rapport d’une simulation multi-agent de propagation épidémique

--- Hengshuo LI

Introduction

Objectif

Nous simulons la propagation des maladies infectieuses, analysons les causes des maladies infectieuses, retrouvons le processus de transmission et prenons certaines mesures pour arrêter la propagation des maladies infectieuses.

De nombreux modèles pour la simulation d'épidémies existent, qu'ils soient microscopiques ou macroscopiques, mathématiques, fondés sur la physique, l'analyse comportementale... Leurs objectifs peuvent être la reproduction de phénomènes existants, l'analyse de cause ou encore la prédiction de l'évolution future du phénomène

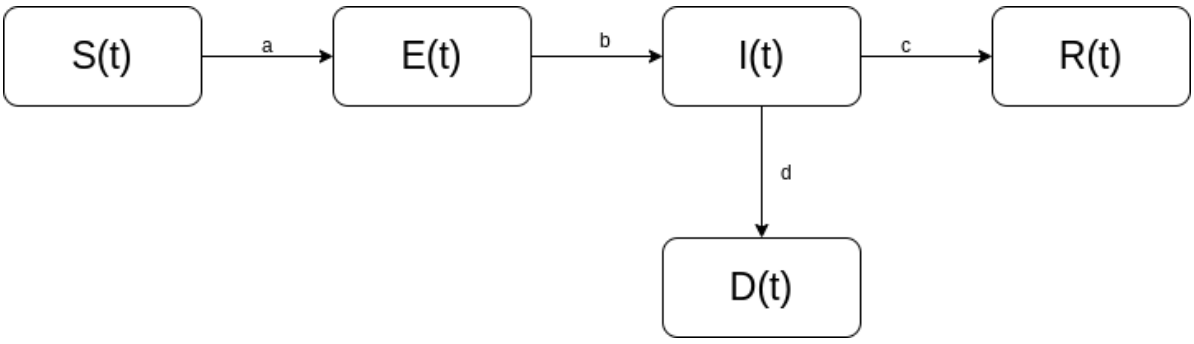
Contexte

Nous simulons un modèle de maladie infectieuse où le virus ne peut être transmis que par voie aérienne et dans certaines conditions initiales dans une zone spécifique, c'est-à-dire une zone finie, un certain nombre initial de personnes infectées par le virus, une certaine densité de population, pour un virus spécifique, c'est-à-dire un certain taux d'infection, un certain taux de mortalité (par le virus) et un certain temps de guérison, en partant de l'hypothèse que l'activité de la population est intense et que la population ne migre pas et n'entre pas en contact avec le monde extérieur, en utilisant des simulations multi-agents, pour simuler le processus de transmission du virus dans le temps.

Modélisation

Modélisation d'une épidémie

Pour une population donnée, on étudie la taille de cinq sous-populations au cours du temps t : $S(t)$ représente les personnes **saines au temps** t(susceptible en anglais), $E(t)$ les personnes **infectées non-infectieuses** (exposed en anglais), qui ne sont donc pas contagieuses, $I(t)$ **les personnes infectées** , $R(t)$ **les personnes retirées** et $D(t)$ **les personnes sont mort**(dead en anglais) , Pour $D(t)$ que j'ai ajouté moi-même pour rendre le nombre de décès plus intuitif et pour garder la population totale constante tout au long de la période.
$N = S(t) + E(t) + I(t) + R(t) + D(t)$ représente alors la population constante totale au cours du temps. Il convient de bien différencier les personnes *saines* des personnes *retirées* : les personnes saines n'ont pas encore été touchées par le virus, alors que les personnes retirées sont guéries, et donc immunisées. Autrement dit, les personnes retirées ne sont plus prises en compte. Le modèle SEIRD peut donc être représenté par le schéma suivant :



Ici, *a* représente le *taux de transmission*, c'est à dire le taux de personnes saines qui deviennent infectées et *c* le *taux de guérison*, c'est à dire le taux de personnes infectées qui deviennent retirées,ce qui permet de prendre en compte la temps de guérison d'une maladie. , *b* le *taux d'incubation*, aussi prendre en compte la durée d'incubation d'une maladie,d le *taux de mortalité d'une maladie*.$d = 1 - c$, Mathématiquement, le modèle SEIRD est donné par le système suivant :

$$\begin{cases} \frac{dS(t)}{dt} = -aS(t)I(t) \\ \frac{dE(t)}{dt} = aS(t)I(t) - bE(t) \\ \frac{dI(t)}{dt} = bE(t) - cI(t) - dI(t) \\ \frac{dD(t)}{dt} = dI(t) \\ \frac{dR(t)}{dt} = cR(t) \end{cases}$$

Les dérivées permettent de connaître la variation (c'est à dire si c'est croissant ou décroissant) des fonctions *S*, *E*,*I* ,*R* et *D* en fonction du temps *t*, afin d'en décrire l'évolution au cours du temps. Le terme *S*(*t*)*I*(*t*) représente le nombre de contacts entre des personnes saines et des personnes infectées.

Modélisation informatique

Dans le cadre de ce projet, nous nous intéressons à une méthode de modélisation particulière, la simulation à base d'agents.

Les simulations à base d'agents permettent de simuler des interactions existant entre agents autonomes. L'objectif est de déterminer l'évolution du système afin de prévoir l'organisation qui en résulte à l'échelle globale, alors que la modélisation est réalisée à l'échelle individuelle.

Nous simulons un système simple à base de billes qui se déplacent en ligne droite et peuvent avoir un état parmi cinqs : Sain, Infecté, Contagieux, Remis et Mort, Pour les billes contagieux, elles peuvent, si elles entrent en contact avec une autre et sont infectées, infecter l'autre bille avec une certaine probabilité.

Analyse problèmatique

Dans le modèle des maladies infectieuses, la population est divisée en cinq états, au total. Neuf facteurs influent sur le processus de simulation : pour la maladie infectieuse elle-même : taux de transmission, période d'incubation, taux de mortalité(par le virus), temps de guérison, portée effective de la transmission du virus, et pour l'environnement : intensité de l'activité de la population, densité de population et nombre initial d'infections. Nous devons identifier tous ces facteurs qui peuvent affecter le processus d'interprétation au stade initial. Pour rapprocher la simulation de la situation réelle, nous laissons les deux facteurs temporels, le temps de guérison et la période d'incubation, être aléatoires et distribués en loi normale, nous introduisons donc deux variables supplémentaires correspondant à la variance de la distribution en loi normale des deux variables.

Pour l'impact des maladies infectieuses sur la population, nous examinons le taux de décès dans la population totale, la durée totale de l'action et le nombre de personnes touchées par le virus.

Le but de cette expérience est d'étudier les effets des maladies infectieuses sur une population fermée, avec trois objectifs spécifiques,

- Obtenir des changements quantitatifs dans le développement de chaque population au fil du temps
- Explorer l'impact de différents facteurs sur les populations
- L'impact des mesures prises par le gouvernement pour prévenir l'épidémie

Nous avons différents protocoles expérimentaux pour différents objectifs expérimentaux

Expérience 1

But de l'expérience : étudier l'évolution du nombre de populations individuelles en fonction de la propagation de l'épidémie.

Protocole expérimental : compter le nombre de personnes dans chaque population à chaque unité de temps et dessiner un graphique linéaire.

Expérience 2

Objectif expérimental : étudier l'effet de différents facteurs sur la population à la suite d'une maladie infectieuse.

L'objectif de l'expérience est assez général et nous avons divisé l'expérience en trois parties spécifiques

Partie I.

Objectif expérimental : les facteurs environnementaux provoquent des maladies infectieuses qui affectent les populations

Protocole expérimental.

Variables de contrôle, permettant de contrôler séparément les facteurs du virus elle même (taux de transmission, période d'incubation, taux de mortalité(par le virus), temps de guérison, portée effective de la transmission du virus,) constante et les facteurs environnementaux(intensité de l'activité de la population, densité de population et nombre initial d'infections.) changeants.

Partie II.

Objectif expérimental : L'effet des facteurs du virus elle même sur les populations

Protocole expérimental.

Variables de contrôle, contrôlant un environnement constant, faisant varier séparément les facteurs de maladie infectieuse per se

Partie III.

Objectif expérimental : étudier les facteurs qui rendent l'impact des maladies infectieuses sur les populations relativement important.

Protocole expérimental.

Variables de contrôle, filtrez d'abord les facteurs qui ont un impact plus important par comparaison visuelle.

ANOVA, l'identification des facteurs, la liste des différentes paires de niveaux, la collecte des données expérimentales et l'analyse des données pour l'ANOVA à l'aide du langage R.

Expérience 3

Objectif expérimental : L'impact des mesures de prévention de l'épidémie prises par le gouvernement sur l'épidémie.

Protocole expérimental.

Des expériences contrôlées avec différentes mesures, chacune comparée aux mesures non réalisées, pour explorer les moyens les plus appropriés.

Chaque scénario est bien sûr résumé en conséquence et nous ne les répéterons pas ici. Pour conclure, nous commençons par une analyse de l'implémentation du code, en insérant un petit interlude.

Implementation du code

Nous avons implémenté le code en python3.

Statut des billes

La foule est divisée en six catégories：Saine, Infecté non Cantagious, Contagious, retiré et mort,plus les personnes que nous avons mises en quarantaine après.

Statut différent

```
ball_color_red=(255,0,0) #Contagious
ball_color_reddark=(100,0,0) #Contagious mais dans confinement
ball_color_black=(0,0,0)#Mort
ball_color_blue=(0,0,255)#Infecté mais non Cantagious
ball_color_green=(0,255,0)#Retiré
ball_color_grey=(150,150,150)#Saine
status =
(ball_color_grey,ball_color_blue,ball_color_red,ball_color_green,ball_color_black
,ball_color_reddark)
```

Nos billes ont un total de six variables qui sontball=[postion_x , postion_y,speed_x,speed_y,status,time]， Le statut est l'état de la balle et la variable time représente le temps restant pour rétirer lorsque la balle est Contagieuse et Infectée non contagieuse.

Voici le code permettant d'attribuer des valeurs initiales aux billes , balls est une liste de toutes les billes formées

```
t = random.random#set initial speed direction(random)
m = speed*random.random()#set initial speed magnitude
balls.append([pos_x , pos_y , math.cos(2*math.pi*t)*m,math.sin(2*math.pi*t)*m,
status[1],time ]
```

La vitesse de la balle dépend de la variable speed, est dans n'importe quelle direction.mais pour s'assurer que nous déterminons ensuite si la bille est en contact, nous fixons cette variable speed pour qu'elle ne soit pas supérieure au rayon de la balle.

Mouvement des billes

Nous utilisons le paquet pygame pour déplacer la balle.

Le mouvement des billes est obtenu en dessinant la position de chaque bille sur un canevas à une certaine fréquence, une fois toutes les 0,01s, et en changeant constamment la position des billes.

```
while true :

    for i in range(quantity):
            pygame.draw.circle(screen , balls[i][4] , [balls[i][0],balls[i][1]]
,ball_radius)#draw balls in the screen
    pygame.display.flip()   #update
    pygame.time.delay(10)
    balls[i][0] += balls[i][2] #move the balls
    balls[i][0] += balls[i][2]
```

Après avoir présenté les principes du mouvement du bille, explorons la façon dont les billes se déplace.

Nous pouvons réaliser deux modèles dans lesquels les billes n'entrent pas en collision et dans lesquels elles entrent en collision.

Aucune collision signifie que les billes coïncident et que la situation est simple, seule la collision avec la frontière doit être notée .

liste représente la liste des billes, pour la i-ème bille.

```
if liste[i][0] >= width - ball_radius or liste[i][0] <= ball_radius #boundary
condition

            #cut off the repeat area ,rectify the position
            while  liste[i][0] >= width - ball_radius or liste[i][0] <=
ball_radius:
                liste[i][0] -= liste[i][2]
                liste[i][1] -= liste[i][3]

            liste[i][2] = -liste[i][2] #change the direction of speed at the x-
axis

if liste[i][1] >= height-ball_radius or liste[i][1] <= ball_radius :
            #cut off the repeat area ,rectify the position
            while liste[i][1] >= height-ball_radius or liste[i][1] <=
ball_radius:
                liste[i][1] -= liste[i][3]
                liste[i][0] -= liste[i][2]

            liste[i][3] = -liste[i][3]#change the direction of speed at the x-
axis
```

Lorsqu'une collision se produit, nous supposons que toutes les billes ont des masses égales, que leurs volumes sont donc négligeables et qu'aucune énergie n'est dissipée, ce qui signifie que les sphères sont dans une collision parfaitement élastique. En supposant que les masses sont toutes égales et que les volumes sont négligeables, les boules entrent en collision sous la forme d'un échange de vitesses.

dist(balls[i], balls[j]) Distance entre deux billes，module(balls[i]) Longueur de la bille à partir de l'origine

Pour réduire la complexité de l'opération et déterminer plus facilement si les boules sont entrées en collision, nous trions la taille de la distance des boules par rapport à l'origine. Nous utilisons une sorte de rapide.

```
def partition(arr, low, high):
    i = low
    j = high         # index of smaller element
    pivot = module(arr[high])     # pivot

    while i <=j :
        while module(arr[i]) < pivot and i<= high :
            i += 1
        while module(arr[j]) >= pivot and j>= low:
            j -= 1
        if i<j :
            arr[i], arr[j] = arr[j], arr[i]
            i = i+1
            j= j-1

    tem =arr[i]
    arr[i]=arr[high]
    arr[high]= tem
    return i
def quick_sort(arr, low, high):
    if low  >= high:
        return arr
    if low < high:

        # pi is partitioning index, arr[p] is now
        # at right place
        pi = partition(arr, low, high)

        # Separately sort elements before
        # partition and after partition
        quick_sort(arr, low, pi-1)
        quick_sort(arr, pi+1, high)
        return arr
```

Premièrement, nous déterminons si les billes sont entrées en collision, et deuxièmement, lorsque les billes sont entrées en collision de sorte que leurs vitesses sont échangées, liste représente un groupe des billes, nous vérifions d'abord pour chaque bille si elle est entrée en collision avec une bille voisine. Si une collision se produit, il faut d'abord échanger les vitesses des billes, puis corriger les positions des billes pour les ramener à la position avant la collision. S'il y a une collision au même moment, choisissez celle qui est la plus proche de vous pour échanger les vitesses, puis corrigez toutes les boules en collision pour qu'elles retrouvent la position dans laquelle elles étaient avant la collision.

```
def elastic_collision(liste):#liste is already sorted and we will take the
elastic collision in count

    for i in range(quantity):
```

```
        if liste[i][4] != status[4]  and liste[i][0] < width - ball_radius and
liste[i][0] > ball_radius and liste[i][1] < height-ball_radius and liste[i][1] >
ball_radius :# don't collide with dead one :when the ball bumps into the wall
don't collide with other ball

                j=1
                while (i+j<len(liste) and (liste[i][4] == status[4] or
dist(liste[i], liste[i+j]) > ball_radius*2) ) :
                    j=j+1

                if i+j < len(liste):
                     #cut off the repeat area ,rectify the position
                    while i+j < len(liste) and dist(liste[i], liste[i+j]) <=
ball_radius*2:
                        liste[i+j][0] -= liste[i+j][2]      #move the ball
                        liste[i+j][1] -= liste[i+j][3]

                    liste[i][2] ,liste[i+j][2] = liste[i+j][2] , liste[i][2]#swap
speed with nearest ball
                    liste[i][3] ,liste[i+j][3] = liste[i+j][3] , liste[i][3]
                    #cut off the repeat area ,rectify the position
                    while i+j < len(liste):
                        while  dist(liste[i], liste[i+j]) <= ball_radius*2:
                            liste[i+j][0] -= liste[i+j][2]      #move the ball
                            liste[i+j][1] -= liste[i+j][3]
                        j+=1

    return liste
```

J'ai eu un bug avec mon propre programme lorsque plus de trois balles se heurtaient parfois.

L'une d'elles est que lorsque les boules corrigent leur position, si elles sont trop lentes, elles resteront coincées dans de nombreuses boucles, il n'y a aucun moyen d'éviter cela. Deuxièmement, certaines boules se déplaçaient soudainement vers le mur lorsqu'elles entraient en collision, puis restaient coincées dans le mur, et je ne comprends toujours pas pourquoi c'est un problème.

Bien que les simulations que j'ai vues sur le site Web recommandé par l'enseignant suivent un modèle de collision entièrement élastique pour le mouvement du blob, je ne pense pas que cela ait beaucoup de sens, si ce n'est d'augmenter la confusion du blob, et il pourrait être plus facile de calculer les chances d'infection, mais cela peut également être fait dans le modèle sans collision, et je montrerai ci-dessous comment j'ai résolu ce problème, principalement en attribuant le taux d'infection à chaque moment, bien que ce modèle soit plus aléatoire et que les chances d'infection dépendent surtout de la vitesse du blob.

