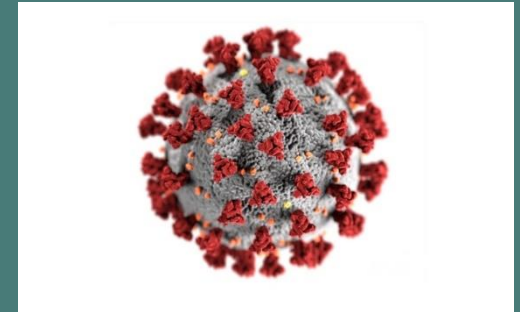
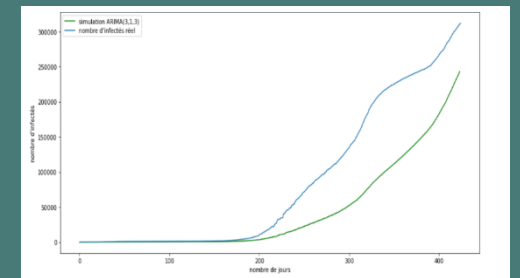


Prédiction de la propagation d'une épidémie



* Voir annexe



Objectif : Prédire l'évolution du Covid-19 en Tunisie en ayant recours à deux différents modèles :

- un modèle épidémiologique : SIR
- un modèle statistique : ARIMA

Plan :

1.

Prédiction d'une épidémie :

Modèle mathématique continue : SIR

Présentation du modèle :
équations différentielles

Choix des constantes

Simulation des résultats

Comparaison

2.

Modèle statistique : ARIMA (p,d,q)

Présentation du modèle et
explication des composantes

Rendre la série stationnaire

Simulation des résultats

Test de blancheur du résidu

1. Première approche : modélisation SIR

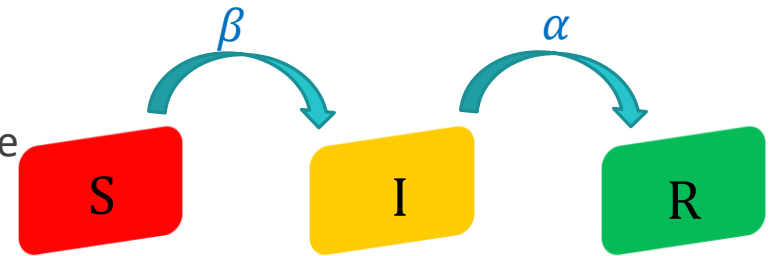
Le modèle mathématique SIR est un modèle de compartiments utilisé en épidémiologie .

La population est divisée en 3 groupes :

- S : les personnes susceptibles d'attraper la maladie .
- I : les personnes infectées
- R : les personnes rétablies

le modèle consiste en un système de 3 équations différentielles non linéaires d'ordre couple qui lient les 3 compartiments avec l'utilisation de deux constantes :

- α : taux de guérison
- β : taux de transmission

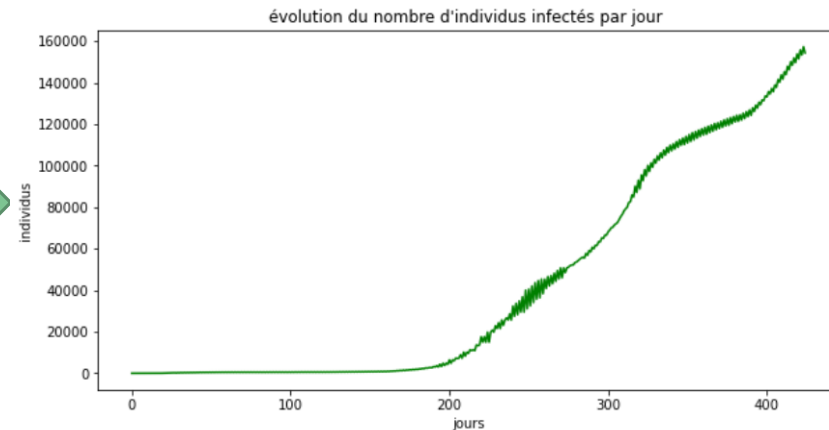


$$\left\{ \begin{array}{l} \frac{\delta S(t)}{\delta t} = -\frac{\beta(t)}{N} S(t) I(t) \\ \frac{\delta I(t)}{\delta t} = \frac{\beta(t)}{N} S(t) I(t) - \alpha(t) I(t) \\ \frac{\delta R(t)}{\delta t} = \alpha(t) I(t) \end{array} \right.$$

Présentation de la base de données collectée sur le site de WHO :

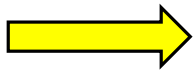
Annexe code 1

SNo		ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
3417	3418	03/04/2020	NaN	Tunisia	2020-03-04T01:33:07	1.0	0.0	0.0
3586	3587	03/05/2020	NaN	Tunisia	2020-03-04T01:33:07	1.0	0.0	0.0
3776	3777	03/06/2020	NaN	Tunisia	2020-03-04T01:33:07	1.0	0.0	0.0
3992	3993	03/07/2020	NaN	Tunisia	2020-03-04T01:33:07	1.0	0.0	0.0
4194	4195	03/08/2020	NaN	Tunisia	2020-03-08T21:13:10	2.0	0.0	0.0
4459	4460	03/09/2020	NaN	Tunisia	2020-03-08T21:13:10	2.0	0.0	0.0
4699	4700	03/10/2020	NaN	Tunisia	2020-03-10T05:13:07	5.0	0.0	0.0
4907	4908	03/11/2020	NaN	Tunisia	2020-03-11T18:52:03	7.0	0.0	0.0
5130	5131	03/12/2020	NaN	Tunisia	2020-03-11T18:52:03	7.0	0.0	0.0
5374	5375	03/13/2020	NaN	Tunisia	2020-03-11T20:00:00	16.0	0.0	0.0



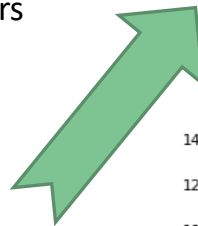
Pendant les 14 premiers jours :

La durée de l'infection étant fixée à 14 jours, le nombre de personnes rétablies pendant les 14 premiers jours n'évolue pas.

