An agent-based epidemic model for COVID-19

Abstract:

Background: India has adopted many different kinds of policies in tackling the COVID-19 pandemic. India closed down large areas completely to increase social distancing. There is clearly a need for a model that can realistically simulate different policy actions and their impacts on the disease and health care capacity in a country or region. Specifically, there is a need to identify destructive policies.
Method: We developed an agent based simulation model. It estimates the number of infected persons in a particular day. It estimates the number of people recovered, died and quarantined. It estimates the recovery and death rate.
Result: The model uses the total population of India (i.e. 135,00,00,000). The other, a suppression strategy , consists of the same mobility restrictions but also aggressive testing, tracing, and isolating all coronavirus positive patients and their contacts. Taking starting point as 28 -03 -2020. If a person having common or serious symptoms delay in visiting doctor/ getting tested the peak of epidemic delays having a larger value. If the number of contacts increases the slope increase (number of infected persons on a per day basis increases).
Discussion: The agent-based model can be used to simulate epidemic outcomes of various types of policy actions on a timeline. Results lend support to the strategy of combining comprehensive testing, contact tracing and targeted isolation measures with social isolation measures. The flexibility of the modelling logic supports the future implementation of several already identified refinements in terms of more realistic population model and new types of more specific policy interventions. Improving estimates of epidemic parameters will make it possible to improve modelling accuracy further.

Background:

There are two main approaches to the coronavirus crisis. First, one can assume that a pandemic is already so widespread globally that it will inevitably infect a large fraction of the population. Then the key thing is to slow down the spread in each country so that the health care system can cope with the COVID-19 patients, many of which need intensive care with respirators. Eventually, the epidemic fades away due to herd immunity, i.e. there are enough recovered people to prevent effective spread of the virus in the population. This is called the *mitigation strategy*.

Second, one can assume that the disease can (and must) be suppressed so that only a small number of patients shall exist in the population. Rather than relying on herd immunity like in mitigation, suppression has to be actively maintained by isolating infected people. Total eradication in any one country is impossible in practice as long as the disease is still common in other countries. The epidemic is kept under control by actively searching for all patients (including those without symptoms) and isolating them so that the infection chains are cut. This is called the *suppression strategy*.

The asymptomatic fraction appears to be close to 0.8*, said by R Gangakhedkar, the head of the epidemiology division at the Indian Council of Medical Research (ICMR), said at a press conference on April 20, 2020.*
The recovery of the person being symptomatic and having mild symptoms is 2 weeks and that for serious symptoms it is 6 weeks, *by World Health Organisation.* Therefore total time taken from getting infected to getting recovered/death will be around 1 month for mild symptoms and that for serious symptoms will be around 3 months. The quarantine period is about 14 -18 days, *mentioned in Guidelines for quarantine facilities COVID-19 issued by government of India.*

Method:

This model is based on random interactions between agents, and the interactions lead to outcomes with fixed probability distributions (see Table 1). Although the individual events and times spent in different states are probabilistic, many key parameters whose real-world values still involve significant uncertainty are assumed as known constants in a simulation. Therefore, sensitivity analyses were done with parameters: In addition, the code was run for 100 times with random seeds to investigate their sensitivity to initial values.

The agents represent individual people, who have the following properties: symptomatic or asymptomatic, days of incubation period, days of illness since the onset of symptoms, other people infected, state, and symptom getting critical. The possible states of a person are the following: incubation, hospitalized/quarantined, provided artificial breathing, recovered, and dead. The symptom severity of a patient can be asymptomatic, common, serious, or critical. The model follows every individual through the whole disease path from susceptible to recovery or death or hospitalized.
For number of person getting infected- **I = (I-Quarantine[a])*1.075**
For number of person getting tested - **T = (Quarantine[a])*120**

