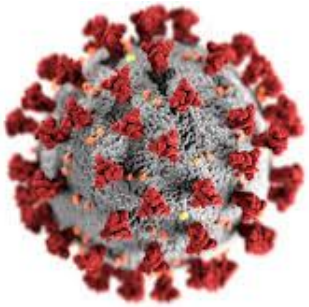
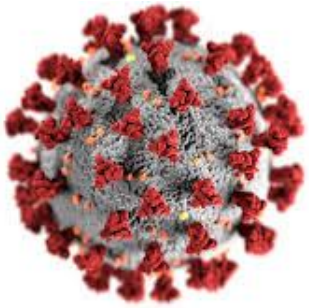


Influence d'une pandémie mondiale impliquant des stratégies de confinement telle que celle provoquée par la covid-19 sur un plan sanitaire et sur un plan alimentaire

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 :

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -\beta SI(t)$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \beta S(t)I(t) - \gamma I(t) - \delta I(t)$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = \gamma I(t)$$

$$\frac{\mathrm{d}D}{\mathrm{d}t} = \delta I(t)$$

En temps discret :

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

$$I(t+1) - I(t) = \beta S(t)I(t) - \gamma I(t) - \delta I(t)$$

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

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

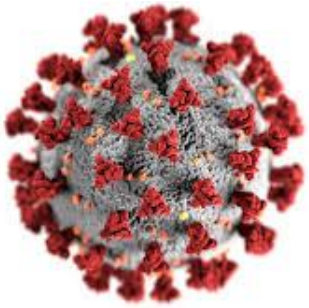
Paramètres:

- 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

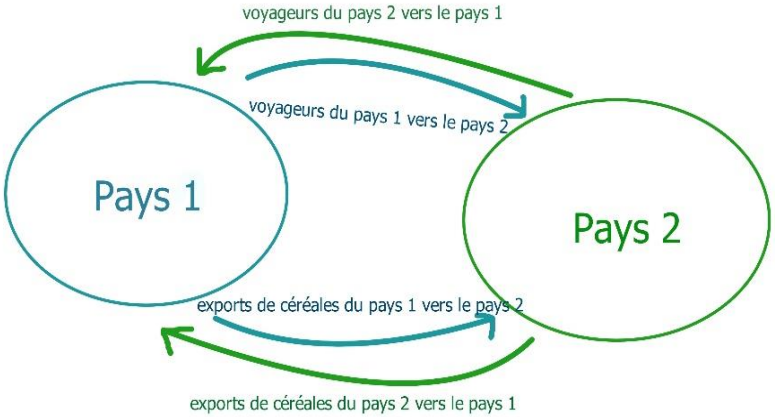
Estimation du paramètre β :

- Sans confinement : β=0.39
- Avec confinement: β=0.22

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

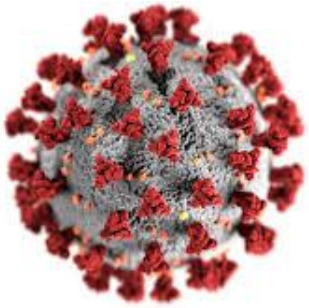


Echanges internationaux



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)

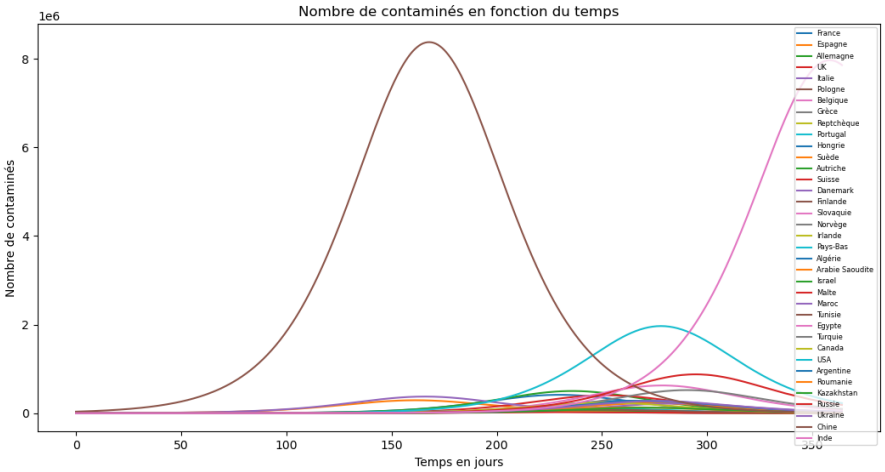
Pays	Contaminés au 1er mars 2020	Morts au 1er mars 2020
France	116	2
Espagne	1486	0
Allemagne	114	0
UK	35	0
Italie	1578	41
Pologne	0	0
Belgique	1	0
Grece	7	0
Republique Tcheque	3	0
Portugal	0	0
Hongrie	0	0
Suede	14	0
Autriche	14	0
Suisse	23	0
Danemark	4	0
Finlande	5	0
Slovaquie	0	0
Norvege	19	0
Irlande	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

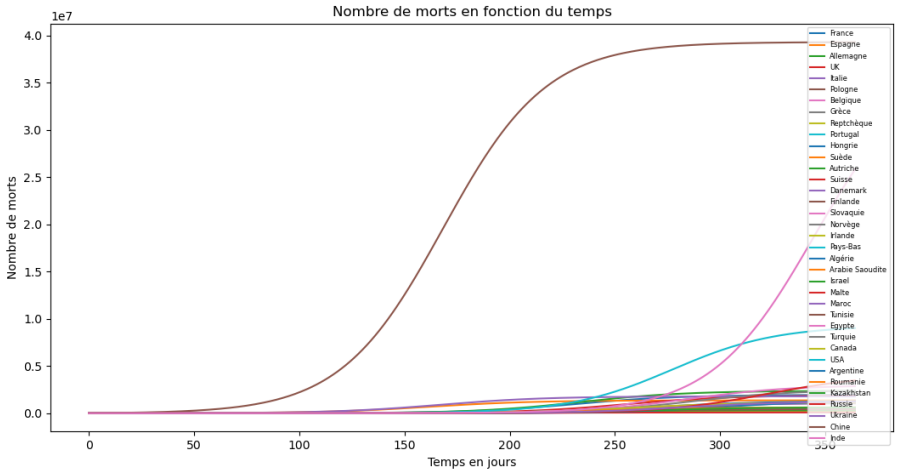


Sans confinement : au niveau mondial

NOMBRE DE CONTAMINES

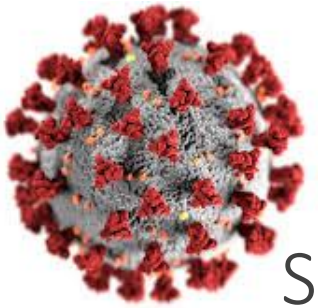
On observe des courbes en cloche pour tous les pays. La première phase de montée est exponentielle. On distingue seulement les courbes de la Chine et l'Inde dû aux tailles de populations.





NOMBRE DE MORTS

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

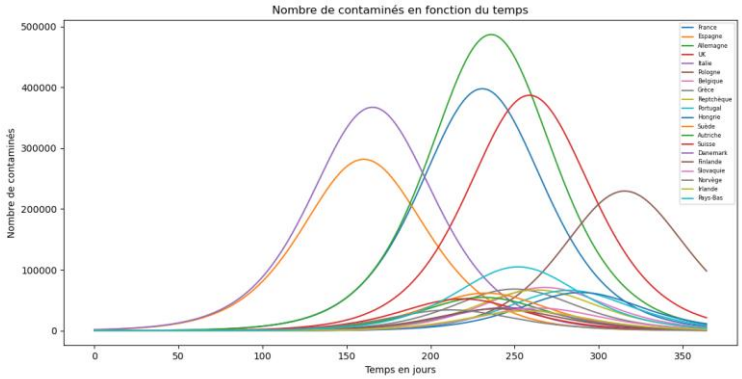


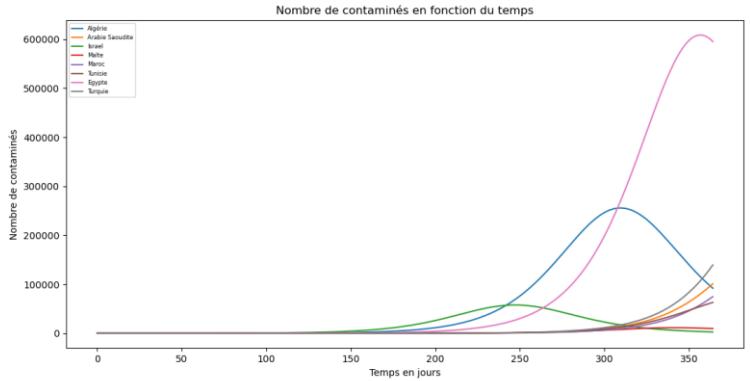
Sans confinement : dynamiques territoriales

Europe

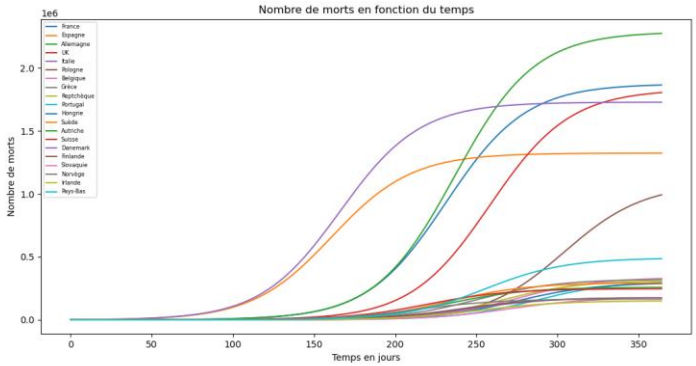
AFNMO

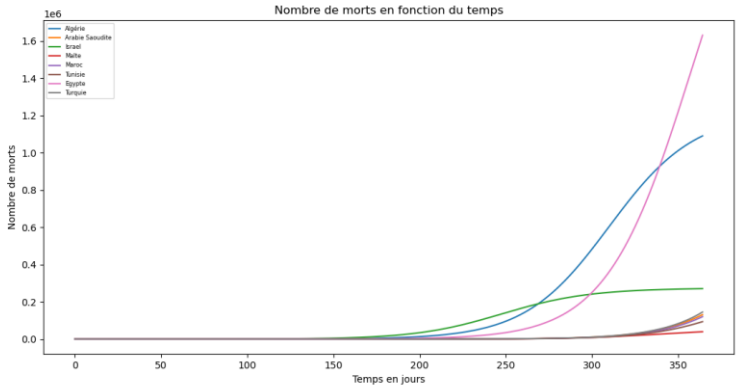
CONTAMINES

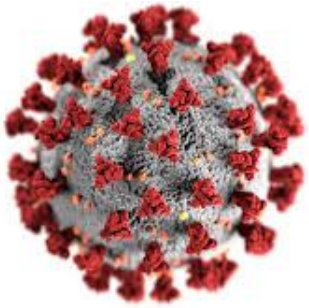




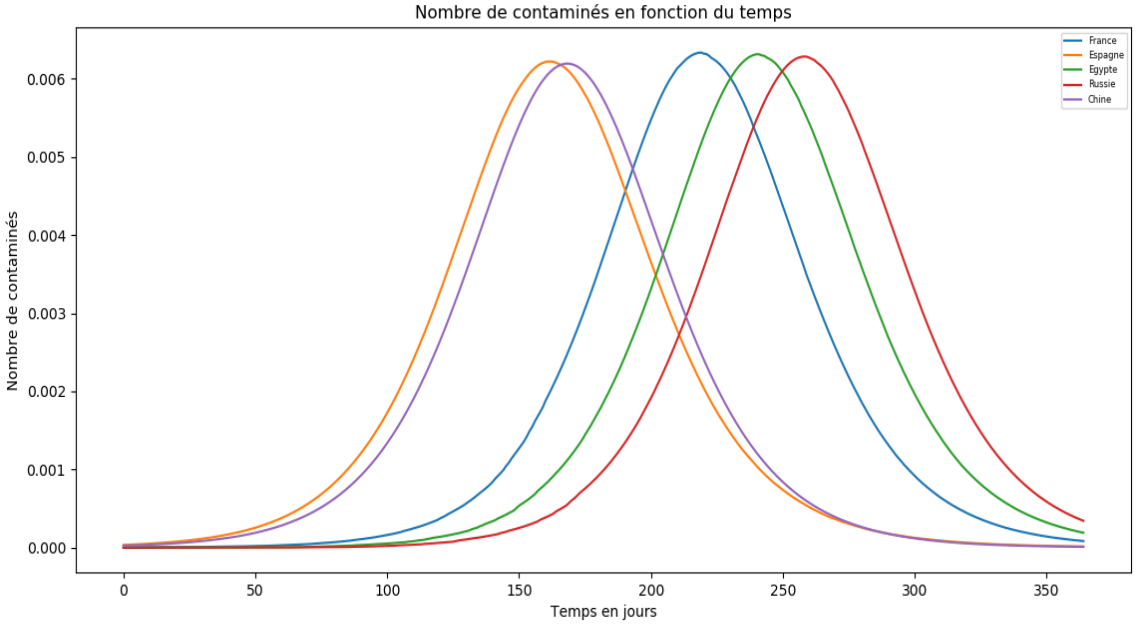
MORTS





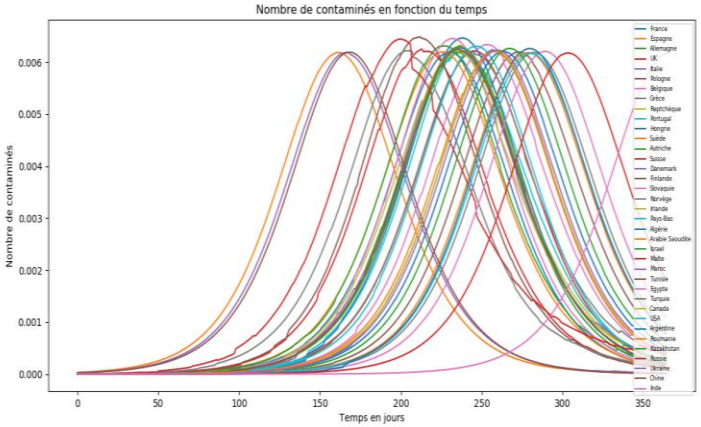


Sans confinement – taux de contaminés

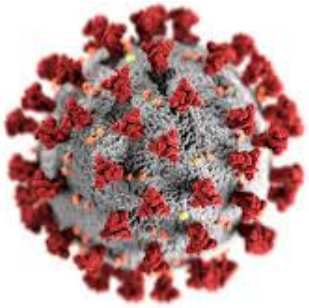


TAUX DE CONTAMINES

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.



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



Cohérence du modèle

Immunité de groupe:

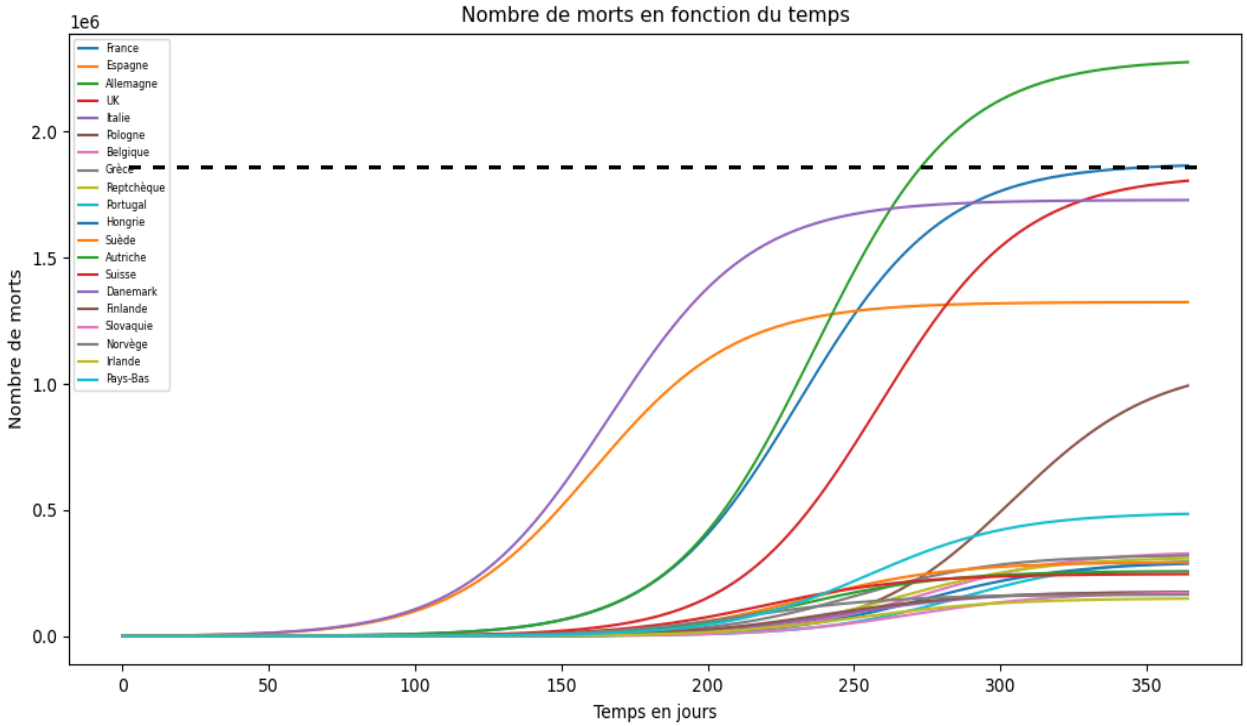
R0 = nombre moyen d'individus immunologiquement naïfs qu'un sujet va infecter après contact

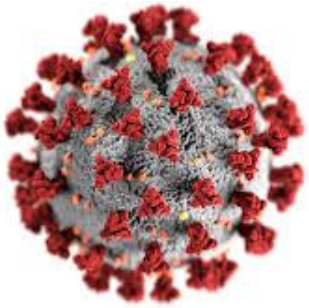
Immunité collective = 1 - 1/R0

Pour le covid-19 : R0 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.

