



Propagation d' épidémies et automates cellulaires

Saes-Vincensini
Mickaël
N° de candidat :
13358



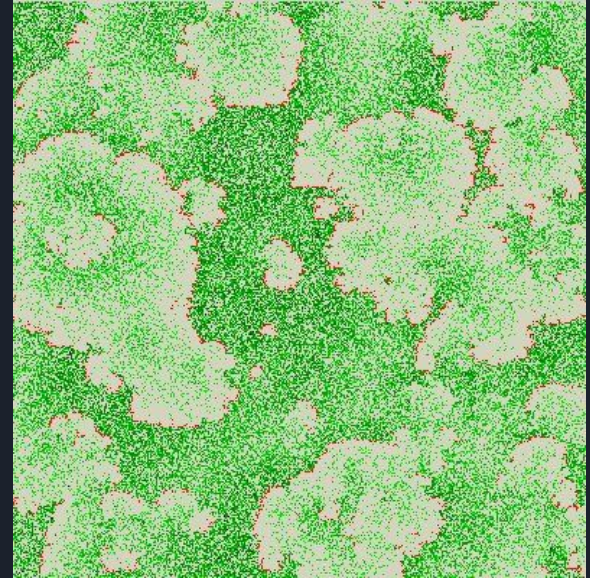
Sommaire

- Un automate cellulaire, qu'est ce que c'est ?
- Quel modèle pour une propagation d'épidémie ?
- Comment déterminer des critères permettant de réduire la propagation d'une épidémie ?
- Un modèle d'automate cellulaire est-il pertinent pour simuler une épidémie ?

Automate Cellulaire

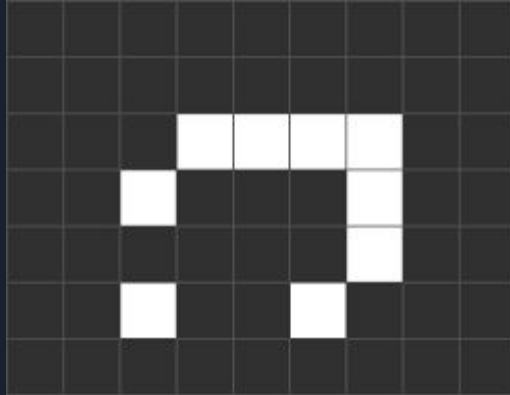
Objet mathématique qui évolue par étape en suivant des règles simples pour simuler des situations du monde réel

On le modélise par une grille finie de cellules symbolisées par un état, les cellules interagissent entre-elles entre chaque étape pour évoluer



Exemple concret : Le jeu de la vie

Créé par John Conway en 1970, c'est un automate cellulaire basé sur des règles simples qui permettent cependant de créer des structures complexes





Comment modéliser ?

Chaque individu sera codé par un doublet :

Son état :

- Sain : Il peut tomber malade
- Malade : Il peut mourir, ou guérir si il est malade depuis trop longtemps
- Guéris : Si il survit à la maladie, n'interagit plus avec les autres
- Décédé : N'interagit plus avec les autres

Le deuxième élément du doublet ne sert que si l'individu est malade et sert à compter depuis combien de temps un individu est malade

On va répéter des cycles de propagation tant que le nombre d'infectés est différent de 0



Une première approche simple : Une droite

Le premier automate cellulaire réalisé consiste à représenter les gens que sur une droite, chaque personne (hors bord gauche et droite) ne pouvant interagir qu'avec leurs deux plus proches voisins.

On définit la matrice ligne A (comportant n lignes, n connu) tel que $\forall i$
 $i \in \llbracket 1, n \rrbracket$,

$$A_i = (0, 0)$$

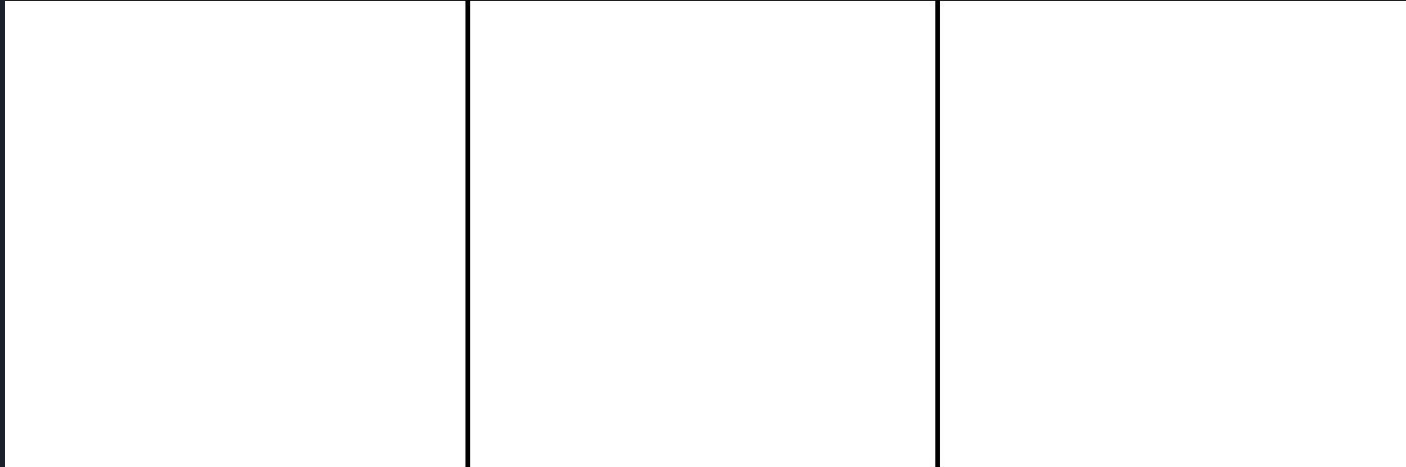
On choisit aussi aléatoirement un nombre i et on remplace

$A_i = (1, 0)$, on crée le patient zéro à la position i



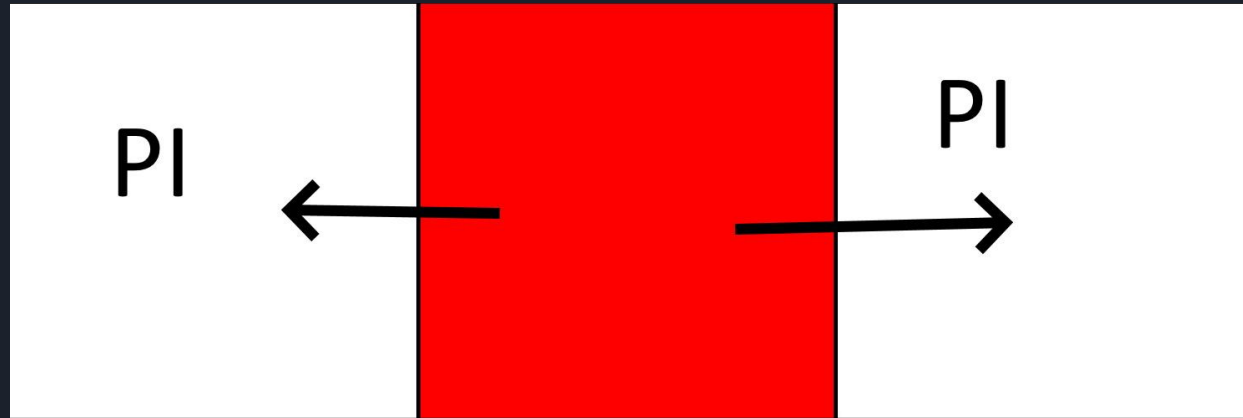
Une première approche simple : Une droite

pour $n = 3$ par exemple, on aurait 3 individus, celui du centre interagit avec les deux autres, celui de gauche et de droite n'interagissent qu'avec celui du centre



Comment fonctionne l'automate en 1D?

