La fouloscopie au service de l'épidémiologie.

I- Introduction

II- Approche théorique

1. Epidémiologie
2. Fouloscopie

III- Mise en pratique.

1. Objectif
2. Les différents programmes

IV- Application

1. Mise en application de nos programmes
2. D'autres exemples

V- Conclusion

VI- Annexe/Bibliographie

I – Introduction

A l'image de la crise sanitaire que nous traversons depuis maintenant plus d'un an, le thème de cette année, Santé et Prévention, nous a orienté sur le thème de l'épidémiologie qui est aujourd'hui au cœur de l'actualité.

Nous nous sommes d'abord demandé comment cette science fonctionnait, et comment elle évoluait. C'est alors que nous avons pensé à la Fouloscopie, un terme nouveau, mais qui décrit en réalité une science qui existe depuis bien plus longtemps : l'étude des foules.
La question qui se pose tout naturellement est donc : Comment la fouloscopie sert-elle l'épidémiologie ?

Les objectifs de notre TIPE sont de répondre dans un premier temps à cette question, puis dans un second temps de créer notre propre simulation d'épidémie qui soit la plus réaliste possible, grâce aux données apportées par la fouloscopie.

Afin de réaliser notre projet, nous nous sommes réparti les tâches : Théo s'est occupé de la partie épidémiologie, Hugo s'est occupé de la partie fouloscopie, et Julien s'est occupé de la partie expérimentale, en commençant à programmer des simulations d'épidémies.

(Toutes les démarches scientifiques effectuées dans ce dossier sont tirés d'article, de documentation ou de codes sources libres de droit – ceux-ci sont trouvables dans le VI en annexe/bibliographie)

II – Approche théorique.

1) Epidémiologie.

A. Définition.

L'épidémiologie, comme vous le savez, est la discipline scientifique qui étudie les épidémies, c'est-à-dire, le comportement et l'évolution d'une maladie au sein d'une population.

Cette discipline n'est toutefois pas si âgée que ça, puisque la première véritable étude épidémiologique a été réalisée par John Snow en 1854, sur l'épidémie de choléra de Broad Street, un des épisodes de la troisième pandémie de choléra. (cf. VI – Bibliographie – 1)

L'épidémiologie a, depuis ce temps, bien évolué, pour aujourd'hui s'appliquer dans différents domaines tels que :

- L'étude de terrain, qui, comme son nom l'indique vise à étudier le comportement d'une maladie directement sur le terrain, au sein de la population contaminée afin d'avoir des données brutes.

- La recherche, qui vise à lutter contre l'émergence d'épidémies, grâce à des vaccins par exemple.

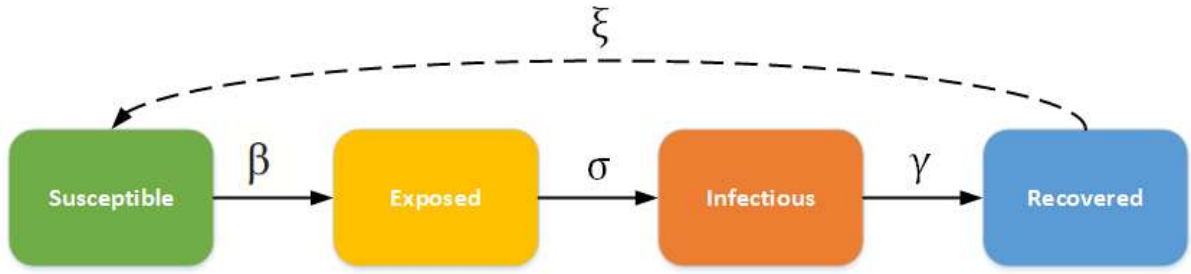
- Les simulations, qui visent à prédire à l'avance l'évolution d'une épidémie en train de se propager afin d'adapter au mieux les mesures sanitaires pour lutter contre cette épidémie.

C'est sur ce dernier domaine que nous allons nous concentrer au cours de ce dossier, car notre article scientifique porte sur cet aspect de l'épidémiologie.

B. Comment simuler une épidémie ?

En effet, l'article (cf. VI – Annexe – Epidémiologie), qui s'intitule ''SEIR and SEIRS models'' présente et explique en détails les modèles de simulation du même nom.

Ces modèles se présentent sous la forme suivante :



Chaque lettre a une signification :

- S : Susceptible : La personne est saine, et non immunisée, donc susceptible d'être infectée.
- E : Exposed : La personne a été exposée et contaminée par l'agent infectieux, mais n'est néanmoins pas encore infectieuse.
- I : Infectious : La personne est infectée depuis un certain moment, et est désormais infectieuse, c'est-à-dire qu'elle peut contaminer d'autres personnes.
- R : Recovered : La personne n'est plus contaminée et a gagné une immunité contre l'agent contagieux.
- S (Seulement pour le modèle SEIRS) : Susceptible : La personne est revenue à l'état initial, c'est-à-dire saine, et non immunisée.

Le modèle SEIRS est donc légèrement différent du modèle SEIR, dans la mesure où il prend en compte la possibilité que l'on puisse perdre notre immunité, et donc qu'une même personne puisse être contaminée plusieurs fois.

Chaque lettre représentant donc un stade de la contamination, on va les considérer pour la simulation en tant que variables auxquelles on va associer un certain nombre de personnes qui va changer en fonction des jours. Par exemple, au 1er jour, toute la population est au stade ''Suspicious'', puis au second jour, on aura les premières personnes au stade ''Exposed'', etc…

Le nombre de personnes associé à chaque variable dépendra ainsi de taux que l'on fixera :

S → E : taux d'infection (noté β)

C'est le taux qui indique la probabilité qu'une personne ''Infectious'' contamine une personne ''Susceptible''.

E → I : taux d'incubation (noté σ)

C'est le taux qui indique combien en moyenne de personnes deviennent infectieuses après avoir été infectée, basé donc de la durée moyenne d'incubation (qui vaut $\frac{1}{\sigma}$).

I → R : taux de récupération (noté ϒ)

C'est le taux qui indique combien en moyenne de personnes vont guérir chaque jour, basé donc sur la durée moyenne d'infection D. (On a $\Upsilon = \frac{1}{D}$).

R → S : taux de perte d'immunité (noté ξ)

C'est le taux qui indique combien de personnes en moyenne reviennent au stade de ''Susceptible'', basé donc sur la durée moyenne d'immunité.

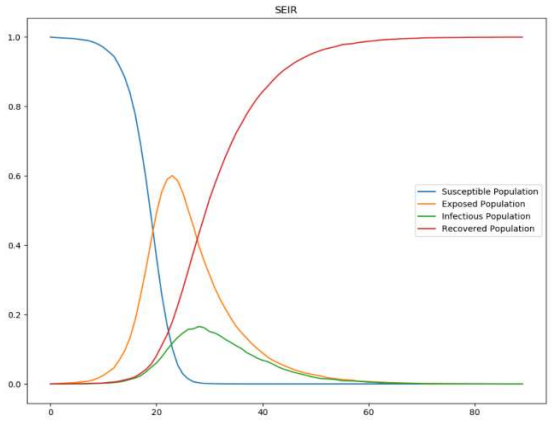
De ces différents taux, on peut traduire le schéma précédent par le système d'équations différentielles suivant :

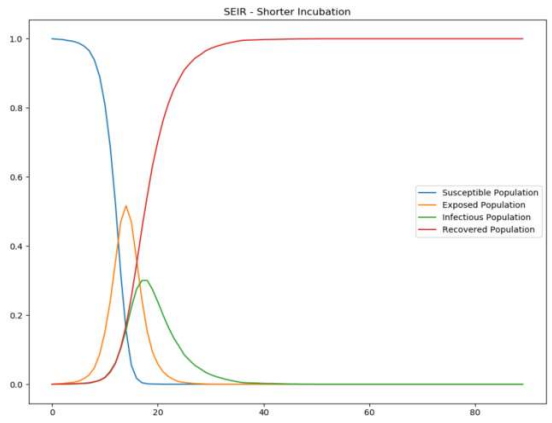
$$\frac{dS}{dt} = -\frac{\beta SI}{N} + \xi R$$
$$\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E$$
$$\frac{dI}{dt} = \sigma E - \gamma I$$
$$\frac{dR}{dt} = \gamma I - \xi R$$

avec $N = S + E + I + R$

(Actuellement nous ne savons pas résoudre ce système, nous nous pencherons plus en détails sur celui-ci l'année prochaine.)