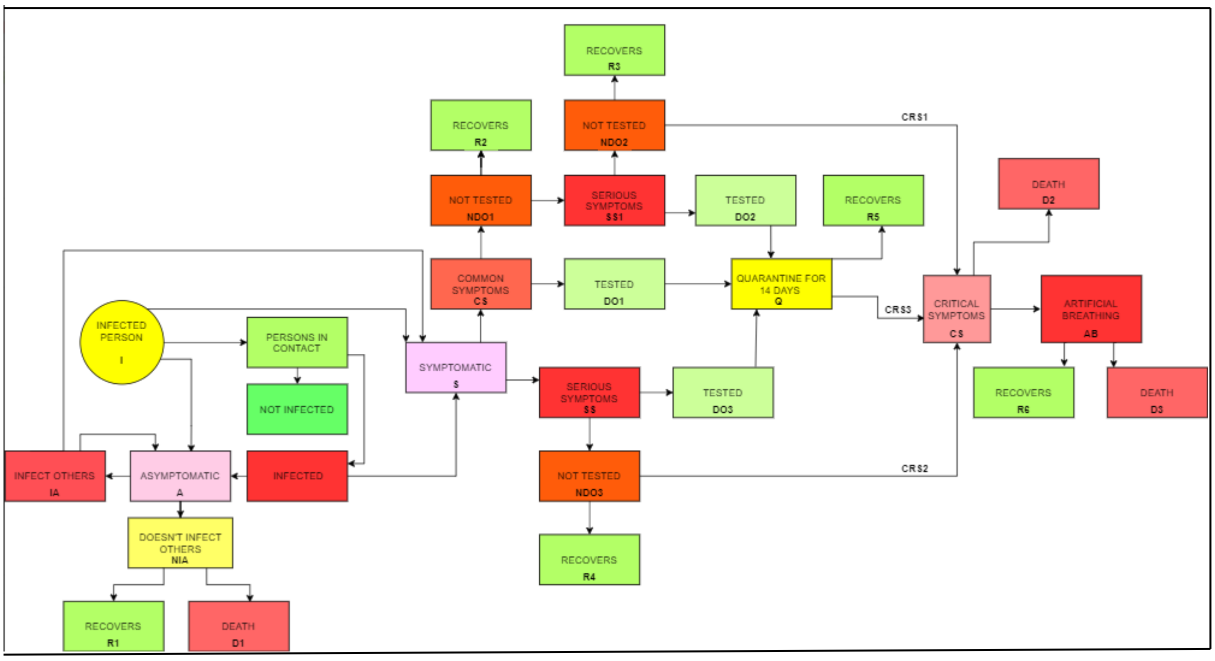

Figure1. Diagram of the corona virus agent model

Diagram shows all the possible ways for recovery and death. The arrows shows the path way. Green boxes show the recovery and the red boxes shows death. An infected person will be either symptomatic or asymptomatic. A symptomatic person may have serious symptoms or common symptoms. And further we proceed with diagram follows every individual through the whole disease path from susceptible to recovery or death.

<u>Table 1. Model parameters for Simulation</u>

Parameter	Value	Description and Reference
Infected person	115	*Worldometer.info data*
Total Population	135,00,00,000	Total population of India by *Worldometer.info data*
Asymptomatic(AS), Symptomatic(S)	0.85, 0.15	*Indian Council of Medical Research (ICMR)*
Common Symptoms(CS), Serious Symptoms(SS)	0.20, 0.80	Common symptoms 80% mentioned in *Corona virus disease 2019 (COVID-19) Situation Report – 46 by WHO.*
Visits a Doctor\Tested(DO1), Doesn't visits a Doctor\Not Tested(NDO1) - CS	0.30, 0.70	Peoples having common symptoms don't take it seriously and doesn't get tested by *sensitivity analysis*
Days needed	30	Time taken for symptoms to be visible will be around 10 days then getting tested and quarantine will take 18 days by *World health Organisation*.
Recovery(R2), Serious Symptoms(SS1) – NDO1	0.70, 0.30	Recovery percentage is around 60% by *Health minister Dr Harsh Vardhan*.
Days needed	50	Days needed to visit + Recovery time around 2 weeks mentioned by *World health Organisation*.
Visits a Doctor\Tested(DO2), Doesn't visits a Doctor\Not Tested(NDO2) - SS1	0.70, 0.30	Peoples having Serious symptoms prefers visiting a doctor by *sensitivity analysis.*
Days needed	95	Time taken for symptoms to be visible will be around 15 days then the time taken for the symptoms getting serious 15 days getting tested and quarantine will take 18 days and then recovery will take 6 weeks mentioned by *World health Organisation.*
Recover(R3), Critical Symptoms(CRS1) – NDO2	0.85, 0.15	Critical symptoms 15% mentioned in *Corona virus disease 2019 (COVID-19) Situation Report – 46 by WHO.*
Days needed	70	Getting tested + 6 weeks recovery time by *World health Organisation*.
Visits a Doctor\Tested(DO3), Doesn't visits a Doctor\Not Tested(NDO3) - SS	0.70, 0.30	Peoples having Serious symptoms prefer visiting a doctor by *Sensitivity analysis*.
Days needed	95	Time taken for symptoms to be visible will be around 15 days then the time taken for the symptoms getting serious 15 days getting tested and quarantine will take 18 days and then recovery will take 6 weeks mentioned by *World health Organisation.*
Recover(R4), Critical Symptoms(CRS2) – NDO3	0.85, 0.15	Critical symptoms 15% mentioned in *Corona virus disease 2019 (COVID-19) Situation Report – 46 by WHO.*
Days needed	70	Getting tested + 6 weeks recovery time.
Recover(R5), Critical Symptoms(CRS3) - Quarantine	0.85, 0.15	Critical symptoms 15% mentioned in *Corona virus disease 2019 (COVID-19) Situation Report – 46 by WHO.*
Days needed	14	Quarantine Period, mentioned *in Guidelines for Quarantine facilities COVID-19*
Death(D2), Artificial Breathing(AB) - CRS	0.05, 0.95	Ventilation 5% mentioned in *Corona virus disease 2019 (COVID-19) Situation Report – 46 by WHO.*
Days needed	15	Quarantine + Providing hospital bed *Guidelines for Quarantine facilities COVID-19.*
Death(D3), Recovery(R6), Condition improved(Q2) - AB	0.03, 0.45, 0.52	Death rate is around 3% by *Health minister Dr Harsh Vardhan*.
Days needed	30	Quarantine + Time needed to examine and Providing Treatment.
Infect Others(IA), Doesn't infect others(NIA) - AS	0.1, 0.9	Spread by asymptomatic person is very rare mentioned by *World health Organisation.*
Recovers(R1), Death(D1), Quarantine(Q1) - NIA	0.03, 0.90, 0.07	Death rate is 3% by *Health minister Dr Harsh Vardhan*.

