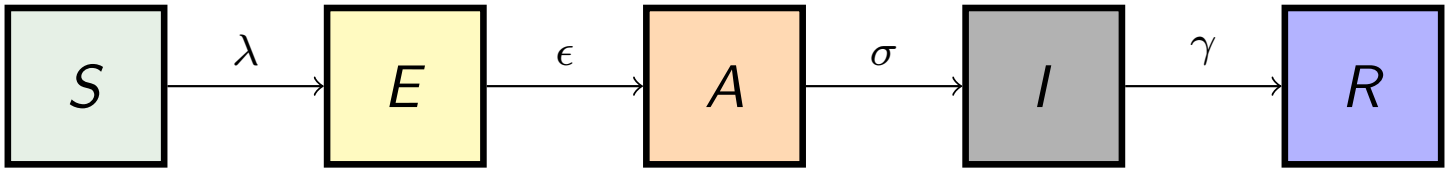
Étude du modèle SEAIR dans le cadre d'une épidémie à plusieurs vagues et à plusieurs variants.

Joris ROUSERE
15890

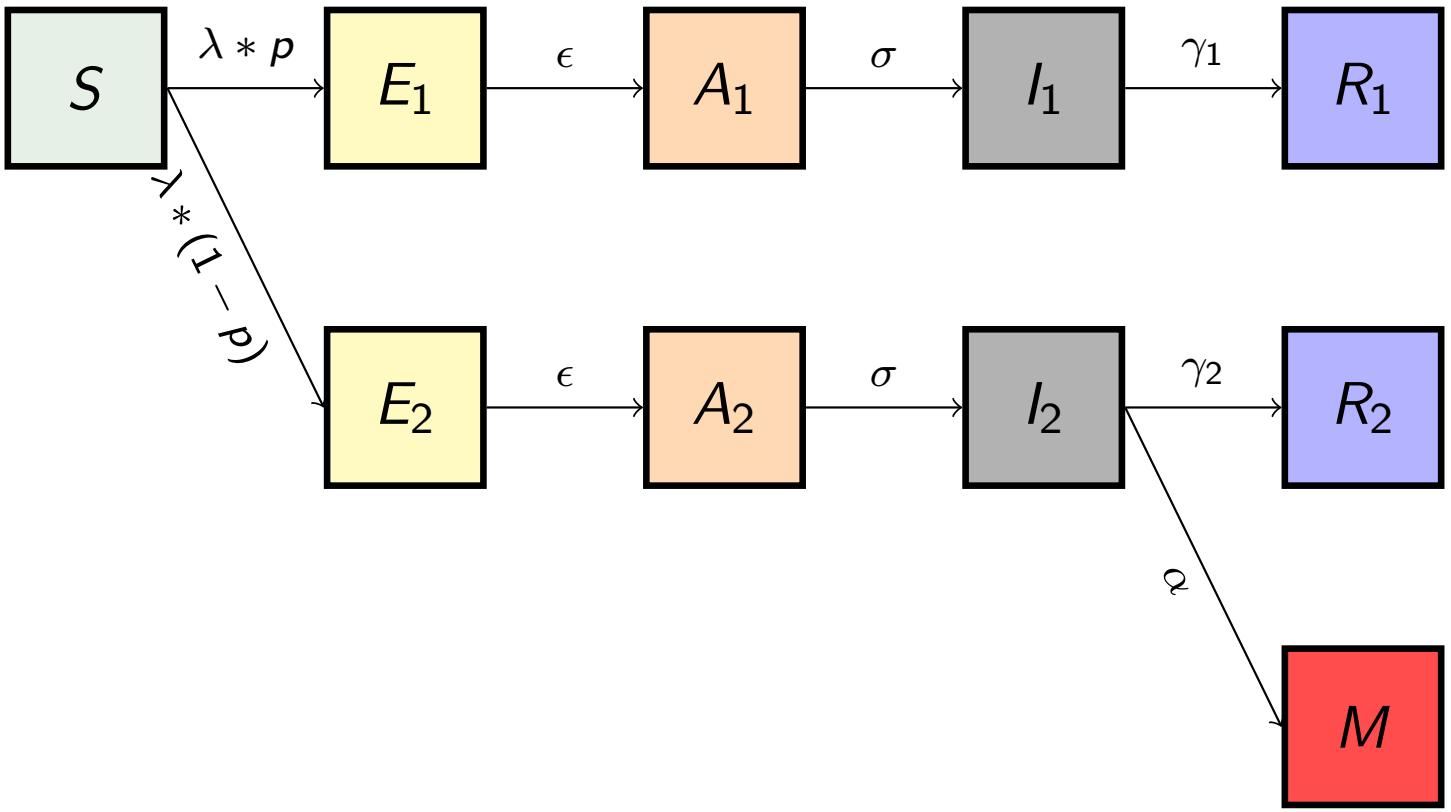
Objectifs

- Appliquer le modèle SEAIR
- Comparer le modèle aux résultats expérimentaux sur :
  1. La présence des différentes vagues
  2. Les instants des différents pics épidémiologiques
  3. L'amplitude de ces pics

Présentation du modèle



Présentation du modèle



Hypothèses

- Restriction à 7000 Personnes (Petite ville)
- Population uniforme
- Aucune naissance / Aucun décès hors COVID-19

Équations différentielles

Système d'équations différentielles

$$\frac{dS}{dt} = -\lambda S + r(R_1 + R_2)$$
$$\frac{dE_1}{dt} = p\lambda S - \epsilon E_1$$
$$\frac{dA_1}{dt} = \epsilon E_1 - \sigma A_1$$
$$\frac{dI_1}{dt} = \sigma A_1 - \gamma_1 I_1$$
$$\frac{dR_1}{dt} = \gamma_1 I_1 - rR_1$$
$$\frac{dM}{dt} = \alpha I_2$$
$$\frac{dE_2}{dt} = (1-p)\lambda S - \epsilon E_2$$
$$\frac{dA_2}{dt} = \epsilon E_2 - \sigma A_2$$
$$\frac{dI_2}{dt} = \sigma A_2 - (\gamma_2 + \alpha) I_2$$
$$\frac{dR_2}{dt} = \gamma_2 I_2 - rR_2$$

Système non linéaire :