Si un individu est malade, il a une probabilité p_l d'infecter son voisin gauche, et une probabilité p_r d'infecter son voisin droit, les deux tirages sont indépendants.





Comment fonctionne l'automate en 1D?

à chaque étape, on s'intéresse à un individu malade depuis t périodes.

Il a une probabilité $p = p_M + 0.01 \times t$ de décéder à cette étape.

Si il ne décède pas, on incrémente t de 1.

Si un individu est malade depuis 5 périodes et qu'il ne décède pas, alors il devient "guéris", il ne peut plus tomber malade, et ne propage donc plus la maladie.



Comment exploiter les données ?

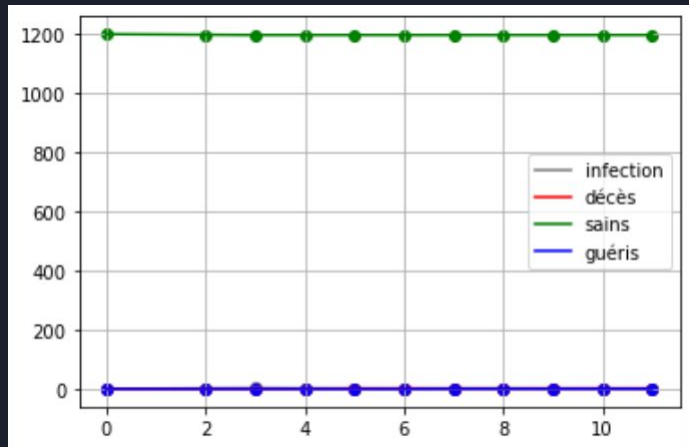
On choisit de représenter l'évolution du nombre de personnes saines, guéries, décédées et infectées en fonction du nombre d'étapes.

Pour simplifier certaines exploitations de données on peut aussi sommer les sains et guéris pour avoir la courbe des “non malades”

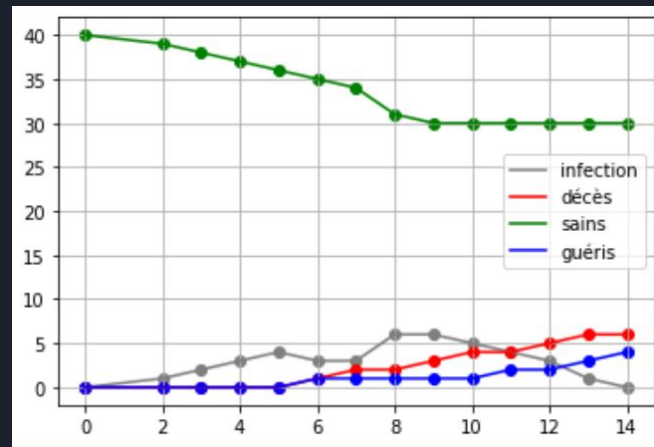
L'échelle sera linéaire

Une vision trop simpliste

Pour $p_I = 0.3$, $p_M = 0.01$



$n=1200$ (1200 individus)



$n=40$ (40 individus)



Une vision trop simpliste

L'algorithme se termine en très peu d'étapes et si on considère trop de personnes pour une probabilité d'infection cohérente, rien ne se passe.

Le modèle ne convient pas

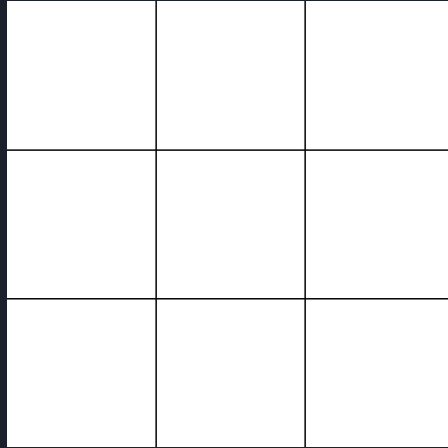


Vers un modèle de plan

On décide donc de changer de modèle de modélisation.

Un vrai automate cellulaire, vers le plan

- On modélise, dans un plan en deux dimensions (une matrice informatiquement), chaque élément de la matrice par une personne, elle va interagir avec ses 4 plus proches voisins (en haut, en bas, à gauche, à droite)
- On va modéliser la propagation d'un virus de sorte que les propagations se fassent entre voisins.

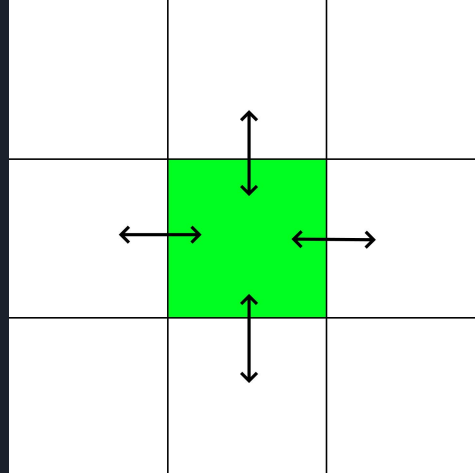


Ici un exemple de grille pour $n=3$

Comment modéliser ?

Règles de modélisation :

- L'espace est modélisé par un plan en deux dimensions, (informatiquement une matrice)
- Chaque case de la matrice symbolise une personne
- Chaque individu interagit avec ses plus proches voisins pour être infecté ou infecter





Création de l'automate cellulaire

On se donne $n \in \mathbb{N}$, on se donne la matrice

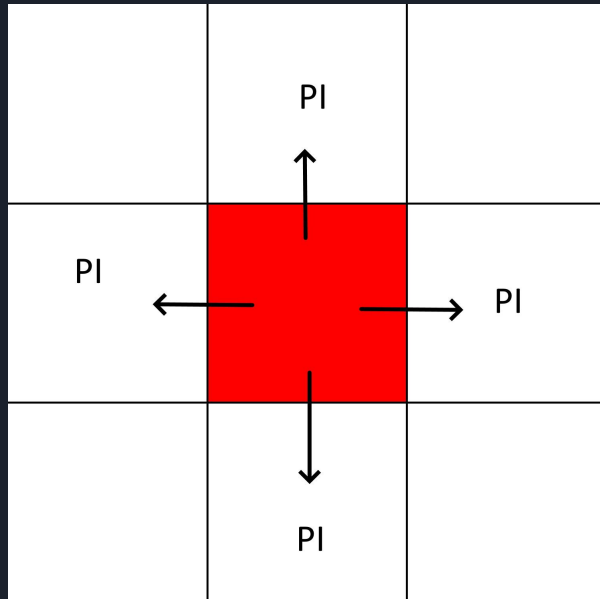
$A \in M_n(\mathbb{R} \times \mathbb{R})$, tel que $\forall (i,j) \in \llbracket 1, n \rrbracket^2$,

$A_{i,j} = (0,0)$

On choisit aussi aléatoirement un couple (i,j) et on remplace

$A_{i,j} = (1,0)$, on crée le patient zéro à la position i,j

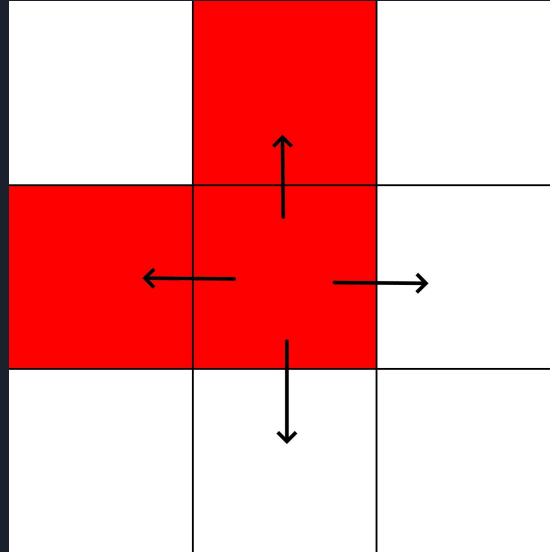
Comment être infecté ?



