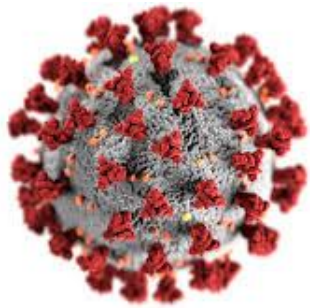


Influence d'une pandémie mondiale impliquant des stratégies de confinement telle que celle provoquée par la covid-19 sur un plan sanitaire et sur un plan alimentaire

Comment établir un modèle cohérent sur l'expansion de la covid-19 au niveau sanitaire et alimentaire ? Comment prévoir des mesures gouvernementales et évaluer les stocks nécessaires pour permettre aux pays de survivre ?



Epidémiologie : Modèle SIRD

En temps continu :

$$\frac{dS}{dt} = -\beta SI(t)$$

$$\frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t) - \delta I(t) \quad I(t+1) - I(t) = \beta S(t)I(t) - \gamma I(t) - \delta I(t)$$

$$\frac{dR}{dt} = \gamma I(t)$$

$$\frac{dD}{dt} = \delta I(t)$$

En temps discret :

$$S(t+1) - S(t) = -\beta SI(t)$$

$$R(t+1) - R(t) = \gamma I(t)$$

$$D(t+1) - D(t) = \delta I(t)$$

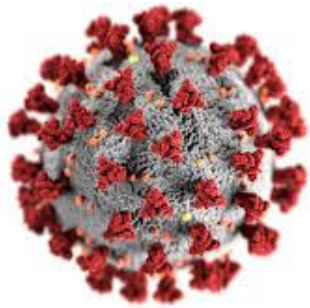
Paramètres:

- β : taux de contact : modélise les interactions sociales
- γ : taux moyen de guérison : 0.3 (dépend du virus uniquement)
- δ : taux de létalité (\neq taux de mortalité) : 0.048

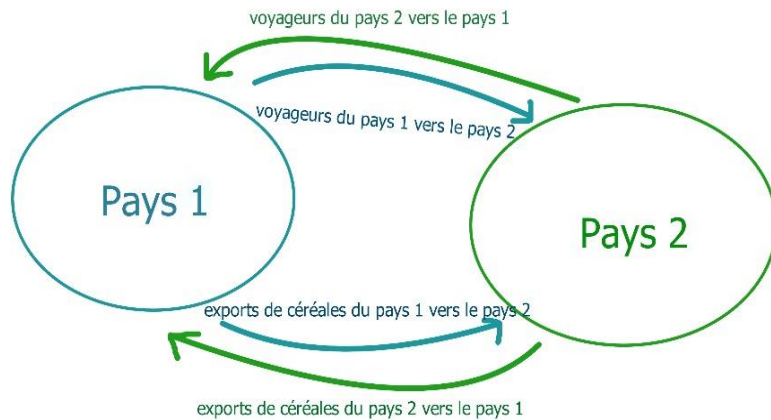
Estimation du paramètre β :

- Sans confinement : $\beta=0.39$
- Avec confinement: $\beta=0.22$

SIRD = Sains, Infectés, Recovered (guéris), Dead (morts)



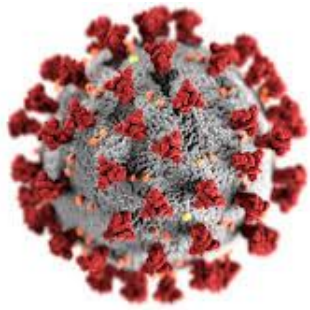
Echanges internationaux



Remarque : la destination et le nombre de voyageurs de chaque pays sont aléatoires (mais ce nombre ne peut excéder 0.002% de la population pour chaque pays)

Pays	Contaminés au 1 ^{er} mars 2020	Morts au 1 ^{er} mars 2020
France	116	2
Espagne	1486	0
Allemagne	114	0
UK	35	0
Italie	1578	41
Pologne	0	0
Belgique	1	0
Grèce	7	0
Republique Tchèque	3	0
Portugal	0	0
Hongrie	0	0
Suède	14	0
Autriche	14	0
Suisse	23	0
Danemark	4	0
Finlande	5	0
Slovaquie	0	0
Norvège	19	0
Irlande	1	0
Pays-bas	10	0

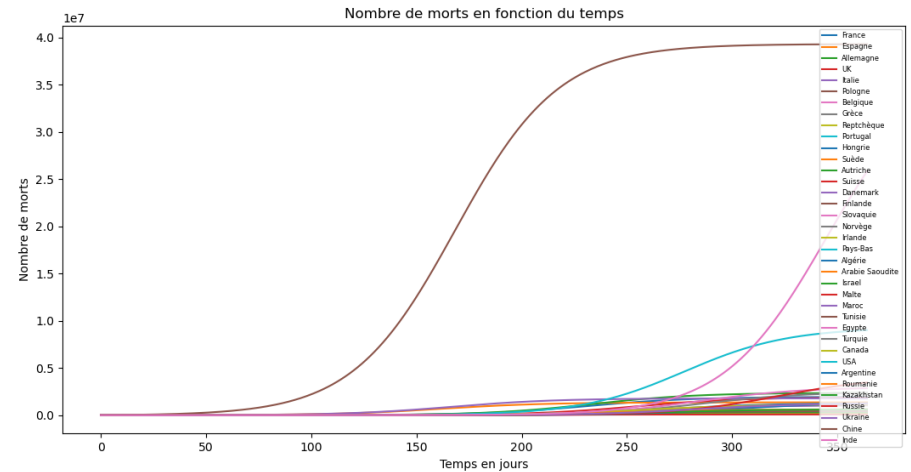
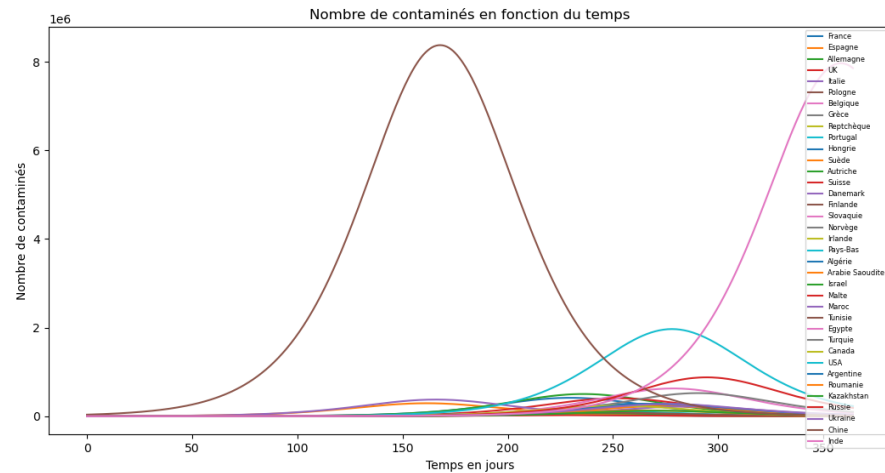
Algerie	3	0
Arabie saoudite	0	0
Israel	9	0
Malte	0	0
Maroc	0	0
Tunisie	0	0
Egypte	1	0
Turquie	0	0
Canada	20	0
USA	65	1
Argentine	0	0
Roumanie	2	0
Kazakhstan	0	0
Russie	0	0
Ukraine	0	0
Chine	32616	2912
Inde	0	0



Sans confinement : au niveau mondial

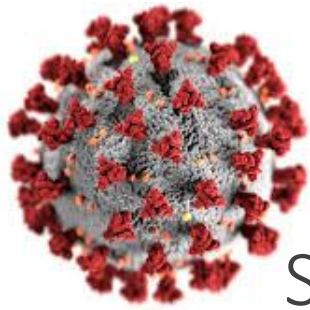
NOMBRE DE CONTAMINÉS

On observe des courbes en cloche pour tous les pays. La première phase de montée est exponentielle. On distingue seulement les courbes de la Chine et l'Inde dû aux tailles de populations.



NOMBRE DE MORTS

On a à nouveau une montée exponentielle puis une stagnation pour chaque pays.

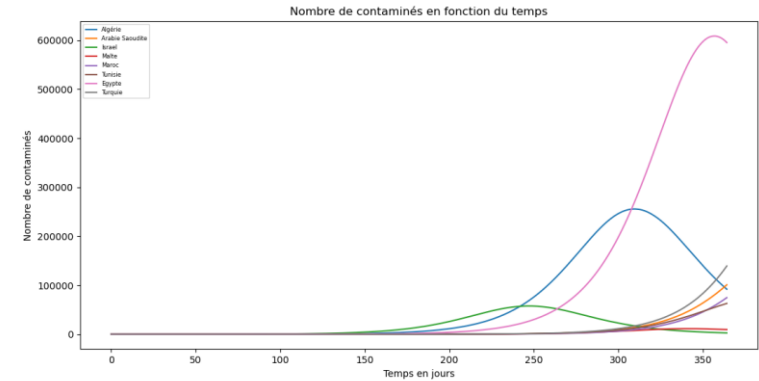
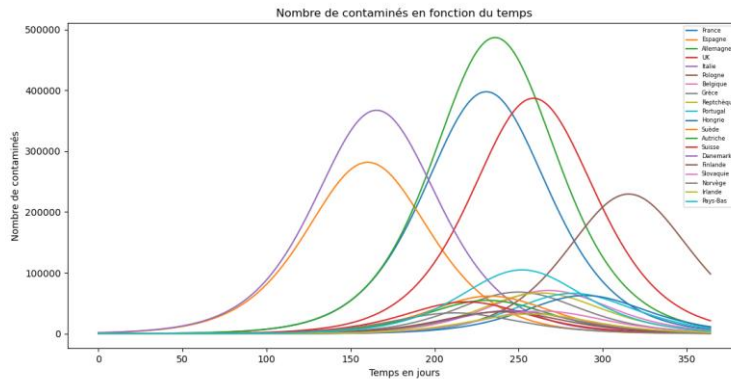


Sans confinement : dynamiques territoriales

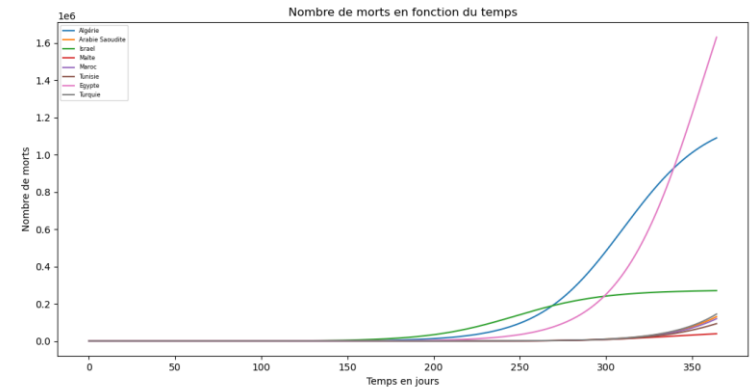
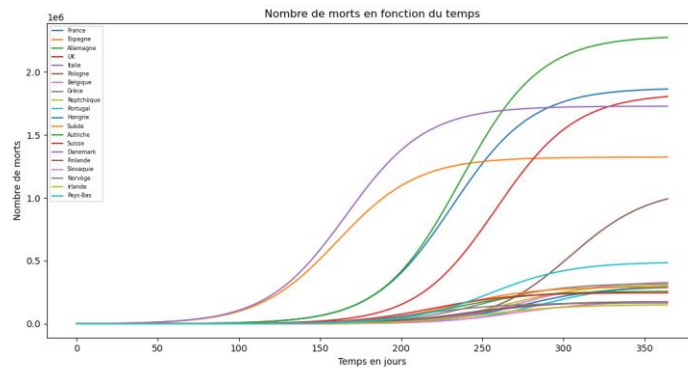
Europe

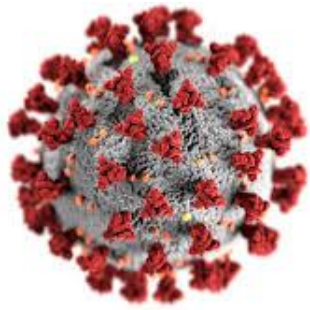
AFNMO

CONTAMINÉS



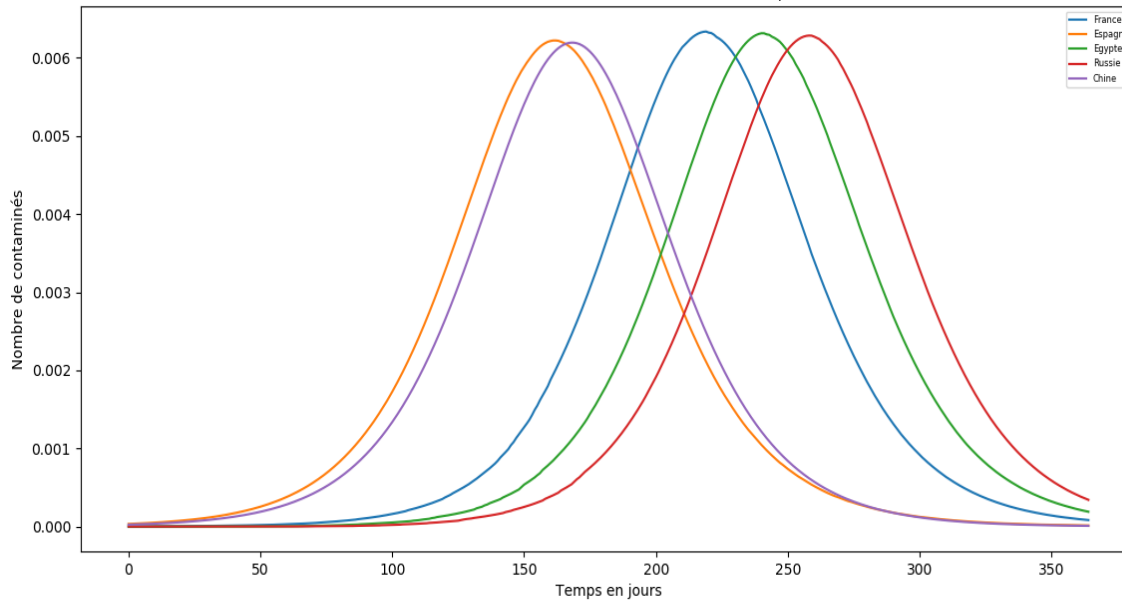
MORTS





Sans confinement – taux de contaminés

Nombre de contaminés en fonction du temps

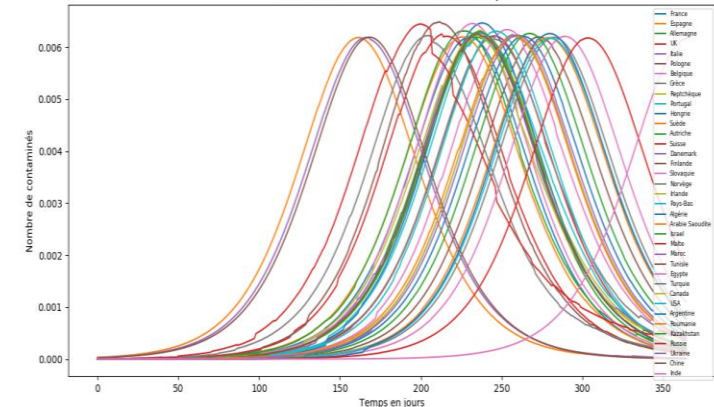


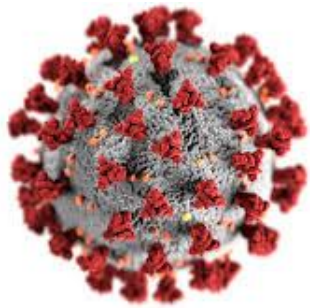
Remarque : Avec la situation initiale, on pouvait prévoir l'avance épidémiologique de l'Italie, Espagne ou encore la Chine.

TAUX DE CONTAMINES

Si l'on considère un taux de contaminés (sur quelques pays à gauche pour plus de lisibilité), on se rend compte que les pays suivent la même tendance. Le nombre de contaminés est proportionnel à la population.

Nombre de contaminés en fonction du temps





Cohérence du modèle

Immunité de groupe:

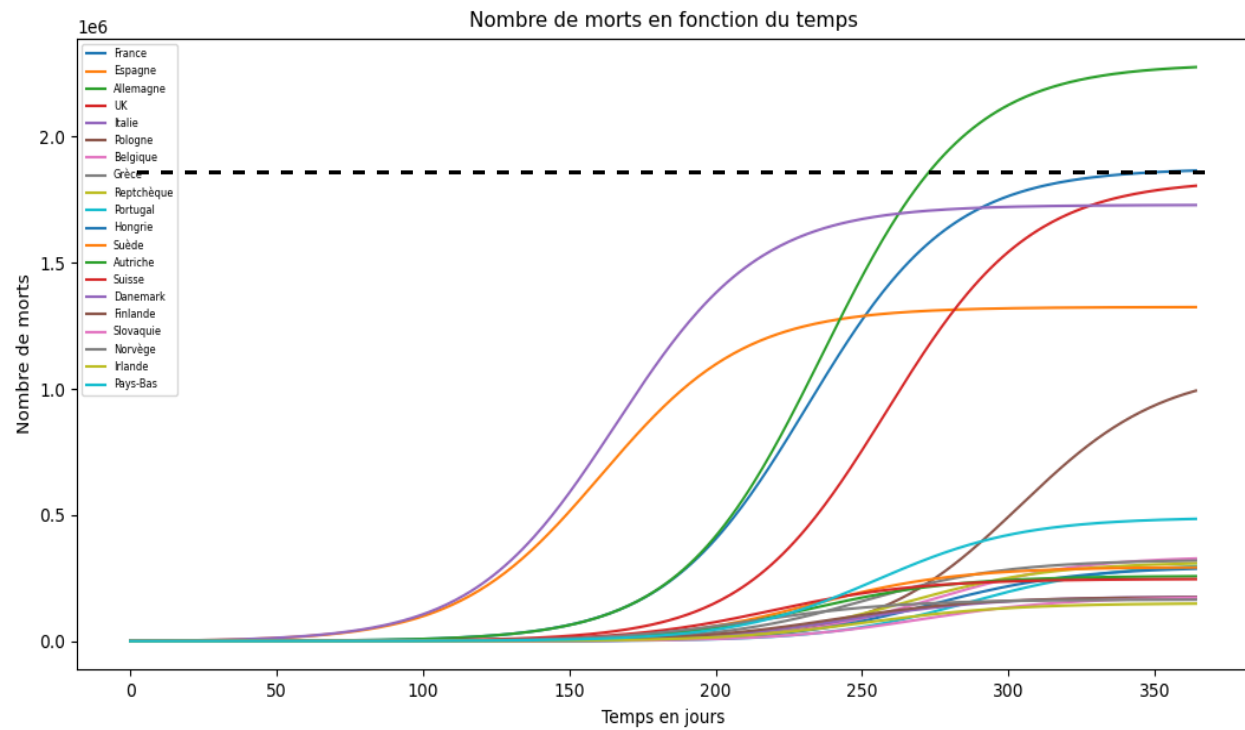
R_0 = nombre moyen d'individus immunologiquement naïfs qu'un sujet va infecter après contact

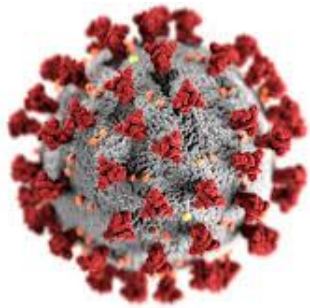
Immunité collective = $1 - 1/R_0$

Pour le covid-19 : R_0 estimé à 3 donc $p > 1 - 1/3 = 0,67$

Cohérence:

Pour la France, cela représente environ 40 millions de contaminés soit 2 millions de morts environ. C'est ce que l'on observe graphiquement.





Influence d'un confinement - Règles

But

Réduction des interactions sociales

Comment ?

Lorsqu'un pays dépasse un certain taux de contaminés, il se confine. Aucun habitant ne peut aller dans un autre pays, aucun habitant d'un autre pays ne peut venir dans le pays confiné.

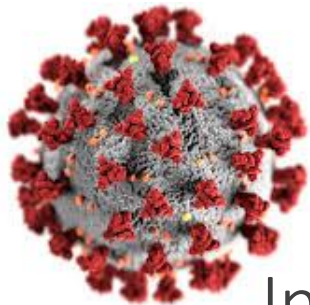
De plus, les interactions à l'intérieur même du pays sont fortement diminuées. Tant que le pays n'est pas confiné, $\beta=0.39$. Quand il se confine, $\beta=0.22$.

Pour se déconfiner, il doit passer en dessous du taux de déconfinement.