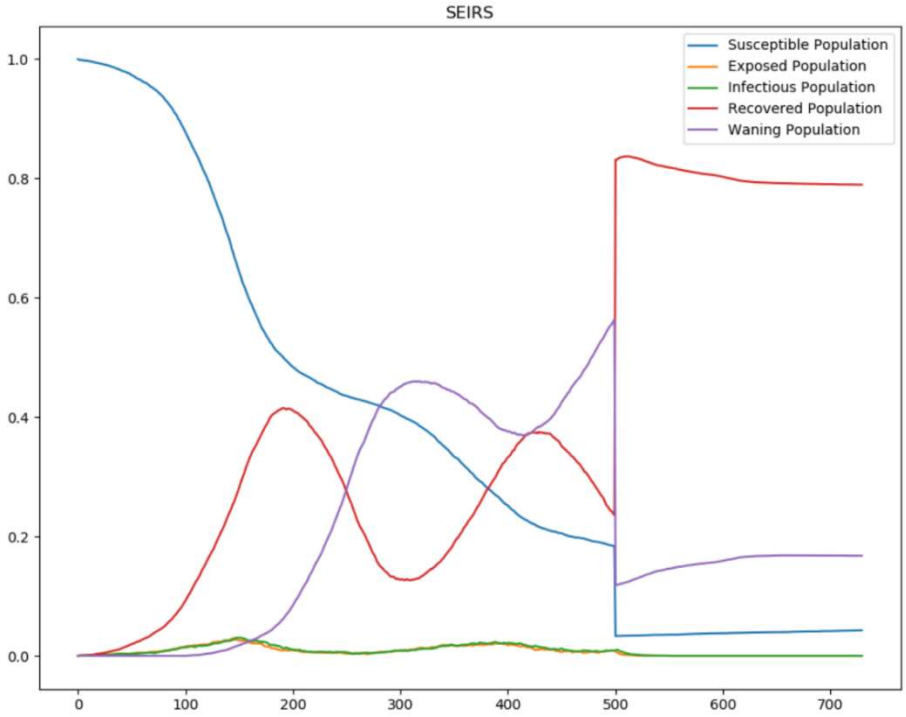
Voici deux simulations informatiques suivant le modèle SEIR dans des conditions légèrement différentes : La première a une période d'incubation de 8 jours, la seconde a une période d'incubation de 2 jours :





On remarque donc que le modèle SEIR ne dépend pas du nombre de jours, il se comporte toujours de la même façon.

Voici maintenant une simulation suivant le modèle SEIRS, et qui est programmée pour éradiquer l'épidémie peu importe ce qu'il se passe au jour 500 :



On remarque cette fois ci que les courbes évoluent complètement différemment par rapport au modèle SEIR.

On remarque toutefois un point commun : Les deux modèles partent du principe que la maladie n'est pas létale, et ne prennent donc pas en compte le nombre de morts dans leurs simulations.

C. Application à notre sujet.

Nous avons donc décidé de faire notre propre simulation d'épidémie, basée en grande partie sur le modèle SEIRS, qui est plus intéressant de par le fait qu'il prend en compte le fait qu'on puisse perdre notre immunité, mais cette fois ci, en tenant compte aussi du nombre de morts.

Ce qui fait que nous devons rajouter une variable de plus à notre modèle : La variable M, pour le nombre de mort, mais aussi lui associer un taux (noté τ), qui est logiquement le taux de létalité de la maladie. (Note : Nous avons aussi changé le nom de la variable R en T, pour ''Temporairement immunisé'' pour plus de clarté)

Pour réaliser cette simulation, nous avons donc utilisé les suites et les matrices, en posant que chaque variable était une suite dont l'indice représente le jour de la simulation, (Exemple : S1 est le nombre de personnes ''Susceptibles'' au jour 1), et nous avons utilisé une matrice pour la récurrence au jour n+1 pour chaque suite :

$$\begin{pmatrix} 1-e & 0 & 0 & d & 0 \\ e & 1-f & 0 & 0 & 0 \\ 0 & f & 1-c-g & 0 & 0 \\ 0 & 0 & g & 1-d & 0 \\ 0 & 0 & c & 0 & 1 \end{pmatrix}$$

```
c=eval(input("taux de mortalité : "))
d=eval(input("taux de perte d'immunitée : "))
e=eval(input("taux d'infection : "))
f=eval(input("taux d'incubation : "))
g=eval(input("taux de récuperation : "))
```

En Français, cette matrice donnerait : Au jour n+1 :

Le nombre de personnes saines = (le nombre de personnes saines au jour n – le nombre de personnes qui ont été infectées) + (le nombre de personnes qui ont perdu leur immunité)

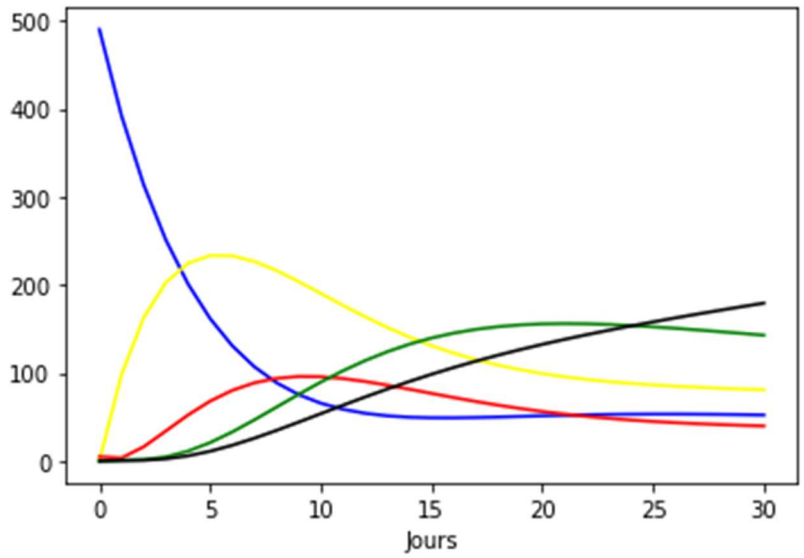
Le nombre de personnes infectées = (le nombre de personnes infectées au jour n + le nombre de personnes qui ont été infectées) - (le nombre de personnes qui sont devenu infectieuses)

Le nombre de personnes infectieuses = (le nombre de personnes infectieuses au jour n + le nombre de personnes qui sont devenu infectieuses) – (le nombre de personnes qui ont guéri et acquis une immunité temporaire) – (le nombre de personnes mortes)

Le nombre de personnes temporairement immunisées = (le nombre de personnes immunisées au jour n + le nombre de personnes qui ont guéri et acquis une immunité temporaire) – (le nombre de personnes qui ont perdu leur immunité)

Le nombre de personnes mortes = (le nombre de personnes mortes au jour précédant + le nombre de personnes qui sont mortes au jour n)

En réalisant notre programme sur Python, nous avons ainsi obtenu les courbes suivantes :



- Bleu : population saine
- Jaune : population infectée mais non infectieuse
- Vert : population temporairement immunisée
- Rouge : population infectieuse
- Noire : population décédée

Nous les analyserons en détails plus tard dans le dossier.

2) Fouloscopie

A. Qu'est-ce que la fouloscopie ?

Littéralement, la fouloscopie est l'étude des foules mais plus précisément cela se rapporte au comportement qu'adopte une foule dans certaines situations.

Cette science du comportement des foules se divise en trois axes majeurs :

Le déplacement collectif : on s'intéresse au mouvement de la foule, par exemple aux mouvements des piétons dans la rue ou encore celui des supporters dans un stade. Ce domaine se focalise plutôt sur la manière dont vont s'organiser les individus face à une situation.

La contagion sociale : on cherche à comprendre comment une information circule au sein d'un groupe d'individu et comment celle-ci peut être plus ou moins déformer. Par exemple, on peut parler des « fake news », et comment celles-ci évoluent, se transmettent, etc …

L'intelligence collective : ici, il s'agit plus du potentiel de la foule par rapport à l'individu seul. En général, ce domaine est appelé « science participative » ; on peut prendre l'exemple de l'encyclopédie en ligne « Wikipédia » ou encore de la célèbre partie d'échec opposant Kasparov à la foule en 1997.

Le domaine qui nous intéresse le plus est celui de la contagion sociale car c'est celui qui se rapproche le plus de l'épidémiologie.

B. Comment simuler le comportement humain ?

Le comportement humain peut se résumer par son interaction avec les autres, en effet un individu seul ne nous intéresse pas pour ce qui est de véhiculer une information ou un virus par exemple. Néanmoins, le mouvement solitaire et/ou collectif d'individus peut éveiller notre curiosité car il peut être un facteur important pour développer l'interaction entre individus. Finalement, ce qu'il nous faut se sont des outils mathématiques pour simuler le déplacement d'individus, ainsi que pour simuler les interactions qu'ils peuvent avoir entre eux.

(Les informations qui suivent sont tirées d'un article de l'Université de Washington, « Continuum Crowds » cf. VI – Annexe – Fouloscopie)

Pour ce qui est du mouvement, on peut formuler plusieurs hypothèses :

Hypothèse n°1 : Chaque personne veut rejoindre une destination géographique finale. De plus on peut mentionner le fait que l'environnement joue un rôle très important, un piéton qui se balade ne va pas adopter le même comportement et la même allure de marche qu'une armée sur un champ de bataille.

Hypothèse n°2 : Chaque personne se déplace à une vitesse maximale. Ceci peut être traduit comme un champ de vitesse f où une personne situé à la position x va dans une direction θ :

$$\dot{x} = f(x, \theta) n_\theta$$

Où $n_\theta = [\cos\theta, \sin\theta]^T$ est le vecteur unitaire pointant dans la direction θ. Cependant, les individus peuvent avoir des préférences quant au chemin à prendre d'où l'hypothèse qui suit.