Pour la case (i,j) , si elle est infectée, on raisonne de la manière suivante :

Chacun des 4 plus proches voisins a une probabilité p d'être infecté par la case i,j (voisin haut, voisin bas, voisin gauche, voisin droit)

Comment être infecté ?



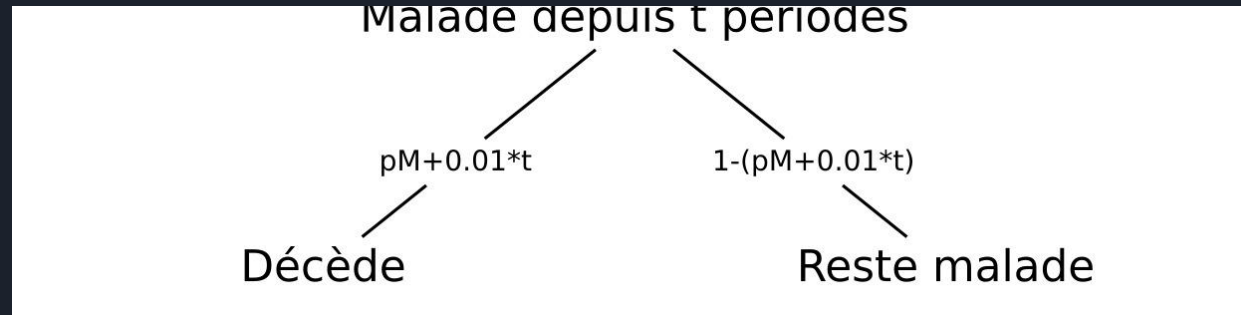
Par exemple ici la case du centre a infecté ses voisins du haut et de gauche

Comment un individu décède ?

Lorsqu'un individu à la position (i,j) est infecté, à chaque étape, il a une probabilité

$$p = pM + 0.01 \times t$$

Où $t = A_{i,j}$ (1) Le deuxième élément du couple, correspondant au temps d'infection (en étapes) ; Si il ne décède pas, on incrémente t de 1

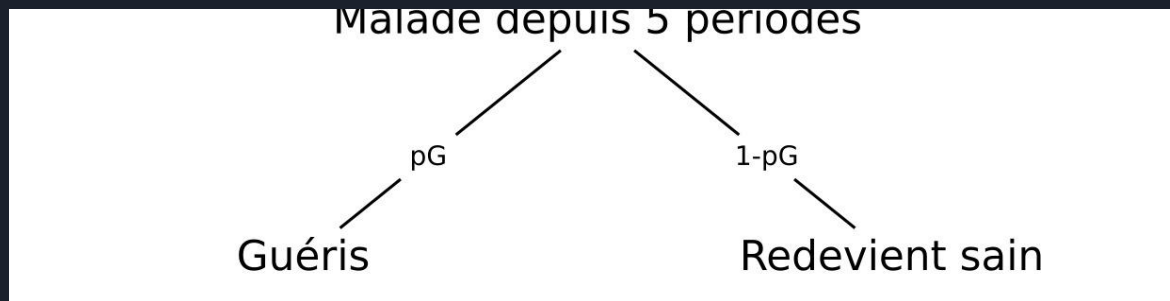


Comment un individu guérit ?

Par souci de modélisation, on décide qu'au bout de 5 cycles infectés, si l'individu survit, il peut guérir.

Dans ce cas là, il a une probabilité p_G de guérir, si il ne guérit pas, il redevient sain (Et peut donc être re infecté)

Ce choix semble cohérent pour la modélisation de l'épidémie actuelle, de nombreuses personnes sont infectées plusieurs fois par la Covid.

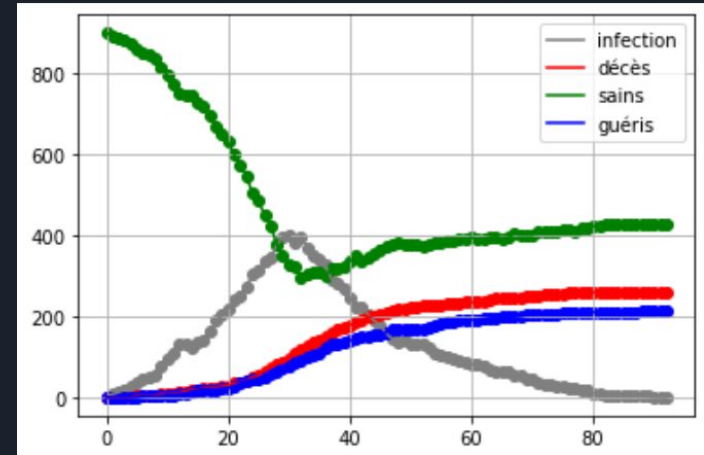


Algorithme

On répète donc les étapes de propagation telles que définies jusqu'à ce qu'il n'y ai plus d'infectés, on affiche ensuite la courbe représentant les personnes qui sont :

- Saines
- Infectées
- Guéris
- Décédées

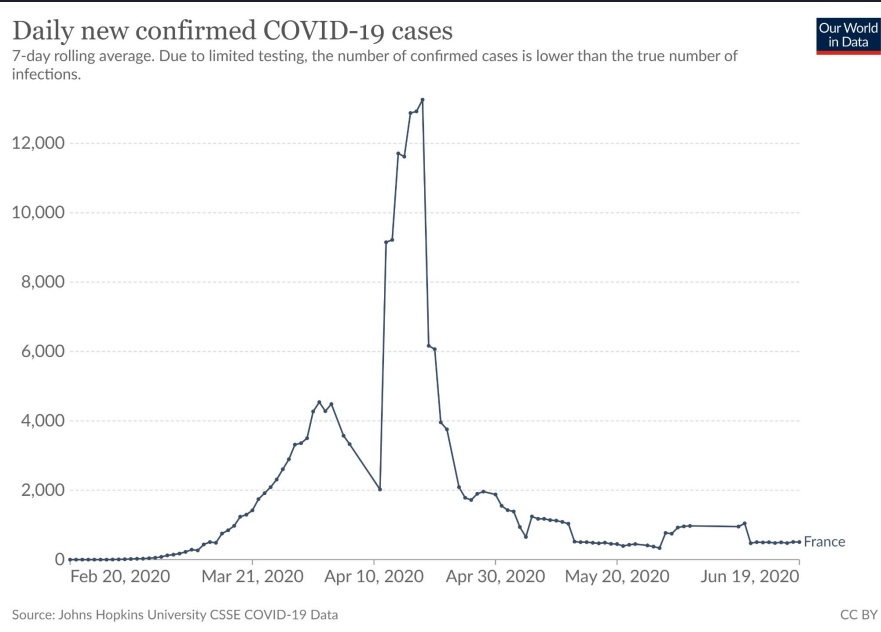
En fonction du nombre des étapes



Exemple de courbe pour $n = 30$ (900 individus)

