

Modélisation d'une épidémie

PREVOT Alexia – 41780

Comment modéliser l'évolution d'une épidémie afin de mieux la comprendre et prévoir les influences des différentes mesures prises pour contrer celle-ci ?

SOMMAIRE

I. Modèle SIR

- Présentation du modèle
- Complément du modèle

II. Taux de reproduction

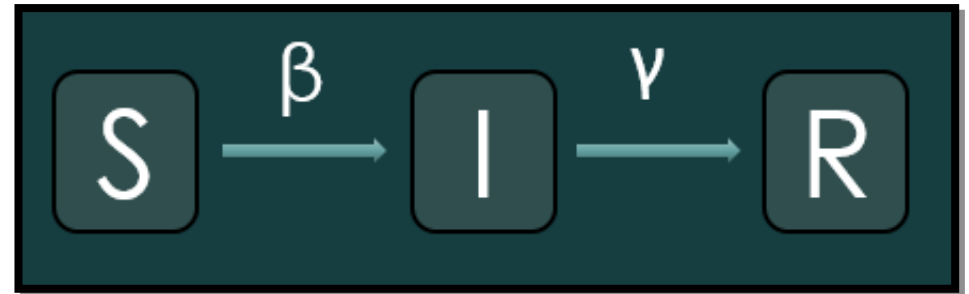
- Définition
- Intérêts et difficultés de le calculer

III. Simulation dans l'espace

- Autre modèle épidémiologique
- Comment affiner le modèle

Modèle SIR, développé par Kermack et McKendrick en 1927

- $S(t)$: susceptibles
- $I(t)$: infectés
- $R(t)$: retirés
- Population totale :
 $N = S + I + R$
- β = taux de transmission
- γ = taux de guérison

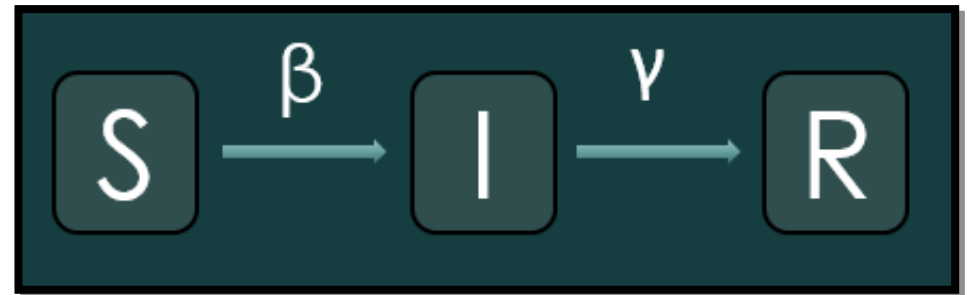


Système différentiel:

$$\begin{cases} S'(t) = -\beta I(t)S(t) \\ I'(t) = \beta I(t)S(t) - \gamma I(t) \\ R'(t) = \gamma I(t) \end{cases}$$

Modèle SIR, développé par Kermack et McKendrick en 1927

- $S(t)$: susceptibles
- $I(t)$: infectés
- $R(t)$: retirés
- Population totale :
 $N = S + I + R$
- β = taux de transmission
- γ = taux de guérison



Système différentiel:

$$\begin{cases} S'(t) = -\beta I(t)S(t) \\ I'(t) = \beta I(t)S(t) - \gamma I(t) \\ R'(t) = \gamma I(t) \end{cases}$$

$$\beta = k \cdot \tau / N$$

→ τ est le facteur de transmissibilité
→ k le nombre de contacts possibles d'un individu infecté avec d'autre personne en une unité de temps

$$\gamma = 1/D$$

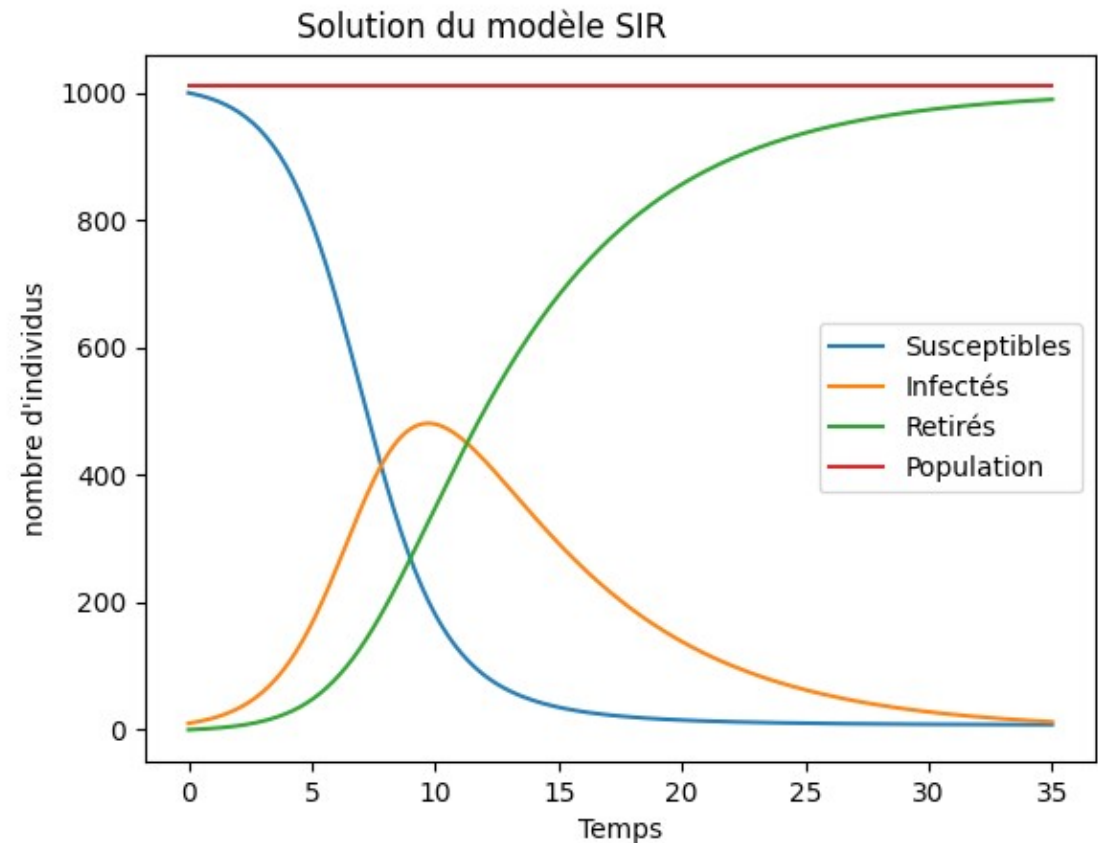
→ D la durée d'infection d'une personne

Modèle SIR, développé par Kermack et McKendrick en 1927

- $S(t)$: susceptibles
- $I(t)$: infectés
- $R(t)$: retirés
- Population totale :
 $N = S + I + R$
- β = taux de transmission
- γ = taux de guérison

Système différentiel:

$$\begin{cases} S'(t) = -\beta I(t)S(t) \\ I'(t) = \beta I(t)S(t) - \gamma I(t) \\ R'(t) = \gamma I(t) \end{cases}$$