Hypothèse n°3 : il existe un champ d'inconfort g, où en tout point de l'espace tout est égal mais un individu peut préférer la position x à la position x' si $g(x') > g(x)$. En général, les individus choisissent le chemin avec le minimum de distance même si parfois le sentiment de vouloir éviter l'attroupement ou d'autres situations amenant à de la perte de temps intervient. On peut donc résumer ce comportement par trois points : la longueur du chemin, le temps qui faut pour atteindre la destination, le champ d'inconfort suivant le chemin.

Hypothèse n°4 : Cette tendance à vouloir optimiser les trois points précédemment évoqués peut se traduire par :

$$\min\left(\underbrace{\alpha \int_P 1 ds}_{\text{Longueur du chemin}} + \underbrace{\beta \int_P 1 dt}_{\text{Temps}} + \underbrace{\gamma \int_P g dt}_{\text{Inconfort}}\right)$$

Ou α, β, et γ sont des coefficient permettant de faire plus ou moins influer un paramètre ; ds indique que l'intégrale dépend de la longueur du trajet, alors que dt veut dire que l'intégrale dépend du temps. Or on sait que $v = \frac{d}{t}$ donc $f = \frac{ds}{dt}$, ainsi on peut réécrire :

$$\min\left(\alpha \int_P 1ds + \beta \int_P \frac{1}{f} ds + \gamma \int_P \frac{g}{f} ds\right) = \min\left(\int_P C ds\right) \text{ avec } C = \frac{\alpha f + \beta + \gamma g}{f}$$

Où C est appelé champ des coûts unitaires.

a) Optimisation du chemin.

On peut considérer l'application linéaire où la fonction $\phi : \mathbb{R}^2 \to \mathbb{R}$ est équivalente au coût du chemin optimal pour se rendre à une destination prévue. Si on représente par un gradient cette fonction, intuitivement la stratégie la plus optimale est de remonter se gradient ce qui va diminuer le coût optimal pour se déplacer. Si on considère qu'on recherche $\phi = 0$ alors on se retrouve avec une équation eikonale (à l'instar de celle qui décrit le mouvement des rayons lumineux) :

$$\|\nabla\phi(x)\| = C$$

Où le coût C est évalué par rapport à la norme du gradient $\nabla\phi$. Si on relie cette équation à celle de l'hypothèse n°2, grâce au théorème de calcul de la variations :

$$\dot{x} = -f(x, \theta) \frac{\nabla\phi(x)}{\|\nabla\phi(x)\|}$$

On calcul un champ potentiel pour ensuite trouver un chemin optimal. Pour illustrer, on peut prendre pour exemple un groupe de personnes qui ont tous le même champ de vitesse, le même inconfort et le même but à atteindre (par exemple quand une foule est composé d'individu voulant aller au même endroit, donc à une vitesse a peu près équivalente). On doit calculer la fonction qui définit le groupe puis on dérive pour pouvoir l'appliquer à tous les individus simultanément. En réalité, chaque personne va à une vitesse différente, a un inconfort qui varie, et des objectifs différents. Cependant une foule peut être divisé en plusieurs petits groupes qui suivent la même tendance. Donc à chaque unité de temps qui passe on construit une fonction potentielle ϕ pour chaque groupe, qui chacun dépendent de la vitesse f et de l'inconfort g, mais qui au final sont tous reliés. Cette méthode de simulation est la plus basique mais aussi la plus lente donc on préfère avoir une foule homogène, ayant le moins de groupes possibles.

b) La vitesse.

Le champ de vitesse f mesure la vitesse maximale pour chaque point dans toutes les directions. La vitesse est une variable qui dépend de la densité donc d'un champ de densité ρ. On convertit chaque personne en un champ de densité ρ_i. La vitesse et la direction moyenne de la foule peut donc être déduit d'un champ de vitesse moyenne $\bar{v}$ exprimé en fonction du champ de densité moyen $\bar{\rho}$ équivalent à un disque de rayon r :

$$\rho = \sum_i \rho_i \ , \quad \text{et } \bar{v} = \frac{\sum_i \rho_i \dot{x_i}}{\rho}$$

On peut maintenant exprimé la vitesse de l'hypothèse n°1 en fonction de la densité sur plusieurs niveaux :

Si la densité est très faible ($\rho < \rho_{min}$), alors la vitesse f dépend du terrain (c'est donc une vitesse topologique notée f_T) où les pentes maximale et minimale sont s_{max} et s_{min} :

$$f_T(x,\theta) = f_{max} + \left(\frac{\nabla h(x).n_\theta - s_{min}}{s_{max} - s_{min}}\right)(f_{min} - f_{max})$$

Si la densité est très élevée ($\rho > \rho_{max}$), alors la vitesse f est égale à la vitesse moyenne $f_{\bar{v}}$:

$$f_{\bar{v}}(x,\theta) = \bar{v}(x + rn_\theta).n_\theta$$

Et enfin si la densité est modéré ($\rho_{min} < \rho < \rho_{max}$), on interpole entre les deux équations précédentes :

$$f(x,\theta) = f_T(x,\theta) + \left(\frac{\rho(x + rn_\theta) - \rho_{min}}{\rho_{max} - \rho_{min}}\right)\left(f_{\bar{v}}(x,\theta) - f_T(x,\theta)\right)$$

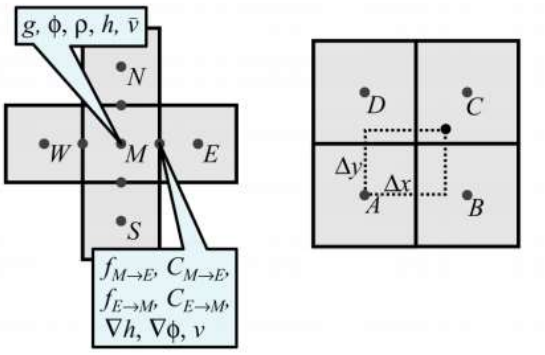
c) L'implémentation

A chaque itération :

- On traduit la foule en un champ de densité
  Pour chaque groupe :
  - On construit le champ de coût unitaire C
  - On définit la fonction ϕ et son gradient $\nabla\phi$
  - On met à jour la position des individus
- On impose une distance minimum entre les individus

Pour traduire cela, on réduit l'espace considéré en une grille symbolisant les positions probables. Tous les champs scalaires sont définis au centre des cellules de la grille, ; la vitesse moyenne $\bar{v}$ est traduit par deux valeurs ; la direction θ peut prendre quatre valeurs correspondants aux directions possibles pour passer d'une cellule à l'autre, soit $\theta = \{0°, 90°, 180°, 270°\}$ donc est, nord, ouest et sud ; enfin, la vélocité v, le gradient ∇h et $\nabla\phi$ sont enregistrés sur les faces de chaque cellule.

Graphiquement, cela donne ceci :



d) L'interaction.

La simulation de l'échange d'information entre plusieurs individus dans notre cas sera régit d'après notre modèle épidémiologique. Sinon, pour illustrer on peut aisément modéliser l'interaction entre individus dans le cas du troc. Dans les pages suivantes on verra que ceci se fait très simplement grâce au langage informatique, notamment via l'utilisation des « class » et grâce aux « agent-based simulation ».

III – Mise en pratique

A) Objectif :

L'objectif étant de pouvoir anticiper et mieux réagir à une épidémie, la meilleur solution reste l'outil informatique.
Nous avons donc pris la décision de réaliser 3 programmes permettant de mieux comprendre le comportement d'une épidémie dans une foule afin de pouvoir anticiper et mieux réagir face à une futur épidémie

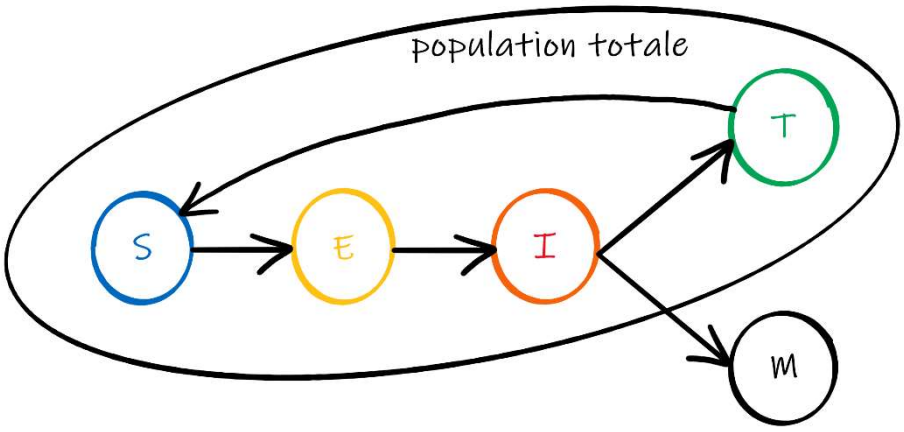
B) Les différents programmes :

1° **premier programme** : étude statistique d'une épidémie

a) Algorithme :

Notre premier algorithme se base uniquement sur l'étude du système SEIRS, nous avons ainsi commencé par réaliser un schéma qui dirigera la création des algorithmes.

Voici notre schéma :



Ce schéma présente les différents états d'un individu au cours d'une épidémie :
à l'instar du système SEIRS nous avons choisi de rajouter l'état M qui représente les personnes décédées à cause de l'épidémie.
Enfin chaque passage d'un état à l'autre est représenté par des flèches et calculé à l'aide de taux.