Result:

Table 2. Outcomes of Simulation

Total Tests Performed	Total Infected Person	Recovered	Death	Quarantined/Hospitalised
21824185	1124160	980986	30066	113108

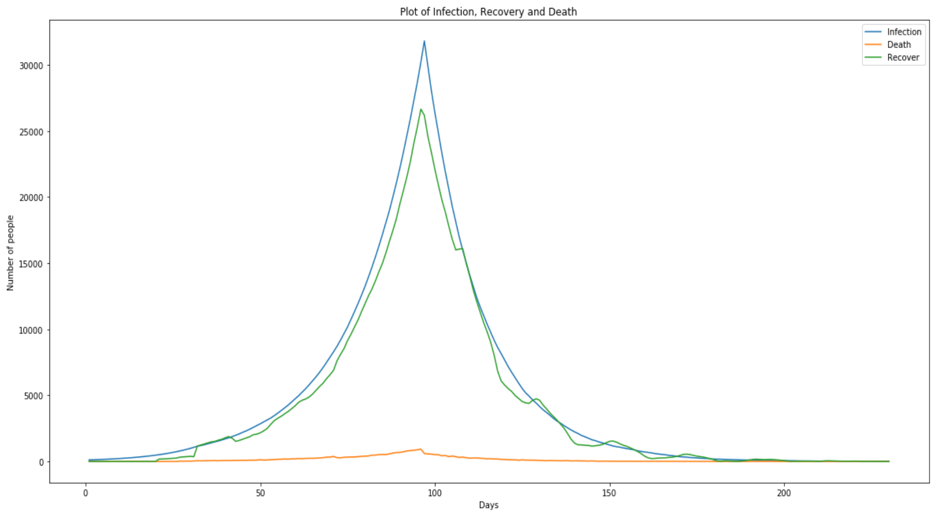

Figure 2:- Plot for Day-wise number of infection, Recovery and Death

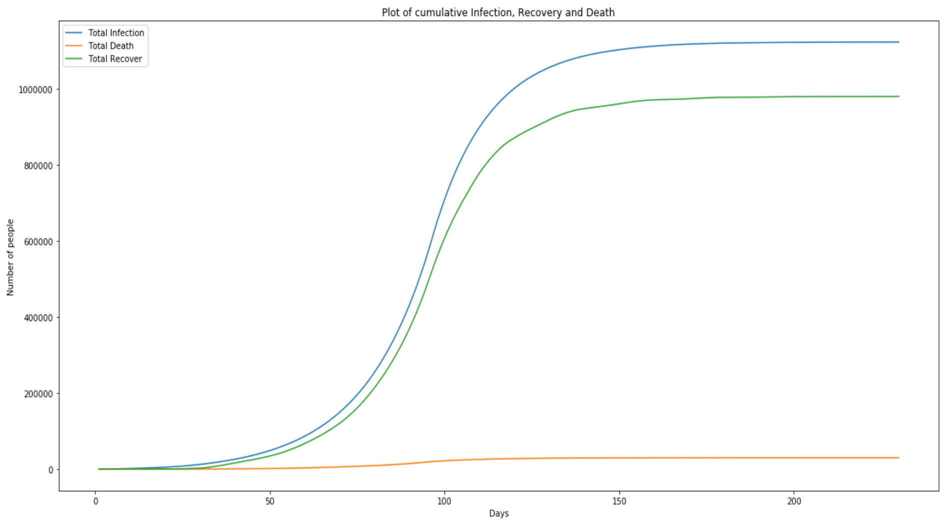

Figure 3:- Plot for Cumulative number of infection, Recovery and Death

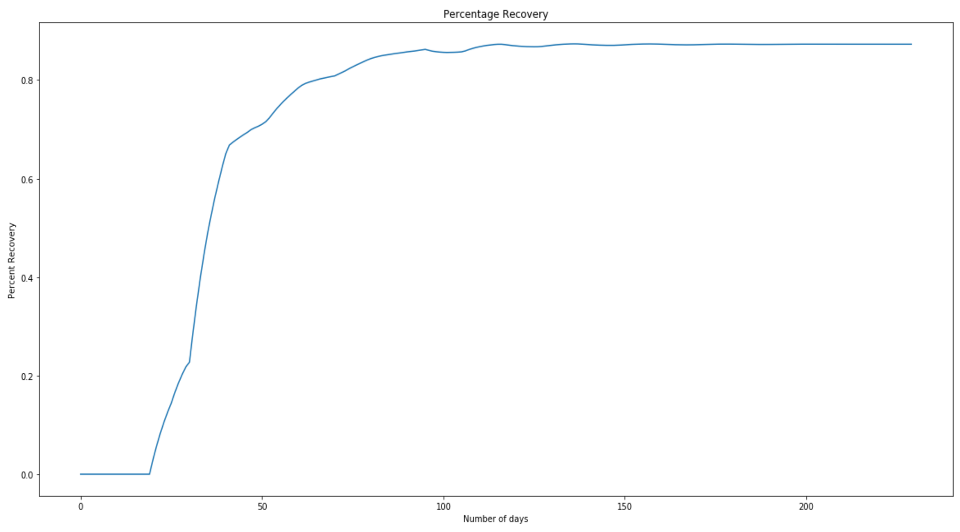

Figure 4:- Plot for percentage recovery

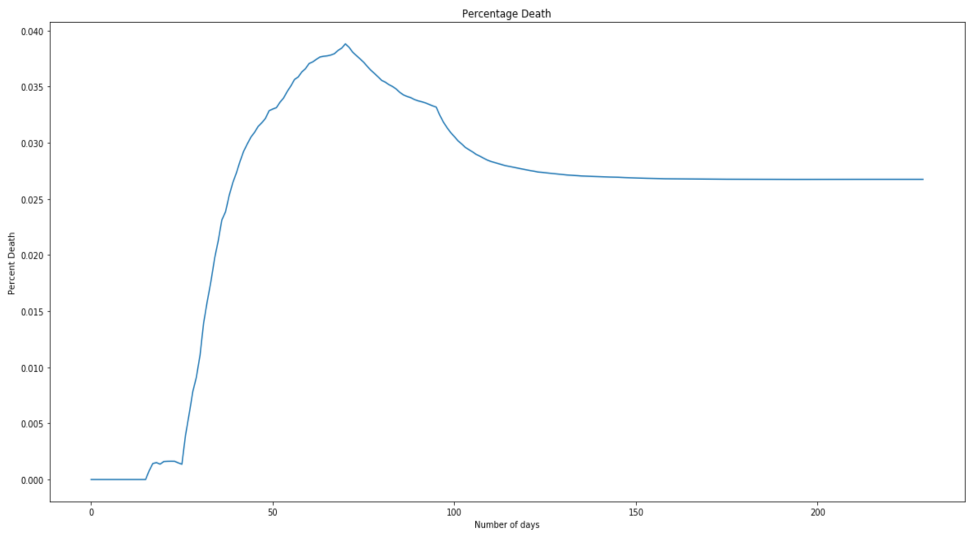

Figure 5:- Plot for Percentage death

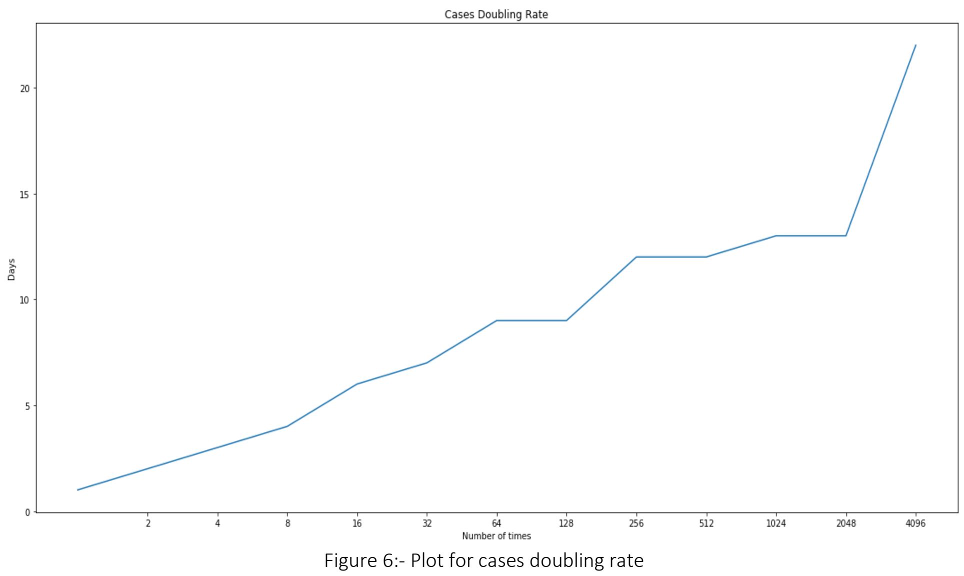

Figure 6:- Plot for cases doubling rate

Comparison with original data

(All data is collected from worldometer.info)

