Identifying Epidemic Parameters of COVID-19 Based on U.S. Case Data

Harley Jackson

Abstract

Publicly available COVID-19 case data from the U.S. were retrieved and processed to model the pandemic within the United States using a modified version of the SIR model, dubbed the SIRD model. The additional state D represents deceased and is placed into the model, along with its corresponding parameter delta, in order to quantitatively model that any amount of cases in the I state will lead to some cases ending up in the D state, representing deaths, which are important to prevent. The SIRD model was analyzed based on actual data, including deriving actual parameters beta, gamma, and delta over each day. The parameters were then iterated through and used to simulate the SIRD states and compared to the actual SIRD states. Mean squared error was tracked for each iteration, and then the minimum MSE was used to select the parameters beta, gamma, and delta that would be used to represent the predictive SIRD model. Based on the predictive model, qualities were examined by simulating to t=10000, and a vaccination strategy was calculated, with limitations due to data accuracy.

I. Introduction

The COVID-19 virus has been infecting individuals around the world for over a year and is still ongoing as of May 2021, including a recent surge in India [1]. Having accurate mathematical models of COVID-19's spread within populations is crucial to being able to allocate resources correctly, namely by being able to predict how the disease will spread under different conditions and how to stabilize and possibly eradicate disease spread altogether [2]. Hota, Godbole, Bhariya, and Paré point out that COVID-19 is best modeled under SIR, since there is no strong evidence of becoming reinfected with the virus after recovering [3] [4].

The goal of this project is to identify the parameters of the SIR model for COVID-19, using public test data made publicly available by the CDC [5], which has been done with Indian case data [6] and the pandemic in Brazil [7]. Since the data includes deaths, and deaths from COVID-19 is a great concern in the global pandemic, it may be useful to have a separate compartment for deaths. Slighton, et. al extended the SIR model to an SI(S/D) model for Ebola to account for deaths and the possibility to be reinfected [8]. Since deaths are important to prevent and are a major risk of COVID-19 infection, an SIRD model will be examined with parameters inferred from the U.S. case data. Once the parameters for the SIRD model are calculated, the values of β and γ will be used to recommend a vaccine mitigation strategy for converging the I state to zero.

The data set does not include network information, even though individual characteristics are represented, including the date of report to CDC, date of positive laboratory specimen, and death if applicable. The parameters will be inferred based on aggregations and assuming uniform interactions. The project results include insights into how to mitigate disease spread based on the parameters of the COVID-19 model, with the goal of stabilization and positive outcomes for the public (minimization of disease spread and death, especially). The project also produces a public repository of Python code on Github that can be cloned and used to reproduce or extend the analysis.

II. Data Characteristics and Process

A. Data Characteristics

The data set contains 20,565,345 rows, each representing an individual and having up to 12 fields, with the description of each field coming from the CDC form available at www.cdc.gov/coronavirus/2019-ncov/downloads/pui-form.pdf The fields are as follows:

- **cdc_report_dt**(DATE): Report date of case to CDC
- **pos_spec_dt**(DATE): Date of first positive specimen collection
- **onset_dt**(DATE): Date of symptom onset
- **cdc_case_earliest_dt**(DATE): $argmin(cdc\_report\_dt, pos\_spec\_dt, onset\_dt)$
- **current_status**(BINARY): Lab-confirmed case OR Probable case
- **sex**(CATEGORICAL): Sex of case
- **age_group**(CATEGORICAL): Age group of case
- **race_ethnicity_combined**(CATEGORICAL): Race/Ethnicity of case
- **hosp_yn**(BINARY): Was the patient hospitalized? Y/N
- **icu_yn**(BINARY): Was the patient admitted to the intensive care unit? Y/N
- **death_yn**(BINARY): Did the patient die as a result of this illness? Y/N
- **medcond_yn**(BINARY): Did they have any underlying medical conditions and/or risk behaviors? Y/N

The author is with the School of Electrical and Computer Engineering at Purdue University. Email: jacks832@purdue.edu

B. Process

The data is being processed, manipulated, and modeled using Python, in Jupyter Notebook, including libraries such as Pandas and Numpy. Please see the code in the Appendix.