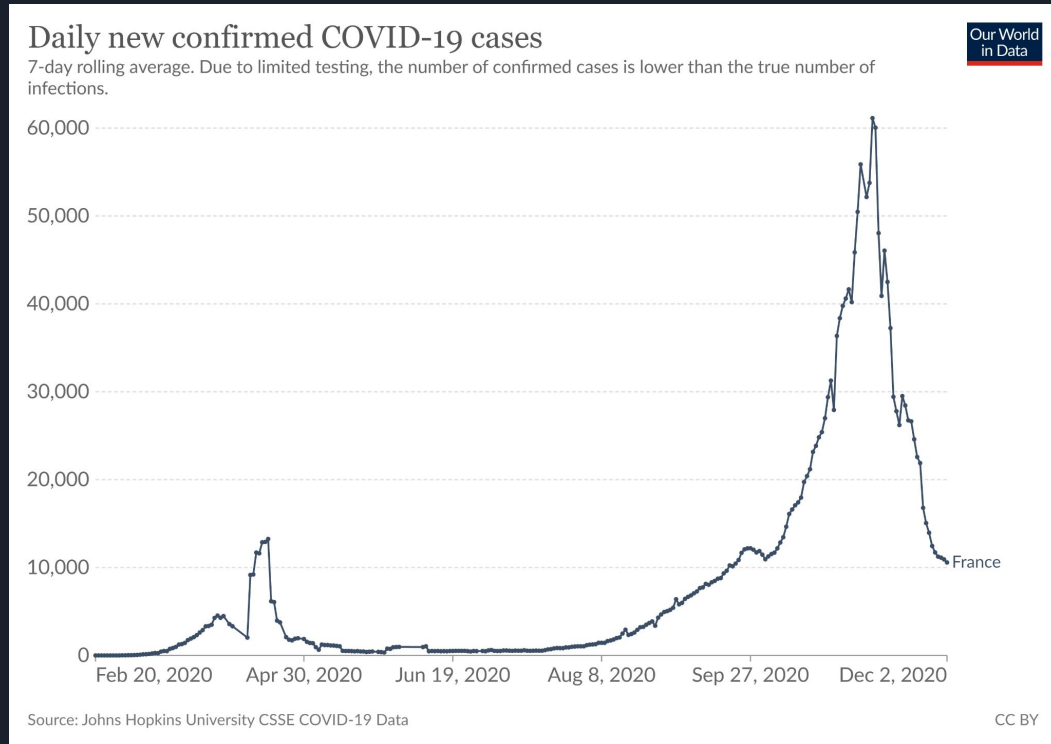
À quoi comparer ?

Afin de comparer nos résultats (surtout la forme de la courbe obtenue) on compare à la première vague du covid-19 en France:



Source : ourworldindata.org

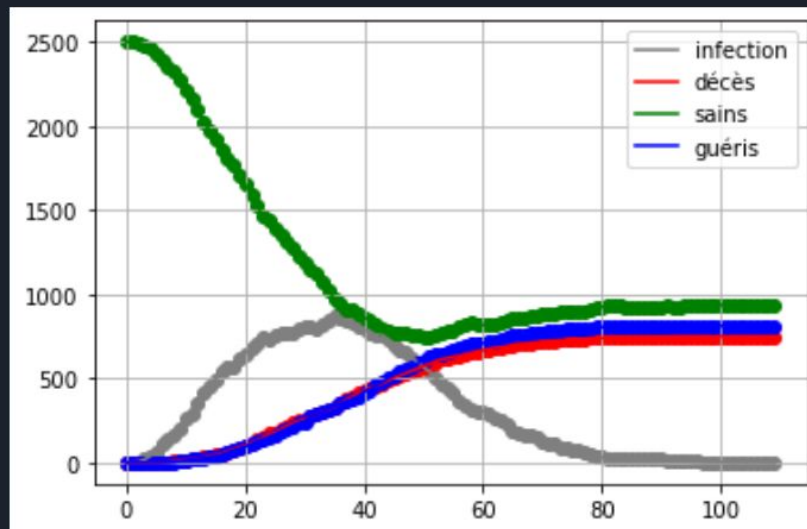
À quoi comparer ?



Même courbe pour une plage de temps plus longue

Comment faire varier les résultats ?

Il est important de choisir un nombre n d'individus suffisamment grand pour simuler une véritable population, bien qu'on ne pourra pas simuler tout un pays par limitation de complexité par exemple




$n = 50$ (2500 individus)



Observations sur l'algorithme

Problèmes directs de l'algorithme:

- 1 seule vague de contamination
- Courbes peu réalistes, pas de “pic de contamination” brutal comme le graphique



Améliorations de l'algorithme pour exploiter les données

On se propose d'améliorer l'algorithme selon 3 améliorations qui semblent être importantes pour simuler de manière plus réaliste une épidémie:

- 1: On introduit des mouvements
- 2: On crée un confinement
- 3: On repense le système d'infection



1: Des mouvements d'individus

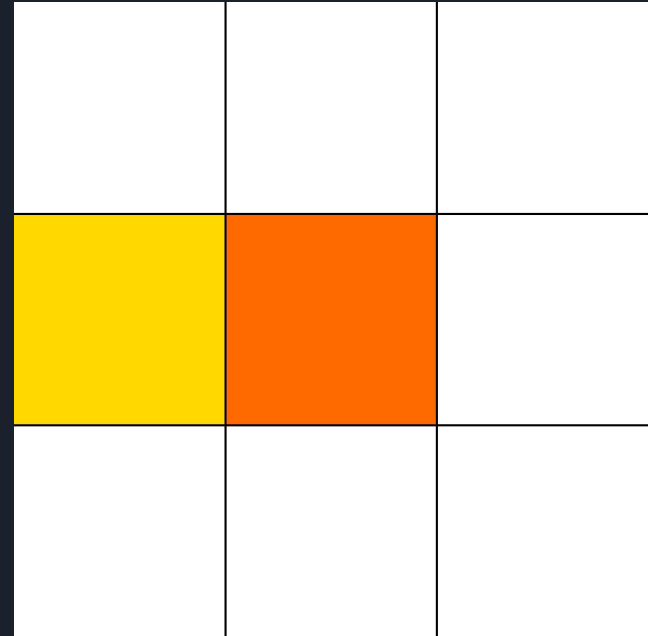
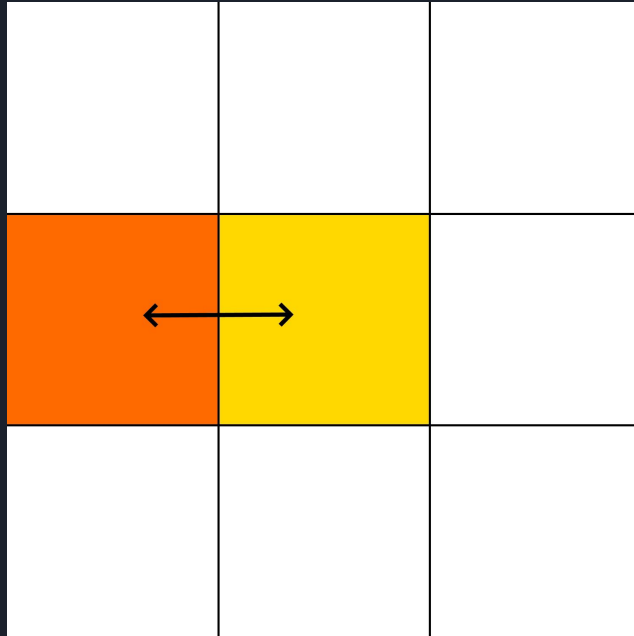
L'objectif est de simuler des mouvements de personnes lors d'épidémies, on n'interagit pas tout le temps avec les mêmes personnes, on peut se déplacer, c'est le but de cette modification

À chaque étape, un individu à une probabilité pD de se déplacer, si il se déplace, on tire une direction aléatoire (haut, bas, gauche, droite), il échange ensuite sa position avec son voisin de la direction associée

On peut décider de répéter nbd fois le déplacement à chaque étape.