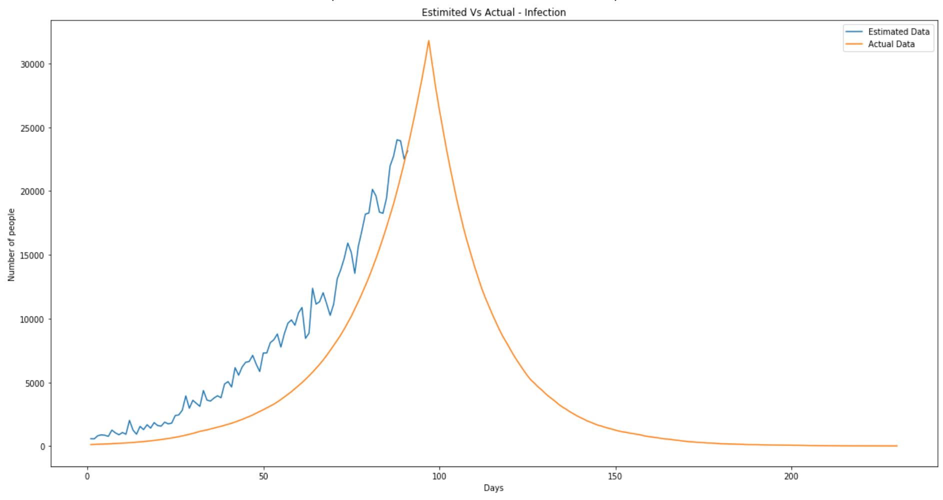

Figure 7:- Plot for Infected

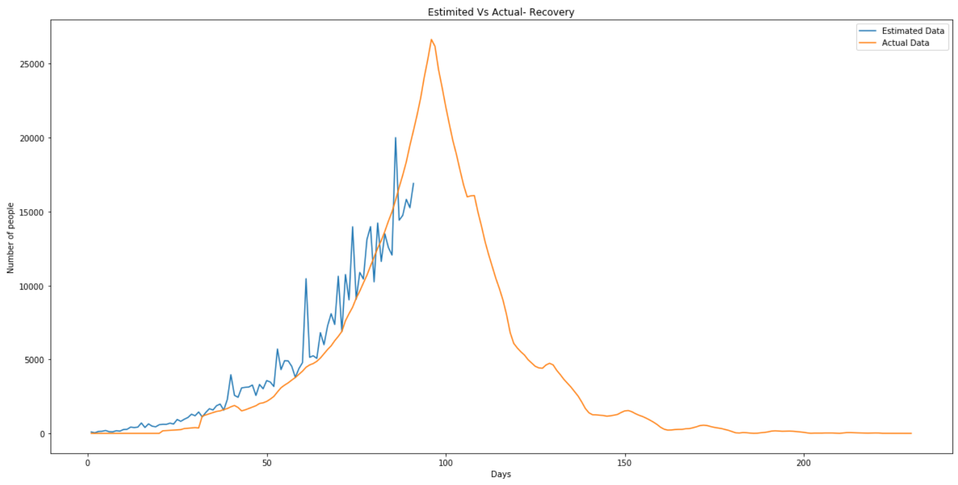

Figure 8:- Plot for Recovery

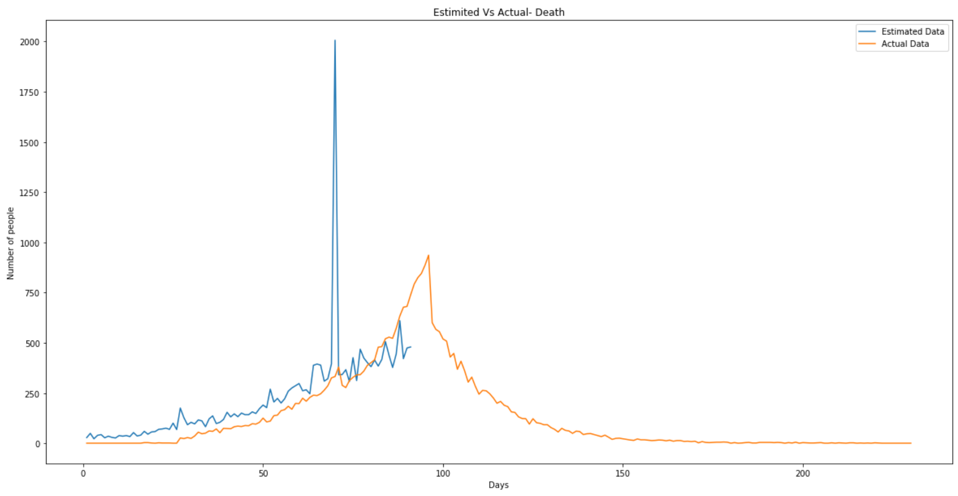

Figure 9:- Plot for Death

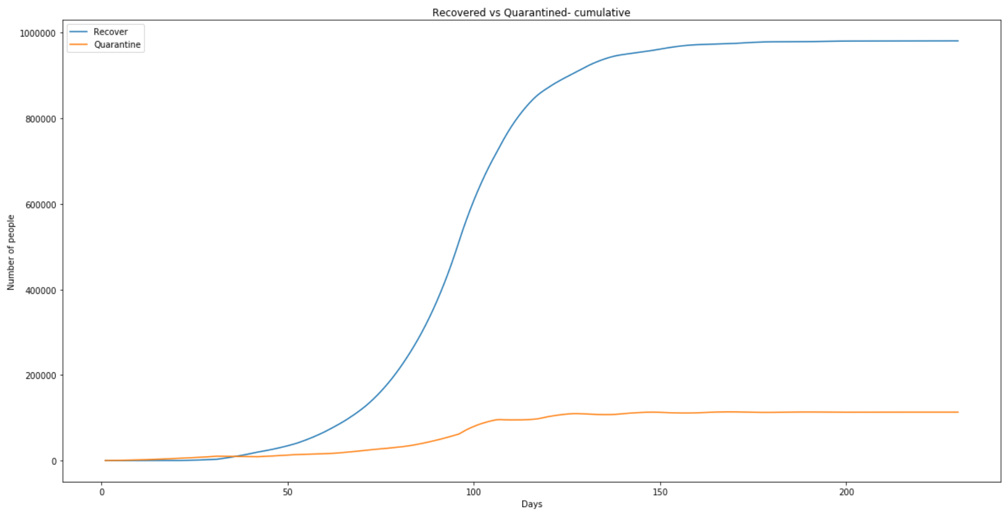

Figure 10:- Plot for Total Quarantined VS Total Recovered (Cumulative)

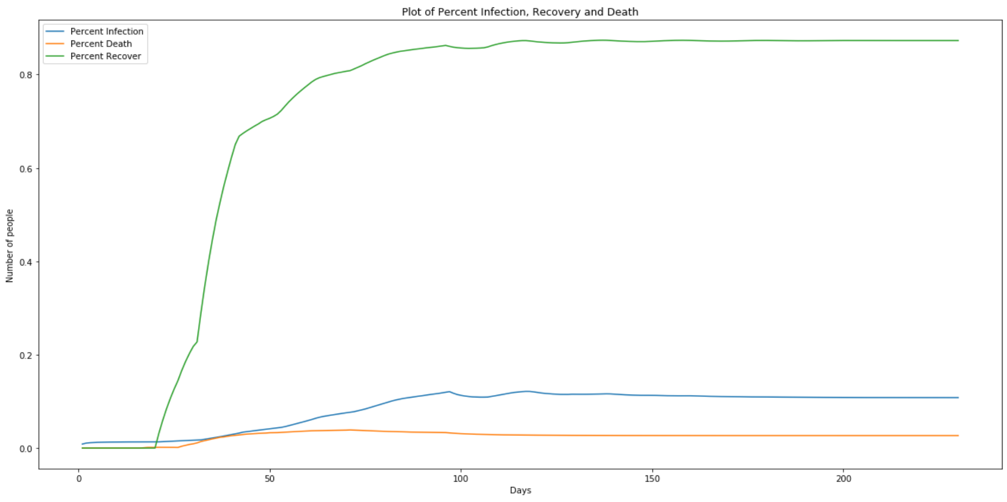

Figure 11:- Plot for Infected rate, Recovery rate, Death rate

After 230 days we are getting our peak on 102th day the value is around 39,000. The recovery rate after 230 days is approx. 0.80 and death rate is approx. 0.03.
We are taking the beginning day to be 10th of April. This means we will get our peak in last week of July 2020.