1) The data were cleaned by first checking that all rows had a valid date for the field **cdc_case_earliest_dt**. Since the date was present for all rows, none were dropped. In examining the field **death_yn**, it was found that some fields were 'missing' or 'None.' Instead of dropping these rows, they were changed to 'No.' Each row is translated into 410-column row, each column representing the $t = 0...409$, where 0 represents 2020/01/01 and 490 represents 2021/02/13. The value in each field is either 'S', 'I', 'R', or 'D', representing the state the case is in at time t. **cdc_case_earliest_dt** is used to transition the case from S to I at time t_i, and the case will be transitioned from state I to R at $t_i + 11$, unless the patient died. If the patient died, the case will be transitioned from state I to state D at time $t_i + 19$. Because the data does not included recovery time, the recovery or death time is based on the average recovery time found for COVID-19 in a 2020 study, specifically the average recovery time for mild and severe cases was found to be 10.63 ± 1.93 days and 18.70 ± 2.50 days, respectively [9]. It is assumed that cases resulting in death were severe and that others were mild.

2) Based on the aggregated amount of cases in each state at time t, with the total population being the population of the United States on January 1, 2020 - 329,168,006 according to the U.S. Census Population Clock [10], the parameters β, γ, and δ were estimated for each day t from 1/1/2020 to 2/13/2021. Please see Figure 1 for a visual depiction of the SIRD model with transition parameters.

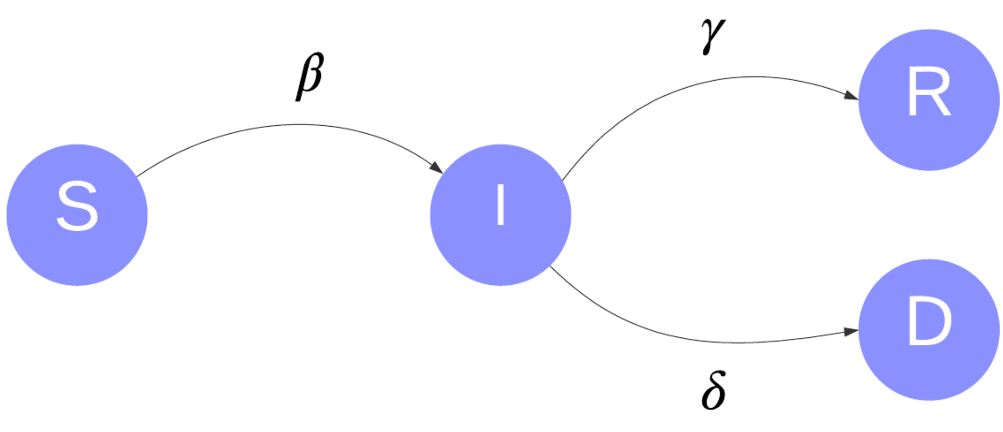

Fig. 1: SIRD Diagram

The steps below were carried out to derive the equations needed to estimate parameters at each time step t:

a) Continuous time state space equations

$$\dot{S} = -\beta S(t)I(t) \tag{1}$$
$$\dot{I} = \beta S(t)I(t) - \gamma I(t) - \delta I(t) \tag{2}$$
$$\dot{R} = \gamma I(t) \tag{3}$$
$$\dot{D} = \delta I(t) \tag{4}$$
$$\tag{5}$$

b) Discrete time equations

$$S[k+1] = S[k] - h\{\beta S[k]I[k]\} \tag{6}$$
$$I[k+1] = I[k] + h\{\beta S[k]I[k] - \gamma I[k] - \delta I[k]\} \tag{7}$$
$$R[k+1] = R[k] + h\{\gamma I[k]\} \tag{8}$$
$$D[k+1] = D[k] + h\{\delta I[k]\} \tag{9}$$
$$\tag{10}$$

c) Discrete time in matrix form.