Dans le modèle de non-collision, nous considérons le centre de chaque tache comme une personne et la zone occupée par la tache comme la portée effective de la transmission du virus. Nous choisissons donc de déplacer les sphères sans collisions.

Intensité de l'activité de la population

Ce niveau d'intensité de l'activité de la population est directement contrôlé par la vitesse à laquelle les billes se déplace, et nous contrôlons la magnitude de la vitesse initiale pour contrôler le niveau d'intensité de l'activité de la population.

Taux d'infection

Lorsque le bille infectieuse rencontre une autre bille, il y a une certaine probabilité de infection, nous appelons cette probabilité le taux de infection.

L'application de cette infection est quelque peu délicate lorsque la balle se déplace. Afin de ne pas compliquer le programme en introduisant des variables supplémentaires, nous attribuons la probabilité de infection à chaque fois que la balle change de position (l'ordinateur exécute une fois dans la boule qui dessine la bille)

```
def infection():#La probabilité qu'une personne en bonne santé soit infectée,mais
retourner 0 ou 1 signifie infecte ou pas, taux de transmision
    if (random.random()< 1-math.pow(1-ir,1/(((8*ball_radius)/speed)*math.pi))):
return 1
    else : return 0
```

Cependant, nous constatons que la situation réelle est souvent légèrement inférieure, car certaines balles se rencontrent mais il le prends pas en compte, car il y a un écart entre les moments et si le temps de rencontré est plus petite que cette écarte, il prends pas en compte. L'ordinateur calcule le temps de rencontre pour qu'il soit inférieur à la valeur théorique, de sorte que le taux de infection réel à chaque instant est inférieur à la valeur théorique, et plus la vitesse est élevée, plus la valeur réelle est faible. Si nous prenons en compte cette erreur, le temps moyen devient $\frac{8R\pi}{speed} + 1$

```
def infection():#La probabilité qu'une personne en bonne santé soit infectée,mais
retourner 0 ou 1 signifie infecte ou pas, taux de transmision
    if (random.random()< 1-math.pow(1-ir,1/((8*ball_radius)/(speed*math.pi)+1))):
return 1
    else : return 0
```

Un résultat de 1 signifie que la balle a été infectée à ce moment-là. Un résultat de 0 signifie que la balle n'a pas été infectée à ce moment-là.

Période d'incubation, temps de guérison

Plus près de la réalité, la période d'incubation et le temps de guérison sont modélisés comme étant distribués en loi normale, de sorte que la moyenne et la variance des deux variables sont nécessaires respectivement.

ht_mean,ht_var Moyenne et variance correspondant au temps de guérison, respectivement

ip_mean,ip_var Moyenne et variance correspondant à la période d'incubation respectivement

```
def healing_time():#temps de guérison qui suis loi de gausiennne
    return round (ht_mean + ht_var*
(math.sqrt(-2*math.log(random.random()))*math.cos(2*math.pi*random.random())))

def incubation_period():#période de incubation qui suis loi de gausiennne
    return round (ip_mean + ip_var*
(math.sqrt(-2*math.log(random.random()))*math.cos(2*math.pi*random.random())))
```

Nous prenons 500 passages (dans la boucle pour dessiner le bille) comme unité de temps, donc ici

```
ht_mean = 5000 # the mean time of the healing
ht_var = 100   # the ecart type  of the healing time
ip_mean = 1000  # the mean time of incubation period
ip_var = 100    #the ecart type of the incubation period
```

Le temps moyen de guérison est de dix unités de temps et la période d'incubation, moyenne est de deux unités de temps.

Taux d'incubation, taux de mortalité

De la même manière que le taux de infection est appliqué, nous distribuons le taux d'incubation et le taux de maladie et de décès de manière égale à chaque instant (l'ordinateur s'exécute une fois dans la boucle de dessin des billes). Il est important de noter ici que cr = 1 - taux de'incubation.

```
def death():#taux de mortalité, mais retourner 0 ou 1 ca signifie mort ou pas
    if (random.random()< 1 - math.pow(1-dr,1/ht_mean)): return 1 # pendant le
temps de guésion , taux de mortalite a chaque instant t0
    else : return 0
def contagion():# La probabilité qu'une personne infectée devienne une personne
sont contagieuse , taux de incubation a chaque instant

    if (random.random()< 1 - math.pow(1-cr,1/ip_mean)): return 1
    else : return 0
```

Le résultat est le même que le taux d'infection. Si le résultat est 1, alors le moment est mort ou devient contagieux. 0, alors le moment n'est pas mort ou la période d'incubation n'est pas passée.

Changement d'état de la bille

Nous bouclons une fois (l'ordinateur exécute une fois la boucle qui dessine la bille) pour modifier l'état de la billes en termes de position et d'état.

La position permet de déterminer si la limite est atteinte, et si oui, le rebond se produit. liste est toujours une liste des balles. Pour la balle i, le premier et le deuxième élément représentent la position, le troisième et le quatrième la vitesse.

```
if liste[i][0] >= width - ball_radius or liste[i][0] <= ball_radius :
 #boundary condition
            #cut off the repeat area ,rectify the position when choc elastic
modeal
            if  liste[i][0] >= width - ball_radius or liste[i][0] <= ball_radius:
                liste[i][0] -= liste[i][2]
                liste[i][1] -= liste[i][3]
            liste[i][2] = -liste[i][2]
if liste[i][1] >= height-ball_radius or liste[i][1] <= ball_radius :
            #cut off the repeat area ,rectify the position when choc elastic
modeal
            if  liste[i][1] >= height-ball_radius or liste[i][1] <= ball_radius:
                liste[i][1] -= liste[i][3]
                liste[i][0] -= liste[i][2]
            liste[i][3] = -liste[i][3]
```

L'état est déterminé une fois pour chaque boule et il existe différentes actions pour l'état correspondant.

La première étape consiste à déterminer si le statut de la bille est infecté non contagieux, c'est-à-dire le status[1], est devenue contagieux, si elle est devenue contagieux avant de déterminer si elle veut être isolée, et si elle n'est pas contagieux alors elle est à un pas de passer la période d'incubation.

```
        if liste[i][4] == status[1] :# case infect mais non contagious
            if contagion()==1 :

                liste[i][4] = status[2]
                liste[i][5] = healing_time()

                if confinement() == 1:
                    liste[i][4] = status[5]
                    liste[i][2] = 0
                    liste[i][3] = 0


            else :
                liste[i][5] -= 1
                if liste[i][5] <= 0 : liste[i][4] = status[3]
```

Pour le statut de la bille comme contagieux, c'est-à-dire le status[2].

Premièrement, déterminer s'il est mort à l'heure actuelle, deuxièmement, déterminer s'il est guéri, et si aucun de ces cas ne se produit, alors rencontrer un bille dans sa zone contagieux avec une certaine probabilité d'infection.

```
        if liste[i][4] == status[2] : #case contagious

            if death() ==1:
                liste[i][4] = status[4] #case death
                liste[i][2] = 0
                liste[i][3] = 0
                liste[i][5] = 0
            else:
                liste[i][5] -=1
                if liste[i][5] == 0 : liste[i][4] = status[3]     #case heal
```

```
                else:
                    j=0
                    for j in range (quantity):
                        if i!=j and dist(liste[i], liste[j]) < ball_radius and
liste[j] == status[0] and infection() == 1:# infect other balls
                            liste[j][4] = status[1]
                            liste[j][5] = incubation_period()
```

un état de bille est dans confinement, cet état est l'état bille ajouté ultérieurement en prenant des mesures d'isolement, c'est-à-dire status[5].

```
        if liste[i][4] == status[5] :# case confinement contagious
            if death() == 1:
                liste[i][4] = status[4]
                liste[i][5] = 0
            else:
                liste[i][5] -=1
                if liste[i][5] <= 0 : liste[i][4] = status[3]
```

Statistiques

Ici, nous utilisons le paquet matplotlib.pyplot pour tracer l'image

J'ai pris 500 passages (dans la boucle de dessin des boules) comme unité de temps pour collecter le nombre d'individus (ceci a été testé à plusieurs reprises). statistic(balls , data) regarde l'état des billes et collecte les données dans une liste de données.

```
if (t//500 == a): #registry the time and data
    time_list.append(a)
    statistic(balls,data)
    a=a+1
```

Les résultats montrent

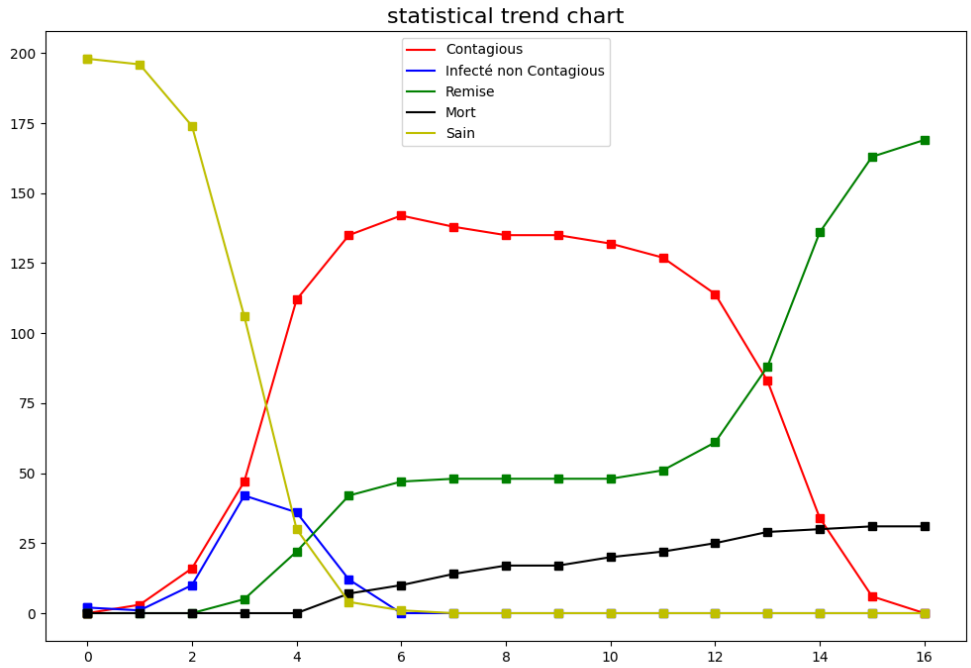
Pour la simulation, le lecteur est invité à compiler le code lui-même ; je ne ferai ici qu'une analyse des résultats.

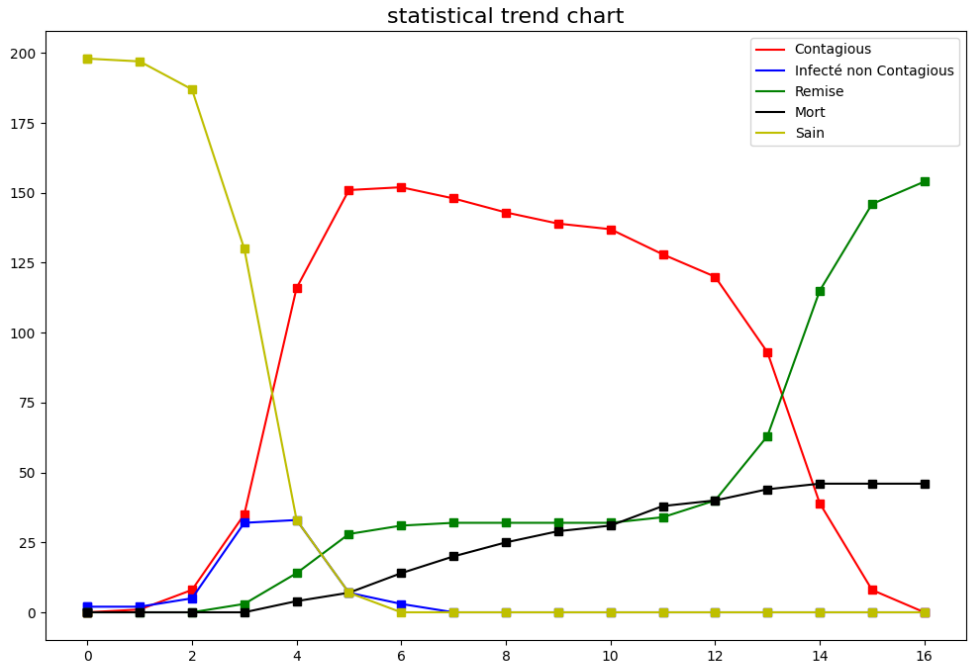
Nous fixons une série de valeurs initiales.

quantity200 ball_size10 confine_rate0 death_rate0.2 heal_time_mean10.0 var 0.2 contagious_rate0.8 incubation_Period_mean1.0 var 0.2 infect_rate0.7

Précision de la simulation

Pour cet série de valeurs initiales, nous effectuons deux simulations.





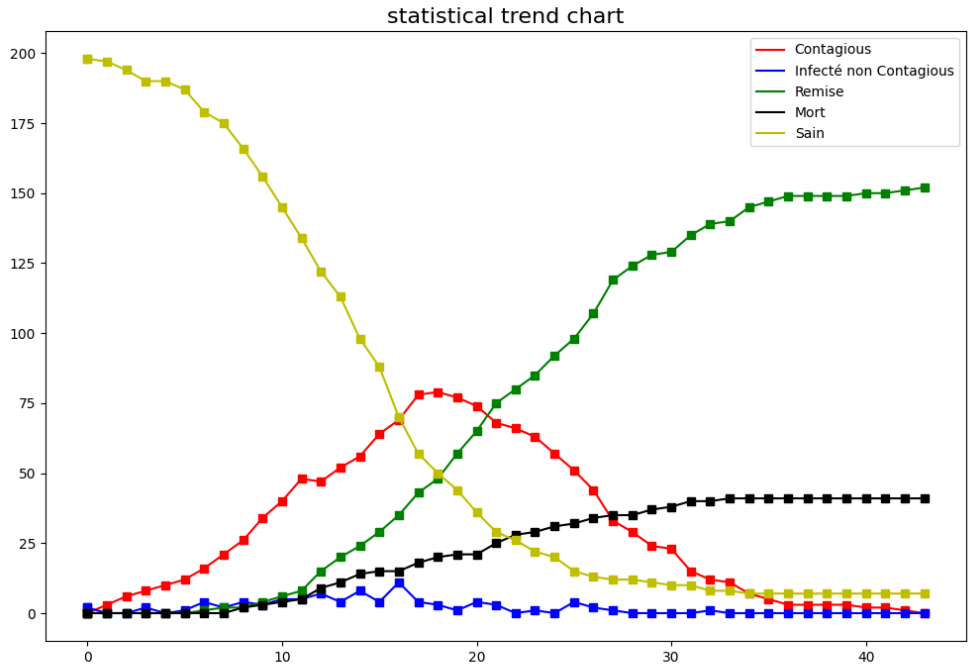
La ligne jaune indique le nombre de personnes infectées dans le temps, la ligne bleue indique le nombre de personnes infectées mais non contagieuses dans le temps, la ligne rouge indique le nombre de personnes contagieuses, la ligne verte indique le nombre total de personnes guéries dans le temps et la ligne noire indique le nombre de décès dans le temps.

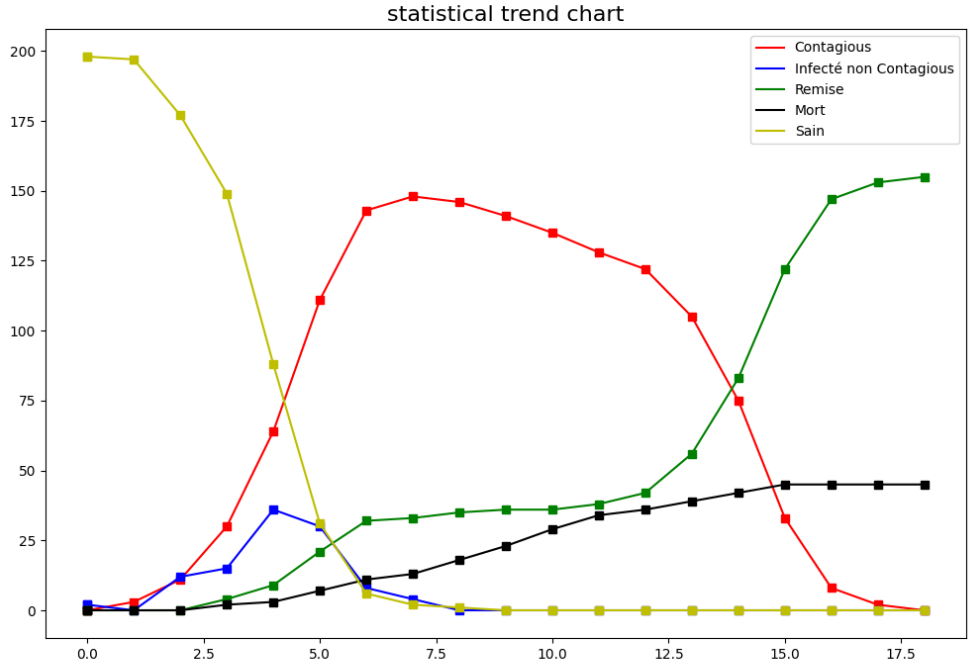
Tout d'abord, le nombre total de personnes restant le même à chaque instant, vérifions que notre méthode de simulation est correcte.