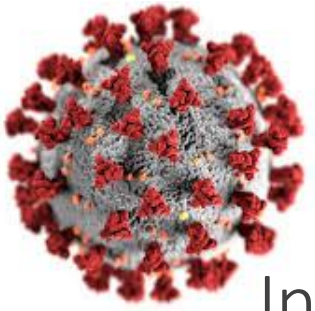
De plus, les interactions à l'intérieur même du pays sont fortement diminuées. Tant que le pays n'est pas confiné, $\beta=0.39$. Quand il se confine, $\beta=0.22$.

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

Taux de contaminés limites

Taux seuil de confinement : 0.003

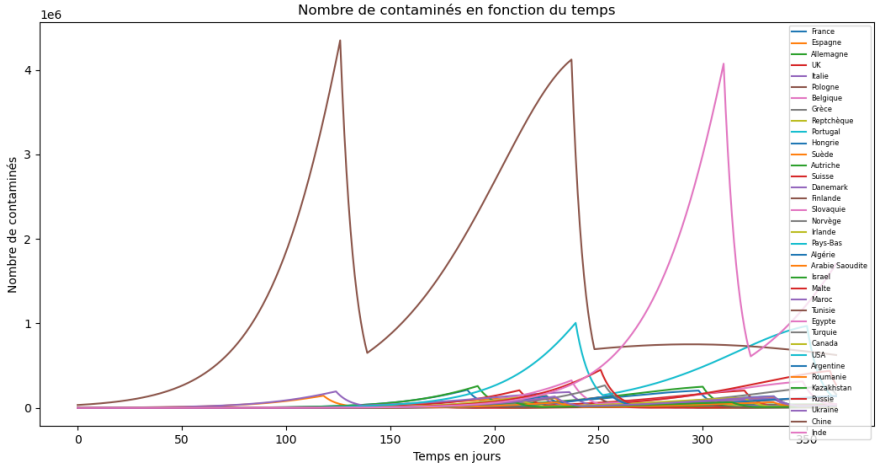
Taux seuil de déconfinement : 0.0006

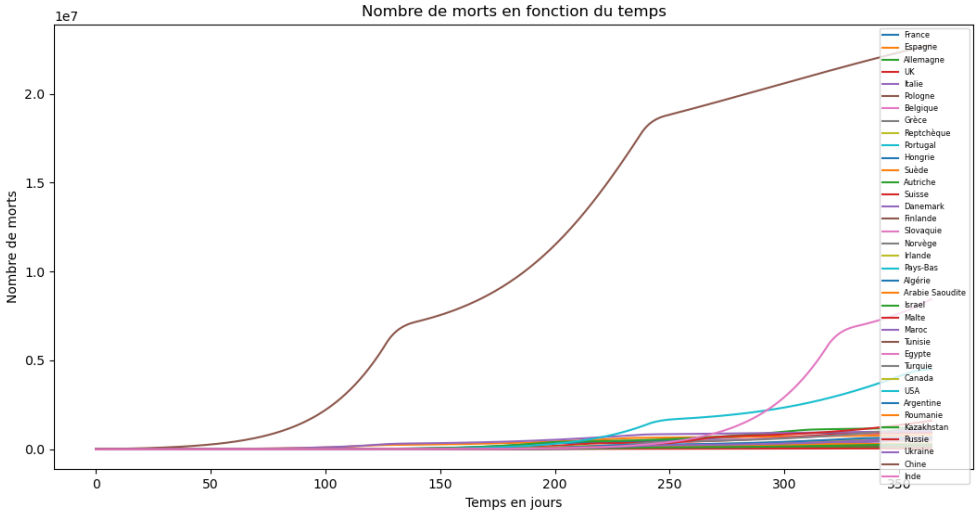


Influence d'un confinement – Résultats monde

NOMBRE DE CONTAMINES

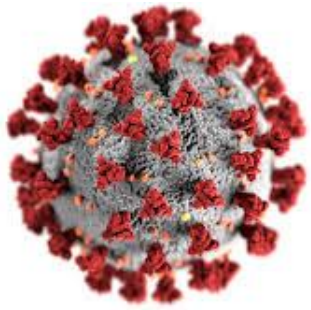
Dans un premier temps, le nombre de contaminés augmente, puis lors du confinement diminue d'où les pics. On observe non pas un confinement, mais une succession de confinements qui s'enchaînent.





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.

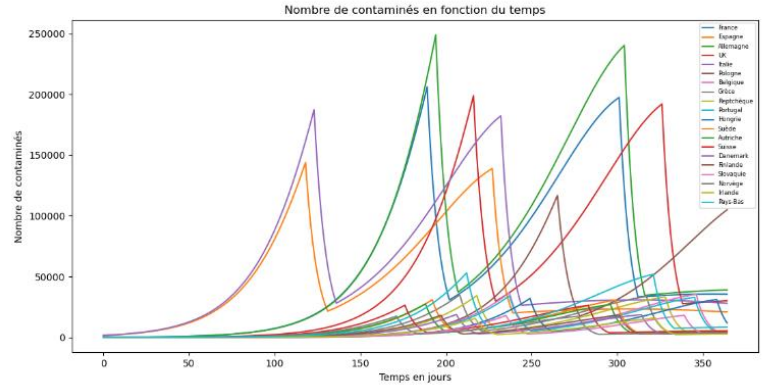


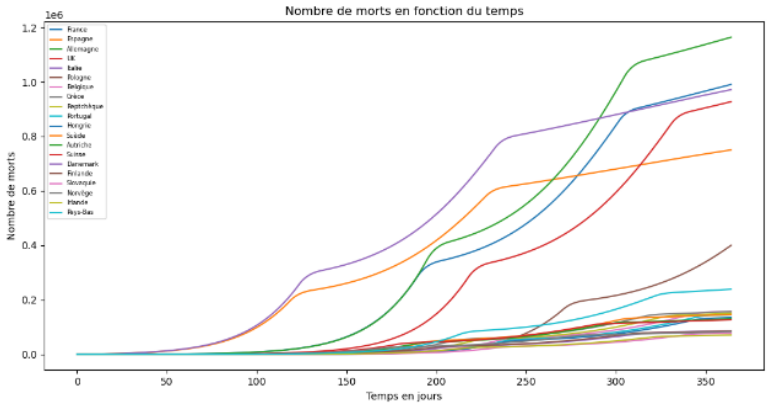
Confinement : résultats Europe et AFNMO

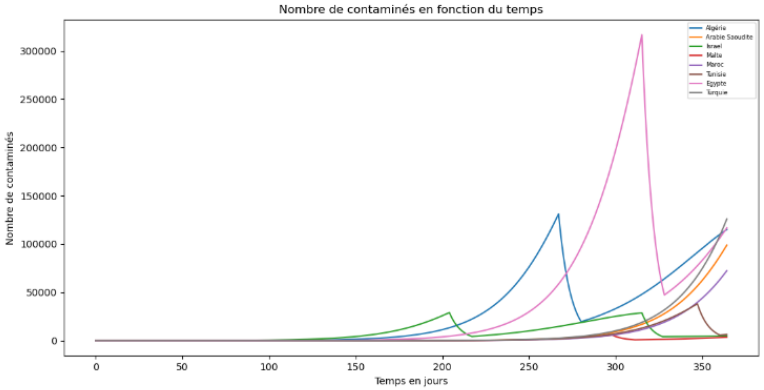
Europe

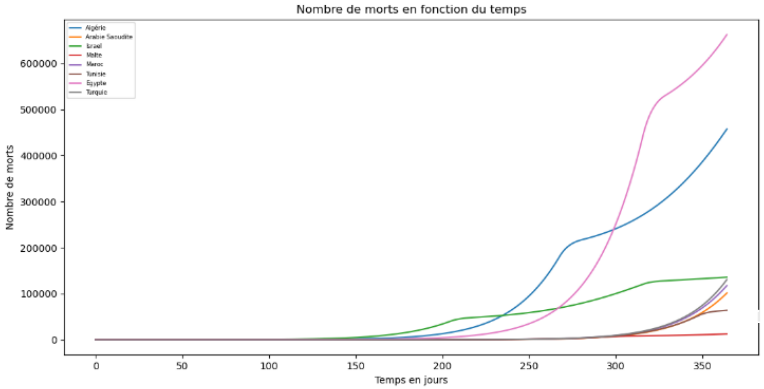
AFNMO

CONTAMINES

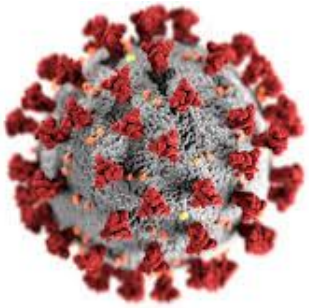








MORTS



Biens alimentaires : Modèle

Biens alimentaires considérés:

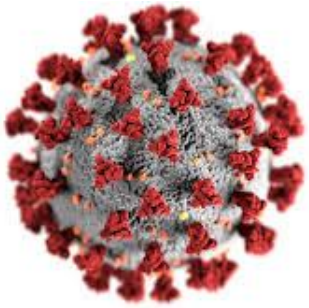
- Uniquement le blé

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

- Production par habitant journalière harmonisée : production annuelle/ 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:

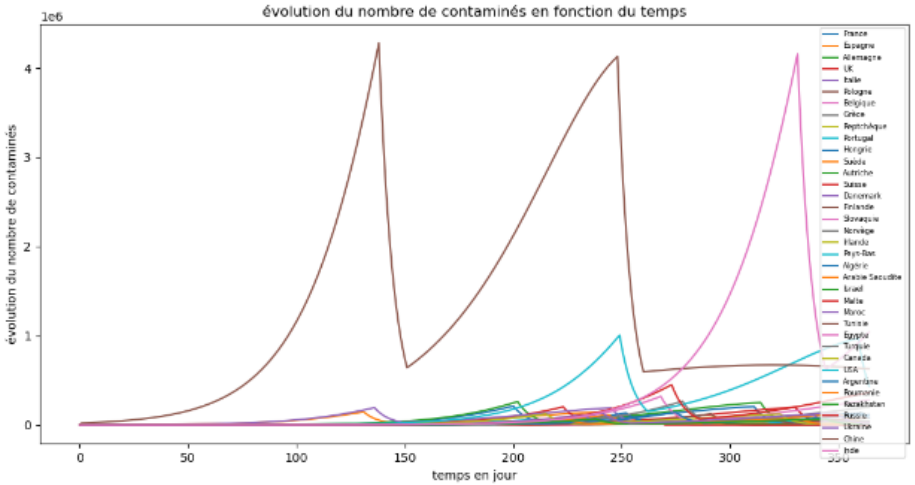
- Les pays ne peuvent ni exporter ni importer de nourriture

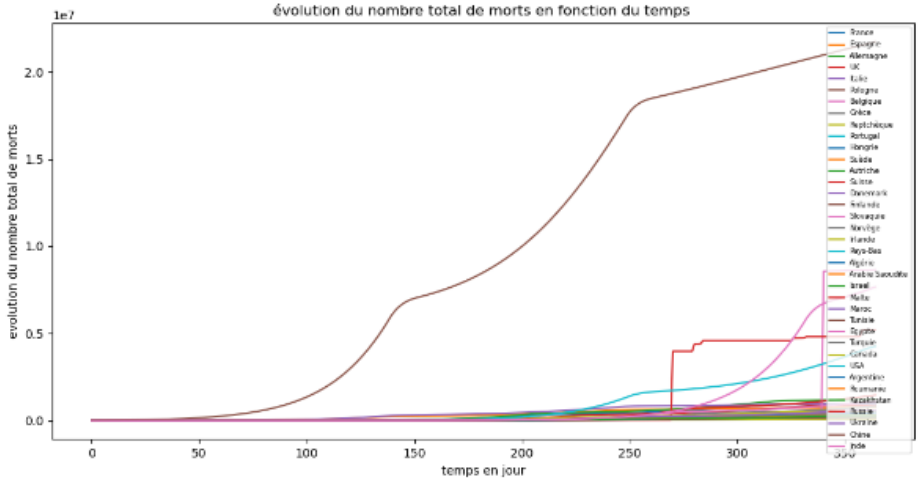


Biens alimentaires: 30 ans de stock

NOMBRE DE CONTAMINES

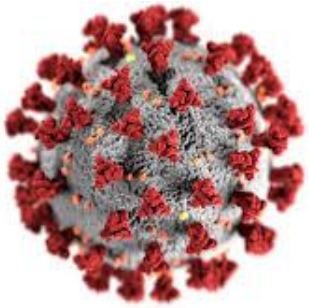
On observe la même tendance que lorsque l'on ne prenait pas en compte les stocks.





NOMBRE DE MORTS

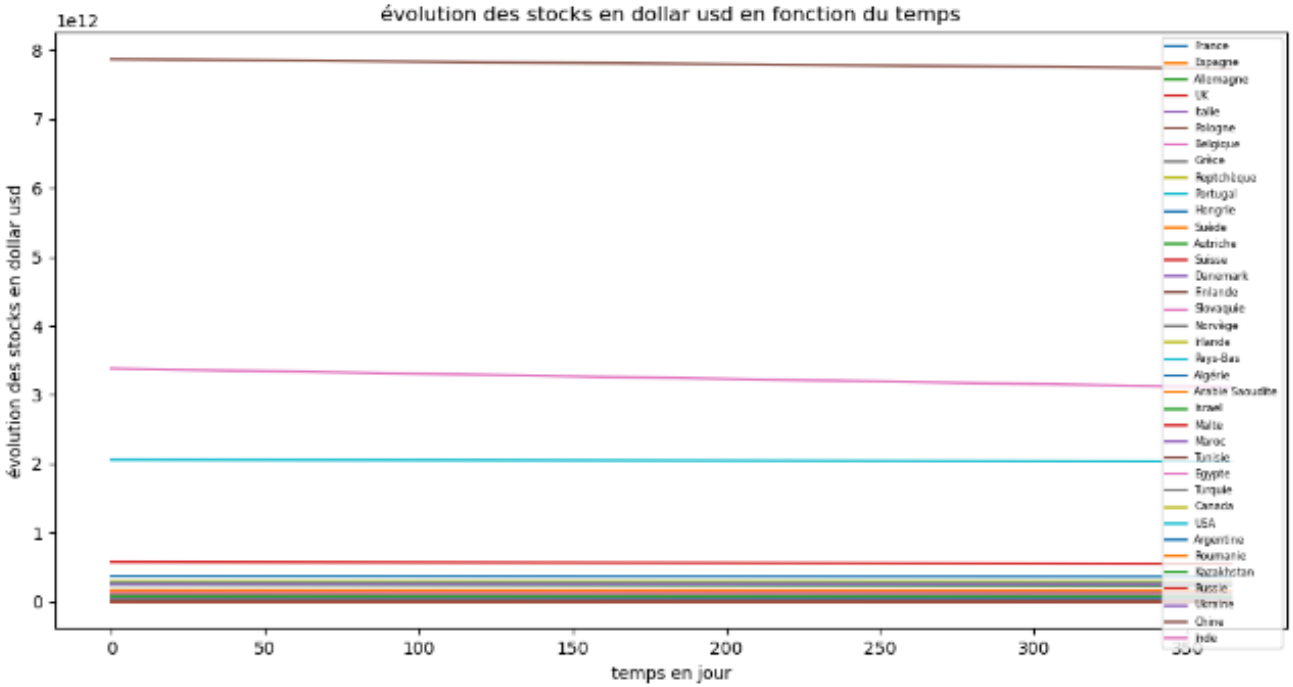
On retombe sur une courbe analogue au début de l'étude ce qui paraît cohérent. En effet, en se plaçant avec 30 ans de stocks, on a des stocks « infinis » donc il n'y aura aucun mort à cause de pénurie. Donc on a seulement l'effet de l'épidémie.