$$\begin{bmatrix} S[k+1]-S[k] \\ I[k+1]-I[k] \\ R[k+1]-R[k] \\ D[k+1]-D[k] \end{bmatrix} = \begin{bmatrix} -S[k]I[k] & 0 & 0 \\ S[k]I[k] & -I[k] & -I[k] \\ 0 & I[k] & 0 \\ 0 & 0 & I[k] \end{bmatrix} \begin{bmatrix} h\beta \\ h\gamma \\ h\delta \end{bmatrix} \tag{11}$$

d) For identifying parameters, we can omit the state D and rearrange the equation. This was directly translated into code for the method $params\_at\_t$:

$$\begin{bmatrix} \beta \\ \gamma \\ \delta \end{bmatrix} = \begin{bmatrix} -S[k]I[k] & 0 & 0 \\ S[k]I[k] & -I[k] & -I[k] \\ 0 & I[k] & 0 \end{bmatrix}^{-1} \begin{bmatrix} S[k+1]-S[k] \\ I[k+1]-I[k] \\ R[k+1]-R[k] \end{bmatrix} \tag{12}$$

3) With the parameters β, γ, and δ estimated at each $t \in [1, 409]$, states S, I, R, and D were simulated over the entire range for each daily β, γ, and δ. The mean squared error was then calculated for each of S, I, R, and D, and the minimum of the mean squared errors were used to select the set of parameters to estimate the overall model for the COVID-19 virus on the interval $[0, 409]$.

$$MSE_{state} = \frac{1}{t}\sum_{i=1}^{n}(state_i - \hat{state_i})^2, state \in S, I, R, D \tag{13}$$

Take the overall mean of the MSE:

$$MSE_{overall} = \frac{1}{t}\sum_{i=1}^{n} MSE_{state}, state \in S, I, R, D \tag{14}$$

Use lowest overall MSE for simulation:

$$index = argmin(MSE_{overall}(t)), t \in [1, 409] \tag{15}$$
$$simulation\_parameters = beta(index), gamma(index), delta(index) \tag{16}$$

4) Based on parameters, recommend a vaccination strategy based on $S(0)$, β, and γ:

$$\rho_v > 1 - \frac{\gamma}{\beta S(0)} \tag{17}$$

5) Made code available in Github, along with a Readme file for straightforward reproducability

III. Model and Simulation

After the data were cleaned and processed as outlined in Step 1 of the Process in the previous section, the number of cases in each state, SIRD, were stored in an array for days 0 through 409 covered by the dataset. The data were then charted to get a visual picture of the trend. The actual values of SIRD are charted in Figures 2-4, with Figures 3 and 4 being zoomed in for more detail.

The parameters β, γ, and δ were predicted for each day $t \in [1, 409]$. In Figures 5-7, you can see the parameter values plotted over time. Using the mean squared error for each set of parameters, as compared with the actual values of SIRD, the value found to have the lowest MSE was that at index 277. So a simulation was carried out using the parameters at index 277, namely:

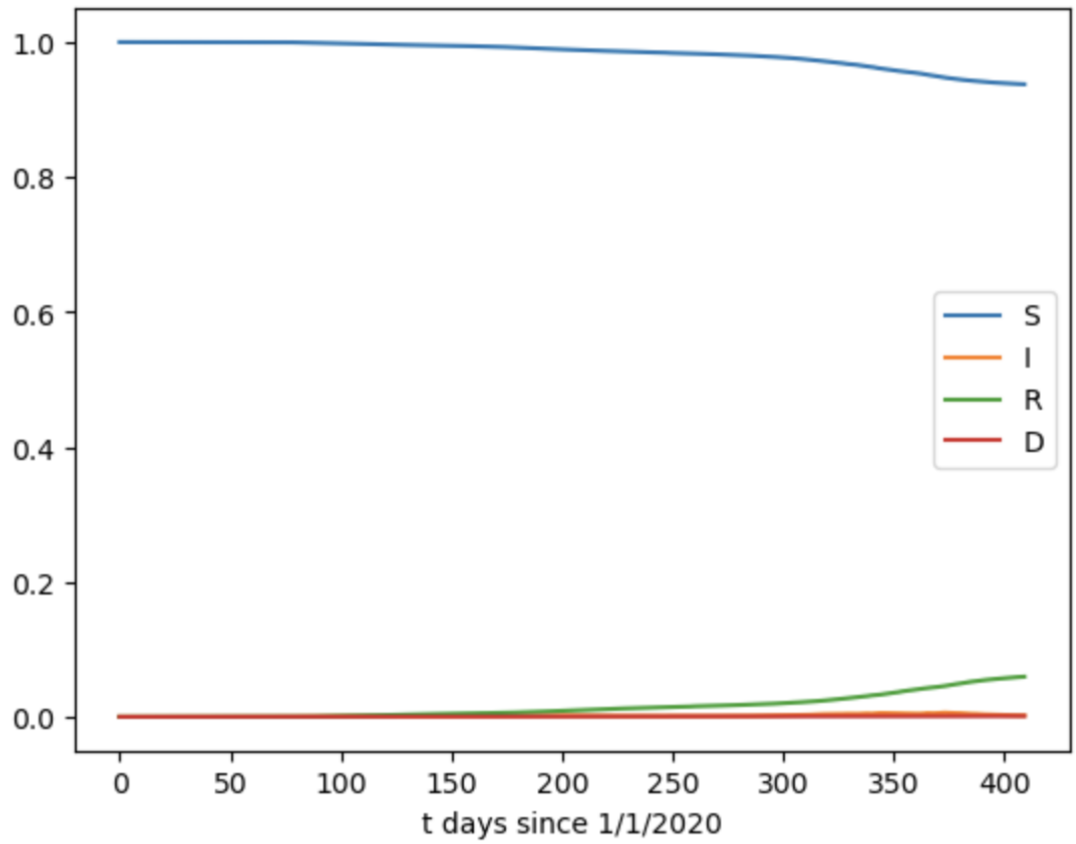

Fig. 2: Actual SIRD Values

$$\beta = 3.24 * 10^{-10} \tag{18}$$
$$\gamma = 0.0795 \tag{19}$$
$$\delta = 0.0012 \tag{20}$$
$$\tag{21}$$

The predicted vs actual graphs of each state can be seen in Figures 8-11, and the model can be seen simulated to t=10000 in Figure 12.

IV. Analysis

Based on the parameters selected from the daily parameters of the data, using the minimal MSE, the model can be seen to become asymptotically stable when we run the simulation to t=10000, visualized in Figure 12. While our model becomes asymptotically stable without intervention, assuming the parameters stay at:

$$\beta = 3.24 * 10^{-10} \tag{22}$$
$$\gamma = 0.0795 \tag{23}$$
$$\delta = 0.0012 \tag{24}$$
$$\tag{25}$$

it is worth examining how to intervene so as to minimize infections, which in turn minimizes deaths, as δ will always lead to nonzero deaths so long as there are infections. The percentage of population that should be vaccinated in order to achieve herd immunity would be given by the equation:

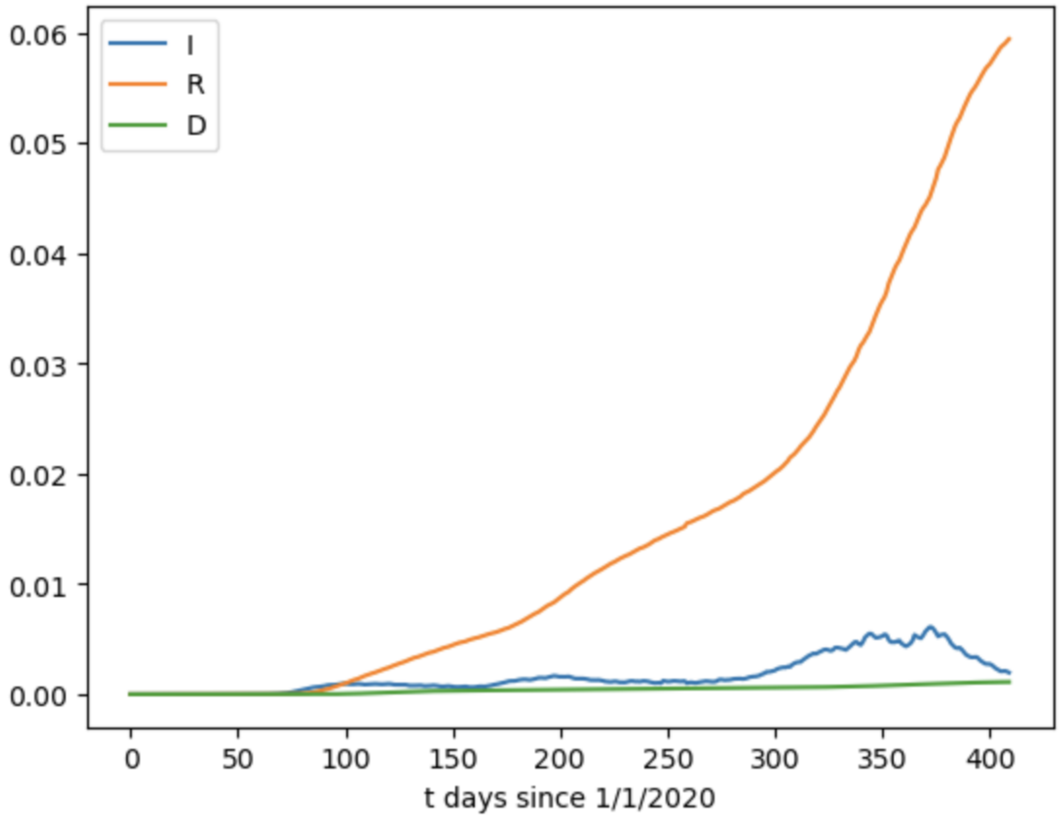

Fig. 3: Actual I, R, & D Values

$$\rho_v > 1 - \frac{\gamma}{\beta S(0)} \tag{26}$$

From our data and the simulated model with the lowest MSE, we have:

$$\beta = 3.24 * 10^{-10} \tag{27}$$
$$\gamma = 0.0795 \tag{28}$$
$$S(0) = 329167741 \tag{29}$$
$$\rho_v > 1 - \frac{0.0795}{3.24 * 10^{-10} * 329167741} \tag{30}$$
$$\rho_v > 1 - 0.745 \tag{31}$$
$$\rho_v > 0.255 \tag{32}$$
$$\tag{33}$$

The calculation in lines 30-32 suggests that a vaccination level of 26% of the population should lead to herd immunity. However, this assumes that the model parameters are accurate.

V. Conclusion

Upon modeling the COVID-19 case data from the CDC, it was found that the parameters observed at day 277 most closely modeled the overall behavior on the range 0 to 410 days, when each day's parameters were compared based on minimal mean squared error. Using the parameters of day 277, the model was simulated for up to 10000 days and found to be asymptotically stable without intervention, with the states I and D converging to zero shortly after 2000 days. However, there are limitations to the model, such as it's inaccuracy and basis on limited data. COVID-19 cases that were not reported are not included, namely cases that were asymptomatic or never brought to a healthcare facility, where the case would be reported.

With the limitations in mind, the percentage needed to be vaccinated was calculated at about 26% based on the estimated parameters of the model. Even though the model leads to I and D converging to zero on their own, it appears to be after 2000 days. Moreover, there are deaths happening anytime that there are cases in state I. Herd immunity should be achieved before then in order to curtail the pandemic, shorten the period required for I and D to converge to zero, and thus minimize deaths. The source code can be downloaded and/or forked from the following Github repository: https://github.com/harleyjj/COVID-19Project

APPENDIX

[3]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[85]:
```python
!python --version
```

```
Python 3.8.5
```

