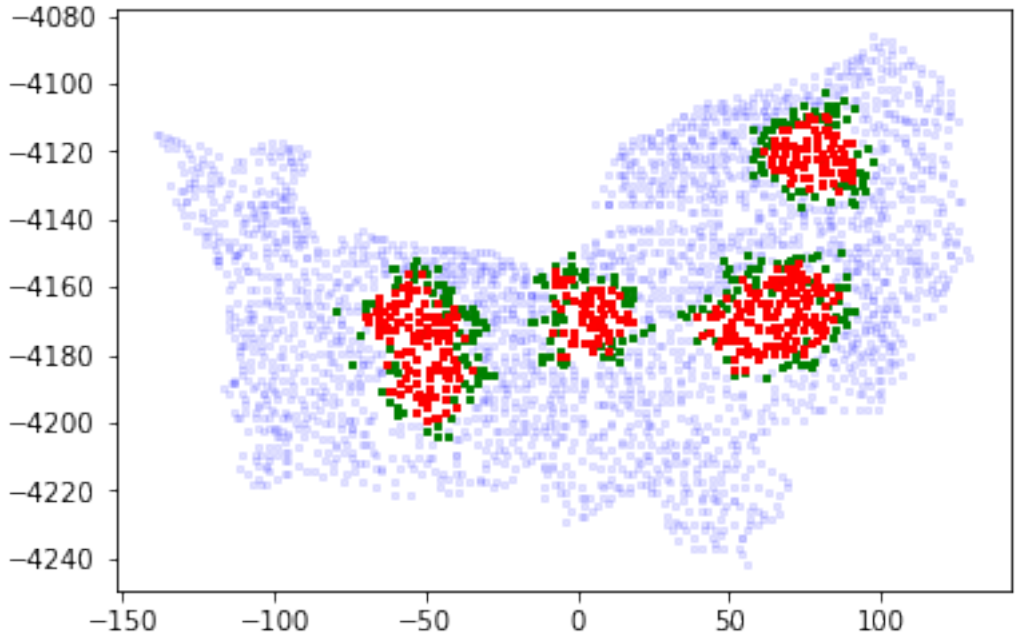
Reprenez la figure de la question précédente et affichez-y les villes nouvellement contaminées en vert. Que remarquez-vous sur les villes contaminées au début de l'épidémie ?

In [15]:



À l'aide d'une boucle `for` réitérer plusieurs fois les opérations précédentes pour voir la propagation de la maladie au bout d'un certain temps. À chaque itération, on prendra comme `ville_src`, toutes les villes précédemment contaminées.

```
In [16]: for i in range(5):
             ...
```