Pour le calcul du nombre d'individus dans chaque état pour un jour donné, nous avons réalisé une suite matricielle : $\begin{pmatrix} S \\ E \\ I \\ T \\ M \end{pmatrix}_n = \begin{pmatrix} 1-e & 0 & 0 & d & 0 \\ e & 1-f & 0 & 0 & 0 \\ 0 & f & 1-c-g & 0 & 0 \\ 0 & 0 & g & 1-d & 0 \\ 0 & 0 & c & 0 & 1 \end{pmatrix}^n * \begin{pmatrix} So \\ Eo \\ Io \\ To \\ Mo \end{pmatrix}$

(cf. II – 1)

Pour retranscrire cette suite sur python , nous avons commencé par créer des variables qui seront entrées par l'utilisateur :

```
#Calcul de la suite complète:
def u() :

    a=eval(input("nombre de personnes infectées : "))
    b=eval(input("nombre de personnes total : "))
    c=eval(input("taux de mortalité : "))
    d=eval(input("taux de perte d'immunitée : "))
    e=eval(input("taux d'infection : "))
    f=eval(input("taux d'incubation : "))
    g=eval(input("taux de récuperation : "))
    nbrj=eval(input("nombre de jours : "))

    #Matrice de changement d'état :
    m=M(a,b,c,d,e,f,g)

    #Matrice au jour 0:
    m1=M1(a,b,c,d,e,f,g)

    #Création liste vide
    A=[]

    # calcul de tout les termes jusqu'au jour n
    for i in range (0,nbrj):
        un=n(i,m,m1)
        A.append([un])

    #Retourne la liste des matrices SEITM
    return(A)
```

Ces différentes variables correspondent aux taux impliqués dans le cycle d'une épidémie comme vue précédemment, mais aussi aux valeurs initiales correspondants aux nombres de personnes totales et infectés au jour 0.

Ces variables sont ensuite transmises dans les matices m1 et m correspondant respectivement aux matrices $SEITM_0$ et à la matrice de passage.

```
#Calcul de la matrice M1
def M1(a,b,c,d,e,f,g):
    M1=[[b-a],[0],[a],[0],[0]]
    return(M1)
```

```
#Réalisation de la matrice M
def M(a,b,c,d,e,f,g):
    M=[[1-e,0,0,d,0],
       [e,1-f,0,0,0],
       [0,f,1-c-g,0,0],
       [0,0,g,1-d,0],
       [0,0,c,0,1]]
    return(M)
```

Enfin la dernière partie du programme va calculer tous les termes de la suite et les placer dans une liste A créée au préalable

```
#Calcul du terme n
def n(j,M,M1):
    for i in range (0,j):
        M1=np.dot(M,M1)
    return M1
```

A l'issue de ce premier programme on obtient une liste de la forme suivante :

A
(SEITM)0
(SEITM)1
(SEITM)2
(SEITM)3
...
(SEITM)n

L'objectif maintenant est de séparer les suites S,E,I,T et M pour les representer graphiquement

On a donc créer deux sous programmes permettant de séparer les suites, l'un va **récuperer** les valeurs d'une des suites S ,E , I, T ou M et l'autre va utiliser plusieurs fois le programme de récupération pour separer les suites.

```
#Récupère la valeur de la
#sous suite (sous liste) concernée
def recup(A,n):
    a=len(A)
    S=[None]*(a)
    for i in range (0,a):
        S[i]=A[i][0][n][0]
    return S
```

```
#création de la sous suite complète
def ListeS(A):
    b=len(A[0][0])
    S=[None]*b
    for i in range(0,b):
        S[i]=recup(A,i)
    return S
```

Ces programmes nous permettent de faire passer la liste A de la forme :

A
(SEITM)0
(SEITM)1
(SEITM)2
(SEITM)3
...
(SEITM)n

A la forme suivante :

A				
S0	E0	I0	T0	M0
S1	E1	I1	T1	M1
S2	E2	I2	T2	M2
S3	E3	I3	T3	M3
…	…	…	…	…
Sn	En	In	Tn	Mn

Enfin on a réalisé un dernier programme pour tracer les courbes du graphique :

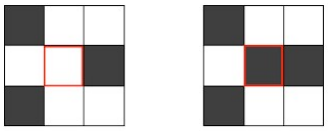
```
#tracer d'une courbe à partir d'une liste
def trace(P):
    xmax = len(P[0])

    S = P[0]
    E = P[1]
    I = P[2]
    T = P[3]
    M = P[4]
    x = [None]*xmax
    for i in range (0,xmax):
        x[i] = i

    plt.plot(x,S,color='blue')
    plt.plot(x,E,color='yellow')
    plt.plot(x,I,color='red')
    plt.plot(x,T,color='green')
    plt.plot(x,M,color='black')


    plt.xlabel ('Jours')
```

2° **deuxième programme** : simulation de propagation d'un agent infectieux dans une foule statique.

- Objectifs :

Ce second programme ainsi que le troisième sont des projets pour l'année prochaine et ne sont donc pas encore réalisé, cependant nous allons expliquer leurs fonctionnement et nos attentes.
Ce programme va donc fonctionner de la même manière que le jeu de la vie.
Nous allons donc rappeler ce qu'est le jeu de la vie :

Le jeu de la vie est donc un algorithme « cellulaire » , c'est-à-dire un algorithme qui va appliquer des « règles » aux cellules d'une grille, créer par John Horton Conway en 1970.
Ainsi le jeu de la vie applique certaines règles aux cellules les faisant passer de l'état cellule vivante à l'état cellule morte.



T T+1

Notre second programme va donc se baser sur ce type d'algorithme pour comprendre la diffusion d'une épidémie dans une foule statique.
En effet nos cellules pourront appartenir à 5 états différents correspondant à notre système SEITM
Ainsi les cellules infectées auront un impact sur les cellules vivantes et auront un pourcentage de chances de devenir des cellules mortes. Nous pourrons donc observer la propagation de l'agent infectieux et nous pourrons appliqués des contraintes telle que la distanciation sociale ou le port du masque sur certaines cellules pour voir l'impact sur la propagation.

3° **troisième programme** : simulation de propagation d'un agent infectieux dans une foule en mouvement.

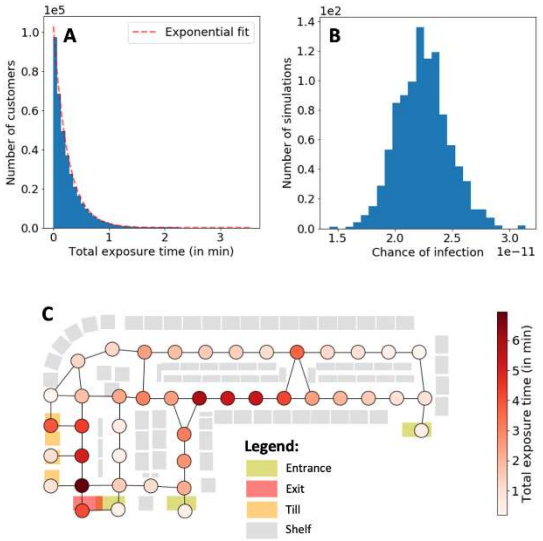
- Objectifs

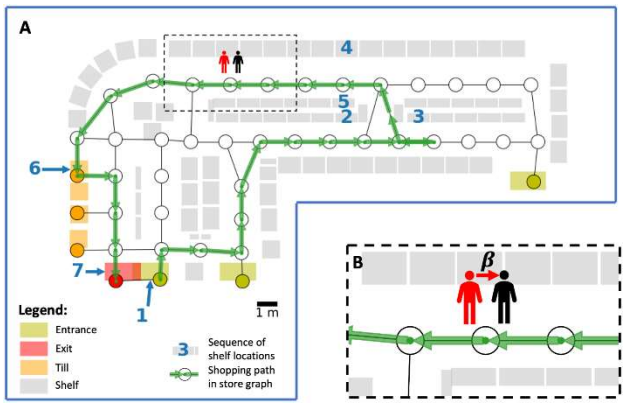
Ce programme, tout comme le précèdent est un projet et sera réalisé l'année prochaine

Ce dernier algorithme sera également un algorithme cellulaire, Celui-ci, contrairement au dernier, permettras aux cellules de se déplacer, Ainsi il nous permettra de pouvoir simuler plusieurs situations telle qu'un supermarché, une rue ou une fête par exemple.

Ainsi en faisant changer différents facteurs tels que les paramètres initiaux, ou en appliquant certaines règles que les cellules devront suivre telles que : Un sens de circulation, des points d'intérêts ou en appliquant des règles sanitaires.

De ce fait cette algorithme sera bien plus visuel et factuel car il est celui qui prends le plus en compte la réalité





Ces photos issues de l'algorithme de Fabian Ying et de Neave O'Clery, nous permettent de vous représenter le type de rendu que nous souhaiterions atteindre.