[2]:
```python
# import csv file into Pandas dataframe
covid_data = pd.read_csv("COVID-19_Case_Surveillance_Public_Use_Data.csv")
type(covid_data)
```

```
/Users/hjackson/opt/anaconda3/envs/tf/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3156: DtypeWarning: Columns (2) have
mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

[2]:
```
pandas.core.frame.DataFrame
```

[4]:
```python
len(covid_data)
covid_data.shape
```

[4]:
```
(20565345, 12)
```

[30]:
```python
covid_data.head()
```

[30]:
```
  cdc_case_earliest_dt cdc_report_dt pos_spec_dt    onset_dt  \
0           2020/01/01    2021/01/31  2020/01/01         NaN
1           2020/01/01    2021/02/02  2020/01/01         NaN
2           2020/01/02    2021/01/27  2020/01/02         NaN
3           2020/01/02    2021/02/02  2020/01/02         NaN
4           2021/01/01    2020/01/03  2020/01/03  2021/01/01

               current_status     sex    age_group race_ethnicity_combined  \
0  Laboratory-confirmed case  Female  0 - 9 Years                 Unknown
1  Laboratory-confirmed case    Male  0 - 9 Years                 Unknown
2  Laboratory-confirmed case    Male  0 - 9 Years                 Unknown
3  Laboratory-confirmed case    Male  0 - 9 Years                 Unknown
4  Laboratory-confirmed case    Male  0 - 9 Years     White, Non-Hispanic

   hosp_yn   icu_yn death_yn medcond_yn  infection_t  deceased_t  recovered_t  \
0  Missing  Missing       No    Missing            0          -1           11
1  Missing  Missing       No    Missing            0          -1           11
2  Missing  Missing       No    Missing            1          -1           12
3  Missing  Missing       No    Missing            1          -1           12
4       No  Missing       No        Yes          366          -1          377
```

```
   infection_end_t
0               11
1               11
2               12
3               12
4              377
```

[6]:
```
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20565345 entries, 0 to 20565344
Data columns (total 12 columns):
cdc_case_earliest_dt       object
cdc_report_dt              object
pos_spec_dt                object
onset_dt                   object
current_status             object
sex                        object
age_group                  object
race_ethnicity_combined    object
hosp_yn                    object
icu_yn                     object
death_yn                   object
medcond_yn                 object
dtypes: object(12)
memory usage: 1.8+ GB
```

[7]:
```
covid_data.describe()
```

[7]:
```
       cdc_case_earliest_dt  cdc_report_dt pos_spec_dt    onset_dt  \
count              20565345       18233264     5933794     9278464
unique                  410            424         418         424
top              2020/12/31     2020/12/31  2021/01/04  2020/11/30
freq                 277049         256387       70590       76115

                  current_status       sex      age_group  \
count                   20565345  20565330       20565268
unique                         2         5             10
top     Laboratory-confirmed case    Female  20 - 29 Years
freq                    18856641  10621441        3835094

       race_ethnicity_combined   hosp_yn    icu_yn  death_yn medcond_yn
count                 20565337  20565345  20565345  20565345   20565345
unique                       9         6         4         4          4
top                    Unknown   Missing   Missing        No    Missing
freq                   8112749   9047578  16696875  10517575   16097975
```

[22]:
```
covid_data.keys()
```

[22]:
```
Index(['cdc_case_earliest_dt', 'cdc_report_dt', 'pos_spec_dt', 'onset_dt',
       'current_status', 'sex', 'age_group', 'race_ethnicity_combined',
       'hosp_yn', 'icu_yn', 'death_yn', 'medcond_yn', 'infection_t'],
      dtype='object')
```

[15]:
```
# Remove space from the column name
covid_data.rename(columns={'cdc_case_earliest_dt ': 'cdc_case_earliest_dt'},␣
 ↪inplace=True)
```

[26]:
```
# Replace missing and unknown values for death_yn column with 'No'
covid_data.death_yn.replace({'Missing': 'No', 'Unknown': 'No'}, inplace=True)
```

[27]:
```
covid_data.death_yn.unique()
```

[27]:
```
array(['No', 'Yes'], dtype=object)
```

[32]:
```
# pickle updated dataframe
covid_data.to_pickle('cdc_df')
```

[3]:
```
covid_data = pd.read_pickle('cdc_df')
```

[5]:
```
unique_ages = covid_data.age_group.unique()
print(unique_ages)
```