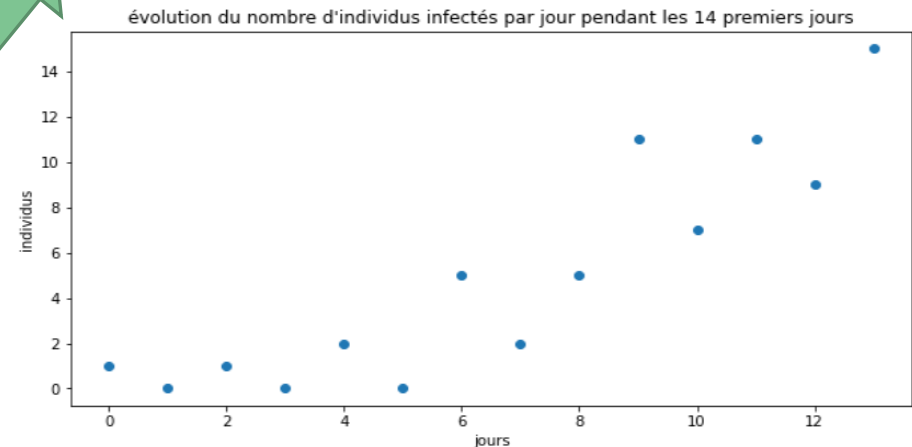


$$\frac{\delta R}{\delta t} \approx 0$$

$$\begin{cases} \frac{\delta R}{\delta t} = \alpha I(t) \\ \frac{\delta I}{\delta t} = (\frac{\beta}{N} S(t) - \alpha) I(t) \\ \frac{S(t)}{N} \approx 1 \end{cases} \Rightarrow \frac{\delta I}{\delta t} = \beta I(t)$$



La courbe du nombre d'infectés par jours a une allure exponentielle pendant les 14 premiers jours.



Choix des constantes :

1. $\alpha = \frac{1}{\gamma}$ avec γ la durée de l'infection ainsi, comme $\gamma \approx 14$, on suppose que $\alpha = 0,07$
2. On cherche β qui permet d'approcher exponentiellement le nuage de points pendant les 14 premiers jours

Méthode : Minimisation de la somme des carrés des résidus :

Soit Y la variable à expliquer et \hat{Y} la variable prédite . On pose $\varepsilon = Y - \hat{Y}$ le résidu .

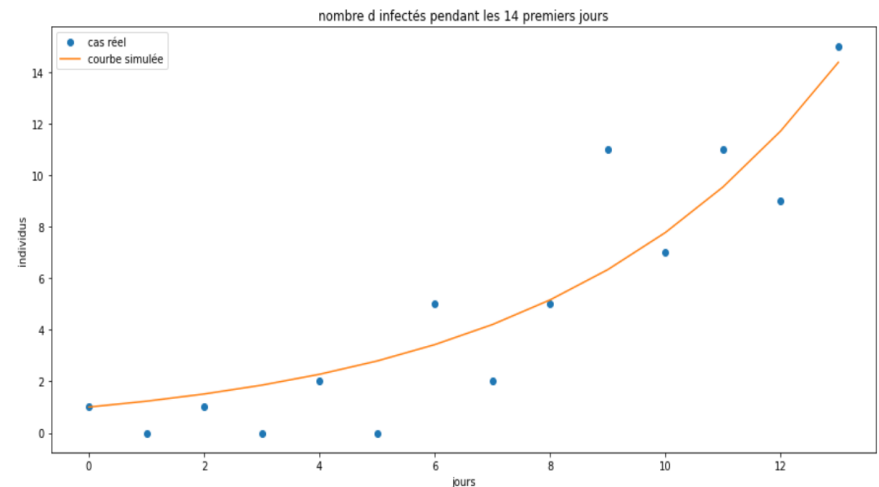
La somme des carrés des résidus $SCR = \sum \varepsilon_i^2$. Elle mesure la distance de la courbe de la variable prédite aux points du nuage de points qui est minimale au sens des moindres carrés. On cherche à minimiser SCR pour retrouver une distance qui est minimale au sens des moindres carrés.

On sait que $\beta \in [0,2 ; 0,3]$

Voir annexe i1

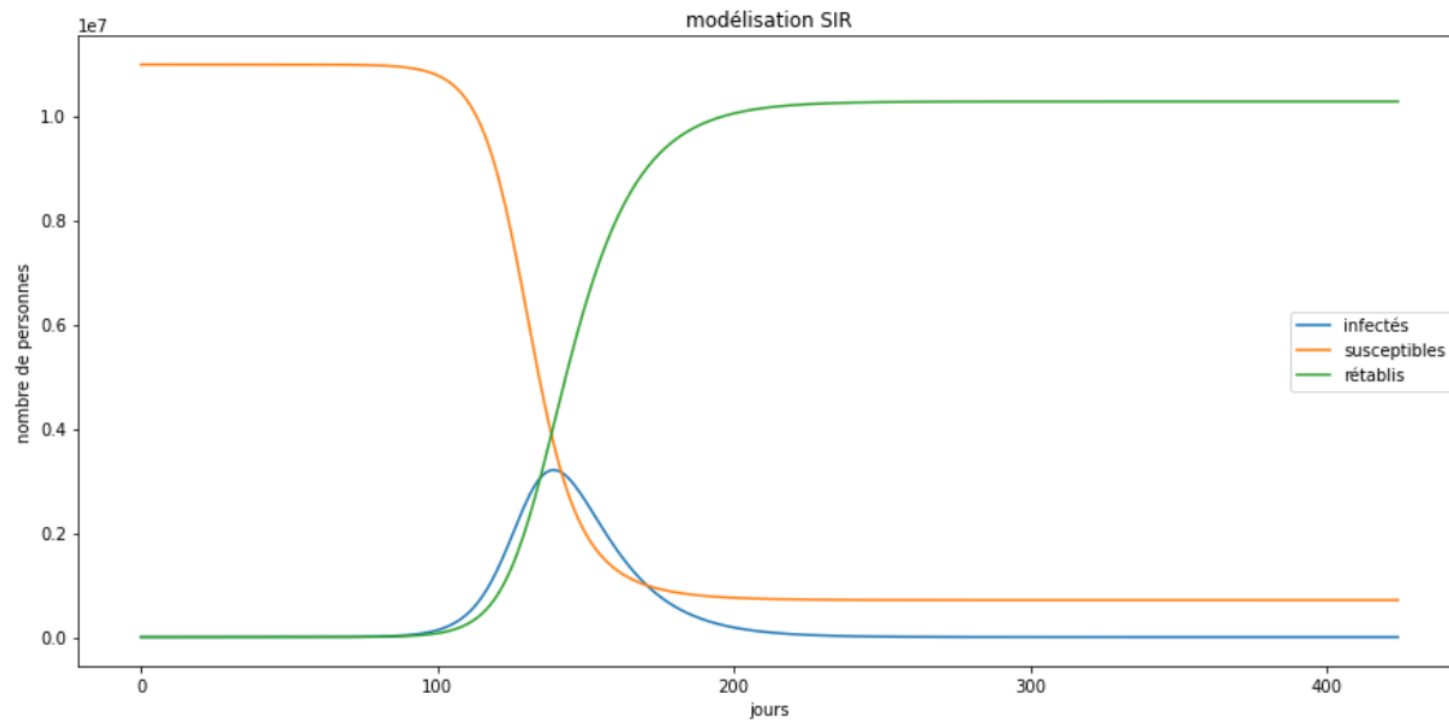
Minimiser la somme des carrés des résidus