On choisit de répéter 10 fois un possible déplacement (Au plus, si les 10 fois le test est réussi) ($nbd = 10$)

1: Des mouvements d'individus



Exemple d'échange



2: Un confinement

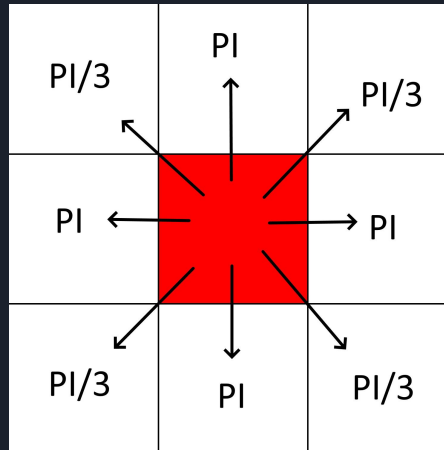
Il semble naturel de vouloir voir l'effet, sur une simulation, d'un confinement. Recréer un confinement de manière très concrète peut être très complexe (pas tout le monde ne se confine de la même manière, ne respecte les gestes barrières etc..)

On choisit de simplifier en proposant ce modèles de confinement, si plus de la moitié de la population est infectée, la probabilité d'infection est divisée par 1,5. Elle redevient normale lorsque le taux d'infectés passe en dessous de $n^2/(2.5)$ (valeur arbitraire)

On va aussi l'ajouter au fait de réduire les déplacements pendant l'épidémie, pendant le confinement, on peut se déplacer 2 fois moins et on a une chance de se déplacer divisée par 2

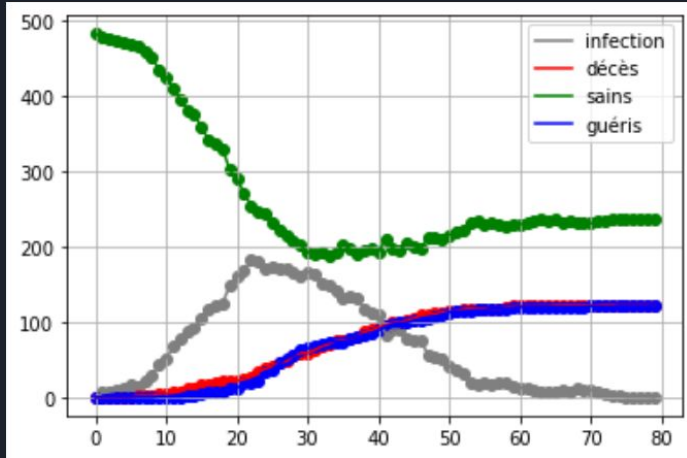
3: Un meilleur modèle de propagation

Ne pouvoir être infecté uniquement par ses 4 plus proches voisins peut sembler peu réaliste, on décide que les 4 autres voisins éloignés ont une probabilité de $pl/3$ d'infecter un individu

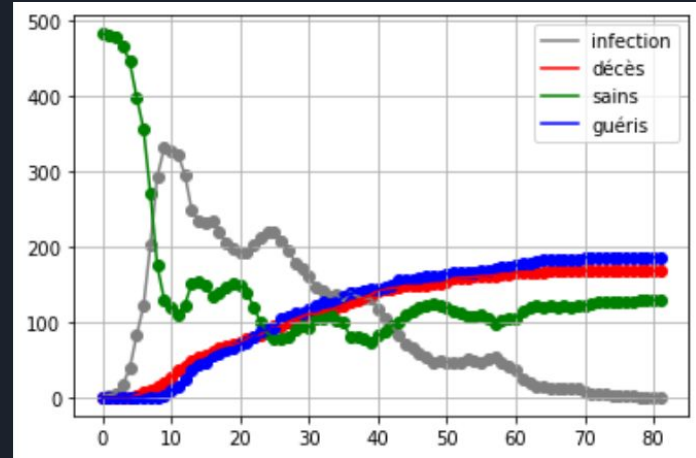


Impact sur les résultats

On implémente les 3 modifications décrites, les nouveaux graphiques sont très différents des premiers obtenus



Programme sans amélioration, n=22

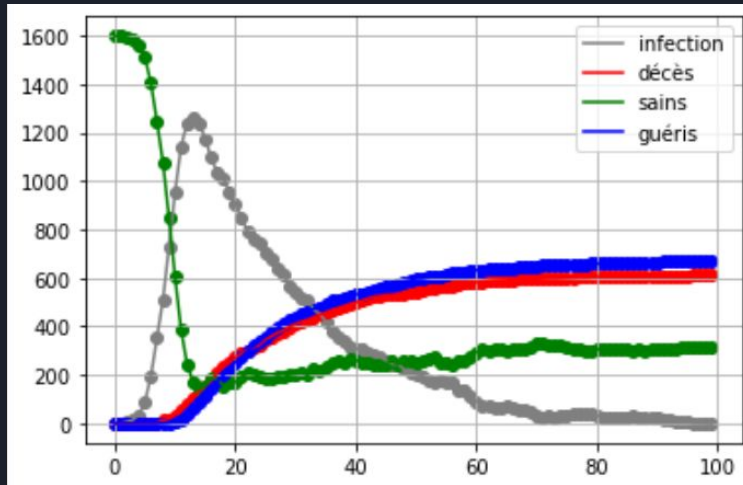


Programme avec améliorations, n=22

Comment faire varier les résultats ?

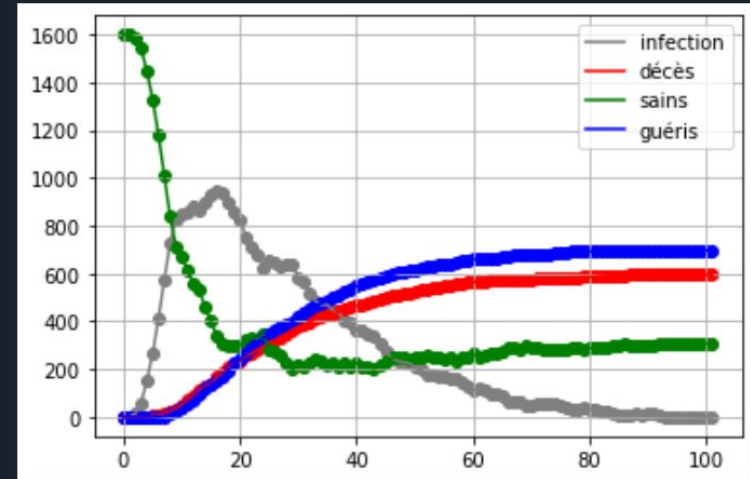
Impact du confinement

Pour $n = 40$ (1600 individus)



