LA PROPAGATION DU SIDA EN FRANCE

AFERIAT SOPHIA
BENBOUAZZA ALEX
GOVART ALICE
RAMAHERISON MANON

Conception du projet

- On prend 2 individus dans une population de N personnes. On vérifie s'il y a un rapport ou non entre eux en fonction des paramètres étudiés, puis on observe ainsi s'il y a contamination ou pas.

Liste des Agents			
Nom	Rôle	actions	Attributs caractéristiques
Homme hétérosexuel	Étude de son infection après une ou plusieurs rencontres	- rencontre avec des gens de sa région - rapport sexuel ou non - rapport protégé ou non - infecter/être infecté ou non	(1, False) **Type de rapport possible** : RI Probabilité de transmettre le sida par le RI : 0,0006
Femme hétérosexuel			(0, False) **Type de rapport possible** : RVR RAR Probabilité de transmettre le sida par le RVR : 0,0015 Probabilité de transmettre le sida par le RAR : 0,015
Homme homosexuel			(1, True) **Type de rapport possible** : RAR RI Probabilité de transmettre le sida par le RAR : 0,015 Probabilité de transmettre le sida par le RI : 0,0006
Femme homosexuel			(0, True) Aucun rapport étudié

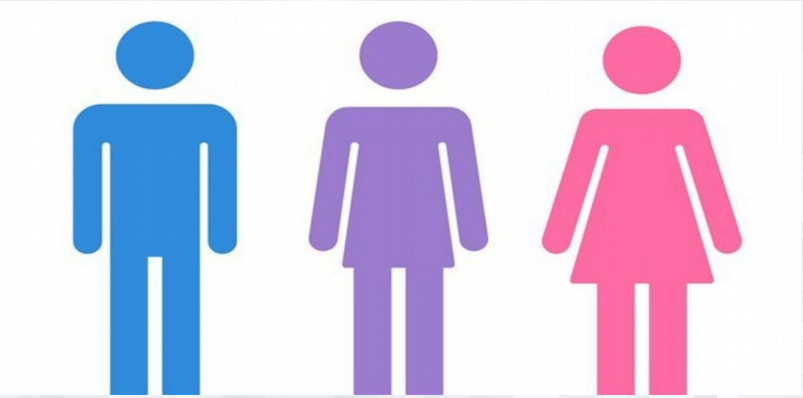
Liste des Paramètres				
Nom	Type	Intervalle	Valeur initial	fixe
Pourcentage de gay en France	float	0; 1	0,07	oui
Pourcentage de lesbienne en France	float	0; 1	0,01	oui
Probabilité que les deux individus aient un rapport sexuel après la rencontre	Float	0; 1	0,2	non
Probabilité que le rapport soit protégé	Float	0; 1	0,7	non
Probabilité que la femme pratique le sexe annal	Float	0; 1	0,1	oui
Probabilité de transmission du sida selon le rapport :				
RI	Float	0; 1	0	oui
RVR	Float	0; 1	0	oui
RAR	Float	0; 1	0,02	oui

I- Création d'une matrice

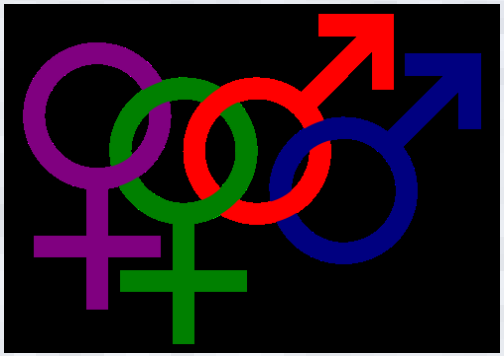
La matrice représente une population où chaque terme est un individu caractérisé par différents paramètres :

- Genre : homme (1) & femme (0)
- Orientation sexuelle : hétérosexuel (0) & homosexuel (1)
- Région
- Contamination : 'infecté' & 'non infecté

- a) GENRE → 'np.random.randint(2)'