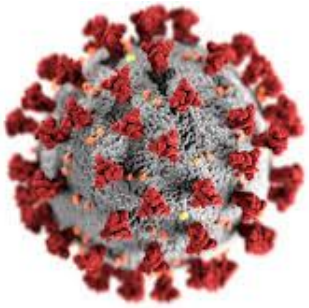


Biens alimentaires: 30 ans de stock

STOCKS

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

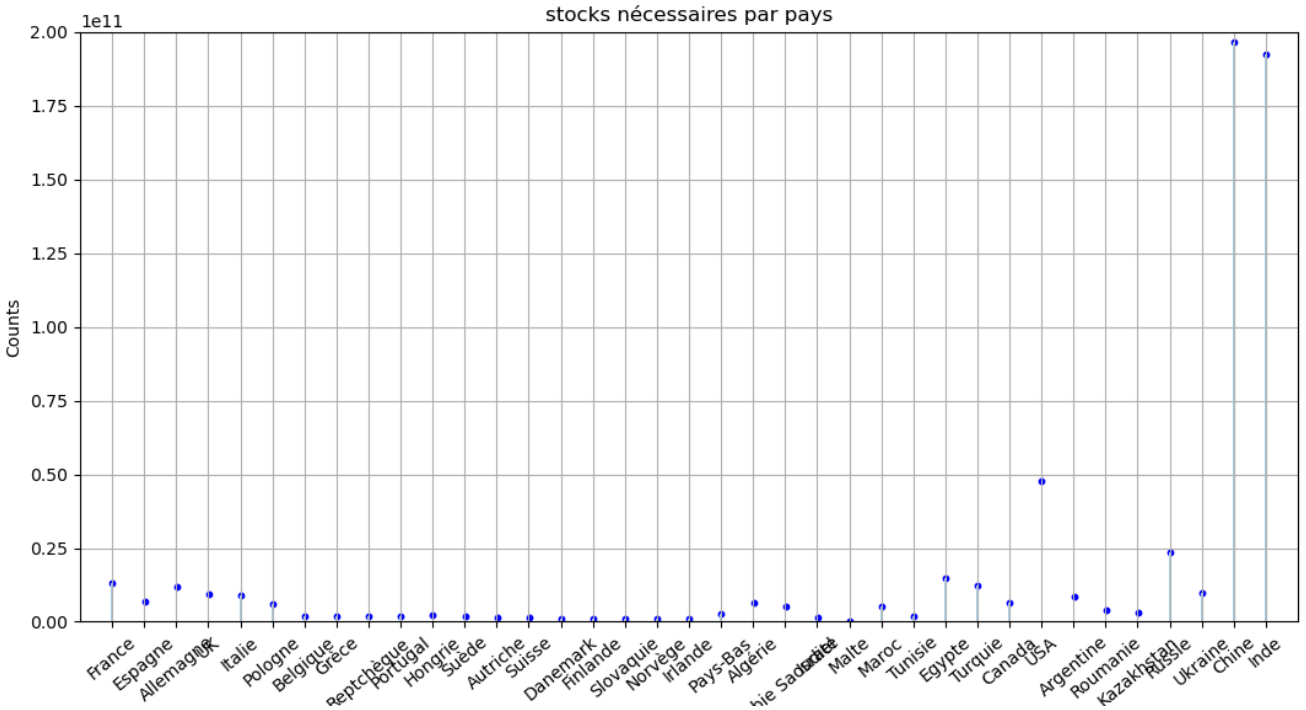


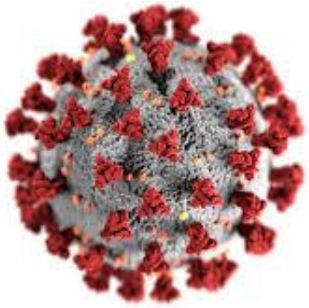


Biens alimentaires : prévision des stocks

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.



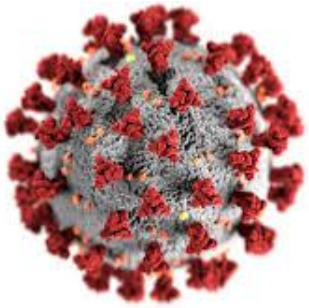


Biens alimentaires : prévision des stocks

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.

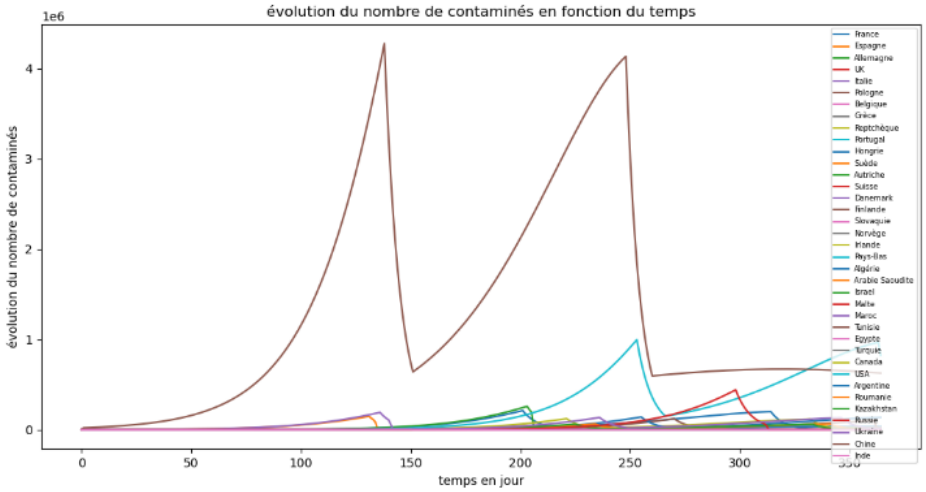
Pays	Stock nécessaire (10^9 USD)	Proportion de la production par an
France	13,18	1.07
Espagne	6,89	1.94
Allemagne	11,97	1.47
UK	9,35	2.14
Italie	8,79	2.04
Pologne	6,06	1.12
Belgique	2,00	4.36
Grèce	1,76	2.00
Republique Tchèque	1,67	1.20
Portugal	1,68	6.62
Hongrie	2,44	1.06
Suède	1,74	1.77
Autriche	1,52	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

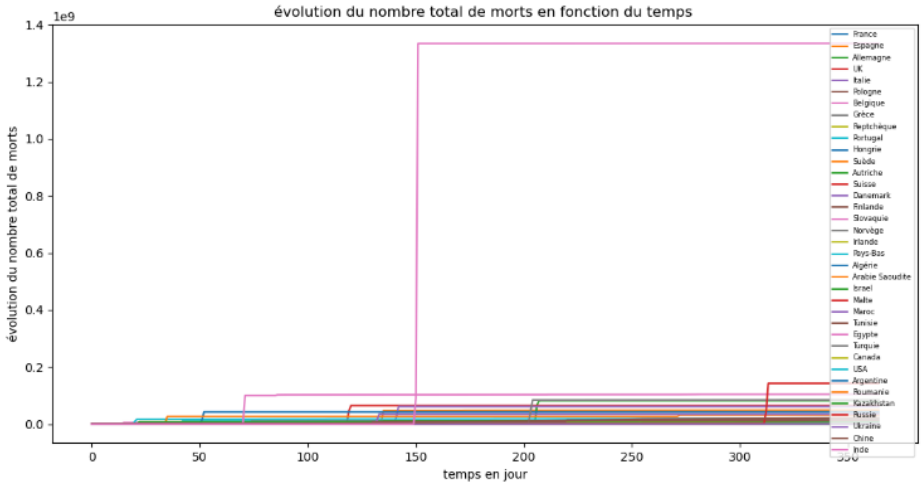


Biens alimentaires: 1 an de stock

NOMBRE DE CONTAMINES

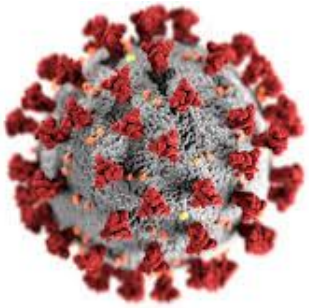
On observe des courbes analogues en terme de forme mais il n'y a que 2 vagues car après cela toute la population est morte de faim.





NOMBRE DE MORTS

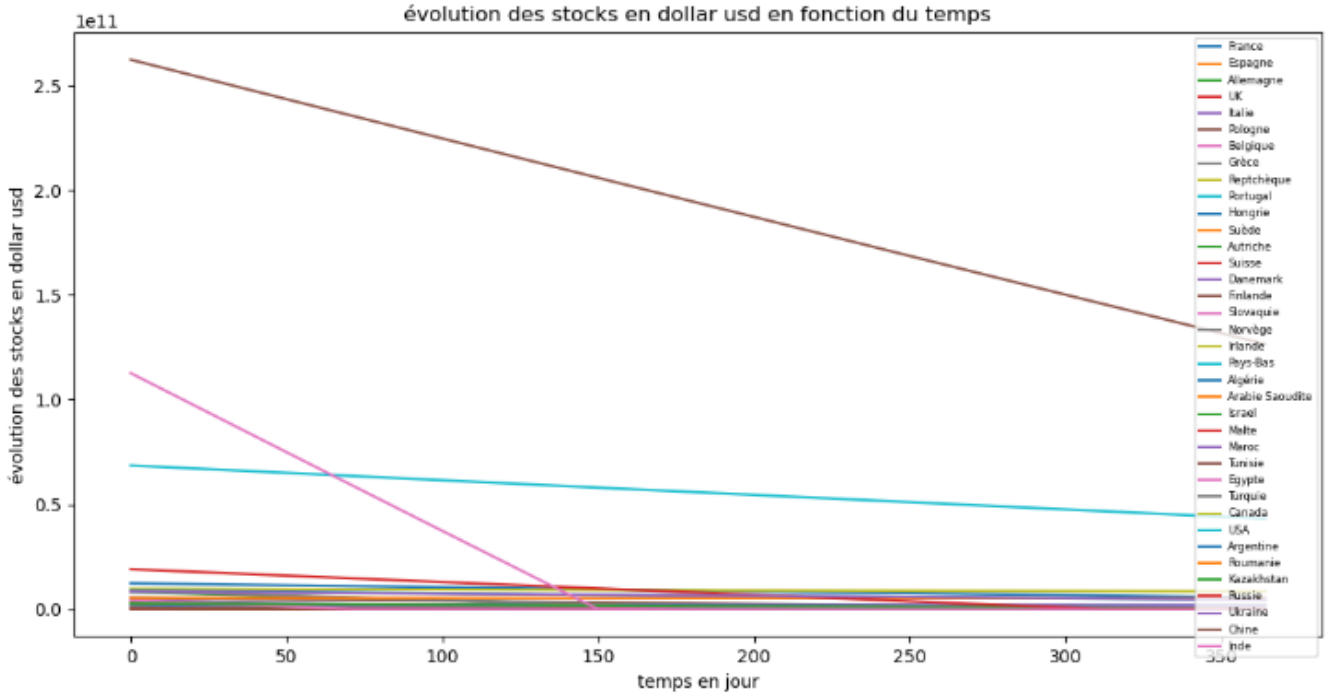
On n'observe plus du tout de courbes analogues à lorsque l'on ne prenait pas les stocks en compte, mais des paliers. Les pays « meurent » complètement à la fin, à cause donc d'un manque de nourriture.

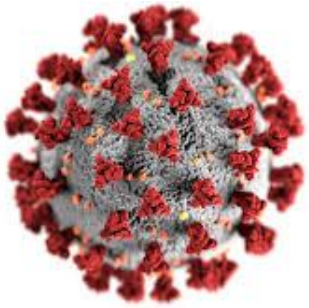


Biens alimentaires: 1 an de stock

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.





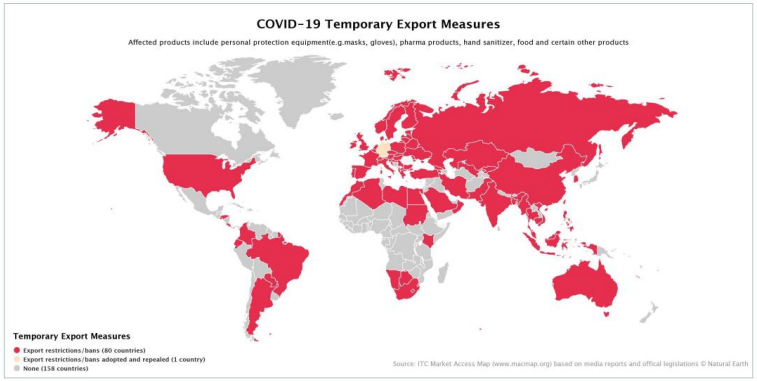
Situation réelle : le blé

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.

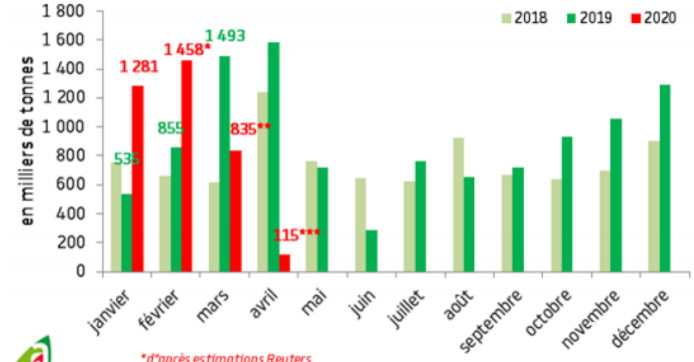
Exemple: La Russie a décidé de limiter ses exportations de céréales à 7 millions de tonnes pour la période d'avril à juin 2020. De fait, pour limiter l'inflation sur les produits issus de la transformation céréalière comme le pain, l'Ukraine a modifié ses exportations de blé à 20.2 millions de tonnes.

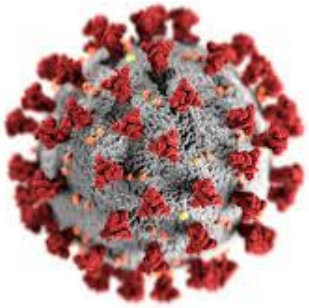
On a observé des appels d'offre de l'Algérie et l'Egypte en lien avec le blé par exemple car ce sont de gros importateurs. La France a donc répondu à ces appels d'offre. Ses exportations de blé vers l'Algérie et le Maroc ont sur cette période représenté 62% de ses exportations totales.

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



Graphique 2 - Exportations françaises de blé vers les pays tiers

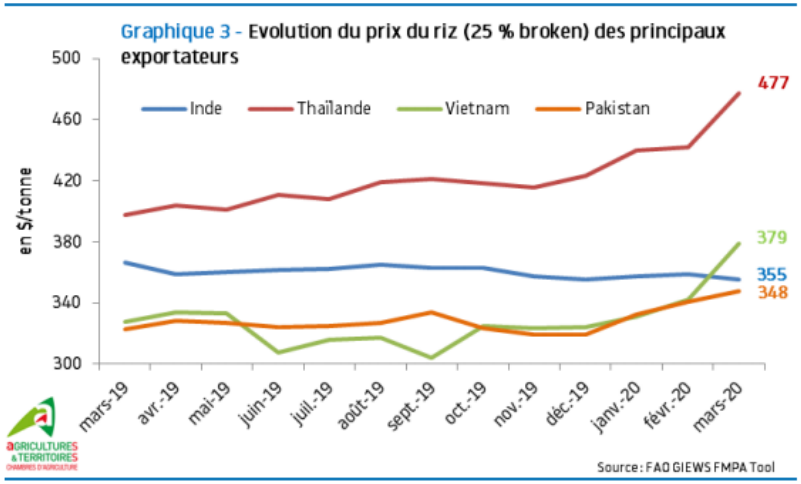


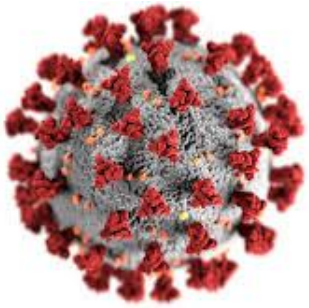


Situation réelle : le riz, le maïs

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.

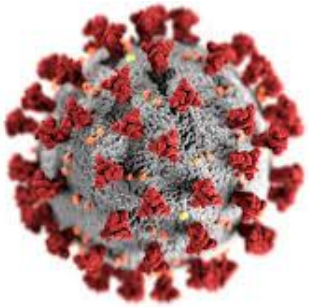
En Afrique, une menace accrue. La fermeture des frontières fragilise les équilibres alimentaires en entravant le commerce transfrontalier et intercontinental. La céréale ici est le maïs. Des pays comme le Bénin, le Sénégal et la Côte d'Ivoire font partie des premiers importateurs mondiaux de riz également. 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	• 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	A partir du tableau des stocks nécessaires, on peut s'assurer de prévoir une sécurité pour les pays non autosuffisants
Limites	• 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)	• 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

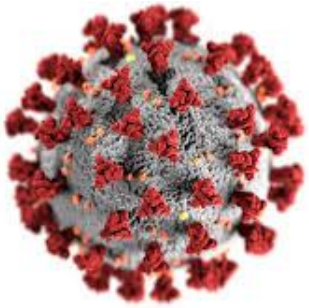
Annexe 2 : Taux de contaminés 1 an de stock

Annexe 3 : Taux de contaminés 30 ans de stock

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

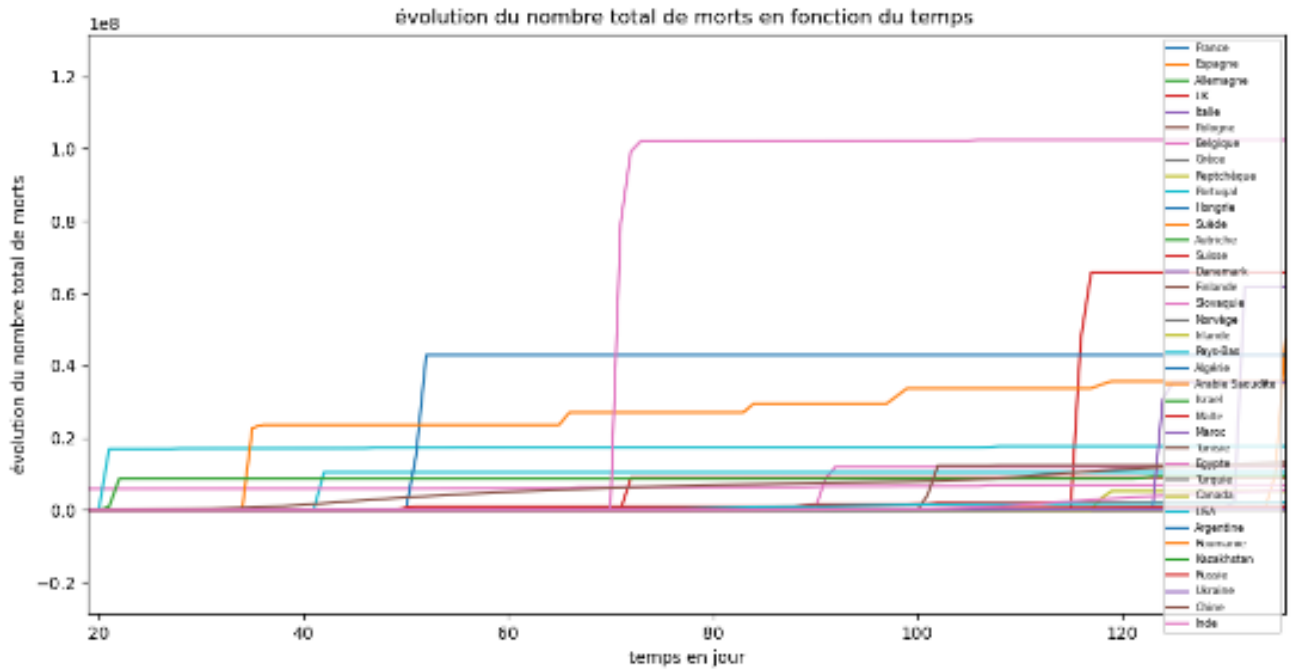
Annexe 5 : Cas particulier : la France, cohérence avec l'épidémie réelle dans le pays

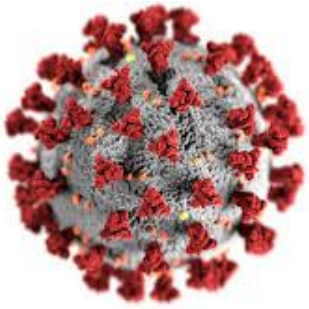
Annexes jusqu'à la fin : Programmes Python



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.