```
['0 - 9 Years' '10 - 19 Years' '20 - 29 Years' '60 - 69 Years'
 '70 - 79 Years' '40 - 49 Years' '50 - 59 Years' '80+ Years'
 '30 - 39 Years' 'Missing' nan]
```

[8]:
```
# Get uniques dates and sort them in ascending order
unique_dates = covid_data.cdc_case_earliest_dt.unique()
unique_dates.sort()
print(unique_dates)
```

```
['2020/01/01' '2020/01/02' '2020/01/03' '2020/01/04' '2020/01/05'
 '2020/01/06' '2020/01/07' '2020/01/08' '2020/01/09' '2020/01/10'
 '2020/01/11' '2020/01/12' '2020/01/13' '2020/01/14' '2020/01/15'
 '2020/01/16' '2020/01/17' '2020/01/18' '2020/01/19' '2020/01/20'
 '2020/01/21' '2020/01/22' '2020/01/23' '2020/01/24' '2020/01/25'
 '2020/01/26' '2020/01/27' '2020/01/28' '2020/01/29' '2020/01/30'
 '2020/01/31' '2020/02/01' '2020/02/02' '2020/02/03' '2020/02/04'
 '2020/02/05' '2020/02/06' '2020/02/07' '2020/02/08' '2020/02/09'
 '2020/02/10' '2020/02/11' '2020/02/12' '2020/02/13' '2020/02/14'
 '2020/02/15' '2020/02/16' '2020/02/17' '2020/02/18' '2020/02/19'
 '2020/02/20' '2020/02/21' '2020/02/22' '2020/02/23' '2020/02/24'
 '2020/02/25' '2020/02/26' '2020/02/27' '2020/02/28' '2020/02/29'
 '2020/03/01' '2020/03/02' '2020/03/03' '2020/03/04' '2020/03/05'
```