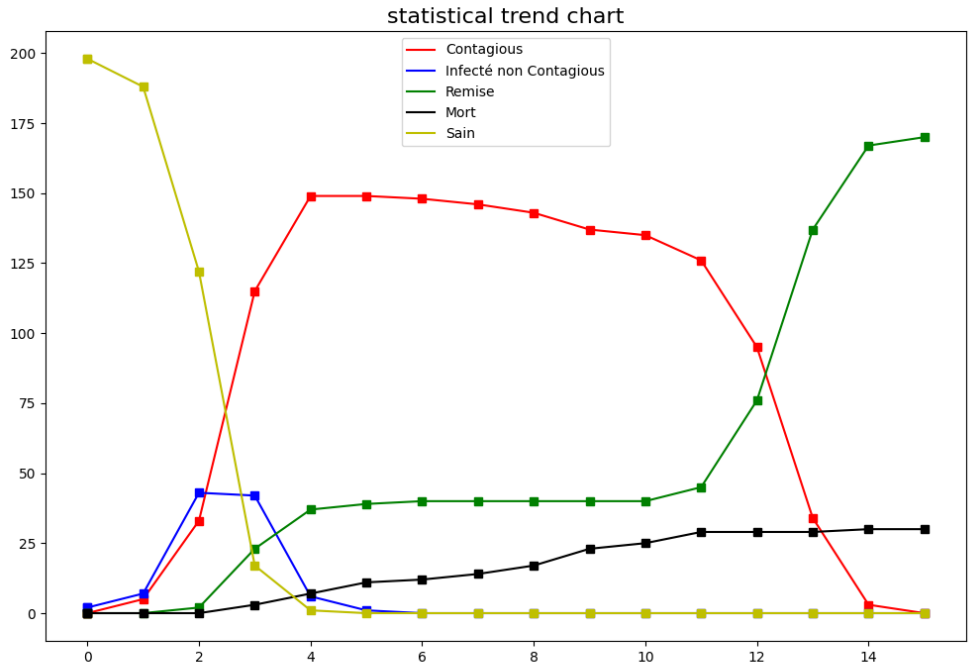
Lorsque nous examinons le taux de maladie et de décès, notre moyenne du nombre de décès / (nombre de guéris + nombre de décès) pour les deux tests est égale à 0,2 et il n'y a pas de divergence avec notre taux de mortalité(par le virus) fixé.

indiquant que notre modèle de calcul du taux de mortalité(par le virus), du taux d'infection et du taux d'incubation à chaque instant est correct.

Cependant, nous avons fixé le taux d'infection à 0,1, 0,5 et 1, respectivement.







Nous pouvons voir qu'au premier point de temps, le taux de infection de 0,5 infecté 3 ou 4 mais le taux de infection de 1 infecté d'environ 10 donc notre temps moyen est également correct:

Avec une période d'incubation moyenne d'une unité et un temps de guérison moyen de 10 unités, nous pouvons également examiner le ratio du nombre de personnes initialement infectées en bleu. Vous pouvez voir que 0,1 a beaucoup moins de chances d'être infecté que 0,5 et 1.

Vous pouvez vous demander pourquoi cela est correct alors que le nombre infecté de 0.5 n'est pas strictement le nombre infecté de 0.5 deux fois que dont de 1.

Tout d'abord, notre probabilité d'infection pour chaque bille est très affectée par sa vitesse, il y a donc une grande marge d'erreur car la vitesse des deux bille qui infecté a l'intiale est aléatoire et le nombre de billes rencontrés est également aléatoire, donc le nombre de personnes infectées au départ est très aléatoire.

Deuxièmement, selon le modèle que nous utilisons pour calculer le taux d'infection à chaque instant, il s'agit d'une fonction qui croît de manière exponentielle avec le temps moyen, et s'il y avait un vrai problème, il n'aurait pas un tel effet moyen, et vous trouveriez qu'il est soit très similaire au cas où le taux d'infection est de 1, soit très similaire au cas où le taux d'infection est de 0.

Bien sûr, si nous utilisons le modèle de collision, le caractère aléatoire au début est lié au nombre des billes rencontrées, le caractère aléatoire diminue et le rapport de linéarité au début est un peu plus précis.

On constate que dans le cas d'une population dense, la durée de la contagion diminue de façon exponentielle par rapport à la taille du taux de contagion. Plus le taux de contagion est faible, plus le temps vécu est long.

Analyse des facteurs

Nous avons suivi le schéma de la deuxième expérience, qui était divisée en trois parties

Facteurs environnementaux qui contribuent à l'impact des maladies infectieuses sur les populations

Effets de la densité de la foule

La densité de population est un facteur décisif pour la propagation d'un virus. Si la population est clairsemée, un virus ne peut pas se propager efficacement.

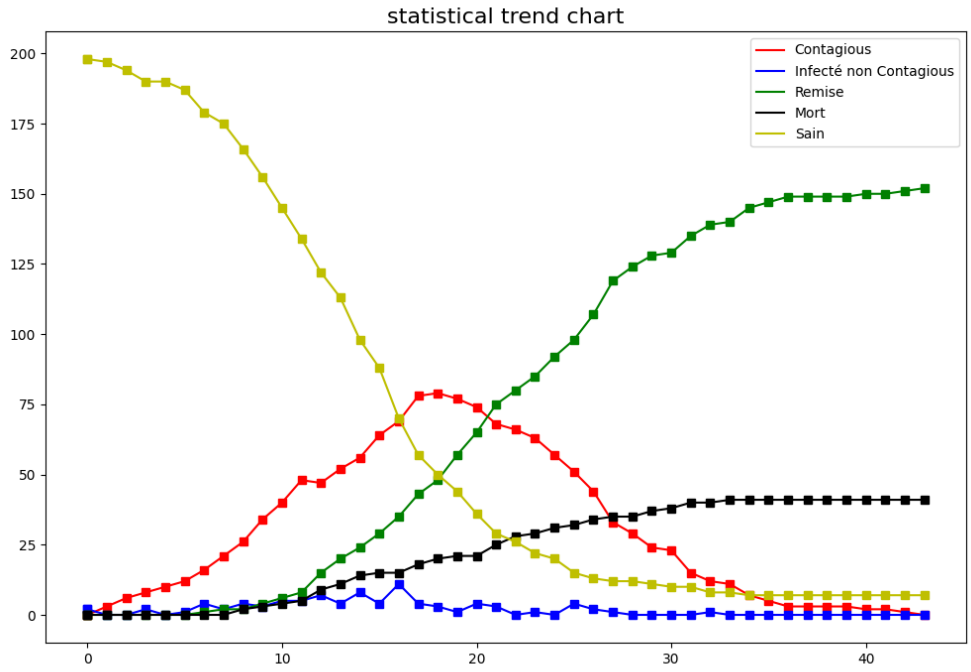
Choisissons un virus spécifique avec ses propres paramètres, à savoir

death_rate0.2 heal_time_mean10.0 var 0.2 contagious_rate0.8 incubation_Period_mean1.0 var 0.2 infect_rate0.1

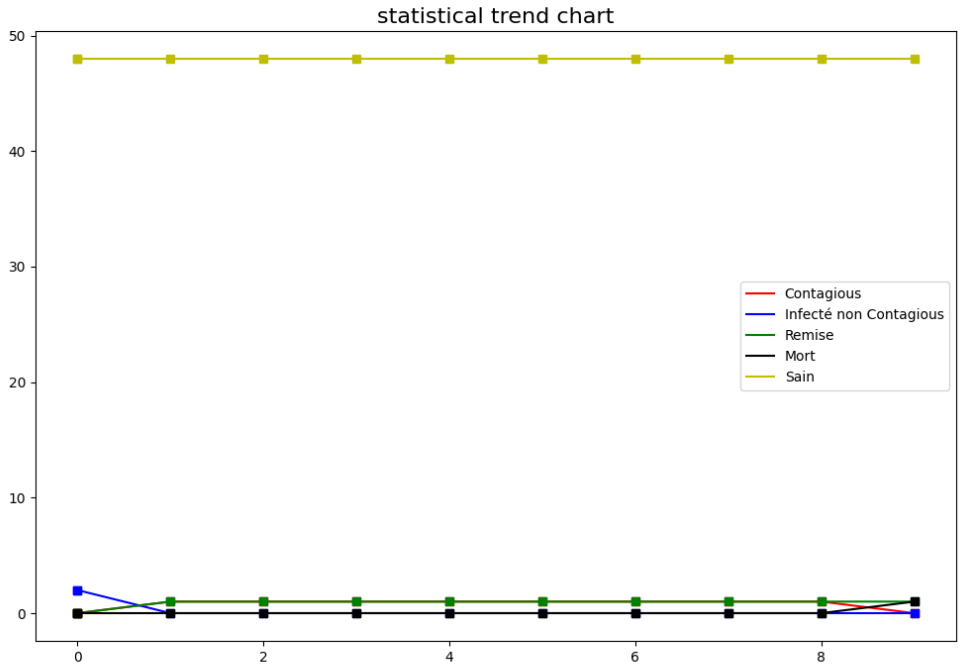
ball_size10 La portée de la propagation effective de son virus

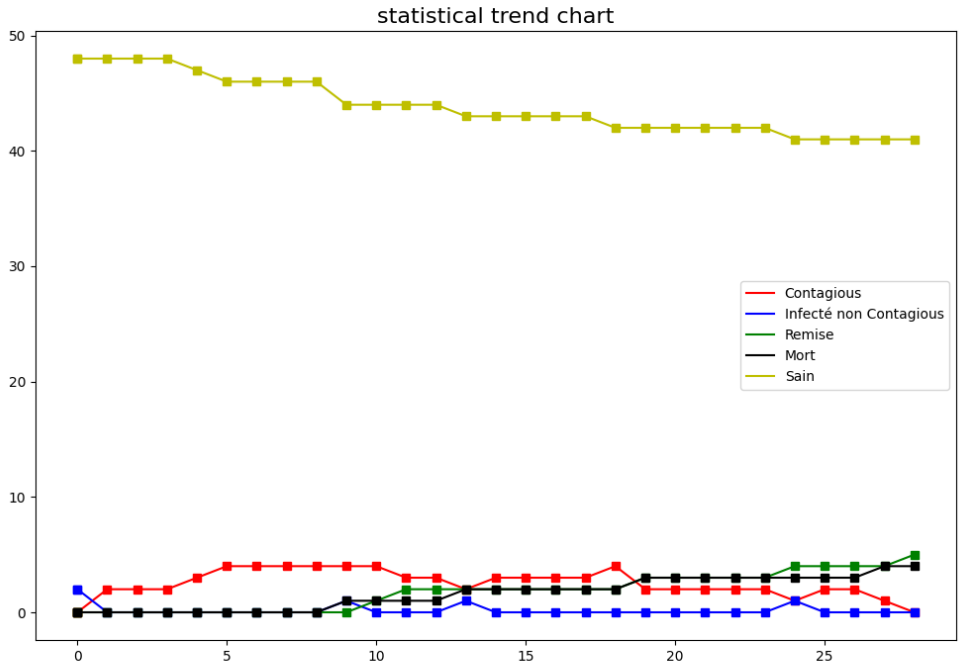
Le nombre initial d'infections est de 2.

Il y a 200 personnes dans une zone de 800*600. Le résultat est



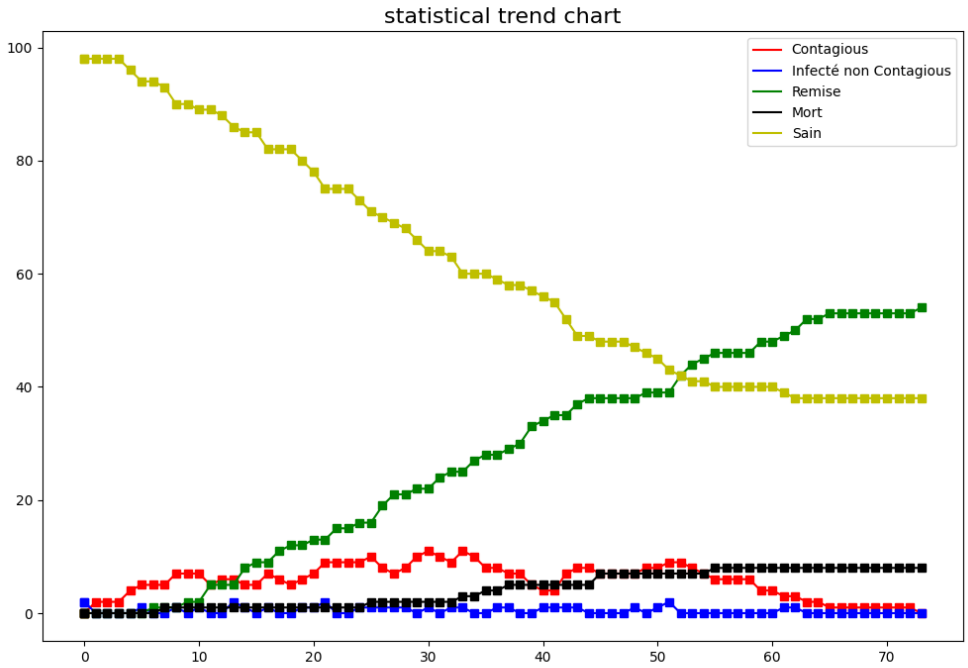
Nous changeons le nombre de personnes, mais la zone reste la même et le nombre de personnes est maintenant de 50.





Nous avons mené deux expériences, qui ont toutes deux montré que le virus ne s'est pas propagé efficacement.

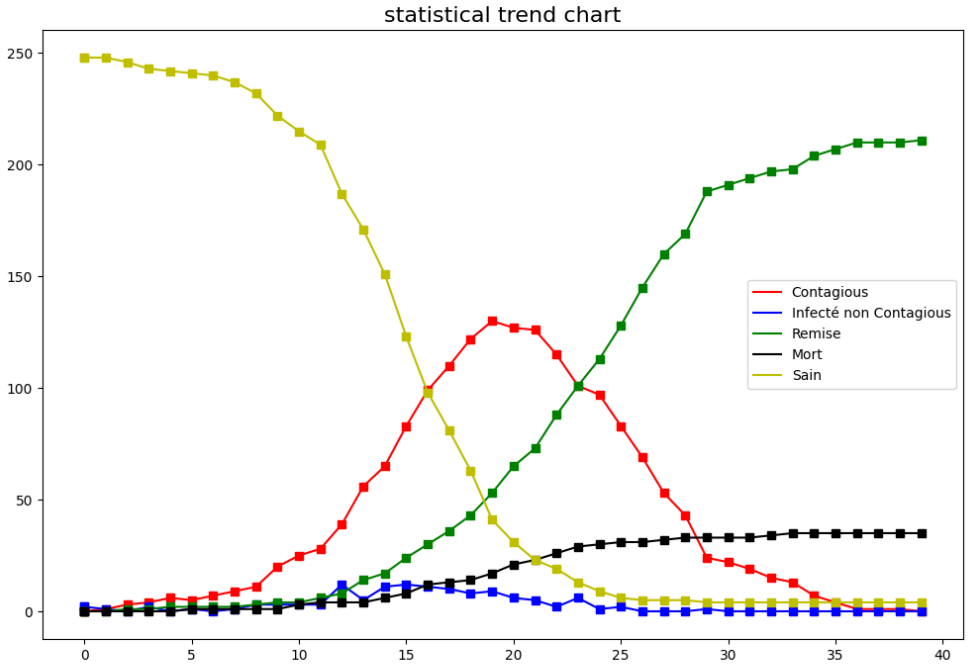
Gardons la même zone et augmentons le nombre de personnes à 100.