- b) ORIENTATION → 'np.random.random()<C' avec C = pourcentage de gay



- c) REGION → création d'une liste de tuples tel que :

```
In [*]: #on va rajouter un parametre dans la matrice qui est la provenance des regions
        #on fait une liste de tuple avec les regions et les probabilités de venir de là
        L = [ ('Haut de France',0.09), ('Grand Est',0.08), ('Normandie',0.05), ('Bretagne', 0.05),
             ('Pays de la Loire', 0.06), ('Centre Val de Loire',0.04), ('IDF',0.18),('Bourgogne Franche Comté', 0.04),
             ('Nouvelle Aquitaine', 0.09), ('Auvergne Rhone Alpes',0.12),('Occitanie',0.09), ('Provence Alpes Cotes d Azur',0.07),
             ('Corse', 0.01), ('Guadeloupe', 0.01), ('Martinique', 0.01), ('Guyane', 0.01), ('Reunion', 0.01), ('Mayotte',0.01) ]
```

- La fonction provenance permet d'attribuer une région à un individu dans la matrice en fonction d'une probabilité

```
def provenance (L) :
    e=0
    i,j=L[e]
    s=j
    p = np.random.random()

    while  p> s:
        e=e+1
        s = s +L[e][1]
    return L[e][0]
```

- d) CONTAMINATION → vérification si à l'état initial, l'individu est infecté ou non

  on note 'np.random.random' un nombre choisi aléatoirement, et S la probilité qu'une personne soit infectée du SIDA en France

```
#On va rajouter un autre parametre afin de savoir si la personne est infectee ou non, parametre a rajouter dans la matrice


def infection (S) :
    if np.random.random() < S:
        return 'infecté'
    else:
        return 'non infecté'
```

II- Intéractions entre les individus

a) Compatibilité entre deux individus

- → En se basant sur le genre et l'orientation sexuelle de deux individus pris aléatoirement, on regarde s'ils sont compatibles

```
def compatibilite(a,b):

    #x défini le genre et y defini lorientation sexuelle
    #Pour le genre, femme (0) et homme (1)
    #Pour lorientation sexuelle, homosexuel (1) et hétéro(0)

    x1,y1,c,d = a
    x2,y2,j,k = b
    if y1 == y2 and y1 == 1 :
        if x1 == x2 :
            return 'compatible'
        else :
            return 'non compatible'
    elif y1==y2 and y1==0:
        if x1==x2:
            return 'non compatible'
        else :
            return 'compatible'
    elif y1 != y2 :
        return 'non compatible'

print(compatibilite(newstate[1],newstate[1]))
```

b) Rapport sexuel

- → A l'aide la fonction compatibilité, on vérifie si les deux individus peuvent avoir une relation sexuelle en fonction de la région

```
def rapport_sexuel(a,b,Prs,newstate):


    (ya,zb,c,d) = a
    (e,f,g,h) = b


    if compatibilite(a,b)== 'compatible' and (_,_,c,_)==(_,_,g,_):

        y=np.random.random()
        #print(y)
        if y <= Prs :
            return 'rapport'
    return 'non rapport'
```

- → On vérifie ensuite grâce à une probabilité si le rapport est protégé ou non

```
def protection(a,b,Prs,Prp,newstate):

    if rapport_sexuel(a,b,Prs,newstate)== 'rapport':
        z=np.random.random()

        if z <= Prp:
            return 'rapport protégé'
        else:
            return 'rapport non protégé'
    return 'pas de rapport'
```

- c) Transmission du sida
- Si le rapport est non protégé, et que les conditions de rencontre sont respectées, 2 individus prit au hasard ayant un rapport se transmettent le virus

```
def sida(a,b,Prs,RAR,RI,RVR,newstate):

    print(a,b)
    protege = protection(a,b,Prs,Prp,newstate)

    (ya,zb,_,d) = a
    (e,f,_,h) = b

    if protege == 'rapport non protégé':
        print("non protégé")


        if (ya,zb,_,d) ==(1,False,_,'infecté') and (e,f,_,h)==(0,False,_,'non infecté'):
            s = random.random()
            print('r')
            if s <= RVR:
                newstate[ligne_d]= np.array([0, False,_, 'infecté'], dtype=object)
                return newstate                                  #'femme infectée'
        elif (ya,zb,_,d)==(1,False,_,'non infecté') and (e,f,_,h)==(0,False,_,'infecté'):
            s1 = random.random()
            print('r')
            print(s1)
            if s1 <= RI:
                newstate[ligne_c]= np.array([1, False,_, 'infecté'], dtype=object)
                return newstate
            print(newstate[ligne_a])                                    #'homme infecté'
```

```
elif (ya,zb,_,d) ==(0,False,_,'infecté') and (e,f,_,h)==(1,False,_,'non infecté'):
    s2 = random.random()
    print(newstate[5])
    if s2 <= RI:
        newstate[ligne_d]= np.array([1, False,_, 'infecté'], dtype=object)
        return newstate                                    #'homme infecté'

elif (ya,zb,_,d) ==(0,False,_,'non infecté') and (e,f,_,h)==(1,False,_,'infecté'):
    s3 = random.random()
    print('r')
    if s3 <= RVR:
        newstate[ligne_c]= np.array([0, False,_, 'infecté'], dtype=object)
        return newstate                                      #'femme infectée'
elif (ya,zb,_,d) ==(1,True,_,'infecté') and (e,f,_,h)==(1,True,_,'non infecté'):
    s4 = random.random()
    print('r')
    if s4 <= 0.5:
        if s4<=RI:
            newstate[ligne_d]= np.array([1, True,_, 'infecté'], dtype=object)
            return newstate
    elif s4>0.5:
        if s4>RAR:
            newstate[ligne_d]= np.array([1, True,_, 'infecté'], dtype=object)
            return newstate
```