Taux de contaminés limites

Taux seuil de confinement : 0.003

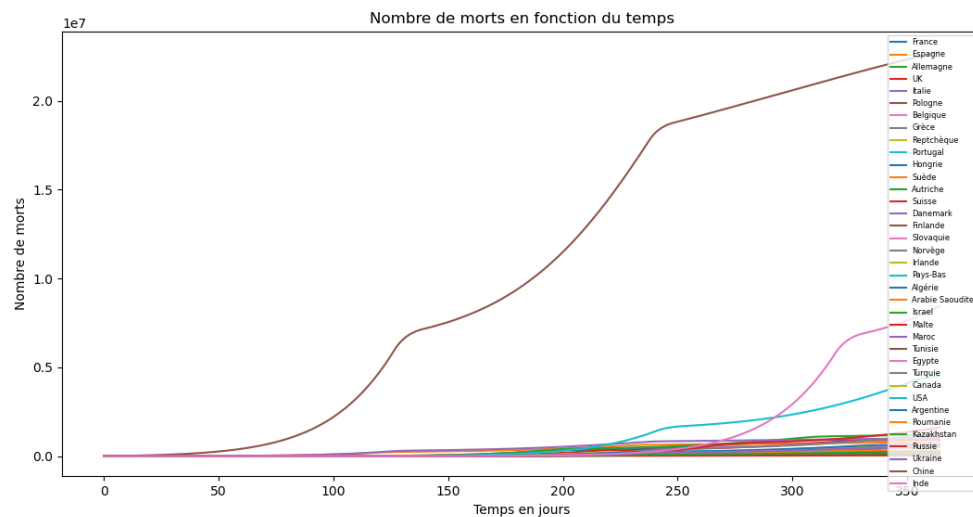
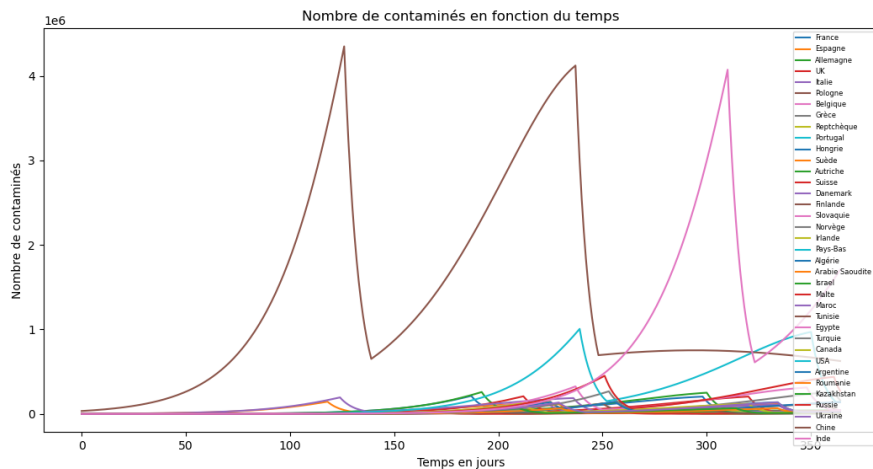
Taux seuil de déconfinement : 0.0006



Influence d'un confinement – Résultats monde

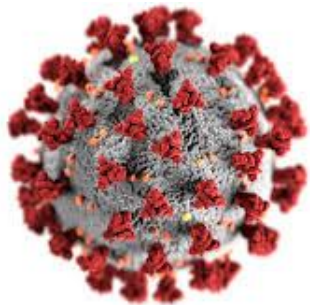
NOMBRE DE CONTAMINÉS

Dans un premier temps, le nombre de contaminés augmente, puis lors du confinement diminue d'où les pics. On observe non pas un confinement, mais une succession de confinements qui s'enchaînent.



NOMBRE DE MORTS

Le nombre de morts ne peut diminuer comme celui de contaminés car ce sont ici les morts cumulés. Néanmoins, lors des confinements, le nombre de morts augmente beaucoup moins vite.

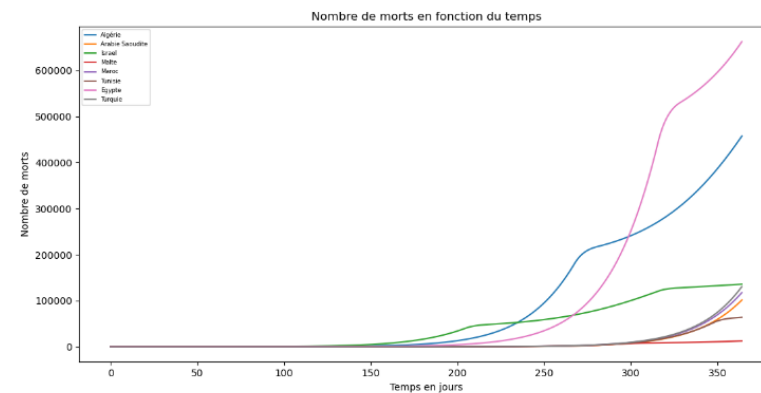
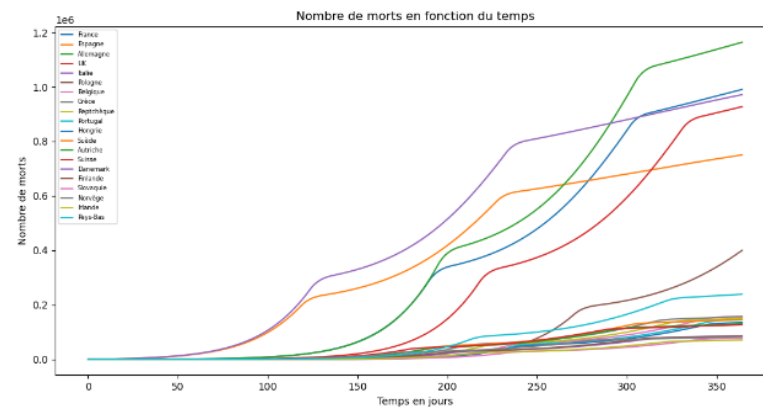
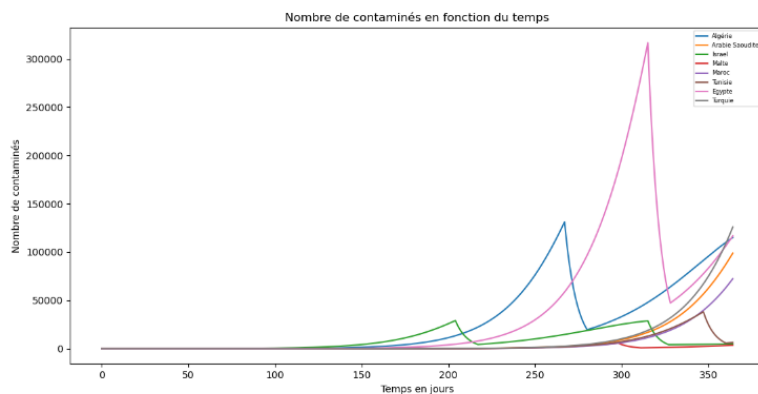
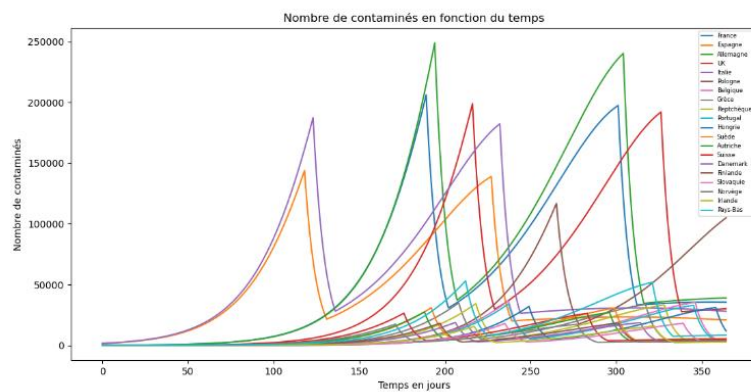


Confinement : résultats Europe et AFNMO

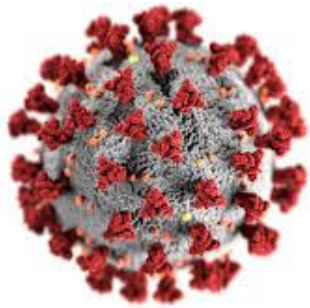
Europe

AFNMO

C
O
N
T
A
M
I
N
E
S



M
O
R
T
S



Biens alimentaires : Modèle

Biens alimentaires considérés:

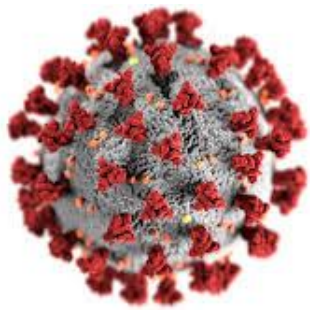
- Uniquement le blé

On introduit une production et une consommation par habitant (en dollars USD):

- Production par habitant journalière harmonisée : $\text{production annuelle} / \text{nombre d'habitants} / 365$
- Consommation par habitant journalière : estimation basée sur les données relatives à la France. On la considère uniforme égale à 0.394 dollars USD.

Effet du confinement:

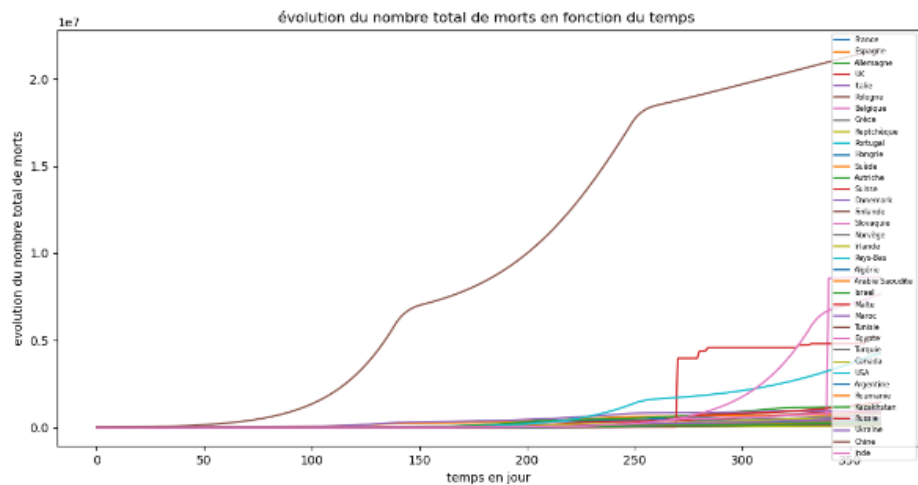
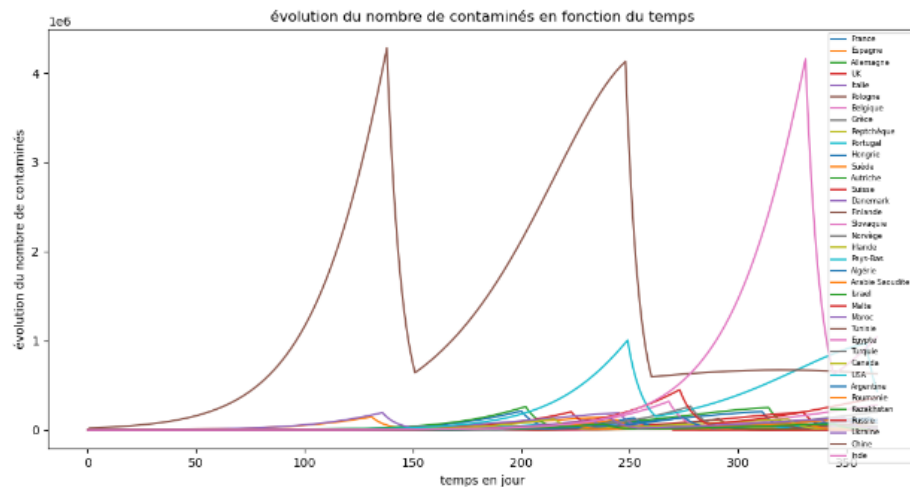
- Les pays ne peuvent ni exporter ni importer de nourriture



Biens alimentaires: 30 ans de stock

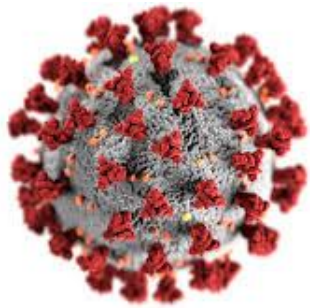
NOMBRE DE CONTAMINÉS

On observe la même tendance que lorsque l'on ne prenait pas en compte les stocks.



NOMBRE DE MORTS

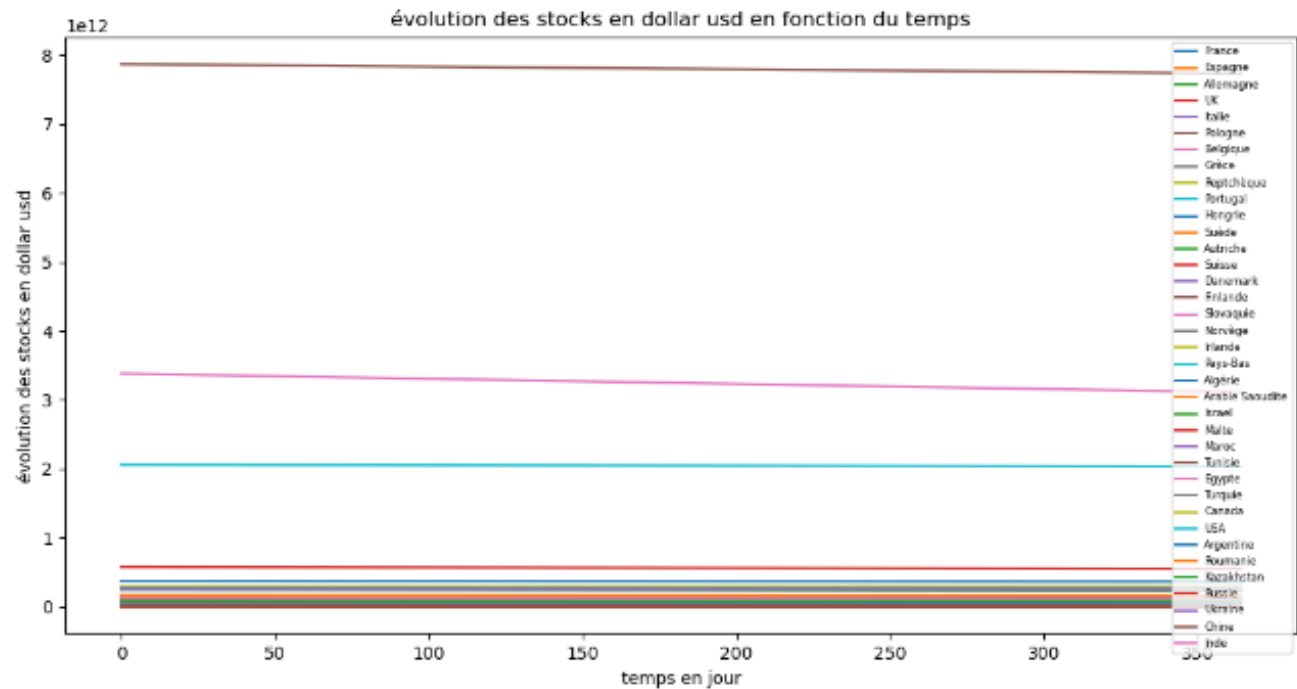
On retombe sur une courbe analogue au début de l'étude ce qui paraît cohérent. En effet, en se plaçant avec 30 ans de stocks, on a des stocks « infinis » donc il n'y aura aucun mort à cause de pénurie. Donc on a seulement l'effet de l'épidémie.

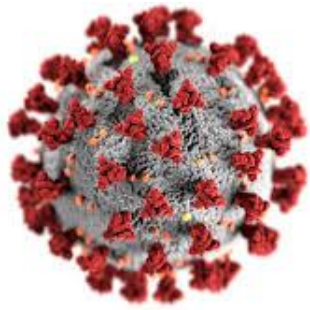


Biens alimentaires: 30 ans de stock

STOCKS

On ne retombe pas à 0 ce qui est cohérent car aucun pays ne perd toute sa population.



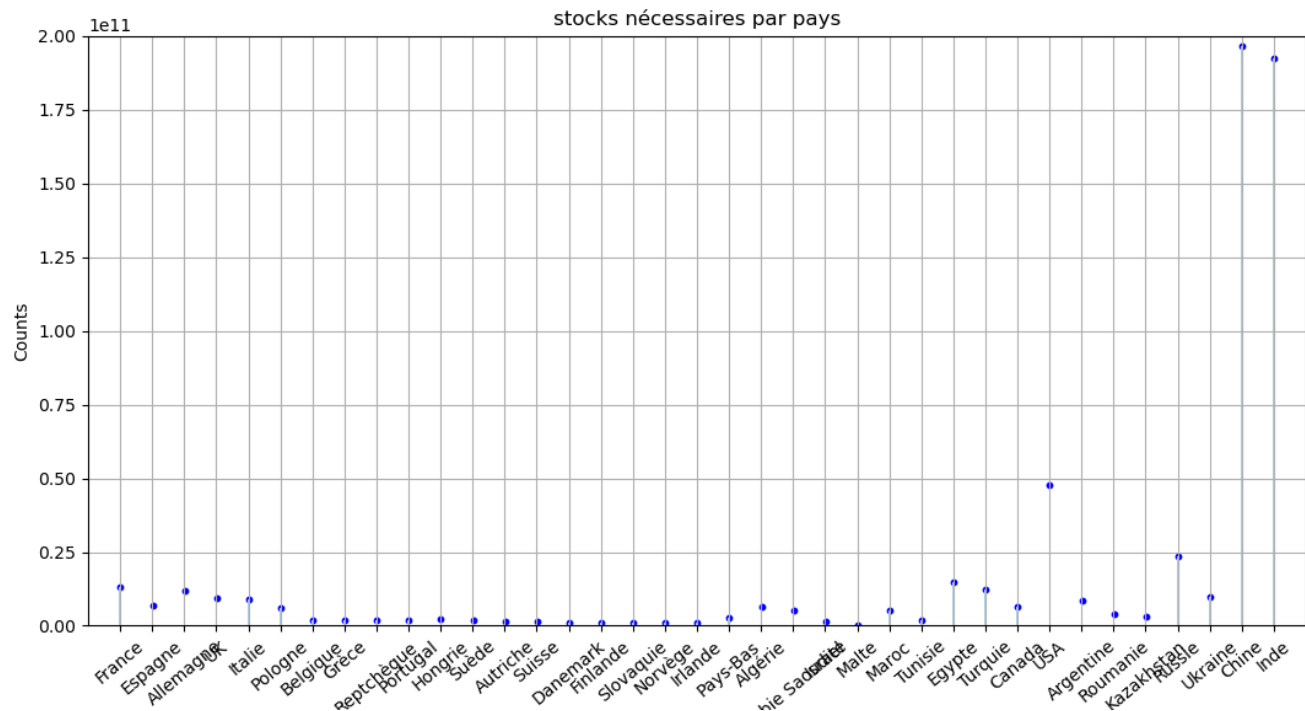


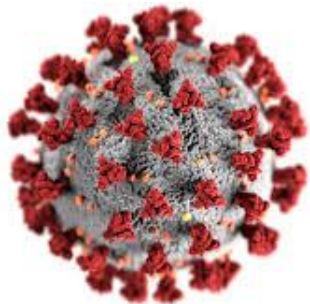
Biens alimentaires : prévision des stocks

PREVISION DES STOCKS NECESSAIRES

Avec le modèle considéré, on peut estimer les stocks nécessaires par pays pour survivre à un an de pandémie.

L'hétérogénéité provient à la fois des grandes différences de population ainsi que d'auto-suffisance en céréales.



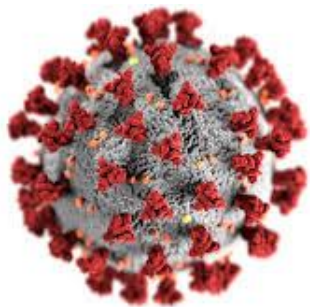


Biens alimentaires : prévision des stocks

PROPORTION DE LA PRODUCTION PAR AN

Ici, on évalue l'autosuffisance des pays en céréales. Le constat est que les pays de l'AFNMO sont les plus dépendants des imports-exports. La Chine plus nombreuse a besoin de 0.75 ans de production pour survivre à un an de pandémie, contrairement à l'Algérie qui a besoin de plus de 5 ans de production, ce qui illustre la dépendance en céréales.