Sans confinement

Peu de variation du nombre de morts

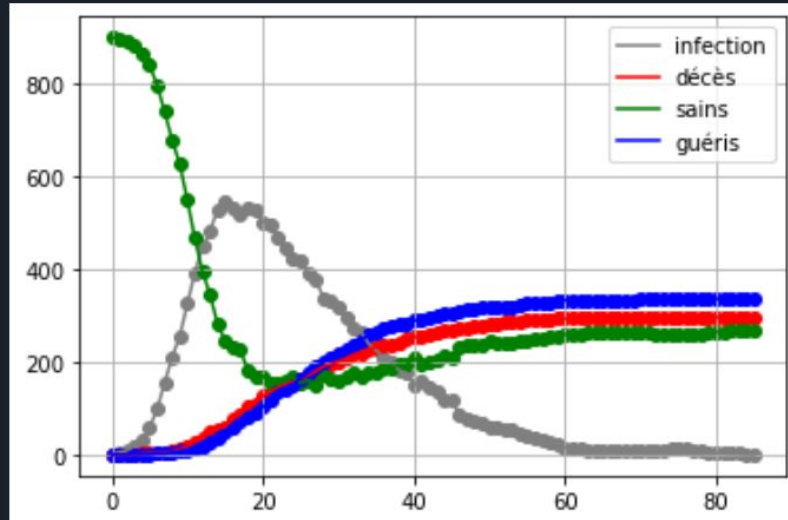


Avec confinement

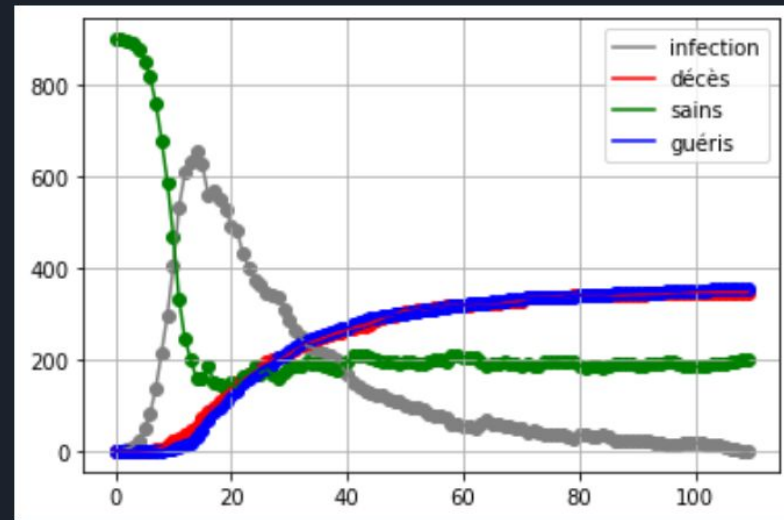
Comment faire varier les résultats ?

Variations des déplacements

On se fixe $n = 30$ (900 individus), probabilité de déplacement $pD = 0.75$



Au plus 2 déplacements

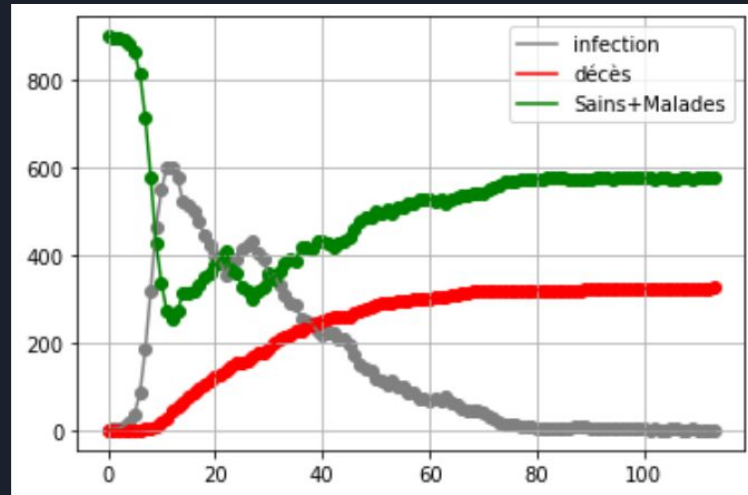


Au plus 10 déplacements

Comment faire varier les résultats ?

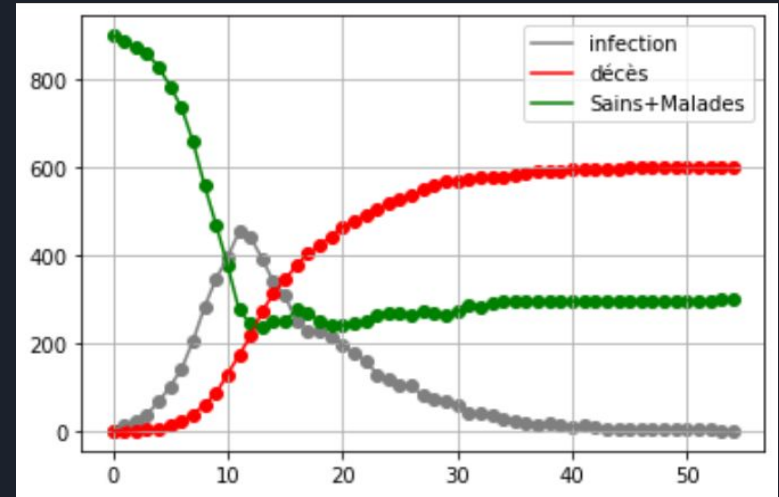
Variations des probabilités

On se fixe $n = 30$ (900 individus)



$pM = 0.01$

Pas de “deuxième saut” si pM trop élevé

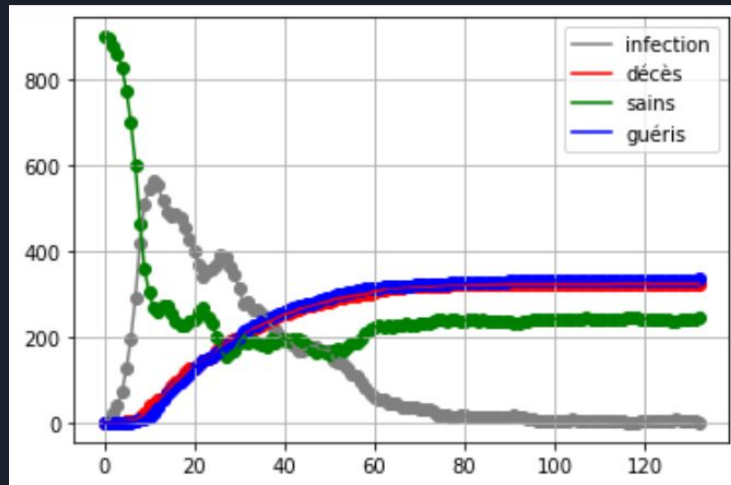


$pM = 0.1$

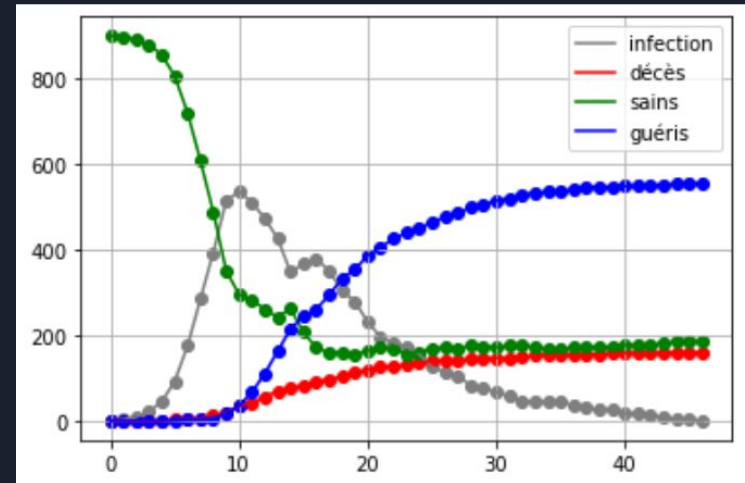
Comment faire varier les résultats ?