$$\lambda = \lambda(A_1, I_1, A_2, I_2)$$

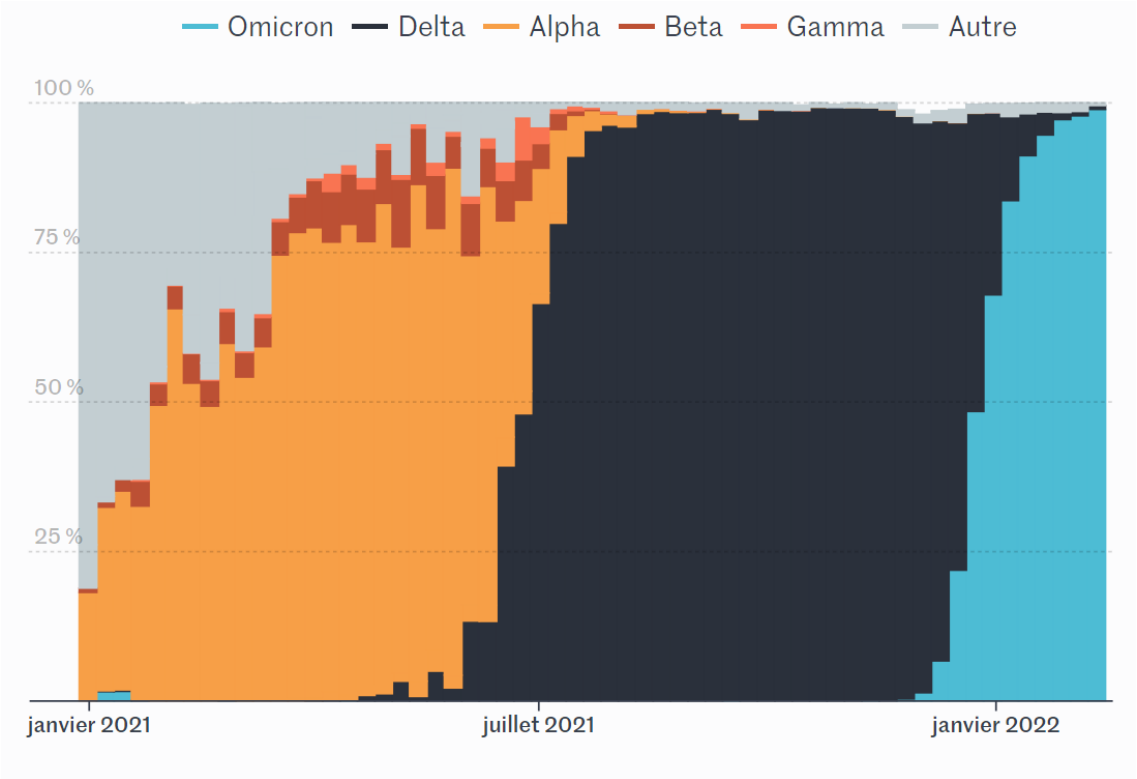
Méthode d'Euler

```
def day(t, epsilon, sigma, gamma1, gamma2, alpha,
        S, E1, A1, I1, R1, M, E2, A2, I2, R2, b, retour =1/180, S0=S0base ):


    cstes = constante(t, A1, I1, A2, I2, gamma1, sigma, alpha, gamma2)
    l=cstes[1]
    p=cstes[2]

    R1 += I1*gamma1
    I1 -= I1*gamma1
    I1 += sigma*A1
    A1 -= A1*sigma
    A1 += E1*epsilon
    E1 -= E1*epsilon
    E1 += S*l*p
    M += I2*alpha
    R2 += I2*gamma2
    I2 -= I2*alpha
    I2 -= I2*gamma2
    I2 += sigma*A2
    A2 -= A2*sigma
    A2 += E2*epsilon
    E2 -= E2*epsilon
    E2 += S*l*(1-p)
    S -= S*l
    S+=R1*retour
    S+=R2*retour
    R1-=R1*retour
    R2-=R2*retour

    return(S,E1,A1,I1,R1,M,E2,A2,I2,R2)
```

Conditions initiales

- $S = 6999$
- $E_1 = 0$
- $A_1 = 0$
- $I_1 = 1$
- $R_1 = 0$
- $M = 0$
- $E_2 = 0$
- $A_2 = 0$
- $I_2 = 0$
- $R_2 = 0$

Variants



Année 2020 : Variant initial

Janvier 2021 - Mai 2021 : Variant Alpha

Juin 2021 - Novembre 2021 : Variant Delta

A partir de Décembre 2021 : Variant Omicron

source : lemonde.fr

Variants

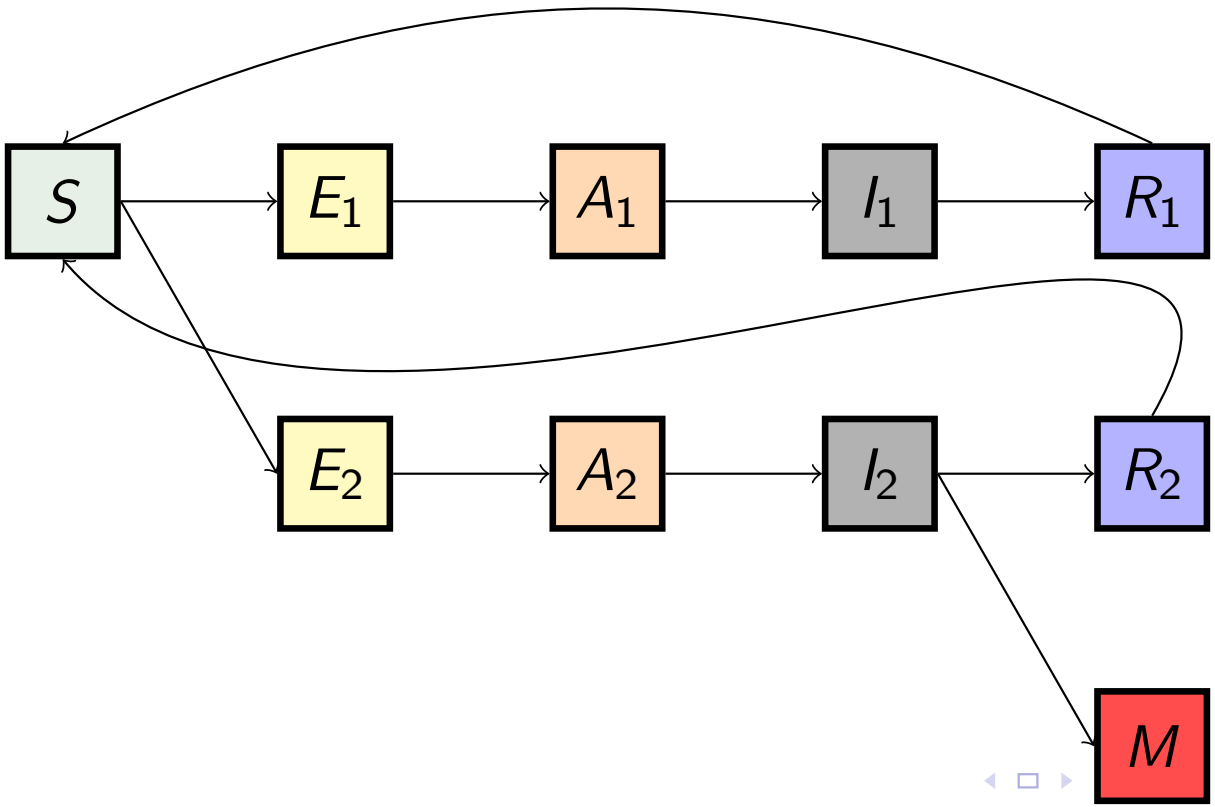
Quatre variants : quatre R_0

- Février 2020 - Décembre 2020: $R_0 = 2.8$ *(journaldesfemmes.fr)*
- Janvier 2021 - Mai 2021 : $R_0 = 4.5$ *(journaldesfemmes.fr)*
- Juin 2021 - Novembre 2021 : $R_0 = 6.6$ *(journaldesfemmes.fr)*
- A partir de Décembre 2022 : $R_0 = 12$ *(theconversation.com)*