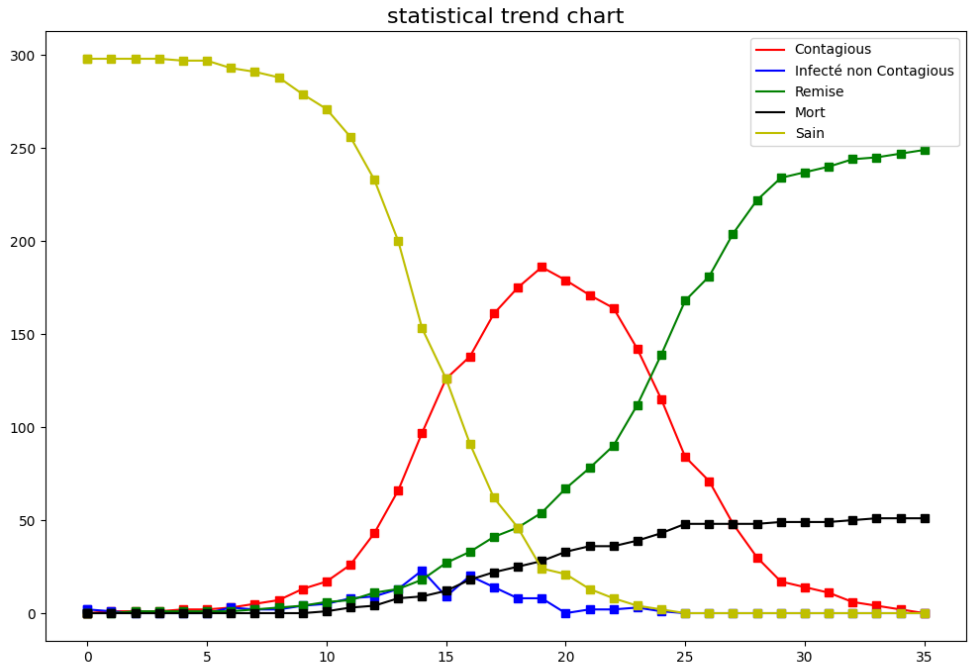
Cette fois, le virus peut encore survivre, il lui faut en moyenne 11 unités de temps, mais il peut maintenant durer jusqu'à 73 unités de temps, et il lui faut 10 unités de temps pour atteindre le premier pic du nombre de personnes infectées.

Ainsi, une densité de population de 100 est nécessaire pour que le virus soit efficace pour propager la maladie à plus de la moitié du nombre total de personnes.

Nous portons le nombre total de personnes à 250.



Nous avons révisé le nombre total de personnes à 300.



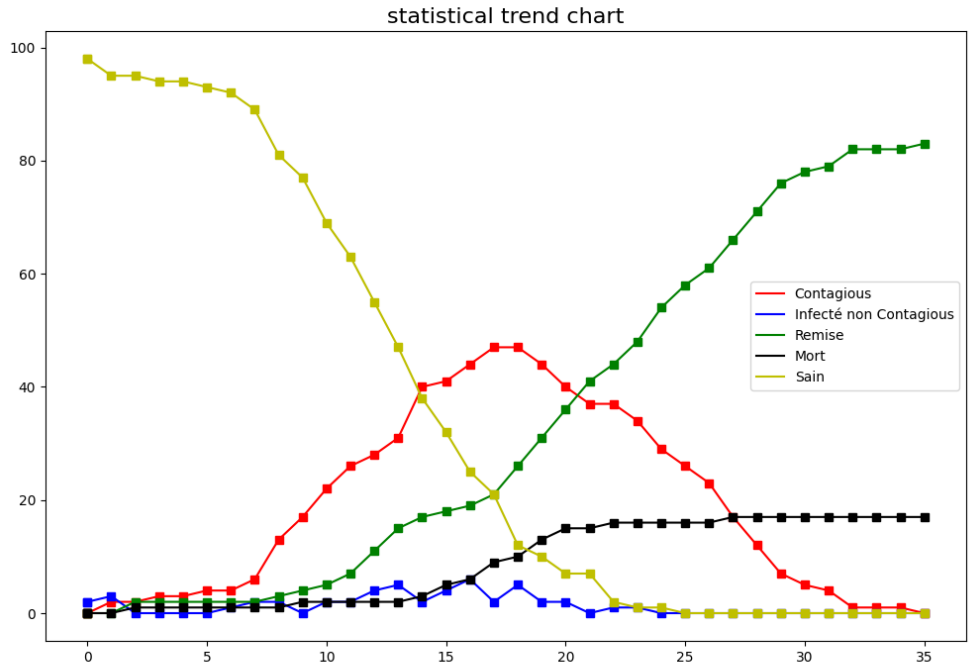
Nous constatons que la courbe coïncide essentiellement avec la taille de la population de 250 . À 20 unités de temps au pic du nombre de personnes infectieuses, et entre 35 et 40 unité de temps le virus disparaît et infecte toute la population.

Ainsi, par une série de comparaisons, nous trouvons que pour ce virus, avec une balle_taille 10 taux_de_mort0,2 heal_time_mean10,0 var 0,2 contagious_rate0,8 incubation_Period_mean1,0 var 0,2 infect_rate0,1, l'effet du virus sur la population varie en fonction de la densité de la population lorsque celle-ci se situe dans l'intervalle (100/800∗600,250/800∗600).

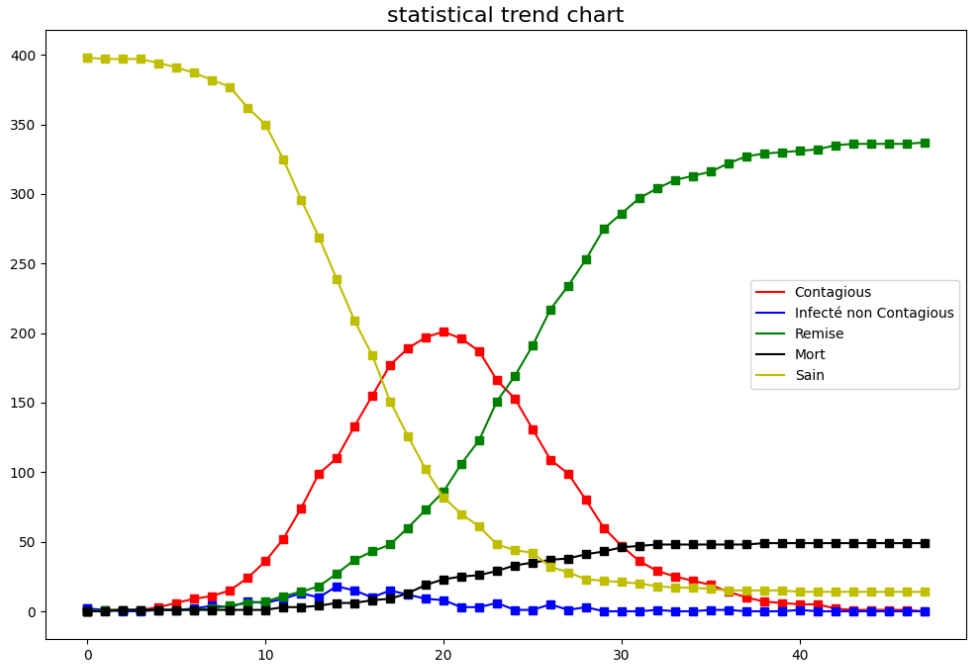
Par conséquent, pour un virus donné, la densité de population est un facteur déterminant de l'impact du virus sur la transmission de la population dans une certaine fourchette.

Impact de la taille du zone

Si nous réduisons la zone à la moitié 600∗400, la population devient 100, contrairement à l'image ci-dessus de 800∗600 avec une population de 200.



Nous avons ensuite augmenté la surface à 800*1200 et le nombre de personnes à 400.



La comparaison est faite avec une surface de 800∗600 et une population de 200 habitants, respectivement.

Dans la comparaison réduite, nous trouvons des tendances similaires en fonction du nombre de contagieux, atteignant toutes deux un pic à environ 20 unités de temps, mais prenant légèrement moins de temps pour y parvenir.

Dans la comparaison zoomée, nous constatons que la tendance en fonction du nombre de contagieux, est similaire, atteignant également un pic à 20 unités de temps, légèrement plus tard que dans les deux premiers cas, avec une légère augmentation de la durée de la présence de l'infection.

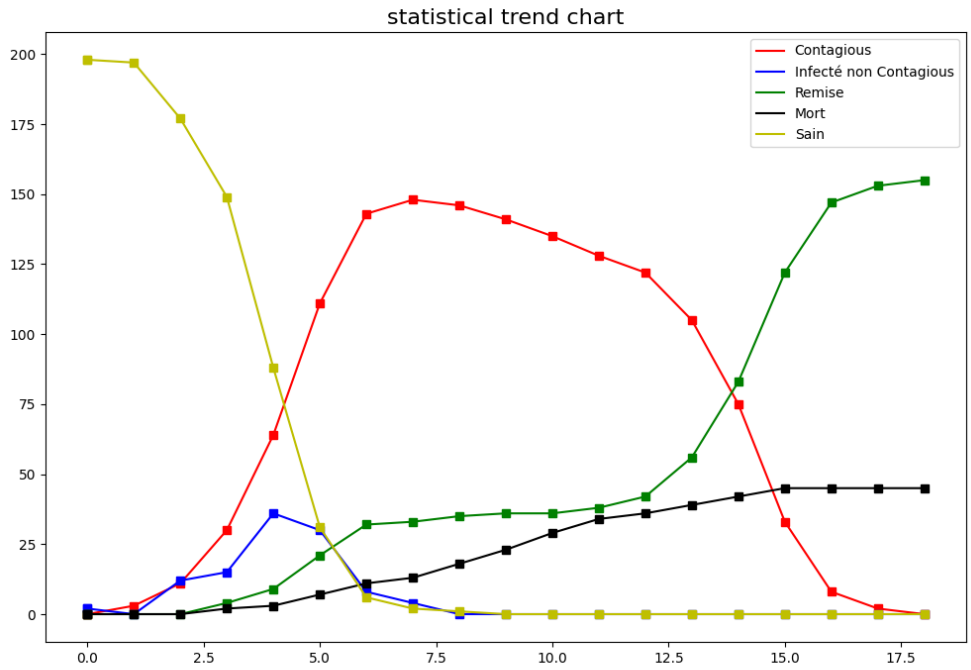
Nous constatons que plus la zone est grande, plus la maladie infectieuse existe longtemps.

L'impact des maladies infectieuses sur les populations en raison de la virulence de la maladie infectieuse elle-même.

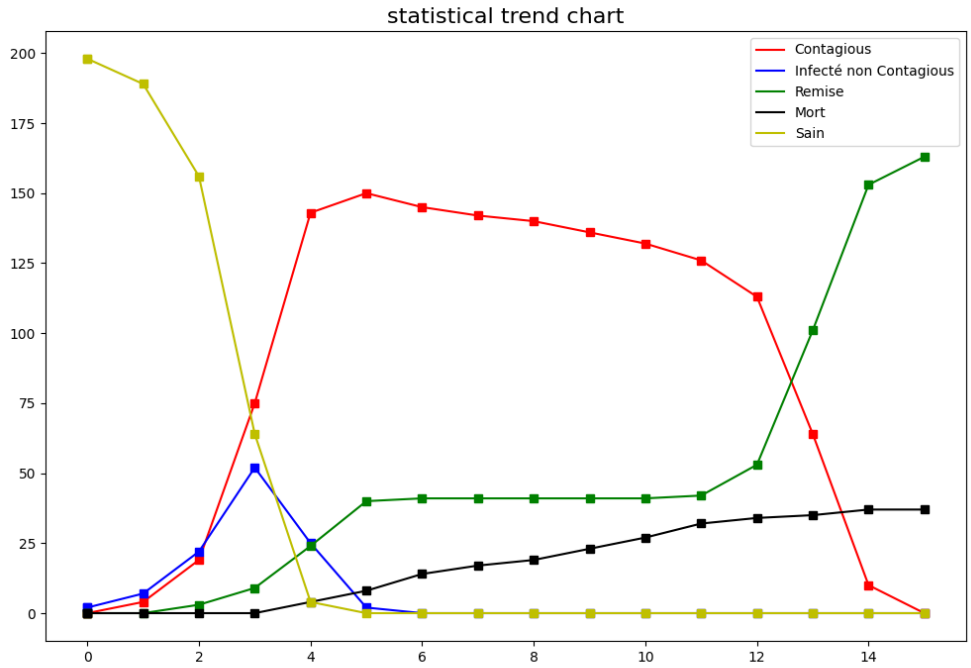
Nous avons choisi une zone de 800*600 et une population de 200 personnes pour la simulation. Le nombre initial de personnes infectées est de deux

Portée effective de la transmission du virus

Le facteur est simple : plus la portée effective du virus est grande, plus il a de chances de se propager. Avec un taux de transmission de 0,5, et d'autres facteurs étant certains.



Nous augmentons le rayon de la bille à 15

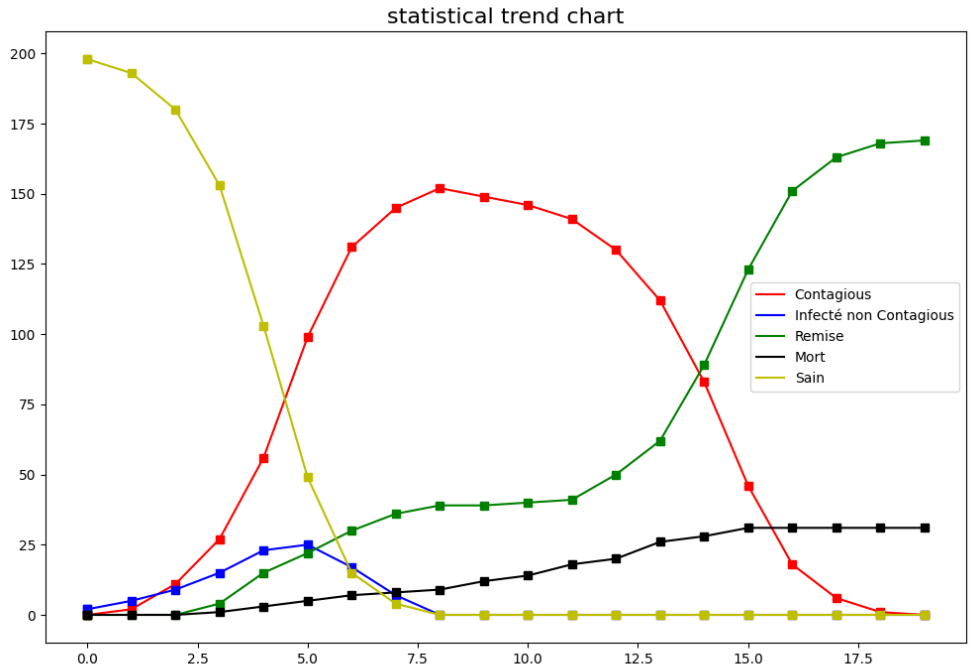


Nous constatons que le pic du nombre de personnes contagieuse est atteint plus tôt et que le temps vécu par la maladie infectieuse se raccourcit, ce qui témoigne d'un taux de transmission accéléré.

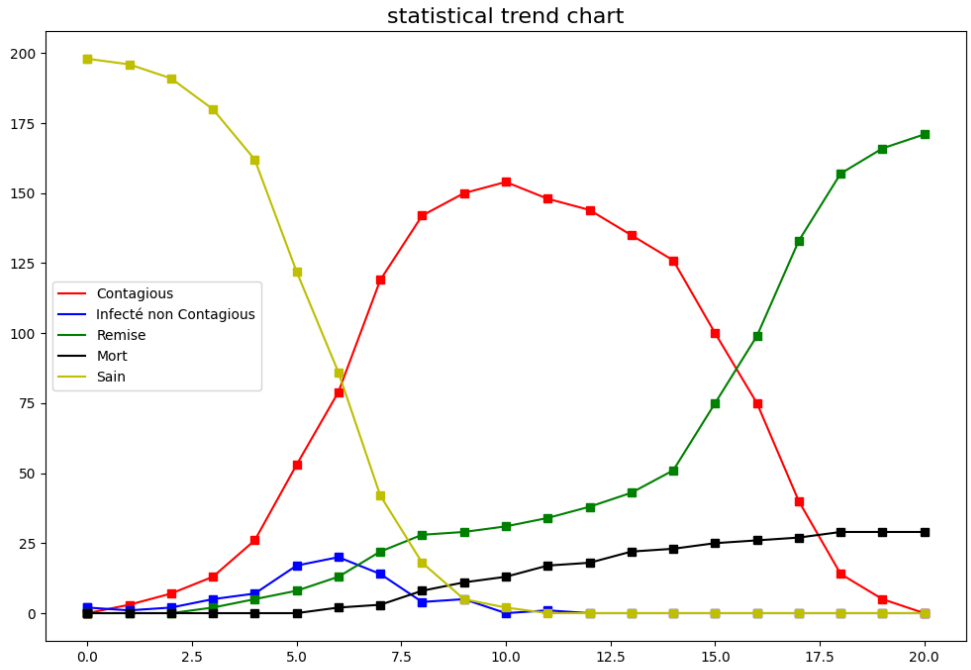
Nous testons un corollaire

Si, pour un nombre donné de personnes, les paramètres surface totale de la population/rayon effectif du virus au carré sont égaux, le résultat est le même

Nous choisissons 800*600 pour une population de 200 personnes et un rayon de transmission de 9 %.