$\beta = 0,205$




Voir annexe code 2


Première simulation pour $\alpha=0,07$ et $\beta=0,205$:



Réajustement des valeurs de α et β selon l'évolution journalière :

□ Pour plus de précision on choisit de modifier les valeurs de α et β selon l'évolution journalière de la pandémie.  $\alpha(t), \beta(t)$

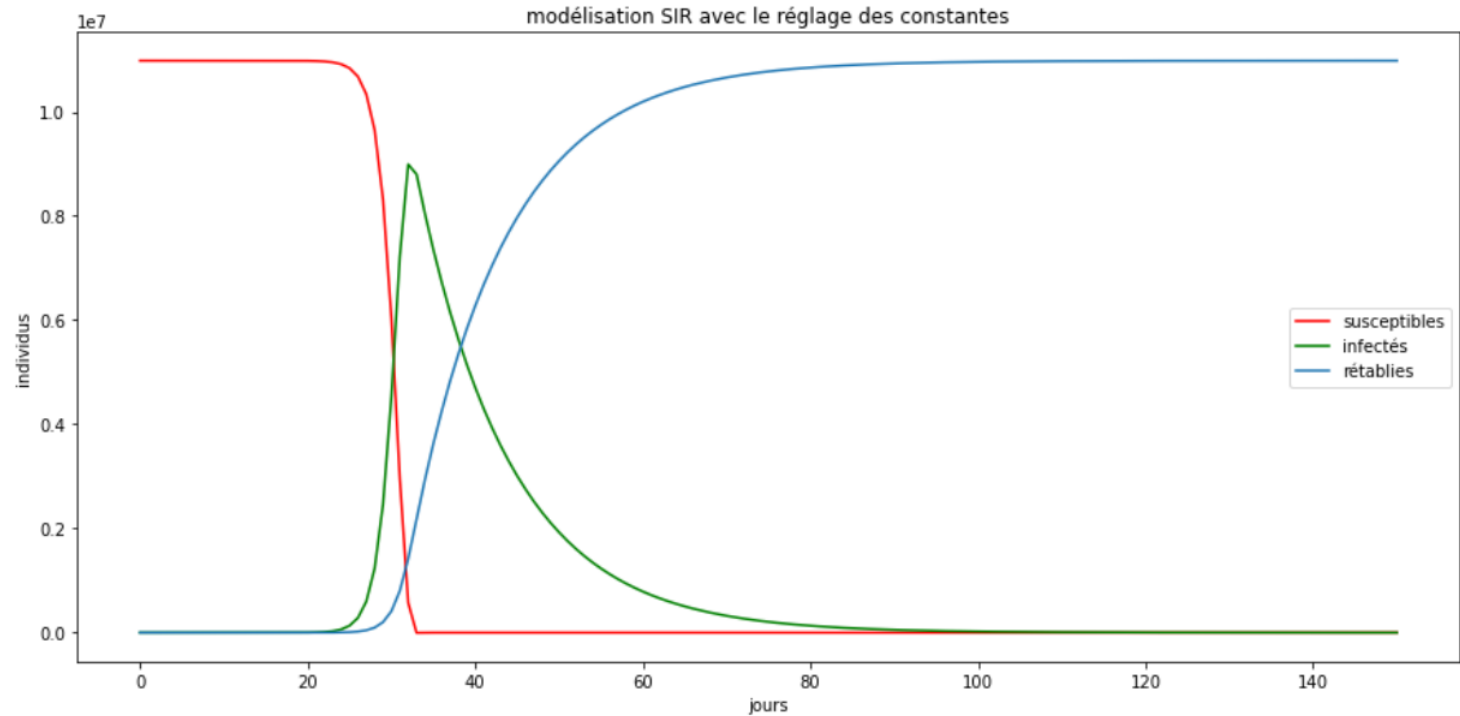
□ On suppose que $\alpha(t) = \alpha(t-1)$ et $\beta(t) = \beta(t-1)$


$$\left\{ \begin{array}{l} S(t) - S(t-1) = -\frac{\beta(t)}{N} S(t) I(t) \\ I(t) - I(t-1) = \frac{\beta(t)}{N} S(t) I(t) - \alpha(t) I(t) \\ R(t) - R(t-1) = \alpha(t) I(t) \end{array} \right.$$
$$\left\{ \begin{array}{l} \beta(t) = N \frac{S(t-1) - S(t)}{I(t-1)S(t-1)} \\ \alpha(t) = \frac{R(t) - R(t-1)}{I(t)} \end{array} \right.$$

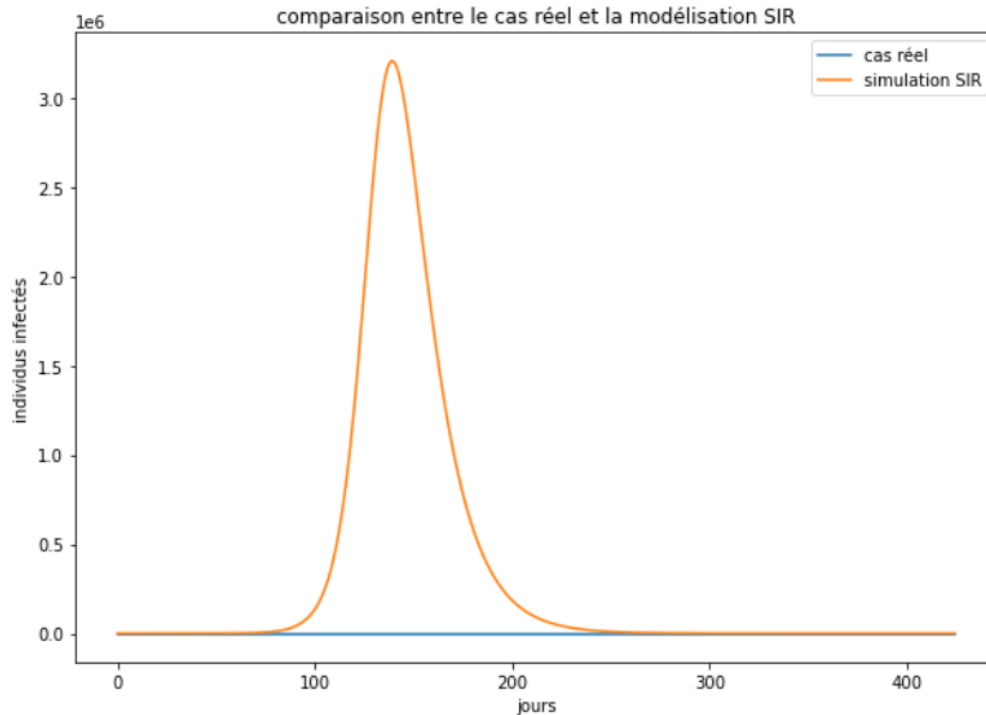
Annexe Code 3

Annexe Code 4

Simulation des résultats du modèle SIR en actualisant les valeurs de α et β selon l'évolution journalière :