<u>Formulas Used:</u>

Recovery Rate = Total Recovered/ Total Infected
Death Rate = Total Death/ Total Infected
Infection Rate = Total Infected/ Total Tested

Table 3. Table for Descriptive Statistics

	INFECTION	RECOVERY	DEATH
Standard Deviation	3339.659	2515.082	122.552
Average Gap	1055.875	331.769	55.780
Maximum Gap	1583	1990	1647
Minimum Gap	-246	-1224	-59
Quartile 1	4261	3777	169
Quartile 2	10066	9396.5	203.5
Quartile 3	299	527	4

Discussion:

The model predicts that the day-wise infected, recovered and died person through suppression strategy allowing aggressive combined testing and tracing of contacts. The suppression strategy has been successfully implemented in practice in India.
Testing makes targeted isolation of patients possible. This is important as general restrictions are detrimental for society and economy, especially if they last for several months or more. Also, the longer the restrictions, the harder it becomes to maintain them.
Suppression may consist of strong initial restrictions to get the epidemic under control. However, the idea is to reduce the cases enough so that active testing and contact tracing becomes feasible. The long-term objective is to prevent the spreading of the virus with testing and targeted isolation rather than general restrictions.
As the number of contact on a per day basis be increased then the value to infected person will increase hence the peak value will increase.
Less the presymptomatic and quarantine period earlier will be the peak. Sensitivity analyses were performed for various parameters. These parameters had drastic effects on the rate of disease spread and thus model outcomes. Suppression with aggressive testing also requires a shorter time with strict mobility restrictions.