Boucle de retour

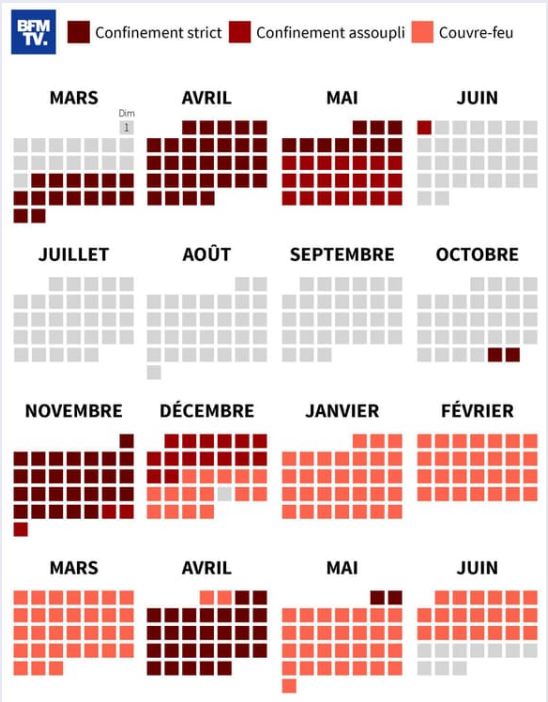
Boucle de retour

- On définit $r = 180^{-1}$



Politique de contrôle

Des confinements et couvres feux



- Confinements :
  $c = 0.84$ *(20minutes.fr)*
- Couvres feux :
  $c = 0.35$ *(radiofrance.fr)*

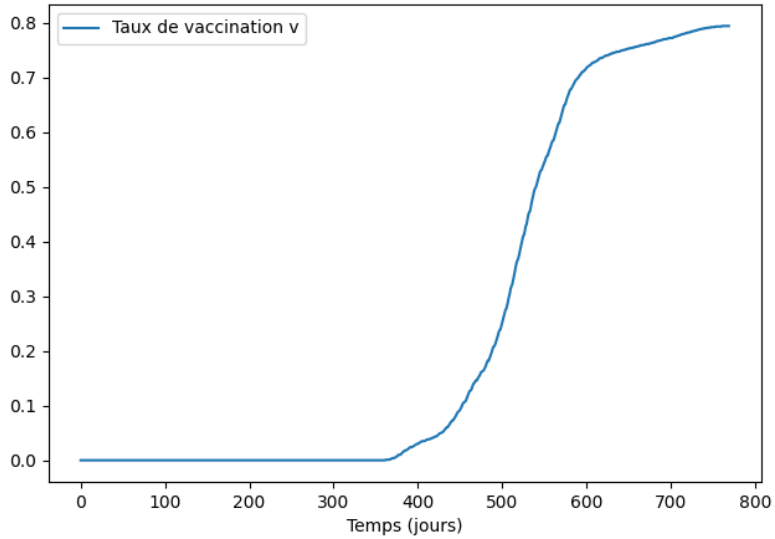
source : bfmtv.com

Vaccination

Hypothèses

- On ne tient pas compte des différents rappels
- La vaccination ne change pas la probabilité d'infection mais immunise des formes graves

fra	jour	n_dose1	n_complet	n_rappel	n_cum_dose	n_cum_com	n_cum_rapp	couv_dose1	couv_
FR	12/03/2021	294746	62784	0	5123113	2318050	0	7.6	3.5
FR	13/03/2021	231351	21269	0	5354464	2339319	0	8.0	3.5
FR	14/03/2021	63327	5458	0	5417791	2344777	0	8.1	3.5
FR	15/03/2021	158572	37151	0	5576363	2381928	0	8.3	3.5
FR	16/03/2021	141516	43549	0	5717879	2425477	0	8.5	3.6
FR	17/03/2021	150400	43369	0	5868279	2468846	0	8.7	3.7
FR	18/03/2021	167746	50057	0	6036025	2518903	0	9.0	3.8
FR	19/03/2021	229931	44677	0	6265956	2563580	0	9.3	3.8
FR	20/03/2021	166627	16329	0	6432583	2579909	0	9.6	3.8
FR	21/03/2021	49897	4359	0	6482480	2584268	0	9.7	3.9
FR	22/03/2021	169934	39645	0	6652414	2623913	0	9.9	3.9
FR	23/03/2021	250285	50198	0	6902699	2674111	0	10.3	4.0
FR	24/03/2021	256819	51819	0	7159518	2725930	0	10.7	4.1
FR	25/03/2021	341723	55647	0	7501241	2781577	0	11.2	4.1
FR	26/03/2021	358727	53569	0	7859968	2835146	0	11.7	4.2
FR	27/03/2021	195221	23094	0	8055189	2858240	0	12.0	4.3
FR	28/03/2021	59310	5917	0	8114499	2864157	0	12.1	4.3
FR	29/03/2021	202908	60846	0	8317407	2925003	0	12.4	4.4
FR	30/03/2021	285362	79888	0	8602769	3004891	0	12.8	4.5
FR	31/03/2021	262928	83497	0	8865697	3088388	0	13.2	4.6
FR	01/04/2021	315313	101581	0	9181010	3189969	0	13.7	4.8
FR	02/04/2021	281246	103886	0	9462256	3293855	0	14.1	4.9
FR	03/04/2021	141029	66995	0	9603285	3360850	0	14.3	5.0
FR	04/04/2021	46306	20705	0	9649591	3381555	0	14.4	5.0



source : data.gouv.fr

Liste des variables

Définies expérimentalement

- $\epsilon = 0.25$ *(quebec.ca)*
- $\gamma_1 = 0.1$ *(futura-sciences.com)*
- $\sigma = 0.5$ *(qare.fr)*
- $a = 0.18125$: rapport décès/guérison des cas sévères *(medrxiv.org)*

- $\alpha = \gamma_1 * a = 0.018125$
- $\gamma_2 = \gamma_1 * (1 - a) = 0.081875$

Calcul : Préambule

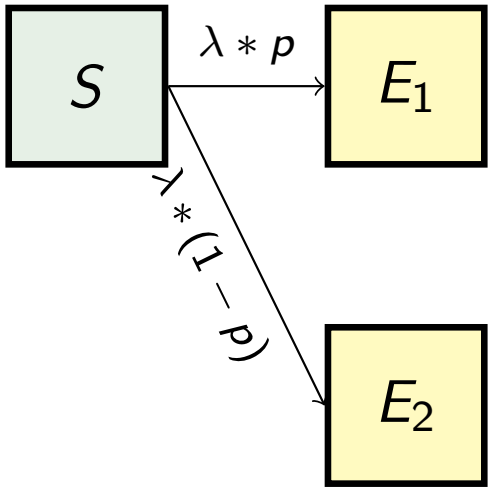
On a modélisé

- R_0
- L'impact de la politique de contrôle : c
- Le taux de vaccination : v

On cherche

- β : taux de transmission
- λ : taux d'infection
- p : taux de formes graves

Détermination : p



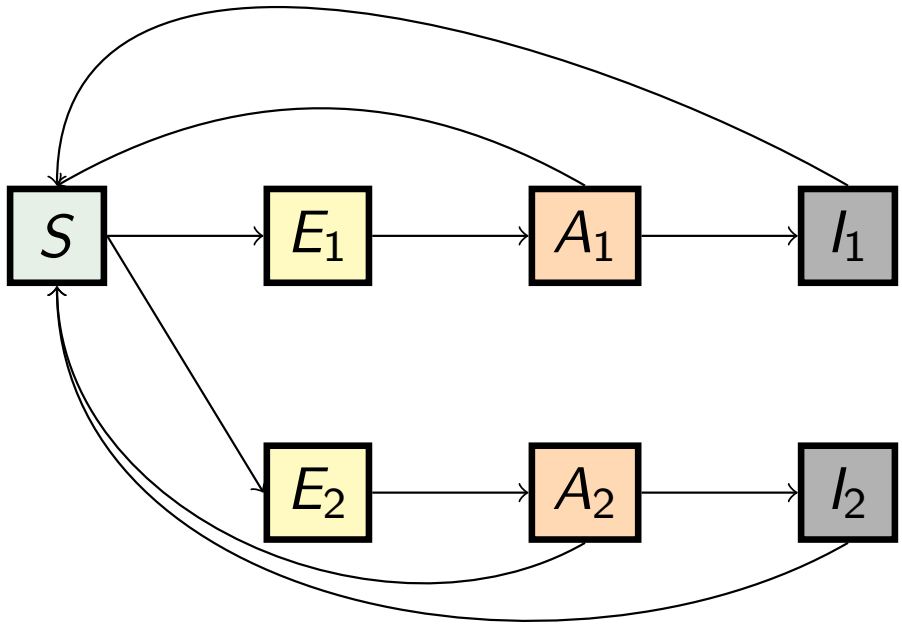
Taux de formes graves sans vaccination : 3.2 % *(medrxiv.org)*