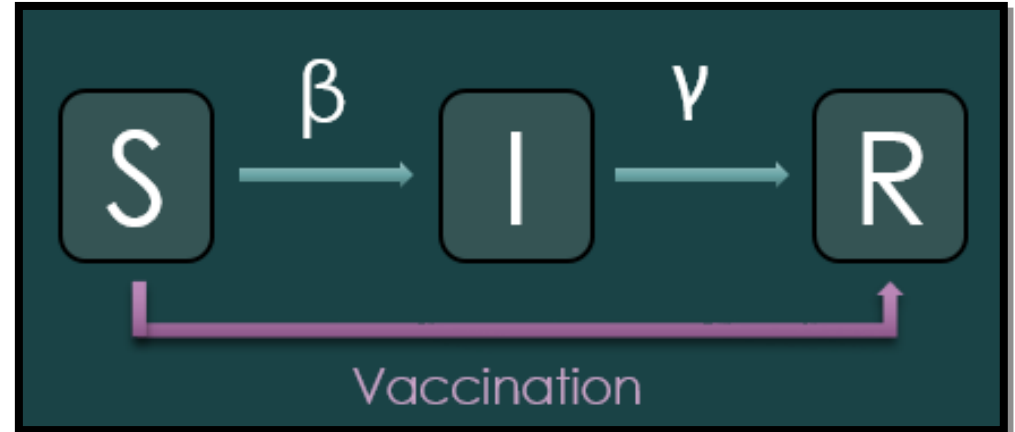


Le taux de transmission est de 0,008 et le taux de guérison est de 1/10

Modèle avec vaccination

- c = taux de vaccination

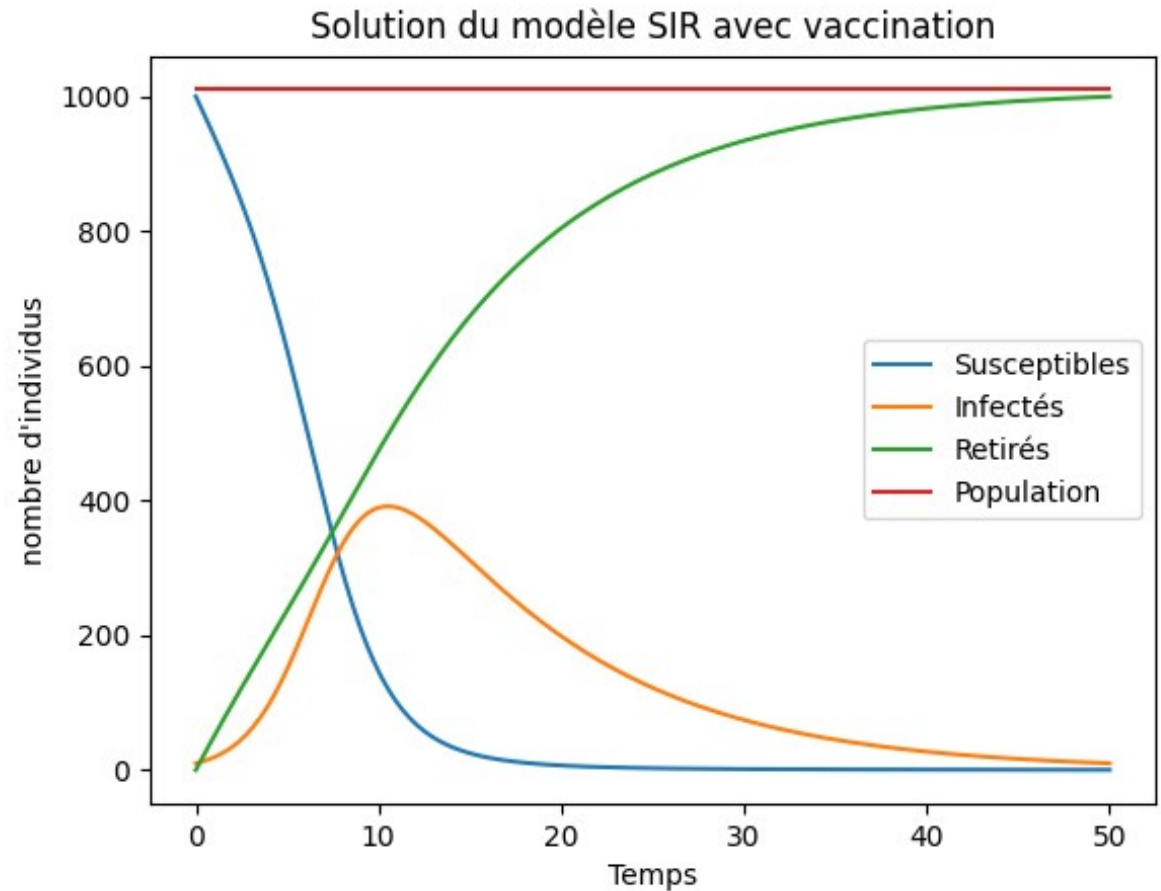
$$\begin{cases} S'(t) = -\beta S(t)I(t) - cS(t) \\ I'(t) = \beta S(t)I(t) - \gamma I(t) \\ R'(t) = \gamma I(t) + cS(t) \end{cases}$$



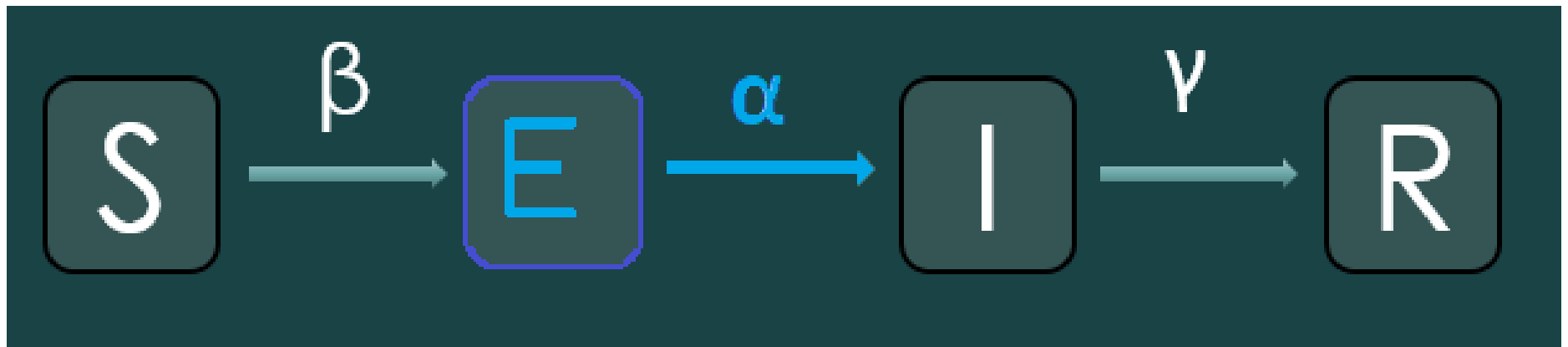
Modèle avec vaccination

- c = taux de vaccination

$$\begin{cases} S'(t) = -\beta S(t)I(t) - cS(t) \\ I'(t) = \beta S(t)I(t) - \gamma I(t) \\ R'(t) = \gamma I(t) + cS(t) \end{cases}$$



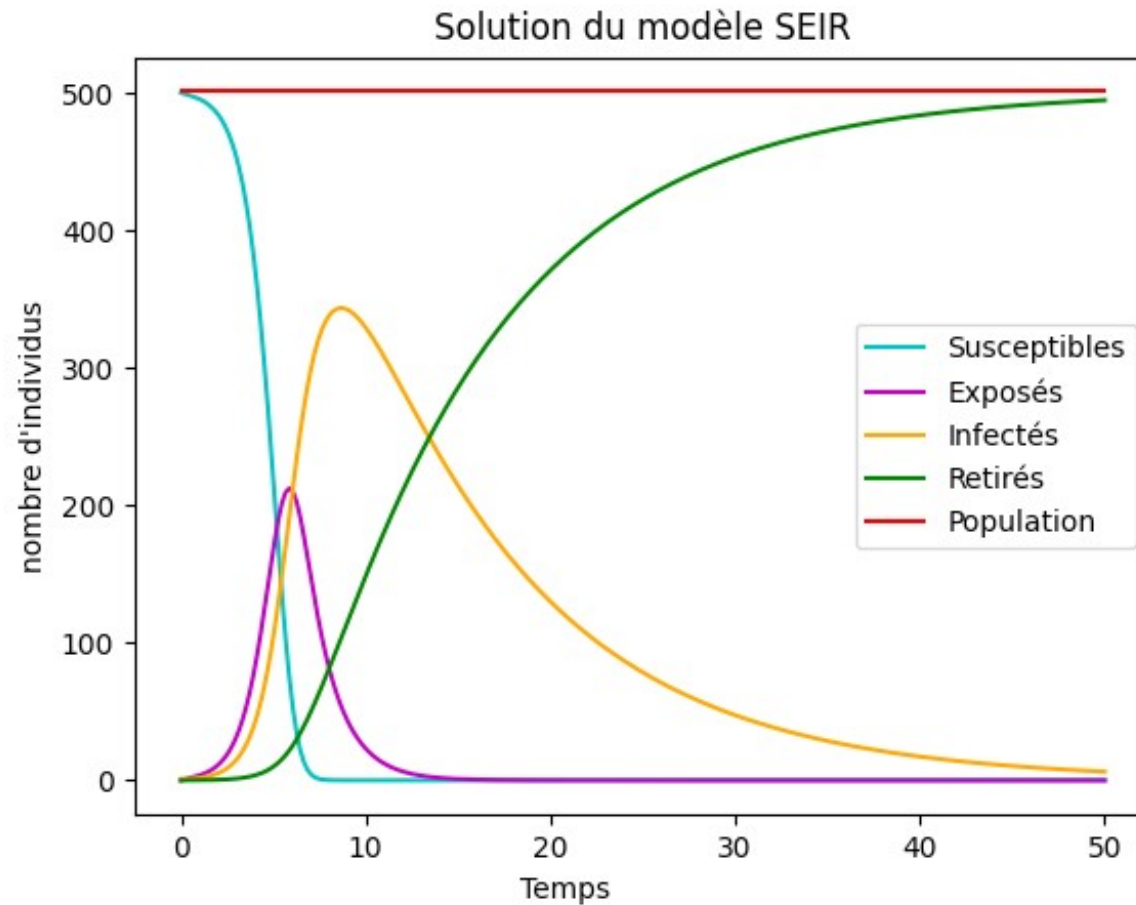
Modèle SEIR



- $E(t)$: exposés (infectés non infectieux)
- α = taux d'incubation

$$\begin{cases} S'(t) = -\beta S(t)I(t) \\ E'(t) = \beta S(t)I(t) - \alpha E(t) \\ I'(t) = \alpha E(t) - \gamma I(t) \\ R'(t) = \gamma I(t) \end{cases}$$