Variations des probabilités

On se fixe $n = 30$ (900 individus)



$pG = 0.1$

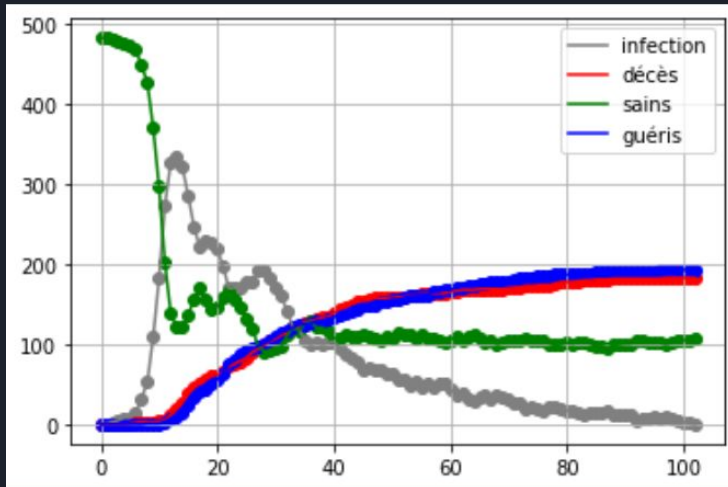


$pG = 0.4$

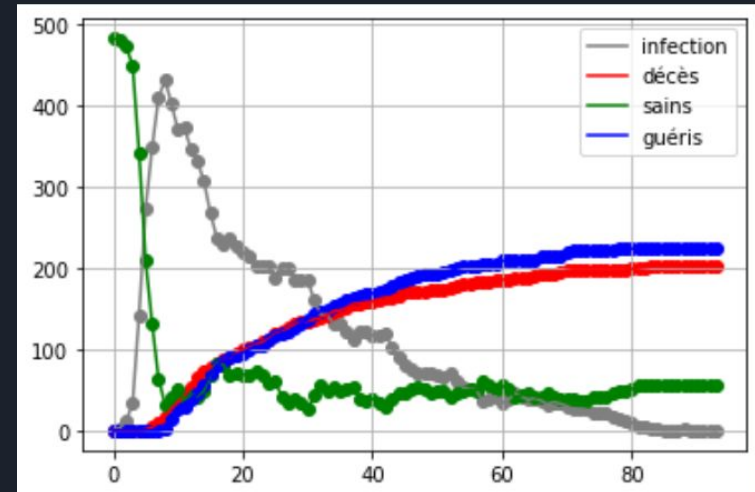
Comment faire varier les résultats ?

Variations des probabilités

On se fixe $n = 30$ (900 individus)



$p_I = 0.25$



$p_I = 0.5$



Finalemment, quels critères ?

- Complexité et réalisme du comportement humain permettent d'obtenir des courbes se rapprochant plus de la réalité

- Augmenter les probabilités peut donner des résultats aberrants, mais permettent non pas de changer l'allure de la courbe mais les "élargir" ou "compacter"

Pour réduire l'épidémie : Confiner semble fonctionner pour réduire le nombre de cas seulement, réduire les déplacements d'individus aussi.



Enfin, quels critères ?

Ainsi, pour simuler le mieux possible une épidémie, bien choisir ses probabilités initiales (cohérentes) ; Complexifier l'automate cellulaire semble rapprocher les résultats de la réalité, même si on reste toujours loin d'une réelle épidémie à cause du manque de longévité de l'algorithme



Limites

Modèle d'automate cellulaire : Pas une ville, plutôt une foule de personnes.
Limitation du nombre de personne (complexités en $O(n^2)$, pour 1 million de personne on ne peut pas lancer l'algorithme)

Limitation du nombre de personnes : On ne peut comparer que les tendances des courbes, pas leur réelles valeurs.



Valeurs des probabilités

On choisit les probabilités suivantes qui semblent convenir pour un comportement cohérent de l'épidémie (La probabilité de guérison est cependant factice car pas réellement existante)

$p_I = 0.32$ (selon covid19risk, à Paris)

$p_G = 0.12$ (peu quantifiable)

$p_M = 0.01$ (28,8M cas, 145k décès en France)

$p_D = 0.75$ (En moyenne chaque individu se déplace de 10 cases)

$nbD = 10$ (Le nombre maximum de déplacements)



Conclusion

Le modèle d'automate cellulaire essaie, par sa nature, comparée à des modèles comme le SIR, de tenir compte de l'aléatoire, il semble donc être une bonne approche pour une épidémie

Sans complexifier le programme, les résultats sont cependant aberrants.
Les critères importants pour recréer une épidémie sont donc, d'après les résultats obtenus:

Simuler des déplacements dans le modèle

Tenir compte des probabilités réelles et adapter aux valeurs choisies

Affiner le modèle semble rapprocher de résultats réelles selon les voies choisies

-Créer un confinement nous rapproche des données réelles (d'infection) et permet aussi de confirmer l'efficacité de celui-ci mais ne change pas le nombres de morts



Conclusion

-Comment déterminer des critères permettant de réduire la propagation d'une épidémie ?

-Un modèle d'automate cellulaire est-il pertinent pour simuler une épidémie ?

le modèle d'automate cellulaire est cohérent lorsque celui-ci possède des "fonctionnalités" supplémentaires, et semble plus proche de la réalité

Pour réduire la propagation de l'épidémie, selon nos résultats, il faut limiter le plus possible les déplacements et confiner à des moments cruciaux d'infectiosité.



Bibliographie

- [1] Corentin Bayette : Modélisation d'une épidémie, partie 1
- [2] Hervé Lehning : Le jeu de la vie et celui des épidémies
- [3] Mines-Ponts : Modélisation de la propagation d'une épidémie
- [4] Lucas Gerin : L'automate Epidémie et le modèle d'Eden face à l'irrégularité

Annexe : Codes

```
81 def Dim1 (n):
82     si = [0]
83     cpt1,cpt2,cpt3 = 0,0,0
84     m,I,g,s = [],[],[],[]
85     A = []
86     k=0
87     for i in range(n): ###attribution malade initiaux
88         A.append([0,0])
89         A[rd.randint(0,n)][0]=1
90         I.append(cpt1)
91         g.append(0)
92         m.append(0)
93         s.append(n-cpt1)
94         cpt1 = 0
95
96     while I[-1]!=0 or k<10:
97         k+=1
98         si.append(k+1)
99         for i in range(n):
100             if A[i][0] == 1:
101                 if (rd.binomial(1,pM+0.1*A[i][1])) == 1: #Test de décès
102                     A[i][0] = 2
103                     A[i][1] = 0
104                 if A[i][0] == 1 and A[i][1] < 4:
105                     A[i][1] +=1
106                 elif A[i][0] == 1 and A[i][1] == 4:
107                     A[i][0] = 3
108
109             for i in range(n):
110                 if A[i][0] == 1:
111                     if 1<i and (rd.binomial(1,pI) == 1) and (A[i-1][0] == 0):
112                         A[i-1][0] = 1
113                     if i<(n-2) and (rd.binomial(1,pI) == 1) and (A[i+1][0] == 0):
114                         A[i+1][0] = 1
```

Annexe : Codes

```
117     for x in A:
118         if x[0] == 1:
119             cpt1+=1
120         if x[0] == 2:
121             cpt2+=1
122         if x[0] == 3:
123             cpt3+=1
124
125     I.append(cpt1)
126     g.append(cpt3)
127     m.append(cpt2)
128     s.append(n-cpt2-cpt1-cpt3)
129     cpt1,cpt2,cpt3 = 0,0,0
130
131
132     ##Affichage
133     plt.scatter(si,I,c = 'grey')
134     plot(si,I, label = "infection", c = "grey")
135
136     plt.scatter(si,m,c= "red")
137     plot(si,m, label = "décès", c = "red")
138
139     plt.scatter(si,s, c = "green")
140     plot(si,s, label = "sains", c = "green")
141
142     plt.scatter(si,g, c = 'blue')
143     plot(si,g, label = "guéris", c = "blue")
```



Annexe : Codes

```
10 def sommeL(L1,L2):  
11     n = len(L1)  
12     L=[]  
13     for i in range(n):  
14         L.append(L1[i]+L2[i])  
15     return L #Renvoie la liste somme de deux listes L1 et L2 de même taille
```