III- Dynamique

- On crée maintenant un programme simulateur de la dynamique de propagation du sida au sein de la population représentée par la matrice

```
def simulation(nb,newstate,Prs):
    i = 0

    while i<= nb :
        ligne_c=np.random.randint(N)
        a=(newstate[ligne_c])
        ligne_d=np.random.randint(N)
        b=(newstate[ligne_d])
        print(a,b)
        protege = protection(a,b,Prs,Prp,newstate)

        (ya,zb,_,d) = a
        (e,f,_,h) = b

        if protege == 'rapport non protégé':
            print("non protégé")


            if (ya,zb,_,d) ==(1,False,_,'infecté') and (e,f,_,h)==(0,False,_,'non infecté'):
                s = random.random()
                print('r')
                if s <= RVR:
                    newstate[ligne_d]= np.array([0, False,_, 'infecté'], dtype=object)
                    return newstate                              #'femme infectée'
            elif (ya,zb,_,d)==(1,False,_,'non infecté') and (e,f,_,h)==(0,False,_,'infecté'):
                s1 = random.random()
                print('r')
                print(s1)
                if s1 <= RI:
```

```
            return newstate                                    #'femme infectée'
        elif (ya,zb,_,d) ==(1,True,_,'infecté') and (e,f,_,h)==(1,True,_,'non infecté'):
            s4 = random.random()
            print('r')
            if s4 <= 0.5:
                if s4<=RI:
                    newstate[ligne_d]= np.array([1, True,_, 'infecté'], dtype=object)
                    return newstate
            elif s4>0.5:
                if s4>RAR:
                    newstate[ligne_d]= np.array([1, True,_, 'infecté'], dtype=object)
                    return newstate
        elif (ya,zb,_,d) ==(1,True,_,'non infecté') and (e,f,_,h)==(1,True,_,'infecté'):
            s5 = random.random()
            print('r')
            if s5 <= 0.5:
                if s5<=RI:
                    newstate[ligne_c]= np.array([1, True,_, 'infecté'], dtype=object)
                    return newstate
            elif s5>0.5:
                if s5>RAR:
                    newstate[ligne_c]= np.array([1, True,_, 'infecté'], dtype=object)
                    return newstate



    i+=1
return newstate
```

```
def nb_infecte(prc_gay,L,newstate,Prs):
    newstate2 = simulation(6000,population3(prc_gay,0.01,L,0.5),Prs)
    x=newstate2[0]
    print(x)
    (_,_,_,d)=x
    N=20
    compteur=0
    for i in range(0,N):
        (_,_,_,d)=newstate2[i]
        if  (_,_,_,d)==(_,_,_, 'infecté'):
            compteur=compteur+1

    return compteur

print(nb_infecte(prc_gay,L,newstate,,Prs))
```

On a réalisé différents graphique caractérisant le nombre d'infectés en fonction des paramètres

```
from pylab import *
%matplotlib inline
import random


gay = np.linspace(0,1,11)
print(gay)

results = []


for prc_gay in gay:
    results.append(np.mean(nb_infecte(prc_gay,L,newstate,0.9)))

rapport = np.linspace(0,1,11)
print(rapport)

results1 = []

for Prs in rapport:
    results1.append(np.mean(nb_infecte(0.07,L,newstate,Prs)))

plot(rapport, results1, color="brown")
plot(gay, results, color="green")



show()
```