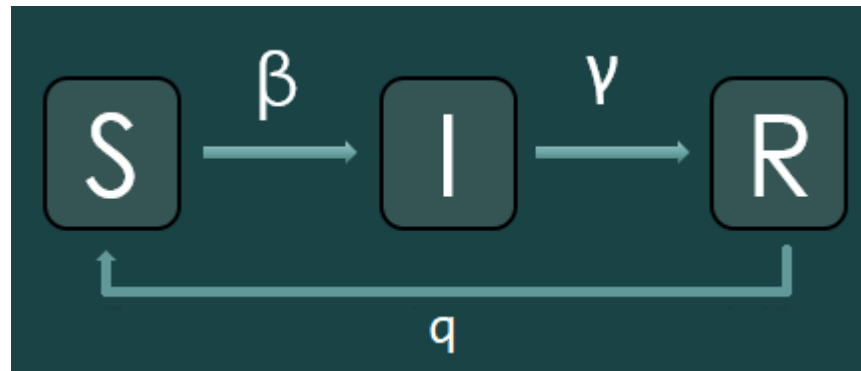
Résolution numérique



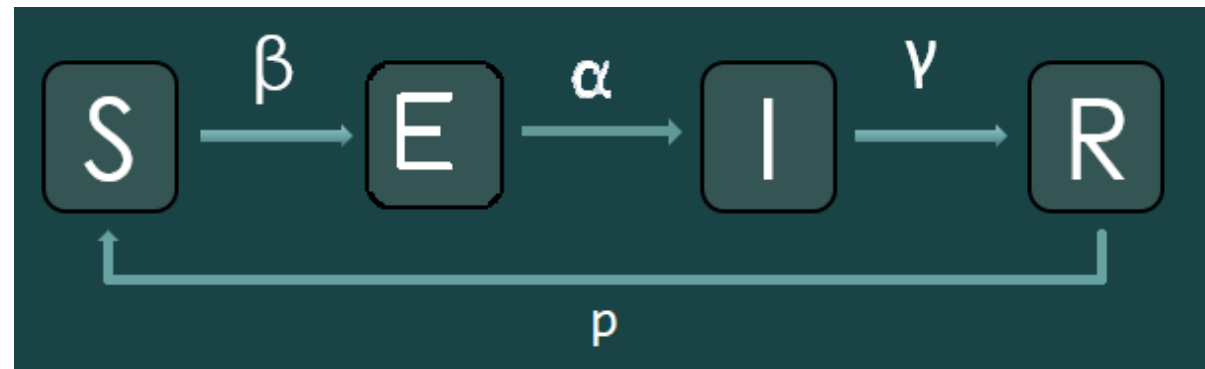
Le taux de transmission est de 0,008, le taux de guérison est de 1/10 et le taux d'incubation de 0,6