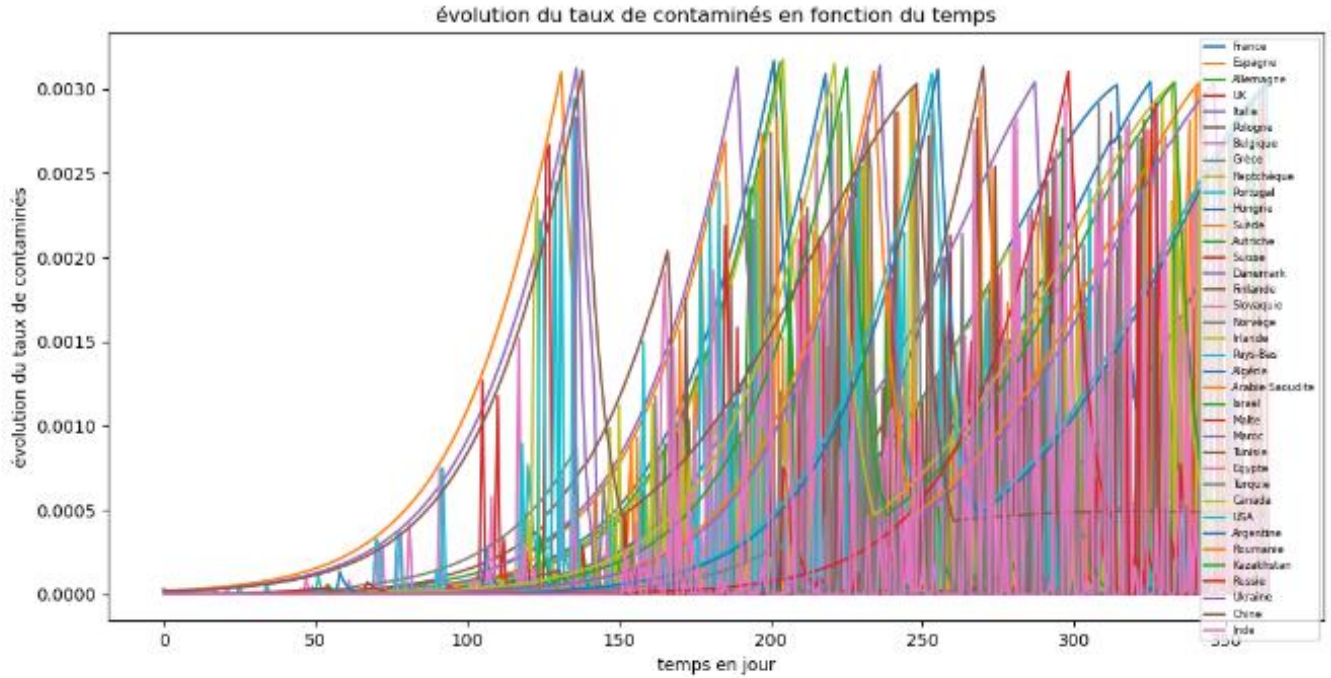


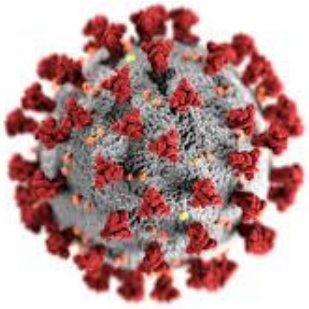


1 an de stock, taux de contaminés

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.

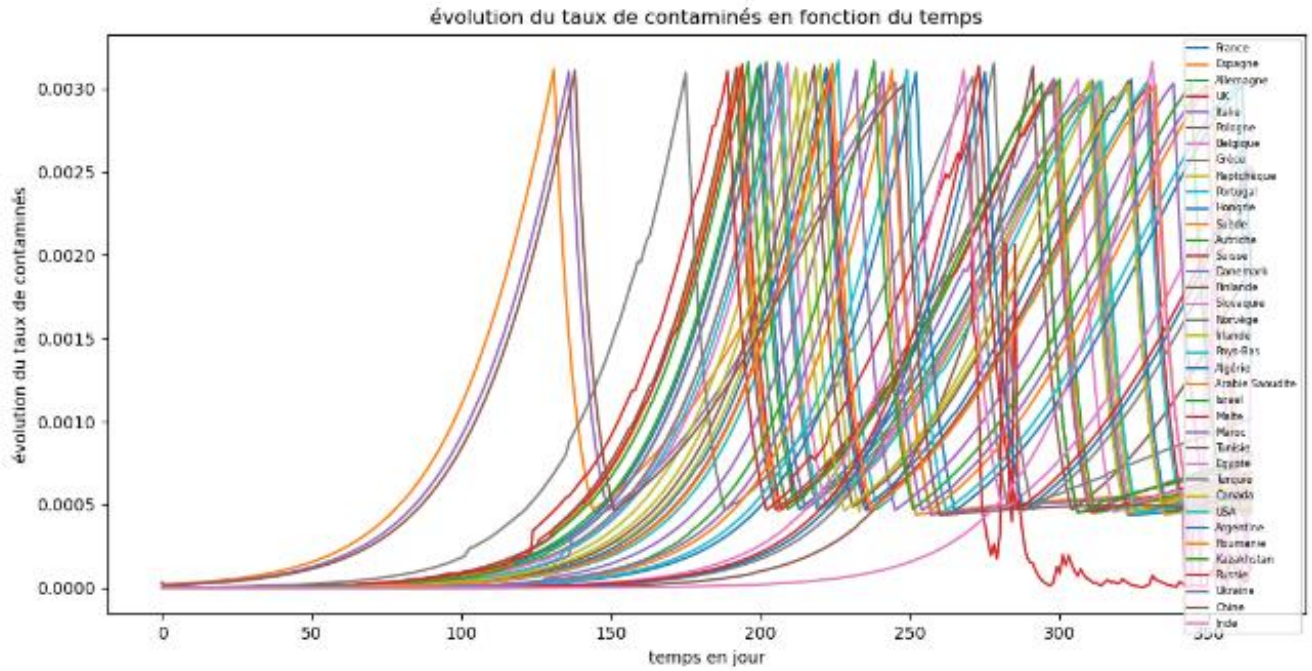


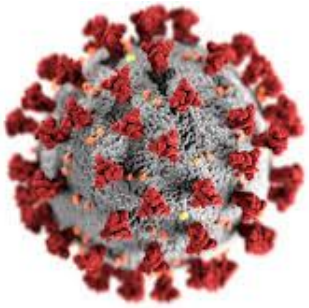


30 ans de stock, taux de contaminés

TAUX DE CONTAMINES

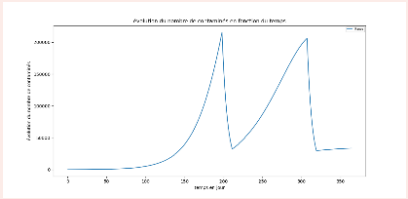
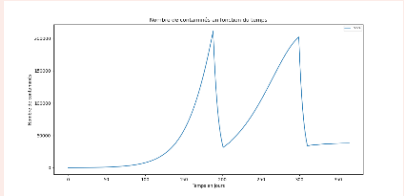
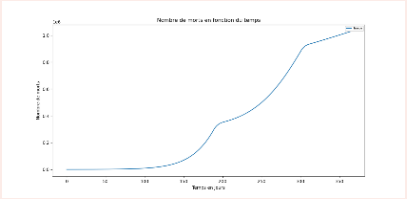
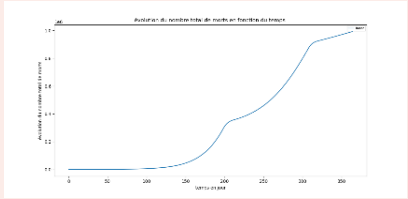
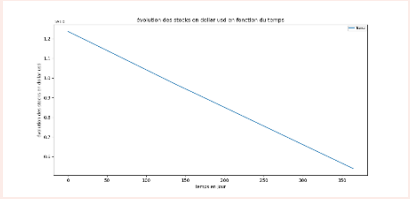
On ne retombe pas à 0 car aucun pays ne perd toute sa population.

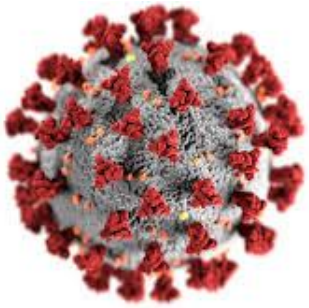




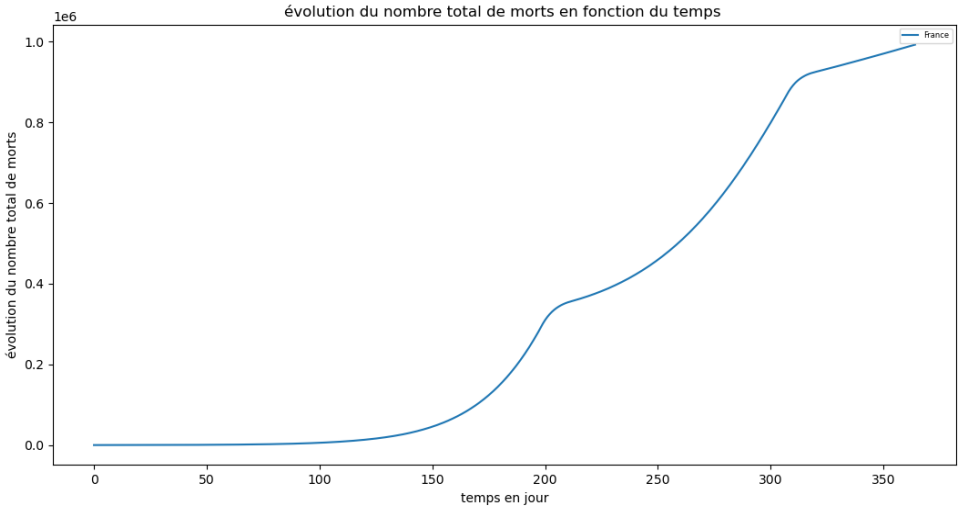
Cas particulier : la France

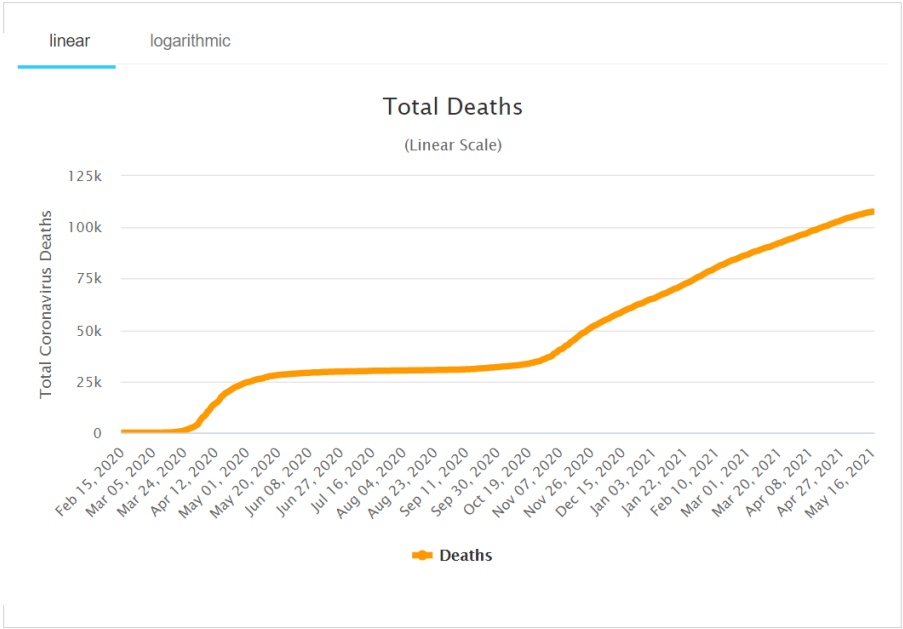
La France est censée d'après les prévisions effectuées survivre à un an de pandémie, car elle est autosuffisante en production de céréales. C'est bien ce que l'on observe.

Courbes obtenues avec prise en compte des stocks	Analyse	Comparaison avec stratégies de confinement
	On a les mêmes courbes étant donné que la France a assez de stocks pour survivre, c'est cohérent.	
	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

```python
# épidémie sans confinement.py

##data
NOMS_PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
"Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie","Norvège","Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israel", "Malte", "Maroc",
"Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]

import numpy as np
import matplotlib.pyplot as plt
import random as rd
from math import floor
from math import ceil

#données relatives aux pays (nombre hab)
#AFNMO où on a de la donnée:
algerie=[42230000, 20]
arabiesaoudite=[34813900, 21]
israel=[8655540, 22]
malte=[514564, 23]
maroc=[34660000, 24]
tunisie=[11180000,25]
egypte=[102334000, 26]
turquie=[84339100, 27]
AFNMO=[algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie]

#europe
france=[66352469,0]
espagne=[46439864, 1]
allemagne=[81174000, 2]
UK=[64767115, 3]
italie=[60795612,4]
pologne=[38005614, 5]
belgique=[11258434,6]
grece=[10812467, 7]
reptcheque=[10538275, 8]
portugal=[10374822, 9]
hongrie=[9849000, 10]
suede=[9747355, 11]
autriche=[8584926, 12]
suisse=[8236573, 13]
danemark=[5659715, 14]
finlande=[5471553, 15]
slovaquie=[5421349, 16]
norvege=[5165802, 17]
irlande=[4625885,18]
paysbas=[16900726, 19]
europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas]

#autres pays importants
canada=[37590000,28]
USA=[328200000, 29]
argentine=[44490000, 30]
roumanie=[19410000,31]
kazakhstan=[18280000, 32]
russie=[144500000,33]
ukraine=[41980000,34]
chine=[1393000000, 35]
inde=[1353000000,36]
autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]

monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas,algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie,canada, USA, argentine, roumanie, kazakhstan,
russie, ukraine, chine, inde]

##liste avec les numéros des pays concernés
def listenum(H):
    listenum=[0]*(len(H))
    for i in range (len(H)):
        listenum[i]=H[i][1]
    return (listenum)

##creerseuildepart
def seuildep(T, sc, sd, S):
    for i in range (len(T)):
        if S[i]==0:
            if T[i]>=sc:
                S[i]=1
            else:
                S[i]=0
        else:
            if T[i]>sd:
                S[i]=1
            else:
                S[i]=0
    return S

##taux
def taux(C, L):
    T=[0]*(len(C))
    for i in range (len(C)):
            if (L[i][0]+L[i][1]+L[i][2])==0:
                T[i]=0
            else:
                T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

def tauxlocal(C, L, i):
    T=0
    if (L[i][0]+L[i][1]+L[i][2])==0:
        T=0
    else:
        T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

##creertableau
def creertableau(n, a):
    N=[0]*n
    for i in range (n):
        N[i]=[0]*a
    return N

##SIRevol
#Lt=[St, It, Rt, Dt] #D les dead
#d le taux de létalité différent du taux de mortalité
def SIRevol(Lt, beta): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
#modèle d'évolution de t à t+1
    gamma=0.3
    d=0.048
    N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
    if N==0:
```

```
        return ([0, 0, 0, 0])
    else:
        L1=[0,0,0,0]
        L1[0]= (-beta*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
        L1[1]= beta*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
        L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
        L1[3]= Lt[3]+d*Lt[1]
        Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
        return (Lt) #on renvoie la liste au temps t+1


##renvoyer une case
def case(Lt,n):
    return (Lt[n])

##creerLt pour le début seulement
def creerLt(C, Pop, i): #crée un Lt pour le pays en position i
    Lt=[0, 0, 0, 0]
    Lt[0]=Pop[i]-C[i]
    Lt[1]=C[i]
    return Lt

##creerL au depart
def creerL(C, Pop): #crée la liste des Lt-> cad on a L[0][0] qui donne le nombre de sains dans le pays 0
#L[0][1] le nombre de contaminés du pays 0
#L[0][2] le nombre de recovered du pays 1
    L=creertableau(len(C), len(C))
    for i in range (len(C)):
        L[i]=creerLt(C,Pop,i)
    return L

##verifS
def verifS(S):
    n=len(S)
    res=0
    for i in range (n):
        if S[i]==1:
            res=res+1
    if res==n:
        return False

##creerV V=voyageurs en prenant en compte les stratégies de confinement

def creerV(L, C, Pop):
    paspossible=[]
    for i in range (len(C)):
        if Pop[i]==0:
            paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
    V=[0]*(len(C))
    for i in range (len(C)):
            V[i]=[0]*2
    for i in range (len(C)):
        V[i][1]= rd.randint(0,(len(C)-1))
        while V[i][1] in paspossible :
            V[i][1]=rd.randint(0, len(C)-1)
        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

    return (V)
##creerD
def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
    D=[0]*(len(C))
    T=taux(C, L)
    for i in range (len(C)):
        D[i]=[0]*2
    for i in range (len(C)):
        D[i][0]=(V[i][0]*T[i])
        D[i][1]=V[i][1]
    return (D)

##trouver les contaminés
def recherche(D, i):
    c=0 #res
    for k in range (len(D)):
        if D[k][1]==i:
            c=c+D[k][0]
    return c

##expansion sans confinement
def expansionnonconfined(C, H, n):# d le taux de gens qui meurent
    Pop=[0]*len(H)
    beta=0.39
    gamma=0.3
    d=0.048
    for i in range (len(H)):
        Pop[i]=H[i][0]
    L=creerL(C, Pop)
    V=creerV(L,C,Pop)
    D=creerD(Pop,C,V,L)
    evolcontamines=creertableau(len(C), n) #evolution des contaminés
    evolmorts=creertableau(len(C),n)
    for k in range (n):
        V=creerV(L,C, Pop)
        D=creerD(Pop,C,V,L)
        for i in range (len(C)):
            L[i]=SIRevol(L[i], beta) #épidémie en interne
            C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine par les voyages
            Pop[i]=Pop[i]+recherche(V,i)-V[i][0] #on voyage en global
            L[i]=[Pop[i]-C[i]-L[i][2]-L[i][3],C[i],L[i][2],L[i][3]]
            evolcontamines[i][k]=C[i]
            evolmorts[i][k]=L[i][3]

    return (evolcontamines, evolmorts)

##tracer
def tracercontamines(C, H, n):
    evol=expansionnonconfined(C, H, n)[0]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de contaminés en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de contaminés")
        plt.legend(fontsize=6, loc='best')
    return(plt.show())


def tracermorts(C, H, n):
```

```
    evol=expansionnonconfined(C, H, n)[1]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de morts en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de morts")
        plt.legend(fontsize=6, loc='best')
    plt.show()

def tracermortsfrance(C, H, n):
    evol=expansionnonconfined(C, H, n)[1]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[liste[0]]))
    plt.legend()
    plt.title("Nombre de morts en fonction du temps")
    plt.xlabel("Temps en jours")
    plt.ylabel("Nombre de morts")
    plt.legend(fontsize=6, loc='best')
    plt.show()

def tracercontaminesfrance(C, H, n):
    evol=expansionnonconfined(C, H, n)[0]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[liste[0]]))
    plt.legend()
    plt.title("Nombre de contaminés en fonction du temps")
    plt.xlabel("Temps en jours")
    plt.ylabel("Nombre de contaminés")
    plt.legend(fontsize=6, loc='best')
    plt.show()


##application
#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,
365)
#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)
#tracercontamines([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10], europe, 365)
#tracermorts([116, 1486, 114,35, 1578, 0,1, 7, 3,0, 0,14, 14, 23, 4, 5, 0, 19, 1, 10], europe, 365)
#tracercontamines([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365)
#tracermorts([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365)

#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,
0,0,0,32616,0],monde, 365)
#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,
365)
```