Rappel : v : taux de vaccination

On obtient : $p = 1 - 0.032 * (1 - v)$

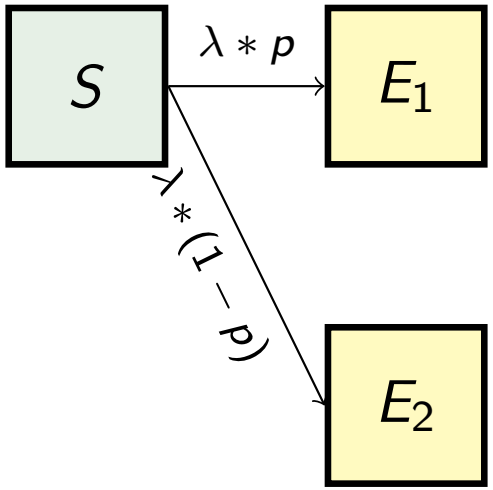
Détermination : β

On pose $b = 0.2$: diminution contagiosité en milieu hospitalier



$$\beta = \frac{\gamma_1 * \sigma * R_0 * (\alpha + \gamma_2)}{S_0 * (\gamma_1 + p * \sigma) * (\alpha + \gamma_2) + b * \gamma_1 * \sigma * (1 - p))}$$

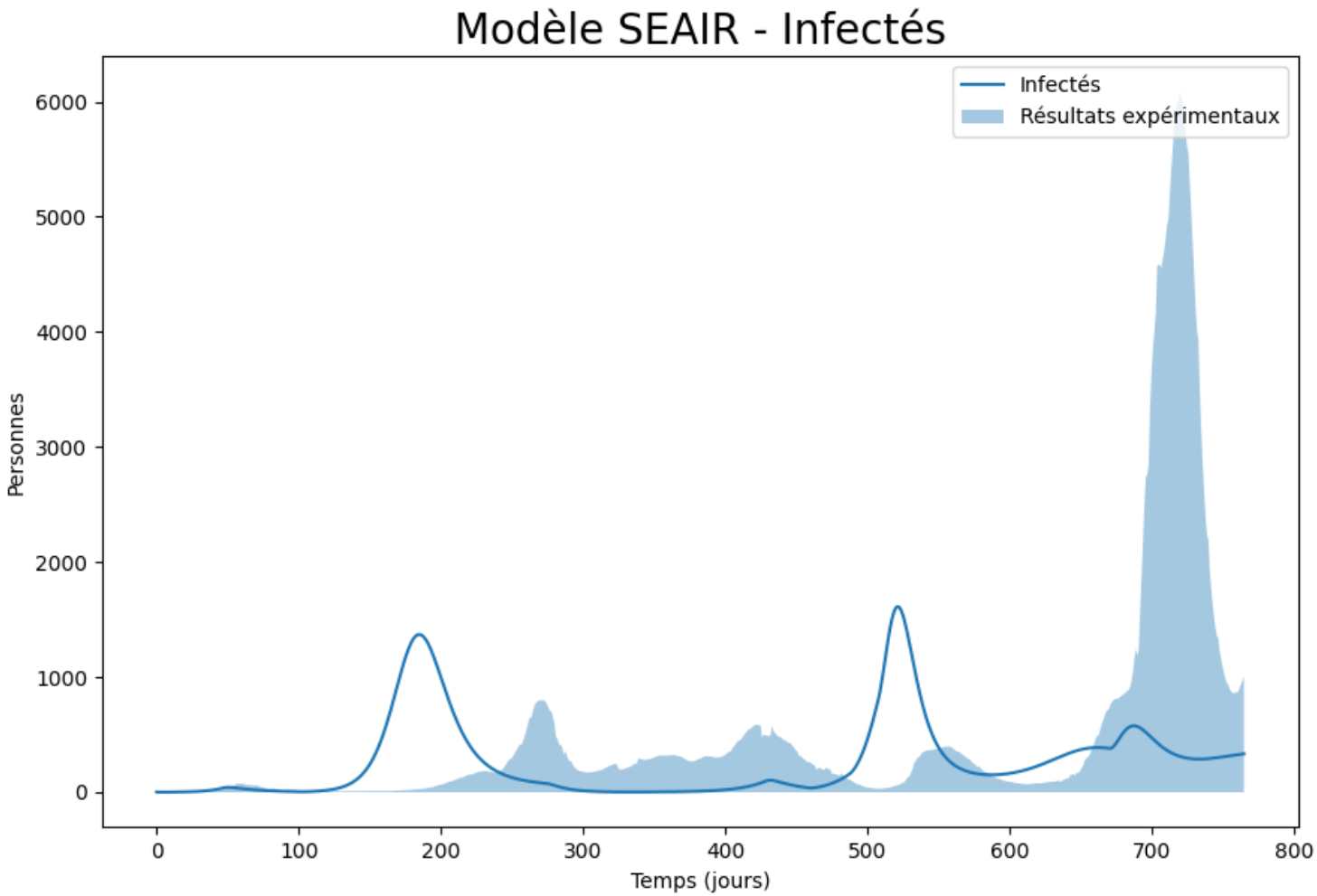
Détermination : λ



On déduit du graphe précèdent :