Nous avons choisi 400*600 avec une population de 200 et un rayon d'infection de 3



On peut voir que, dans la marge d'erreur, les deux images fonctionnelles se rapprochent l'une de l'autre. Par conséquent, si la surface totale de la population/le carré du rayon effectif de transmission du virus est égal pour un nombre donné de personnes et que les paramètres du virus sont égaux, les résultats sont les mêmes.

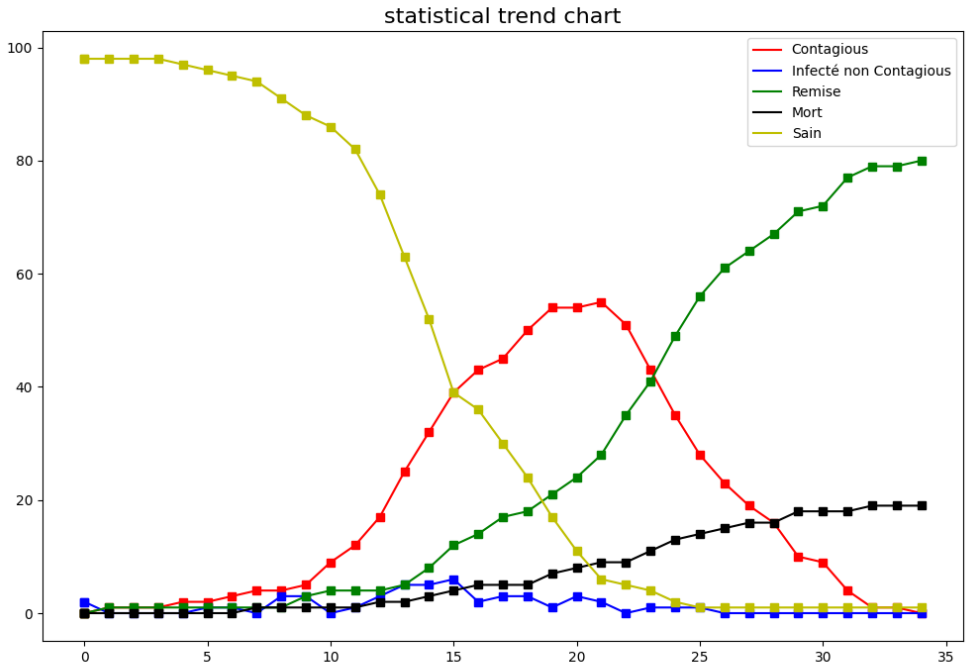
Taux de mortalité(par le virus)

Si nous prenons 1 comme taux de mortalité, alors le taux de mortalité à chaque instant est toujours de 1. Je ne comprends pas quelle distribution de probabilité utiliser pour construire le taux de mortalité à chaque instant, et il n'y a aucune explication dans la littérature, donc je dois en rester là.

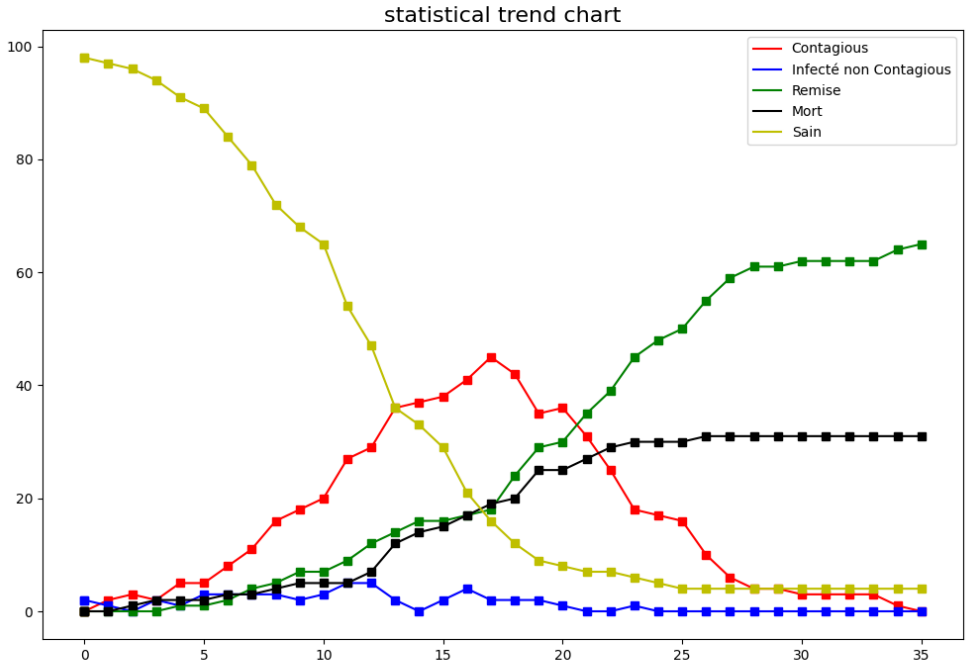
Nous suivons donc également une distribution uniforme de la probabilité de décès à chaque instant de la durée de la cure, initialement simulée avec les mêmes paramètres viraux que précédemment

ball_size 10 death_rate0.2 heal_time_mean10.0 var 0.2 contagious_rate0.8 incubation_Period_mean1.0 var 0.2 infect_rate0.2,

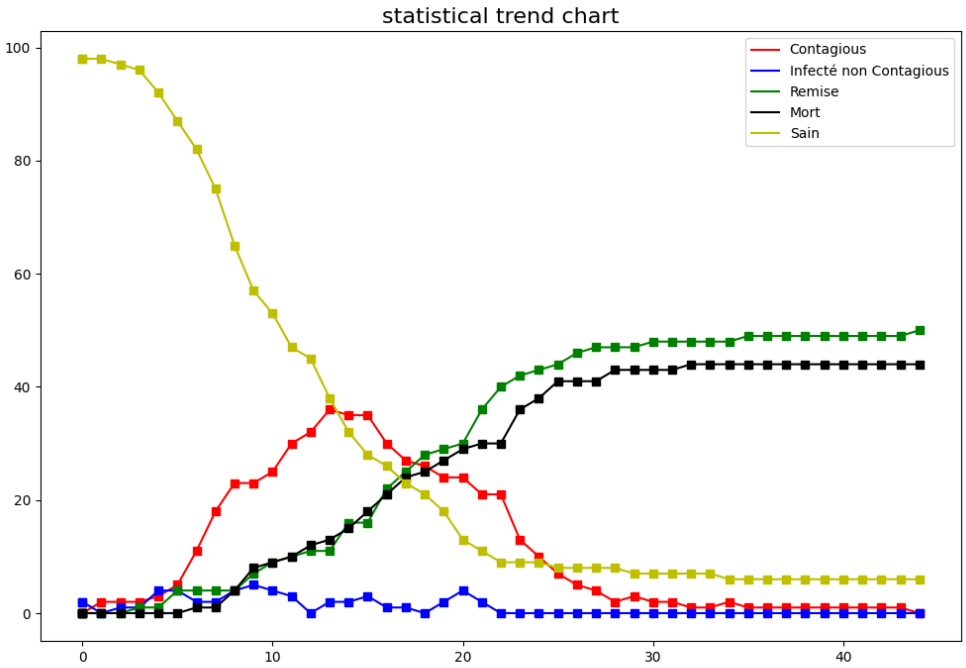
Mais nous diminuons la densité de la population et modulons 100



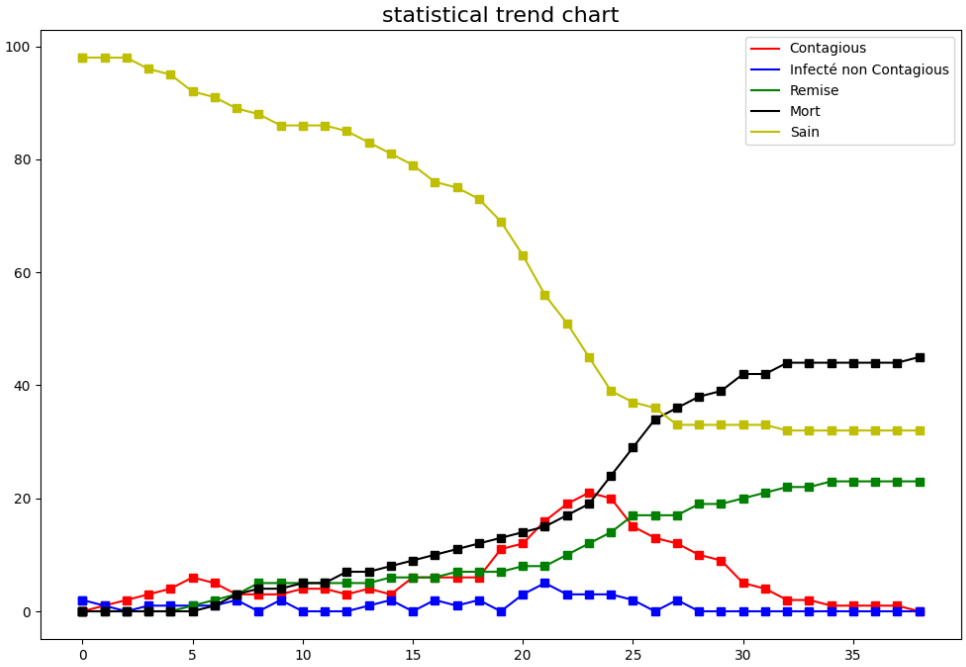
Nous avons ensuite fixé le taux de mortalité virale à 0,4.



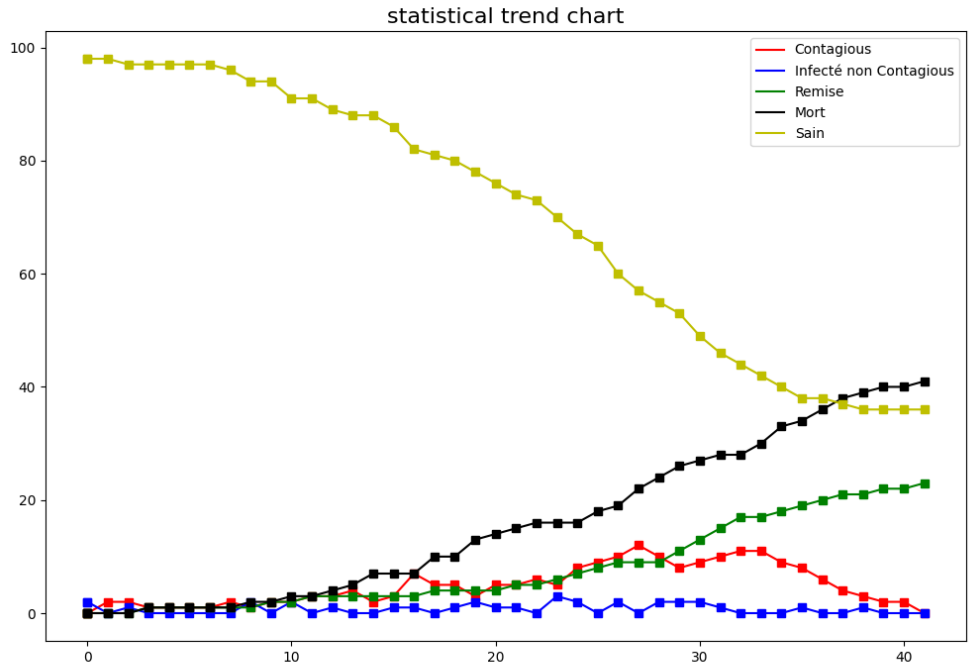
Nous avons ensuite fixé le taux de mortalité virale à 0,6



Nous avons ensuite fixé le taux de mortalité virale à 0,8.



Nous avons ensuite fixé le taux de mortalité virale à 0,9.



L'image d'une fonction avec un taux de mortalité virale de 0,9 est le résultat de quatre essais avant que je ne dispose de ce temps total relativement long. On peut donc voir qu'un taux de mortalité virale plus important affecte plutôt la propagation du virus, provoquant une tendance à la baisse du nombre total de décès.

Taux de infection

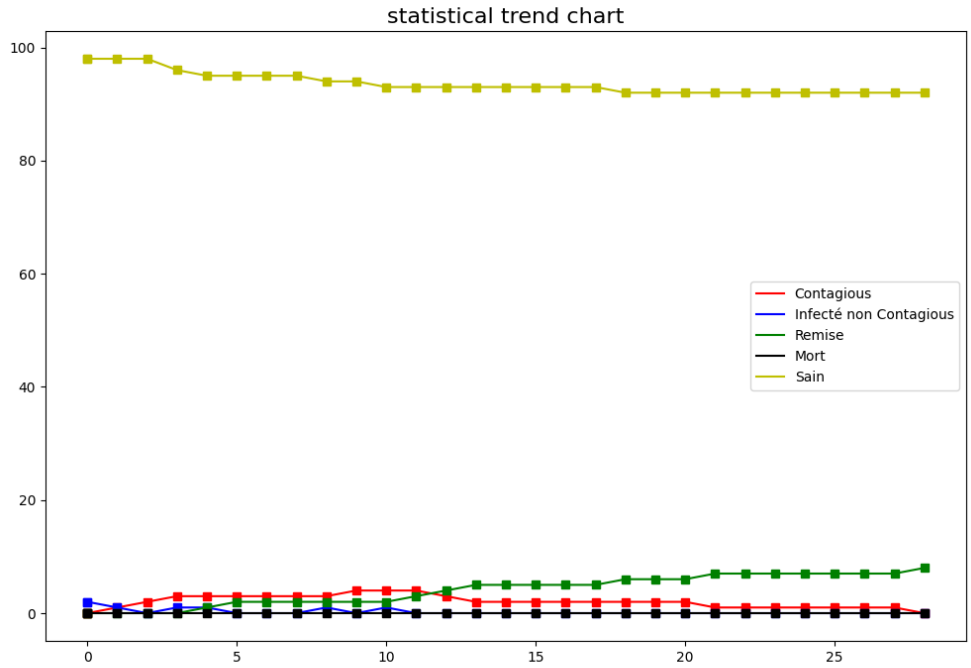
Notre simulation du taux de transmission a été réalisée dans le module précédent et il est clair que plus le taux de infection est élevé, plus la propagation est rapide et plus vite le nombre maximal de personnes capables de transmettre est atteint.

Nous comparons les taux de infection de 0.05, 0.1, 0.2, 0.5 et leur durée totale de infection

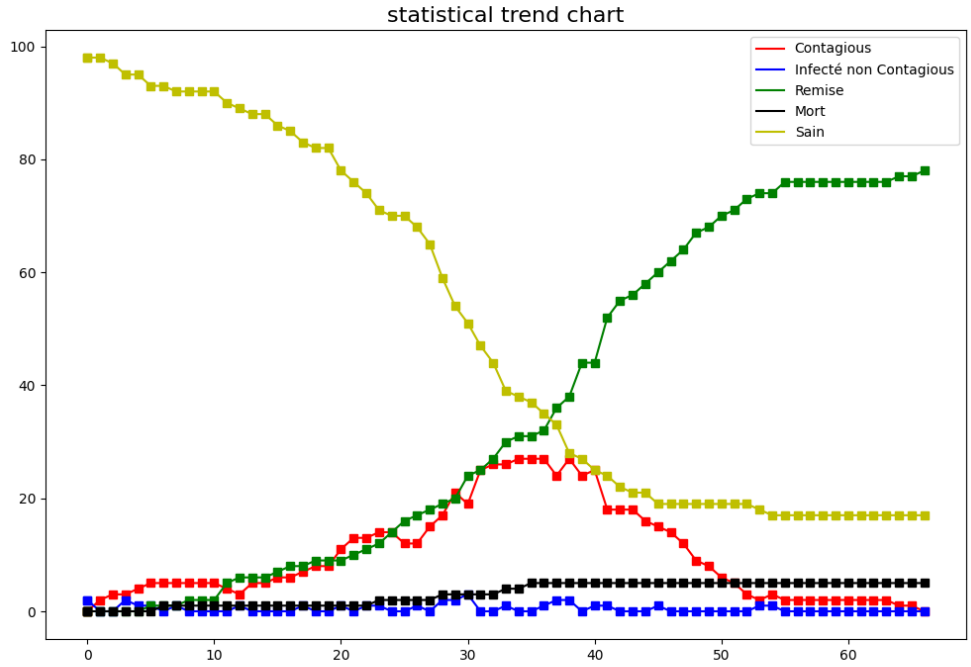
Rapport entre la densité de population et la propagation du virus : 100/800*600 /10**2

Les autres paramètres du virus sont death_rate0.1 heal_time10.0 0.2 contagious_rate0.8 incubation_Period1.0 0.2