stratégie de confinement épidémie.py

```
##data
NOMS_PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
"Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie","Norvège","Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israel", "Malte", "Maroc",
"Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]

AFFICHER = ["évolution du nombre total de morts", "évolution du nombre de contaminés"]
#données relatives aux pays (nombre hab)
#AFNMO où on a de la donnée:
algerie=[42230000, 20]
arabiesaoudite=[34813900, 21]
israel=[8655540, 22]
malte=[514564, 23]
maroc=[34660000, 24]
tunisie=[11180000,25]
egypte=[102334000, 26]
turquie=[84339100, 27]
AFNMO=[algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie]

#europe
france=[66352469,0]
espagne=[46439864, 1]
allemagne=[81174000, 2]
UK=[64767115, 3]
italie=[60795612,4]
pologne=[38005614, 5]
belgique=[11258434,6]
grece=[10812467, 7]
reptcheque=[10538275, 8]
portugal=[10374822, 9]
hongrie=[9849000, 10]
suede=[9747355, 11]
autriche=[8584926, 12]
suisse=[8236573, 13]
danemark=[5659715, 14]
finlande=[5471553, 15]
slovaquie=[5421349, 16]
norvege=[5165802, 17]
irlande=[4625885,18]
paysbas=[16900726, 19]
europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas]

#autres pays importants
canada=[37590000,28]
USA=[328200000, 29]
argentine=[44490000, 30]
roumanie=[19410000,31]
kazakhstan=[18280000, 32]
russie=[144500000,33]
ukraine=[41980000,34]
chine=[1393000000, 35]
inde=[1353000000,36]
autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]

monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas,algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie,canada, USA, argentine, roumanie, kazakhstan,
russie, ukraine, chine, inde]

import numpy as np
import matplotlib.pyplot as plt
import random as rd
from math import floor
from math import ceil

##liste avec les numéros des pays concernés
def listenum(H):
    listenum=[0]*(len(H))
    for i in range (len(H)):
        listenum[i]=H[i][1]
    return (listenum)

##creerseuildepart
def seuildep(T, sc, sd, S):
    for i in range (len(T)):
        if S[i]==0:
            if T[i]>=sc:
                S[i]=1
            else:
                S[i]=0
        else:
            if T[i]>sd:
                S[i]=1
            else:
                S[i]=0
    return S

##taux
def taux(C, L):
    T=[0]*(len(C))
    for i in range (len(C)):
            if (L[i][0]+L[i][1]+L[i][2])==0:
                T[i]=0
            else:
                T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

def tauxlocal(C, L, i):
    T=0
    if (L[i][0]+L[i][1]+L[i][2])==0:
        T=0
    else:
        T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

##creertableau
def creertableau(n, a):
    N=[0]*n
    for i in range (n):
        N[i]=[0]*a
    return N

##SIRevol
#Lt=[St, It, Rt, Dt] #Dt les dead à t
#d le taux de létalité =! taux de mortalité
def SIRevol(Lt, beta, i, k): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
#modèle d'évolution de t à t+1
    gamma=0.3
    d=0.048
    N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
```

```
    if N==0:
        return ([0, 0, 0, 0])
    else:
        L1=[0,0,0,0]
        L1[0]= (-beta[i][k]*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
        L1[1]= beta[i][k]*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
        L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
        L1[3]= Lt[3]+d*Lt[1]
        Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
        return (Lt) #on renvoie la liste au temps t+1


##renvoyer une case
def case(Lt,n):
    return (Lt[n])

##creerLt pour le début seulement
def creerLt(C, H, i): #crée un Lt pour le pays en position i
    Lt=[0, 0, 0, 0]
    Lt[0]=H[i][0]-C[i]
    Lt[1]=C[i]
    return Lt

##creerL au depart
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
#L[0][1] le nombre de contaminés du pays 0
#L[0][2] le nombre de recovered du pays 1
    L=creertableau(len(C), len(C))
    for i in range (len(C)):
        L[i]=creerLt(C,H,i)
    return L

##verifS
def verifS(S):
    n=len(S)
    res=0
    for i in range (n):
        if S[i]==1:
            res=res+1
    if res==n:
        return False

##creerV V=voyageurs en prenant en compte les stratégies de confinement

def creerV(L, C, S, Pop):
    paspossible=[]
    for i in range (len(C)):
        if Pop[i]==0:
            paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
    V=[0]*(len(C))
    for i in range (len(C)):
            V[i]=[0]*2
    if verifS(S)==False:
        for i in range (len(C)):
            V[i][0]=0
    else :
        for i in range (len(C)):
            if S[i]==0:
                V[i][1]= rd.randint(0,(len(C)-1))
                while S[V[i][1]]==1 :
                    V[i][1]=rd.randint(0, len(C)-1)
                    while V[i][1] in paspossible :
                        V[i][1]=rd.randint(0, len(C)-1)
                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
cohérent
            else:
                V[i][0]=0
    return (V)
##creerD
def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
    D=[0]*(len(C))
    T=taux(C, L)
    for i in range (len(C)):
        D[i]=[0]*2
    for i in range (len(C)):
        D[i][0]=(V[i][0]*T[i])
        D[i][1]=V[i][1]
    return (D)

##trouver les contaminés
def recherche(D, i):
    c=0 #res
    for k in range (len(D)):
        if D[k][1]==i:
            c=c+D[k][0]
    return c


##creer beta
def creertableaubeta(n,C):
    beta=creertableau(len(C), n+1)
    for i in range (len(C)):
        beta[i][0]=0.39
    return beta


##expansion avec confinement
def expansionconfined(C, H, n, sc, sd):# d le taux de gens qui meurent
    Pop=[0]*len(H)
    for i in range (len(H)):
        Pop[i]=H[i][0]
    L=creerL(C, H)
    beta=creertableaubeta(n, C)
    gamma=0.3
    d=0.048
    S=[0]*(len(C))
    T=taux(C,L)
    S=seuildep(T,sc, sd,S)
    V=creerV(L,C,S, Pop)
    D=creerD(H,C,V,L) #D le tableau des gens dangereux qui se déplacent et vers quel pays ils se déplacent
    evolcontamines=creertableau(len(C), n) #evolution des contaminés
    evolmorts=creertableau(len(C),n)
    listedeS=creertableau(len(C),n)
    for k in range (n):
        T=taux(C,L)
        S=seuildep(T,sc, sd ,S)
        V=creerV(L,C,S, Pop)
        D=creerD(H,C,V,L)
        for i in range (len(C)):
            L[i]=SIRevol(L[i], beta, i, k)
            C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine
```

```
            Pop[i]=Pop[i]+recherche(V,i)-V[i][0]
            L[i]=[Pop[i]-C[i]-L[i][2],C[i],L[i][2],L[i][3]]
            evolcontamines[i][k]=C[i]
            evolmorts[i][k]=L[i][3]
            listedeS[i][k]=S[i]
            if listedeS[i][k]==1:
                beta[i][k+1]=0.22
            else:
                beta[i][k+1]=0.39

    return ([evolmorts, evolcontamines]) #on renvoie la liste des états de chaque pays


##tracer des évolutions en fonction
def tracercontamines(C, H, n,sc, sd):
    evol=expansionconfined(C, H, n, sc, sd)[1]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de contaminés en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de contaminés")
        plt.legend(fontsize=6, loc='best')
    return(plt.show())

def tracermorts(C, H, n,sc, sd):
    evol=expansionconfined(C, H, n,sc, sd)[0]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de morts en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de morts")
        plt.legend(fontsize=6, loc='best')
    plt.show()


def tracermortsfrance(C, H, n,sc, sd):
    evol=expansionconfined(C, H, n,sc, sd)[0]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[liste[0]]))
    plt.legend()
    plt.title("Nombre de morts en fonction du temps")
    plt.xlabel("Temps en jours")
    plt.ylabel("Nombre de morts")
    plt.legend(fontsize=6, loc='best')
    plt.show()

def tracercontaminesfrance(C, H, n,sc, sd):
    evol=expansionconfined(C, H, n,sc, sd)[1]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    plt.plot(abscisse,evol[0], label=str(NOMS_PAYS[liste[0]]))
    plt.legend()
    plt.title("Nombre de contaminés en fonction du temps")
    plt.xlabel("Temps en jours")
    plt.ylabel("Nombre de contaminés")
    plt.legend(fontsize=6, loc='best')
    plt.show()

##application
#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,
365, 0.003, 0.0006)
#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,
0.003, 0.0006)
#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)
#tracermorts([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)
#tracercontamines([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365, 0.003, 0.0006)
#tracermorts([3, 0, 9, 0, 0, 0,1, 0], AFNMO, 365, 0.003, 0.0006)

#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,
365, 0.003, 0.0006)

#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,
0,0,0,32616,0],monde, 365, 0.003, 0.0006)
```