Comparaison des résultats de la modélisation SIR avec le cas réel :



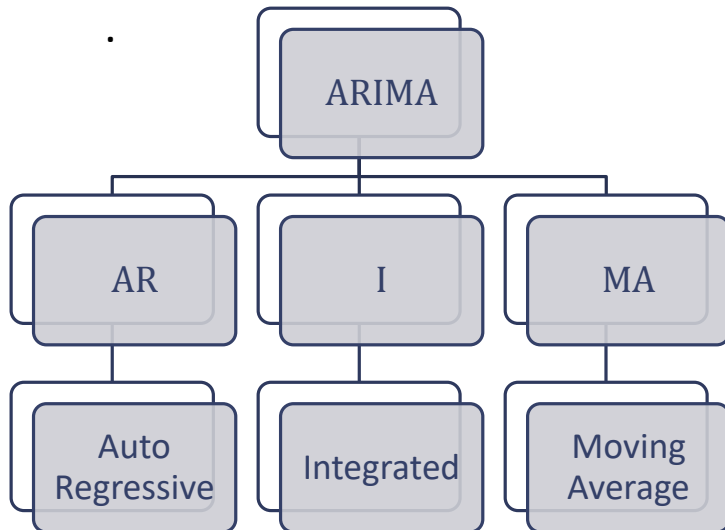
- L'écart entre la modélisation épidémiologiste SIR et la courbe réelle est très grand .
- C'était prévisible puisqu'on n'a pas pris en considération l'effet des mesures sanitaires instaurées .



On choisit donc d'adopter une autre démarche : prédire la propagation du virus à l'aide d'un modèle statistique adapté aux séries temporelles : l'**ARIMA** puis comparer les résultats .

2. Deuxième approche : ARIMA

L' ARIMA est un modèle de prédiction de séries temporelles



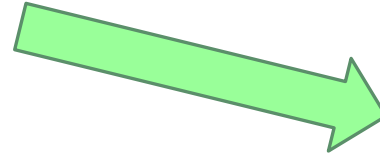
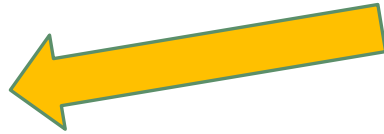
Ce modèle est constitué de 3 paramètres :

p , d et q et chacune correspond à une des composantes du modèle .

- Dans le modèle auto-régressif d'ordre p $AR(p)$, La modélisation de X_t se résume à une relation linéaire le liant aux p derniers instants : $\forall t, X(t) = \varepsilon_t + \sum_{i=1}^p \varphi_i X(t - i)$
- Dans le modèle de moyenne mobile d'ordre q $MA(q)$, le processus est la combinaison linéaire de bruit blanc : $\forall t, X(t) = \varepsilon_t + \sum_{i=1}^q \theta_i \varepsilon_{t-i}$
- d : le nombre de fois qu'il faut différencier la série afin de la rendre stationnaire.

Notions de stationnarité et bruit blanc :

Pour appliquer l'ARIMA, le processus doit être stationnaire et le résidu doit être un bruit blanc .



Notion de stationnarité :

Un processus $(X_t)_{t \in \mathbb{Z}}$ est (faiblement) stationnaire si son espérance et ses autocovariances sont invariantes par translation dans le temps .

$$\begin{cases} \forall t, & E(X(t)) = \mu \\ \forall t, \forall h, & Cov(X(t), X(t-h)) = C_h \end{cases}$$

Notion de bruit blanc :

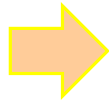
Un résidu ε sans dépendance linéaire temporelle .

$$\begin{cases} \forall t, & E(\varepsilon(t)) = 0 \\ \forall t, & E(\varepsilon(t)^2) = \sigma^2 \\ \forall t, \forall t', & Cov(\varepsilon(t), \varepsilon(t')) = 0 \end{cases}$$

Effet de la transformation logarithmique

Calcul de variance :

$$V_k = \sqrt{\sum_{i=k}^{k+T} (x_i^2 - \hat{x}_i^2)}$$



```
def somme_carre(ni,j,a):  
    t=0  
    for k in range(j,j+a):  
        t+=ni[k]**2 -x[k]**2  
    return(t)
```

```
def variance(ni,a,n):  
    a=7  
    x=moymob(ni,a,n)  
    v=[]  
    for k in range(n-a):  
        t=math.sqrt(somme_carre(ni,k,a)/a)  
        v.append(t)  
    return(v)
```

Ayant une base avec une évolution journalière, j'ai décidé de faire une moyenne hebdomadaire

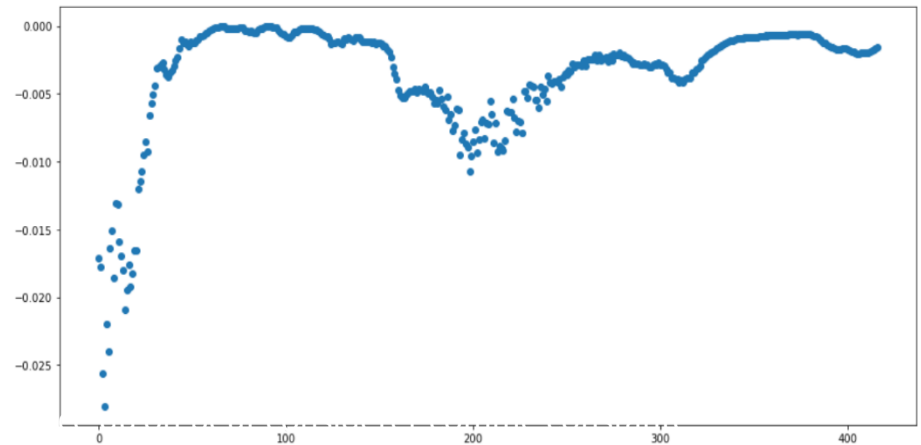
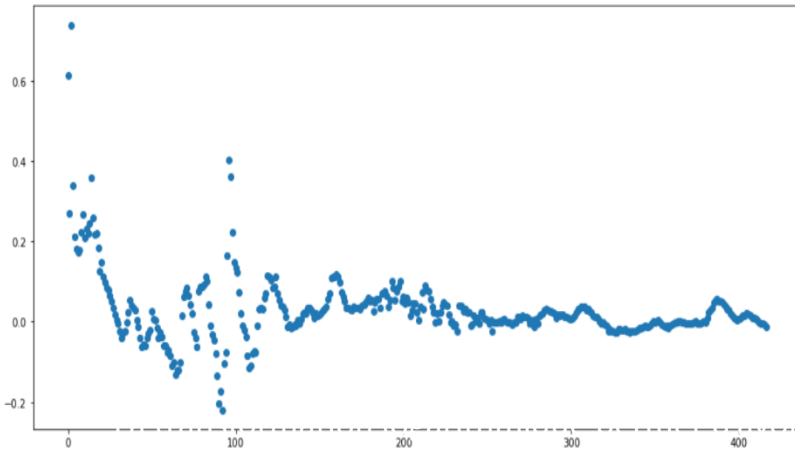
Appliquer le log

```
nilog=[math.log(ni[k]) for k in range(len(ni))]
```

```
xlog=moymob(nilog,a,n)
```