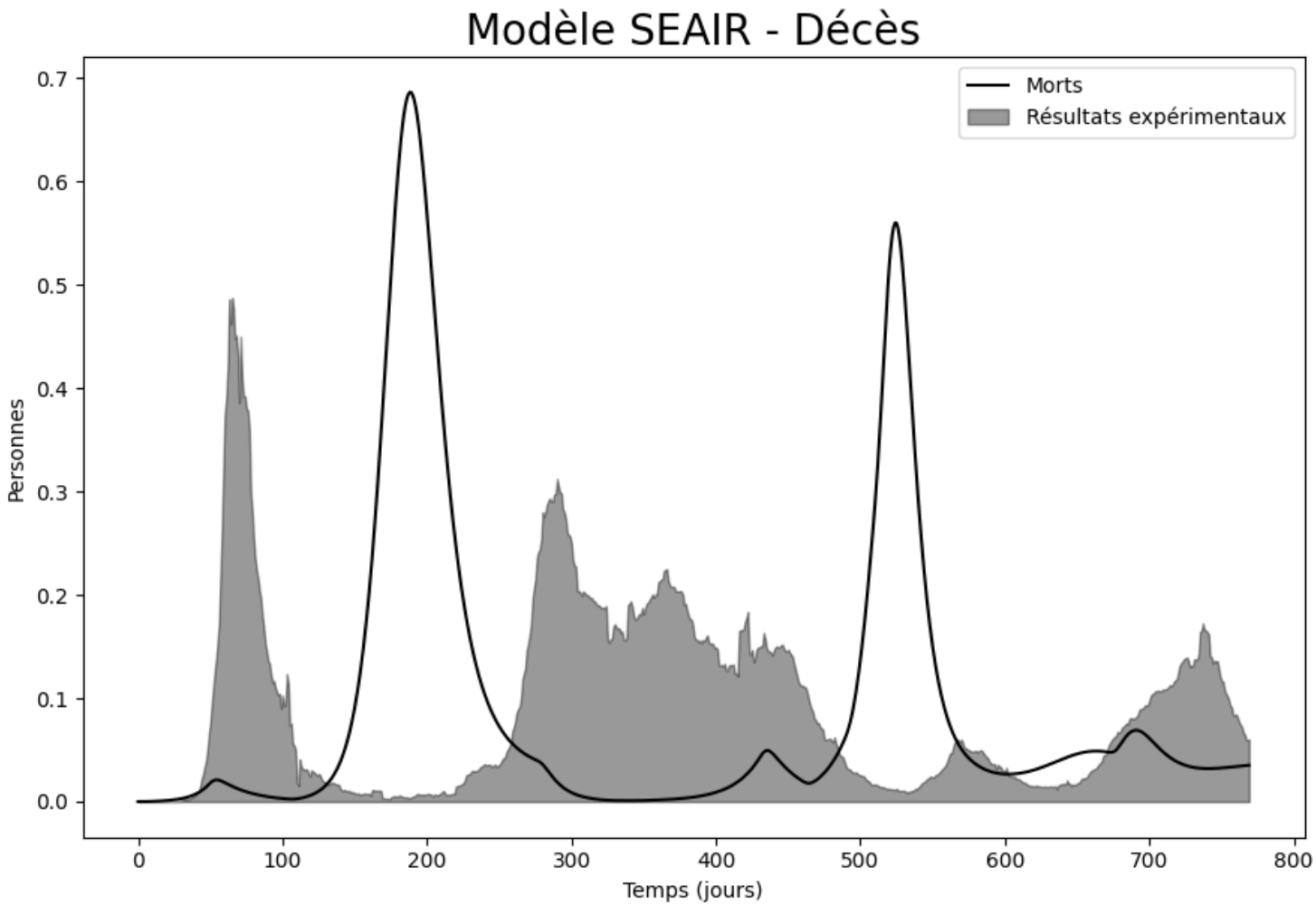
$\lambda = (\beta * (A_1 + I_1 + A_2) + \beta * b * I_2) * (1 - c)$

Courbe d'infection



source (résultats expérimentaux) : data.gouv.fr

Courbe de décès



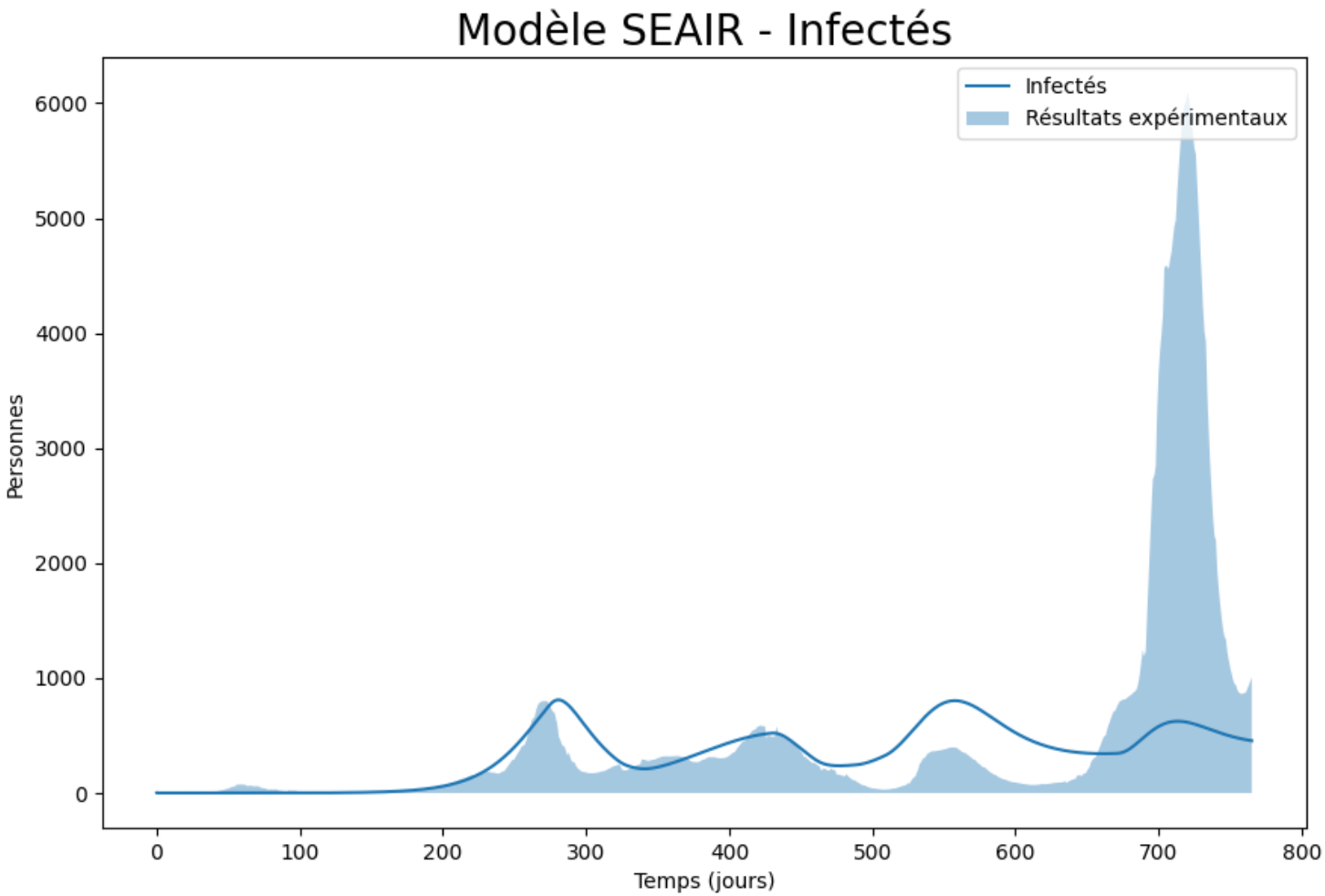
 source (résultats expérimentaux) : data.gouv.fr