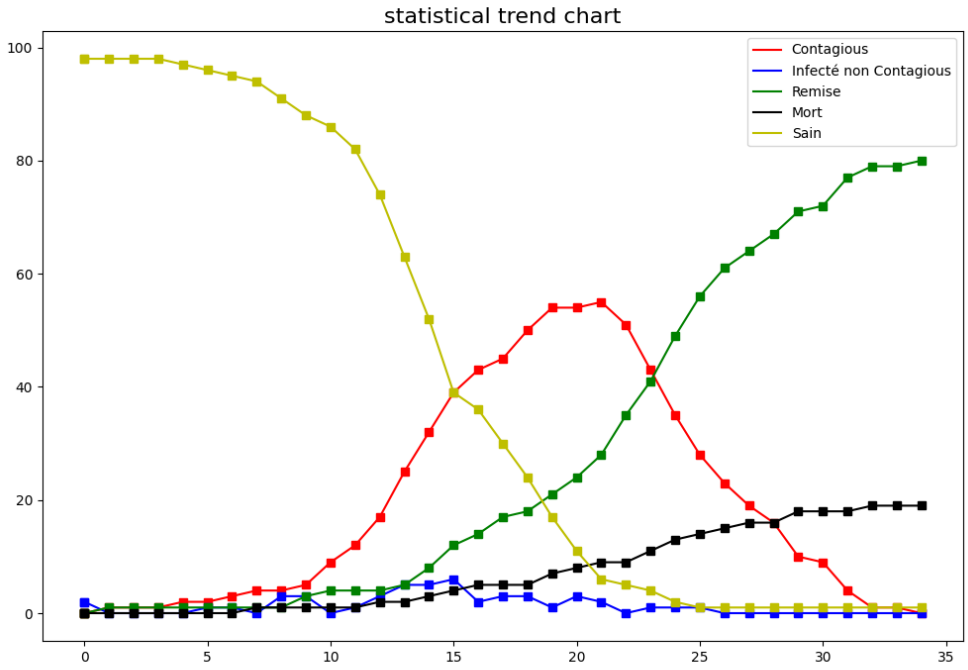
0.05


statistical trend chart
Contagious
Infecté non Contagious
Remise
Mort
Sain

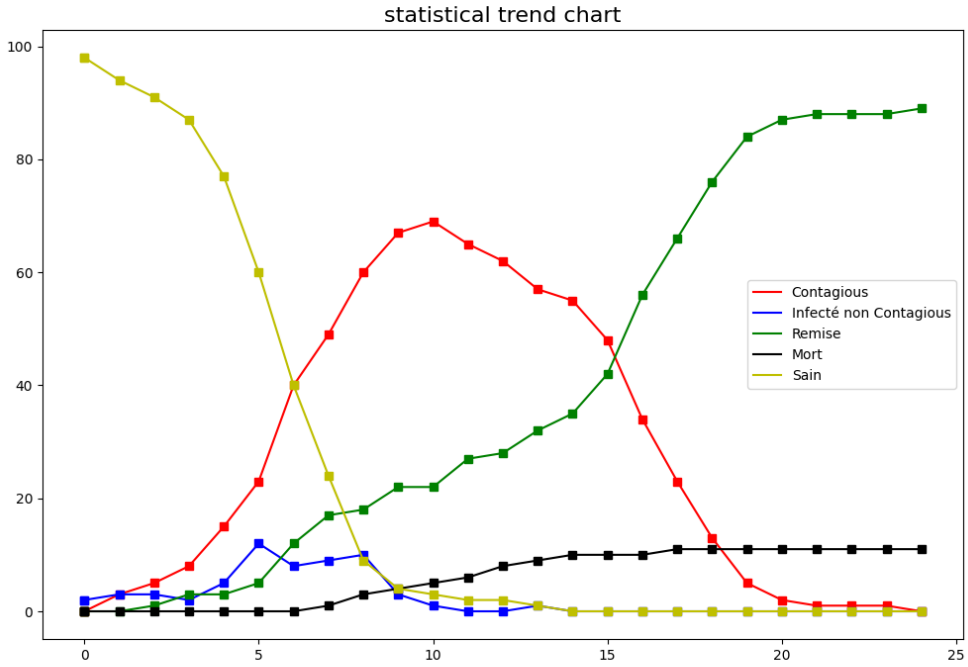
0.1


statistical trend chart
Contagious
Infecté non Contagious
Remise
Mort
Sain

0.2



0.5



On constate que la durée totale de l'infection augmente puis diminue lorsque la valeur du taux de infection augmente.

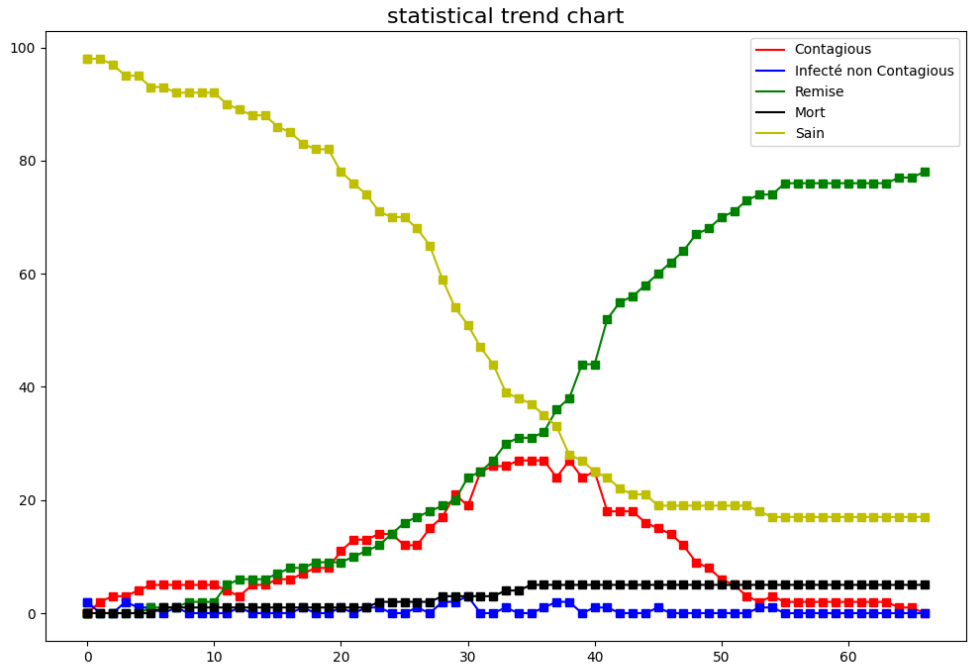
Lorsque le taux de infection est faible, plus le nombre de personnes infectées est élevé et plus le nombre de personnes malades augmente, plus le temps est long. Cependant, une fois que le taux de infection atteint un certain niveau et que la majorité de la population est déjà infectée, l'augmentation du taux de transmission ne fera qu'accélérer le processus d'infection.

Temps de guérison

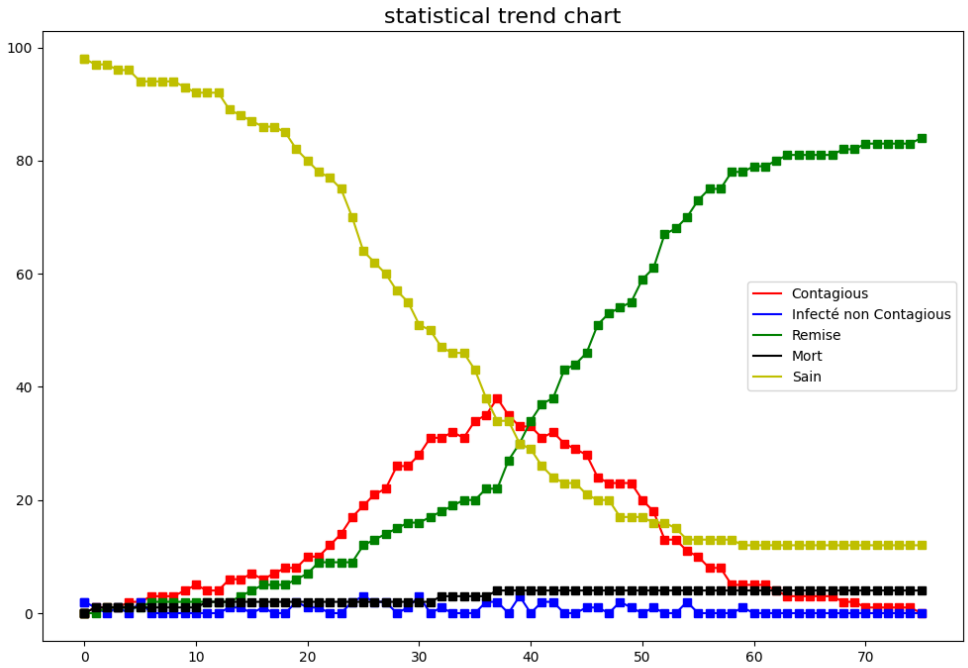
Nous nous référons au modèle de taux de transmission que nous venons de décrire, le rapport entre la densité de la population et l'étendue de la transmission du virus : 100/800*600 /10**2

Les autres paramètres du virus sont death_rate0.1 contagious_rate0.8 incubation_Period1.0 0.2 infect_rate0.1

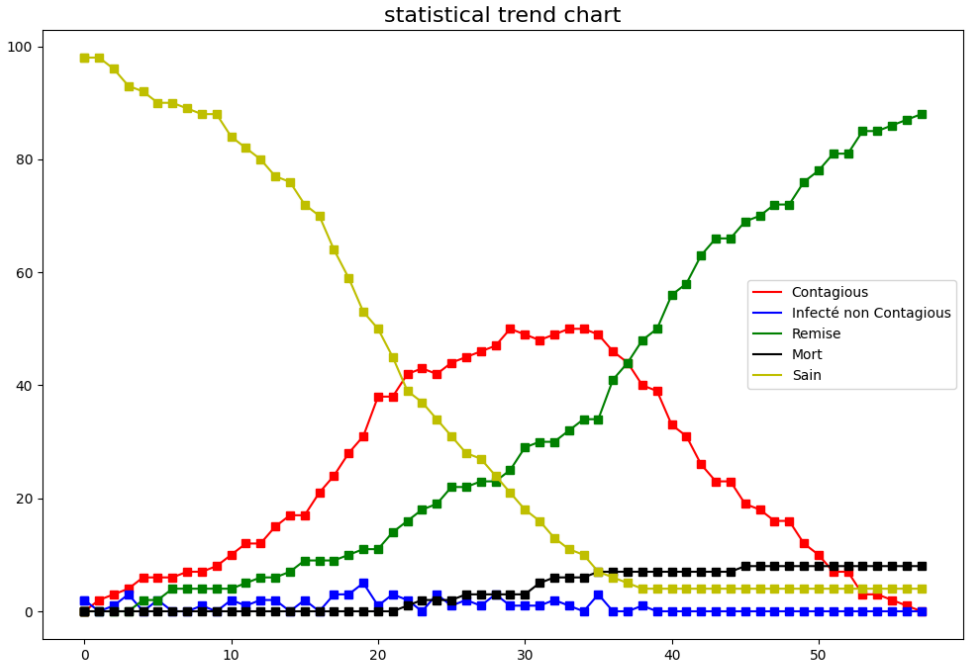
heal_time10.0 0.2



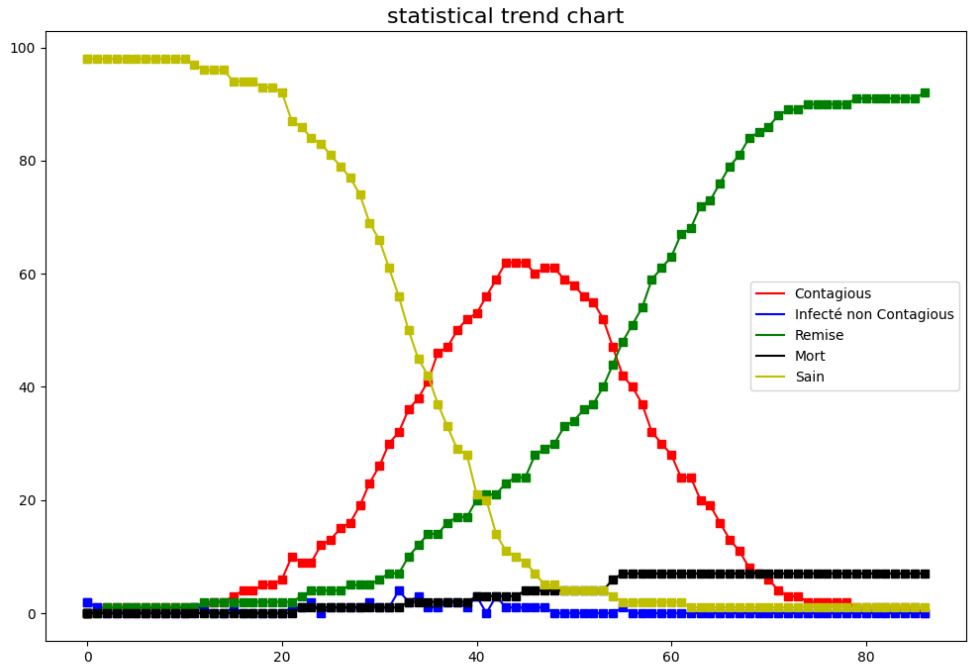
heal_time15.0 0.2



heal_time20.0 0.2



heal_time25.0 0.2



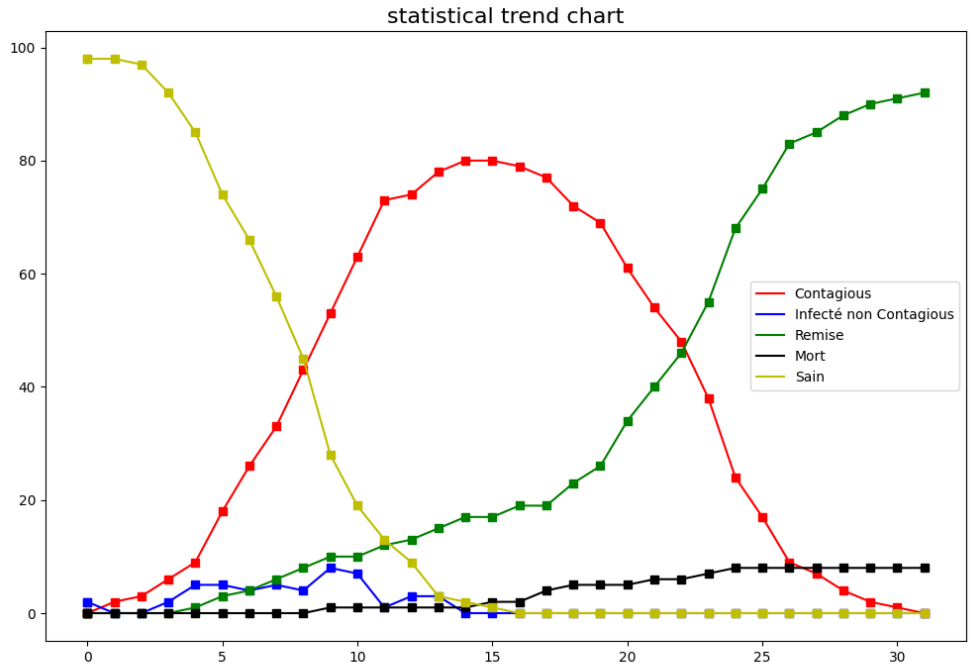
Nous pouvons constater que le temps total fluctue deux fois avec l'augmentation du temps de guérison. Lorsque nous sommes à une certaine distance de l'infection de toute la population, l'augmentation du temps de guérison fait que le temps total augmente également. Et quand il était temps de se rapprocher infecter de la population entière, l'augmentation du temps de guérison a plutôt fait diminuer le temps total, et ensuite, après que presque toute la population ait été infectée, l'augmentation de notre temps principal a fait augmenter le temps total à nouveau.