Autres modèles particuliers

- SI
- SIS
- SIRS

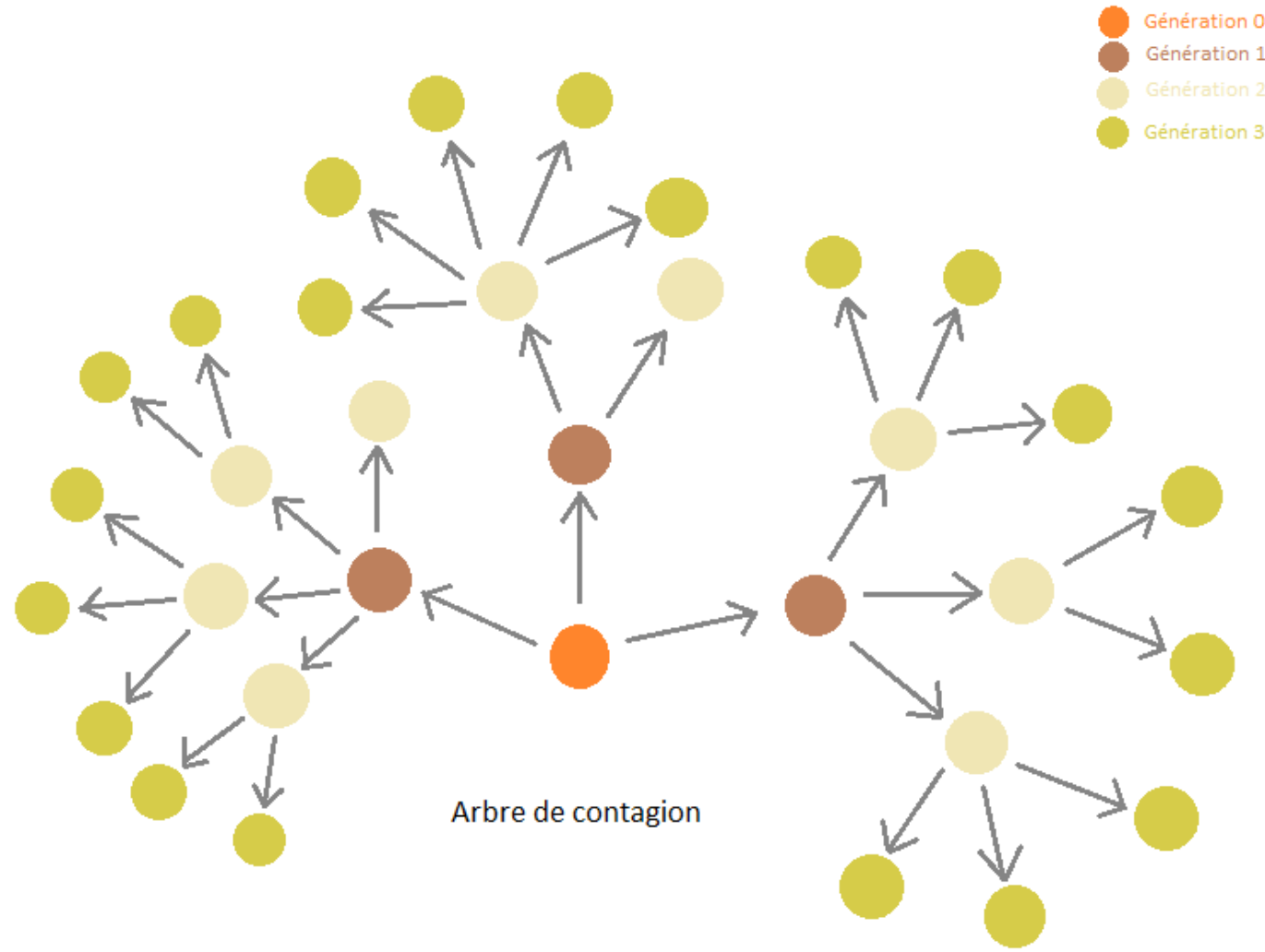


- SEI
- SEIS
- SEIRS



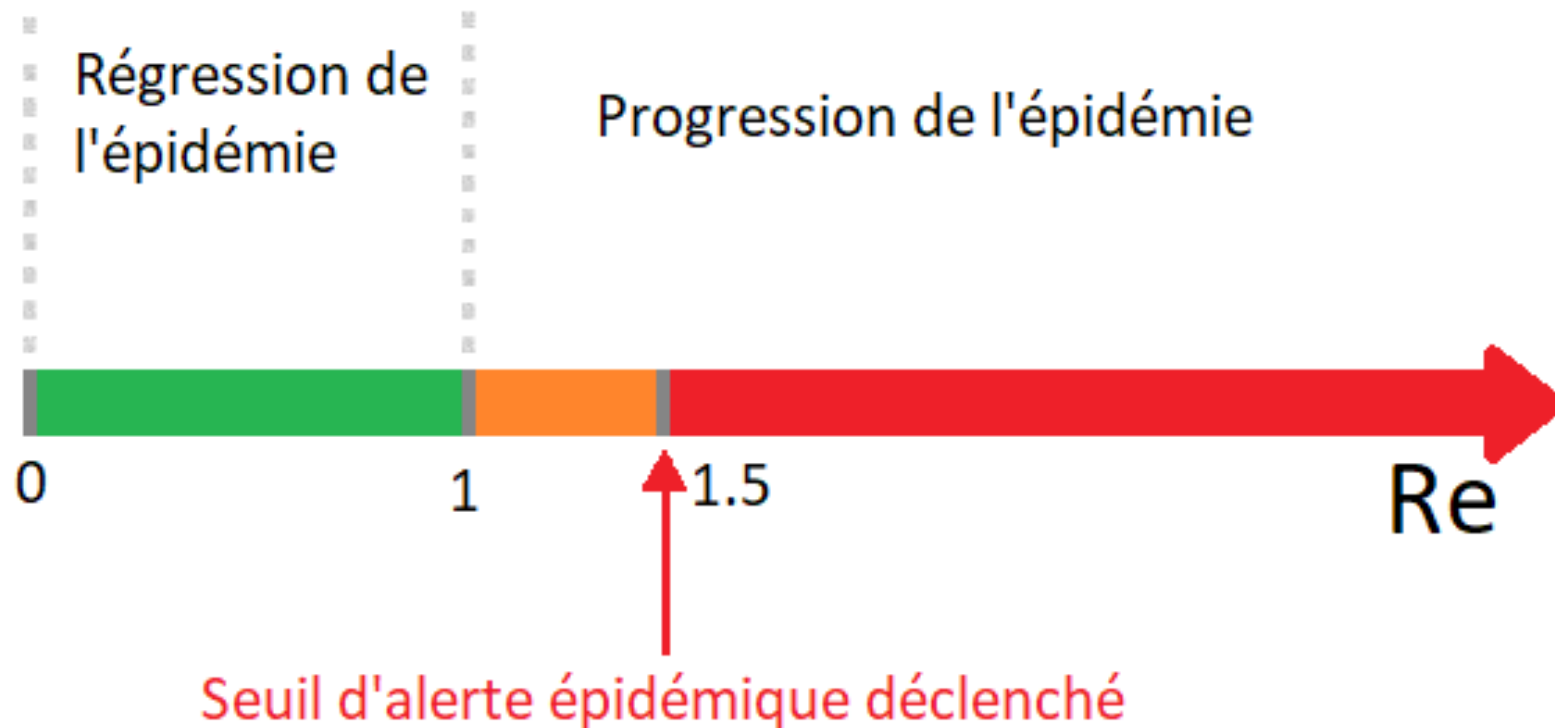
Nombre de reproduction

- R_0 : nombre de reproduction de base
- R_e : nombre de reproduction effectif



Interprétation de R avec le modèle SIR

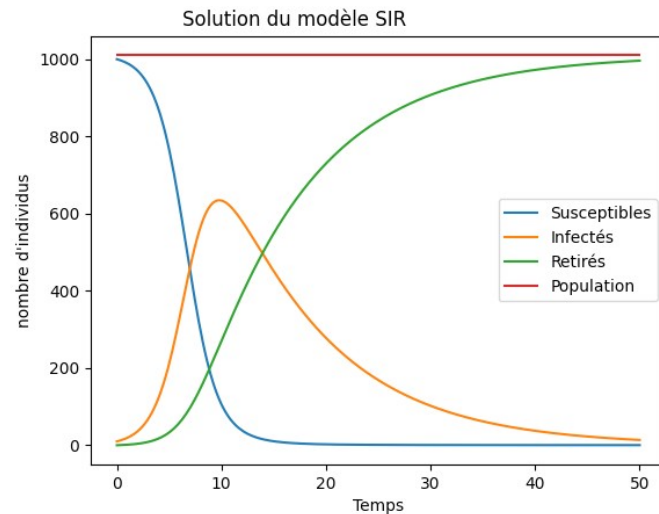
- $R_0 = D \cdot \kappa \cdot \tau = N\beta/\gamma$
 - $Re = S(0)\beta/\gamma$
- $I'(t) = (\beta S(t) - \gamma)I(t) \leq (\beta S(0) - \gamma)I(t) = \gamma(Re - 1)I(t) \leq 0$ pour $Re < 1$



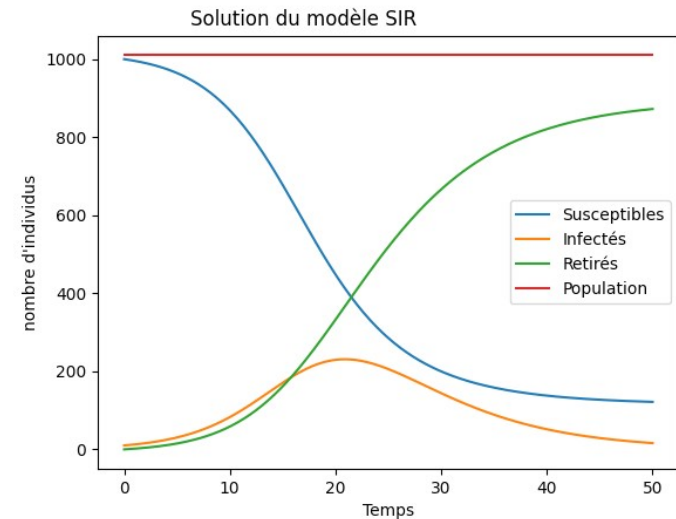
Comment atténuer l'intensité d'une épidémie ?

$$R \propto \beta/\gamma = \beta * D$$

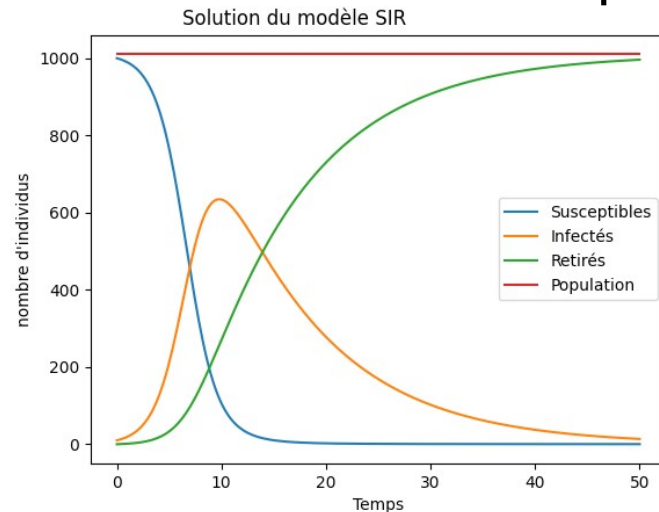
➤ Mesures sanitaires pour faire diminuer β



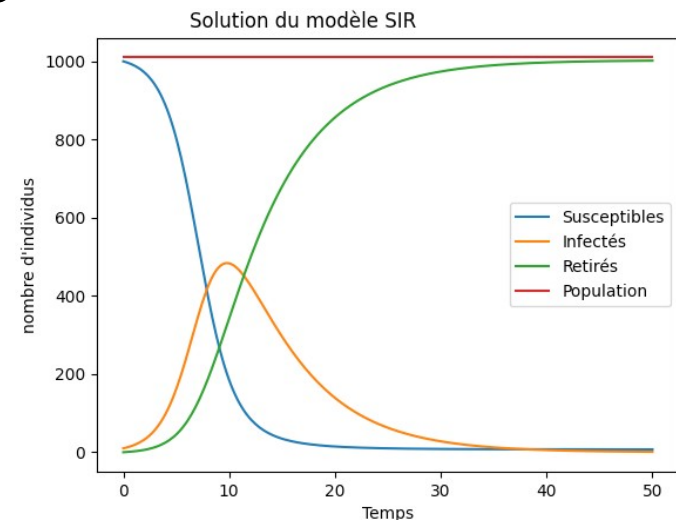
β diminue



➤ Mesures médicales pour faire diminuer D



γ augmente /
 D diminue

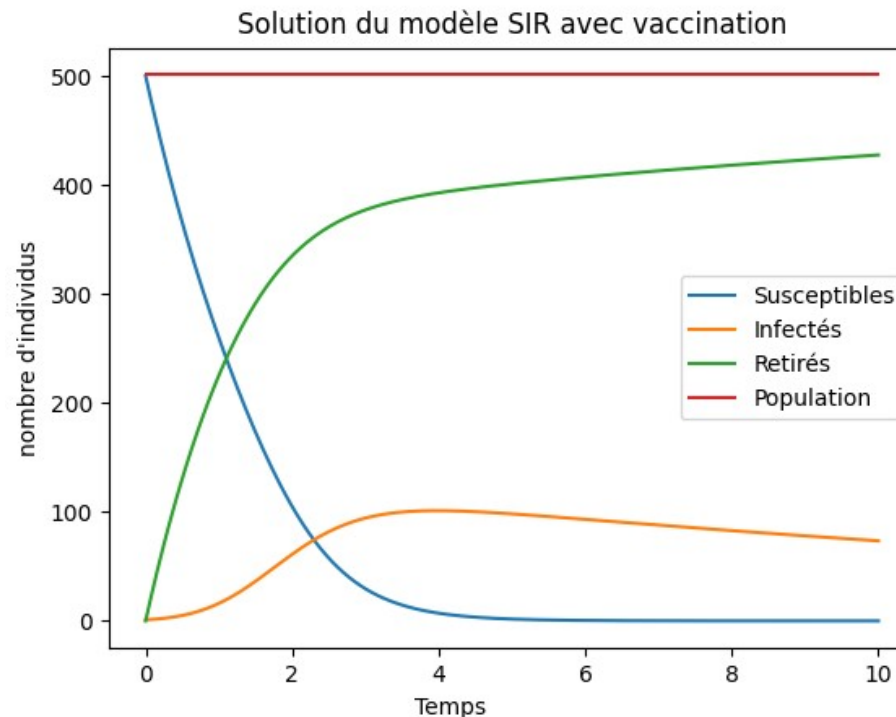


Lien avec la vaccination

$$R = p \cdot R_0$$







- p proportion de susceptibles
- v proportion de vaccinés

Si $p = S(0) = 1$, on a avec la vaccination la nouvelle proportion de susceptibles : $p = 1 - v$
Or il n'y a pas d'épidémie pour $R < 1 \iff (1 - v)R_0 < 1 \iff v > 1 - 1/R_0$



Prédiction de l'évolution de la maladie dans l'espace

- Flux d'individus qui se déplacent et rentrent en contact dans un espace défini

					
Susceptible	Exposé	Infecté	Retiré	Vacciné	Mort

Simulation à l'aide de Python:

Dictionnaires:

- Balles
- Balles mortes

Données utilisées:

- Nombre d'individus
- Nombre de malade initial
- Durée d'infection
- Probabilité d'infection
- Taux de mortalité

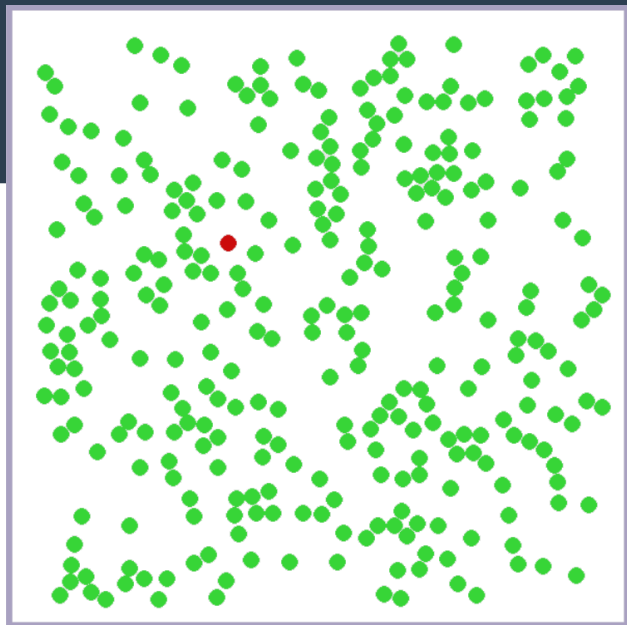
Fonctions:

- Somme vectorielle
- Produit scalaire
- Rebond au contact du contour d'une balle
- Collision entre balles
- Mouvement des balles
- Affichage des balles

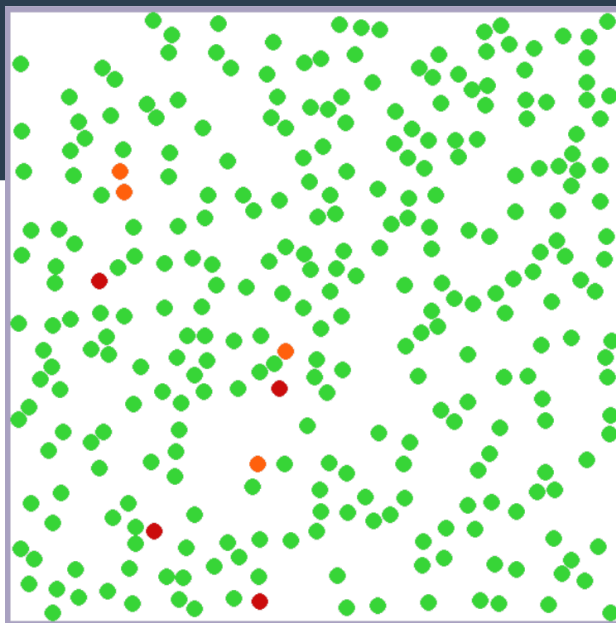
Classes:

- Balle
- Bordure

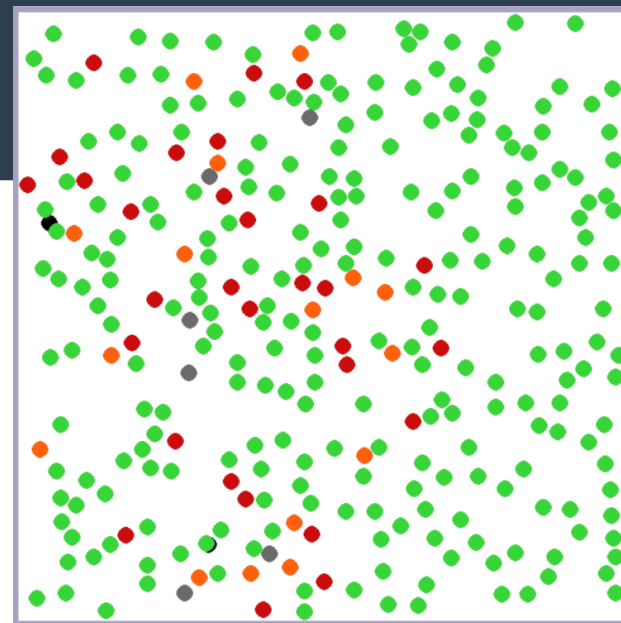
État initial



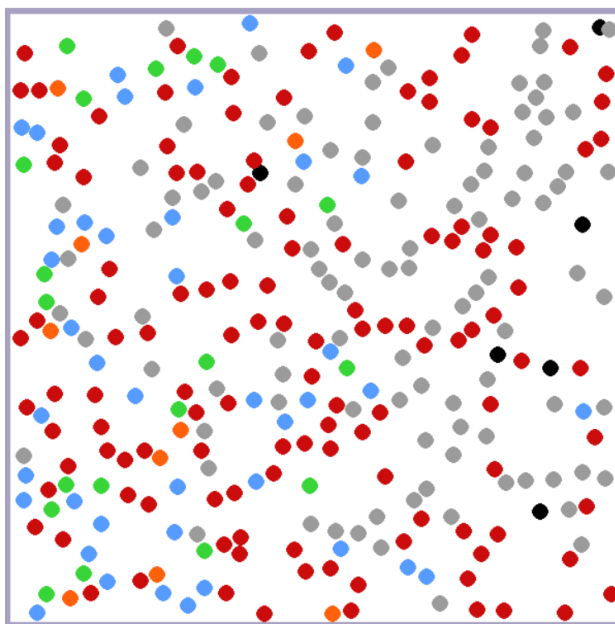
Premières contaminations



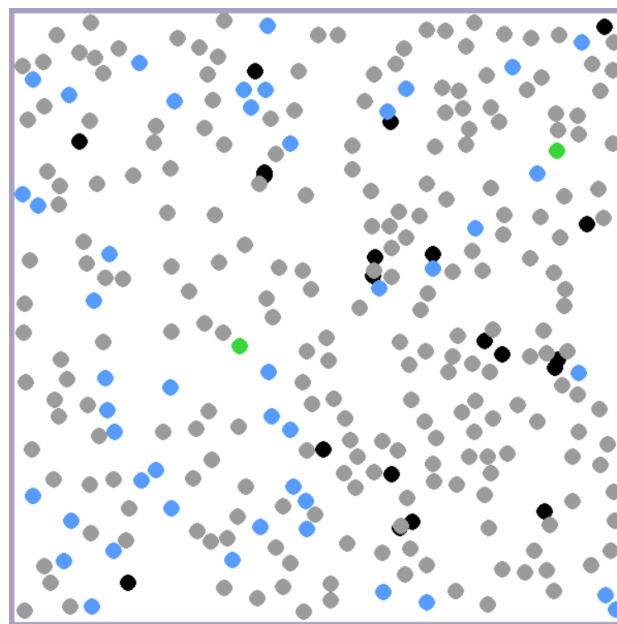
Premiers morts et retirés



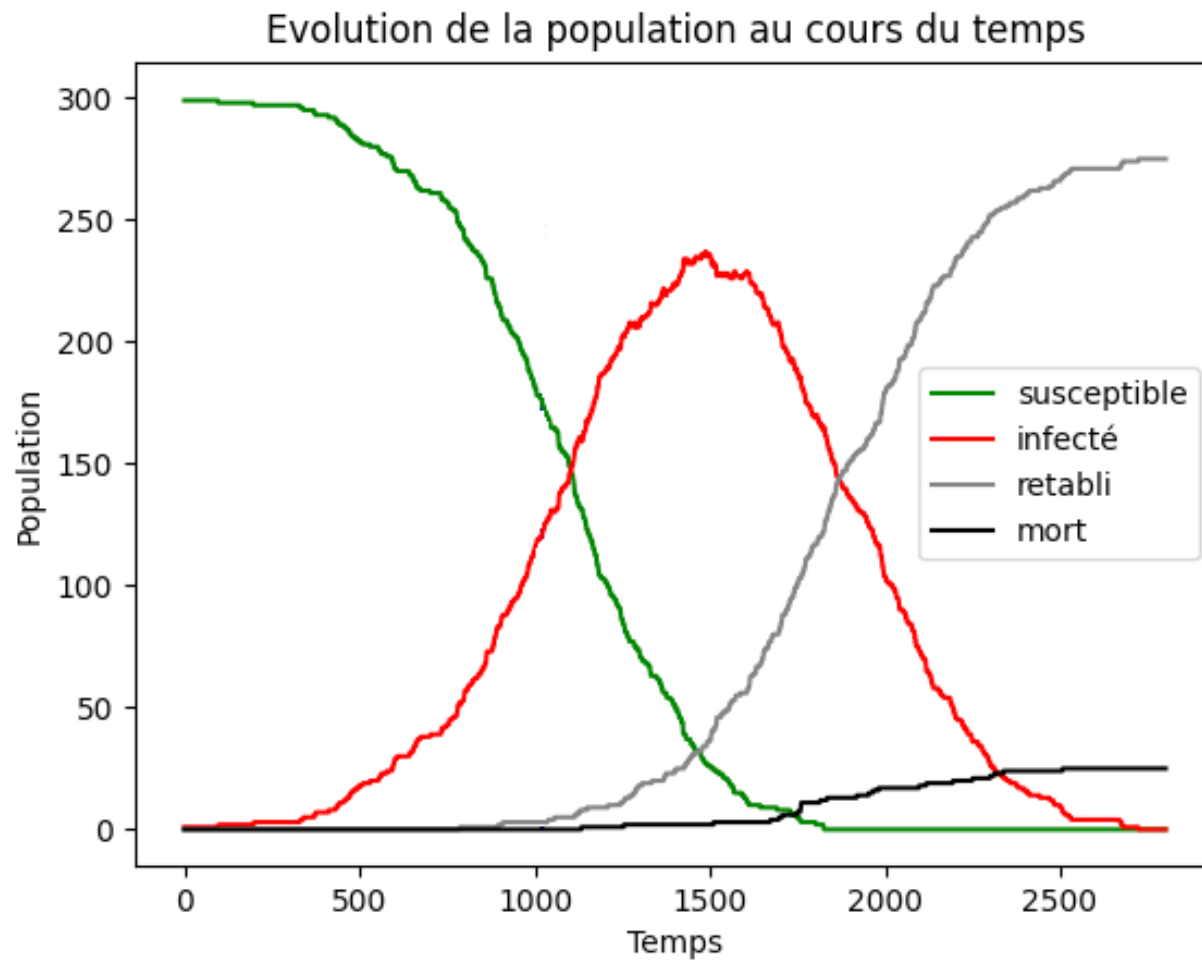
Début de la vaccination



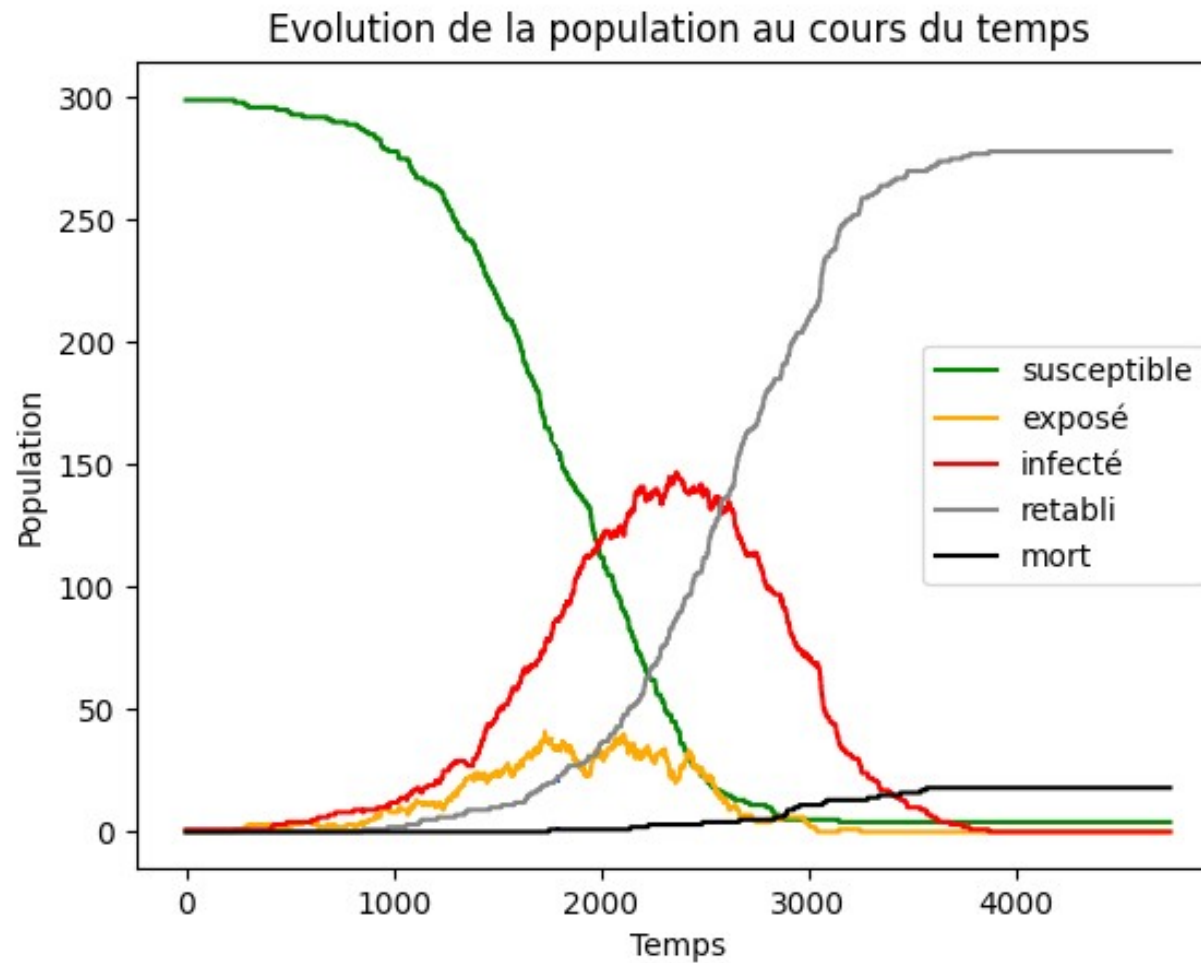
État final



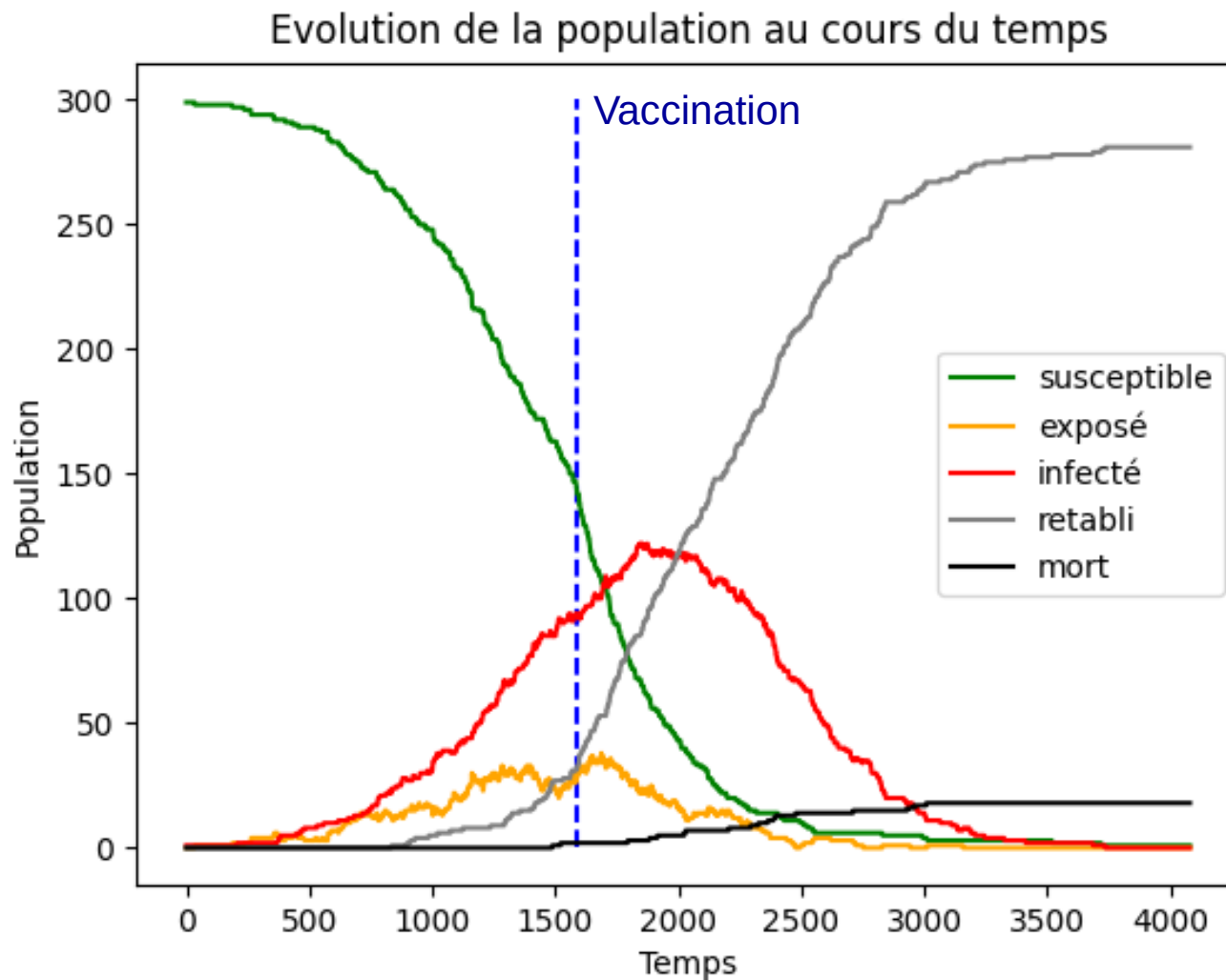
Résultat : SIR



Résultat : SEIR



Résultat global : SEIR + Vaccination



Affiner le modèle

- les délais (durée avant d'être contagieux, durée de contagion, etc.)
- différencier les ages
- la variation des taux dans le temps
- les différents niveaux d'infectiosité
- la prise en compte d'un confinement
- existence d'un vaccin

The background of the slide is a dark blue gradient. It is decorated with several stylized virus particles rendered in a glowing blue wireframe or low-poly mesh style. These particles are scattered across the frame, with some appearing larger and more detailed than others. The central focus is the word 'CONCLUSION' in a large, bold, white sans-serif font.

CONCLUSION

ANNEXES

Modèle SEIR et SIR : méthode Euler explicite

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 # Euler explicite :
5 T = 50 # temps final
6 k = 8
7 h = T/(2**k)
8 t = np.linspace(0,T,2**k+1)
9
10 β=0.008
11 γ=1/10
12 α=0.60
13
14 S0 = 500
15 E0 = 0
16 I0 = 1
17 R0 = 0
18 N0 = S0+E0+I0+R0
19 Y = [[S0,E0,I0,R0,N0]]
20
21 for i in range (2**k):
22     S0,E0,I0,R0,N0 = Y[-1]
23     S1 = S0 + h*(-β*S0*I0)
24     E1 = E0 + h*(β*S0*I0-α*E0)
25     I1 = I0 + h*(α*E0-γ*I0)
26     R1 = R0 + h*(γ*I0)
27     N1=S1+E1+I1+R1
28     Y.append([S1,E1,I1,R1,N1])
29
30 YEulerExpl = np.array(Y)
31
32 S = YEulerExpl[:,0]
33 E = YEulerExpl[:,1]
34 I = YEulerExpl[:,2]
35 R = YEulerExpl[:,3]
36 N = YEulerExpl[:,4]
37 plt.figure(0)
38 plt.plot(t,S,'c',label="Susceptibles")
39 plt.plot(t,E,'m',label="Exposés")
40 plt.plot(t,I,'orange',label="Infectés")
41 plt.plot(t,R,'g',label="Retirés")
42 plt.plot(t,N,'r',label="Population")
43 plt.xlabel("Temps en années")
44 plt.ylabel("nombre d'individus")
45 plt.title("Solution du modèle SEIR")
46 plt.legend()
47 plt.show()
```