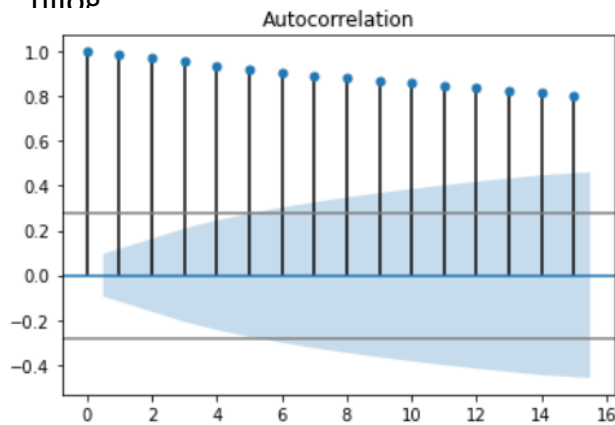
```
v_ni=variance(ni,a,n)  
check_v_cte=[(v_ni[k+1]-v_ni[k])/v_ni[k] for k in range(len(v_ni)-1)]
```

```
v_nilog=variance(nilog,a,n)  
check_vlog_cte=[(v_nilog[k+1]-v_nilog[k])/v_nilog[k] for k in range(len(v_ni)-1)]
```



Choix de la constante d :

On trace la courbe d'autocorrélation pour nilog

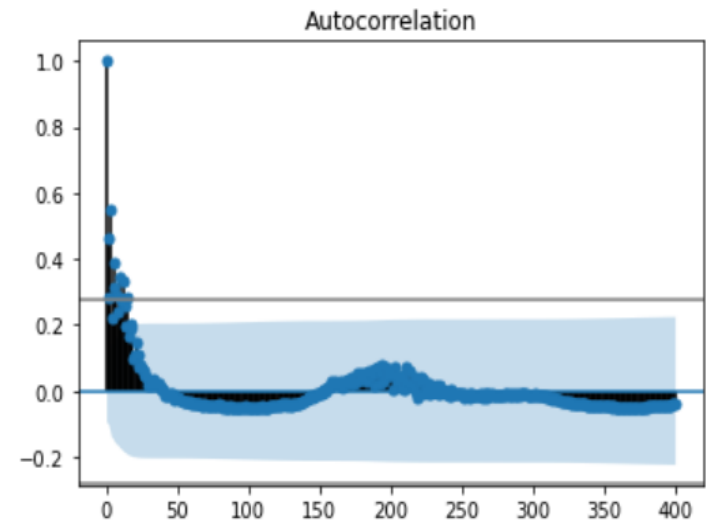


La courbe d'autocorrélation diminue lentement .

On différencie

On trace la courbe d'autocorrélation pour i .

```
#d=1  
i=[nilog[k+1]-nilog[k] for k in range(len(nilog)-1)]
```



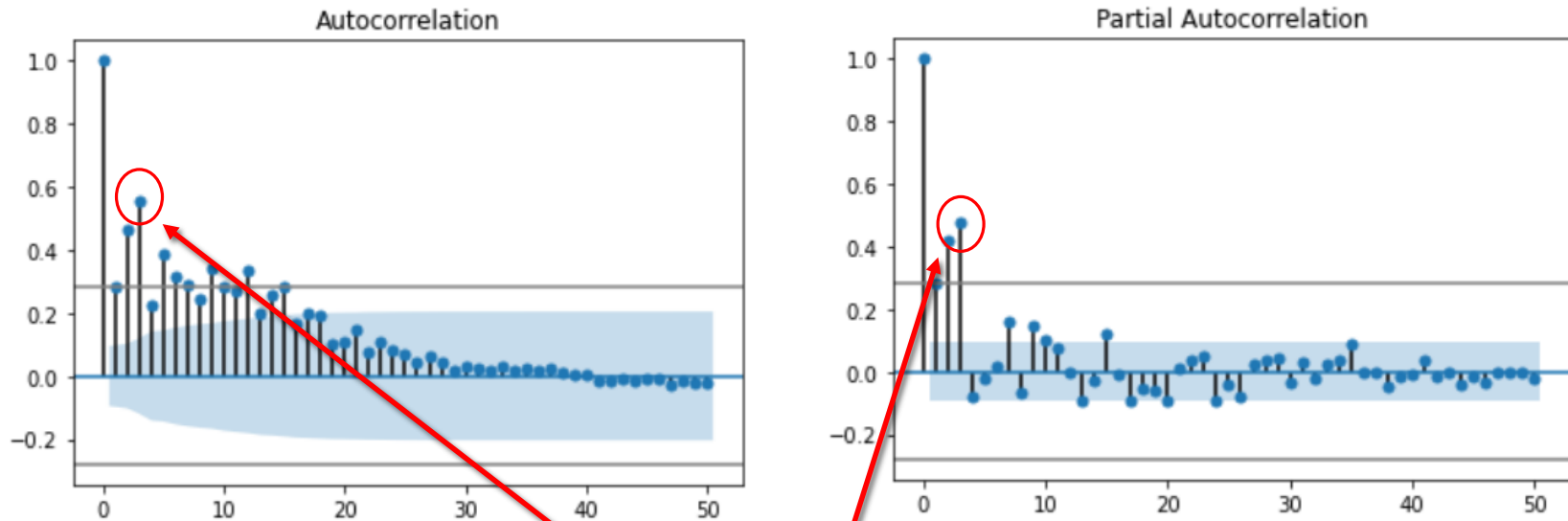
La courbe d'autocorrélation décroît rapidement vers 0 .

d=1 ←

Voir annexe code 5

Choix des constantes p et q :

Traçage des courbes d'autocorrélation et d'autocorrélation partielle pour i .