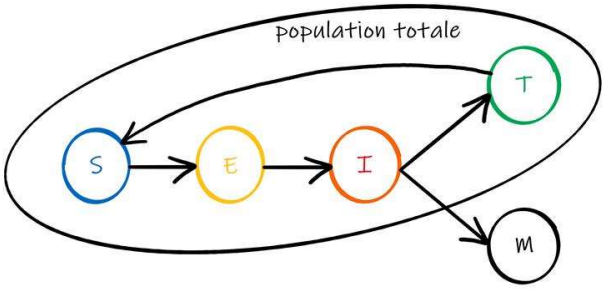
Nous devrons mettre en place des réseaux que nous ne maitrisons pas présentement c'est pourquoi on le réalisera l'année prochaine

IV – Application

1) Mise en application de nos programmes.

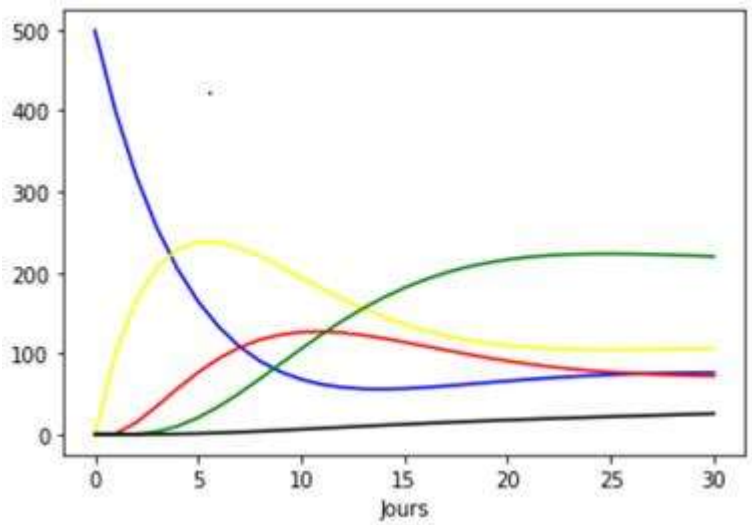
A. Epidémiologie.

Pour rappel, dans le domaine de l'épidémiologie nous nous basons sur notre propre modèle (ci-dessous) (cf. II – 1):



Grâce au premier programme que nous avons réalisé (cf. III – B), nous obtenons le graphique suivant :

Etude statistique d'une population infectée sur 31 jours.



Paramètres d'initialisation :

- Population totale : 500
- Nombre de personnes infectées : 1
- Taux de mortalité : 0,01
- Taux de perte d'immunité : 0,001
- Taux d'infection : 0,2
- Taux d'incubation : 0,14
- Taux de récupération : 0,2
- Nombre de jours : 31

Paramètres de sortie :

- Bleu : population saine
- Jaune : population exposée
- Vert : population immunisée temporairement
- Rouge : population infectée
- Noire : population décédée

(Il faut repréciser que cet algorithme est une représentation statistique donc il n'est pas représentatif de la réalité sur le long terme)

L'évolution dessinée par ces courbes nous permet de visualiser l'évolution d'un agent infectieux quelconque sur une durée de 31 jours soit 1 mois :

Tout d'abord, on constate qu'une épidémie tend en général à ce stabiliser (avant de subir un rebond dans la plupart des cas). On observe que dans le cas d'une immunité innée, la population tant à devenir immunisée (courbe verte) plutôt rapidement après la forte montée exponentielle de l'épidémie (courbe rouge) jusqu'à avoir une inversion des courbes entre les exposés et les immunisés (courbes jaunes et vertes), traduisant alors le ralentissement de l'épidémie.

Bien sûr, ici la population saine est celle d'origine donc non immunisée (courbe bleue), c'est-à-dire que une fois qu'un individu est ou a été contaminé il ne peut plus être considéré comme sain ce qui traduit le caractère décroissant de la courbe. La légère croissance de fin peut se traduire par la partie de la population qui a perdu son immunité temporaire.

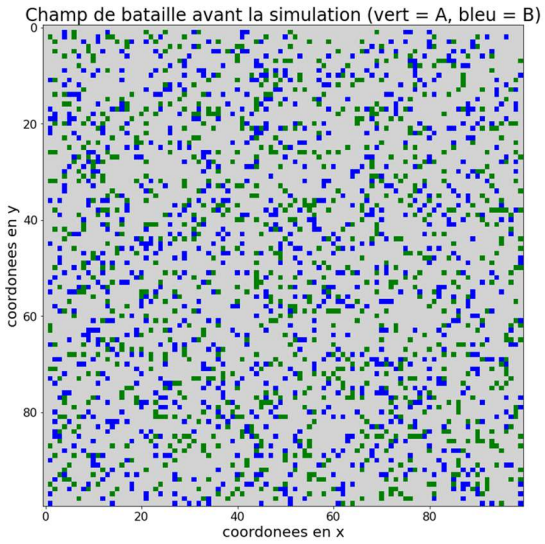
La population infectée présente elle aussi un léger saut avant de se stabiliser, on peut penser que pour certaine épidémie ce saut constitue les rebonds possibles de celle-ci sur des durées plus espacées. Par exemple, le cas de la grippe : les épidémie de grippes sont cycliques, tous les ans il y a des vagues de malades, on peut donc parler d'un rebond annuel pour ce qui est de cette épidémie.

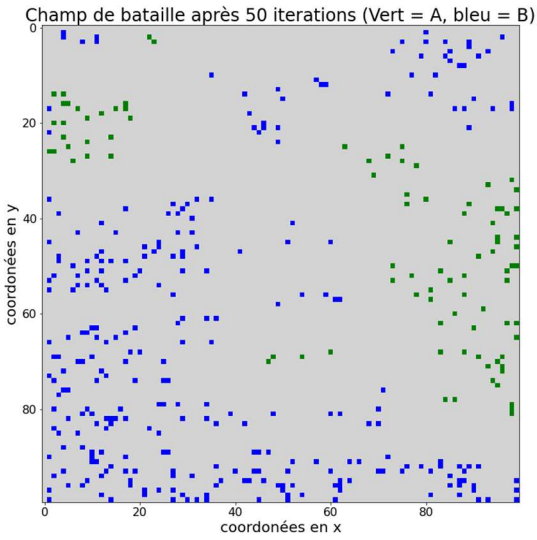
Enfin, la courbe noire représente les décès au cours du temps donc logiquement elle ne peut être que croissante.

B. Fouloscopie.

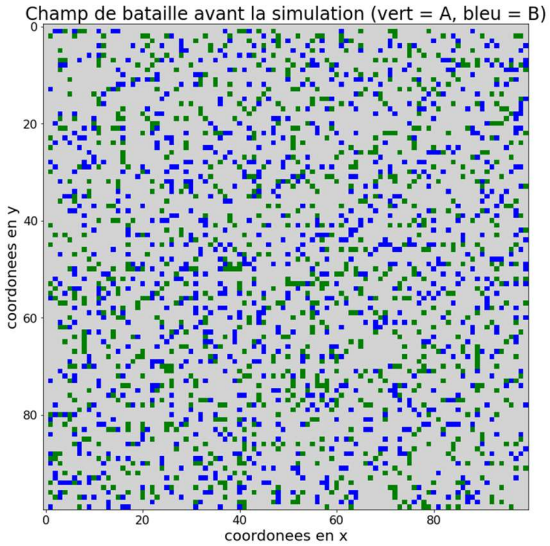
Pour ce qui est de la fouloscopie, les principes de bases ayant été expliqué (cf. II – 1), après retranscription en langage python, on peut s'intéresser à la simulation d'un champ de bataille (cf. III – B) :

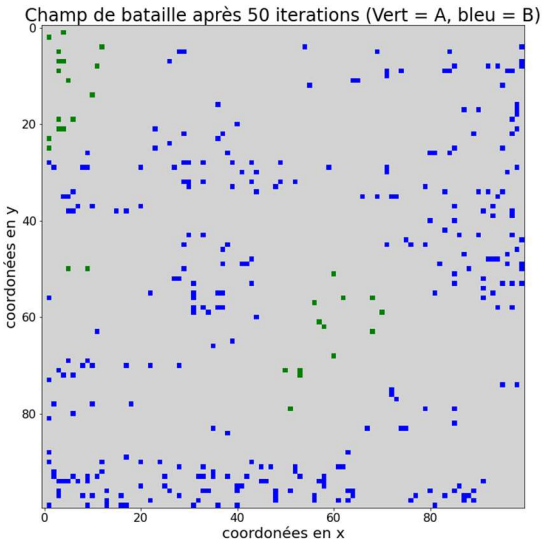
A la page suivante, on va exécuter le programme trois fois pour voir comment celui-ci se comporte :


Champ de bataille avant la simulation (vert = A, bleu = B)
coordonees en y
coordonees en x
0
20
40
60
80

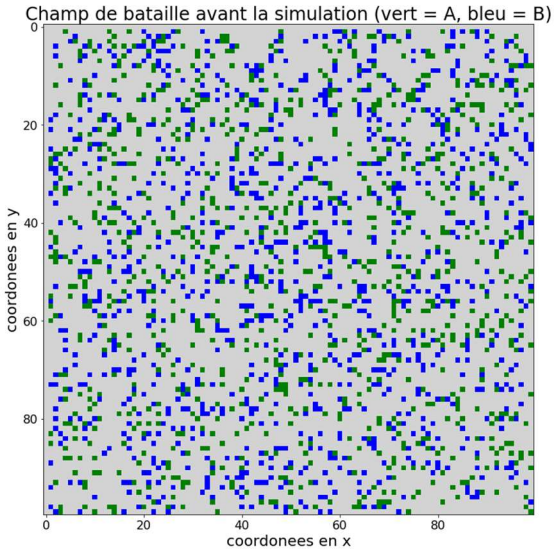

Champ de bataille après 50 iterations (Vert = A, bleu = B)
coordonées en y
coordonées en x
0
20
40
60
80