épidémie et flux.py

```
##data

NOM_PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
"Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie","Norvège","Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israel", "Malte", "Maroc",
"Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]

AFFICHER = ["évolution du nombre total de morts", "évolution du nombre de contaminés", "évolution des stocks en dollar usd", "évolution du nombre de
vivants", " évolution des exports", "évolution du taux de contaminés", " évolution des imports"]

AFFICHERBIS=["évolution du nombre total de morts en fonction du temps ", "évolution du nombre de contaminés en fonction du temps", "évolution des
stocks en dollar usd en fonction du temps", "évolution du nombre de vivants en fonction du temps", " évolution des exports en fonction du temps",
"évolution du taux de contaminés en fonction du temps", " évolution des imports en fonction du temps"]

#données relatives aux pays (nombre hab, production 2017, consommation par habitant, numéro du pays, quantité exportée par an)
#production en dollars usd DE CEREALES par an

#AFNMO où on a de la donnée:
algerie=[42230000,1217055450, 0.393750627, 20, 0]
arabiesaoudite=[34813900,591101519, 0.393750627, 21, 0]
israel=[8655540, 125803727, 0.393750627, 22, 0]
malte=[514564, 12295953, 0.393750627, 23, 0]
maroc=[34660000, 2350715863, 0.393750627, 24, 0]
tunisie=[11180000,354334516, 0.393750627, 25, 0]
egypte=[102334000, 4303123738, 0.393750627, 26,0]
turquie=[84339100, 8556108904, 0.393750627, 27, 185858000]
AFNMO=[algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie]

#europe
france=[66352469, 12366941872, 0.393750627, 0, 5602340000]
espagne=[46439864, 3547151265, 0.393750627, 1, 0]
allemagne=[81174000, 8131825487, 0.393750627, 2, 2512970000]
UK=[64767115, 4359410688, 0.393750627, 3, 438740000]
italie=[60795612, 4301291729, 0.393750627, 4, 0]
pologne=[38005614, 5429724123, 0.393750627, 5, 978450000]
belgique=[11258434, 460108034, 0.393750627, 6, 0]
grece=[10812467,877721695, 0.393750627, 7, 0]
reptcheque=[10538275, 1389583863, 0.393750627, 8, 0]
portugal=[10374822, 253498428, 0.393750627, 9, 0]
hongrie=[9849000, 2311067527, 0.393750627, 10, 1747190000]
suede=[9747355, 983369105, 0.393750627, 11, 0]
autriche=[8584926, 708800003, 0.393750627, 12, 0]
suisse=[8236573,400991999, 0.393750627, 13, 0]
danemark=[5659715, 1776259292, 0.393750627, 14, 0]
finlande=[5471553, 565659239, 0.393750627, 15, 0]
slovaquie=[5421349, 59431977, 0.393750627, 16, 0]
norvege=[5165802, 459500938, 0.393750627, 17, 0]
irlande=[4625885, 344572386, 0.393750627, 18, 0]
paysbas=[16900726, 252251731, 0.393750627, 19, 0]
europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas]

#autres pays importants
canada=[37590000,9711442097, 0.393750627, 28, 6305310000]
USA=[328200000,68607473570, 0.393750627, 29,18740200000]
argentine=[44490000,8846326657, 0.393750627, 30, 7511940000]
roumanie=[19410000,5391922117, 0.393750627,31, 2184270000]
kazakhstan=[18280000,2763295937, 0.393750627, 32, 852409000]
russie=[144500000,19082405580, 0.393750627, 33, 8142170000]
ukraine=[41980000,8391279873, 0.393750627, 34, 7947470000]
chine=[1393000000,2.62353E+11, 0.393750627, 35, 721030000]
inde=[1353000000,1.12689E+11, 0.393750627, 36, 7554560000]
autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]

monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas,algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie,canada, USA, argentine, roumanie, kazakhstan,
russie, ukraine, chine, inde]

##exports data
exportsfrance=[10.46140423199625, 20, 2.59628620601895, 21, 0.223940423377463 17, 22, 0.024384925299463987, 23,2.461897838345699, 24,
1.1658194663411847, 25, 0.7627146474383644, 26, 0.1740402455860384, 27, 0.0, 0, 11.60814377532809, 1, 6.408126274848944, 2, 2.909145701872827, 3,
5.703404947900281, 4, 0.6288688368175116, 5, 13.64848985498942, 6, 0.5503789165705348, 7, 0.207588356659343, 8, 3.038198925197493 2, 9,
0.463072444222083 16, 10, 0.11401233614984244, 11, 0.34679945368724713, 12, 0.8546931388491361, 13, 0.30463071389250035, 14,0.011785544860433152, 15,
0.05442148656141191, 16, 0.017798885524516053, 17, 0.7045103325393965, 18, 9.489925687618346, 19, 0.002637430115825833, 28, 0.05032216660995689, 29,
0.0005274860231651666, 30,0.40878659692377084, 31, 0.008319208136776342, 32, 0.3686524460755183, 33, 0.3207567151721212, 34, 0.6010929299405573, 35,
0.03018727155428082, 36] #export par personne
exportsespagne=[0]*74
exportsallemagne=[0.5889693744302363, 20,5.938859733412176, 21, 0.18519476679725036, 22, 0.0020942666371005494, 23, 0.5850025870352576,24, 0.0, 25,
0.0, 26, 0.2414812624732057, 27, 1.6679355458644394,0, 0.952509424199867, 1, 0.0, 2,0.7528642176066228, 3, 1.1632296055387192, 4,0.9158228004040703, 5,
3.1513292433537834, 6, 0.09580653903959396, 7,0.09580653903959396, 8, 0.7167565969399069, 9, 0.057629290166802176, 10, 0.12348781629585828,
11,0.5125655998226033,12, 0.6984009658264961, 13, 0.4261832606499618, 14,0.016212087614260722, 15,0.028346514893931554, 16, 0.25333234779609237, 17,
0.037832310838445 81, 18, 5.927218074753, 19,0.0035323956365338656, 28, 0.16778771527829 11, 29, 4.927686204942469e-05, 30, 0.01740705151895927, 31,
0.0002710227412718358, 32, 0.043437553896567865, 33, 0.037684480252297536, 34, 0.009313326927341267, 35, 0.0, 36]
exportsUK=[0.06600571910606177, 20, 0.008507403795892406, 21, 0.35135114479006824, 22, 0.014096660010253661, 23, 0.0, 24,0.0702825809054487, 25,
1.5439934293815618e-05, 26, 0.0035511848875775924, 27, 0.2991332870083838, 0, 1.0207494960984445, 1, 0.4161216691526248, 2, 0.0, 3, 0.16358610384297648,
4, 0.014266499287485633, 5,0.8785785811209902, 6, 0.06625275805476283, 7, 0.011965949077707105, 8, 0.4695129619406392, 9,0.0022542304068970806,10,
0.04000486975527627,11, 0.010900593611433827, 12, 0.005681895820124148, 13,0.02011823438484175, 14,0.006052454243175723, 15, 0.00072567691180933 41, 16,
0.021183589851115 03, 17,1.469295027268082, 18, 1.0202554182010424, 19, 0.10130140890172427, 28, 0.19211910241794775, 29, 0.00010807954005670933, 30,
0.0002933587515824968, 31, 0.0, 32,0.0, 33, 0.0, 34, 0.005975254571706645, 35,7.71996714690781e-05, 36]
exportsitalie=[0]*74
exportspologne=[0.15589802075030285,20,2.5434926534800884, 21, 0.0, 22, 0.05988588948990536, 23, 5.2623804472676064e-05, 24, 0.0,
25,1.4206585374465994, 26, 0.2910622625383713, 27, 0.10569491128336987, 0, 1.547192475301149, 1, 9.243608062745677, 2, 0.9485966994244587, 3,
0.3341611584014 93, 4, 0.0, 5, 0.06625336983109917, 6, 0.31953174075808904, 7, 0.5947805500524213, 8, 0.3518953805087848, 9, 0.12716542350822171, 10,
0.18860371523007102, 11, 0.06609549841768113, 12, 0.026048783213974653, 13, 0.4116760223897448, 14, 0.08803962488278705, 15, 0.09448604093068987,
16,0.3983095760536851, 17, 0.03773126780690874, 18, 1.0172444523590647, 19, 0.0030521806594152115, 28, 0.007577827844065353, 29, 0.0, 30,
0.05717576355956254, 31, 0.0, 32, 0.028864156753262822, 33, 0.007946194475374086,34, 0.0, 35, 0.0, 36]
exportsbelgique=[0]*74
exportsgrece=[0]*74
exportsreptcheque=[0]*74
exportsportugal=[0]*74
exportshongrie=[0.030663011473246016, 20, 2.659660879277084, 21, 0.032389075032998274, 22, 0.07533759772565743, 23, 1.4272514976139 71,24, 0.0, 25,
0.0, 26, 3.052695705147731, 0,0.6209767489085186, 1, 11.790537110366534, 2, 0.084374048126713 37, 3, 57.27058584627881, 4, 4.7326632145395475, 5,
0.7912478424205504, 6, 3.1041730124885776, 7, 1.598334856330592, 8, 0.02832774901005178, 9, 0.0, 10, 0.0,11,1.6961315869631435, 12, 1.7761194029850746,
13, 0.01015331505736623, 14, 0.0018275967103259215,15,5.084069448674993, 16, 0.0031475276677835314, 17, 0.013808508478018074, 18,2.8561275256371204, 19,
0.0005076657528683116, 28, 0.007107320540156361, 29, 0.0, 30, 28.19768504416692, 31,0.04457305310183775, 32,8.774291806274748, 33, 5.544623819677125, 34,
0.005787389582698751, 35, 0.0, 36]
exportssuede=[0]*74
exportsautriche=[0]*74
exportssuisse=[0]*74
exportsdanemark=[0]*74
exportsfinlande=[0]*74
exportsslovaquie=[0]*74
exportsnorvege=[0]*74
exportsirlande=[0]*74
exportspaysbas=[0]*74
exportsalgerie=[0]*74
exportsarabiesaoudite=[0]*74
exportsisrael=[0]*74
exportsmalte=[0]*74
exportsmaroc=[0]*74
```

```
exportstunisie=[0]*74
exportsegypte=[0]*74
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,
0.03414786261650883, 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,
4.742758696737338e-05, 18, 0.0194215968631394, 19, 0.04470050071674941, 28, 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]
exportscanada=[8.608273476988561, 20, 0.9110667730779463, 21, 0.006517690875232775, 22, 0.014285714285714285, 23, 5.052274541101356, 24,
0.5281989890928438, 25, 0.047326416600015962, 26, 0.95312583133812l8, 27, 0.281511040170258, 0, 1.1488427773343974, 1, 0.2590316573556797, 2,
3.086778398510242, 3, 5.02487363660548, 4, 0.0, 5, 1.3740888534184623, 6, 0.012396914072891726, 7, 0.0, 8, 0.7461558925246076, 9, 0.0, 10,
0.0033785812716l1479, 11, 0.005826017557861133, 12, 0.4795424314977388, 13, 0.0027400904495876563, 14, 2.6602819898909284e-05, 15, 0.0, 16,
0.0003192338387869114, 17, 1.7341846235700984, 18, 0.6632349028997073, 19, 0.0, 28, 28.00053205639798, 29, 0.0002394253790918357, 30, 0.0, 31, 0.0, 32,
0.0022346368715083797, 33, 0.0, 34, 11.695956371375365, 35, 0.0031391327480712957, 36]
exportsUSA=[0.5279128580134065, 20, 1.1996739792809263, 21, 0.20313223644119438, 22, 0.0007099329677026203, 23, 0.5454478976234004, 24,
0.009926873857404022, 25, 0.19794028031687996, 26, 0.10457647775746497, 27, 0.0769926873857404, 0, 0.1509354052407069,1, 0.033769043266301035, 2,
0.185767824497257776, 3, 0.4285588056063376, 4, 0.004722730042656917, 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, 0.005551492992078001,
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, 0.010645947592931l4, 34, 4.107876294942108, 35,
0.015426569165143206, 36]
exportsargentine=[19.821532928748034, 20, 8.547336480107889, 21, 0.022454484153742415, 22, 0.0, 23, 5.671813890761969, 24, 1.1792312879298719, 25,
11.572285906945382, 26, 0.059451562148797l48, 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,
0.0010114632501685772, 18, 0.15014610024724656, 19, 0.04236907170150596, 28, 2.0945830523713194, 29, 0.0, 30, 0.008676106990334907, 31, 0.0, 32,
0.022252191503708697, 33, 0.0005169701056417172, 34, 0.004495392222971455, 35, 0.8626432906271072, 36]
exportsroumanie=[0.119629057187017, 20, 5.6155074703760945, 21, 0.00087583771973209686, 22, 0.37980422462648117, 23, 3.6187017001545594, 24,
0.9254507985574446, 25, 12.294178258629572, 26, 3.8095311695002576, 27, 1.6410097887686759, 0, 1.3582689335394127, 1, 0.767284904688305, 2,
2.2951056156620298, 3, 7.916486347243689, 4, 0.2605358062854199, 5, 0.9037609479649665, 6, 2.3135497166409067, 7, 0.18428645028335591, 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, 0.00844925296239052, 32, 2.060226687274601, 33, 2.2080886141164346, 34, 0.0, 35, 0.0, 36]
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, 1, 0.030251641137855558, 2, 0.2799781181619256, 3, 2.938074398249453, 4, 0.05300875273522976, 5, 0.38938730853391684, 6, 0.0, 7,
0.0007658643326039387, 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,
1.886159737417943, 33, 0.2400437636761488, 34, 3.09753829321663, 35, 0.0, 36]
exportsrussie=[0.11103806228373703, 20, 1.9170795847750866, 21, 0.0, 22, 0.5084152249134948, 23, 0.39909342560553634, 24, 10.573314878892733, 25,
5.25235294117647, 26, 0.020851211072664436, 0, 0.06303114186851211, 1, 0.1391833910034602, 2, 0.32438754325259517, 3, 0.2810242214532872, 4,
0.013307958477508655, 5, 0.05903114186851211, 6, 0.30187543252595156, 7, 0.0002837370242214533, 8, 0.0, 9, 0.002304498269896194, 10, 0.016138408304498827,
11, 0.0031903114186851212, 12, 0.015363321799307696, 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, 0.062809688581314487, 34, 0.04119031141868512, 35, 0.6239861591695501, 36]
exportsukraine=[1.398808956646022, 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, 0.7002382086707957, 5, 2.1504287756074323, 6, 0.7276560266793711, 7, 0.021676989042401142, 8,
3.0632682229633157, 9, 0.05452596474511672, 10, 0.027489280609814197, 11, 0.1316579323487375, 12, 0.07532158170557408, 13, 0.06476893758932825, 14,
0.0033587422582181l99, 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]
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, 0.0001385489231873655, 6, 2.871500358937545e-05, 7,
5.025125628140703 6e-05, 8, 1.7946877243359657e-05, 9, 7.178750897343862e-07, 10, 9.117013639626705e-05, 11, 4.307250538406317 5e-05,
12,2.4407753050969132e-05, 13, 1.8664752333094043e-05, 14, 2.871500358937545e-06, 15, 0.0002957645369705671 4, 16, 2.081837760229 72e-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]
exportinde=[0.017706577974870658, 20, 0.5929563932002956, 21, 0.01785957132298596, 22, 0.0002062084257206 2085, 23, 0.0012483370288248338, 24,
0.007502586844050259, 25, 0.03042867701404287, 26, 0.006866962305986696, 27, 0.045895787 13968958, 0, 0.003478935698447893 5, 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, 0.0001441241685144 1242, 10, 0.005830746489283074 5, 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, 0.0001167775314 1167775, 31, 0.0005018477457501847, 32, 0.019248337028824832,
33, 0.008032520325203253, 34, 5.764966740576497e-05, 35, 0.0, 36]


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

import numpy as np
import matplotlib.pyplot as plt
import random as rd
from math import floor
from math import ceil

##liste avec les numéros des pays concernés
def listenum(H):
    listenum=[0]*(len(H))
    for i in range (len(H)):
        listenum[i]=H[i][1]
    return (listenum)

##creerseuildepart
def seuildep(T, sc, sd, S):
    for i in range (len(T)):
        if S[i]==0:
            if T[i]>=sc:
                S[i]=1
            else:
                S[i]=0
        else:
            if T[i]>sd:
                S[i]=1
            else:
                S[i]=0
    return S

##taux
def taux(C, L):
    T=[0]*(len(C))
    for i in range (len(C)):
            if (L[i][0]+L[i][1]+L[i][2])==0:
                T[i]=0
            else:
                T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

def tauxlocal(C, L, i):
    T=0
    if (L[i][0]+L[i][1]+L[i][2])==0:
        T=0
    else:
        T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

##creertableau
def creertableau(n, a):
```

```
    N=[0]*n
    for i in range (n):
        N[i]=[0]*a
    return N

##SIRevol
#Lt=[St, It, Rt, Dt] #D les dead
#d le taux de létalité =! taux de mortalité
def SIRevol(Lt, beta, i, k): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
#modèle d'évolution de t à t+1
    gamma=0.3
    d=0.048
    N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
    if N==0:
        return ([0, 0, 0, 0])
    else:
        L1=[0,0,0,0]
        L1[0]= (-beta[i][k]*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
        L1[1]= beta[i][k]*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
        L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
        L1[3]= Lt[3]+d*Lt[1]
        Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
        return (Lt) #on renvoie la liste au temps t+1


##renvoyer une case
def case(Lt,n):
    return (Lt[n])

##creerLt pour le début seulement
def creerLt(C, H, i): #crée un Lt pour le pays en position i
    Lt=[0, 0, 0, 0]
    Lt[0]=H[i][0]-C[i]
    Lt[1]=C[i]
    return Lt

##creerL au depart
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
#L[0][1] le nombre de contaminés du pays 0
#L[0][2] le nombre de recovered du pays 1
    L=creertableau(len(C), len(C))
    for i in range (len(C)):
        L[i]=creerLt(C,H,i)
    return L

##verifS
def verifS(S):
    n=len(S)
    res=0
    for i in range (n):
        if S[i]==1:
            res=res+1
    if res==n:
        return False

##creerV V=voyageurs en prenant en compte les stratégies de confinement

def creerV(L, C, S, Pop):
    paspossible=[]
    for i in range (len(C)):
        if Pop[i]==0:
            paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
    V=[0]*(len(C))
    for i in range (len(C)):
            V[i]=[0]*2
    if verifS(S)==False:
        for i in range (len(C)):
            V[i][0]=0
    else :
        for i in range (len(C)):
            if S[i]==0:
                V[i][1]= rd.randint(0,(len(C)-1))
                while S[V[i][1]]==1 :
                    V[i][1]=rd.randint(0, len(C)-1)
                    while V[i][1] in paspossible :
                        V[i][1]=rd.randint(0, len(C)-1)
                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
cohérent
            else:
                V[i][0]=0
    return (V)

##creerD
def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
    D=[0]*(len(C))
    T=taux(C, L)
    for i in range (len(C)):
        D[i]=[0]*2
    for i in range (len(C)):
        D[i][0]=(V[i][0]*T[i])
        D[i][1]=V[i][1]
    return (D)

##creerE
def creerE(H, S, E, exports, Pop):
    paspossible=[]
    for i in range (len(H)):
            if Pop[i]==0:
                paspossible.append(i)
    if verifS(S)==False:
        for i in range (len(E)):
            for k in range (len(E[i])):
                E[i][k]=0
    else:
        for i in range (len(E)):
            if S[i]==0:
                for k in range (len(E[i])):
                    if k%2==0:
                        E[i][k]=exports[i][k]*Pop[i]/365
                    else:
                        if k in paspossible:
                            E[i][k-1]=0
                        else:
                            E[i][k]=exports[i][k]

            else:
                for k in range (len(E[i])):
                    E[i][k]=0
    return E

##trouver les contaminés
```

```
def recherche(D, i):
    c=0 #res
    for k in range (len(D)):
        if D[k][1]==i:
            c=c+D[k][0]
    return c

def recherchebis(E,H,i):
    c=0
    for k in range (len(E)):
        for a in range (len(E[i])):
            if a%2==1:
                if E[k][a]==H[i][3]:
                    c=c+abs(E[k][a-1])
    return c

##creer beta
def creertableaubeta(n,C):
    beta=creertableau(len(C), n+1)
    for i in range (len(C)):
        beta[i][0]=0.39
    return beta

##creerZ
def creerZ(H, prop):
    Z=[0]*len(H)
    for i in range (len(Z)):
        Z[i]=H[i][1]*prop
    return Z

##faire manger les gens
def manger(Pop, Z, i, L):
    consoparhab=0.393750627
    viv=Pop[i] # nos vivants avant de voir qui aura assez à manger
    ts=0
    tc=0
    tr=0
    survivants=0
    capacité=0
    nouveauxmorts=0
    if Pop[i]!=0:
        ts=L[i][0]/viv
        tc=L[i][1]/viv
        tr=L[i][2]/viv
        capacité=Z[i]//consoparhab
        if capacité<viv:
            survivants=capacité
            nouveauxmorts=viv-survivants
            Pop[i]=survivants
            L[i][0]=ts*Pop[i]
            L[i][1]=tc*Pop[i]
            L[i][2]=tr*Pop[i]
            L[i][3]=L[i][3]+nouveauxmorts
            Z[i]=0
        else:
            Z[i]=Z[i]-Pop[i]*consoparhab

    return (Z[i], L[i])


##expansion avec confinement
def expansionconfined(C, H, n, sc, sd, exports, prop):# d le taux de gens qui meurent
    Pop=[0]*len(H)
    for i in range (len(H)):
        Pop[i]=H[i][0]
    L=creerL(C, H)
    beta=creertableaubeta(n, C)
    gamma=0.2
    d=0.048
    S=[0]*(len(C))
    T=taux(C,L)
    evoltaux=creertableau(len(C), n)
    Z=creerZ(H, prop)
    E=creertableau(len(C), len(exports))
    S=seuildep(T,sc, sd,S)
    V=creerV(L,C,S, Pop)
    D=creerD(H,C,V,L) #D le tableau des gens dangereux qui se déplacent et vers quel pays ils se déplacent
    evolcontamines=creertableau(len(C), n) #evolution des contaminés
    evolmorts=creertableau(len(C),n)
    evoldeZ=creertableau(len(C), n)
    evoldeE=creertableau(len(C), n)
    evolimports=creertableau(len(C),n)
    evoldeH=creertableau(len(C), n)
    listedeS=creertableau(len(C),n)
    prodpartravailleurparjour=[]
    for i in range (len(C)):
        evoldeZ[i][0]=Z[i]
        evolmorts[i][0]=0
        evoldeH[i][0]=Pop[i]
        evoldeE[i][0]=0
        evoltaux[i][0]=T[i]
        evolimports[1][0]=0
        listedeS[i][0]=S[i]
    for i in range (len(H)):
        prodpartravailleurparjour.append(H[i][1]/H[i][0]/365)
    for k in range (1, n):
        T=taux(C,L)
        S=seuildep(T,sc, sd ,S)
        V=creerV(L,C,S, Pop)
        D=creerD(H,C,V,L)
        E=creerE(H, S, E, exports, Pop)
        for i in range (len(C)):
            #partie épidémiologique
            L[i]=SIRevol(L[i], beta, i, k)
            C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine
            Pop[i]=Pop[i]+recherche(V,i)-V[i][0]
            L[i]=[Pop[i]-C[i]-L[i][2],C[i],L[i][2],L[i][3]]
            evolcontamines[i][k]=C[i]
            evolmorts[i][k]=L[i][3]
            listedeS[i][k]=S[i]
            if listedeS[i][k]==1:
                beta[i][k+1]=0.22
            else:
                beta[i][k+1]=0.39
            evoltaux[i][k]=tauxlocal(C,L, i)
            #partie échanges de céréales
            travailleurs=(L[i][0]+L[i][2])
            Z[i]=Z[i]+prodpartravailleurparjour[i]*travailleurs #on produit de la nourriture
            Z[i]=Z[i]+recherchebis(E,H,i)-E[i][0] #on importe et exporte
            Z[i]=manger(Pop, Z, i, L)[0]
            L[i]=manger(Pop, Z, i, L)[1]
```

```
            evolmorts[i][k]=L[i][3]
            evolcontamines[i][k]=L[i][1]
            evolmorts[i][k]=L[i][3]
            evoldeZ[i][k]=Z[i]
            evoldeH[i][k]=Pop[i]
            evoldeE[i][k]=somme3(E, i)
            evolimports[i][k]=recherchebis(E,H,i)


    return ([evolmorts, evolcontamines, evoldeZ, evoldeH, evoldeE, evoltaux, evolimports]) #on renvoie la liste des états de chaque pays