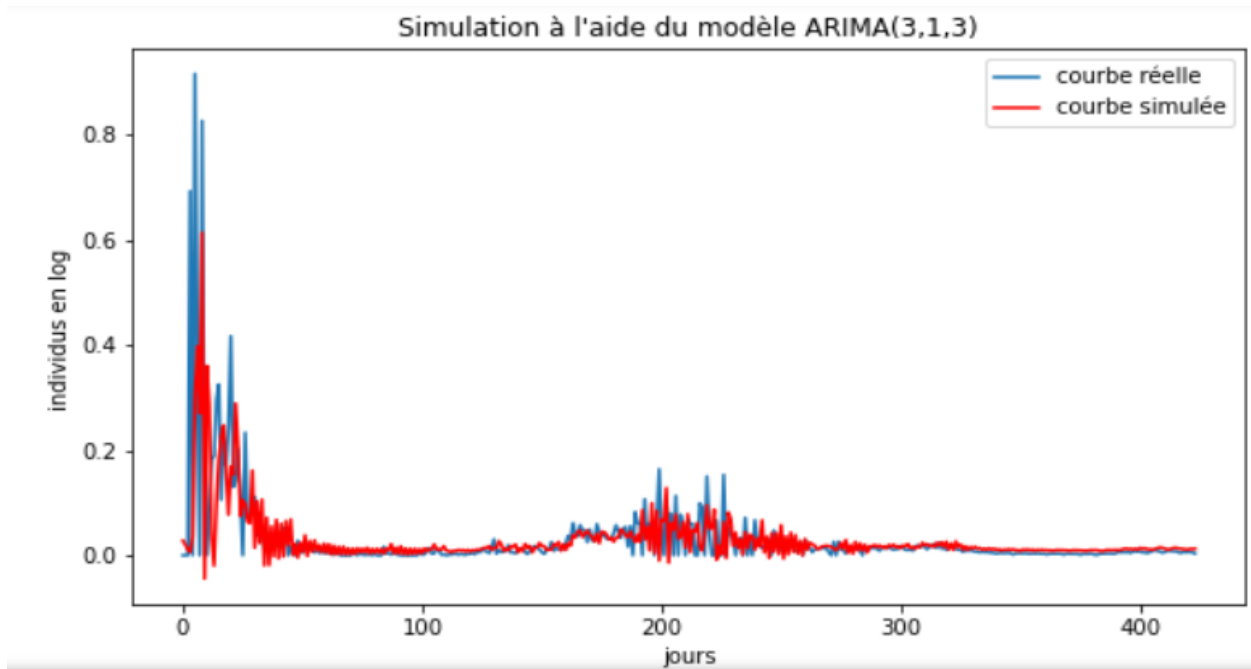


En s'intéressant aux pics importants en dehors de l'intervalle délimité en bleu, p et q devraient être égales à 3

$p=3, q=3$

Voir annexe code 6

Implémentation du modèle ARIMA (3,1,3) :




- La courbe réelle décrit l'évolution du processus après la transformation logarithmique .
- La courbe simulée est le résultat du modèle ARIMA(3,1,3) .
 - Interprétation :
Les 2 courbes sont presque confondues. Le modèle semble approprié . Il faut tester la blancheur du résidu .

Voir annexe code 7

Le résidu est-il un Bruit Blanc ?

La statistique de Portemanteau Q :

soit $X(t)$ le processus stationnaire
 $S(t)$ le processus simulé
 $\varepsilon(t) = X(t) - S(t)$ le résidu


$$Q_k = \sum_{h=1}^k \hat{\rho}^2(h)$$

$$\text{avec } \hat{\rho}(h) = \frac{T-h}{T} \frac{\sum_{t=h+1}^T (\varepsilon_t - \bar{\varepsilon}_T)(\varepsilon_{t-h} - \bar{\varepsilon}_T)}{\sum_{t=1}^T (\varepsilon_t - \bar{\varepsilon}_T)^2}$$

$$\text{Si } h \ll T ; \hat{\rho}(h) = \frac{\sum_{t=h+1}^T (\varepsilon_t - \bar{\varepsilon}_T)(\varepsilon_{t-h} - \bar{\varepsilon}_T)}{\sum_{t=1}^T (\varepsilon_t - \bar{\varepsilon}_T)^2}$$

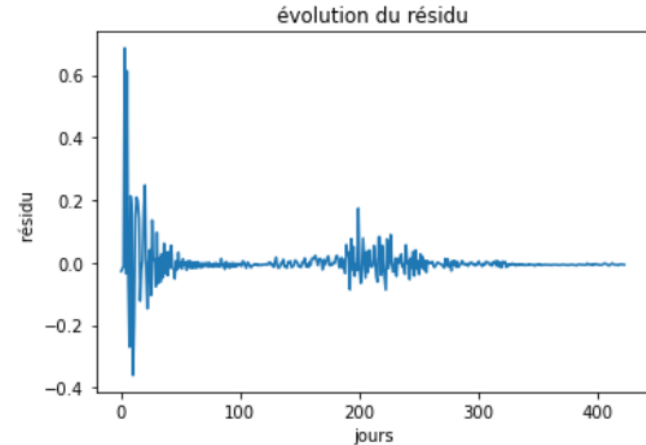
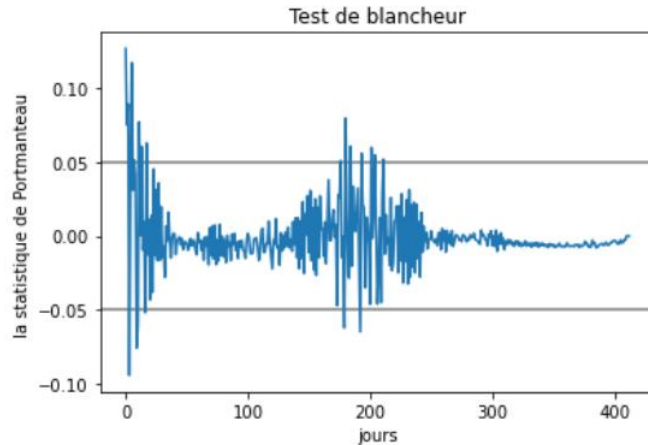
Test de blancheur

Q_k faible

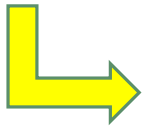


Bruit Blanc

Le résidu est-il un Bruit Blanc ?



La statistique de Portemanteau reste toujours dans l'intervalle $\pm 5\%$ sauf pour quelques pics qui n'atteignent pas des valeurs très importantes.