Annexe : Codes

```
35 def Infection(A,n,pI): #Prend en entrée une matrice d'individus A
36 #n la longueur / largeur de la matrice A, pI la probabilité d'infection
37     for i in range(n):
38         for j in range(n):
39             if A[i][j][0] == 1:
40                 if 1<i and (A[i-1][j][0] == 0) and (rd.binomial(1,pI)) ==1 :
41                     A[i-1][j][0] = 1
42
43                 if i<(n-2) and (A[i+1][j][0] == 0) and (rd.binomial(1,pI)) == 1:
44                     A[i+1][j][0] = 1
45
46                 if 1<j and (A[i][j-1][0] == 0) and (rd.binomial(1,pI)) == 1:
47                     A[i][j-1][0] = 1
48
49                 if j<(n-2) and (A[i][j+1][0] == 0) and (rd.binomial(1,pI)) ==1:
50                     A[i][j+1][0] = 1
51 #Va modifier la matrice A pour tester l'infection des 4 voisins les plus proches de A
52
```

Annexe : Codes

```
54 def Infection2(A,n,pI): #Prend en entrée une matrice d'individus A
55 #n la longueur / largeur de la matrice A, pI la probabilité d'infection
56     for i in range(n):
57         for j in range(n):
58             if A[i][j][0] == 1:
59
60                 if 1<i and (A[i-1][j][0] == 0) and (rd.binomial(1,pI)) ==1 :
61                     A[i-1][j][0] = 1
62                 if i<(n-2) and (A[i+1][j][0] == 0) and (rd.binomial(1,pI)) == 1:
63                     A[i+1][j][0] = 1
64                 if 1<j and (A[i][j-1][0] == 0) and (rd.binomial(1,pI)) == 1:
65                     A[i][j-1][0] = 1
66                 if j<(n-2) and (A[i][j+1][0] == 0) and (rd.binomial(1,pI)) ==1:
67                     A[i][j+1][0] = 1
68
69
70                 if 1<i and 1<j and (A[i-1][j-1][0] == 0) and (rd.binomial(1,pI/3)) ==1:
71                     A[i-1][j-1][0] == 1
72                 if i<(n-2) and 1<j and (A[i+1][j-1][0] == 0) and (rd.binomial(1,pI/3)) ==1:
73                     A[i+1][j-1][0] == 1
74                 if 1<i and j<(n-2) and (A[i-1][j+1][0] == 0) and (rd.binomial(1,pI/3)) ==1:
75                     A[i-1][j+1][0] == 1
76                 if i<(n-2) and j<(n-2) and (A[i+1][j+1][0] == 0) and (rd.binomial(1,pI/3)) ==1:
77                     A[i+1][j+1][0] == 1
78 #Va modifier la matrice A pour tester l'infection des 8 voisins les plus proches de A
```


Annexe : Codes

```
81 def TestGM(A,n): #A la matrice des individus, n la longueur/largeur de A
82
83     for i in range(n):
84         for j in range(n):
85
86             if A[i][j][0] == 1 and A[i][j][1] == 4:
87                 if rd.binomial(1,pG)==1:
88                     A[i][j][0] = 3
89                 else:
90                     A[i][j][0] = 0
91                     A[i][j][1] = 0
92
93             elif (A[i][j][0] == 1) == 1 and (rd.binomial(1,pM+0.01*A[i][j][1])):
94                 A[i][j][0] = 2
95                 A[i][j][1] = 0
96
97             elif A[i][j][0] == 1 and A[i][j][1] < 4:
98                 A[i][j][1] +=1
99
100
101 #Modifie A afin de guérir, faire décéder, ou augmenter le temps d'infection des
102 #individus de A
```

Annexe : Codes

```
81 def TestGM(A,n): #A la matrice des individus, n la longueur/largeur de A
82
83     for i in range(n):
84         for j in range(n):
85
86             if A[i][j][0] == 1 and A[i][j][1] == 4:
87                 if rd.binomial(1,pG)==1:
88                     A[i][j][0] = 3
89                 else:
90                     A[i][j][0] = 0
91                     A[i][j][1] = 0
92
93             elif (A[i][j][0] == 1) == 1 and (rd.binomial(1,pM+0.01*A[i][j][1])):
94                 A[i][j][0] = 2
95                 A[i][j][1] = 0
96
97             elif A[i][j][0] == 1 and A[i][j][1] < 4:
98                 A[i][j][1] +=1
99
100
101 #Modifie A afin de guérir, faire décéder, ou augmenter le temps d'infection des
102 #individus de A
```

Annexe : Codes

```
17 def Deplacement(A,pD): #Prend en entrée une matrice d'individus A
18     # 0 : gauche ; 1 : droite ; 2 : Haut ; 3 : Bas
19     n=len(A)
20     for i in range(n):
21         for j in range(n):
22             if rd.binomial(1,pD) == 1:
23                 side = rd.randint(0,3)
24                 if side == 0 and i>0:
25                     A[i][j],A[i-1][j] = A[i-1][j],A[i][j]
26                 elif side == 1 and i<(n-1):
27                     A[i][j],A[i+1][j] = A[i+1][j],A[i][j]
28                 elif side == 2 and j<(n-1):
29                     A[i][j],A[i][j+1] = A[i][j+1],A[i][j]
30                 else:
31                     A[i][j],A[i][j-1] = A[i][j-1],A[i][j]
32
33     #Déplace chaque individu d'au plus une case avec une probabilité pD
```

Annexe : Codes

```
105 def Programme(n):
106     #Prend en entrée  $n \in \mathbb{N}$  pour réaliser l'algorithme décrit
107     #definition des bases
108     x = [0]
109     A = []
110     pI = prI
111     cpt1,cpt2,cpt3 = 0,0,0
112
113     S = []
114     I = []
115     G = []
116     M = []
117     #Creation de la matrice "2D" initiale
118     for i in range(n):
119         A.append([])
120         for j in range(n):
121             A[i].append([0,0])
122     A[rd.randint(0,n)][rd.randint(0,n)][0]=1
123     S.append((n*n)-1)
124     I.append(1)
125     G.append(0)
126     M.append(0)
127     cpt1=0
128     k=0
```

Annexe : Codes

```
135 #Confinement
136
137 if I[-1]>= ( (n*n)/2 ):
138     pI = prI/1.5
139     pD = prD/2
140     nbd=5
141 elif I[-1] <= ( (n*n)/(2.5) ):
142     pI = prI
143     pD = prD
144     nbd=10
145
146
147 TestGM(A,n)
148
149 Infection2(A,n,pI)
150
151 for i in range(nbd):
152     Deplacement(A,pD)
153
154 for i in range(n): #On compte les états des individus après modification
155     for j in range(n):
156         if A[i][j][0] == 1:
157             cpt1+=1
158         elif A[i][j][0] == 2:
159             cpt2+=1
160         elif A[i][j][0] == 3:
161             cpt3+=1
162 #Ajout des valeurs dans les listes correspondantes
163 I.append(cpt1)
164 M.append(cpt2)
165 G.append(cpt3)
166 S.append((n*n)-cpt1-cpt2-cpt3)
167 cpt1,cpt2,cpt3 = 0,0,0
```

Annexe : Codes

```
163     ##Affichage
164     graph = plt.figure(1) #4 données
165     plt.scatter(x,I,c = 'grey')
166     plot(x,I, label = "infection", c = "grey")
167
168     plt.scatter(x,M,c= "red")
169     plot(x,M, label = "décès", c = "red")
170
171     plt.scatter(x,S, c = "green")
172     plot(x,S, label = "sains", c = "green")
173
174     plt.scatter(x,G, c = 'blue')
175     plot(x,G, label = "guéris", c = "blue")
176
177     graph2 = plt.figure(1) #Guéris et sains confondus
178     plt.scatter(x,I,c = 'grey')
179     plot(x,I, label = "infection", c = "grey")
180
181     plt.scatter(x,M,c= "red")
182     plot(x,M, label = "décès", c = "red")
183
184     plt.scatter(x,sommeL(G,S), c = "green")
185     plot(x,sommeL(G,S), label = "guéris + sains", c = "green")
186
187     plt.legend()
188     plt.grid()
189     show()
190     graph.show()
```