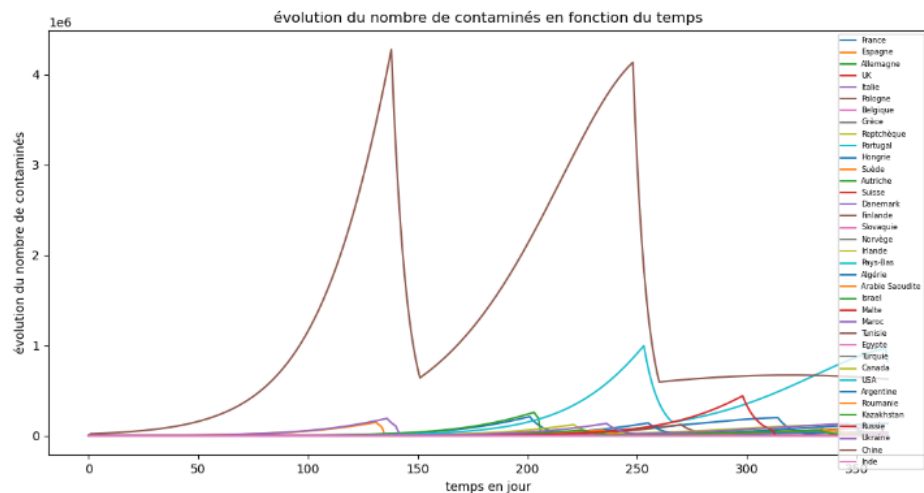
Pays	Stock nécessaire (10^9 USD)	Proportion de la production par an
France	13,18	1.07
Espagne	6,89	1.94
Allemagne	11,97	1.47
UK	9,35	2.14
Italie	8,79	2.04
Pologne	6,06	1.12
Belgique	2,00	4.36
Grèce	1,76	2.00
Republique Tchèque	1,67	1.20
Portugal	1,68	6.62
Hongrie	2,44	1.06
Suède	1,74	1.77
Autriche	1,52	2.15
Suisse	1,53	3.81
Danemark	1,07	0.60
Finlande	1,13	1.99
Slovaquie	1,01	16.94
Norvège	1,02	2.21
Irlande	0,95	2.76
Algérie	6,25	5.13
Arabie Saoudite	5,13	8.68
Israël	1,38	11.04
Malte	0,27	21.74
Maroc	5,13	2.18
Tunisie	2,03	5.72
Egypte	14,62	3.39
Turquie	12,14	1.42
Canada	6,54	0.67
USA	47,62	0.69
Argentine	8,72	0.99
Roumanie	3,99	0.74
Kazakhstan	2,97	1.07
Russie	23,35	1.22
Ukraine	9,66	1.15
Chine	196,70	0.75
Inde	192,20	1.71



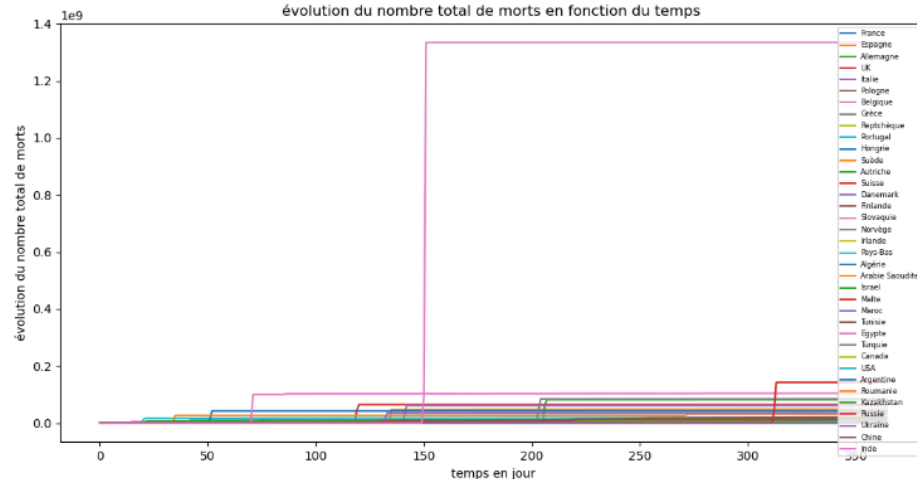
Biens alimentaires: 1 an de stock

NOMBRE DE CONTAMINÉS

On observe des courbes analogues en terme de forme mais il n'y a que 2 vagues car après cela toute la population est morte de faim.

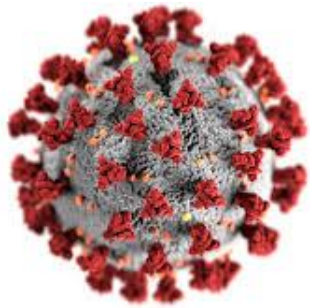


évolution du nombre total de morts en fonction du temps



NOMBRE DE MORTS

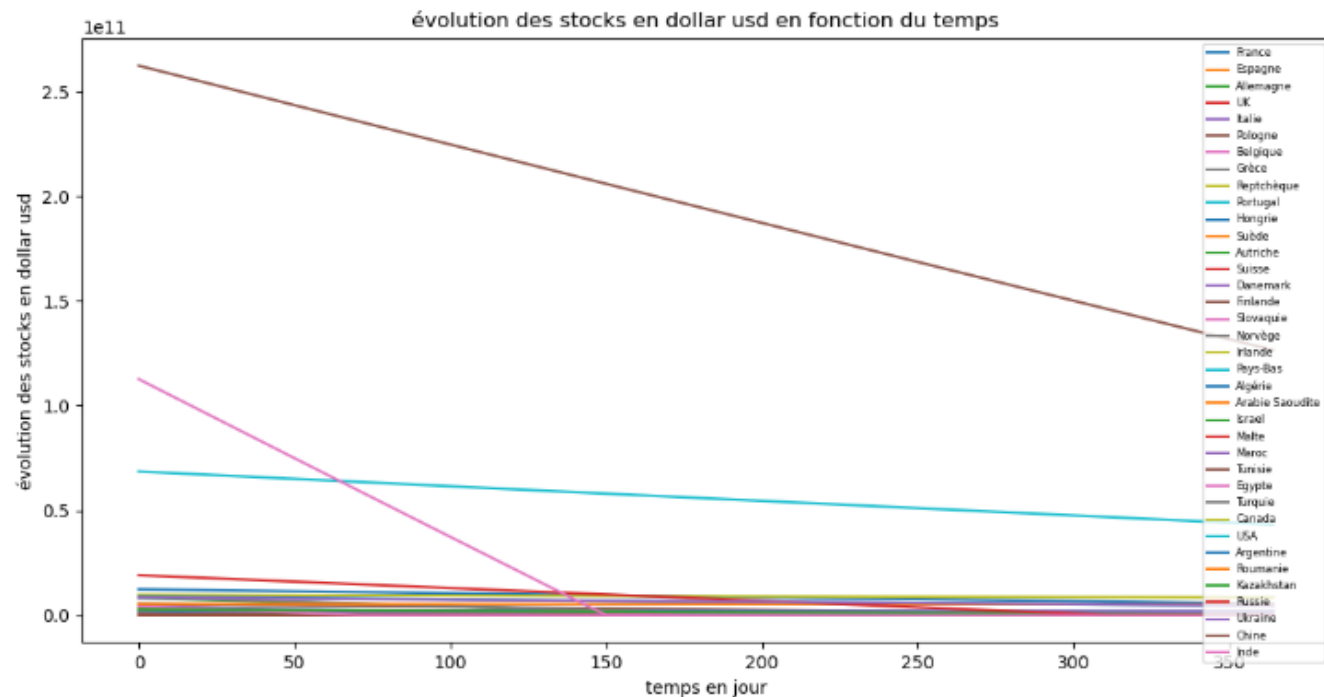
On n'observe plus du tout de courbes analogues à lorsque l'on ne prenait pas les stocks en compte, mais des paliers. Les pays « meurent » complètement à la fin, à cause donc d'un manque de nourriture.

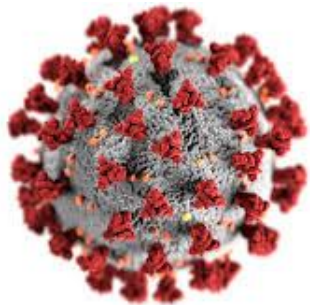


Biens alimentaires: 1 an de stock

STOCKS

Les stocks chutent tous à zéro petit à petit, ce qui est cohérent avec la « mort » massive de nombreux pays. Quand ils n'ont plus à manger, ils meurent.





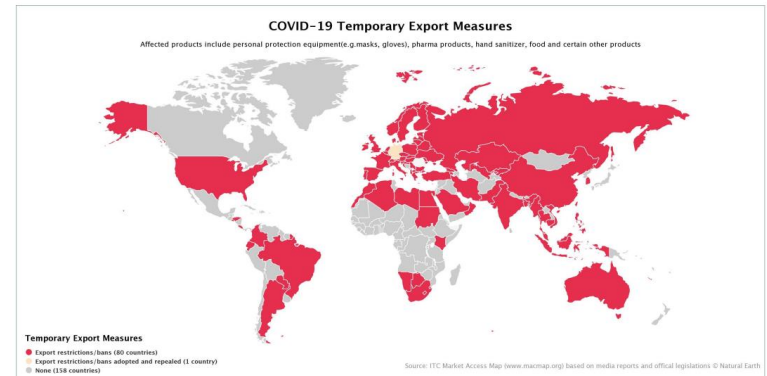
Situation réelle : le blé

D'après les données de l'ITC (International Trade Center), **80 pays ont adopté des mesures de restrictions** aux importations tandis que **57 ont adopté des mesures de libéralisation** sur leurs importations pour favoriser les approvisionnements.

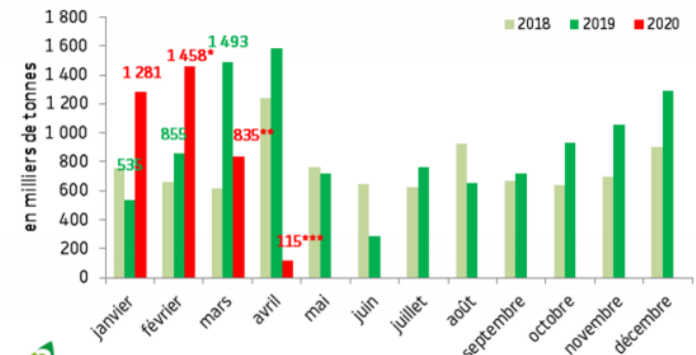
Exemple: La **Russie** a décidé de limiter ses exportations de céréales à 7 millions de tonnes pour la période d'avril à juin 2020. De fait, pour limiter l'inflation sur les produits issus de la transformation céréalière comme le pain, **l'Ukraine** a modifié ses exportations de blé à 20.2 millions de tonnes.

On a observé des **appels d'offre de l'Algérie et l'Egypte** en lien avec le blé par exemple car ce sont de gros importateurs. La **France** a donc répondu à ces appels d'offre. Ses exportations de blé vers l'Algérie et le Maroc ont sur cette période représenté 62% de ses exportations totales.

Carte - Mesures temporaires aux exportations dans le contexte Covid-19



Graphique 2 - Exportations françaises de blé vers les pays tiers

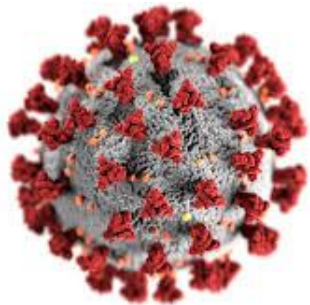


*d'après estimations Reuters

**d'après Commission européenne, calculs APCA

***Données arrêtées au 6 avril 2020

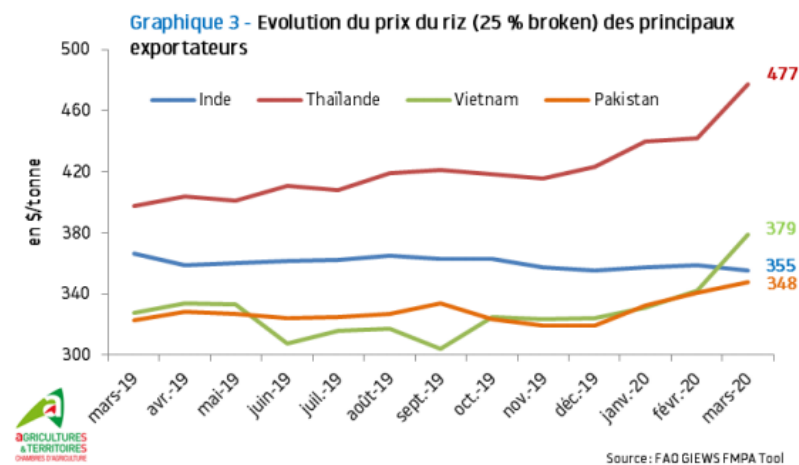


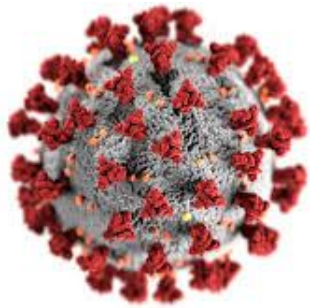


Situation réelle : le riz, le maïs

Le **riz** est également crucial pour les pays d'Asie. Le Vietnam, Myanmar, le Cambodge ont suspendu leurs certificats d'exportation sur le riz. Il ne reste alors que la **Thaïlande et l'Inde**. La Thaïlande était plus à même de fournir les pays dépendants en terme de stocks mais également car l'Inde a confiné pendant 3 semaines entraînant une **fermeture des ports et plus de 500 000 tonnes de riz qui sont restées à quai**. D'où la hausse du prix du riz thaïlandais.

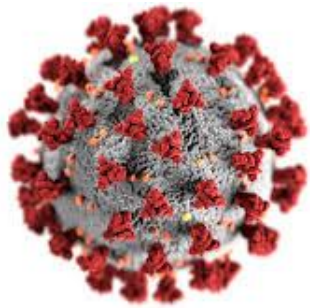
En Afrique, une menace accrue. La fermeture des frontières fragilise les équilibres alimentaires en entravant le commerce transfrontalier et intercontinental. La céréale ici est le **maïs**. Des pays comme le **Bénin, le Sénégal et la Côte d'Ivoire** font partie des premiers importateurs mondiaux de riz également. La **Russie** pèse également pour **40% des approvisionnements du Nigéria** par exemple d'où la peur suite aux limitations des exportations.





Conclusion et limites des modèles

	Epidémiologie	Biens alimentaires
Mesures retenues	<ul style="list-style-type: none"> • Un confinement national ainsi que des restrictions de voyage permettent de réduire le nombre de morts • L'immunité de groupe est bien atteinte pour environ 67% de la population contaminée 	<p>A partir du tableau des stocks nécessaires, on peut s'assurer de prévoir une sécurité pour les pays non autosuffisants</p>
Limites	<ul style="list-style-type: none"> • On a considéré des pays ayant un taux de contact uniforme ce qui est faux (Suède – Inde) • On a considéré un confinement unique aussi strict/ laxiste dans chaque pays (Brésil – Espagne) 	<ul style="list-style-type: none"> • On a considéré une production/consommation par habitant uniformes ce qui est faux (France – Inde) • On a considéré un arrêt des exportations/importations pendant les confinements, ce qui est faux dans la réalité • On a considéré uniquement le blé, or ce n'est pas la céréale universelle (riz, maïs), on n'a pas considéré la viande et ses dérivé(e)s, ni les légumes, etc...



Annexes

Annexe 1 : Zoom sur la courbe des morts

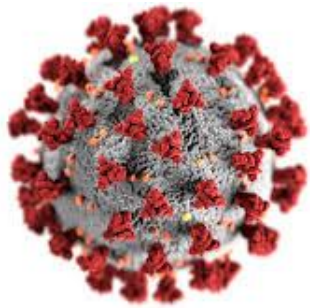
Annexe 2 : Taux de contaminés 1 an de stock

Annexe 3 : Taux de contaminés 30 ans de stock

Annexe 4 : Cas particulier : la France, cohérence des simulations

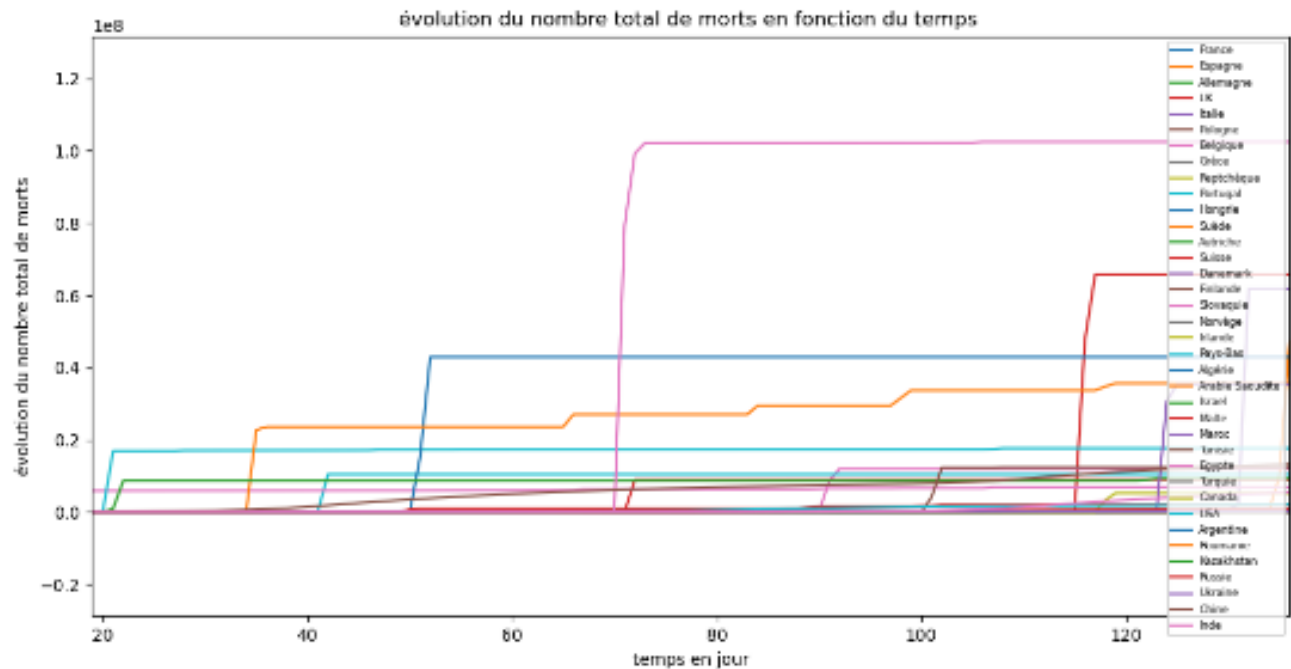
Annexe 5 : Cas particulier : la France, cohérence avec l'épidémie réelle dans le pays

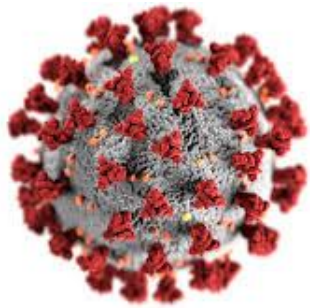
Annexes jusqu'à la fin : Programmes Python



Annexe - Zoom sur la courbe des morts

Contrairement à ce que l'on pouvait penser à vue d'œil, les pays passent par des paliers de morts et non pas d'un coup. En effet, dans le programme on nourrit le plus de personnes possibles chaque jour.

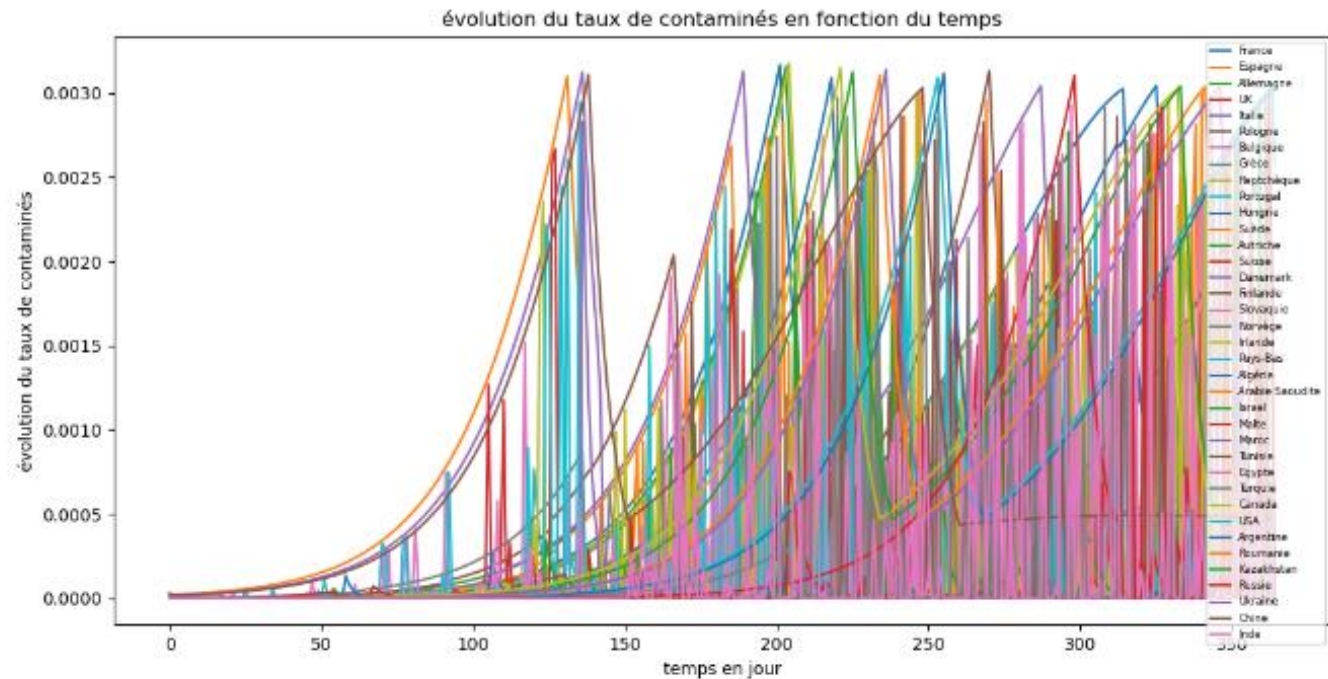


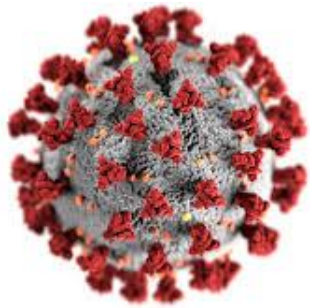


1 an de stock, taux de contaminés

TAUX DE CONTAMINES

De manière plutôt surprenante, on arrive à des taux de contaminés nul. Cela s'explique par la « mort » de pays. Quand il n'y a plus d'habitants, il n'y a plus de contaminés.