Nouvelles constantes

Nouvelles constantes

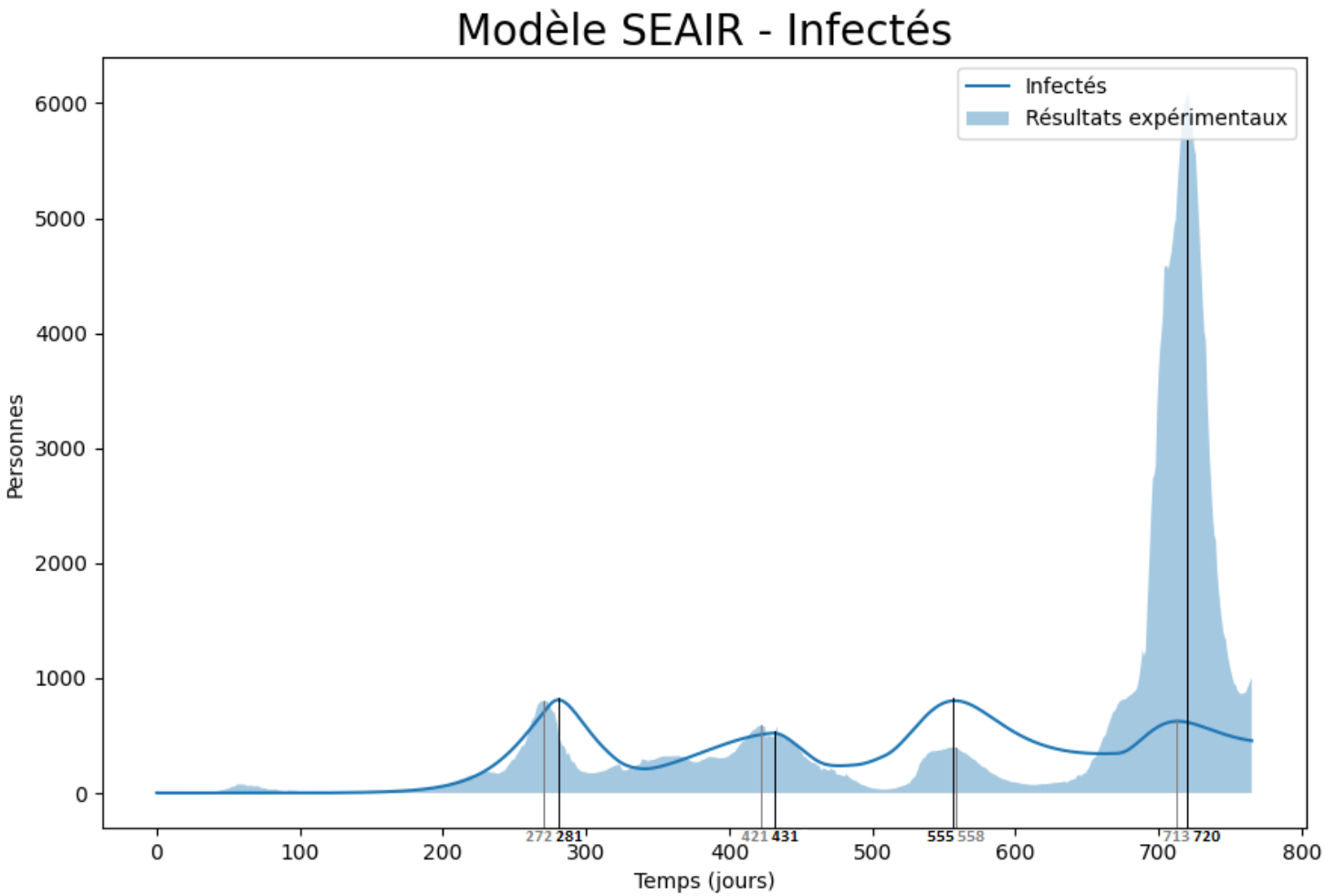
Variable	Ancienne valeur	Nouvelle valeur
ϵ	0.25	$1/14$
σ	0.5	0.2
γ_1	0.1	$1/15$
γ_2	0.081875	0.055

Courbe d'infection



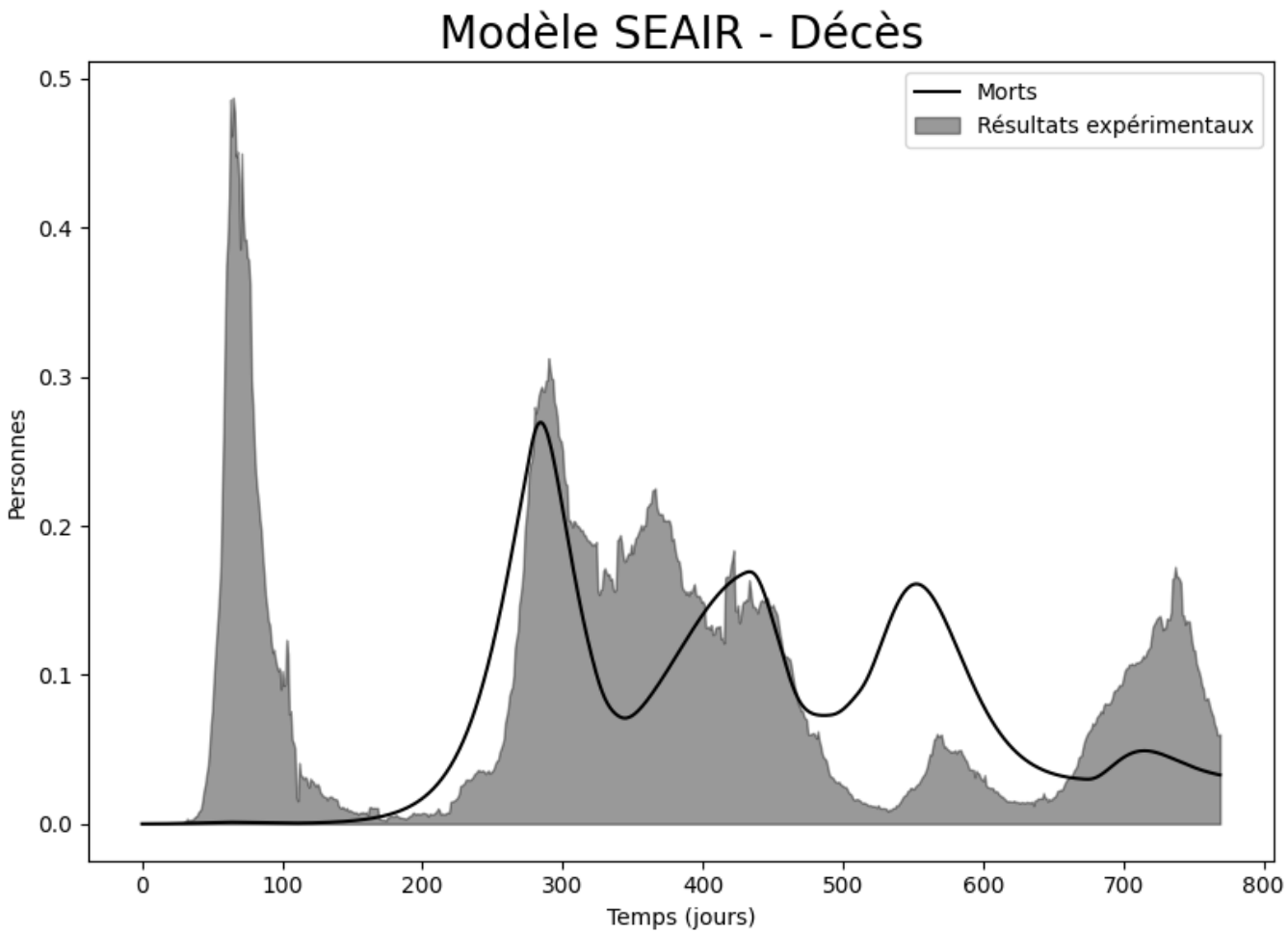
source (résultats expérimentaux) : data.gouv.fr