Période d'incubation, taux d'incubation

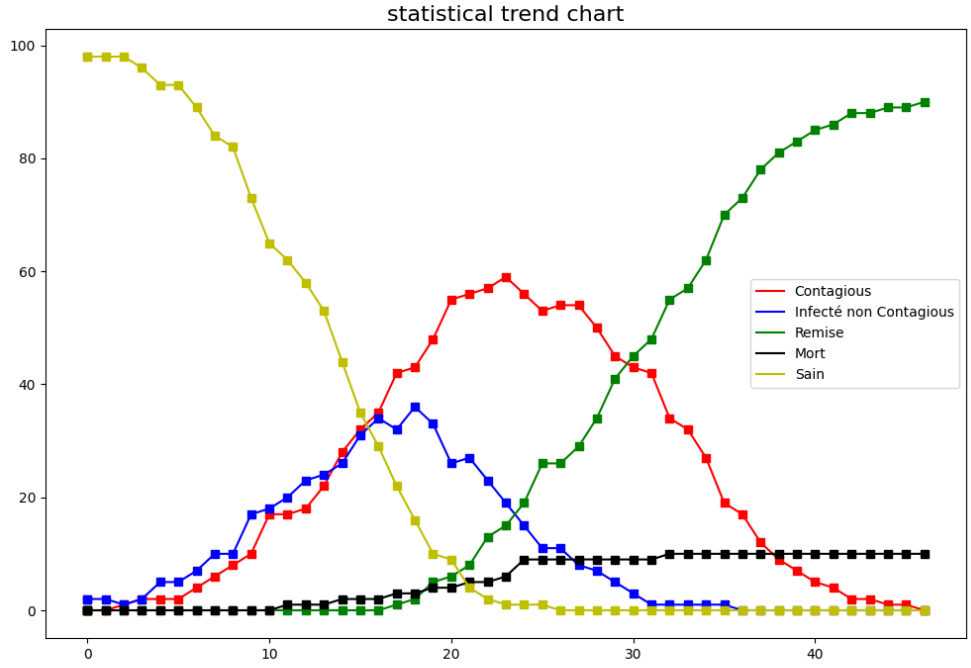
Nous nous référons au modèle de taux de Infection que nous venons de décrire, le rapport entre la densité de la population et l'étendue de la transmission du virus : 100/800*600 /10**2

Les autres paramètres du virus sont death_rate0.1 contagious_rate0.8 infect_rate0.3 heal_time15.0 0.2

incubation_Period1.0 0.2



incubation_Period10.0 0.2



On constate que la durée totale de présence de la maladie infectieuse augmente de manière significative.

Le taux d'incubation peut être facilement vu dans le graphique ci-dessus et n'est pas modélisé ici.

Le roi de la virus

Selon le principe de fer viral, un virus qui a un taux de transmission élevé ne peut pas avoir un taux de mortalité élevé.

Nous avons donc deux types de virus, l'un avec un faible taux de transmission mais un taux de mortalité élevé et l'autre avec le scénario inverse, un taux de transmission élevé et un taux de mortalité élevé. La portée effective de la transmission du virus est la même pour les deux.

Rapport entre la taille de la population et la portée effective de transmission du virus : 800*600 /10**2

Supposons que le virus numéro 1 ait death_rate0.6 heal_time10.0 0.2 contagious_rate0.8 incubation_Period1.0 0.2 infect_rate0.1

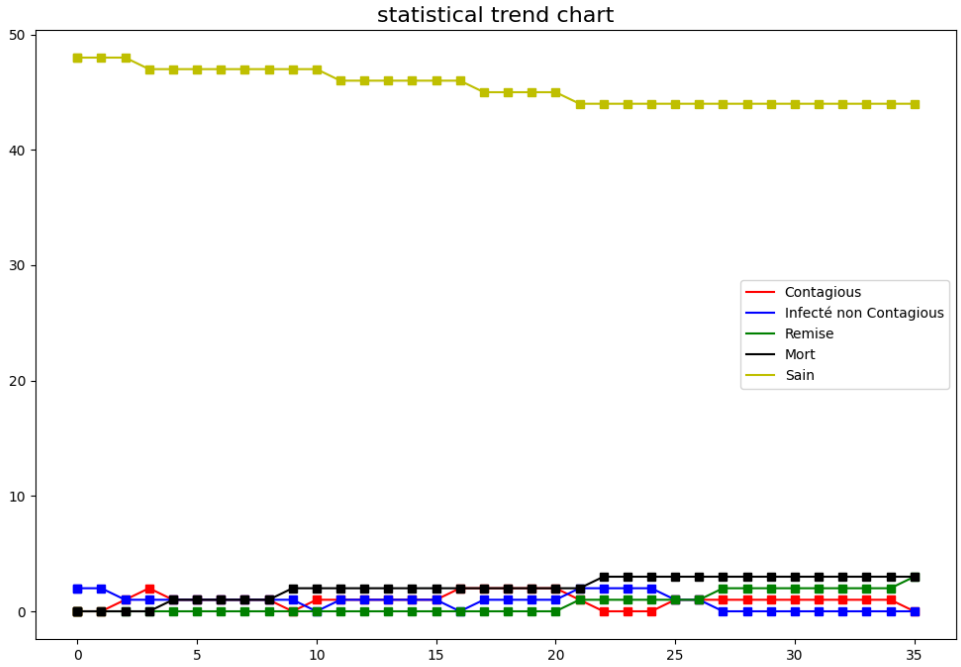
L'autre virus a pour paramètre taux de mortalité0.1 heal_time10.0 0.2 taux de contagion0.8 incubation_Period1.0 0.2 infect_rate0.6

Nous avons divisé l'analyse en deux cas

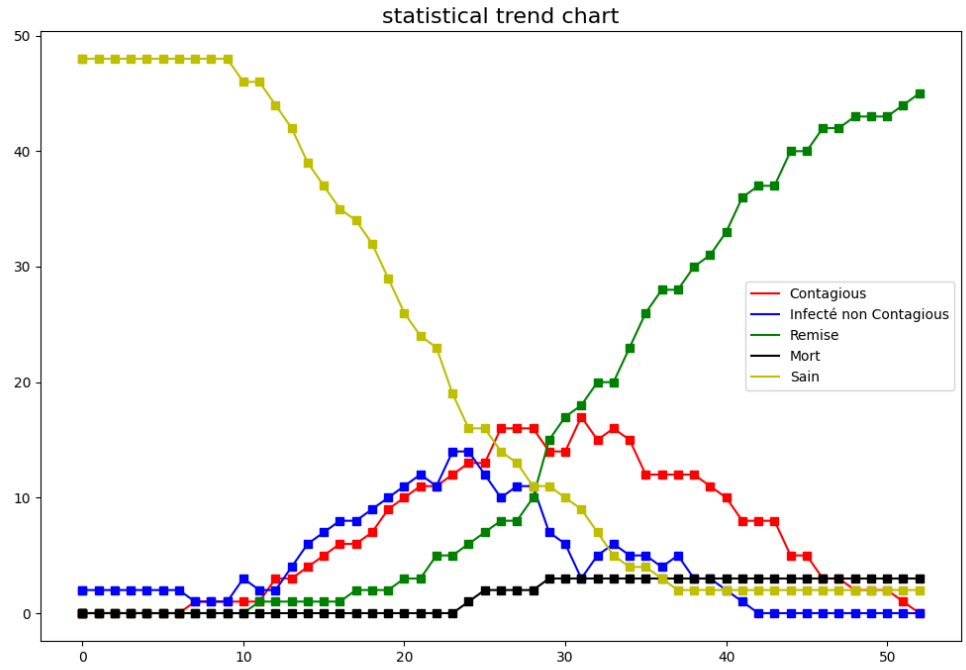
Premier scénario.

Avec une population de 50 personnes, ce qui peut être décrit comme une très faible densité de population, nous comparons deux modèles de virus à ce stade.

Virus numéro un.



Virus numéro deux.

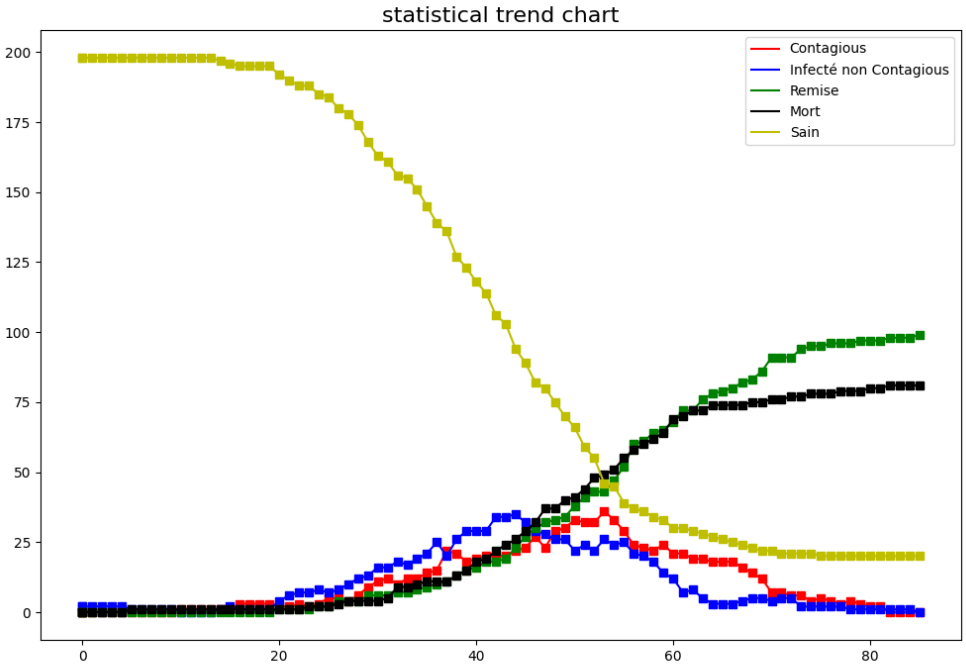


Nous constatons donc qu'à une population de 50 personnes, le virus numéro 1 est incapable de se propager, le nombre de décès est légèrement inférieur au nombre de décès du virus numéro 2, et la durée totale de l'infection causée par le virus numéro 2 est plus forte que celle du virus numéro 1, donc le virus numéro 2 gagne.

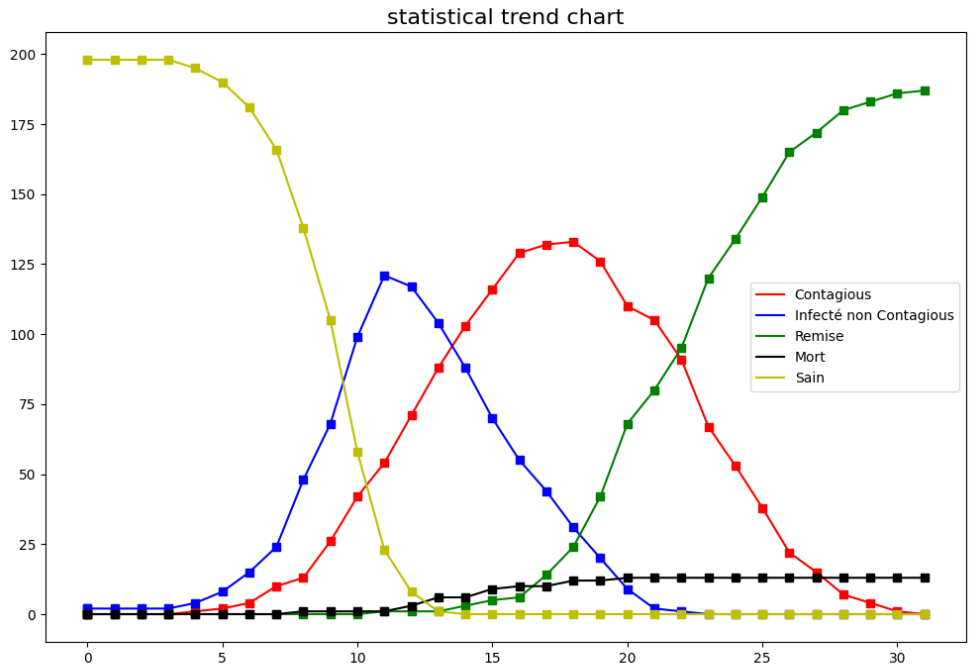
Deuxième scénario.

Avec une population de 200 personnes, nous pouvons dire que la densité de la population est très faible. À ce stade, nous comparons deux modèles de virus.

Virus numéro un.



Virus numéro deux



Nous pouvons constater qu'avec une population totale de 200 personnes et une forte densité de population, nous avons le virus numéro un, qui est capable de se propager suffisamment. À ce stade, il est le roi des virus et les dommages causés à la population humaine sont particulièrement graves : sa présence infectieuse totale est longue et le taux de mortalité de la population est élevé, s'en approchant à mesure que la densité de population augmente.

L'impact des deux virus varie donc en fonction de la densité de la population.

Analyse des mesures

Nous prenons des mesures pour contenir la propagation d'un virus.

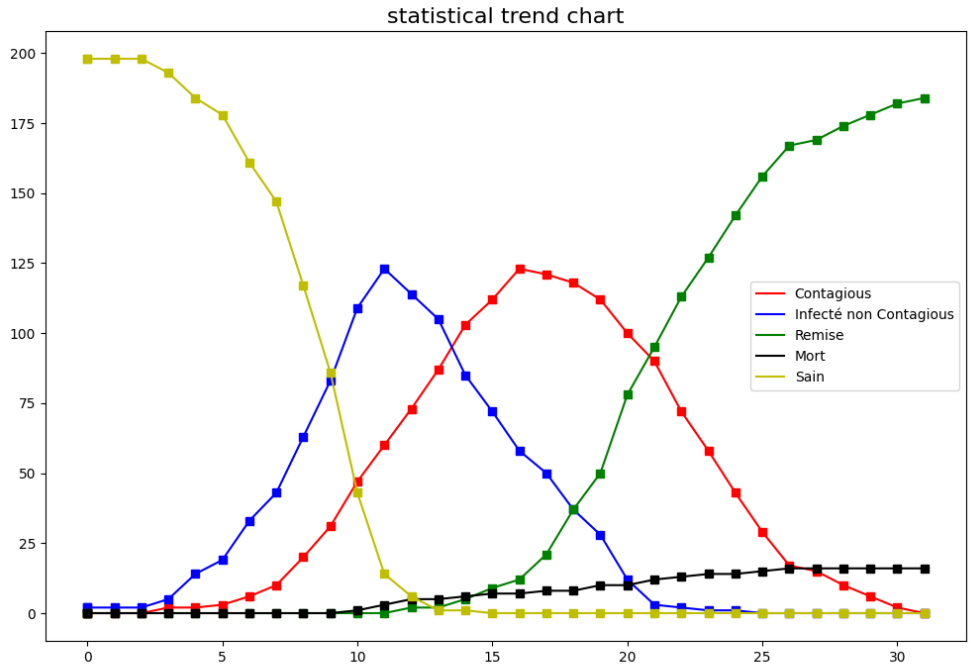
Avec un masque,Réduction de la distance communication sociale

Le port d'un masque a pour effet de réduire le taux de transmission. Prenons l'exemple du virus numéro 2 ci-dessus.

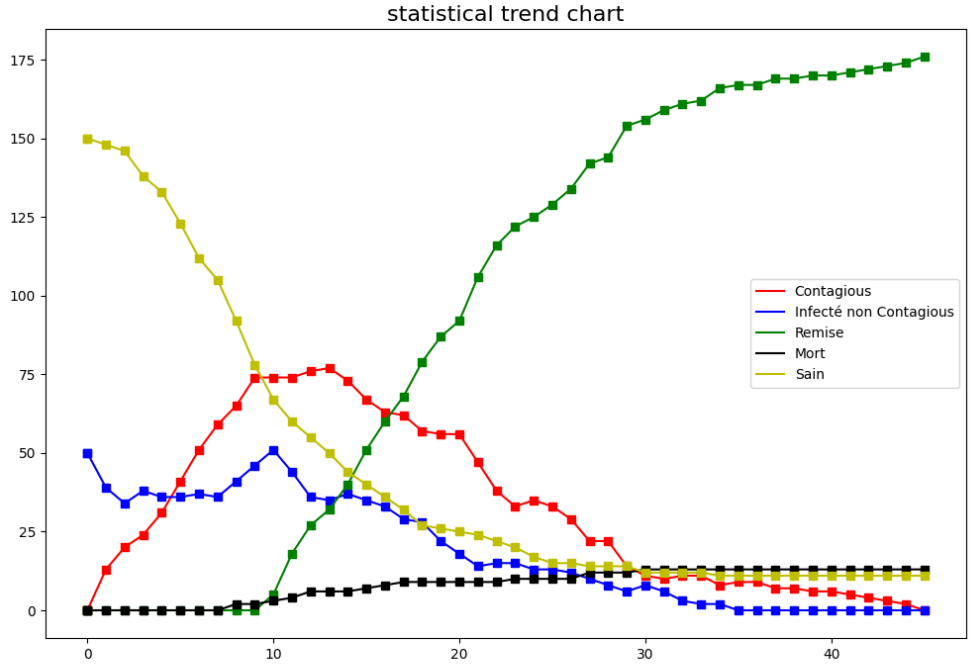
Densité de la foule : 200/800*600

Les paramètres du virus sont les suivants : death_rate0.1 heal_time10.0 0.2 contagious_rate0.8 incubation_Period1.0 0.2 infect_rate0.6