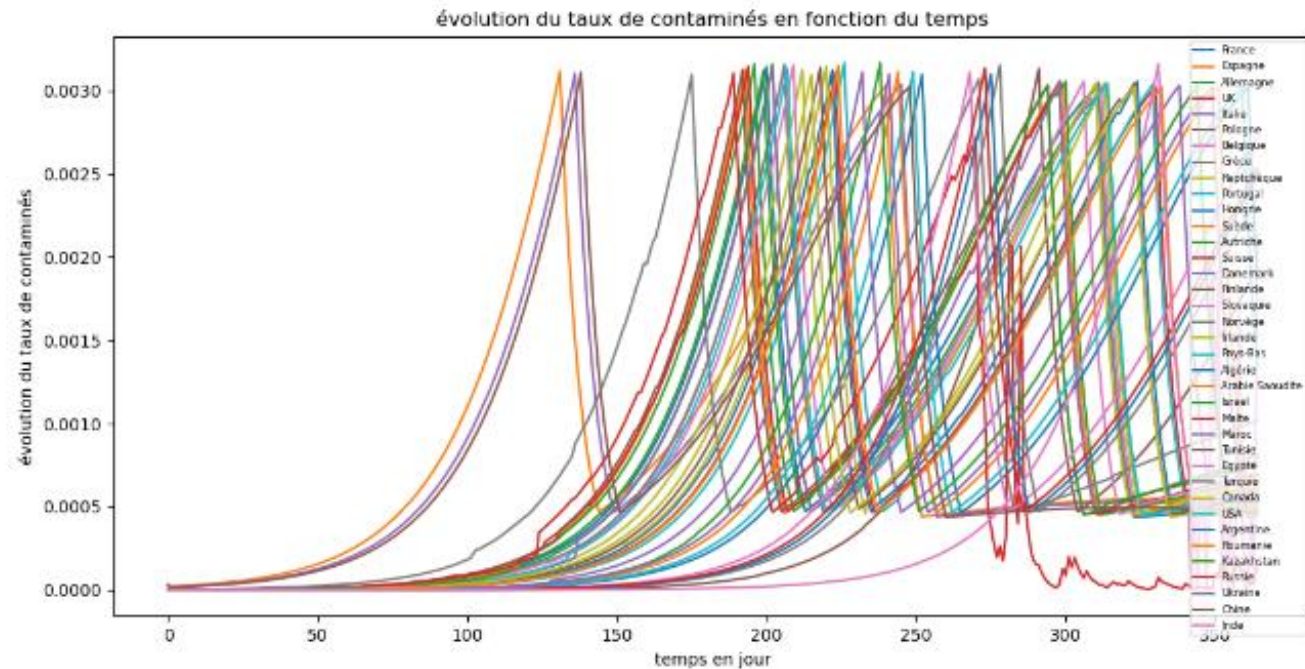


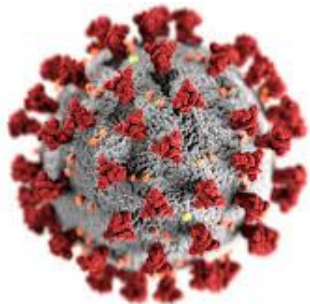


30 ans de stock, taux de contaminés

TAUX DE CONTAMINÉS

On ne retombe pas à 0 car aucun pays ne perd toute sa population.

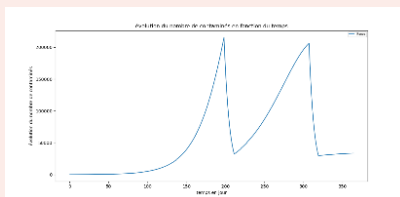




Cas particulier : la France

La France est censée d'après les prévisions effectuées survivre à un an de pandémie, car elle est autosuffisante en production de céréales. C'est bien ce que l'on observe.

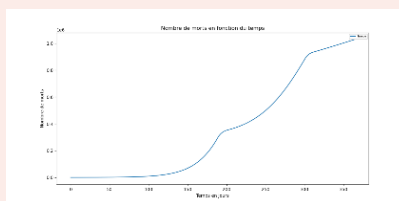
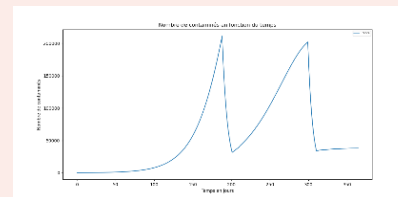
Courbes obtenues avec prise en compte des stocks



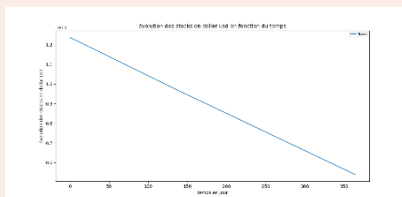
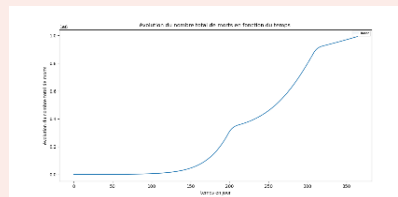
Analyse

On a les mêmes courbes étant donné que la France a assez de stocks pour survivre, c'est cohérent.

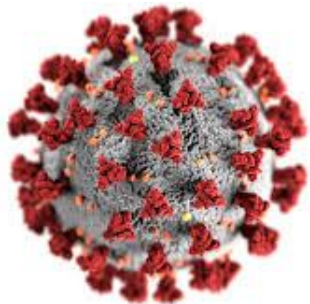
Comparaison avec stratégies de confinement



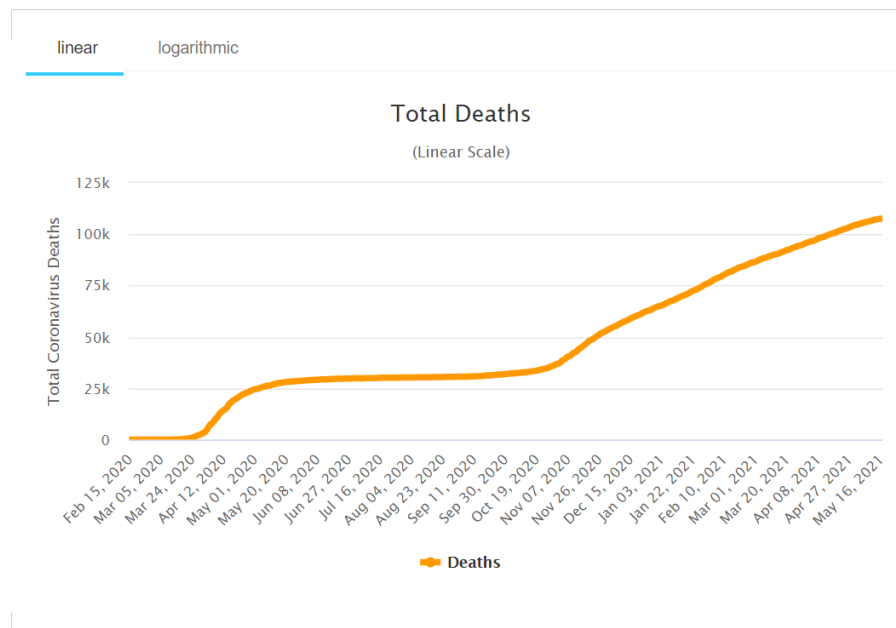
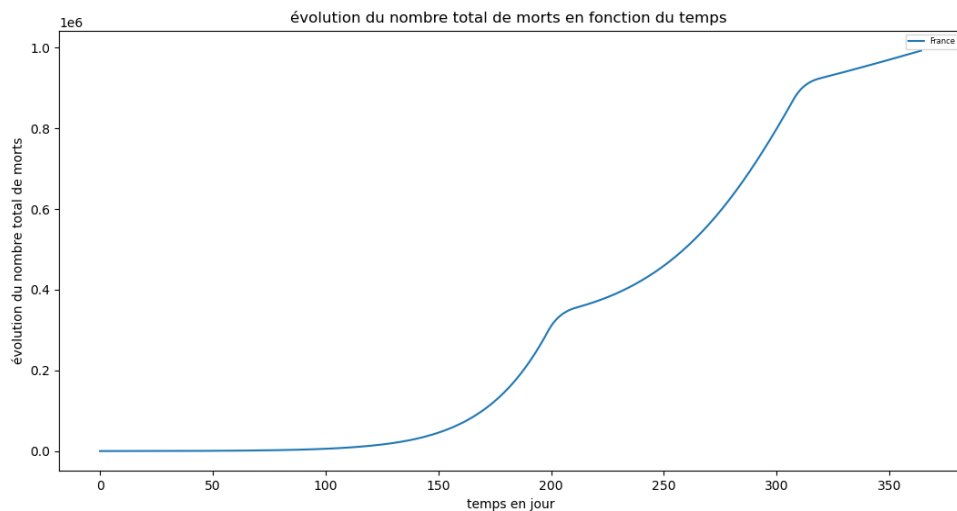
On a les mêmes courbes étant donné que la France a assez de stocks pour survivre, c'est cohérent.



Les stocks baissent au fil du temps car les gens mangent. La vitesse de décroissance est assez cohérente puisque la France consomme 90% de ses stocks sur 1 an.



Cas particulier : la France



Remarque : il est plus pertinent de comparer les morts plutôt que les contaminés vis-à-vis de la data disponible

Source: worldometer

```
# épidémie sans confinement.py
```

```
0001 ##data
0002 NOMS PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
0003 "Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie", "Norvège", "Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israël", "Malte", "Maroc",
0004 "Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]
0005
0006 import numpy as np
0007 import matplotlib.pyplot as plt
0008 import random as rd
0009 from math import floor
0010 from math import ceil
0011
0012 #données relatives aux pays (nombre hab)
0013 #AFMMO où on a de la donnée:
0014 algerie=[42230000, 20]
0015 arabiesaudite=[34813900, 21]
0016 israel=[8655540, 22]
0017 malte=[514564, 23]
0018 maroc=[34660000, 24]
0019 tunisie=[11180000, 25]
0020 egypte=[102334000, 26]
0021 turquie=[84339100, 27]
0022 AFMMO=[algerie, arabiesaudite, israel, malte, maroc, tunisie, egypte, turquie]
0023
0024 #europe
0025 france=[66352469, 0]
0026 espagne=[46439864, 1]
0027 allemagne=[81174000, 2]
0028 UK=[64767115, 3]
0029 italie=[60795612, 4]
0030 pologne=[38005614, 5]
0031 belgique=[11258434, 6]
0032 grece=[10812467, 7]
0033 reptcheque=[10538275, 8]
0034 portugal=[10374822, 9]
0035 hongrie=[9849000, 10]
0036 suedes=[9747355, 11]
0037 autriches=[8584926, 12]
0038 suisse=[8236573, 13]
0039 danemark=[5659715, 14]
0040 finlande=[5471553, 15]
0041 slovaquie=[5421349, 16]
0042 norvege=[5165882, 17]
0043 irlande=[4625885, 18]
0044 paysbas=[16900726, 19]
0045 europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suedes, autriches, suisse, danemark, finlande,
0046 slovaquie, norvege, irlande, paysbas]
0047
0048 #autres pays importants
0049 canada=[37590000, 28]
0050 USA=[328200000, 29]
0051 argentine=[44490000, 30]
0052 roumanie=[19410000, 31]
0053 kazakhstan=[18280000, 32]
0054 russie=[144500000, 33]
0055 ukraine=[41980000, 34]
0056 chine=[1393000000, 35]
0057 inde=[1353000000, 36]
0058 autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]
0059
0060 monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suedes, autriches, suisse, danemark, finlande,
0061 slovaquie, norvege, irlande, paysbas, algerie, arabiesaudite, israel, malte, maroc, tunisie, egypte, turquie, canada, USA, argentine, roumanie, kazakhstan,
0062 russie, ukraine, chine, inde]
0063
0064 ##liste avec les numéros des pays concernés
0065 def listenum(H):
0066     listenum=[0]*(len(H))
0067     for i in range(len(H)):
0068         listenum[i]=H[i][1]
0069     return listenum
0070
0071 ##creer seuil depart
0072 def seuildep(T, sc, sd, S):
0073     for i in range(len(T)):
0074         if S[i]==0:
0075             if T[i]>=sc:
0076                 S[i]=1
0077             else:
0078                 S[i]=0
0079         else:
0080             if T[i]>sd:
0081                 S[i]=1
0082             else:
0083                 S[i]=0
0084     return S
0085
0086 ##taux
0087 def taux(C, L):
0088     T=[0]*(len(C))
0089     for i in range(len(C)):
0090         if (L[i][0]+L[i][1]+L[i][2])==0:
0091             T[i]=0
0092         else:
0093             T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
0094     return (T)
0095
0096 def tauxlocal(C, L, i):
0097     T=0
0098     if (L[i][0]+L[i][1]+L[i][2])==0:
0099         T=0
0100     else:
0101         T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
0102     return (T)
0103
0104 ##creer tableau
0105 def creer tableau(n, a):
0106     N=[0]*n
0107     for i in range(n):
0108         N[i]=[0]*a
0109     return N
0110
0111 ##SIRevol
0112 #Lt=[St, It, Rt, Dt] #D les dead
0113 #d le taux de létalité différent du taux de mortalité
0114 def SIRevol(Lt, beta): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
0115     #modèle d'évolution de t à t+1
0116     gamma=0.3
0117     d=0.048
0118     N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
0119     if N==0:
```

```

115         return ([0, 0, 0, 0])
116     else:
117         L1=[0,0,0,0]
118         L1[0]= (-beta*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
119         L1[1]= beta*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
120         L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
121         L1[3]= Lt[3]+d*Lt[1]
122         Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
123         return (Lt) #on renvoie la liste au temps t+1
124
125
126 ##renvoyer une case
127 def case(Lt,n):
128     return (Lt[n])
129
130 ##creerLt pour le début seulement
131 def creerLt(C, Pop, i): #créé un Lt pour le pays en position i
132     Lt=[0, 0, 0, 0]
133     Lt[0]=Pop[i]-C[i]
134     Lt[1]=C[i]
135     return Lt
136
137 ##creerL au depart
138 def creerL(C, Pop): #créé la liste des Lt-> cad on a L[0][0] qui donne le nombre de sains dans le pays 0
139     #L[0][1] le nombre de contaminés du pays 0
140     #L[0][2] le nombre de recovered du pays 1
141     L=creertableau(len(C), len(C))
142     for i in range (len(C)):
143         L[i]=creerLt(C,Pop,i)
144     return L
145
146 ##verifs
147 def verifs(S):
148     n=len(S)
149     res=0
150     for i in range (n):
151         if S[i]==1:
152             res=res+1
153     if res==n:
154         return False
155
156 ##creerV V=voyageurs en prenant en compte les stratégies de confinement
157
158 def creerV(L, C, Pop):
159     paspossible=[]
160     for i in range (len(C)):
161         if Pop[i]==0:
162             paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
163     V=[0]*(len(C))
164     for i in range (len(C)):
165         V[i]=[0]*2
166     for i in range (len(C)):
167         V[i][1]= rd.randint(0,(len(C)-1))
168         while V[i][1] in paspossible :
169             V[i][1]=rd.randint(0, len(C)-1)
170         V[i][0]=rd.uniform(0, 0.0002)*(L[i][0]+L[i][1]+L[i][2])#on ne veut pas que trop de la population parte d'un pays sinon ce n'est pas cohérent
171
172     return (V)
173
174 ##creerD
175 def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
176     D=[0]*(len(C))
177     T=taux(C, L)
178     for i in range (len(C)):
179         D[i]=[0]*2
180     for i in range (len(C)):
181         D[i][0]=(V[i][0]*T[i])
182         D[i][1]=V[i][1]
183     return (D)
184
185 ##trouver les contaminés
186 def recherche(D, i):
187     c=0 #res
188     for k in range (len(D)):
189         if D[k][1]==i:
190             c=c+D[k][0]
191     return c
192
193 ##expansion sans confinement
194 def expansionnonconfined(C, H, n):# d le taux de gens qui meurent
195     Pop=[0]*len(H)
196     beta=0.39
197     gamma=0.3
198     d=0.048
199     for i in range (len(H)):
200         Pop[i]=H[i][0]
201     L=creerL(C, Pop)
202     V=creerV(L,C,Pop)
203     D=creerD(Pop,C,V,L)
204     evolcontamines=creertableau(len(C), n) #evolution des contaminés
205     evolmorts=creertableau(len(C),n)
206     for k in range (n):
207         V=creerV(L,C, Pop)
208         D=creerD(Pop,C,V,L)
209         for i in range (len(C)):
210             L[i]=SIREvol(L[i], beta) #épidémie en interne
211             C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine par les voyages
212             Pop[i]=Pop[i]+recherche(V,i)-V[i][0] #on voyage en global
213             L[i]=[Pop[i]-C[i]-L[i][2]-L[i][3],C[i],L[i][2],L[i][3]]
214             evolcontamines[i][k]=C[i]
215             evolmorts[i][k]=L[i][3]
216
217     return (evolcontamines, evolmorts)
218
219 ##tracer
220 def tracercontamines(C, H, n):
221     evol=expansionnonconfined(C, H, n)[0]
222     abscisse=[0]*n
223     liste=listenum(H)
224     for i in range (n):
225         abscisse[i]=i
226     for k in range (len(C)):
227         plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
228     plt.legend()
229     plt.title("Nombre de contaminés en fonction du temps")
230     plt.xlabel("Temps en jours")
231     plt.ylabel("Nombre de contaminés")
232     plt.legend(fontsize=6, loc='best')
233     return(plt.show())
234
235 def tracermorts(C, H, n):

```

```

236     evol=expansionnonconfined(C, H, n)[1]
237     abscisse=[0]*n
238     liste=listenum(H)
239     for i in range(n):
240         abscisse[i]=i
241     for k in range(len(C)):
242         plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[listenum(k)]))
243         plt.legend()
244         plt.title("Nombre de morts en fonction du temps")
245         plt.xlabel("Temps en jours")
246         plt.ylabel("Nombre de morts")
247         plt.legend(fontsize=6, loc='best')
248     plt.show()
249
250 def tracermortsfrance(C, H, n):
251     evol=expansionnonconfined(C, H, n)[1]
252     abscisse=[0]*n
253     liste=listenum(H)
254     for i in range(n):
255         abscisse[i]=i
256     plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[listenum(0)]))
257     plt.legend()
258     plt.title("Nombre de morts en fonction du temps")
259     plt.xlabel("Temps en jours")
260     plt.ylabel("Nombre de morts")
261     plt.legend(fontsize=6, loc='best')
262     plt.show()
263
264 def tracercontaminesfrance(C, H, n):
265     evol=expansionnonconfined(C, H, n)[0]
266     abscisse=[0]*n
267     liste=listenum(H)
268     for i in range(n):
269         abscisse[i]=i
270     plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[listenum(0)]))
271     plt.legend()
272     plt.title("Nombre de contaminés en fonction du temps")
273     plt.xlabel("Temps en jours")
274     plt.ylabel("Nombre de contaminés")
275     plt.legend(fontsize=6, loc='best')
276     plt.show()
277
278 ##application
279 #tracercontamines([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde,
280 365)
281 #tracermorts([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365)
282 #tracercontamines([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10], europe, 365)
283 #tracermorts([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10], europe, 365)
284 #tracercontamines([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365)
285 #tracermorts([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365)
286
287 #tracercontaminesfrance([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,0,1, 0, 20,65,0, 2,
0,0,0,32616,0],monde, 365)
288 #tracermortsfrance([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde,
365)