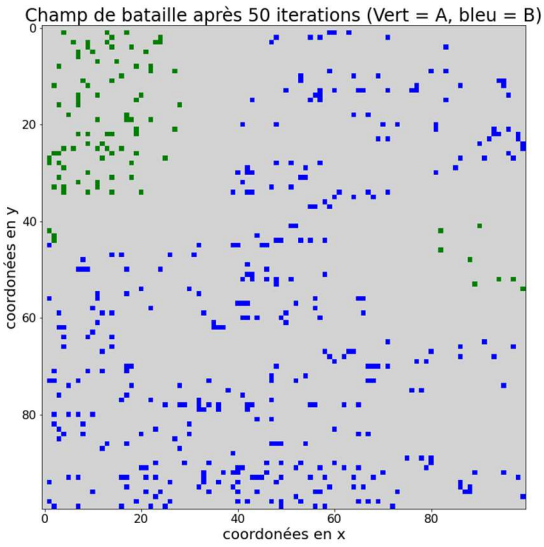
1


Champ de bataille avant la simulation (vert = A, bleu = B)
coordonees en y
coordonees en x
0
20
40
60
80


Champ de bataille après 50 iterations (Vert = A, bleu = B)
coordonées en y
coordonées en x
0
20
40
60
80

2

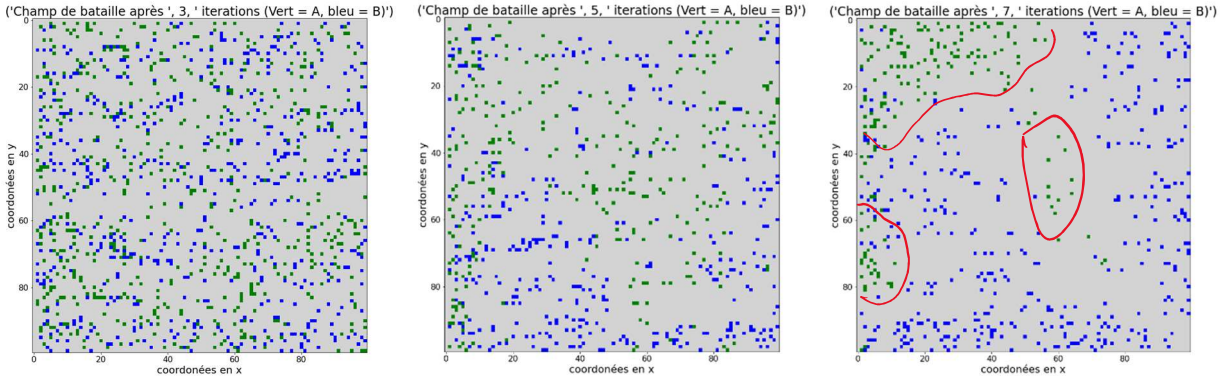

Champ de bataille avant la simulation (vert = A, bleu = B)
coordonees en y
coordonees en x
0
20
40
60
80


Champ de bataille après 50 iterations (Vert = A, bleu = B)
coordonées en y
coordonées en x
0
20
40
60
80

3

On constate dans les 3 situations qu'à l'origine les agents A (vert) et B (bleu) sont répartis proportionnellement sur la grille de départ à des positions aléatoires. On remarque ensuite, au 3 reprises, que les agents A et B restant ce sont rassemblés en petits groupes.

Les agents de même nature ont tendance à se rassembler pour se préserver, ce qui explique les résultats obtenu. Ce comportement ne dépend pas du nombre d'itérations, de plus les groupes commencent à apparaître dès la 5ème itération :



D'après l'algorithme, on a vu que les cellules possédaient un certain nombre de points de vie, ce qui déterminait à chaque itérations lesquelles d'entre-elles seraient supprimées au tour suivant. On peut donc en déduire que plus on se rapproche du centre d'un groupe, plus la cellule a de points de vie.

Néanmoins ici, chaque agent est statique, sa position ne change pas au cour des itérations. L'ajout d'une fonction influant sur les coordonnées x et y de chaque agent permettrait d'avoir une foule mobile, ainsi le résultat serait différent. Cet ajout fait partie des objectifs que nous nous sommes fixés pour l'année prochaine.

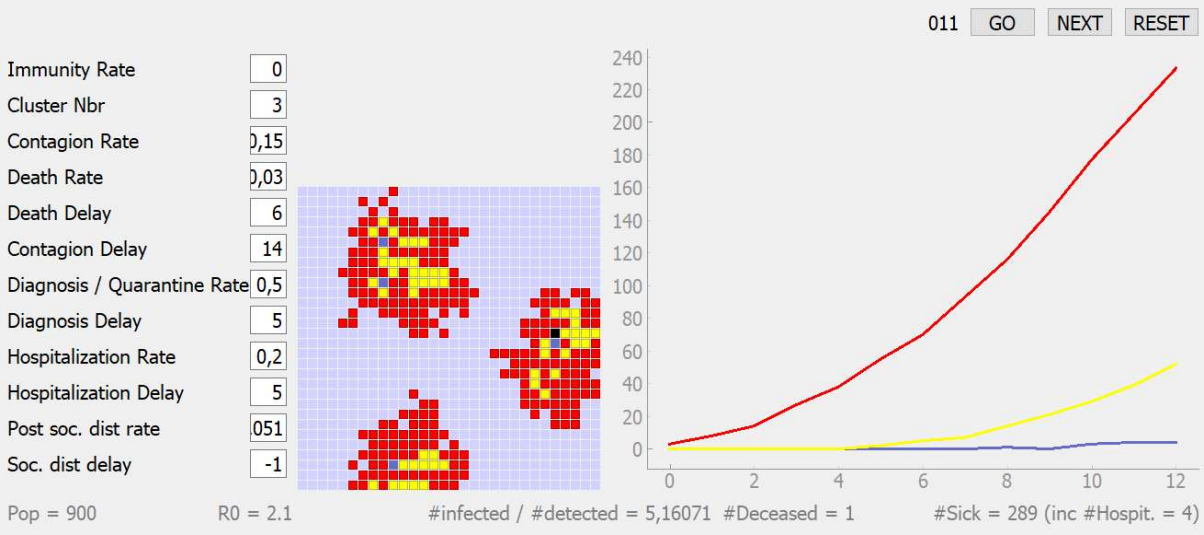
Nous le rappelons, l'intérêt de ce programme est d'entrevoir tout le potentiel de l'utilisation des classes en python pour simuler l'interaction entre chacun des agents.

2) <u>D'autres exemples :</u>

A. Jeu de la vie version épidémiologique.

(Les deux programmes qui suivent n'ont pas été réalisé par nous)

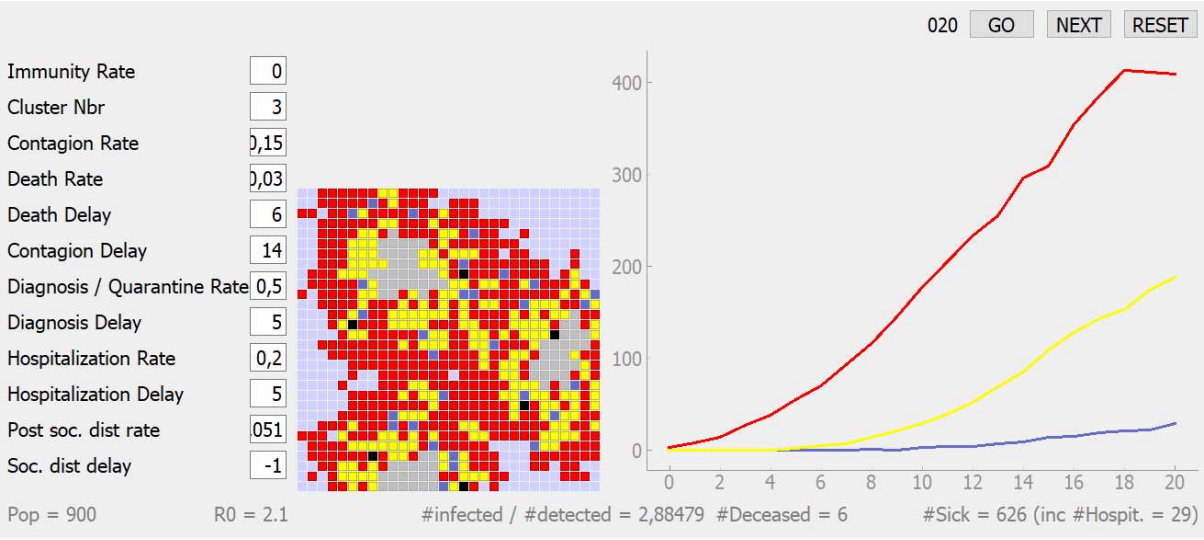
A l'instar de l'algorithme imaginé par John Conway, on peut rajouter d'autres paramètres pour simuler la propagation d'un virus :