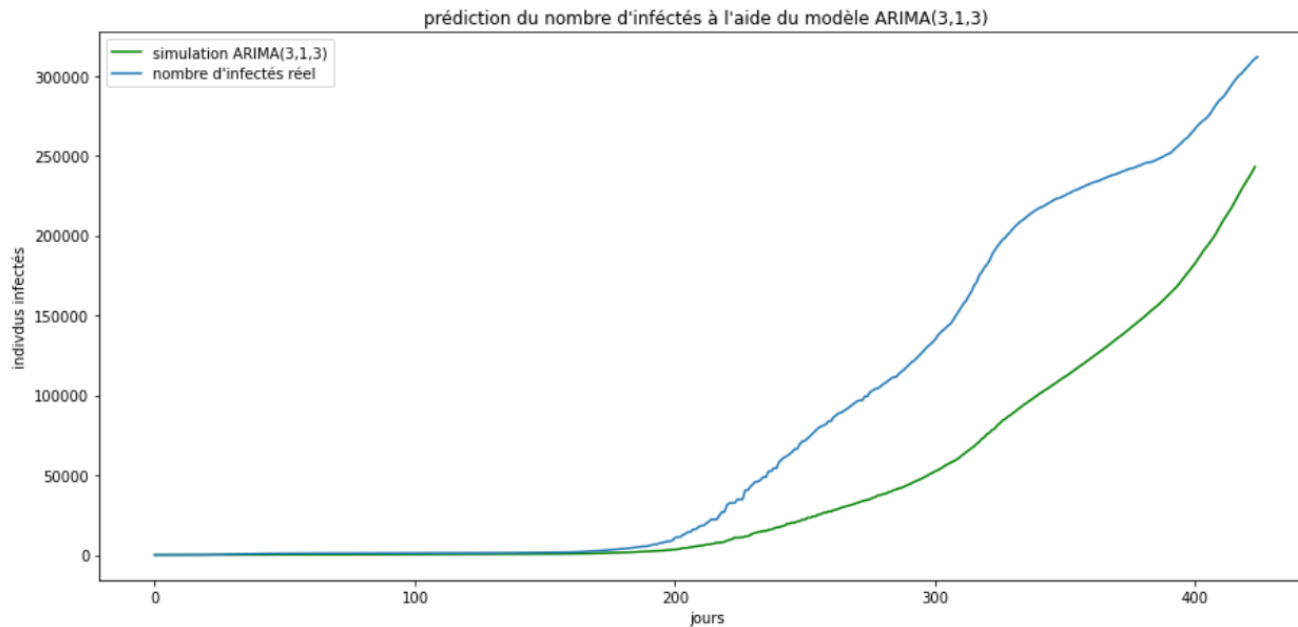
Q_k est
faible



Le résidu peut être considéré comme un bruit blanc

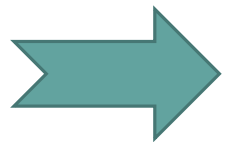
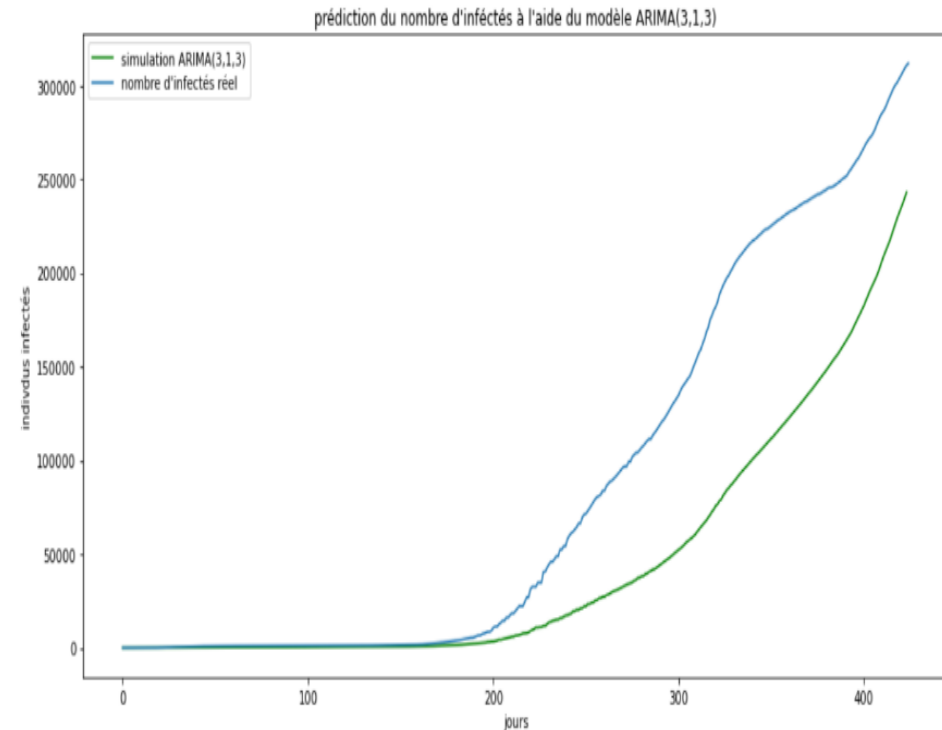
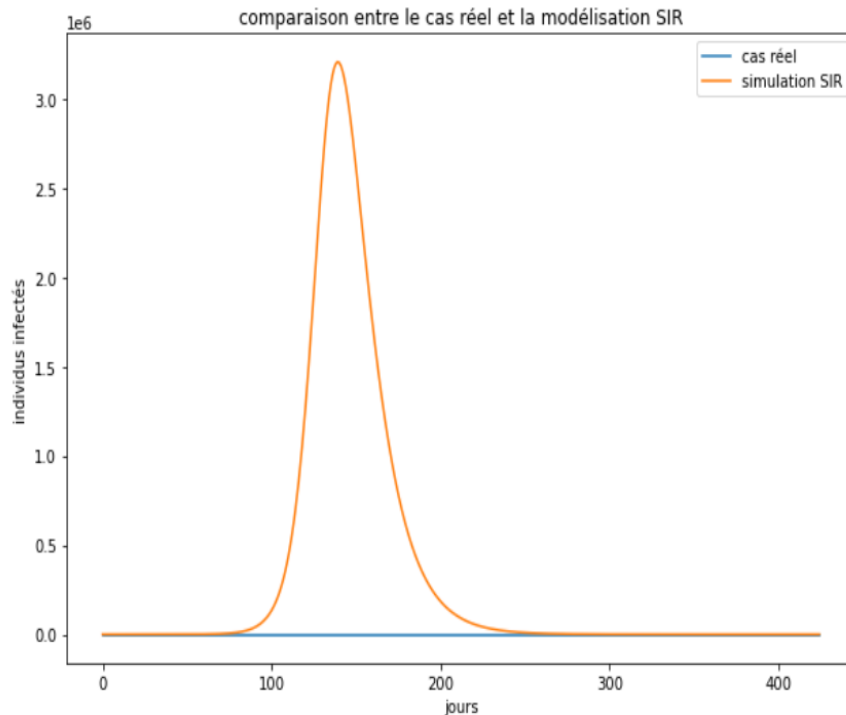
Voir annexe code 8

Prédiction grâce au modèle ARIMA(3,1,3) :



Voir annexe code 9

Comparaison :



Conclusion :

Malgré que le SIR est un modèle mathématique épidémiologiste, l'ARIMA s'avère dans le cas tunisien beaucoup plus efficace pour la prédiction de l'évolution du virus en Tunisie .