Rayon efficace de propagation du virus 10

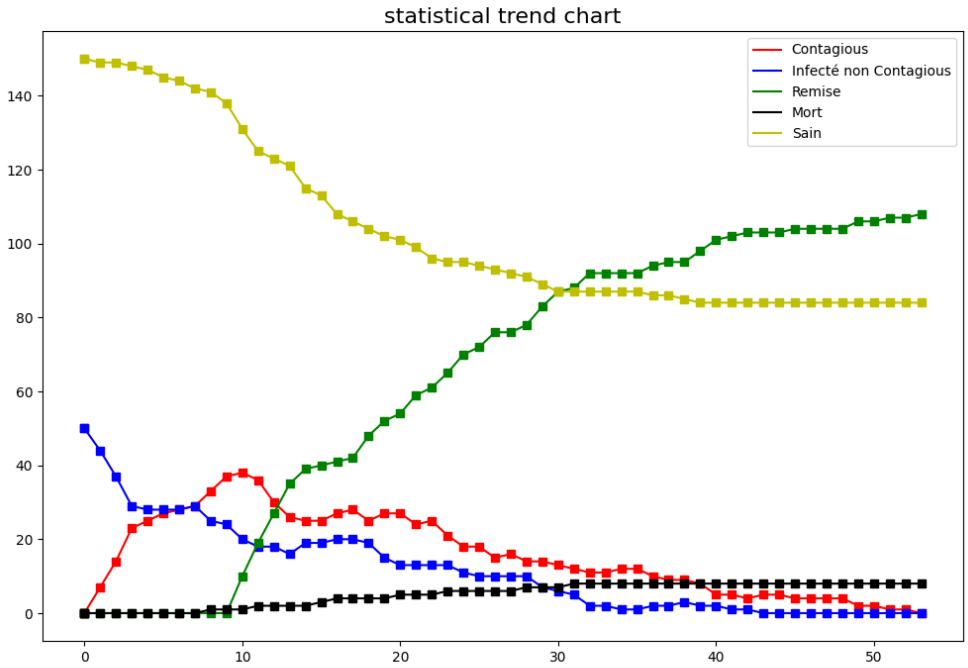


Le taux de transmission devient 0,6*0,1=0,06 (le port d'un masque réduit le taux de transmission de 90 %) et on augmente le nombre initial d'infections à 50, on constate que si l'épidémie est effectivement contenue

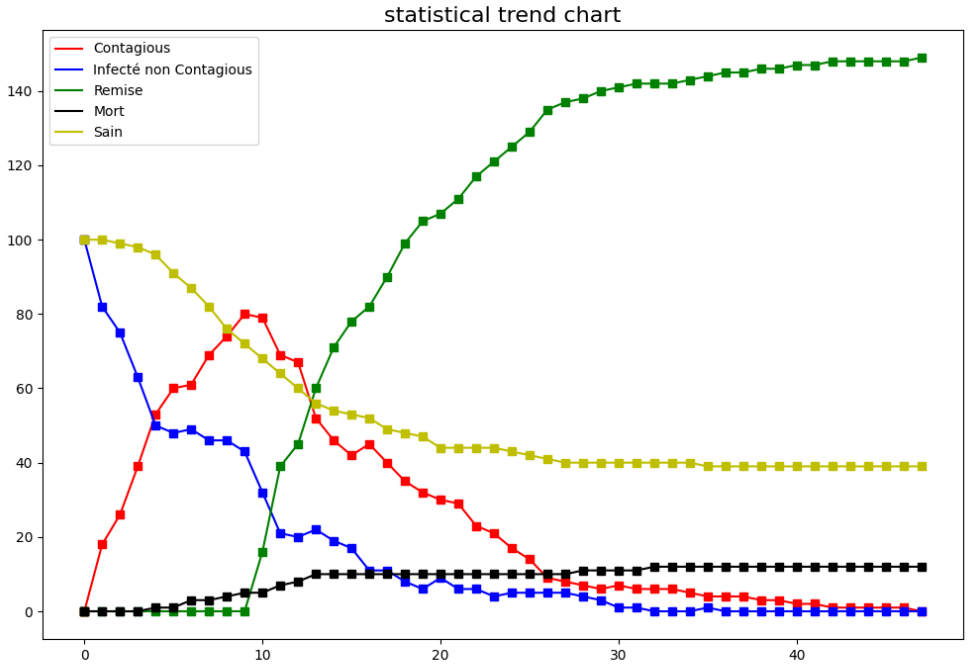


Nous pouvons constater que l'épidémie n'a pas été efficacement contenue.

Nous changeons à nouveau la bille à 5, (augmentant la distance communication sociale) et le taux de transmission à 0.6*0.1=0.06 (porter un masque réduit le taux de transmission de 90%) et augmentons le nombre initial de personnes infectées à 50, nous voyons que si l'épidémie est effectivement contenue



Nous augmentons ensuite le nombre initial d'infections à 100 et nous voyons si l'épidémie peut être contenue efficacement.

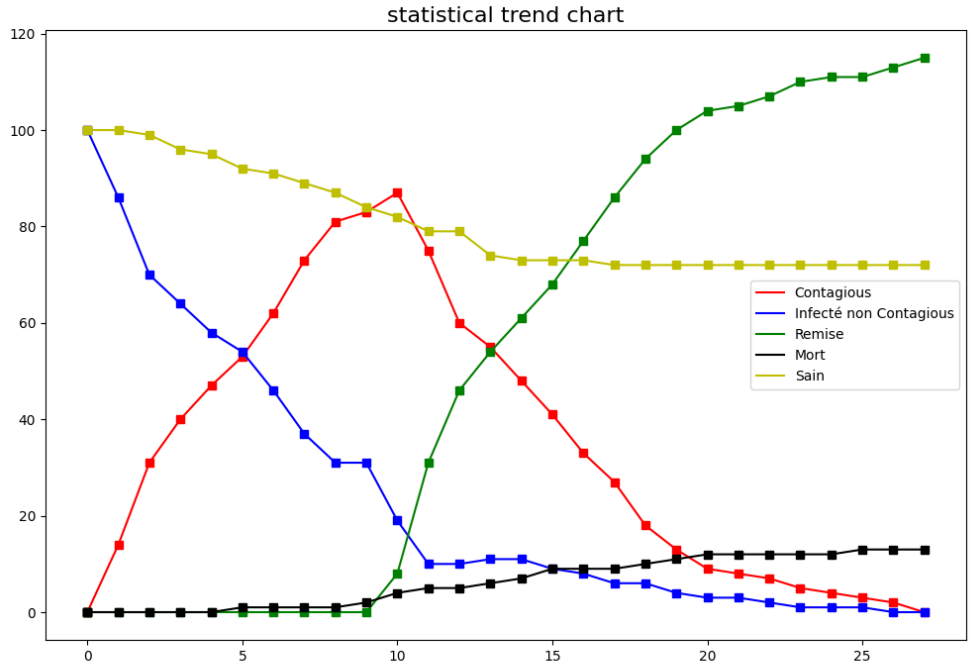


Par observation, nous pouvons à peine accepter que 60 personnes sur 100 contractent la maladie. Si nous ne pouvons pas accepter ce résultat, nous devons prendre d'autres mesures d'isolement.

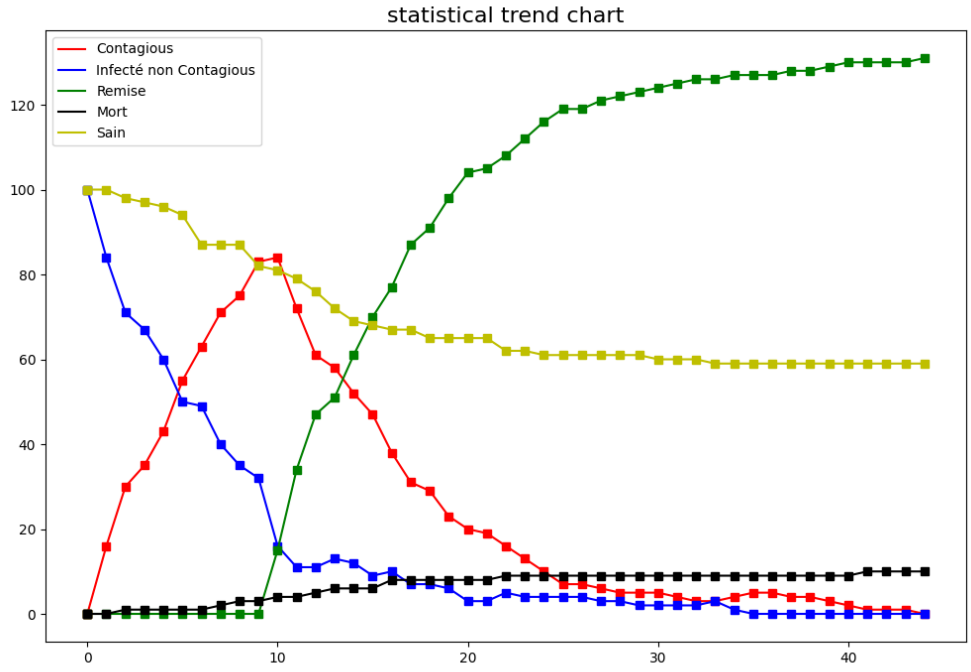
Confinement

Si nous suivons le scénario précédent, nous ne pouvons qu'accepter que les 20 % restants de la population qui ne sont pas infectés peuvent qu'être infectés.

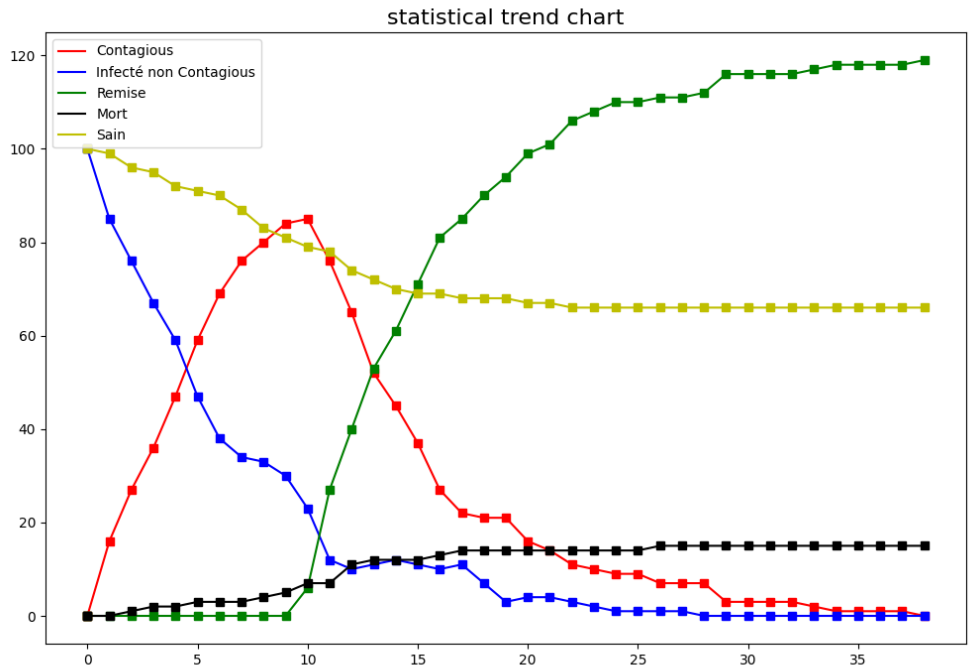
Nous appelons les gens à prendre leurs propres mesures de quarantaine et à isoler ceux qui sont contagieuse; Si 30 pour cent de la population est mise en quarantaine



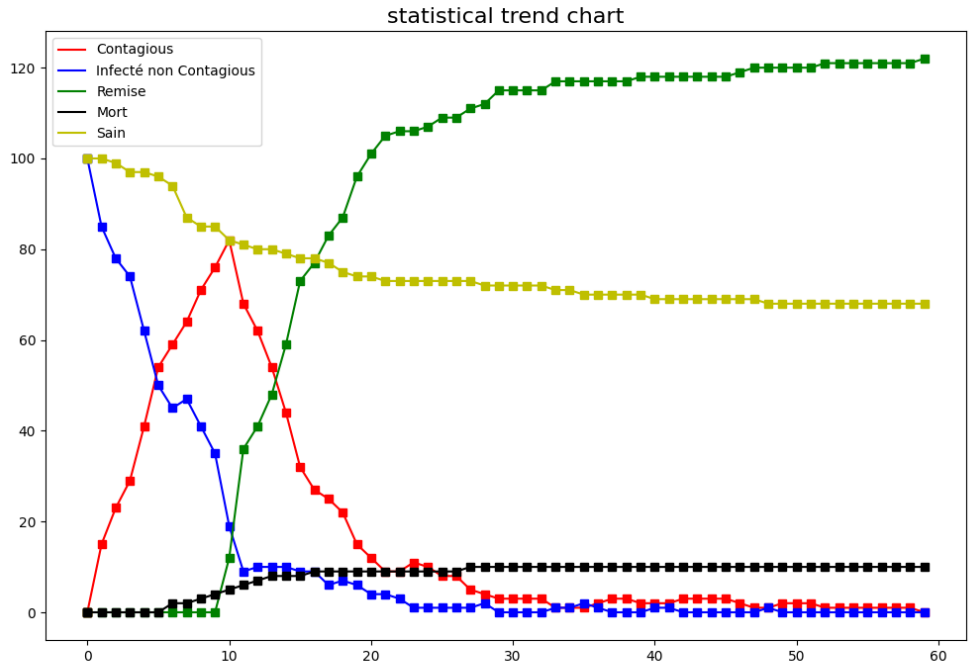
Il se trouve que nous sommes très proches des 20 % et nous appelons la foule à accepter la isolation pour atteindre 35 %.



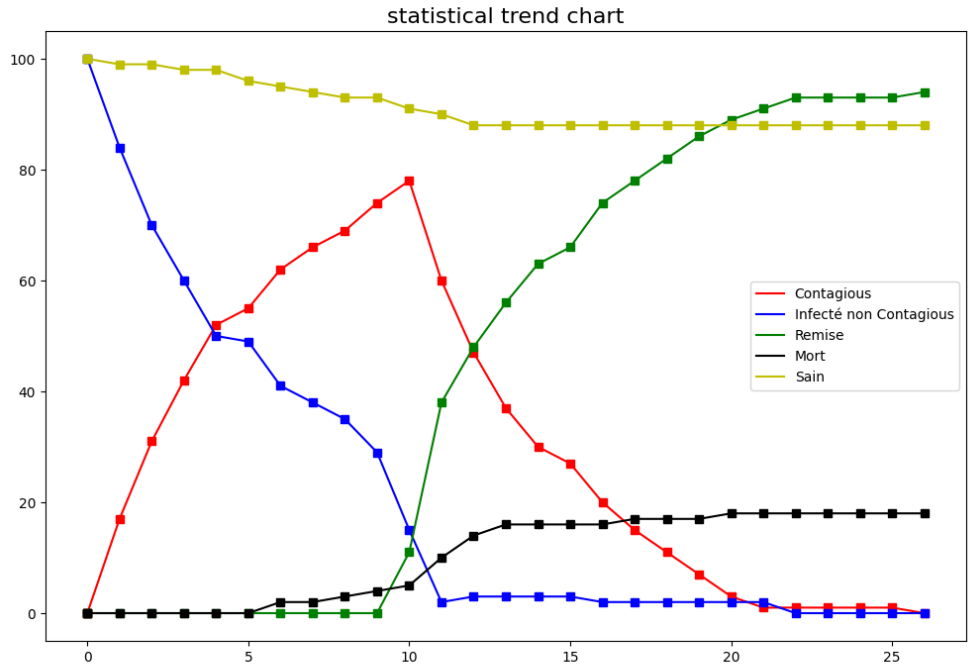
Il semble que ce que nous avions avant était une situation fortuite, et nous avons ensuite appelé jusqu'à 40 pourcent de la population accepter en quarantaine



Nous mettons alors jusqu'à 45% de la population qui sont contagieuse en quarantaine.



Il semble que notre politique de ségrégation volontaire ait eu peu d'effet et nous adoptons maintenant la ségrégation obligatoire, ce qui porte le taux de ségrégation à 80 %.



Le nombre de personnes touchées par la maladie a été réduit à 10. Il semble que le gouvernement doive prendre des mesures obligatoires afin de contrôler efficacement l'épidémie.

Conclusion

Tout d'abord, ce projet n'est qu'une analyse qualitative de l'impact des maladies infectieuses sur la population. Par exemple, on ne connaît pas l'évolution du taux de mortalité dans le temps ni la fonction de probabilité du taux d'infection dans le temps.

Le processus de mon projet présente un certain nombre de lacunes, notamment

1. Il existe de nombreux facteurs qui font qu'une maladie infectieuse impressionne une population. L'analyse de la variance que j'avais l'intention d'utiliser n'a finalement pas été utile.
2. Le modèle de collision des sphères était problématique et ne permettait pas de résoudre le problème.
3. n'a pas commencé le projet à l'avance, de sorte qu'il a pris beaucoup de retard.

Annexe

> https://images.math.cnrs.fr/Modelisation-d-une-epidemie-partie-1.html
>
> INFO_Saunier_projet1
>
> https://www.pygame.org/wiki/tutorials