```

```
# stratégie de confinement épidémie.py
```

```
001 #data
002 NOMS_PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
003 "Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie", "Norvège", "Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israël", "Malte", "Maroc",
004 "Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]
005
006 AFFICHER = ["évolution du nombre total de morts", "évolution du nombre de contaminés"]
007
008 #données relatives aux pays (nombre hab)
009 #AFNMO où on a de la donnée:
010 algerie=[42230000, 20]
011 arabiesaudite=[34813900, 21]
012 israel=[8635540, 22]
013 malte=[514564, 23]
014 maroc=[34660000, 24]
015 tunisie=[11180000, 25]
016 egypte=[102334000, 26]
017 turquie=[84339100, 27]
018 AFNMO=[algerie, arabiesaudite, israel, malte, maroc, tunisie, egypte, turquie]
019
020 #europe
021 france=[66352469, 0]
022 espagne=[46439864, 1]
023 allemagne=[81174000, 2]
024 UK=[64767115, 3]
025 italie=[60795612, 4]
026 pologne=[38005614, 5]
027 belgique=[11258434, 6]
028 grece=[10812467, 7]
029 reptcheque=[10538275, 8]
030 portugal=[10374822, 9]
031 hongrie=[9849000, 10]
032 suede=[9747355, 11]
033 autriches=[8584926, 12]
034 suisse=[8236573, 13]
035 danemark=[5659715, 14]
036 finlande=[5471553, 15]
037 slovaquie=[5421349, 16]
038 norvege=[5165802, 17]
039 irlande=[4625885, 18]
040 paysbas=[16900726, 19]
041 europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
042 slovaquie, norvege, irlande, paysbas]
043
044 #autres pays importants
045 canada=[375900000, 28]
046 USA=[328200000, 29]
047 argentine=[44490000, 30]
048 roumanie=[19410000, 31]
049 kazakhstan=[18280000, 32]
050 russie=[144500000, 33]
051 ukraine=[41980000, 34]
052 chine=[1393000000, 35]
053 inde=[1353000000, 36]
054 autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]
055
056 monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
057 slovaquie, norvege, irlande, paysbas, algerie, arabiesaudite, israel, malte, maroc, tunisie, egypte, turquie, canada, USA, argentine, roumanie, kazakhstan,
058 russie, ukraine, chine, inde]
059
060 import numpy as np
061 import matplotlib.pyplot as plt
062 import random as rd
063 from math import floor
064 from math import ceil
065
066 ##liste avec les numéros des pays concernés
067 def listenum(H):
068     listenum=[0]*(len(H))
069     for i in range (len(H)):
070         listenum[i]=H[i][1]
071     return (listenum)
072
073 ##creer seuil depart
074 def seuildep(T, sc, sd, S):
075     for i in range (len(T)):
076         if S[i]==0:
077             if T[i]>=sc:
078                 S[i]=1
079             else:
080                 S[i]=0
081         else:
082             if T[i]>sd:
083                 S[i]=1
084             else:
085                 S[i]=0
086     return S
087
088 ##taux
089 def taux(C, L):
090     T=[0]*(len(C))
091     for i in range (len(C)):
092         if (L[i][0]+L[i][1]+L[i][2])==0:
093             T[i]=0
094         else:
095             T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
096     return (T)
097
098 def tauxlocal(C, L, i):
099     T=0
100     if (L[i][0]+L[i][1]+L[i][2])==0:
101         T=0
102     else:
103         T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
104     return (T)
105
106 ##creer tableau
107 def creertableau(n, a):
108     N=[0]*n
109     for i in range (n):
110         N[i]=[0]*a
111     return N
112
113 ##SIRevol
114 #Lt=[St, It, Rt, Dt] #Dt les dead à t
115 #d le taux de létalité =! taux de mortalité
116 def SIRevol(Lt, beta, i, k): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
117 #modèle d'évolution de t à t+1
118 gamma=0.3
119 d=0.048
120 N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
```

```

115     if N==0:
116         return ([0, 0, 0, 0])
117     else:
118         L1=[0,0,0,0]
119         L1[0]= (-beta[i][k]*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
120         L1[1]= beta[i][k]*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
121         L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
122         L1[3]= Lt[3]+d*Lt[1]
123         Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
124         return (Lt) #on renvoie la liste au temps t+1
125
126
127 ##renvoyer une case
128 def case(Lt,n):
129     return (Lt[n])
130
131 ##creerLt pour le début seulement
132 def creerLt(C, H, i): #crée un Lt pour le pays en position i
133     Lt=[0, 0, 0, 0]
134     Lt[0]=H[i][0]-C[i]
135     Lt[1]=C[i]
136     return Lt
137
138 ##creerL au depart
139 def creerL(C, H): #crée la liste des Lt-> cad on a L[0][0] qui donne le nombre de sains dans le pays 0
140     #L[0][1] le nombre de contaminés du pays 0
141     #L[0][2] le nombre de recovered du pays 1
142     L=creertableau(len(C), len(C))
143     for i in range (len(C)):
144         L[i]=creerLt(C,H,i)
145     return L
146
147 ##verifs
148 def verifs(S):
149     n=len(S)
150     res=0
151     for i in range (n):
152         if S[i]==1:
153             res=res+1
154     if res==n:
155         return False
156
157 ##creerV V=voyageurs en prenant en compte les stratégies de confinement
158 def creerV(L, C, S, Pop):
159     paspossible=[]
160     for i in range (len(C)):
161         if Pop[i]==0:
162             paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
163     V=[0]*(len(C))
164     for i in range (len(C)):
165         V[i]=0*2
166     if verifs(S)==False:
167         for i in range (len(C)):
168             V[i][0]=0
169     else :
170         for i in range (len(C)):
171             if S[i]==0:
172                 V[i][1]= rd.randint(0,(len(C)-1))
173                 while S[V[i][1]]==1 :
174                     V[i][1]=rd.randint(0, len(C)-1)
175                 while V[i][1] in paspossible :
176                     V[i][1]=rd.randint(0, len(C)-1)
177                 V[i][0]=rd.uniform(0, 0.0002)*(L[i][0]+L[i][1]+L[i][2])#on ne veut pas que plus de la moitié de la population parte sinon pas
178                 V[i][0]=rd.uniform(0, 0.0002)*(L[i][0]+L[i][1]+L[i][2])#on ne veut pas que plus de la moitié de la population parte sinon pas
179             else:
180                 V[i][0]=0
181     return (V)
182
183 ##creerD
184 def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
185     D=[0]*(len(C))
186     T=taux(C, L)
187     for i in range (len(C)):
188         D[i]=0*2
189     for i in range (len(C)):
190         D[i][0]=(V[i][0]*T[i])
191         D[i][1]=V[i][1]
192     return (D)
193
194 ##trouver les contaminés
195 def recherche(D, i):
196     c=0 #res
197     for k in range (len(D)):
198         if D[k][1]==i:
199             c=c+D[k][0]
200     return c
201
202 ##creer beta
203 def creertableaubeta(n,C):
204     beta=creertableau(len(C), n+1)
205     for i in range (len(C)):
206         beta[i][0]=0.39
207     return beta
208
209 ##expansion avec confinement
210 def expansionconfined(C, H, n, sc, sd):# d le taux de gens qui meurent
211     Pop=[0]*len(H)
212     for i in range (len(H)):
213         Pop[i]=H[i][0]
214     L=creerL(C, H)
215     beta=creertableaubeta(n, C)
216     gamma=0.3
217     d=0.040
218     S=[0]*(len(C))
219     T=taux(C, L)
220     S=seuildep(T,sc, sd,S)
221     V=creerV(L,C,S, Pop)
222     D=creerD(H,C,V,L) #D le tableau des gens dangereux qui se déplacent et vers quel pays ils se déplacent
223     evolcontaminés=creertableau(len(C), n) #evolution des contaminés
224     evolmorts=creertableau(len(C),n)
225     listedeS=creertableau(len(C),n)
226     for k in range (n):
227         T=taux(C, L)
228         S=seuildep(T,sc, sd ,S)
229         V=creerV(L,C,S, Pop)
230         D=creerD(H,C,V,L)
231         for i in range (len(C)):
232             L[i]=SIRevol(L[i], beta, i, k)
233             C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine
234

```

```

235     Pop[i]=Pop[i]+recherche(V,i)-V[i][0]
236     L[i]=[Pop[i]-C[i]-L[i][2],C[i],L[i][2],L[i][3]]
237     evolcontamines[i][k]=C[i]
238     evolmorts[i][k]=L[i][3]
239     listedeS[i][k]=S[i]
240     if listedeS[i][k]==1:
241         beta[i][k+1]=0.22
242     else:
243         beta[i][k+1]=0.39
244
245     return ([evolmorts, evolcontamines]) #on renvoie la liste des états de chaque pays
246
247
248 ##tracer des évolutions en fonction
249 def tracercontamines(C, H, n,sc, sd):
250     evol=expansionconfined(C, H, n, sc, sd)[1]
251     abscisse=[0]*n
252     liste=listenum(H)
253     for i in range (n):
254         abscisse[i]=i
255     for k in range (len(C)):
256         plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[listenum[k]]))
257         plt.legend()
258         plt.title("Nombre de contaminés en fonction du temps")
259         plt.xlabel("Temps en jours")
260         plt.ylabel("Nombre de contaminés")
261         plt.legend(fontsize=6, loc='best')
262     return(plt.show())
263
264 def tracermorts(C, H, n,sc, sd):
265     evol=expansionconfined(C, H, n,sc, sd)[0]
266     abscisse=[0]*n
267     liste=listenum(H)
268     for i in range (n):
269         abscisse[i]=i
270     for k in range (len(C)):
271         plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[listenum[k]]))
272         plt.legend()
273         plt.title("Nombre de morts en fonction du temps")
274         plt.xlabel("Temps en jours")
275         plt.ylabel("Nombre de morts")
276         plt.legend(fontsize=6, loc='best')
277     plt.show()
278
279
280 def tracermortsfrance(C, H, n,sc, sd):
281     evol=expansionconfined(C, H, n,sc, sd)[0]
282     abscisse=[0]*n
283     liste=listenum(H)
284     for i in range (n):
285         abscisse[i]=i
286     plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[listenum[0]]))
287     plt.legend()
288     plt.title("Nombre de morts en fonction du temps")
289     plt.xlabel("Temps en jours")
290     plt.ylabel("Nombre de morts")
291     plt.legend(fontsize=6, loc='best')
292     plt.show()
293
294 def tracercontaminesfrance(C, H, n,sc, sd):
295     evol=expansionconfined(C, H, n,sc, sd)[1]
296     abscisse=[0]*n
297     liste=listenum(H)
298     for i in range (n):
299         abscisse[i]=i
300     plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[listenum[0]]))
301     plt.legend()
302     plt.title("Nombre de contaminés en fonction du temps")
303     plt.xlabel("Temps en jours")
304     plt.ylabel("Nombre de contaminés")
305     plt.legend(fontsize=6, loc='best')
306     plt.show()
307
308 ##application
309 #tracercontamines([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde,
310 365, 0.003, 0.0006)
311 #tracermorts([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
312 0.003, 0.0006)
313 #tracercontamines([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10], europe, 365,0.003, 0.0006)
314 #tracermorts([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365, 0.003, 0.0006)
315
316 #tracermortsfrance([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde,
317 365, 0.003, 0.0006)
318 #tracercontaminesfrance([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2,
319 0,0,0,32616,0],monde, 365, 0.003, 0.0006)