##tracer des évolutions en fonction
def tracercontamines(C, H, n,sc, sd):
    evol=expansionconfined(C, H, n, sc, sd)[1]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de contaminés en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de contaminés")
        plt.legend(fontsize=6, loc='best')
    return(plt.show())

def tracermorts(C, H, n, sc, sd, exports, prop):
    evol=expansionconfined(C, H, n, sc, sd, exports, prop)[0]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOM_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de morts en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de morts")
        plt.legend(fontsize=6, loc='best')
    plt.show()

def touttracer(C, H, n, sc, sd, exports, prop):
    RES=expansionconfined(C, H, n, sc, sd, exports, prop)
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for a in range (len(RES)):
        for k in range (len(liste)):
            plt.plot(abscisse, RES[a][k], label=str(NOM_PAYS[k]))
            plt.legend()
            plt.xlabel("temps en jour")
            plt.title(str(AFFICHERBIS[a]))
            plt.ylabel(str(AFFICHER[a]))
            plt.legend(fontsize=6, loc='best')
        plt.show()

def tracerfrance(C, H, n, sc, sd, exports, prop):
    RES=expansionconfined(C, H, n, sc, sd, exports, prop)
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for a in range (len(RES)):
        for k in range (1):
            plt.plot(abscisse, RES[a][k], label=str(NOM_PAYS[k]))
            plt.legend()
            plt.xlabel("temps en jour")
            plt.title(str(AFFICHERBIS[a]))
            plt.ylabel(str(AFFICHER[a]))
            plt.legend(fontsize=6, loc='best')
        plt.show()

##somme
def somme3(E, i):
    s=0
    for k in range (len(E)):
        s=s+E[i][k]
    return (s)

##application A FAIRE
#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,
0.003, 0.0006, exports, 30)

#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,
0.003, 0.0006, exports, 1)

#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,
0.003, 0.0006, exports, 2)

#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,
0.003, 0.0006, exports, 1)
```

```
# calcul stocks.py

##data

NOM_PAYS = ["France", "Espagne", "Allemagne", "UK", "Italie", "Pologne", "Belgique", "Grèce", "Reptchèque", "Portugal", "Hongrie", "Suède",
"Autriche", "Suisse", "Danemark", "Finlande", "Slovaquie","Norvège","Irlande", "Pays-Bas", "Algérie", "Arabie Saoudite", "Israel", "Malte", "Maroc",
"Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]

AFFICHER = ["évolution du nombre total de morts", "évolution du nombre de contaminés", "évolution des stocks en dollar usd", "évolution du nombre de
vivants", " évolution des exports", "évolution du taux de contaminés", " évolution des imports"]

#données relatives aux pays (nombre hab, production 2017, consommation par habitant, numéro du pays, quantité exportée par an)
#production en dollars usd DE CEREALES par an

#AFNMO où on a de la donnée:
algerie=[42230000,1217055450, 0.393750627, 20, 0]
arabiesaoudite=[34813900,591101519, 0.393750627, 21, 0]
israel=[8655540, 125803727, 0.393750627, 22, 0]
malte=[514564, 12295953, 0.393750627, 23, 0]
maroc=[34660000, 2350715863, 0.393750627, 24, 0]
tunisie=[11180000,354334516, 0.393750627, 25, 0]
egypte=[102334000, 4303123738, 0.393750627, 26,0]
turquie=[84339100, 8556108904, 0.393750627, 27, 185858000]
AFNMO=[algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie]

#europe
france=[66352469, 12366941872, 0.393750627, 0, 5602340000]
espagne=[46439864, 3547151265, 0.393750627, 1, 0]
allemagne=[81174000, 8131825487, 0.393750627, 2, 2512970000]
UK=[64767115, 4359410688, 0.393750627, 3, 438740000]
italie=[60795612, 4301291729, 0.393750627, 4, 0]
pologne=[38005614, 5429724123, 0.393750627, 5, 978450000]
belgique=[11258434, 460108034, 0.393750627, 6, 0]
grece=[10812467,877721695, 0.393750627, 7, 0]
reptcheque=[10538275, 1389583863, 0.393750627, 8, 0]
portugal=[10374822, 253498428, 0.393750627, 9, 0]
hongrie=[9849000, 2311067527, 0.393750627, 10, 1747190000]
suede=[9747355, 983369105, 0.393750627, 11, 0]
autriche=[8584926, 708800003, 0.393750627, 12, 0]
suisse=[8236573,400991999, 0.393750627, 13, 0]
danemark=[5659715, 1776259292, 0.393750627, 14, 0]
finlande=[5471553, 565659239, 0.393750627, 15, 0]
slovaquie=[5421349, 59431977, 0.393750627, 16, 0]
norvege=[5165802, 459500938, 0.393750627, 17, 0]
irlande=[4625885, 344572386, 0.393750627, 18, 0]
paysbas=[16900726, 252251731, 0.393750627, 19, 0]
europe=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas]

#autres pays importants
canada=[37590000,9711442097, 0.393750627, 28, 6305310000]
USA=[328200000,68607473570, 0.393750627, 29,18740200000]
argentine=[44490000,8846326657, 0.393750627, 30, 7511940000]
roumanie=[19410000,5391922117, 0.393750627,31, 2184270000]
kazakhstan=[18280000,2763295937, 0.393750627, 32, 852409000]
russie=[144500000,19082405580, 0.393750627, 33, 8142170000]
ukraine=[41980000,8391279873, 0.393750627, 34, 7947470000]
chine=[1393000000,2.62353E+11, 0.393750627, 35, 721030000]
inde=[1353000000,1.12689E+11, 0.393750627, 36, 7554560000]
autres=[canada, USA, argentine, roumanie, kazakhstan, russie, ukraine, chine, inde]

monde=[france, espagne, allemagne, UK, italie, pologne, belgique, grece, reptcheque, portugal, hongrie, suede, autriche, suisse, danemark, finlande,
slovaquie, norvege, irlande, paysbas,algerie, arabiesaoudite, israel, malte, maroc, tunisie, egypte, turquie,canada, USA, argentine, roumanie, kazakhstan,
russie, ukraine, chine, inde]


##exports
exportsfrance=[10.46140423199625, 20, 2.59628620601895, 21, 0.223940423377463 17, 22, 0.024384925299463987, 23,2.461897838345699, 24,
1.1658194663411847, 25, 0.7627146474383644, 26, 0.1740402455860384, 27, 0.0, 0, 11.608143775323809, 1, 6.408126274848944, 2, 2.909145701872827, 3,
5.703404947900281, 4, 0.6288688368175116, 5, 13.648489854989342, 6, 0.5503789165705348, 7, 0.207588356659343, 8, 3.0381989251974932, 9,
0.46307244422208316, 10, 0.11401233614984244, 11, 0.34679945368724713, 12, 0.8546931388491361, 13, 0.30463071389250035, 14,0.011785544860433152, 15,
0.05442148656141191, 16, 0.017798885524516053, 17, 0.7045103325393965, 18, 9.489925687618346, 19, 0.002637430115825833, 28, 0.05032216660995689, 29,
0.0005274860231651666, 30,0.40878659692377084, 31, 0.008319208136776342, 32, 0.3686524460755183, 33, 0.3207567151721212, 34, 0.6010929299405573, 35,
0.03018727155428082, 36] #export par personne
exportsespagne=[0]*74
exportsallemagne=[0.5889693744302363, 20, 5.938859733412176, 21, 0.18519476679725036, 22, 0.0020942666371005494, 23, 0.5850025870352576, 24, 0.0, 25,
0.0, 26, 0.2414812624732057, 27, 1.6679355458644394, 0, 0.952509424199867, 1, 0.0, 2, 0.7528642176066228, 3, 1.1632296055387192, 4, 0.9158228004040703, 5,
3.1513292433537834, 6, 0.09580653903959396, 7, 0.09580653903959396, 8, 0.7167565969399069, 9, 0.057629290166802176, 10, 0.12348781629585828, 11,
0.5125655998226033, 12, 0.6984009658264961, 13, 0.4261832606499618, 14, 0.016212087614260722, 15, 0.028346514893931554, 16, 0.25333234779609237, 17,
0.03783231083844581, 18, 5.927218074753, 19, 0.0035232956365338656, 28, 0.1677877152782911, 29, 4.927686204942469e-05, 30, 0.01740705151895927, 31,
0.0002710227412718358, 32, 0.043437553896567865, 33, 0.037684480252297536, 34, 0.009313326927341267, 35, 0.0, 36]

exportsUK=[0.066005719106061 77, 20, 0.008507403795892406, 21, 0.35135114479006824, 22, 0.014096660010253661, 23, 0.0, 24,0.0702825809054487, 25,
1.5439934293815618e-05, 26, 0.0035511848875775924, 27, 0.2991332870083838, 0, 1.0207494960984445, 1, 0.4161216691526248, 2, 0.0, 3, 0.16358610384297648,
4, 0.014266499287485633, 5,0.8785785811209902, 6, 0.06625275805476283, 7, 0.011965949077707105, 8, 0.4695129619406392, 9,0.0022542304068970806,10,
0.04000486975527627,11, 0.010900593611433827, 12, 0.005681895820124148, 13,0.02011823438484175, 14,0.006052454243175723, 15, 0.00072567691180933 41, 16,
0.021183589851115 03, 17,1.469295027268082, 18, 1.0202554182010424, 19, 0.10130140890172427, 28, 0.19211910241794775, 29, 0.00010807954005670933, 30,
0.0002933587515824968, 31, 0.0, 32,0.0, 33, 0.0, 34, 0.005975254571706645, 35,7.71996714690781e-05, 36]
exportsitalie=[0]*74
exportspologne=[0.15589802075030285,20,2.5434926534800884, 21, 0.0, 22, 0.05988588948990536, 23, 5.2623804472676064e-05, 24, 0.0,
25,1.4206585374465994, 26, 0.2910622625383713, 27, 0.10569491128336987, 0, 1.547192475301149, 1, 9.243608062745677, 2, 0.9485966994244587, 3,
0.334161158401493, 4, 0.0, 5, 0.06625336983109917, 6, 0.31953174075808904, 7, 0.5947805500524213, 8, 0.3518953805087848, 9, 0.12716542350822171, 10,
0.18860371523007102, 11, 0.06609549841768113, 12, 0.026048783213974653, 13, 0.4116760223897448, 14, 0.08803962488278705, 15, 0.09448604093068987,
16,0.3983095760536851, 17, 0.03773126780690874, 18, 1.0172444523590647, 19, 0.0030521806594152115, 28, 0.007577827844065353, 29, 0.0, 30,
0.05717576355956254, 31, 0.0, 32, 0.028864156753262822, 33, 0.007946194475374086,34, 0.0, 35, 0.0, 36]
exportsbelgique=[0]*74
exportsgrece=[0]*74
exportsreptcheque=[0]*74
exportsportugal=[0]*74
exportshongrie=[0.030663011473246016, 20, 2.659660879277084, 21, 0.032389075032998274, 22, 0.07533759772565743, 23, 1.427251497613971,24, 0.0, 25,
0.0, 26, 3.0526957051477 31, 0,0.6209767489085186, 1, 11.790537110366534, 2, 0.084374048126717 37, 3, 57.270585846278881, 4, 4.7326632145395475, 5,
0.7912478424205504, 6, 3.1041730124885776, 7, 1.598334856330592, 8, 0.02832774901005178, 9, 0.0, 10, 0.0,11,1.6961315869631435, 12, 1.7761194029850746,
13, 0.010153331505736623, 14, 0.0018275967103259215,15,5.084069448674993, 16, 0.0031475276677835314, 17, 0.013808508478018074, 18,2.8561275256371204, 19,
0.0005076657528683116, 28, 0.007107320540156361, 29, 0.0, 30, 28.19768504416692, 31,0.04457305310183775, 32,8.774291806274748, 33, 5.544623819677125, 34,
0.005787389582698751, 35, 0.0, 36]
exportssuede=[0]*74
exportsautriche=[0]*74
exportssuisse=[0]*74
exportsdanemark=[0]*74
exportsfinlande=[0]*74
exportsslovaquie=[0]*74
exportsnorvege=[0]*74
exportsirlande=[0]*74
exportspaysbas=[0]*74
exportsalgerie=[0]*74
exportsarabiesaoudite=[0]*74
exportsisrael=[0]*74
exportsmalte=[0]*74
exportsmaroc=[0]*74
exportstunisie=[0]*74
exportsegypte=[0]*74
```

```
088| exportsturquie=[8.299827719290 34e-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,
0.03414786261650883, 6, 0.010517067410015046, 7, 0.0006995569077687573, 8, 0.0, 9, 0.0145958398892091 57, 10, 0.0025255190060126 32, 11,
0.005821736300245082, 12, 0.008157544958388 22, 13, 0.000213424141353180 2, 14,0.0004031344892226737, 15, 0.0, 16, 0.002715229353882126, 17,
4.742758696737338e-05, 18, 0.0194215968631394, 19, 0.04470050071674941, 28, 0.6195702823482822, 29, 0.0, 30, 0.009604086360893108, 31,
0.006758431142850707, 32, 0.003865348337840 9304, 33, 0.017334783036574968, 34, 1.1856896741843345e-05, 35, 0.000782555184961660 7, 36]
089| exportscanada=[8.608273476988561, 20, 0.9110667730779463, 21, 0.006517690875232775, 22, 0.014285714285714285, 23, 5.052274541101356, 24,
0.5281989890928438, 25, 0.04732641660015962, 26, 0.9531258313381218, 27, 0.281511040170258, 0, 1.1488427773343974, 1, 0.2590316573556797, 2,
3.086778398510242, 3, 5.024873636606548, 4, 0.0, 5, 1.37408885341846 23, 6, 0.012396914072891726, 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,
0.0003192338387869114, 17, 1.7341846235700984, 18, 0.6632349028997073, 19, 0.0, 28, 28.00053205639798, 29, 0.0002394253790901835 7, 30, 0.0, 31, 0.0, 32,
0.0022346368715083797, 33, 0.0, 34, 11.695956371375365, 35, 0.0031391327480712957, 36]
090| exportsUSA=[0.5279128580134065, 20, 1.1996739792809263, 21, 0.20313223644119438, 22, 0.0007099329677026203, 23, 0.5454478976234004, 24,
0.009926873857404022, 25, 0.19794028031687996, 26, 0.10457647775746497, 27, 0.0769926873857404, 0, 0.1509354052407069,1, 0.033769043266301035, 2,
0.185767824497257 76, 3, 0.4285588056063376, 4, 0.004722730042656917, 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, 0.005551492992078001,
14, 0.0017032297379646556, 15, 0.0, 16, 0.006733698964046313, 17, 0.05971358927483242, 18, 0.12470444850700792, 19, 1.5059689213893968, 28, 0.0, 29,
0.0057306520414381 47, 30, 0.0005971968312004875, 31, 0.0006703229737964656, 32, 0.010923217550274223, 33, 0.010645947592931 14, 34, 4.107876294942108, 35,
0.015426569165143206, 36]
091| exportsargentine=[19.821532928748034, 20, 8.547336480107889, 21, 0.022454484153742415, 22, 0.0, 23, 5.671813890761969, 24, 1.1792312879298719, 25,
11.572285906945382, 26, 0.059451562148797 48, 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,
0.0010114632501685772, 18, 0.15014610024724656, 19, 0.04236907170150596, 28, 2.0945830523713194, 29, 0.0, 30, 0.008676106990334907, 31, 0.0, 32,
0.022252191503708697, 33, 0.0005169701056417172, 34, 0.004495392222971455, 35, 0.8626432906271072, 36]
092| exportsroumanie=[0.1196290571870 17, 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, 1.3582689335394127, 1, 0.767284904688305, 2,
2.2951056156620298, 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, 0.0084492529623905 2, 32, 2.060226687274601, 33, 2.2080886141164346, 34, 0.0, 35, 0.0, 36]
093| 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, 1, 0.030251641137855 58, 2, 0.2799781181619256, 3, 2.938074398249453, 4, 0.05300875273522976, 5, 0.38938730853391684, 6, 0.0, 7,
0.0007658643326039387, 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, 0.0061816192560 17505, 19, 0.225054704595186, 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]
094| exportsrussie=[0.11103806228373703, 20, 1.9170795847750866, 21, 0.0, 22, 0.5084152249134948, 23, 0.39909342560553634, 24, 10.573314878892733, 25,
5.25235294117647, 26, 0.020851211072664 36, 0, 0.06303114186851211, 1, 0.1391833910034602, 2, 0.32438754325259517, 3, 0.2810242214532872, 4,
0.013307958477508 65, 5, 0.059031141868512 11, 6, 0.30187543252595156, 7, 0.0002837370242214533, 8, 0.0, 9, 0.002304498269896194, 10, 0.01613840830449827,
11, 0.003190311418685121 2, 12, 0.015363321799307 96, 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, 0.0628096885813148 7, 34, 0.041190311418685 12, 35, 0.6239861591695501, 36]
095| exportsukraine=[1.398808956646022, 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, 0.7002382086707957, 5, 2.1504287756074323, 6, 0.7276560266793711, 7, 0.021676989042401142, 8,
3.0632682229633157, 9, 0.054525964745116 72, 10, 0.027489280609814197, 11, 0.1316579323487375, 12, 0.07532158170557408, 13, 0.06476893758932825, 14,
0.003358742258218199, 15, 0.001357789423535 0166, 16, 0.002000952834683 1827, 17, 0.7855645545497856, 18, 11.760576464983325, 19, 0.012148642210576465, 28,
0.003477846593616 0076, 29, 0.0, 30, 0.092996665078 60886, 31, 0.0031205535874225824, 32, 0.49187708432586946, 33, 0.0, 34, 12.348451643639828, 35,
12.067651262505954, 36]
096| exportschine=[0.000169418521177 31515, 20, 0.0005190236898779613, 21, 8.183776022972003e-05, 22, 2.871500358937545e-05, 23, 9.834888729361091e-05, 24,
3.589375448671931e-06, 25, 0.000183058147882 2685, 26, 0.0054666188083273 51, 27, 0.0010244077530509692, 0, 0.0002390524048815506, 1, 0.001256281407035176,
2, 0.0006898779612347451, 3, 0.000226130653266 33166, 4, 6.460875807609476e-06, 5, 0.0001385498923187 3655, 6, 2.871500358937545e-05, 7,
5.0251256281407036e-05, 8, 1.7946877243359657e-05, 9, 7.178750897343862e-07, 10, 9.117013639626705e-05, 11, 4.3072505384063175e-05,
12,2.4407753050969132e-05, 13, 1.8664752333094043e-05, 14, 2.871500358937545e-06, 15, 0.000295764536970 6714, 16, 2.08183776022972e-05, 17,
8.399138549892319e-05,18, 0.0016611629576453698, 19, 0.003267767408470926, 28, 0.007354630294328787, 29, 0.0, 30, 0.0, 31, 2.1536252692031 59e-06, 32,
0.002158650394831 2994, 33, 0.0,34, 0.0, 35, 7.394113424264179e-05, 36]
097| exportinde=[0.017706577974870658, 20, 0.5929563932002956, 21, 0.01785957132298596, 22, 0.0002062084257206 2085, 23, 0.0012483370288248338, 24,
0.007502586844050259, 25, 0.030428677014042 87, 26, 0.006866962305986696, 27, 0.04589578713968958, 0, 0.0034789356984478935, 1, 0.025127864005912787, 2,
0.1392239467849224, 3, 0.036182557280118 26, 4, 0.006658536585 3658535, 5, 0.020244641537324463, 6, 0.0025883222468 58832, 7, 0.0013325942350332594, 8,
0.005578713968957871, 9, 0.000144124168514412 42, 10, 0.005830746489283074 5, 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,
0.038518107908351 81, 28, 0.12163266814486327, 29, 0.0001500369549150037, 30, 0.000116777531411677 75, 31, 0.0005018477457501847, 32, 0.019248337028824832,
33, 0.008032520325203253, 34, 5.764966740576497e-05, 35, 0.0, 36]
098|
099|
100| exports=[exportsfrance, exportsespagne, exportsallemagne, exportsUK, exportsitalie, exportspologne, exportsbelgique, exportsgrece,
exportsreptcheque,exportsportugal, exportshongrie, exportssuede,exportsautriche,exportssuisse,exportsdanemark, exportsfinlande, exportsslovaquie,
exportsnorvege, exportsirlande, exportspaysbas, exportsalgerie, exportsarabiesaoudite, exportsisrael, exportsmalte, exportsmaroc, exportstunisie,
exportsegypte, exportsturquie, 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 creerpph(Pp, H):
109|     for i in range (len(Pp)):
110|         for k in range (len(Pp[i])):
111|             if k%2==0:
112|                 Pph[i][k]=Pp[i][k]/H[i][0]
113|     return Pph
114|
115|
116| ##liste avec les numéros des pays concernés
117| def listenum(H):
118|     listenum=[0]*(len(H))
119|     for i in range (len(H)):
120|         listenum[i]=H[i][1]
121|     return (listenum)
122|
123| ##creerseuildepart
124| def seuildep(T, sc, sd, S):
125|     for i in range (len(T)):
126|         if S[i]==0:
127|             if T[i]>=sc:
128|                 S[i]=1
129|             else:
130|                 S[i]=0
131|         else:
132|             if T[i]>sd:
133|                 S[i]=1
134|             else:
135|                 S[i]=0
136|     return S
137|
138| ##taux
139| def taux(C, L):
140|     T=[0]*(len(C))
141|     for i in range (len(C)):
142|             if (L[i][0]+L[i][1]+L[i][2])==0:
143|                 T[i]=0
144|             else:
145|                 T[i]=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
146|     return (T)
147|
148| def tauxlocal(C, L, i):
149|     T=0
150|     if (L[i][0]+L[i][1]+L[i][2])==0:
151|         T=0
```

```
    else:
        T=(L[i][1])/(L[i][0]+L[i][1]+L[i][2])
    return (T)

##creertableau
def creertableau(n, a):
    N=[0]*n
    for i in range (n):
        N[i]=[0]*a
    return N

##SIRevol
#Lt=[St, It, Rt, Dt] #D les dead
#d le taux de létalité =! taux de mortalité
def SIRevol(Lt, beta, i, k): # Lt la liste au temps t, L1 la liste au temps t+1, beta param de l'épidémie
#modèle d'évolution de t à t+1
    gamma=0.3
    d=0.048
    N=(Lt[0]+Lt[1]+Lt[2]+Lt[3])
    if N==0:
        return ([0, 0, 0, 0])
    else:
        L1=[0,0,0,0]
        L1[0]= (-beta[i][k]*Lt[0]*Lt[1])/(N) +Lt[0] #St+1
        L1[1]= beta[i][k]*Lt[0]*Lt[1]/(N) - gamma*Lt[1] +Lt[1] -d*Lt[1] #It+1
        L1[2]= Lt[2]+ gamma*Lt[1] #Rt+1
        L1[3]= Lt[3]+d*Lt[1]
        Lt=L1 #Lt devient L1 pour pouvoir ensuite le faire à chaque temps
        return (Lt) #on renvoie la liste au temps t+1


##renvoyer une case
def case(Lt,n):
    return (Lt[n])

##creerLt pour le début seulement
def creerLt(C, H, i): #crée un Lt pour le pays en position i
    Lt=[0, 0, 0, 0]
    Lt[0]=H[i][0]-C[i]
    Lt[1]=C[i]
    return Lt

##creerL au depart
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
#L[0][1] le nombre de contaminés du pays 0
#L[0][2] le nombre de recovered du pays 1
    L=creertableau(len(C), len(C))
    for i in range (len(C)):
        L[i]=creerLt(C,H,i)
    return L

##verifS
def verifS(S):
    n=len(S)
    res=0
    for i in range (n):
        if S[i]==1:
            res=res+1
    if res==n:
        return False

##creerV V=voyageurs en prenant en compte les stratégies de confinement

def creerV(L, C, S, Pop):
    paspossible=[]
    for i in range (len(C)):
        if Pop[i]==0:
            paspossible.append(i) # on fait la liste des gens qui ne peuvent pas voyager car ils sont morts
    V=[0]*(len(C))
    for i in range (len(C)):
            V[i]=[0]*2
    if verifS(S)==False:
        for i in range (len(C)):
            V[i][0]=0
    else :
        for i in range (len(C)):
            if S[i]==0:
                V[i][1]= rd.randint(0,(len(C)-1))
                while S[V[i][1]]==1 :
                    V[i][1]=rd.randint(0, len(C)-1)
                    while V[i][1] in paspossible :
                        V[i][1]=rd.randint(0, len(C)-1)
                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
cohérent
            else:
                V[i][0]=0
    return (V)

##creerD
def creerD(H, C, V, L): #on renvoie la liste des contaminés qui se déplacent et où ils se déplacent
    D=[0]*(len(C))
    T=taux(C, L)
    for i in range (len(C)):
        D[i]=[0]*2
    for i in range (len(C)):
        D[i][0]=(V[i][0]*T[i])
        D[i][1]=V[i][1]
    return (D)

##creerE
def creerE(H, S, E, exports, Pop):
    paspossible=[]
    for i in range (len(H)):
            if Pop[i]==0:
                paspossible.append(i)
    if verifS(S)==False:
        for i in range (len(E)):
            for k in range (len(E[i])):
                E[i][k]=0
    else:
        for i in range (len(E)):
            if S[i]==0:
                for k in range (len(E[i])):
                    if k%2==0:
                        E[i][k]=exports[i][k]*Pop[i]/365
                    else:
                        if k in paspossible:
                            E[i][k-1]=0
                        else:
                            E[i][k]=exports[i][k]