Annexe :

Code 1 :

```
import pandas as pd
data=pd.read_csv(r'C:\Users\user\Downloads\covid_19_data.csv')
print(data)
```

	SNo	ObservationDate	Province/State	Country/Region \
0	1	01/22/2020	Anhui	Mainland China
1	2	01/22/2020	Beijing	Mainland China
2	3	01/22/2020	Chongqing	Mainland China
3	4	01/22/2020	Fujian	Mainland China
4	5	01/22/2020	Gansu	Mainland China
...
285302	285303	05/02/2021	Zaporizhia Oblast	Ukraine
285303	285304	05/02/2021	Zeeland	Netherlands
285304	285305	05/02/2021	Zhejiang	Mainland China
285305	285306	05/02/2021	Zhytomyr Oblast	Ukraine
285306	285307	05/02/2021	Zuid-Holland	Netherlands

	Last Update	Confirmed	Deaths	Recovered
0	1/22/2020 17:00	1.0	0.0	0.0
1	1/22/2020 17:00	14.0	0.0	0.0
2	1/22/2020 17:00	6.0	0.0	0.0
3	1/22/2020 17:00	1.0	0.0	0.0
4	1/22/2020 17:00	0.0	0.0	0.0
...
285302	2021-05-03 04:20:39	96531.0	1919.0	78700.0
285303	2021-05-03 04:20:39	26045.0	233.0	0.0
285304	2021-05-03 04:20:39	1344.0	1.0	1322.0
285305	2021-05-03 04:20:39	84641.0	1597.0	68529.0
285306	2021-05-03 04:20:39	359327.0	4138.0	0.0

```
data1=data[data['Country/Region']=='Tunisia']
print(data1)
```

	SNo	ObservationDate	Province/State	Country/Region \
3417	3418	03/04/2020	NaN	Tunisia
3586	3587	03/05/2020	NaN	Tunisia
3776	3777	03/06/2020	NaN	Tunisia
3992	3993	03/07/2020	NaN	Tunisia
4194	4195	03/08/2020	NaN	Tunisia
...
281645	281646	04/28/2021	NaN	Tunisia
282409	282410	04/29/2021	NaN	Tunisia
283173	283174	04/30/2021	NaN	Tunisia
283937	283938	05/01/2021	NaN	Tunisia
284701	284702	05/02/2021	NaN	Tunisia

	Last Update	Confirmed	Deaths	Recovered
3417	2020-03-04T01:33:07	1.0	0.0	0.0
3586	2020-03-04T01:33:07	1.0	0.0	0.0
3776	2020-03-04T01:33:07	1.0	0.0	0.0
3992	2020-03-04T01:33:07	1.0	0.0	0.0
4194	2020-03-08T21:13:10	2.0	0.0	0.0
...
281645	2021-04-29 04:20:55	305313.0	10563.0	255870.0
282409	2021-04-30 04:21:03	307215.0	10641.0	258190.0
283173	2021-05-01 04:20:47	309119.0	10722.0	259957.0
283937	2021-05-02 04:20:48	310734.0	10808.0	261552.0
284701	2021-05-03 04:20:39	311743.0	10868.0	262602.0

*lien de l'image :

https://ichef.bbci.co.uk/news/640/cpsprodpb/17700/production/_117500069_9db48266-1bb0-40c4-9c75-b6d2b06c30be.jpg

Code 2 :

```
alpha=0.07
beta1=0.2
beta2=0.3
import math as m
def minscr(ni,I0,beta1,beta2,alpha):
    a=int(1/alpha)
    B=numpy.linspace(beta1,beta2,100)
    smin=0
    betamin=beta1
    for t in range(a):
        smin=smin+(ni[t]-I0*m.exp(beta2*t))**2
    for beta in B :
        s=0
        for t in range(a):
            s=s+(ni[t]-I0*m.exp(beta*t))**2
        if s<smin :
            smin=s
            betamin=beta
    return(betamin)

betamin=minscr(ni,I0,beta1,beta2,alpha)

print(betamin)

0.20505050505050507
```

Code 3 :

```
def guess_beta(s,i,r,k,alpha,beta):
    c=N(s[k-1]-s[k])/(s[k-1]*i[k-1])
    if c==0:
        return(beta)
    else :
        return(c)
```

Code 4 :

```
def guess_alpha(s,i,r,k,alpha,beta):
    c=(r[k]-r[k-1])/i[k-1]
    if c==0:
        return(alpha)
    else :
        return(c)
```

Code 5

```
tsaplots.plot_acf(nilog, lags=15)
plt.axhline(y=1.96/7, linestyle='--', color='gray')
plt.axhline(y=-1.96/7, linestyle='--', color='gray')
plt.show()

tsaplots.plot_acf(i, lags=400)
plt.axhline(y=1.96/7, linestyle='--', color='gray')
plt.axhline(y=-1.96/7, linestyle='--', color='gray')
plt.show()
```

Code 6

```
tsaplots.plot_acf(i, lags=20)
plt.axhline(y=1.96/7, linestyle='--', color='gray')
plt.axhline(y=-1.96/7, linestyle='--', color='gray')
plt.show()

tsaplots.plot_pacf(i, lags=20)
plt.axhline(y=1.96/7, linestyle='--', color='gray')
plt.axhline(y=-1.96/7, linestyle='--', color='gray')
plt.show()
```

Code 7

```
model = ARIMA(nilog, order=(3,1,3) )
results0 = model.fit(displ=-1)
plt.figure(figsize=(9,5))
plt.plot(i, label="courbe réelle")
plt.plot(results0.fittedvalues, color='red', label="courbe simulée")
plt.xlabel("jours")
plt.ylabel("individus en log ")
plt.legend()
plt.title("Simulation à l'aide du modèle ARIMA(3,1,3)")
plt.show()
```

Code 8

```
def moy(x):
    s=0
    for i in range(len(x)):
        s+=x[i]
    return(s/len(x))

def stat_Portmanteau(x,k):
    X=moy(x)
    s1=0
    for i in range(k+1,len(x)):
        s1+=(x[i]-X)*(x[i-k]-X)
    s2=0
    for i in range(len(x)):
        s2+=(x[i]-X)**2
    return(s1/s2)

Q=[stat_Portmanteau(r,k) for k in range(len(r))]

plt.title('Test de blancheur')
plt.xlabel('jours')
plt.ylabel("la statistique de Portmanteau ")
plt.axhline(y=0.05, linestyle='--', color='gray')
plt.axhline(y=-0.05, linestyle='--', color='gray')
plt.plot(Q[10:])
plt.show()
```

Code 9

```
result_diff=[]
result_diff.append(results.fittedvalues[0])
for k in range(1,len(results.fittedvalues)):
    result_diff.append(results.fittedvalues[k-1]+result_diff[k-1])
r=[math.exp(result_diff[k]) for k in range(len(result_diff))]
ni1=[N*ni[k] for k in range(len(ni))]

plt.figure(figsize=(15,7))
plt.plot(r,'g',label='simulation ARIMA(3,1,3)')
plt.plot(ni1,label="nombre d'infectés réel")
plt.xlabel('jours')
plt.ylabel("individus infectés")
plt.legend()
plt.title("prédiction du nombre d'infectés à l'aide du modèle ARIMA(3,1,3)")
plt.show()
plt.savefig('simulation arima')
```

i1: d'après le document [4], on sait que la constante β serait dans cet intervalle.