```


épidémie et flux.py

```
0001| ##data
0002|
0003| NOM PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
0004| "Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie", "Norvège", "Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israël", "Malte", "Maroc",
0005| "Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]
0006|
0007| AFFICHER = ["évolution du nombre total de morts", "évolution du nombre de contaminés", "évolution des stocks en dollar usd", "évolution du nombre de
0008| vivants", " évolution des exports", "évolution du taux de contaminés", " évolution des imports"]
0009|
0010| AFFICHERBIS=["évolution du nombre total de morts en fonction du temps ", "évolution du nombre de contaminés en fonction du temps", "évolution des
0011| stocks en dollar usd en fonction du temps", "évolution du nombre de vivants en fonction du temps", " évolution des exports en fonction du temps",
0012| "évolution du taux de contaminés en fonction du temps", " évolution des imports en fonction du temps"]
0013|
0014| #données relatives aux pays (nombre hab, production 2017, consommation par habitant, numéro du pays, quantité exportée par an)
0015| #production en dollars usd DE CEREALES par an
0016|
0017| #AFNMO où on a de la donnée:
0018| algerie=[42230000,1217055450, 0.393750627, 20, 0]
0019| arabiesaudite=[34813900,591101519, 0.393750627, 21, 0]
0020| israel=[8655540, 125803727, 0.393750627, 22, 0]
0021| malte=[514564, 12295953, 0.393750627, 23, 0]
0022| maroc=[34660000, 2350715863, 0.393750627, 24, 0]
0023| tunisie=[11180000,354334516, 0.393750627, 25, 0]
0024| egypte=[102334000, 4303123738, 0.393750627, 26,0]
0025| turquie=[84339100, 8556108904, 0.393750627, 27, 185858000]
0026| AFNMO=[algerie, arabiesaudite, israel, malte, maroc, tunisie, egypte, turquie]
0027|
0028| #europe
0029| france=[66352469, 12366941872, 0.393750627, 0, 5602340000]
0030| espagne=[46439864, 3547151265, 0.393750627, 1, 0]
0031| allemagne=[81174000, 8131825487, 0.393750627, 2, 2512970000]
0032| UK=[64767115, 4359410688, 0.393750627, 3, 438740000]
0033| italie=[60795612, 4301291729, 0.393750627, 4, 0]
0034| pologne=[38005614, 5429724123, 0.393750627, 5, 978450000]
0035| belgique=[11258434, 460108034, 0.393750627, 6, 0]
0036| grece=[10812467, 877721695, 0.393750627, 7, 0]
0037| reptcheque=[10538275, 1389583863, 0.393750627, 8, 0]
0038| portugal=[10374822, 253498428, 0.393750627, 9, 0]
0039| hongrie=[9849000, 2311067527, 0.393750627, 10, 1747190000]
0040| suedes=[9747355, 983369105, 0.393750627, 11, 0]
0041| autriches=[8584926, 708800003, 0.393750627, 12, 0]
0042| suisses=[8236573, 400991999, 0.393750627, 13, 0]
0043| danemark=[5659715, 1776259292, 0.393750627, 14, 0]
0044| finlandes=[5471553, 565659239, 0.393750627, 15, 0]
0045| slovaquie=[5421349, 59431977, 0.393750627, 16, 0]
0046| norvege=[5165802, 459500938, 0.393750627, 17, 0]
0047| irlande=[4625885, 344572386, 0.393750627, 18, 0]
0048| paysbas=[16900726, 252251731, 0.393750627, 19, 0]
0049| europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suedes, autriche, suisse, danemark, finlande,
0050| slovaquie, norvege, irlande, paysbas]
0051|
0052| #autres pays importants
0053| canada=[37590000,9711442097, 0.393750627, 28, 6305310000]
0054| USA=[328200000,68607473570, 0.393750627, 29,18740200000]
0055| argentine=[44490000,8846326657, 0.393750627, 30, 7511940000]
0056| roumanie=[19410000,5391922117, 0.393750627,31, 2184270000]
0057| kazakhstan=[18280000,2763295937, 0.393750627, 32, 852409000]
0058| russie=[144500000,19082405580, 0.393750627, 33, 8142170000]
0059| ukraine=[41980000,8391279873, 0.393750627, 34, 7947470000]
0060| chine=[1393000000,2.62353E+11, 0.393750627, 35, 721030000]
0061| indes=[1353000000,1.12689E+11, 0.393750627, 36, 7554560000]
0062| autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]
0063|
0064| monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suedes, autriche, suisse, danemark, finlande,
0065| slovaquie, norvege, irlande, paysbas,algerie, arabiesaudite, israel, malte, maroc, tunisie, egypte, turquie,canada, USA, argentine, roumanie, kazakhstan,
0066| russie, ukraine, chine, inde]
0067|
0068| ##exports data
0069| exportsfrance=[0.146140423199625, 20, 2.59628620601895, 21, 0.22394042337746317, 22, 0.024384925299463987, 23,2.461897838345699, 24,
0070| 1.1658194663411847, 25, 0.7627146474383644, 26, 0.1740402455860384, 27, 0,0, 0, 11.60814377532809, 1, 6.408126274848944, 2, 2.909145701872827, 3,
0071| 5.703404947908281, 4, 0.6288688368175116, 5, 13.64848985498942, 6, 0.5503789165705348, 7, 0.207588356659343, 8, 3.0381989251974932, 9,
0072| 0.46307244422208316, 10, 0.11401233614984244, 11, 0.34679945368724713, 12, 0.8546931388491361, 13, 0.30463071389250835, 14,0.011785544860433152, 15,
0073| 0.05441248656141191, 16, 0.017798885524516053, 17, 0.04715103325393965, 18, 0.489925687618346, 19, 0.002637430115825833, 20, 0.05032216660995689, 29,
0074| 0.0005274860231651666, 30, 0.40878659692377084, 31, 0.008319208136776342, 32, 0.3686524460755183, 33, 0.3207567151721212, 34, 0.6010929299405573, 35,
0075| 0.03018727155420082, 36] #export par personne
0076|
0077| exportsespagne=[0]*74
0078|
0079| exportsallemagne=[0.5889693744302363, 20.5.938859733412176, 21, 0.18519476679725036, 22, 0.0020942666371005494, 23, 0.5850025870352576, 24, 0.0, 25,
0080| 0, 26, 0.2414812624732057, 27, 1.6679355458644394, 0.952509424199867, 0, 0, 2,0.7528642176066228, 3, 1.1632296055387192, 4, 0.9158228004040703, 5,
0081| 3.1513292433537834, 6, 0.09580653903959396, 7,0.09580653903959396, 8, 0.7167565969399069, 9, 0.057629290166802176, 10, 0.12348781629585828,
0082| 11,0.5125655998226033, 12, 0.6984009658264961, 13, 0.4261832606499618, 14, 0.016212087614260722, 15, 0.028346514893931554, 16, 0.25332324779609237, 17,
0083| 0.03783231083844581, 18, 5.927218074753, 19,0.0035232956365386656, 20, 0.1677877152782911, 29, 4.927686204942469e-05, 30, 0.01740705151895927, 31,
0084| 0.0002710227412718358, 32, 0.043437553896567865, 33, 0.037684480252297536, 34, 0.009313326927341267, 35, 0,0, 36]
0085|
0086| exportsUK=[0.06600571910606177, 20, 0.008507403795892406, 21, 0.35135114479006824, 22, 0.014096660010253661, 23, 0,0, 24,0.0702825809054487, 25,
0087| 1.5439934293815618e-05, 26, 0.0035511848875775924, 27, 0.2991332870083838, 0.1.0207494960984445, 1, 0.4161216691526248, 2, 0,0, 3, 0.16358610384297648,
0088| 4, 0.014266499287485633, 5, 0.8785785811209902, 6, 0.06625275805476283, 7, 0.011965949077707105, 8, 0.4695129619406392, 9,0.0022542304068970806, 10,
0089| 0.04000486975527627, 11, 0.010900593611433827, 12, 0.005681895820124148, 13, 0.02011823438484175, 14,0.006052454243175723, 15, 0.0007256769118093341, 16,
0090| 0.02118358985111503, 17, 1.469295027268082, 18, 1.0202554182010424, 19, 0.10130140890172427, 28, 0.19211910241794775, 29, 0.00010807954005670933, 30,
0091| 0.0002935387515824968, 31, 0, 0, 32,0,0, 33, 0,0, 34, 0.005975254571706645, 35, 7.71996714690781e-05, 36]
0092|
0093| exportsitalie=[0]*74
0094|
0095| exportspologne=[0.15589802075030285,20.2.5434926534800884, 21, 0,0, 22, 0.05988588948990536, 23, 5.2623804472676064e-05, 24, 0,0,
0096| 25,1.4206585374465994, 26, 0.2910622625383713, 27, 0.10569491128336987, 0.1.547192475301149, 1, 9.243608062745677, 2, 0.9485966994244587, 3,
0097| 0.334161158401493, 4, 0,0, 5, 0.06625336983109917, 6, 0.31953174075808904, 7, 0.5947805500524213, 8, 0.3518953805087848, 9, 0.12716542350822171, 10,
0098| 0.18860371523007102, 11, 0.06609549841768113, 12, 0.026048783213974653, 13, 0.4116760223897448, 14, 0.08803962488278705, 15, 0.09448604093068987,
0099| 16,0.3983095760536851, 17, 0.037731267806900874, 18, 1.0172444523590647, 19, 0.0030521806594152115, 28, 0.007577827844065353, 29, 0,0, 30,
0100| 0.05717576355956254, 31, 0,0, 32, 0.028864156753262822, 33, 0.007946194475374086,34, 0,0, 35, 0,0, 36]
0101|
0102| exportsbelgique=[0]*74
0103|
0104| exportsgrece=[0]*74
0105|
0106| exportsreptcheque=[0]*74
0107|
0108| exportsportugal=[0]*74
0109|
0110| exportshongrie=[0.030663011473246016, 20, 2.659660879277084, 21, 0.032389075032998274, 22, 0.07533759772565743, 23, 1.427251497613971,24, 0,0, 25,
0111| 0,0, 26, 3.052695705147731, 0.6209767489085186, 1, 11.790537110366534, 2, 0.08437404812671337, 3, 57.27058584627881, 4, 4.7326632145395475, 5,
0112| 0.791247842405504, 6, 3.1041730124885776, 7, 1.598334856330592, 8, 0.02832774901005178, 9, 0,0, 10, 0,0,11,1.6961315806931435, 12, 1.7761194029850746,
0113| 13, 0.01015331505736623, 14, 0.0018275967103259215,15.5.084069448674993, 16, 0.0031475276677835314, 17, 0.013808508478018074, 18,2.8561272526371204, 19,
0114| 0.0005076657528683116, 28, 0.007107320540156361, 29, 0,0, 30, 28.19768504416692, 31,0.04457305310183775, 32, 8.774291806274748, 33, 5.544626381977125, 34,
0115| 0.005787389582698751, 35, 0,0, 36]
0116|
0117| exportssuede=[0]*74
0118|
0119| exportsautriche=[0]*74
0120|
0121| exportssuisse=[0]*74
0122|
0123| exportsdanemark=[0]*74
0124|
0125| exportsfinlande=[0]*74
0126|
0127| exportsslovaquie=[0]*74
0128|
0129| exportsnorvege=[0]*74
0130|
0131| exportsirlande=[0]*74
0132|
0133| exportspaysbas=[0]*74
0134|
0135| exportsalgerie=[0]*74
0136|
0137| exportsarabiesaudite=[0]*74
0138|
0139| exportsisrael=[0]*74
0140|
0141| exportsmalte=[0]*74
0142|
0143| exportsmaroc=[0]*74
```

```

0086| exportstunisie=[0]*74
0087| exportsegypte=[0]*74
0088| exportsturquie=[8.29982771929034e-05, 20, 0.01815290891176216, 21, 0.26981554225738713, 22, 0.010825346725302974, 23, 0.0, 24, 0.006888857007010983,
25, 0.0, 26, 0.030294371175409476, 0, 0.013493148492217727, 1, 0.051399647375890894, 2, 0.13569032631365524, 3, 0.06854472006459637, 4, 0.0, 5,
0.03414786261650883, 6, 0.010517067410015046, 7, 0.0006995569077687573, 8, 0.0, 9, 0.014595839889209157, 10, 0.002525519006012632, 11,
0.005821736300245082, 12, 0.00815754495838822, 13, 0.0002134241413531802, 14, 0.0004031344892226737, 15, 0.0, 16, 0.002715229353882126, 17,
4.74275869737338e-05, 18, 0.0194215968631394, 19, 0.04470050071674941, 28, 0.6195702823482822, 29, 0.0, 30, 0.009604086360893108, 31,
0.006758431142850707, 32, 0.003865348378409304, 33, 0.017334783036574968, 34, 1.1856896741843345e-05, 35, 0.0007825551849616607, 36]
0089| exportscanada=[8.608273476988561, 20, 0.911066730779463, 21, 0.0006517690875232775, 22, 0.014285714285714285, 23, 5.052274541101356, 24,
0.528198989928438, 25, 0.04732641660015962, 26, 0.9531258313381218, 27, 0.281511040170258, 0, 1.1488427773343974, 1, 0.2590316573556797, 2,
3.086778398510242, 3, 5.02487363660548, 4, 0.0, 5, 1.3740888534184623, 6, 0.012587098224320073, 7, 0.0, 8, 0.7461558925246076, 9, 0.0, 10,
0.003378558127161479, 11, 0.005826017557861133, 12, 0.4795424314977388, 13, 0.0027400904495876563, 14, 2.6602819898909284e-05, 15, 0.0, 16,
0.000319233837869114, 17, 1.7341846235700984, 18, 0.6632349028997073, 19, 0.0, 20, 28.00053205639798, 29, 0.000239425379099018357, 30, 0.0, 31, 0.0, 32,
0.0022346368715083797, 33, 0.0, 34, 11.695956371375365, 35, 0.0031391327480712957, 36]
0090| exportsusa=[0.5279128580134005, 20, 1.1996739792809263, 21, 0.20313223644119438, 22, 0.0007099329677026203, 23, 0.5454478976234004, 24,
0.009926873857404022, 25, 0.10794028031687996, 26, 0.10457647775746497, 27, 0.0769926873857404, 0, 0.1509354052407069, 1, 0.033769043266301035, 2,
0.18576782449725776, 3, 0.4285588056063376, 4, 0.004722739042656917, 5, 0.018598415600243754, 6, 0.004567336989640463, 7, 6.093845216331505e-05, 8,
0.060856185252894573, 9, 0.0015234613040828763, 10, 0.012635588056063376, 11, 0.006380255941499086, 12, 0.008488726386349787, 13, 0.00551492992078001,
14, 0.0017032297379646556, 15, 0.0, 16, 0.006733698964046313, 17, 0.05971358927483242, 18, 0.12470444850700792, 19, 1.5059689213893968, 28, 0.0, 29,
0.005730652041438147, 30, 0.0005971968312004875, 31, 0.0006703229737964656, 32, 0.010923217550274223, 33, 0.01064594759293114, 34, 4.107876294942108, 35,
0.015426569165143206, 36]
0091| exportsargentine=[19.821532928748034, 20, 8.547336480107889, 21, 0.022454484153742415, 22, 0.0, 23, 5.671813890761969, 24, 1.1792312879298719, 25,
11.572285906945382, 26, 0.05945156214879748, 27, 0.03387278040008991, 0, 0.42063385030343897, 1, 0.11254214430209036, 2, 0.5785569790964261, 3,
0.045830523713193974, 4, 0.2454484153742414, 5, 0.040908069229040236, 6, 0.012587098224320073, 7, 0.007956844234659474, 8, 0.003888514272870308, 9, 0.0,
10, 0.0, 11, 0.0006293549112160036, 12, 0.14661721735221397, 13, 0.02775904697684873, 14, 0.0, 15, 8.990784445942908e-05, 16, 0.0007866936390200045, 17,
0.0018114632501685717, 18, 0.15014610024724656, 19, 0.04236907170150596, 28, 0.0, 30, 0.0086710670990133497, 31, 0.0, 32,
0.0222521915037080697, 33, 0.0005169701056417172, 34, 0.004495392222971455, 35, 0.8626432906271072, 36]
0092| exportsroumainie=[0.119629057187017, 20, 5.6155074703760945, 21, 0.0008758371973209686, 22, 0.37980422462648117, 23, 3.6187017001545594, 24,
0.9254507985574446, 25, 12.294178258629572, 26, 3.8095311695002576, 27, 1.6410097887686759, 0, 1.358268935394127, 1, 0.767284904688305, 2,
2.2951056156620298, 3, 7.916486347423689, 4, 0.2605358062854199, 5, 0.9037609479649665, 6, 2.3135497166409067, 7, 0.1842864502833591, 8,
0.9379185986604843, 9, 1.735239567233385, 10, 0.43086038124678, 11, 1.7996908809891807, 12, 0.33369397217928903, 13, 0.0028851107676455437, 14, 0.0, 15,
0.23142709943328182, 16, 0.0068006182380216385, 17, 0.19273570324574962, 18, 1.09458380523713194, 29, 0.0, 30, 0.006710670990133497, 31, 0.0, 32,
0.0, 30, 0.0, 31, 0.00844925296239052, 32, 0.060226687274601, 33, 2.2080886141164346, 34, 0.0, 35, 0.0, 36]
0093| exportskazakhstan=[0.15421225382932166, 20, 0.0, 21, 0.0, 22, 0.0, 23, 0.0, 24, 0.22401531728665208, 25, 0.06602844638949672, 26, 1.7224835886214442,
27, 0.0, 0.0, 0.0, 1, 0.03025164137185558, 2, 0.2799781181619256, 3, 2.938074398249453, 4, 0.05300875273522976, 5, 0.38938730853391684, 6, 0.0, 7,
0.000765864332609387, 8, 0.0, 9, 0.0, 10, 0.36504376367614877, 11, 0.0, 12, 0.013074398249452954, 13, 0.0, 14, 0.12784463894967177, 15, 0.0, 16,
0.013457330415754924, 17, 0.0, 18, 0.006181619256017505, 19, 0.225057404595186, 28, 0.0004923413566739606, 29, 0.0, 30, 0.0, 31, 0.0, 32,
1.886159737417943, 33, 0.2400437636761488, 34, 3.09753829321663, 35, 0.0, 36]
0094| exportschina=[0.11103806228373703, 20, 1.9170795847750866, 21, 0.0, 22, 0.5084152249134948, 23, 0.39909342560553634, 24, 10.573314878892733, 25,
5.25325294117647, 26, 0.02085121107266436, 0, 0.06303114186851211, 1, 0.1391833910034602, 2, 0.32438754325259517, 3, 0.2810242214532872, 4,
0.01330795847750865, 5, 0.05903114186851211, 6, 0.30187543252595156, 7, 0.0002837370242214533, 8, 0.0, 9, 0.002304498269896194, 10, 0.01613840830449827,
11, 0.0031903114186851212, 12, 0.01536332179930796, 13, 0.08107266435986159, 14, 0.0008512110726643598, 15, 0.0, 16, 0.07142560553633218, 17,
0.0752318339100346, 18, 0.393038062283737, 19, 0.021833910034602076, 28, 0.01469204152249135, 29, 0.0, 30, 0.01843598615916955, 31, 0.0980553633217993,
32, 0.0, 33, 0.0628096858131487, 34, 0.04119031141868512, 35, 0.6239861591695501, 36]
0095| exportsukraine=[1.398080956640022, 20, 9.371605526441163, 21, 0.18997141409590454, 22, 0.03277751310147689, 23, 3.2256312529776086, 24,
7.072439256788947, 25, 25.496236303001428, 26, 3.9179371129109097, 27, 0.08151500714626013, 0, 12.631729394949977, 1, 0.013411481657933, 2,
1.6448785135778943, 3, 0.731038589804669, 4, 0.7002382086707957, 5, 2.1504287756074323, 6, 0.7276560266793711, 7, 0.0216769899042401142, 8,
0.0632682229633157, 9, 0.0542596474511672, 10, 0.0274892806609814197, 11, 0.1316579323487375, 12, 0.07532158170557408, 13, 0.06476893758932825, 14,
0.0033342258218199, 15, 0.0013577894235350166, 16, 0.0020009528346831827, 17, 0.7855645545497856, 18, 11.760576464983325, 19, 0.012148642210576465, 28,
0.0034778465936160076, 29, 0.0, 30, 0.09299666507860886, 31, 0.0031205335874225824, 32, 0.49187708432586946, 33, 0.0, 34, 12.348451643639828, 35,
12.067651262505954, 36]
0096| exportschine=[0.00016941852117731515, 20, 0.0005190236898779613, 21, 0.183776022972003e-05, 22, 0.871500358937545e-05, 23, 9.834888729361091e-05, 24,
3.589375448671931e-06, 25, 0.0001830581478822685, 26, 0.005466618808327351, 27, 0.0010244077530509692, 0, 0.0002390524048815506, 1, 0.001256281407035176,
2, 0.0006898779612347451, 3, 0.00022613065326633166, 4, 6.460875807609476e-06, 5, 0.00013854989231873655, 6, 2.871500358937545e-05, 7,
5.0251256281407036e-05, 8, 1.7946877243359657e-05, 9, 7.178758897343862e-06, 10, 9.117013639626705e-05, 11, 4.3072505384063175e-05,
12, 2.4467753050969132e-05, 13, 1.8664752333094043e-05, 14, 2.871500358937545e-06, 15, 0.00029576453697056714, 16, 2.08183776022972e-05, 17,
0.399138549892319e-05, 18, 0.0016611629576453698, 19, 0.003267767408470926, 28, 0.007354630294328787, 29, 0.0, 30, 0.0, 31, 2.153625629203159e-06, 32,
0.0021586503948312994, 33, 0.0, 34, 0.0, 35, 7.394113424264179e-05, 36]
0097| exports=[0.017706577974870658, 20, 0.5929563932002956, 21, 0.01785957132298596, 22, 0.00020620842572062085, 23, 0.0012483370288248338, 24,
0.007502586844050259, 25, 0.03042867701404287, 26, 0.006866962305986696, 27, 0.04589578713968958, 0, 0.0034789356984478935, 1, 0.025127864005912787, 2,
0.1392239467849224, 3, 0.03618255728011826, 4, 0.0066585365853658535, 5, 0.020244641537324463, 6, 0.002588322246858832, 7, 0.0013325942350332594, 8,
0.005578713968957871, 9, 0.00014412416851441242, 10, 0.0058307464892830745, 11, 0.006283074648928307, 12, 0.005977827505997783, 13, 0.0007730968218773097,
14, 0.0004449371766444937, 15, 4.7302291204730226e-05, 16, 0.0002992609016999261, 17, 0.0012985957132298595, 18, 0.03643385070214338, 19,
0.03851810790835181, 28, 0.12163266814486327, 29, 0.0001500369549150037, 30, 0.00011677753141167775, 31, 0.0005018477457501847, 32, 0.019248337028824832,
33, 0.0008032520325203253, 34, 5.764966740576497e-05, 35, 0.0, 36]
0098|
0099|
100| exports=[exportsfrance, exportsespagne, exportsallemagne, exportsuk, exportsitalie, exportspologne, exportsbelgique, exportsgrece,
exportsreptcheque, exportsportugal, exportshongrie, exportssuede, exportsautriche, exportssuisse, exportsdanemark, exportsfinlande, exportslovaquie,
exportsnorvege, exportsirlande, exportspaysbas, exportsalgerie, exportsarabiasaoudite, exportsisrael, exportsmalte, exportsmaroc, exportstunisie,
exportsegypte, exportsturquie, exportscanada, exportsusa, exportsargentine, exportsroumanie, exportskazakhstan, exportsrusse, exportsukraine,
exportschine, exportinde]
101|
102| import numpy as np
103| import matplotlib.pyplot as plt
104| import random as rd
105| from math import floor
106| from math import ceil
107|
108| ##liste avec les numéros des pays concernés
109| def listenum(H):
110|     listenum=[0]*(len(H))
111|     for i in range (len(H)):
112|         listenum[i]=H[i][1]
113|     return (listenum)
114|
115| ##creer seuildepart
116| def seuildep(T, sc, sd, S):
117|     for i in range (len(T)):
118|         if S[i]==0:
119|             if T[i]>=sc:
120|                 S[i]=1
121|             else:
122|                 S[i]=0
123|         else:
124|             if T[i]>sd:
125|                 S[i]=1
126|             else:
127|                 S[i]=0
128|     return S
129|
130| ##taux
131| def taux(C, L):
132|     T=[0]*(len(C))
133|     for i in range (len(C)):
134|         if (L[i][0]+L[i][1]+L[i][2])==0:
135|             T[i]=0
136|         else:
137|             T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
138|     return (T)
139|
140| def tauxLocal(C, L, i):
141|     T=0
142|     if (L[i][0]+L[i][1]+L[i][2])==0:
143|         T=0
144|     else:
145|         T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
146|     return (T)
147|
148| ##creer tableau
149| def creertableau(n, a):

```

```

150 N=[0]*n
151 for i in range (n):
152     N[i]=[0]*a
153     return N
154
155 ##SIRevol
156 #Lt=[St, It, Rt, Dt] #D les dead
157 #d le taux de létalité =! taux de mortalité
158 def SIRevol(Lt, beta, l, k): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
159     #modèle d'évolution de t à t+1
160     gamma=0.3
161     d=0.048
162     N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
163     if N==0:
164         return ([0, 0, 0, 0])
165     else:
166         L1=[0,0,0,0]
167         L1[0]= (-beta[i][k]*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
168         L1[1]= beta[i][k]*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
169         L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
170         L1[3]= Lt[3]+d*Lt[1]
171         Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
172         return (Lt) #on renvoie la liste au temps t+1
173
174
175 ##renvoyer une case
176 def case(Lt,n):
177     return (Lt[n])
178
179 ##creerLt pour le début seulement
180 def creerLt(C, H, i): #crée un Lt pour le pays en position i
181     Lt=[0, 0, 0, 0]
182     Lt[0]=H[i][0]-C[i]
183     Lt[1]=C[i]
184     return Lt
185
186 ##creerL au depart
187 def creerL(C, H): #crée la liste des Lt-> cad on a L[0][0] qui donne le nombre de sains dans le pays 0
188     #L[0][1] le nombre de contaminés du pays 0
189     #L[0][2] le nombre de recovered du pays 1
190     L=createtableau(len(C), len(C))
191     for i in range (len(C)):
192         L[i]=creerLt(C,H,i)
193     return L
194
195 ##verifs
196 def verifs(S):
197     n=len(S)
198     res=0
199     for i in range (n):
200         if S[i]==1:
201             res=res+1
202     if res==n:
203         return False
204
205 ##creerV V=voyageurs en prenant en compte les stratégies de confinement
206
207 def creerV(L, C, S, Pop):
208     paspossible=[]
209     for i in range (len(C)):
210         if Pop[i]==0:
211             paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
212     V=[0]*(len(C))
213     for i in range (len(C)):
214         V[i]=[0]*2
215     if verifs(S)==False:
216         for i in range (len(C)):
217             V[i][0]=0
218     else :
219         for i in range (len(C)):
220             if S[i]==0:
221                 V[i][1]= rd.randint(0, len(C)-1)
222                 while S[V[i][1]]==1 :
223                     V[i][1]=rd.randint(0, len(C)-1)
224                     while V[i][1] in paspossible :
225                         V[i][1]=rd.randint(0, len(C)-1)
226                 V[i][0]=rd.uniform(0, 0.0002)*(L[i][0]+L[i][1]+L[i][2])#on ne veut pas que plus de la moitié de la population parte sinon pas
227             else:
228                 V[i][0]=0
229     return (V)
230
231 ##creerD
232 def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
233     D=[0]*(len(C))
234     T=taux(C, L)
235     for i in range (len(C)):
236         D[i]=[0]*2
237     for i in range (len(C)):
238         D[i][0]=(V[i][0]*T[i])
239         D[i][1]=V[i][1]
240     return (D)
241
242 ##creerE
243 def creerE(H, S, E, exports, Pop):
244     paspossible=[]
245     for i in range (len(H)):
246         if Pop[i]==0:
247             paspossible.append(i)
248     if verifs(S)==False:
249         for i in range (len(E)):
250             for k in range (len(E[i])):
251                 E[i][k]=0
252     else:
253         for i in range (len(E)):
254             if S[i]==0:
255                 for k in range (len(E[i])):
256                     if k%2==0:
257                         E[i][k]=exports[i][k]*Pop[i]/365
258                     else:
259                         if k in paspossible:
260                             E[i][k-1]=0
261                         else:
262                             E[i][k]=exports[i][k]
263     else:
264         for k in range (len(E[i])):
265             E[i][k]=0
266     return E
267
268 ##trouver les contaminés
269

```

```

270 def recherche(D, i):
271     c=0 #res
272     for k in range (len(D)):
273         if D[k][1]==i:
274             c=c+D[k][0]
275     return c
276
277 def recherchebis(E,H,i):
278     c=0
279     for k in range (len(E)):
280         for a in range (len(E[i])):
281             if a%2==1:
282                 if E[k][a]==H[i][3]:
283                     c=c+abs(E[k][a-1])
284     return c
285
286 ##creer beta
287 def creertableaubeta(n,C):
288     beta=creertableau(len(C), n+1)
289     for i in range (len(C)):
290         beta[i][0]=0.39
291     return beta
292
293 ##creerZ
294 def creerZ(H, prop):
295     Z=[0]*len(H)
296     for i in range (len(Z)):
297         Z[i]=H[i][1]*prop
298     return Z
299
300 ##faire manger les gens
301 def manger(Pop, Z, i, L):
302     consoparhab=0.393750627
303     viv=Pop[i] # nos vivants avant de voir qui aura assez à manger
304     ts=0
305     tc=0
306     tr=0
307     survivants=0
308     capacité=0
309     nouveauxmorts=0
310     if Pop[i]!=0:
311         ts=L[i][0]/viv
312         tc=L[i][1]/viv
313         tr=L[i][2]/viv
314         capacité=Z[i]/consoparhab
315         if capacité<viv:
316             survivants=capacité
317             nouveauxmorts=viv-survivants
318             Pop[i]=survivants
319             L[i][0]=ts*Pop[i]
320             L[i][1]=tc*Pop[i]
321             L[i][2]=tr*Pop[i]
322             L[i][3]=L[i][3]+nouveauxmorts
323             Z[i]=0
324         else:
325             Z[i]=Z[i]-Pop[i]*consoparhab
326
327     return (Z[i], L[i])
328
329
330 ##expansion avec confinement
331 def expansionconfined(C, H, n, sc, sd, exports, prop):# d le taux de gens qui meurent
332     Pop=[0]*len(H)
333     for i in range (len(H)):
334         Pop[i]=H[i][0]
335     L=creerL(C, H)
336     beta=creertableaubeta(n, C)
337     gamma=0.2
338     d=0.048
339     S=[0]*(len(C))
340     T=taux(C,L)
341     evoltaux=creertableau(len(C), n)
342     Z=creerZ(H, prop)
343     E=creertableau(len(C), len(exports))
344     S=seuildep(T,sc, sd,S)
345     V=creerV(L,C,S, Pop)
346     D=creerD(H,C,V,L) #0 le tableau des gens dangereux qui se déplacent et vers quel pays ils se déplacent
347     evolcontamines=creertableau(len(C), n) #evolution des contaminés
348     evolmorts=creertableau(len(C),n)
349     evoldeZ=creertableau(len(C), n)
350     evoldeE=creertableau(len(C), n)
351     evolimports=creertableau(len(C),n)
352     evoldeH=creertableau(len(C), n)
353     listedeS=creertableau(len(C),n)
354     prodpartravailleurparjour=[]
355     for i in range (len(C)):
356         evoldeZ[i][0]=Z[i]
357         evolmorts[i][0]=0
358         evoldeH[i][0]=Pop[i]
359         evoldeE[i][0]=0
360         evoltaux[i][0]=T[i]
361         evolimports[i][0]=0
362         listedeS[i][0]=S[i]
363     for i in range (len(H)):
364         prodpartravailleurparjour.append(H[i][1]/H[i][0]/365)
365     for k in range (1, n):
366         T=taux(C,L)
367         S=seuildep(T,sc, sd ,S)
368         V=creerV(L,C,S, Pop)
369         D=creerD(H,C,V,L)
370         E=creerE(H, S, E, exports, Pop)
371         for i in range (len(C)):
372             #partie épidémiologique
373             L[i]=SIREvol(L[i], beta, i, k)
374             C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine
375             Pop[i]=Pop[i]+recherche(V,i)-V[i][0]
376             L[i]=Pop[i]-C[i]-L[i][2],C[i][2],L[i][2],L[i][3]]
377             evolcontamines[i][k]=C[i]
378             evolmorts[i][k]=L[i][3]
379             listedeS[i][k]=S[i]
380             if listedeS[i][k]==1:
381                 beta[i][k+1]=0.22
382             else:
383                 beta[i][k+1]=0.39
384             evoltaux[i][k]=tauxlocal(C,L, i)
385             #partie échanges de céréales
386             travailleurs=(L[i][0]+L[i][2])
387             Z[i]=Z[i]+prodpartravailleurparjour[i]*travailleurs #on produit de la nourriture
388             Z[i]=Z[i]+recherchebis(E,H,i)-E[i][0] #on importe et exporte
389             Z[i]=manger(Pop, Z, i, L)[0]
390             L[i]=manger(Pop, Z, i, L)[1]