Courbe d'infection



source (résultats expérimentaux) : data.gouv.fr

Courbe de décès



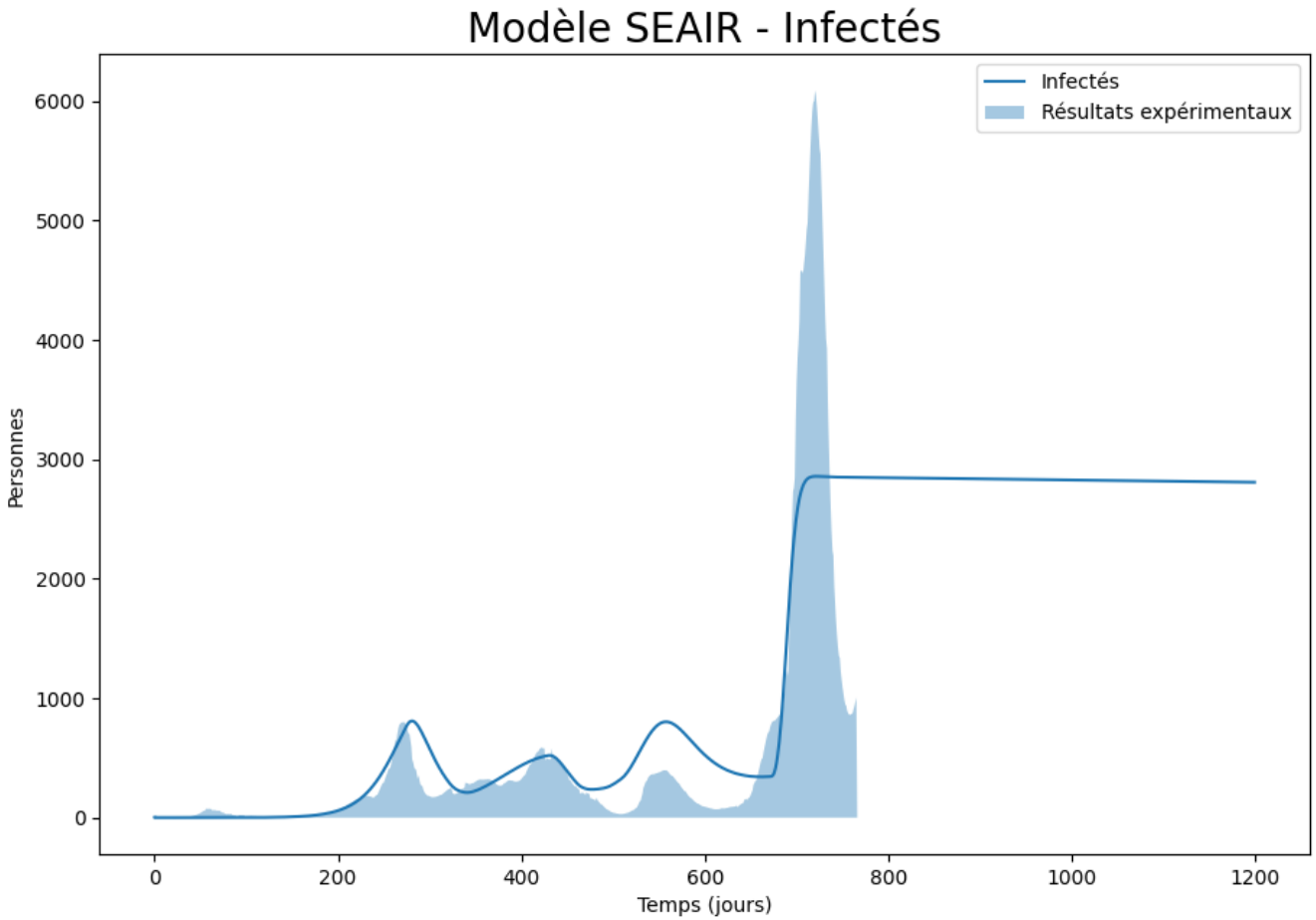
source (résultats expérimentaux) : data.gouv.fr

Changement pour modéliser la dernière vague

On fait varier r au cours du temps

- Avant la vague Omicron (décembre 2021) : $r = 180^{-1}$
- Après décembre 2021 : $r = 1$

Validité du modèle



source (résultats expérimentaux) : data.gouv.fr

Validité du modèle

- En modifiant les constantes : Meilleurs résultats
- Vagues bien retranscrites sauf première et dernière
- Valable uniquement pour certaines tailles de population