```
S0 = 1000
I0 = 10
R0 = 0
N0 = S0+I0+R0
Y = [[S0,I0,R0,N0]]

for i in range (2**k):
    S0,I0,R0,N0 = Y[-1]
    S1 = S0 + h*(-β*S0*I0-α*S0)
    I1 = I0 + h*(β*S0*I0-γ*I0)
    R1 = R0 + h*(γ*I0+α*S0)
    N1=S1+I1+R1
    Y.append([S1,I1,R1,N1])

YEulerExpl = np.array(Y)

S = YEulerExpl[:,0]
I = YEulerExpl[:,1]
R = YEulerExpl[:,2]
N = YEulerExpl[:,3]
plt.figure(0)
plt.plot(t,S,label="Susceptibles")
plt.plot(t,I,label="Infectés")
plt.plot(t,R,label="Retirés")
plt.plot(t,N,label="Population")
plt.xlabel("Temps")
plt.ylabel("nombre d'individus")
plt.title("Solution du modèle SIR avec vaccination")
plt.legend()
plt.show()
```

Simulation

```
1  import pygame
2  import time
3  import math
4  import random
5  import matplotlib.pyplot as plt
6  import numpy as np
7
8
9  # Initialisation de l'affichage
10 pygame.init()
11 clock = pygame.time.Clock()
12
13 resolutionEcran = (600,600)
14 FPS = 60
15 greenColor = (50,210,50)
16 redColor = (200,10,10)
17 greyColor = (150,150,150)
18 whiteColor = (255,255,255)
19 blackColor = (0, 0, 0)
20 blueColor = (80, 150, 255)
21 orangeColor =(255,90,10)
22 arialFontFPS = pygame.font.SysFont("arial", 15)
23 pygame.display.set_caption("TIPE")
24
```



```
25
26 # Initialisation des paramètres du modèle
27 NombreDIndividus = 300
28 NombreMaladeInit_E = 0
29 NombreMaladeInit_I = 1
30 dureeInfectionMin = 600
31 dureeInfectionMax = 1000
32 dureeIncubationMin = 100
33 dureeIncubationMax = 200
34 ProbaInfectionContact = 0.2
35 TauxMortalite = 0.09
36 AvecousansE=False
37 vacc = False
38 TauxVacc = 0
39
40 NbCumuleInfecte = 0
41
42 NbS = NombreDIndividus - NombreMaladeInit_E - NombreMaladeInit_I
43 NbE = NombreMaladeInit_E
44 NbI = NombreMaladeInit_I
45 NbR = 0
46 NbM = 0
47 t = 0
48 listeGeneral = []
49
50 dictBalles = {}
51 dictBallesMortes = {}
52
53 if not AvecousansE:
54     dureeIncubationMin = 0
55     dureeIncubationMax = 0
```

```

56
57 # fonction qui retourne la somme de 2 vecteurs
58 def sommeVect(angle1, taille1, angle2, taille2):
59     dx = math.sin(angle1)*taille1 + math.sin(angle2)*taille2
60     dy = math.cos(angle1)*taille1 + math.cos(angle2)*taille2
61     taille = math.hypot(dx, dy)
62     angle = math.atan2(dx, dy)
63     return (angle, taille, dx, dy)
64
65 # fonction qui retourne le p.s. de 2 vecteurs
66 def produitScalaire(angle1, taille1, angle2, taille2):
67     return taille2*taille1*math.cos(angle1-angle2)
68
69 # Classe Balle qui représente un individu
70 class Balle(pygame.sprite.Sprite):
71     global dictBalles, ProbaInfectionContact, dureeInfection, NbS, NbI, NbR
72     def __init__(self, x, y, idballe, dateFinInfection = 0, dateFinIncubation=0 ,
73     vaMourir = False, masse = 1, rayon = 15, vitesse = 0, angle = 0, couleur =
74     greenColor):
75         super(Balle, self).__init__()
76         dictBalles[idballe] = self
77         self.id = idballe
78         self.sain = True
79         self.invincible = False
80         self.contagieux = False
81         self.dureeInfection = 0
82         self.dateFinInfection = dateFinInfection
83         self.vaMourir = vaMourir
84         self.dateFinIncubation = dateFinIncubation
85         self.scoreInfection = 0
86         self.masse = masse
87         self.rayon = rayon
88         self.x = x
89         self.y = y
90         self.vitesse = vitesse
91         self.angle = angle
92         self.surface = pygame.Surface((self.rayon*2, self.rayon*2), pygame.SRCALPHA)
93         pygame.draw.circle(self.surface, couleur, [self.rayon, self.rayon], self.rayon)
94         self.mask = pygame.mask.from_surface(self.surface)
95         self.rect = self.surface.get_rect()

```

```

94     # Rebond sur le contour
95     def rebondContour(self):
96         if pygame.sprite.collide_mask(self, contour):
97             offx = int(self.x)
98             offy = int(self.y)
99             dx = contour.mask.overlap_area(self.mask, (offx+1,offy)) -
contour.mask.overlap_area(self.mask, (offx-1,offy))
100             dy = contour.mask.overlap_area(self.mask, (offx,offy+1)) -
contour.mask.overlap_area(self.mask, (offx,offy-1))
101             if dx != 0 or dy != 0:
102                 if dy == 0:
103                     if dx > 0:
104                         alpha = math.pi/2
105                     else:
106                         alpha = 3*math.pi/2
107                 else:
108                     alpha = math.atan2(dx,dy)
109
110             self.angle = math.fmod(math.pi + 2*alpha - self.angle, 2*math.pi)
111             angleN = math.atan2(dx,dy)
112             nb = contour.mask.overlap_area(self.mask, (offx,offy))
113
114             self.x -= math.sin(angleN)*(nb/40)
115             self.y -= math.cos(angleN)*(nb/40)
116
117             self.vitesse = self.vitesse
118             return True
119         return False
120     return False

```



```

121 # Rebond entre les balles
122 @staticmethod # méthode propre à la classe entière, pas associée à une Balles
123 def collision(dictBalles):
124     global NbS, NbE, NbI, NbCumuleInfecte
125
126     couplesballes = []
127     for couple in enumerate(dictBalles):
128         couplesballes.append(couple)
129
130     # On parcourt tous les couples de balles
131     for i in range(len(couplesballes)):
132         for j in range(i+1, len(couplesballes)):
133             b1 = dictBalles[couplesballes[i][1]]
134             b2 = dictBalles[couplesballes[j][1]]
135             dx = b1.x + b1.rayon - b2.x - b2.rayon
136             dy = b1.y + b1.rayon - b2.y - b2.rayon
137             distance = math.hypot(dx, dy)
138             # On test si le couple de balle (b1,b2) se touche
139             if distance < b1.rayon + b2.rayon:
140                 # traitement de la collision
141                 angleNormal = math.atan2(dx, dy)
142                 masseTotale = b1.masse + b2.masse
143                 v1nTaille = produitScalaire(b1.angle, b1.vitesse, angleNormal, 1)
144                 v2nTaille = produitScalaire(b2.angle, b2.vitesse, angleNormal, 1)
145                 v1nTailleP = (v1nTaille*(b1.masse - b2.masse) +
146                             2*b2.masse*v2nTaille)/masseTotale
147                 v2nTailleP = (v2nTaille*(b2.masse - b1.masse) +
148                             2*b1.masse*v1nTaille)/masseTotale
149                 (v1tAngle, v1tTaille, dxt1, dyt1) = sommeVect(b1.angle,
150                     b1.vitesse, angleNormal, -v1nTaille)
151                 (v2tAngle, v2tTaille, dxt2, dyt2) = sommeVect(b2.angle,
152                     b2.vitesse, angleNormal, -v2nTaille)

```

```

150     (b1.angle, b1.vitesse, dx1, dy1) = sommeVect(v1tAngle,
151     v1tTaille, angleNormal, v1nTailleP)
152     (b2.angle, b2.vitesse, dx2, dy2) = sommeVect(v2tAngle,
153     v2tTaille, angleNormal, v2nTailleP)
154
155     depassement = (b1.rayon + b2.rayon - distance)/2
156     b1.x += math.sin(angleNormal)*depassement
157     b1.y += math.cos(angleNormal)*depassement
158     b2.x -= math.sin(angleNormal)*depassement
159     b2.y -= math.cos(angleNormal)*depassement
160     # fin de traitement de la collision
161     # traitement de la transmission de la maladie
162     if start:
163         if b1.sain and b2.contagieux:
164             if random.random() < ProbaInfectionContact:
165                 b2.scoreInfection += 1
166                 NbS -= 1
167                 NbE += 1
168                 # changement de couleur de b1
169                 b1.surface = pygame.Surface((b1.rayon*2,
170                 b1.rayon*2),pygame.SRCALPHA)
171                 pygame.draw.circle(b1.surface, orangeColor,
172                 [b1.rayon,b1.rayon], b1.rayon)
173                 b1.mask = pygame.mask.from_surface(b1.surface)
174                 # fin changement de couleur de b1
175                 b1.sain = False
176                 b1.dateFinIncubation =
177                 random.randint(dureeIncubationMin,dureeIncubationMax)
178                 b1.dateFinInfection =
179                 random.randint(dureeInfectionMin,dureeInfectionMax)
180                 if random.random() < TauxMortalite:
181                     b1.vaMourir = True

```