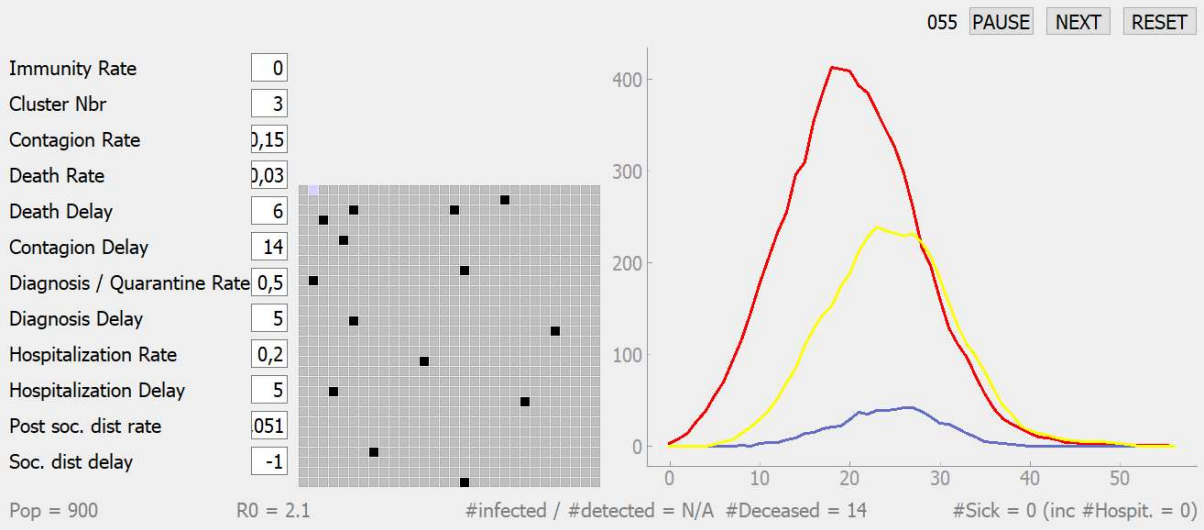
L'avantage de ce type d'algorithme est qu'ils nous permettent de mieux visualiser l'évolution d'un virus spatialement et temporairement. (il faut noter que ce programme n'est pas de nous)

On constate que, à l'image des secousses sismiques, l'épidémie possède un épicentre, et que la propagation de celle-ci peut être assimilé à une onde qui se propage dans toutes les directions. Ici, ce programme est une représentation plutôt fidèle d'une épidémie à grande échelle sur des mois en accéléré.

Ce programme est aussi très représentatif des rebonds que peut avoir une épidémie dont nous avons parlé précédemment. En effet, la propagation d'un virus peut repartir de quelques cas isolés qui se généralisent.



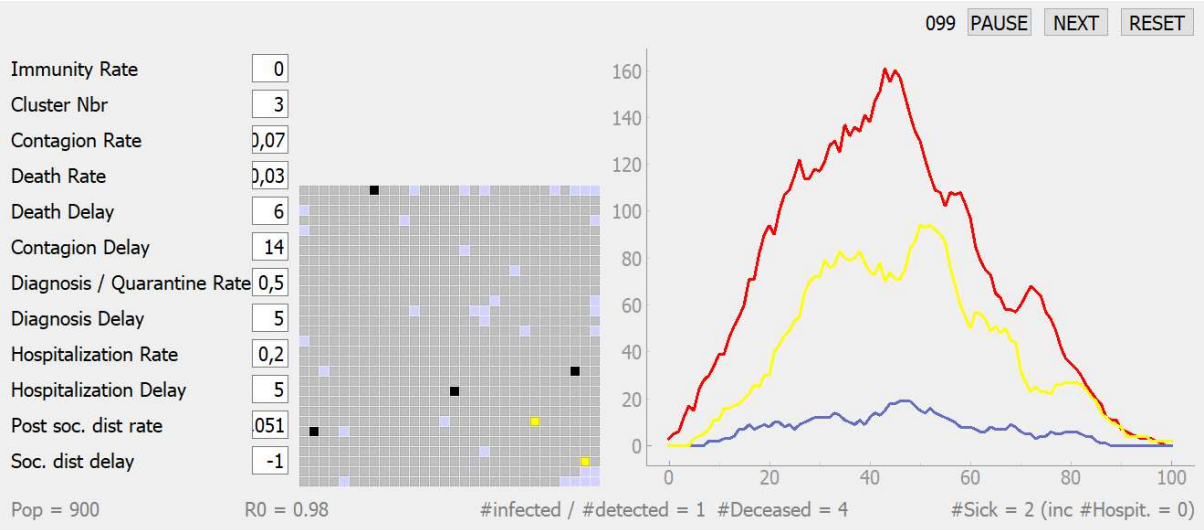
On constate que inversement au début, le statut d'immunisé est plus vite atteint aux anciens épicentres car c'est là que l'épidémie avait commencé.



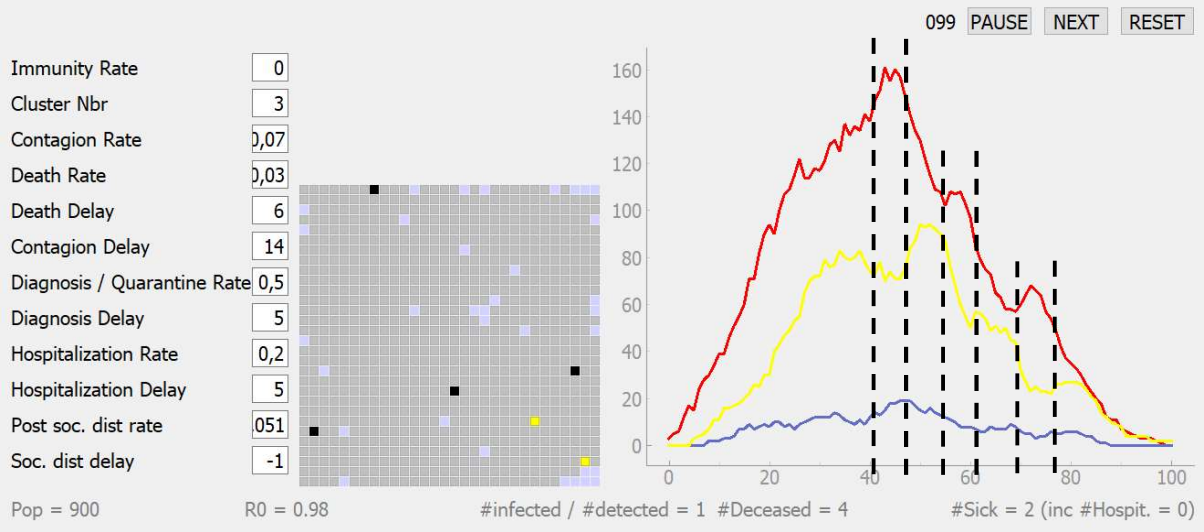
Finalement, on observe ces trois courbes, les personnes infectées au cour du temps (en rouge), les personnes misent en quarantaine au cour du temps (en jaune), et les personnes hospitalisées au cour du temps (en bleue).

On rencontre de nouveau le saut traduit par la courbe rouge que nous avions déjà évoqué. Par ailleurs, on retrouve aussi le caractère exponentiel qu'une épidémie adopte..

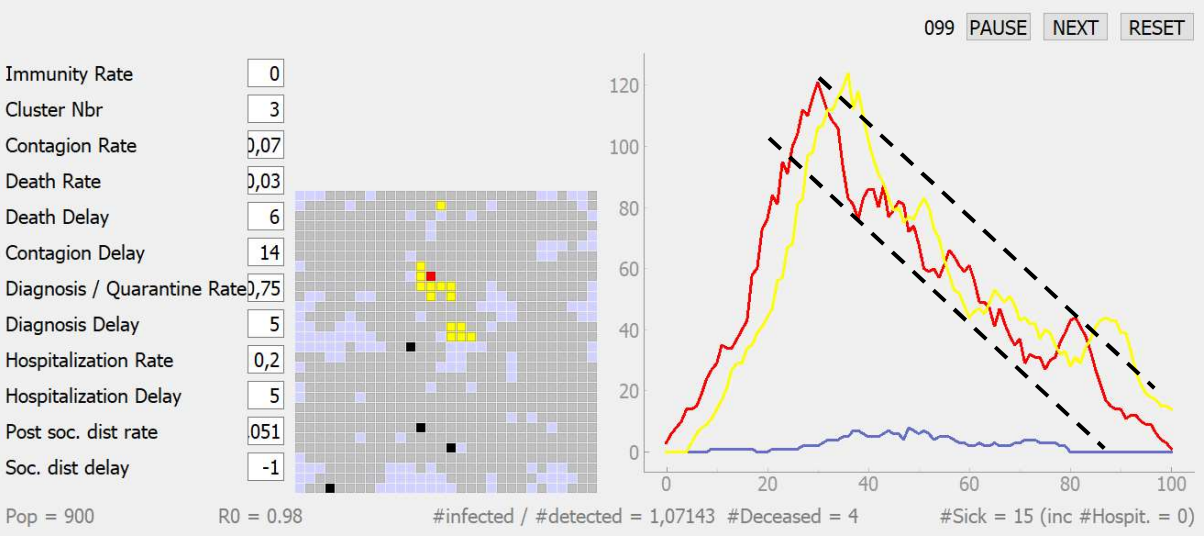
Un autre des avantages de cet algorithme est qu'on peut faire varier les paramètres initiaux, on va donc diminuer le taux de contagion à 0,07 (Contagion Rate) pour se rapprocher un peu plus de la réalité :