1.4 Création d'une animation (partie bonus)

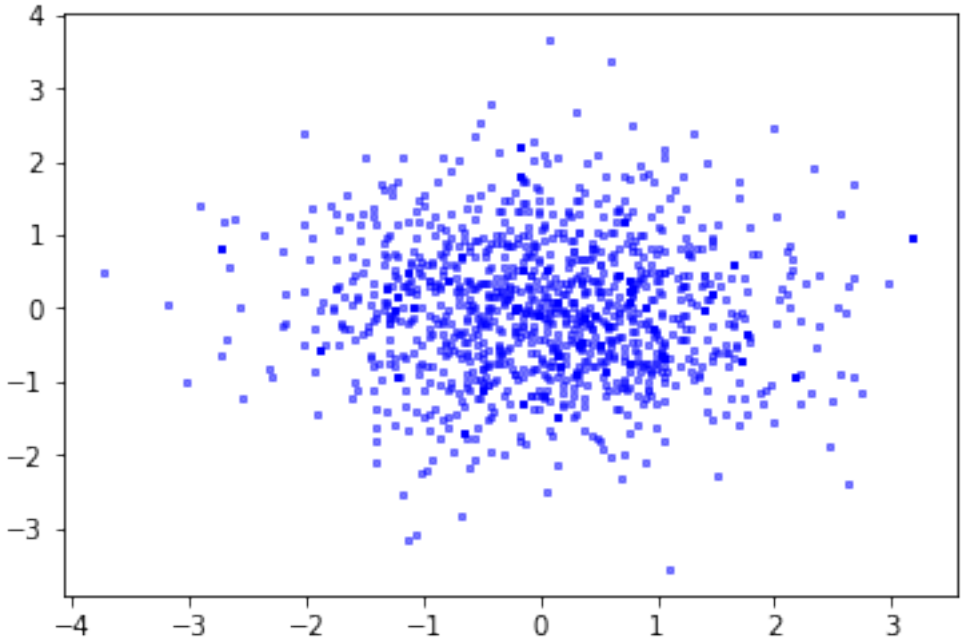
Étudiez le code suivant. Que fait ce code ?

```
In [17]: import matplotlib.animation as animation

         points = np.zeros((0,2))

         fig = plt.figure()
         def updatefig(i):
                 global points
                 fig.clear()
                 plt.scatter(points[:,0],points[:,1],color='b',marker='s',s=5,alpha=0.5)
                 x = np.random.randn(50,2)
                 plt.scatter(x[:,0],x[:,1],color='b',marker='s',s=5)
                 plt.draw()
                 points = np.concatenate([points,x],axis=0)

         anim = animation.FuncAnimation(fig, updatefig, 20)
         anim.save("test.mp4", fps=5)
```



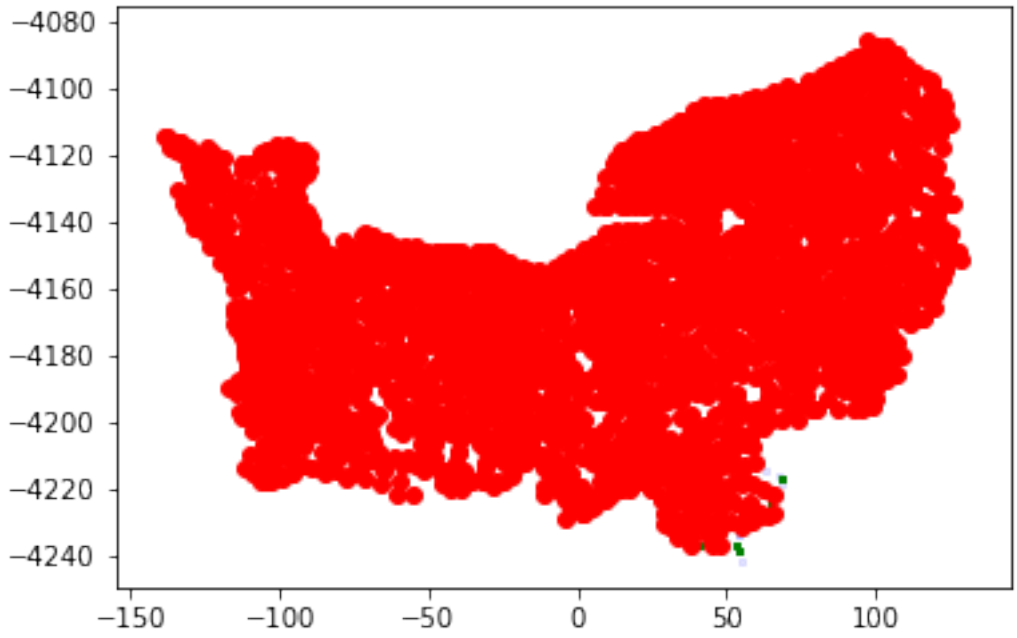

```
In [18]: %%HTML
         <video width="320" height="240" controls>
```

```
  <source src="test.mp4" type="video/mp4">
</video>
```

```
<IPython.core.display.HTML object>
```

Ces codes permettent ... Reprenez le code de la partie précédente et au lieu d'afficher le résultat à l'écran, faites une petite animation de la propagation de la maladie.

In [19]:



In [20]:

```
<IPython.core.display.HTML object>
```