```

176         elif b2.sain and b1.contagieux:
177             if random.random() < ProbaInfectionContact:
178                 b1.scoreInfection += 1
179                 NbS -= 1
180                 NbE += 1
181                 # changement de couleur de b2
182                 b2.surface = pygame.Surface((b2.rayon*2,
183                 b2.rayon*2),pygame.SRCALPHA)
184                 pygame.draw.circle(b2.surface, orangeColor,
185                 [b2.rayon,b2.rayon], b2.rayon)
186                 b2.mask = pygame.mask.from_surface(b2.surface)
187                 # fin changement de couleur de b2
188                 b2.sain = False
189                 b2.dateFinIncubation =
190                 random.randint(dureeIncubationMin,dureeIncubationMax)
191                 b2.dateFinInfection =
192                 random.randint(dureeInfectionMin,dureeInfectionMax)
193                 if random.random() < TauxMortalite:
194                     b2.vaMourir = True
195
196     # deplacement des balles et mise a jour de l'etat de la balle
197     def move(self):
198         global NbS, NbE, NbI, NbR, NbM, TauxVacc, vacc
199         self.rebondContour()
200         dx = math.sin(self.angle)*self.vitesse
201         dy = math.cos(self.angle)*self.vitesse
202         self.x += dx*60/FPS
203         self.y += dy*60/FPS

```



```

199     ## vaccination
200     if vacc:
201         if self.sain:
202             if random.random() < TauxVacc:
203                 NbS-=1
204                 NbR+=1
205                 self.surface = pygame.Surface((self.rayon*2,
206                 self.rayon*2),pygame.SRCALPHA)
207                 pygame.draw.circle(self.surface, blueColor,
208                 [self.rayon,self.rayon], self.rayon)
209                 self.mask = pygame.mask.from_surface(self.surface)
210                 self.invincible = True
211                 self.sain = False
212     ## maladie
213     if not self.sain and not self.invincible: # on incremente de temps passé
214     malade
215         # on regarde si la balle est à la fin de l'incubation / maladie
216         if self.dureeInfection == self.dateFinIncubation:
217             NbE-=1
218             NbI+=1
219             self.contagieux = True
220             self.surface = pygame.Surface((self.rayon*2,
221             self.rayon*2),pygame.SRCALPHA)
222             pygame.draw.circle(self.surface, redColor, [self.rayon,self.rayon],
223             self.rayon)
224             self.mask = pygame.mask.from_surface(self.surface)
225         elif self.dureeInfection > self.dateFinInfection:
226             self.dureeInfection = 0
227             self.contagieux = False

```

```

223     # on regarde si la balle va mourir à la fin de sa maladie
224     if self.vaMourir:
225         self.surface = pygame.Surface((self.rayon*2,
226                                         self.rayon*2),pygame.SRCALPHA)
227         pygame.draw.circle(self.surface, blackColor,
228                             [self.rayon,self.rayon], self.rayon)
229         self.mask = pygame.mask.from_surface(self.surface)
230         dictBallesMortes[self.id] = self
231         NbI -= 1
232         NbM += 1
233         return self.id
234     else:
235         self.surface = pygame.Surface((self.rayon*2,
236                                         self.rayon*2),pygame.SRCALPHA)
237         pygame.draw.circle(self.surface, greyColor,
238                             [self.rayon,self.rayon], self.rayon)
239         self.mask = pygame.mask.from_surface(self.surface)
240         self.invincible = True
241         NbI -= 1
242         NbR += 1
243         self.dureeInfection += 1
244         return False
245
246 # affichage
247 def draw(self, surface):
248     if math.hypot(self.x - resolutionEcran[0]/2, self.y - resolutionEcran[1]/2)
249     > 2000:
250         self.x = resolutionEcran[0]/2
251         self.y = resolutionEcran[1]/2
252     self.rect = pygame.Rect(self.x, self.y, self.rayon*2, self.rayon*2)
253     surface.blit(self.surface, [self.x,self.y])

```



```

250 # Classe bordure
251 class Bordure(pygame.sprite.Sprite):
252     def __init__(self):
253         super(Bordure, self).__init__()
254         self.surface = pygame.Surface(resolutionEcran, pygame.SRCALPHA)
255         pygame.draw.rect(self.surface, (164,156,189), pygame.Rect(0, 0,
            resolutionEcran[0], resolutionEcran[1]), width = 10)
256         self.mask = pygame.mask.from_surface(self.surface)
257         self.x = 0
258         self.y = 0
259         self.rect = self.surface.get_rect()
260
261     def reload(self):
262         self.surface = pygame.Surface(resolutionEcran, pygame.SRCALPHA)
263         pygame.draw.rect(self.surface, (164,156,189), pygame.Rect(0, 0,
            resolutionEcran[0], resolutionEcran[1]), width = 10)
264         self.mask = pygame.mask.from_surface(self.surface)
265         self.x = 0
266         self.y = 0
267         self.rect = self.surface.get_rect()
268
269     def draw(self, surface):
270         self.rect = pygame.Rect(self.x, self.y, resolutionEcran[0],
            resolutionEcran[1])
271         surface.blit(self.surface, [0, 0])
272
273

```

```

274 # Initialisation de la creation/placement des balles
275 for k in range(1,NombreDIndividus+1):
276     x = random.random()*(resolutionEcran[0]-60)+30
277     y = random.random()*(resolutionEcran[1]-60)+30
278     vitesse = 1
279     angle = 2*math.pi*random.random()
280     if NombreMaladeInit_I > 0:
281         NombreMaladeInit_I -= 1
282         dateFinInfection = random.randint(dureeInfectionMin,dureeInfectionMax)
283         vaMourir = False
284         if random.random() < TauxMortalite:
285             vaMourir = True
286         balle = Balle(x, y, idballe=k, dateFinInfection = dateFinInfection,
287             dateFinIncubation=-1, vaMourir= vaMourir, masse = 1, rayon = 8, couleur =
288             redColor, vitesse=vitesse, angle=angle)
289         balle.sain = False
290         balle.contagieux = True
291     elif NombreMaladeInit_E > 0:
292         NombreMaladeInit_E -= 1
293         dateFinIncubation = random.randint(dureeIncubationMin,dureeIncubationMax)
294         dateFinInfection = random.randint(dureeInfectionMin,dureeInfectionMax)
295         vaMourir = False
296         if random.random() < TauxMortalite:
297             vaMourir = True
298         balle = Balle(x, y, idballe=k, dateFinInfection = dateFinInfection,
299             dateFinIncubation = dateFinIncubation, vaMourir= vaMourir, masse = 1, rayon
300             = 8, couleur = orangeColor, vitesse=vitesse, angle=angle)
301         balle.sain = False
302         balle.contagieux = False
303     else:
304         balle = Balle(x, y, idballe=k, masse = 1, rayon = 8, vitesse=vitesse,
305             angle=angle)
306
307 contour = Bordure()

```

```
304 windowSurface = pygame.display.set_mode(resolutionEcran, pygame.RESIZABLE)
305
306
307 running = True
308 start = False
309 end = False
310 affiche = False
311 # Boucle principale
312 while running:
313     for event in pygame.event.get():
314         if event.type == pygame.QUIT:
315             running = False
316         elif event.type == pygame.KEYDOWN:
317             if event.key == pygame.K_RETURN and not start:
318                 print("Start !")
319                 start = True
320             elif event.key == pygame.K_RETURN and not end:
321                 print("End !")
322                 end = True
323                 affiche = True
324
```

```

325 # condition vaccination
326 if (NbE + NbI) > NombreDIndividus*0.4 and not vacc:
327     vacc = True
328     tVacc = t
329
330 if start and not end:
331     listeGeneral.append([t, NbS, NbE, NbI, NbR, NbM])
332     t += 1
333     listeIdMort = []
334     for idb in dictBalles:
335         b = dictBalles[idb]
336         idmort = b.move()
337         if idmort:
338             listeIdMort.append(idmort)
339     for idb in listeIdMort:
340         del dictBalles[idb]
341
342 Balle.collition(dictBalles)
343
344 windowSurface.fill(whiteColor)
345
346 for idb in dictBallesMortes:
347     b = dictBallesMortes[idb]
348     b.draw(windowSurface)
349
350 for idb in dictBalles:
351     b = dictBalles[idb]
352     b.draw(windowSurface)
353
354
355

```



```

356     contour.draw(windowSurface)
357     pygame.display.flip()
358     clock.tick(FPS)
359
360     if affiche:
361         affiche = False
362
363         SumR = 0
364         SumCumuleInfecte = 0
365         for idb in dictBalles:
366             b = dictBalles[idb]
367             SumR += b.scoreInfection
368         for idb in dictBallesMortes:
369             b = dictBallesMortes[idb]
370             SumR += b.scoreInfection
371         R_moy = SumR / NombreDIndividus
372         print(f"Le R moyen vaut : {R_moy}")
373
374         listeGeneral = np.array(listeGeneral)
375         plt.figure()
376         plt.plot(listeGeneral[:,0],listeGeneral[:,1],c="g",label="susceptible")
377         if AvecousansE:
378             plt.plot(listeGeneral[:,0],listeGeneral[:,2],c="orange",label="exposé")
379             plt.plot(listeGeneral[:,0],listeGeneral[:,3],c="r",label="infecté")
380             plt.plot(listeGeneral[:,0],listeGeneral[:,4],c="grey",label="retabli")
381             plt.plot(listeGeneral[:,0],listeGeneral[:,5],c="black",label="mort")
382         plt.title("Evolution de la population au cours du temps")
383         plt.xlabel('Temps')
384         plt.ylabel('Population')
385         if vacc and TauxVacc !=0:
386             plt.vlines(tVacc, ymin=0, ymax=NombreDIndividus, colors="b",
387                        linestyle="dashed")
387         plt.legend()
388         plt.savefig("evolution.png")

```