Code:

```
import randomimport random
import numpy as np
from matplotlib import pyplot as plt
I = 105
TP= 135000000
n = int(input("enter the number of days: "))
Infected = []
Death = []
Recover = []
Quarantine =[]
S = []
AS = []
IA = []
NIA = []
R1 =[]
D1=[]
CS = []
SS = []
DO1=[]
NDO1=[]
R2=[]
SS1=[]
DO2 =[]
NDO2 =[]
R3 =[]
CRS1 =[]
DO3 =[]
NDO3 =[]
R4 =[]
CRS2= []
Q = []
R5 =[]
CRS3 =[]
D2 =[]
AB =[]
CRS =[]
D3 =[]
R6 =[]
Q1=[]
Q2=[]
RC=[]
DC=[]
QC=[]
RP=[]
DP=[]
X=[]
Y=[]
Z=[]
TC=[]
day=[]
dayG=[]
test = []
IP=[]
T = 20000
GovR =
[93,38,129,139,195,111,101,178,149,260,273,422,391,419,702,395,642,484,443,584,614,610,690,631,939,812,956,1072,1295,1189,1445,1111,1414,1668,1580,1871,1980,1569,2289,3
966,2571,2438,3076,3113,3131,3271,2561,3307,3014,3571,3472,3171,5707,4309,4916,4902,4531,3786,4379,4783,10462,5143,5247,5070,6814,5993,7259,8095,7363,10631, 6886,
10744, 9024, 13975, 9071, 10885, 10437, 13114, 13983,10246,14229,11628,13497,12565,12060,20006,14417,14743,15829,15259,16908]
Gov = [ 573, 565, 809, 875, 846, 759, 1248, 1034, 883, 1060, 922, 2013, 1250, 924, 1541, 1290, 1669, 1408, 1836, 1607, 1561, 1873, 1738, 1801, 2394, 2442, 2806,3932, 2963, 3587,
3344, 3113, 4353, 3607, 3524, 3763, 3942, 3787, 4864, 5050, 4630, 6147, 5553, 6198,6568, 6629, 7113, 6414, 5843, 7293, 7300, 8105, 8336, 8782, 7761, 8821, 9633, 9889, 9471, 10438,
10864, 8442, 8852, 12375, 11128, 11320, 12023, 11157,10243, 11135, 13103, 13827, 14721, 15915, 15183, 13540, 15665, 16870,
18185,18276,20131,19620,18339,18256,19428,21948,22721,24015,23932,22510,23135]
GovD =
[28,49,22,39,43,27,35,29,26,38,35,38,33,53,36,40,59,45,56,58,69,71,75,69,100,68,175,127,92,104,96,116,111,82,121,136,98,104,118,154,131,146,132,150,142,142,156,148,172,190,17
7,269,205,223,200,221,259,275,286,297,261,266,246,388,394,389,309,321,395, 2006, 341, 342, 366, 307, 426, 312, 468, 424, 401,381,414,384,417,506,438,377,444,610,421,474,479]
for a in range(len(Gov)):
    dayG.append(a+1)
for a in range(n):
    day.append(a+1)
    print("infected Persons on", a+1, "day:",I)
    A = []
    for i in range(I):
        a_list = [0,1]
        distribution = [0.16, 0.84]
        random_number = random.choices(a_list, distribution)
        A.append(random_number[0])
    AS.append(sum(A))
    S.append(len(A)-sum(A))
# Asymptomatic
    for i in range(len(AS)):
        B = []
        for j in range(AS[i]):
            a_list = [0,1]
            distribution = [0.1, 0.9]
            random_number = random.choices(a_list, distribution)
            B.append(random_number[0])
    IA.append(sum(B))
    NIA.append(len(B)-sum(B))
# Doesn't makes others infected/ makes others infected probablity of death percentage 10

    for i in range(len(AS)):
        C =[]
        for j in range(AS[i]):
            a_list = [0,1,2]
            distribution = [0.01,0.8,0.17]
            random_number = random.choices(a_list, distribution)
            C.append(random_number[0])
    if a<=30:
        R1.append(0)
        D1.append(0)
        Q1.append(0)
    else:
        D1.append(C.count(0))
        R1.append(C.count(1))
        Q1.append(C.count(2))
# Sympomatic

    for i in range(len(S)):
        D=[]
        for j in range(S[i]):
            a_list = [0,1]
            distribution = [0.8, 0.2]
            random_number = random.choices(a_list, distribution)
            D.append(random_number[0])
    CS.append(sum(D))
    SS.append(len(D)-sum(D))

# Common Symptoms probability of visiting a doctor 47

    for i in range(len(CS)):
        E = []
        for  j in range(CS[i]):
            a_list = [0,1]
            distribution = [0.3, 0.7]
            random_number = random.choices(a_list, distribution)
            E.append(random_number[0])
    if a <= 30:
        DO1.append(0)
        NDO1.append(len(E))
    else:
        DO1.append(sum(E))
        NDO1.append(len(E)-sum(E))
# Common Symptoms probability of not visiting a doctor

    for i in range(len(NDO1)):
        F=[]
        for j in range(NDO1[i]):
            a_list = [0,1]
            distribution = [0.3, 0.7]
            random_number = random.choices(a_list, distribution)
            F.append(random_number[0])

    if a<=50:
        R2.append(0)
        SS1.append(len(F))
    else:
        R2.append(sum(F))
        SS1.append(len(F)-sum(F))
# Serious Symptoms

    for i in range(len(SS1)):
        G = []
        for j in range(SS1[i]):
            a_list = [0,1]
            distribution = [0.2, 0.8]
            random_number = random.choices(a_list, distribution)
            G.append(random_number[0])
    if a<= 95:
        DO2.append(0)
        NDO2.append(len(G))
    else:
        DO2.append(sum(G))
        NDO2.append(len(G)-sum(G))

# Serious Symptoms doesn't visit a Doctor

    for i in range(len(NDO2)):
        H =[]
        for j in range(NDO2[i]):
            a_list = [0,1]
            distribution = [0.6, 0.4]
            random_number = random.choices(a_list, distribution)
            H.append(random_number[0])
    if a<=70:
        R3.append(0)
        CRS1.append(len(H))
    else:
        R3.append(len(H)-sum(H))
        CRS1.append(sum(H))
# Serious Symptoms

    for i in range(len(SS)):
        J =[]
        for j in range(SS[i]):
            a_list = [0,1]
            distribution = [0.2, 0.8]
            random_number = random.choices(a_list, distribution)