```

```

391         evolmorts[i][k]=L[i][3]
392         evolcontamines[i][k]=L[i][1]
393         evolmorts[i][k]=L[i][3]
394         evoldeZ[i][k]=Z[i]
395         evoldeH[i][k]=Pop[i]
396         evoldeE[i][k]=somme3(E, i)
397         evolimports[i][k]=recherchebis(E,H,i)
398
399
400     return ([evolmorts, evolcontamines, evoldeZ, evoldeH, evoldeE, voltaux, evolimports]) #on renvoie la liste des états de chaque pays
401
402
403     ##tracer des évolutions en fonction
404     def tracercontamines(C, H, n,sc, sd):
405         evol=expansionconfined(C, H, n, sc, sd)[1]
406         abscisse=[0]*n
407         liste=listenum(H)
408         for i in range (n):
409             abscisse[i]=i
410         for k in range (len(C)):
411             plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
412             plt.legend()
413             plt.title("Nombre de contaminés en fonction du temps")
414             plt.xlabel("Temps en jours")
415             plt.ylabel("Nombre de contaminés")
416             plt.legend(fontsize=6, loc='best')
417         return(plt.show())
418
419     def tracermorts(C, H, n, sc, sd, exports, prop):
420         evol=expansionconfined(C, H, n, sc, sd, exports, prop)[0]
421         abscisse=[0]*n
422         liste=listenum(H)
423         for i in range (n):
424             abscisse[i]=i
425         for k in range (len(C)):
426             plt.plot(abscisse,evol[k], label=str(NOM_PAYS[liste[k]]))
427             plt.legend()
428             plt.title("Nombre de morts en fonction du temps")
429             plt.xlabel("Temps en jours")
430             plt.ylabel("Nombre de morts")
431             plt.legend(fontsize=6, loc='best')
432         plt.show()
433
434     def touttracer(C, H, n, sc, sd, exports, prop):
435         RES=expansionconfined(C, H, n, sc, sd, exports, prop)
436         abscisse=[0]*n
437         liste=listenum(H)
438         for i in range (n):
439             abscisse[i]=i
440         for a in range (len(RES)):
441             for k in range (len(liste)):
442                 plt.plot(abscisse, RES[a][k], label=str(NOM_PAYS[k]))
443                 plt.legend()
444                 plt.xlabel("temps en jour")
445                 plt.title(str(AFFICHERBIS[a]))
446                 plt.ylabel(str(AFFICHER[a]))
447                 plt.legend(fontsize=6, loc='best')
448             plt.show()
449
450     def tracerfrance(C, H, n, sc, sd, exports, prop):
451         RES=expansionconfined(C, H, n, sc, sd, exports, prop)
452         abscisse=[0]*n
453         liste=listenum(H)
454         for i in range (n):
455             abscisse[i]=i
456         for a in range (len(RES)):
457             for k in range (1):
458                 plt.plot(abscisse, RES[a][k], label=str(NOM_PAYS[k]))
459                 plt.legend()
460                 plt.xlabel("temps en jour")
461                 plt.title(str(AFFICHERBIS[a]))
462                 plt.ylabel(str(AFFICHER[a]))
463                 plt.legend(fontsize=6, loc='best')
464             plt.show()
465
466     ##somme
467     def somme3(E, i):
468         s=0
469         for k in range (len(E)):
470             s=s+E[i][k]
471         return (s)
472
473     ##application A FAIRE
474     #touttracer([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
475     0.003, 0.0006, exports, 30)
476     #touttracer([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
477     0.003, 0.0006, exports, 1)
478     #touttracer([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
479     0.003, 0.0006, exports, 2)
480     #tracerfrance([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
481     0.003, 0.0006, exports, 1)

```

calcul stocks.py

```
0001 ##data
0002
0003 NOM PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
0004 "Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie", "Norvège", "Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israël", "Malte", "Maroc",
0005 "Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]
0006
0007 AFFICHER = ["évolution du nombre total de morts", "évolution du nombre de contaminés", "évolution des stocks en dollar usd", "évolution du nombre de
0008 vivants", " évolution des exports", "évolution du taux de contaminés", " évolution des imports"]
0009
0010 #données relatives aux pays (nombre hab, production 2017, consommation par habitant, numéro du pays, quantité exportée par an)
0011 #production en dollars usd DE CEREALES par an
0012
0013 #AFNMO où on a de la donnée:
0014 algerie=[42230000,1217055450, 0.393750627, 20, 0]
0015 arabiesaoudite=[34813900,591101519, 0.393750627, 21, 0]
0016 israel=[8655540,125803727, 0.393750627, 22, 0]
0017 malte=[514564,12295953, 0.393750627, 23, 0]
0018 maroc=[34660000,2350715863, 0.393750627, 24, 0]
0019 tunisie=[11180000,354334516, 0.393750627, 25, 0]
0020 egypte=[102334000,4303123738, 0.393750627, 26,0]
0021 turquie=[84339100,8556108904, 0.393750627, 27,185858000]
0022 AFNMO=[algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie]
0023
0024 #europe
0025 france=[66352469,12366941872, 0.393750627, 0, 5602340000]
0026 espagne=[46439864,3547151265, 0.393750627, 1, 0]
0027 allemagne=[81174000,8131825487, 0.393750627, 2, 2512970000]
0028 UK=[64767115,4359410688, 0.393750627, 3, 438740000]
0029 italie=[60795612,4301291729, 0.393750627, 4, 0]
0030 pologne=[38005614,5429724123, 0.393750627, 5, 978450000]
0031 belgique=[11258434,460108034, 0.393750627, 6, 0]
0032 grece=[10812467,877721695, 0.393750627, 7, 0]
0033 reptcheque=[10538275,1389583863, 0.393750627, 8, 0]
0034 portugal=[10374822,253498428, 0.393750627, 9, 0]
0035 hongrie=[9849000,2311067527, 0.393750627, 10,1747190000]
0036 suedes=[9747355,983369105, 0.393750627, 11, 0]
0037 autriches=[8584926,708800003, 0.393750627, 12, 0]
0038 suisse=[8236573,400991999, 0.393750627, 13, 0]
0039 danemark=[5659715,1776259292, 0.393750627, 14, 0]
0040 finlande=[5471553,565659239, 0.393750627, 15, 0]
0041 slovaquie=[5421349,59431977, 0.393750627, 16, 0]
0042 norvege=[5165802,459500938, 0.393750627, 17, 0]
0043 irlande=[4625885,344572386, 0.393750627, 18, 0]
0044 paysbas=[16900726,252251731, 0.393750627, 19, 0]
0045 europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suedes, autriche, suisse, danemark, finlande,
0046 slovaquie, norvege, irlande, paysbas]
0047
0048 #autres pays importants
0049 canada=[37590000,9711442097, 0.393750627, 28, 6305310000]
0050 USA=[328200000,68607473570, 0.393750627, 29,18740200000]
0051 argentine=[44490000,8846326657, 0.393750627, 30, 7511940000]
0052 roumanie=[19410000,5391922117, 0.393750627, 31, 2184270000]
0053 kazakhstan=[18280000,2763295937, 0.393750627, 32, 852409000]
0054 russie=[144500000,19082405580, 0.393750627, 33, 8142170000]
0055 ukraine=[41980000,8391279873, 0.393750627, 34, 7947470000]
0056 chine=[1393000000,2.62353E+11, 0.393750627, 35, 721030000]
0057 inde=[1353000000,1.12689E+11, 0.393750627, 36, 7554560000]
0058 autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]
0059
0060 monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suedes, autriche, suisse, danemark, finlande,
0061 slovaquie, norvege, irlande, paysbas,algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie,canada, USA, argentine, roumanie, kazakhstan,
0062 russie, ukraine, chine, inde]
0063
0064 ##exports
0065 exportsfrance=[10.46140423199625, 20, 2.59628620601895, 21, 0.22394042337746317, 22, 0.024384925299463987, 23,2.461897838345699, 24,
0066 1.1658194663411847, 25, 0.7627146474383644, 26, 0.1740402455860384, 27, 0.0, 0, 11.60814377532809, 1, 6.408126274848944, 2, 2.909145701872827, 3,
0067 5.703404947900281, 4, 0.6288688368175116, 5, 13.64848985498942, 6, 0.5503789165705348, 7, 0.207588356659343, 8, 3.0381989251974932, 9,
0068 0.4630724442208316, 10, 0.11401233614984244, 11, 0.34679945368724713, 12, 0.8546931388491361, 13, 0.30463071389250035, 14, 0.011785444860433152, 15,
0069 0.05442148656141191, 16, 0.017798885524516053, 17, 0.7045103325393965, 18, 9.489925687618346, 19, 0.002637430115825833, 20, 0.05032216660995689, 29,
0070 0.0085274860231651666, 30, 0.40878659692377084, 31, 0.008319208136776342, 32, 0.3686524460755183, 33, 0.3207567151721212, 34, 0.6010929299405573, 35,
0071 0.03018727155428082, 36] #export par personne
0072
0073 exportsespagne=[0]*74
0074 exportsallemagne=[0.5889693744302363, 20, 5.938859733412176, 21, 0.18519476679725036, 22, 0.0020942666371005494, 23, 0.5850025870352576, 24, 0.0, 25,
0075 0.0, 26, 0.2414812624732057, 27, 1.6679355458644394, 0.952509424199867, 1, 0.0, 2, 0.752864217606228, 3, 1.1632296053871092, 4, 0.9158228064040703, 5,
0076 3.1513292433537834, 6, 0.09580653903959396, 7, 0.09580653903959396, 8, 0.7167565969399069, 9, 0.057629290166802176, 10, 0.12348781629585828, 11,
0077 0.5125655988226033, 12, 0.6984009658264961, 13, 0.4261832606499618, 14, 0.016212087614260722, 15, 0.028346514893931554, 16, 0.25333234779609237, 17,
0078 0.03783231083844581, 18, 5.9272118074753, 19, 0.0035232956365338656, 20, 0.1677877152782911, 29, 4.927686204942469e-05, 30, 0.01740705151895927, 31,
0079 0.0002710227412718358, 32, 0.043437553896567865, 33, 0.037684480252297536, 34, 0.009313326927341267, 35, 0.0, 36]
0080
0081 exportsUK=[0.06600571910606177, 20, 0.008507403795892406, 21, 0.35135114479006824, 22, 0.014096660010253661, 23, 0.0, 24, 0.0702825809054487, 25,
0082 1.5439934293815618e-05, 26, 0.0035511848875775924, 27, 0.2991332870083838, 0.10207494960984445, 1, 0.4161216691526248, 2, 0.0, 3, 0.16358610384297648,
0083 4, 0.0142646992878485633, 5, 0.8785785811209902, 6, 0.06625275805476283, 7, 0.011965949077707105, 8, 0.46951296194066392, 9, 0.0022542304068970806, 10,
0084 0.0406048697552767, 11, 0.010900593611433827, 12, 0.005681895820124148, 13, 0.02011823438484175, 14, 0.006052454243175723, 15, 0.0007256769118093341, 16,
0085 0.02118358985111503, 17, 1.469295027268082, 18, 1.0202554182010424, 19, 0.101380140890172427, 28, 0.19211910241794775, 29, 0.00010807954005670933, 30,
0086 0.0002933587515824968, 31, 0.0, 32, 0.0, 33, 0.0, 34, 0.005975254571706645, 35,7.71996714690781e-05, 36]
0087
0088 exportsitalie=[0]*74
0089 exportspologne=[0.15589802075030285,20,2.5434926534800884, 21, 0.0, 22, 0.05988588948990536, 23, 5.2623804472676064e-05, 24, 0.0,
0090 25,1.4206585374465994, 26, 0.2910622625383713, 27, 0.10569491128336987, 0.1547192475301149, 1, 9.243608062745677, 2, 0.9485966994244587, 3,
0091 0.334161158401493, 4, 0.0, 5, 0.06625336983109917, 6, 0.319531740758008904, 7, 0.5947805500524213, 8, 0.3518953805087848, 9, 0.12716542350822171, 10,
0092 0.18860371523007102, 11, 0.06609549841768113, 12, 0.026048783213974653, 13, 0.4116760223897448, 14, 0.08803962488278705, 15, 0.09448604093068987,
0093 16,0.3983095760536851, 17, 0.037731267806900874, 18, 1.0172444523590647, 19, 0.0030521806594152115, 28, 0.007577827844065353, 29, 0.0, 30,
0094 0.05717576355956254, 31, 0.0, 32, 0.028864156753262822, 33, 0.007946194475374086,34, 0.0, 35, 0.0, 36]
0095
0096 exportsbelgique=[0]*74
0097 exportsgrece=[0]*74
0098 exportsreptcheque=[0]*74
0099 exportsportugal=[0]*74
0100
0101 exportshongrie=[0.030663011473246016, 20, 2.659660879277084, 21, 0.032389075032998274, 22, 0.07533759772565743, 23, 1.427251497613971,24, 0.0, 25,
0102 0.0, 26, 3.052695705147731, 0.6209767489085186, 1, 11.790537110366534, 2, 0.08437404812671337, 3, 57.27058584627881, 4, 4.7326632145395475, 5,
0103 0.791247842450504, 6, 3.1041730124885776, 7, 1.598334856330592, 8, 0.02832774901005178, 9, 0.0, 10, 0.0, 11,1.6961315869631435, 12, 1.7761194029850746,
0104 13, 0.01015331505736623, 14, 0.0018275967103259215,15,5.084069448674993, 16, 0.0031475276677835314, 17, 0.013808508478018074, 18,2.8561275256371204, 19,
0105 0.000507665728683116, 28, 0.007107320540156361, 29, 0.0, 30, 28.19768504416692, 31,0.0445305310183775, 32,8.774291806274748, 33,5.544623819677125, 34,
0106 0.005787389582698751, 35, 0.0, 36]
0107
0108 exportssuede=[0]*74
0109 exportsautriche=[0]*74
0110 exportssuisse=[0]*74
0111 exportsdanemark=[0]*74
0112 exportsfinlande=[0]*74
0113 exportsslovaquie=[0]*74
0114 exportsnorvege=[0]*74
0115 exportsirlande=[0]*74
0116 exportspaysbas=[0]*74
0117 exportsalgerie=[0]*74
0118 exportsarabiesaoudite=[0]*74
0119 exportsisrael=[0]*74
0120 exportsmalte=[0]*74
0121 exportsmaroc=[0]*74
0122 exportstunisie=[0]*74
0123 exportsegypte=[0]*74
```

0088] exportsturquie=[8.29982771929034e-05, 20, 0.01815290891176216, 21, 0.26981554225738713, 22, 0.010825346725302974, 23, 0.0, 24, 0.006888857007010983, 25, 0.0, 26, 0.030294371175409746, 0, 0.013493148492217727, 1, 0.051399647375890894, 2, 0.13569032631365524, 3, 0.06854472006459637, 4, 0.0, 5, 0.034147862616509883, 6, 0.010517067410015046, 7, 0.0006995569077687573, 8, 0.0, 9, 0.014595839889209157, 10, 0.002525519006012632, 11, 0.005821736300245082, 12, 0.00815754495838822, 13, 0.0002134241413531802, 14, 0.0004031344892226737, 15, 0.0, 16, 0.002715229353882126, 17, 4.742758696737338e-05, 18, 0.0194215968631394, 19, 0.04470050071674941, 20, 0.6195702823482822, 29, 0.0, 30, 0.009604086360893108, 31, 0.006758431142850707, 32, 0.0038653483378409304, 33, 0.017334783036574968, 34, 1.1856896741843345e-05, 35, 0.0007825551849616607, 36]

0089] exportcanada=[8.608273476988561, 20, 0.911066730779463, 21, 0.006517690875232775, 22, 0.014285714285714285, 23, 5.052274541101356, 24, 0.528198989928438, 25, 0.04732641660015962, 26, 0.9531258313381218, 27, 0.281511040170258, 1, 0.1488427773343974, 1, 0.2590316573556797, 2, 3.08677398510242, 3, 5.02487363660548, 4, 0.0, 5, 1.3740888534184623, 6, 0.012396914072891726, 7, 0.0, 8, 0.7461558925246076, 9, 0.0, 10, 0.00378558127161479, 11, 0.005826017557861133, 12, 0.4795424314977388, 13, 0.0027400904495876563, 14, 2.6602819898909284e-05, 15, 0.0, 16, 0.0003192338387089114, 17, 1.7341846235700984, 18, 0.6632349028997073, 19, 0.0, 20, 28.00053205639798, 29, 0.00023942537909018357, 30, 0.0, 31, 0.0, 32, 0.0022346368715083797, 33, 0.0, 34, 11.695956371375365, 35, 0.0031391327480712957, 36]

0090] exportsUSA=[0.5279128500134005, 20, 1.1996739792009263, 21, 0.20313223644119438, 22, 0.0007099329677026203, 23, 0.5454478976234004, 24, 0.009926873857404022, 25, 0.197904028031687996, 26, 0.10457647775746497, 27, 0.0769926873857404, 0, 0.1509354052407069, 1, 0.033769043266301035, 2, 0.185767824497257776, 3, 0.4285580956063376, 4, 0.004722790042656917, 5, 0.018598415600243754, 6, 0.004567336989640463, 7, 6.093845216331595e-05, 8, 0.060856185252894573, 9, 0.0015234613040828763, 10, 0.01263558056063376, 11, 0.006380255941499086, 12, 0.008488726386349787, 13, 0.00551492992078001, 14, 0.0017032297379646556, 15, 0.0, 16, 0.006733698964046313, 17, 0.059713589, 18, 0.12470444050700792, 19, 1.5059689213893968, 28, 0.0, 29, 0.005730652041438147, 30, 0.0005971968312004875, 31, 0.0006703229737964656, 32, 0.010923217550274223, 33, 0.01064594759293114, 34, 4.107876294942108, 35, 0.015426569165143206, 36]

0091] exportsargentine=[19.8215329287480034, 20, 8.547336480107889, 21, 0.022445484153742415, 22, 0.0, 23, 5.671813890761969, 24, 1.1792312879298719, 25, 11.57285996945382, 26, 0.05945156214879748, 27, 0.03387278040008991, 0, 0.42063385030343897, 1, 0.11254214430209036, 2, 0.5785569790964261, 3, 0.045830523713193974, 4, 0.24544845153742414, 5, 0.04090806229040236, 6, 0.012587099224320073, 7, 0.007956844234659474, 8, 0.003888514272870308, 9, 0.0, 10, 0.0, 11, 0.0006293549112160036, 12, 0.14661721735221397, 13, 0.02775904697684873, 14, 0.0, 15, 8.990784445942908e-05, 16, 0.0007866936390200045, 17, 0.0018114632501685772, 18, 0.15014610024724656, 19, 0.04236907170150596, 28, 0.09458306327313194, 29, 0.0, 30, 0.008676106990334907, 31, 0.0, 32, 0.02225191503708697, 33, 0.0005169701056417172, 34, 0.004495392222971455, 35, 0.8626432906271072, 36]

0092] exportsromania=[0.119629057187017, 20, 5.6155074703760945, 21, 0.0008758371973209686, 22, 0.37980422462648117, 23, 3.6187017001545594, 24, 0.92545079855744046, 25, 12.2947178258629572, 26, 3.8095311695002576, 27, 1.6410097887686759, 0, 1.3582689335394127, 1, 0.767284904688305, 2, 2.9510561566202298, 3, 7.916486347243689, 4, 0.2605358062854199, 5, 0.9037609479649665, 6, 2.3135497166409067, 7, 0.1842864502833591, 8, 0.9379185986604843, 9, 1.735239567233385, 10, 0.43086038124678, 11, 1.7996908809891807, 12, 0.33369397217928903, 13, 0.0028851107676455437, 14, 0.0, 15, 0.23142709943328182, 16, 0.0068006182380216385, 17, 0.19273570324574962, 18, 1.3801648634724368, 19, 0.10664605873261206, 28, 0.42416280267903145, 29, 0.0, 30, 0.0, 31, 0.0084492596239052, 32, 2.060226687274601, 33, 2.2080886141164346, 34, 0.0, 35, 0.0, 36]

0093] exportskazakhstan=[0.15421225382932166, 20, 0.0, 21, 0.0, 22, 0.0, 23, 0.0, 24, 0.22401531728665208, 25, 0.06602844638949672, 26, 1.7224835886214442, 27, 0.0, 0.0, 0.0, 1, 0.03025164113785558, 2, 0.2799781181619256, 3, 0.938074398249453, 4, 0.05300875273522976, 5, 0.38938730853391684, 6, 0.0, 7, 0.0007568643326039387, 8, 0.0, 9, 0.0, 10, 0.36504376367614877, 11, 0.0, 12, 0.013074398249452954, 13, 0.0, 14, 0.12784463894967177, 15, 0.0, 16, 0.013457330415754924, 17, 0.0, 18, 0.006181619256017505, 19, 0.225054704595186, 28, 0.0004923413566739606, 29, 0.0, 30, 0.0, 31, 0.0, 32, 1.886159737417493, 33, 0.2400437636761488, 34, 3.09753829321663, 35, 0.0, 36]