```

```
            else:
                for k in range (len(E[i])):
                    E[i][k]=0
    return E

##trouver les contaminés
def recherche(D, i):
    c=0 #res
    for k in range (len(D)):
        if D[k][1]==i:
            c=c+D[k][0]
    return c

def recherchebis(E,H,i):
    c=0
    for k in range (len(E)):
        for a in range (len(E[i])):
            if a%2==1:
                if E[k][a]==H[i][3]:
                    c=c+abs(E[k][a-1])
    return c

##creer beta
def creertableaubeta(n,C):
    beta=creertableau(len(C), n+1)
    for i in range (len(C)):
        beta[i][0]=0.39
    return beta

##creerZ
def creerZ(H, prop):
    Z=[0]*len(H)
    for i in range (len(Z)):
        Z[i]=H[i][1]*prop
    return Z

##faire manger les gens
def manger(Pop, Z, i, L):
    consoparhab=0.393750627
    viv=Pop[i] # nos vivants avant de voir qui aura assez à manger
    ts=0
    tc=0
    tr=0
    survivants=0
    capacité=0
    nouveauxmorts=0
    if Pop[i]!=0:
        ts=L[i][0]/viv
        tc=L[i][1]/viv
        tr=L[i][2]/viv
        capacité=Z[i]//0.393750627
        if capacité<viv:
            survivants=capacité
            nouveauxmorts=viv-survivants
            Pop[i]=survivants
            L[i][0]=ts*Pop[i]
            L[i][1]=tc*Pop[i]
            L[i][2]=tr*Pop[i]
            L[i][3]=L[i][3]+nouveauxmorts
            Z[i]=0
        else:
            Z[i]=Z[i]-Pop[i]*consoparhab

    return (Z[i], L[i])


##expansion avec confinement
def prevoirstocks(C, H, n, sc, sd, exports):# d le taux de gens qui meurent
    Pop=[0]*len(H)
    for i in range (len(H)):
        Pop[i]=H[i][0]
    L=creerL(C, H)
    beta=creertableaubeta(n, C)
    gamma=0.3
    d=0.048
    S=[0]*(len(C))
    T=taux(C,L)
    evoltaux=creertableau(len(C), n)
    stocksnecessaires=[0]*len(C)
    E=creertableau(len(C), len(exports))
    S=seuildep(T,sc, sd,S)
    V=creerV(L,C,S, Pop)
    D=creerD(H,C,V,L) #D le tableau des gens dangereux qui se déplacent et vers quel pays ils se déplacent
    evolcontamines=creertableau(len(C), n) #evolution des contaminés
    evolmorts=creertableau(len(C),n)
    evoldeE=creertableau(len(C), n)
    evolimports=creertableau(len(C),n)
    evoldeH=creertableau(len(C), n)
    listedeS=creertableau(len(C),n)
    prodpartravailleurparjour=[]
    for i in range (len(C)):
        evolmorts[i][0]=0
        evoldeH[i][0]=Pop[i]
        evoldeE[i][0]=0
        evoltaux[i][0]=T[i]
        evolimports[i][0]=0
        listedeS[i][0]=S[i]
    for i in range (len(H)):
        prodpartravailleurparjour.append(H[i][1]/H[i][0]/365)
    for k in range (1, n):
        T=taux(C,L)
        S=seuildep(T,sc, sd ,S)
        V=creerV(L,C,S, Pop)
        D=creerD(H,C,V,L)
        E=creerE(H, S, E, exports, Pop)
        for i in range (len(C)):
            #partie épidémiologique
            L[i]=SIRevol(L[i], beta, i, k)
            C[i]=L[i][1] + recherche(D,i)-D[i][0] #on contamine
            Pop[i]=Pop[i]+recherche(V,i)-V[i][0]
            L[i]=[Pop[i]-C[i]-L[i][2],C[i],L[i][2],L[i][3]]
            evolcontamines[i][k]=C[i]
            evolmorts[i][k]=L[i][3]
            listedeS[i][k]=S[i]
            if listedeS[i][k]==1:
                beta[i][k+1]=0.22
            else:
                beta[i][k+1]=0.39
            evoltaux[i][k]=tauxlocal(C,L, i)
            #partie échanges de céréales
            travailleurs=(L[i][0]+L[i][2])
```

```
            evolmorts[i][k]=L[i][3]
            evolcontamines[i][k]=L[i][1]
            evolmorts[i][k]=L[i][3]
            evoldeH[i][k]=Pop[i]
            evoldeE[i][k]=somme3(E, i)
            evolimports[i][k]=recherchebis(E,H,i)
            if recherchebis(E, H, i)-somme3(E, i)>=0:
                conso=(L[i][0]+L[i][1]+L[i][2])*0.393750627
            else:
                conso=(L[i][0]+L[i][1]+L[i][2])*0.393750627+somme3(E, i)-recherchebis(E, H, i)
            stocksnecessaires[i]=stocksnecessaires[i]+conso


    return (stocksnecessaires) #on renvoie la liste des états de chaque pays

##variation des stocks avec l'aléatoire
def variationdesstocks(C, H, n, sc, sd, exports, nombre): #trop grande complexité
    L=creertableau(len(H), nombre)
    for i in range (nombre):
        L[i]=prevoirstocks(C, H, n, sc, sd, exports)
    return L

def remettredansordre(C, H, n, sc, sd, exports, nombre): #L est le résultat de "variation des stocks"
    L1=creertableau(nombre,len(C))
    for i in range (len(L)):
        for k in range (len(L[i])):
            L1[i][k]=L[i][0]
    return L1


##tracer des évolutions en fonction
def tracervariations(L1, nombre): #L1 le resultat de "remettre dans l'ordre"
    abscisse=[0]*(10)
    for i in range (nombre):
        abscisse[i]=i
    for i in range (len(L1)):
        for k in range (nombre):
            plt.plot(abscisse, L1[i])
            plt.title("Stocks prévus au fil des essais")
            plt.xlabel("Essai")
            plt.ylabel("Stock prévu")
            plt.legend(fontsize=6, loc='best')
    return(plt.show())


def tracercontamines(C, H, n,sc, sd):
    evol=expansionconfined(C, H, n, sc, sd)[1]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOMS_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de contaminés en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de contaminés")
        plt.legend(fontsize=6, loc='best')
    return(plt.show())

def tracermorts(C, H, n, sc, sd, exports, prop):
    evol=expansionconfined(C, H, n, sc, sd, exports, prop)[0]
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for k in range (len(C)):
        plt.plot(abscisse,evol[k], label=str(NOM_PAYS[liste[k]]))
        plt.legend()
        plt.title("Nombre de morts en fonction du temps")
        plt.xlabel("Temps en jours")
        plt.ylabel("Nombre de morts")
        plt.legend(fontsize=6, loc='best')
    plt.show()

def touttracer(C, H, n, sc, sd, exports, prop):
    RES=expansionconfined(C, H, n, sc, sd, exports, prop)
    abscisse=[0]*n
    liste=listenum(H)
    for i in range (n):
        abscisse[i]=i
    for a in range (len(RES)):
        for k in range (len(liste)):
            plt.plot(abscisse, RES[a][k], label=str(NOM_PAYS[k]))
            plt.legend()
            plt.xlabel("temps en jour")
            plt.ylabel(str(AFFICHER[a]))
            plt.legend(fontsize=6, loc='best')
        plt.show()

##somme
def somme3(E, i):
    s=0
    for k in range (len(E)):
        s=s+E[i][k]
    return (s)

##utile
def proporprod(L, H):
    for i in range (len(L)):
        L[i]=L[i]/H[i][1]
    return (L)

##application
#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, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
0.003, 0.0006, exports)

import pylab
#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, 0,1, 0, 20,65,0, 2, 0,0,0,32616,0],monde, 365,
0.003, 0.0006, exports)

fig = plt.figure()
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]
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]
width = 0.05
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", "Israel", "Malte", "Maroc",
```

```
"Tunisie", "Egypte", "Turquie", "Canada", "USA", "Argentine", "Roumanie", "Kazakhstan", "Russie", "Ukraine", "Chine", "Inde"]

plt.bar(x, height, width, color=(0.65098041296005249, 0.80784314870834351, 0.89019608497619629, 1.0) )
plt.scatter([i+width/4.0 for i in x],height,color='b',s=10)

plt.xlim(0,38)
plt.ylim(0,200000000000)
plt.grid()

plt.ylabel('Counts')
plt.title('stocks nécessaires par pays')

pylab.xticks(x, BarName, rotation=40)

plt.savefig('SimpleBar.png')
plt.show()

```