On constate que le rebond principal est lui-même constitué de plusieurs rebonds, on observe que ces micro rebonds surviennent quand le nombre de personnes hospitalisées (courbe bleue) diminue où encore quand le nombre de personnes en quarantaine diminue aussi :

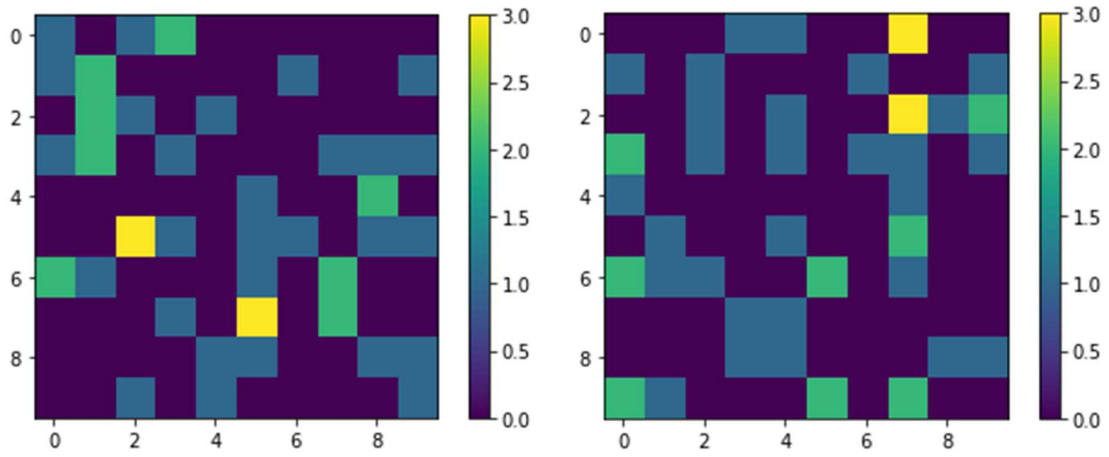


Enfin, pour illustrer, si on augmente le pourcentage de personne qui respectent la mise en quarantaine jusqu'à 0,75 soit 1 personne sur 4 ; on remarque que l'épidémie se fait rattraper par la mise en quarantaine des individus. Cela a pour effet de diminuer le nombre de personnes décédées et aussi de stabiliser la courbe des personnes hospitalisées (en bleue) :



B. Modélisation de l'échange entre individus

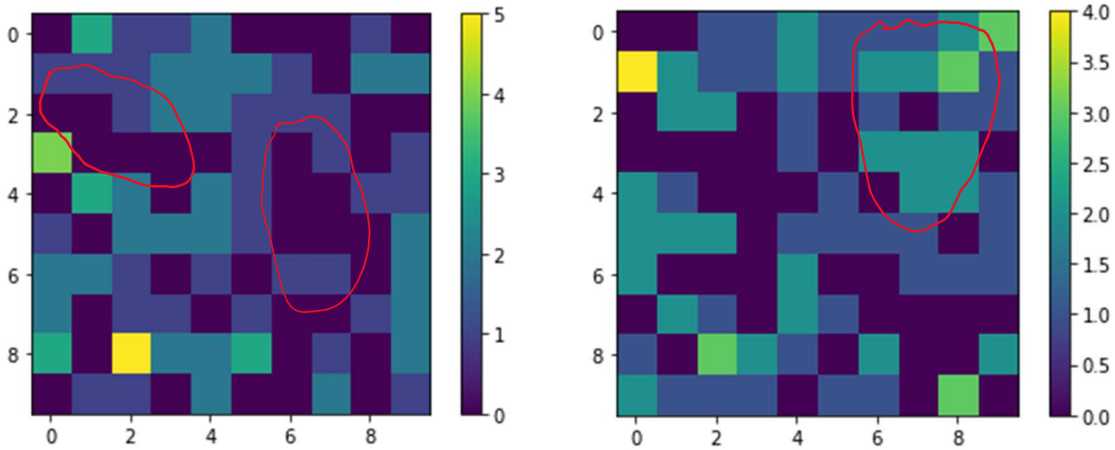
A l'aide d'un tutoriel, on peut coder un programme permettant de simuler l'échange de pièce de 1€ entre les individus sur une grille :



On peut faire varier les paramètres du programme mais ici, chaque case correspond à un individu, la grille fait du 10 par 10 donc il y a 100 individus, à l'origine 50 pièces sont distribuées aléatoirement, ensuite on itère une fonction simuilant un échange aléatoire entre deux cases adjacentes, et cela pendant 20 tours. La couleurs de la case correspond au nombre de pièce quelle possède.

On observe le phénomène préscédement évoqué avec la simulation du champ de bataille, avec le gradient de couleurs on observe que les agents qui ont le plus de pièces sont plus isolés. Avec l'algorithme du champ de bataille on aurait pu observé le même genre de gradient si on avait défini une fonction représentant les agents en fonction de leurs point de vie.

Si on augmente le nombre de pièce de départ à 100, on observe que les indivus possèdant des pièces se rassemblent, ainsi des zones « vides » apparaisent :



Ces deux derniers exemples illustrent bien les phénomènes évoqués précedement.

Avec ces quelques pages, on comprend bien que la simulation informatique ouvre de nombreuses portes. La modélisation permet de mieux comprendre les intéractions entre individus ou encore l'évolution dans le temps de plusieurs phénomènes, on observe des mécaniques très intéressantes sur lesquelles on peut influer pour multiplier le nombre d'application à la vie réelle.

Avec ces quelques applications, on peut entrevoir ce que notre programme final nous permettra de simuler et toutes les mécaniques sociales et virologiques que nous mettrons en application.

V – Conclusion

Pour résumer le travail accompli au cours de cette année :

Nous nous étions initialement fixé pour objectif de nous approprier notre sujet, en répondant à la problématique posée en introduction : Comment la fouloscopie sert-elle l'épidémiologie ?

La réalisation de cet objectif est donc passée par l'étude de notre article scientifique, ainsi que par diverses recherches sur l'épidémiologie et la fouloscopie qui nous ont permis de comprendre et d'assimiler toutes les notions théoriques derrière ces deux sciences, mais aussi par la programmation de premières simulations simples d'épidémies virtuelles.

Ce dossier, qui est un condensé de tout ce que nous avons appris cette année est donc l'accomplissement de notre premier objectif et une réponse à la problématique : La fouloscopie sert l'épidémiologie en lui apportant sans cesse de nouveaux paramètres pour affiner les simulations, et de nouveaux outils permettant de modéliser de plus en plus précisément des foules d'individus, qui sont à la base de la propagation d'une épidémie.

Pour ce qui est de l'objectif final, celui de coder une simulation qui soit la plus précise possible grâce aux données apportées par la fouloscopie, nous n'avons pas pu le réaliser et le maintenons donc comme objectif final pour l'année prochaine.

Pour conclure ce dossier, il est important de rappeler que malgré tous les paramètres que l'on peut prendre en compte dans les simulations, ce ne sont, comme leur nom l'indique, que des simulations. Elles ne permettent que de faire une hypothèse plus ou moins précise de ce qui va se passer selon les conditions que l'on pose, et non une prédiction exacte, car il y aura toujours une part d'aléatoire, d'imprévus peu importe la situation.

Malgré tout, ces simulations nous permettent tout de même de voir l'importance de certains paramètres et leur influence sur l'évolution d'une épidémie, nous permettant ainsi d'anticiper et de limiter les dégâts d'une épidémie en adaptant notre réponse, comme pendant la crise du Covid-19.

VI –

Annexe :

Articles de référence :

Epidémiologie : https://docs.idmod.org/projects/emod-hiv/en/latest/model-seir.html

Fouloscopie : continuum-crowds.pdf (washington.edu)

Bibliographie :

https://www.thelancet.com/journals/lancet/article/PIIS0140673608609782/fulltext

https://fr.wikipedia.org/wiki/%C3%89pid%C3%A9mie_de_chol%C3%A9ra_de_Broad_Street

Qu'est ce que la Fouloscopie ? - Carnets de fouloscopie (mehdimoussaid.com)

http://uel.unisciel.fr/mathematiques/sys_diff/sys_diff_ch02/co/apprendre_ch2_03.html

https://www.wku.edu/da/covid-19-research/excel_sir_model.php

https://towardsdatascience.com/modelling-the-coronavirus-epidemic-spreading-in-a-city-with-python-babd14d82fa2

Outbreak — Melting Asphalt

https://github.com/ssiddhantsharma/Disease-Spread

GitHub - julienbordet/spread: Python disease spreading and visualisation

https://covid.idmod.org/#/ToolsInstituteforDiseaseModeling/covasim: COVID-19 Agent-basedSimulator (Covasim): a model for exploring coronavirus dynamics and interventions

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0249821

Outbreak simulator (outbreak-simulator.com)