0094] exportsrussie=[0.11103806228373703, 20, 1.9170795847750866, 21, 0.0, 22, 0.5084152249134948, 23, 0.39909342560553634, 24, 10.573314878892733, 25, 5.25235294117647, 26, 0.02085121107266436, 0, 0.06303114186851211, 1, 0.1391833910034602, 2, 0.32438754325259517, 3, 0.2810242214532872, 4, 0.013307595847750865, 5, 0.05903114186851211, 6, 0.30187543252595156, 7, 0.0002837370242214533, 8, 0.0, 9, 0.0023044498269896194, 10, 0.01613840830449827, 11, 0.0031903114186851212, 12, 0.01536332179930796, 13, 0.08107266435986159, 14, 0.0008512110726643598, 15, 0.0, 16, 0.07142560553633218, 17, 0.0752318339100346, 18, 0.393038062283737, 19, 0.021833910034602076, 28, 0.01469204152249135, 29, 0.0, 30, 0.01843598615916955, 31, 0.0980553633217993, 32, 0.0, 33, 0.06280968858131487, 34, 0.04119031141868512, 35, 0.6239861591695501, 36]

0095] exportsukraine=[1.398808956640622, 20, 9.371605526441163, 21, 0.18997141495950454, 22, 0.03277751310147689, 23, 3.2256312529776086, 24, 7.072439256788947, 25, 25.496236303001428, 26, 3.9179371129109097, 27, 0.08151500714626013, 0, 12.631729394949977, 1, 1.0134111481657933, 2, 1.6448785135778943, 3, 8.731038589804669, 4, 0.7002382006707957, 5, 2.1504270756074323, 6, 0.727650266793711, 7, 0.0216769899042401142, 8, 3.0632682229633157, 9, 0.05452596474511672, 10, 0.027489280609814197, 11, 0.1316579323487375, 12, 0.07532158170557408, 13, 0.06476893758932825, 14, 0.00358742258218199, 15, 0.0013577894235350166, 16, 0.0020009528346831827, 17, 0.7855645545497856, 18, 11.760576464983325, 19, 0.012148642210576465, 28, 0.0034778465936160076, 29, 0.0, 30, 0.0929966650786086, 31, 0.0031205335874225824, 32, 0.49187708432586946, 33, 0.0, 34, 12.348451643639828, 35, 12.067651262505954, 36]

0096] exportschine=[0.00016941852117731515, 20, 0.0005190236898779613, 21, 8.183776022972003e-05, 22, 2.871500358937545e-05, 23, 9.834888729361091e-05, 24, 3.589375448671931e-06, 25, 0.0001830581478822685, 26, 0.005466618808327351, 27, 0.0010244077530509692, 0, 0.0002390524048815506, 1, 0.001256281407035176, 2, 0.0006898779612347451, 3, 0.00022613065326633166, 4, 6.460875807609476e-06, 5, 0.00013854989231873655, 6, 2.871500358937545e-05, 7, 5.0251256281407036e-05, 8, 1.7944877243359657e-05, 9, 7.178758897343862e-07, 10, 9.117013639626705e-05, 11, 4.3072505384063175e-05, 12, 2.4407753050969132e-05, 13, 1.86647523330904043e-05, 14, 2.871500358937545e-06, 15, 0.00029576453697056714, 16, 2.08183776022972e-05, 17, 8.399138549892319e-05, 18, 0.0016611629576453698, 19, 0.003267767408470926, 28, 0.007354630294328787, 29, 0.0, 30, 0.0, 31, 2.153625269203159e-06, 32, 0.0021586503948312994, 33, 0.0, 34, 0.0, 35, 7.394113424264179e-05, 36]

0097] exportsromania=[0.017706577974870658, 20, 0.5929563932002956, 21, 0.01785957132298596, 22, 0.00020620842572062085, 23, 0.0012483370288248338, 24, 0.007502586844050259, 25, 0.03042867701404287, 26, 0.006686962305986696, 27, 0.04589578713968958, 0, 0.0034789356984478935, 1, 0.025127864005912787, 2, 0.1392239467849224, 3, 0.03618255728011826, 4, 0.0066585365853658535, 5, 0.020244641537324463, 6, 0.002588322246858832, 7, 0.0013325942350332594, 8, 0.005578713968957871, 9, 0.00014412416851441242, 10, 0.0058307464892830745, 11, 0.006283074648928307, 12, 0.005977827050997783, 13, 0.0007730968218773097, 14, 0.0004449371766444937, 15, 4.7302291204730226e-05, 16, 0.0002992609016999261, 17, 0.0012985957132298595, 18, 0.03643385070214338, 19, 0.03851810790835181, 28, 0.12163266814486327, 29, 0.0001500369549150037, 30, 0.0001167753141167775, 31, 0.0005018474757501847, 32, 0.019248337028824832, 33, 0.008032520325203253, 34, 5.764966740576497e-05, 35, 0.0, 36]

0098]

0099]

100] exports=[exportsfrance, exportsespagne, exportsallemagne, exportsUK, exportsitalie, exportspologne, exportsbelgique, exportsgrece, exportsreptcheque, exportsportugal, exportshongrie, exportssuede, exportsautriche, exportsdanemark, exportsfinlande, exportslovaquie, exportsnorvege, exportsirlande, exportspaysbas, exportsalgerie, exportsarabiasaoudite, exportsisrael, exportsmalte, exportsmaroc, exportstunisie, exportsegypte, exportsrussie, exportscanada, exportsUSA, exportsargentine, exportsroumanie, exportskazakhstan, exportsrussie, exportsukraine, exportschine, exportinde]

101]

102] import numpy as np

103] import matplotlib.pyplot as plt

104] import random as rd

105] from math import floor

106] from math import ceil

107]

108] def creerpnp(Hp, H):

109] for i in range (len(Hp)):

110] for k in range (len(Hp[i])):

111] if k%2==0:

112] Pnp[i][k]=Hp[i][k]/H[i][0]

113]

114] return Pnp

115]

116]

117] ##liste avec les numéros des pays concernés

118] def listenum(H):

119] listenum=[0]*(len(H))

120] for i in range (len(H)):

121] listenum[i]=H[i][1]

122] return (listenum)

123]

124] ##creer seuildepart

125] def seuildep(T, sc, sd, S):

126] for i in range (len(T)):

127] if S[i]==0:

128] S[i]=1

129] else:

130] S[i]=0

131]

132] else:

133] if T[i]>sd:

134] S[i]=1

135] else:

136] S[i]=0

137] return S

138]

139] ##taux

140] def taux(C, L):

141] T=[0]*(len(C))

142] for i in range (len(C)):

143] if (L[i][0]+L[i][1]+L[i][2])==0:

144] T[i]=0

145] else:

146] T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])

147] return (T)

148]

149] def tauxlocal(C, L, i):

150] T=0

151] if (L[i][0]+L[i][1]+L[i][2])==0:

152] T=0

```

152     else:
153         T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
154     return (T)
155
156 ##creertableau
157 def creertableau(n, a):
158     N=[0]*n
159     for i in range (n):
160         N[i]=[0]*a
161     return N
162
163 ##SIRevol
164 #Lt=[St, It, Rt, Dt] #D les dead
165 #d le taux de létalité != taux de mortalité
166 def SIRevol(Lt, beta, i, k): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
167     #modèle d'évolution de t à t+1
168     gamma=0.3
169     d=0.048
170     N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
171     if N==0:
172         return ([0, 0, 0, 0])
173     else:
174         L1=[0,0,0,0]
175         L1[0]= (-beta[i][k]*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
176         L1[1]= beta[i][k]*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
177         L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
178         L1[3]= Lt[3]+d*Lt[1]
179         Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
180     return (Lt) #on renvoie la liste au temps t+1
181
182
183 ##renvoyer une case
184 def case(Lt,n):
185     return (Lt[n])
186
187 ##creerLt pour le début seulement
188 def creerLt(C, H, i): #crée un Lt pour le pays en position i
189     Lt=[0, 0, 0, 0]
190     Lt[0]=H[i][0]-C[i]
191     Lt[1]=C[i]
192     return Lt
193
194 ##creerL au depart
195 def creerL(C, H): #crée la liste des Lt-> cad on a L[0][0] qui donne le nombre de sains dans le pays 0
196     #L[0][1] le nombre de contaminés du pays 0
197     #L[0][2] le nombre de recovered du pays 1
198     L=creertableau(len(C), len(C))
199     for i in range (len(C)):
200         L[i]=creerLt(C,H,i)
201     return L
202
203 ##verifS
204 def verifS(S):
205     n=len(S)
206     res=0
207     for i in range (n):
208         if S[i]==1:
209             res=res+1
210     if res==n:
211         return False
212
213 ##creerV V=voyageurs en prenant en compte les stratégies de confinement
214
215 def creerV(L, C, S, Pop):
216     paspossible=[]
217     for i in range (len(C)):
218         if Pop[i]==0:
219             paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
220     V=[0]*(len(C))
221     for i in range (len(C)):
222         V[i]=[0]*2
223     if verifS(S)==False:
224         for i in range (len(C)):
225             V[i][0]=0
226     else :
227         for i in range (len(C)):
228             if S[i]==0:
229                 V[i][1]= rd.randint(0, len(C)-1)
230                 while S[V[i][1]]==1 :
231                     V[i][1]=rd.randint(0, len(C)-1)
232                 while V[i][1] in paspossible :
233                     V[i][1]=rd.randint(0, len(C)-1)
234                 V[i][0]=rd.uniform(0, 0.0002)*(L[i][0]+L[i][1]+L[i][2])#on ne veut pas que plus de la moitié de la population parte sinon pas
235             else:
236                 V[i][0]=0
237     return (V)
238
239 ##creerD
240 def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
241     D=[0]*(len(C))
242     T=taux(C, L)
243     for i in range (len(C)):
244         D[i]=[0]*2
245     for i in range (len(C)):
246         D[i][0]=(V[i][0]*T[i])
247         D[i][1]=V[i][1]
248     return (D)
249
250 ##creerE
251 def creerE(H, S, E, exports, Pop):
252     paspossible=[]
253     for i in range (len(H)):
254         if Pop[i]==0:
255             paspossible.append(i)
256     if verifS(S)==False:
257         for i in range (len(E)):
258             for k in range (len(E[i])):
259                 E[i][k]=0
260     else:
261         for i in range (len(E)):
262             if S[i]==0:
263                 for k in range (len(E[i])):
264                     if k%2==0:
265                         E[i][k]=exports[i][k]*Pop[i]/365
266                     else:
267                         if k in paspossible:
268                             E[i][k-1]=0
269                         else:
270                             E[i][k]=exports[i][k]
271

```



```

272         else:
273             for k in range (len(E[i])):
274                 E[i][k]=0
275     return E
276
277     ##trouver les contaminés
278     def recherche(D, i):
279         c=0 #res
280         for k in range (len(D)):
281             if D[k][1]==i:
282                 c=c+D[k][0]
283         return c
284
285     def recherchebis(E,H,i):
286         c=0
287         for k in range (len(E)):
288             for a in range (len(E[i])):
289                 if a%2==1:
290                     if E[k][a]==H[i][3]:
291                         c=c+abs(E[k][a-1])
292         return c
293
294     ##creer beta
295     def creertableaubeta(n,C):
296         beta=creertableau(len(C), n+1)
297         for i in range (len(C)):
298             beta[i][0]=0.39
299         return beta
300
301     ##creer Z
302     def creerZ(H, prop):
303         Z=[0]*len(H)
304         for i in range (len(Z)):
305             Z[i]=H[i][1]*prop
306         return Z
307
308     ##faire manger les gens
309     def manger(Pop, Z, i, L):
310         consoparhab=0.393750627
311         viv=Pop[i] # nos vivants avant de voir qui aura assez à manger
312         ts=0
313         tc=0
314         tr=0
315         survivants=0
316         capacité=0
317         nouveauxmorts=0
318         if Pop[i]!=0:
319             ts=L[i][0]/viv
320             tc=L[i][1]/viv
321             tr=L[i][2]/viv
322             capacité=Z[i]/0.393750627
323             if capacité<viv:
324                 survivants=capacité
325                 nouveauxmorts=viv-survivants
326                 Pop[i]=survivants
327                 L[i][0]=ts*Pop[i]
328                 L[i][1]=tc*Pop[i]
329                 L[i][2]=tr*Pop[i]
330                 L[i][3]=L[i][3]+nouveauxmorts
331                 Z[i]=0
332             else:
333                 Z[i]=Z[i]-Pop[i]*consoparhab
334
335     return (Z[i], L[i])
336
337
338     ##expansion avec confinement
339     def prevoirstocks(C, H, n, sc, sd, exports):# d le taux de gens qui meurent
340         Pop=[0]*len(H)
341         for i in range (len(H)):
342             Pop[i]=H[i][0]
343         L=creerL(C, H)
344         beta=creertableaubeta(n, C)
345         gamma=0.3
346         d=0.048
347         S=[0]*len(C))
348         T=taux(C,L)
349         evoltaux=creertableau(len(C), n)
350         stocksneessaires=[0]*len(C)
351         E=creertableau(len(C), len(exports))
352         S=seuildep(T,sc, sd,S)
353         V=creerV(L,C,S, Pop)
354         D=creerD(H,C,V,L) #D le tableau des gens dangereux qui se déplacent et vers quel pays ils se déplacent
355         evolcontamines=creertableau(len(C), n) #evolution des contaminés
356         evolmorts=creertableau(len(C),n)
357         evoldeE=creertableau(len(C), n)
358         evolimports=creertableau(len(C),n)
359         evoldeH=creertableau(len(C), n)
360         listedeS=creertableau(len(C),n)
361         prodpartravailleurparjour=[]
362         for i in range (len(C)):
363             evolmorts[i][0]=0
364             evoldeH[i][0]=Pop[i]
365             evoldeE[i][0]=0
366             evoltaux[i][0]=T[i]
367             evolimports[i][0]=0
368             listedeS[i][0]=S[i]
369         for i in range (len(H)):
370             prodpartravailleurparjour.append(H[i][1]/H[i][0]/365)
371         for k in range (1, n):
372             T=taux(C,L)
373             S=seuildep(T,sc, sd ,S)
374             V=creerV(L,C,S, Pop)
375             D=creerD(H,C,V,L)
376             E=creerE(H, S, E, exports, Pop)
377             for i in range (len(C)):
378                 #partie épidémiologique
379                 L[i]=SIREvol(L[i], beta, i, k)
380                 C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine
381                 Pop[i]=Pop[i]+recherche(V,i)-V[i][0]
382                 L[i]=[Pop[i]-C[i]-L[i][2],C[i],L[i][2],L[i][3]]
383                 evolcontamines[i][k]=C[i]
384                 evolmorts[i][k]=L[i][3]
385                 listedeS[i][k]=S[i]
386                 if listedeS[i][k]==1:
387                     beta[i][k+1]=0.22
388                 else:
389                     beta[i][k+1]=0.39
390                 evoltaux[i][k]=tauxlocal(C,L, i)
391             #partie échanges de céréales
392             travailleurs=(L[i][0]+L[i][2])

```

```

393     evolmorts[i][k]=L[i][3]
394     evolcontamines[i][k]=L[i][1]
395     evolmorts[i][k]=L[i][3]
396     evoldeH[i][k]=Pop[i]
397     evoldeE[i][k]=somme3(E, i)
398     evolimports[i][k]=recherchebis(E,H,i)
399     if recherchebis(E, H, i)-somme3(E, i)>=0:
400         conso=(L[i][0]+L[i][1]+L[i][2])*0.393750627
401     else:
402         conso=(L[i][0]+L[i][1]+L[i][2])*0.393750627+somme3(E, i)-recherchebis(E, H, i)
403     stocksnecessaires[i]=stocksnecessaires[i]+conso
404
405     return (stocksnecessaires) #on renvoie la liste des états de chaque pays
406
407
408 ##variation des stocks avec l'aléatoire
409 def variationdesstocks(C, H, n, sc, sd, exports, nombre): #trop grande complexité
410     L=creertableau(len(H), nombre)
411     for i in range (nombre):
412         L[i]=prevoirstocks(C, H, n, sc, sd, exports)
413     return L
414
415 def remettredansordre(C, H, n, sc, sd, exports, nombre): #L est le résultat de "variation des stocks"
416     L1=creertableau(nombre,len(C))
417     for i in range (len(L)):
418         for k in range (len(L[i])):
419             L1[i][k]=L[i][0]
420     return L1
421
422
423 ##tracer des évolutions en fonction
424 def tracervariations(L1, nombre): #L1 le resultat de "remettre dans l'ordre"
425     abscisse=[0]*(10)
426     for i in range (nombre):
427         abscisse[i]=i
428     for i in range (len(L1)):
429         for k in range (nombre):
430             plt.plot(abscisse, L1[i])
431             plt.title("Stocks prévus au fil des essais")
432             plt.xlabel("Essai")
433             plt.ylabel("Stock prévu")
434             plt.legend(fontsize=6, loc='best')
435     return(plt.show())
436
437
438 def tracercontamines(C, H, n,sc, sd):
439     evol=expansionconfined(C, H, n, sc, sd)[1]
440     abscisse=[0]*n
441     liste=listenum(H)
442     for i in range (n):
443         abscisse[i]=i
444     for k in range (len(C)):
445         plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[listenum(k)]))
446         plt.legend()
447         plt.title("Nombre de contaminés en fonction du temps")
448         plt.xlabel("Temps en jours")
449         plt.ylabel("Nombre de contaminés")
450         plt.legend(fontsize=6, loc='best')
451     return(plt.show())
452
453 def tracermorts(C, H, n, sc, sd, exports, prop):
454     evol=expansionconfined(C, H, n, sc, sd, exports, prop)[0]
455     abscisse=[0]*n
456     liste=listenum(H)
457     for i in range (n):
458         abscisse[i]=i
459     for k in range (len(C)):
460         plt.plot(abscisse,evol[k], label=str(NOM_PAYS[listenum(k)]))
461         plt.legend()
462         plt.title("Nombre de morts en fonction du temps")
463         plt.xlabel("Temps en jours")
464         plt.ylabel("Nombre de morts")
465         plt.legend(fontsize=6, loc='best')
466     plt.show()
467
468 def touttracer(C, H, n, sc, sd, exports, prop):
469     RES=expansionconfined(C, H, n, sc, sd, exports, prop)
470     abscisse=[0]*n
471     liste=listenum(H)
472     for i in range (n):
473         abscisse[i]=i
474     for a in range (len(RES)):
475         for k in range (len(liste)):
476             plt.plot(abscisse, RES[a][k], label=str(NOM_PAYS[k]))
477             plt.legend()
478             plt.xlabel("temps en jour")
479             plt.ylabel(str(AFFICHER[a]))
480             plt.legend(fontsize=6, loc='best')
481     plt.show()
482
483 ##somme
484 def somme3(E, i):
485     s=0
486     for k in range (len(E)):
487         s=s+E[i][k]
488     return (s)
489
490 ##utile
491 def proporprod(L, H):
492     for i in range (len(L)):
493         L[i]=L[i]/H[i][1]
494     return (L)
495
496 ##application
497 #prevoirstocks([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
498 0.003, 0.0006, exports)
499
500 import pylab
501 #prevoirstocks([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10, 3, 0, 9, 0, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
502 0.003, 0.0006, exports)
503
504 fig = plt.figure()
505 x = [1,2,3,4,5,6,7,8,9,10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37]
506 height = [13179602243.997686, 6892987472.203469, 11969700784.011234, 9346454127.204073, 8787367548.55477, 6061499757.5266, 2006606849.3050022,
1756815080.9369986, 1674185734.3593967, 1679214782.9196668, 2438800032.144745, 1740571640.707238, 1520457570.8476877, 1529351574.4517322,
1066979659.2297381, 1129617943.1920557, 1006668653.713588, 1015449732.682125, 951629073.4985526, 2582462139.1195393, 6246418598.963533, 5130159497.487438,
1388651385.984678, 267276211.68678853, 5128136633.1681, 2026011634.9270985, 14623673358.506582, 12141549905.948044, 6535768885.866674, 47622085470.957504,
8720893966.515703, 3999550837.424484, 2967643633.1375127, 23354429417.93404, 9662916396.21039, 196699042777.19144, 192200243893.37936]
507 width = 0.05
508 BarName = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
"Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie", "Norvège", "Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israël", "Malte", "Maroc",

```

```

507 "Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]
508 plt.bar(x, height, width, color=(0.65098041296005249, 0.80784314870834351, 0.89019608497619629, 1.0) )
509 plt.scatter([i+width/4.0 for i in x],height,color='b',s=10)
510
511 plt.xlim(0,38)
512 plt.ylim(0,200000000000)
513 plt.grid()
514
515 plt.ylabel('Counts')
516 plt.title('stocks nécessaires par pays')
517
518 pylab.xticks(x, BarName, rotation=40)
519
520 plt.savefig('SimpleBar.png')
521 plt.show()
522

```