            J.append(random_number[0])
    if a<= 95:
        DO3.append(0)
        NDO3.append(len(J))
    else:
        DO3.append(sum(J))
        NDO3.append(len(J)-sum(J))

# Serious Symptoms doesn't visits a Doctor
    for i in range(len(NDO3)):
        K =[]
        for j in range(NDO3[i]):
            a_list = [0,1]
            distribution = [0.6, 0.4]
            random_number = random.choices(a_list, distribution)
            K.append(random_number[0])
    if a<=70:
        R4.append(0)
        CRS2.append(len(K))
    else:
        R4.append(sum(K))
        CRS2.append(len(K)-sum(K))
# Quarantine

    number = DO1[a] + DO2[a] + DO3[a]
    Q.append(number)

    for i in range(len(Q)):
        L=[]
        for j in range(Q[i]):
            a_list = [0,1]
            distribution = [0.6, 0.4]
            random_number = random.choices(a_list, distribution)
            K.append(random_number[0])
    if a<=14:
        R5.append(0)
        CRS3.append(len(L))
    else:
        R5.append(sum(L))
        CRS3.append(len(L)-sum(L))

 # Critical Symptoms

    CRS.append(CRS1[a]+CRS2[a]+CRS3[a])
    for i in range(len(CRS)):
        N=[]
        for j in range(CRS[i]):
            a_list = [0,1]
            distribution = [0.98, 0.02]
            random_number = random.choices(a_list, distribution)
            N.append(random_number[0])
    if a<=15:
        D2.append(0)
        AB.append(len(N))
    else:
        D2.append(sum(N))
        AB.append(len(N)-sum(N))

# Artificial Breadthing

    for i in range(len(AB)):
        M=[]
        for j in range(AB[i]):
            a_list = [0,1,2]
            distribution = [0.16, 0.34, 0.5]
            random_number = random.choices(a_list, distribution)
            M.append(random_number[0])
    if a<=25:
        D3.append(0)
        R6.append(0)
        Q2.append(0)
    else:
        D3.append(M.count(0))
        R6.append(M.count(1))
        Q2.append(M.count(2))
    test.append(T)
    Death.append(D1[a]+D2[a]+D3[a])
    if a< 97:
        Recover.append(R1[a]+R2[a]+R3[a]+R4[a]+R5[a]+R6[a])
    else:
        Recover.append(round(R1[a]+R2[a]+R3[a]+R4[a]+R5[a]+R6[a]+0.5*Quarantine[a-1]+0.5*Quarantine[a-2]))

    Infected.append(I)
    RC.append(sum(Recover))
    DC.append(sum(Death))
    TC.append(sum(Infected))
    RP.append(sum(Recover)/sum(Infected))
    DP.append(sum(Death)/sum(Infected))
    IP.append((sum(Infected))/sum(test))
    Quarantine.append(Infected[a]-Recover[a]-Death[a])
    QC.append(sum(Quarantine))

    I = round((I-Q[a])*1.08)
    T = round((Quarantine[a])*90)
i = TC[0];
while i < TC[n-1]:
        X.append(i)
        i= i*2
for j in range(len(X)):
    res = next(x for x, val in enumerate(TC)
                                  if val >= X[j])
    Y.append(res+1)

for i in range(len(Y)-1):
    res = Y[i+1]-Y[i]
    Z.append(res)
print(test)
print(sum(test))
print('Total infected person',TC[n-1])
print('Recover',sum(Recover))
print("Death",sum(Death))
print("Quarantined",sum(Quarantine))
import math
DR=[]
SDR = []
for j in range(len(GovR)):
    DR.append(GovR[j]-Recover[j])
print("Average gap",sum(DR)/len(GovR))
for j in range(len(GovR)):
    SDR.append( ( Recover[j]-(sum(DR)/len(GovR) ) )*( Recover[j]-(sum(DR)/len(GovR) )) )
print("Standard Deviation",math.sqrt(sum(SDR)/(len(GovR)-1)))
print('Max Gap',max(DR))
print('Min Gap',min(DR))
if n%2 == 0:
    RQ2 = (Recover[round((n/2)-1)]+Recover[round(n/2)])/2
    if n%4 == 0:
        RQ1 = (Recover[round((n/4)-1)]+Recover[round(n/4)])/2
        RQ3 = (Recover[round((3*n/4)-1)]+Recover[round(3*n/4)])/2
    else:
        RQ1 = Recover[round((n-2)/4)]
        RQ3 = Recover[round((3*n-2)/4)]
else:
    RQ2 = Recover[(n-1)/2]
    RQ1 = Recover[(n-3)/4]
    RQ3 = Recover[(3*n-1)/4]
print('Quartile1',RQ1)
print('Quartile2',RQ2)
print('Quartile3',RQ3)
DI=[]
SDI=[]
for j in range(len(Gov)):
    DI.append(Gov[j]-Infected[j])

#print(DI)
print(sum(DI)/len(Gov))
for j in range(len(GovR)):
    SDI.append( ( Infected[j]-( sum(DI)/len(Gov) ) )*( Infected[j]-(sum(DI)/len(Gov) )) )
#print(SDI)
print(math.sqrt(sum(SDI)/(len(Gov)-1)))
print(max(DI))
print(min(DI))
if n%2 == 0:
    IQ2 = (Infected[round((n/2)-1)]+Infected[round(n/2)])/2
    if n%4 == 0:
        IQ1 = (Infected[round((n/4)-1)]+Infected[round(n/4)])/2
        IQ3 = (Infected[round((3*n/4)-1)]+Infected[round(3*n/4)])/2
    else:
        IQ1 = Infected[round((n-2)/4)]
        IQ3 = Infected[round((3*n-2)/4)]
else:
    IQ2 = Infected[(n-1)/2]
    IQ1 = Infected[(n-3)/4]
    IQ3 = Infected[(3*n-1)/4]
print('Quartile1',IQ1)
print('Quartile2',IQ2)
print('Quartile3',IQ3)
DD=[]
SDD =[]
for j in range(len(GovD)):
    DD.append((GovD[j]-Death[j]))

#print(DD)
print(sum(DD)/len(GovD))
for j in range(len(GovD)):
    SDD.append( ( Death[j]-( sum(DD)/len(GovD) ) )*( Death[j]-(sum(DD)/len(GovD) )) )
#print(SDD)
print(math.sqrt(sum(SDD)/(len(GovD)-1)))
print(max(DD))
print(min(DD))
if n%2 == 0:
    DQ2 = (Death[round((n/2)-1)]+Death[round(n/2)])/2
    if n%4 == 0:
        DQ1 = (Death[round((n/4)-1)]+Death[round(n/4)])/2
        DQ3 = (Death[round((3*n/4)-1)]+Death[round(3*n/4)])/2
    else:
        DQ1 = Death[round((n-2)/4)]
        DQ3 = Death[round((3*n-2)/4)]
else:
    DQ2 = Death[(n-1)/2]
    DQ1 = Death[(n-3)/4]
    DQ3 = Death[(3*n-1)/4]
print('Quartile1',DQ1)
print('Quartile2',DQ2)
print('Quartile3',DQ3)
```