Merci de votre attention.

```
___author___ = "Joris ROUSERE"
___title___ = "TIPE"

### INFORMATIONS ###
#minimum jours résolution infectés : 766
#minimum jours résolution décès : 770

################################################

import matplotlib.pyplot as plt
import csv


S0base = 7000
coef = 60
coef2 = 2000


#IMPORTATION DES DONNEES EXPERIMENTALES

#Infectés
def realite():
    #On ouvre le fichier et on met son contenu dans data
    #debut 2/03/2020
    test_pos =[]
    with open('synthese-fra.csv', newline='') as f:
        reader = csv.reader(f)
        data = list(reader)
    for i in range(2,73):

        nouveau = int(data[i][1]) - int(data[i-1][1])
        test_pos.append(nouveau)

    with open('sp-pos-quot-fra-2022-03-14-19h00.csv', newline='') as f:
        reader = csv.reader(f)
        data = list(reader)
    for i in range(1,len(data)-1):
        a = data[i][0].split(';')
        if a[8] == '0':
            test_pos.append(int(a[4]))
```

```
    moyenne = []
    for i in range(len(test_pos)):
        if i <7:
            moyenne.append(test_pos[i]/coef)
        else :

            moyenne.append((test_pos[i]+test_pos[i-1]+test_pos[i-2]+test_pos[i-3]+test_pos[i-4]+test_pos[i-5]+test_pos[i-6])/7/coef)


    return moyenne

#Décès
def realitemorts():
    #On ouvre le fichier et on met son contenu dans data
    #debut 2/03/2020
    #morts hopitaux + ephad

    morts =[]
    with open('synthese-fra.csv', newline='') as f:
        #les cas sans données
        last = 0
        reader = csv.reader(f)
        data = list(reader)
    for i in range(2,744):
        a = last
        b = last
        if data[i][3] != "NaN":
            a=int(data[i][3])
            last = int(data[i][3])
        if data[i-1][3] != "NaN":
            b=int(data[i-1][3])
        nouveau = int(data[i][2]) - int(data[i-1][2]) + int(a) - int(b)
        morts.append(nouveau)

    moyenne = []
    t = []

    for i in range(len(morts)):
        t.append(i)
```

```
        if i <7:
            moyenne.append(morts[i]/coef2)
        else :

            moyenne.append((morts[i]+morts[i-1]+morts[i-2]+morts[i-3]+morts[i-4]+morts[i-5]+morts[i-6])/7/coef2)
    return moyenne
   # return moyenne

#Vaccins
def realitevaccin():
    #On ouvre le fichier et on met son contenu dans data
    #debut 27/12/2020
    #on renvoit le % de vaccinés
    #On connsidere que les Vaccines completement sans dernier rappel

    vax =[]
    with open('vacsi-fra-2022-04-04-19h00.csv', newline='') as f:
        #les cas sans données
        reader = csv.reader(f)
        data = list(reader)
    for i in range(1,443):
        a = data[i][0].split(";")
        vax.append(float(a[9]))


#Ceci permet d'avoir des valeurs nulles pour les instants où la vaccination n'a pas commencé
#et de dupliquer la dernière valeur connue pour ne pas avoir de problèmes de manque de données

    v = [0 for i in range(330)]
    v2 = [vax[-1] for i in range(330)]
    vax = v + vax + 2*v2



    return vax


#########################################################

#CALCUL DES CONSTANTES
```

```
#Calcul de R0(t)
def calculR0(t):
    #On compte à partir du 1er février 2020
    #en 2020: covid classique
    #janv 2020 - avril 2020 : variant alpha
    #mai 2020 -... : variant delta


    if t<335: return 2.8
    if t<486: return 4.5
    if t<669 : return 6.6
    else:return 15

#Calcul de V(t) : taux de vaccination
def calculV(t):

    return realitevaccin()[t]/100


#Calcul de C(t) : politique de contrôle
def calculC(t):
    #On compte à partir du 1er février 2020
    #1er confinement : c=0.84
    #couvre feu : c=0.35

    #confinements : 17 mars-11mai puis 30oct-15dec puis 3avr2021-2mai2021
    #Couvre feu: 16dec - 2avr2021 puis 3 mai - 20juin 2021


    if (t>=45 and t<100) or (t>=272 and t<=318) or (t>=427 and t<=456):
        return 0.84
    elif (t>=319 and t<=426) or (t>=457 and t<=505):
        return 0.35
    else:
        return 0
    return 0


#Calcul de P(t) : proportion cas benin/sévères
```

```
def calculP(t):
    tauxvax = calculV(t)
    p0 = 0.968
    p=1-(1-p0) * (1-tauxvax)
    return p



#Fonction qui renvoie l'ensemble des constantes nécessaires à la résolution
def constante(t, A1, I1, A2, I2, gamma1, sigma, alpha, gamma2,S0 = S0base, b =0.2):
    R0 = calculR0(t)
    c = calculC(t)
    p= calculP(t)
    B = (R0/S0)*gamma1*sigma*((alpha+gamma2)/((gamma1+p*sigma)*(alpha + gamma2) + b*gamma1*sigma*(1-p)))

    lambd = (B*(A1+I1+A2) + B*b*I2)*(1-c)

    return B, lambd, p


#Fonction qui calcule les valeurs au jour suivant
def day(t, epsilon, sigma, gamma1, gamma2, alpha,
        S, E1, A1, I1, R1, M, E2, A2, I2, R2, b, retour =1/180, S0=S0base ):


    cstes = constante(t, A1, I1, A2, I2, gamma1, sigma, alpha, gamma2)
    l=cstes[1]
    p=cstes[2]

    #Si t>= 670 on change le retour

    if t>=670:
        retour = 1


    R1 += I1*gamma1
    I1 -= I1*gamma1
    I1 += sigma*A1
    A1 -= A1*sigma
```

```
    A1 += E1*epsilon
    E1 -= E1*epsilon
    E1 += S*l*p
    M += I2*alpha
    R2 += I2*gamma2
    I2 -= I2*alpha
    I2 -= I2*gamma2
    I2 += sigma*A2
    A2 -= A2*sigma
    A2 += E2*epsilon
    E2 -= E2*epsilon
    E2 += S*l*(1-p)
    S -= S*l
    S+=R1*retour
    S+=R2*retour
    R1-=R1*retour
    R2-=R2*retour

    return(S,E1,A1,I1,R1,M,E2,A2,I2,R2)



#Fonction de résoltuion
def resolution(time):
    #On récupère les valeurs expérimentales

    real = [0 for i in range(29)]
    real = real + realite()

    real_morts = [0 for i in range(29)]
    real_morts = real_morts + realitemorts()

    #constantes
    gam1 = 1/15
    a = 0.18125
    gam2 = gam1*(1-a)
    al = gam1*a
    e = 1/14
    s = 1/5
```

```
b1 = 0.2

#Listes comprenant les valeurs des compartiments
lS = [S0base]
lE1 = [0]
lA1 = [0]
lI1 = [1]
lR1 = [0]
lM = [0]
lE2 = [0]
lA2 = [0]
lI2 = [0]
lR2 = [0]

#la liste du temps pour tracer
t=[0]

#Chaque jour on résout
for i in range(time):
    a = day(t[-1], e, s, gam1, gam2, al, lS[-1], lE1[-1], lA1[-1], lI1[-1], lR1[-1], lM[-1], lE2[-1], lA2[-1], lI2[-1], lR2[-1], b1)
    lS.append(a[0])
    lE1.append(a[1])
    lA1.append(a[2])
    lI1.append(a[3])
    lR1.append(a[4])
    lM.append(a[5])
    lE2.append(a[6])
    lA2.append(a[7])
    lI2.append(a[8])
    lR2.append(a[9])
    t.append(i)

#On combine les valeurs (R = R1 + R2, I = I1+I2...)
lR=[]
lI = []
lHosp = []
lNonMalades =[]
for i in range(len(lR2)):
    lR.append((lR1[i]+lR2[i]))
    lI.append((lI1[i]+lI2[i]))
```

```
        lHosp.append(((lA2)[i]+lI2[i]))
        lNonMalades.append((lR1[i]+lR2[i]+lS[i]))


    #On calcul les décès : Pour connaitre leur nombre chaque jour et effectuer une moyenne sur les 7 derniers jours
    deces = [0]
    for i in range(1,len(lM)):
        if i <7:
            deces.append((lM[i] - lM[i-1]))
        else :

            deces.append(((lM[i] - lM[i-6])/7))



    #On calcule les valeurs de C et R0 pour les tracer (pour vérifier leurs valeurs)
    lC=[]
    lR0=[]
    lvax=[]
    for i in t:
        lC.append(calculC(i)*S0base)
        lR0.append(calculR0(i)*100)
        lvax.append(calculP(i))



    #On complete les resultats expérimentaux
    while len(real)!= len(t):
        real.append(0)
    while len(lvax)!= len(t):
        lvax.append(0)
    while len(real_morts)< len(t):
        real_morts.append(0)


### TRACAGE DES COURBES ###
#Certaines d'entre elles servent à des fins de compréhension du modèle.
```

```
    #plt.plot(t, deces, label = 'Morts', color = "black")
    #plt.plot(t, lR, label = 'Rescapés')
    #plt.plot(t, lNonMalades, label = 'Non malades')
    #plt.plot(t, lS, label = 'Saints')
    #plt.plot(t, lHosp, label = 'Hospitalisés')
    #plt.plot(t, lI, label = 'Infectés')
    #plt.plot(t, lC, '--',color = 'grey', label='c(politique de contrôle)')
    #plt.plot(t, lR0, '--',color = 'black', label='R0')
    #plt.fill_between(t, real, label = 'Résultats expérimentaux', alpha = 0.4)
    #plt.fill_between(t, real_morts, label = 'Résultats expérimentaux', color = "black", alpha = 0.4)

    #Légende
    plt.xlabel('Temps (jours)')
    plt.ylabel('Personnes')
    plt.legend()

    #Titre de la courbe
    plt.title('Modèle SEAIR', fontsize = '20')
    plt.show()


#Fonction de test de constantes
def testcstes():
    v=[]
    tt=[]
    c=[]
    for t in range(770):
        v.append(calculV(t))
        tt.append(t)
        c.append(calculC(t))

    plt.plot(tt,v, label = "Taux de vaccination v")
    #Légende
    plt.xlabel('Temps (jours)')
    plt.ylabel('')
    plt.legend()

    plt.show()
```

```
#On commence l'étude au 1er Février 2020


#JORIS ROUSERE
#Numero inscription : 15890
```