Observation and Challenges:

The main Observations I made while preforming this project:-

- The suppression strategy allowing aggressive combined testing and tracing of contacts decreases the Infection rate and increases the Recovery rate.
- As the number of contacts decreases the peaks goes down.
- The consideration of presymptomatic days are important as it directly affect the shifting of peak of our curve.

The challenges I faced during this project:-

- The biggest challenge for me while trying to comprehend this code was that I wasn't aware about the various parameters which affect the process, So to get over this I applied Sensitivity Analysis and for that I had to run the code over hundred times and subsequently got hold of various parameters involved.
- As each parameter affects other parameters, It was tough to get desired result.
- Without knowing the actual data plotting the various curves was difficult. But, being aware of the trends made the work a bit easier.

References:

Guidelines for Quarantine facilities COVID-19
https://www.mohfw.gov.in/pdf/90542653311584546120quartineguidelines.pdf

Indian Council of Medical Research (ICMR)
https://www.downtoearth.org.in/news/health/covid-19-are-80-cases-in-india-really-asymptomatic-70590 April 20, 2020.

Day-wise update of infected, recovered and died person Worldometer
https://www.worldometers.info/world-population/india-population/

COVID-19 Virtual Press conference
https://www.who.int/docs/default-source/coronaviruse/transcripts/who-audio-emergencies-coronavirus-press-conference-08jun2020.pdf?sfvrsn=f6fd460a_0 8 June 2020

Corona virus disease 2019 (COVID-19) Situation Report – 46
19.pdf?sfvrsn=96b04adf_4https://www.who.int/docs/default-source/coronaviruse/situation-reports/20200306-sitrep-46-covid-19.pdf?sfvrsn=96b04adf_4 06 March 2020