```
'2020/03/06' '2020/03/07' '2020/03/08' '2020/03/09' '2020/03/10'
'2020/03/11' '2020/03/12' '2020/03/13' '2020/03/14' '2020/03/15'
'2020/03/16' '2020/03/17' '2020/03/18' '2020/03/19' '2020/03/20'
'2020/03/21' '2020/03/22' '2020/03/23' '2020/03/24' '2020/03/25'
'2020/03/26' '2020/03/27' '2020/03/28' '2020/03/29' '2020/03/30'
'2020/03/31' '2020/04/01' '2020/04/02' '2020/04/03' '2020/04/04'
'2020/04/05' '2020/04/06' '2020/04/07' '2020/04/08' '2020/04/09'
'2020/04/10' '2020/04/11' '2020/04/12' '2020/04/13' '2020/04/14'
'2020/04/15' '2020/04/16' '2020/04/17' '2020/04/18' '2020/04/19'
'2020/04/20' '2020/04/21' '2020/04/22' '2020/04/23' '2020/04/24'
'2020/04/25' '2020/04/26' '2020/04/27' '2020/04/28' '2020/04/29'
'2020/04/30' '2020/05/01' '2020/05/02' '2020/05/03' '2020/05/04'
'2020/05/05' '2020/05/06' '2020/05/07' '2020/05/08' '2020/05/09'
'2020/05/10' '2020/05/11' '2020/05/12' '2020/05/13' '2020/05/14'
'2020/05/15' '2020/05/16' '2020/05/17' '2020/05/18' '2020/05/19'
'2020/05/20' '2020/05/21' '2020/05/22' '2020/05/23' '2020/05/24'
'2020/05/25' '2020/05/26' '2020/05/27' '2020/05/28' '2020/05/29'
'2020/05/30' '2020/05/31' '2020/06/01' '2020/06/02' '2020/06/03'
'2020/06/04' '2020/06/05' '2020/06/06' '2020/06/07' '2020/06/08'
'2020/06/09' '2020/06/10' '2020/06/11' '2020/06/12' '2020/06/13'
'2020/06/14' '2020/06/15' '2020/06/16' '2020/06/17' '2020/06/18'
'2020/06/19' '2020/06/20' '2020/06/21' '2020/06/22' '2020/06/23'
'2020/06/24' '2020/06/25' '2020/06/26' '2020/06/27' '2020/06/28'
'2020/06/29' '2020/06/30' '2020/07/01' '2020/07/02' '2020/07/03'
'2020/07/04' '2020/07/05' '2020/07/06' '2020/07/07' '2020/07/08'
'2020/07/09' '2020/07/10' '2020/07/11' '2020/07/12' '2020/07/13'
'2020/07/14' '2020/07/15' '2020/07/16' '2020/07/17' '2020/07/18'
'2020/07/19' '2020/07/20' '2020/07/21' '2020/07/22' '2020/07/23'
'2020/07/24' '2020/07/25' '2020/07/26' '2020/07/27' '2020/07/28'
'2020/07/29' '2020/07/30' '2020/07/31' '2020/08/01' '2020/08/02'
'2020/08/03' '2020/08/04' '2020/08/05' '2020/08/06' '2020/08/07'
'2020/08/08' '2020/08/09' '2020/08/10' '2020/08/11' '2020/08/12'
'2020/08/13' '2020/08/14' '2020/08/15' '2020/08/16' '2020/08/17'
'2020/08/18' '2020/08/19' '2020/08/20' '2020/08/21' '2020/08/22'
'2020/08/23' '2020/08/24' '2020/08/25' '2020/08/26' '2020/08/27'
'2020/08/28' '2020/08/29' '2020/08/30' '2020/08/31' '2020/09/01'
'2020/09/02' '2020/09/03' '2020/09/04' '2020/09/05' '2020/09/06'
'2020/09/07' '2020/09/08' '2020/09/09' '2020/09/10' '2020/09/11'
'2020/09/12' '2020/09/13' '2020/09/14' '2020/09/15' '2020/09/16'
'2020/09/17' '2020/09/18' '2020/09/19' '2020/09/20' '2020/09/21'
'2020/09/22' '2020/09/23' '2020/09/24' '2020/09/25' '2020/09/26'
'2020/09/27' '2020/09/28' '2020/09/29' '2020/09/30' '2020/10/01'
'2020/10/02' '2020/10/03' '2020/10/04' '2020/10/05' '2020/10/06'
'2020/10/07' '2020/10/08' '2020/10/09' '2020/10/10' '2020/10/11'
'2020/10/12' '2020/10/13' '2020/10/14' '2020/10/15' '2020/10/16'
'2020/10/17' '2020/10/18' '2020/10/19' '2020/10/20' '2020/10/21'
'2020/10/22' '2020/10/23' '2020/10/24' '2020/10/25' '2020/10/26'
'2020/10/27' '2020/10/28' '2020/10/29' '2020/10/30' '2020/10/31'
```

```
'2020/11/01' '2020/11/02' '2020/11/03' '2020/11/04' '2020/11/05'
'2020/11/06' '2020/11/07' '2020/11/08' '2020/11/09' '2020/11/10'
'2020/11/11' '2020/11/12' '2020/11/13' '2020/11/14' '2020/11/15'
'2020/11/16' '2020/11/17' '2020/11/18' '2020/11/19' '2020/11/20'
'2020/11/21' '2020/11/22' '2020/11/23' '2020/11/24' '2020/11/25'
'2020/11/26' '2020/11/27' '2020/11/28' '2020/11/29' '2020/11/30'
'2020/12/01' '2020/12/02' '2020/12/03' '2020/12/04' '2020/12/05'
'2020/12/06' '2020/12/07' '2020/12/08' '2020/12/09' '2020/12/10'
'2020/12/11' '2020/12/12' '2020/12/13' '2020/12/14' '2020/12/15'
'2020/12/16' '2020/12/17' '2020/12/18' '2020/12/19' '2020/12/20'
'2020/12/21' '2020/12/22' '2020/12/23' '2020/12/24' '2020/12/25'
'2020/12/26' '2020/12/27' '2020/12/28' '2020/12/29' '2020/12/30'
'2020/12/31' '2021/01/01' '2021/01/02' '2021/01/03' '2021/01/04'
'2021/01/05' '2021/01/06' '2021/01/07' '2021/01/08' '2021/01/09'
'2021/01/10' '2021/01/11' '2021/01/12' '2021/01/13' '2021/01/14'
'2021/01/15' '2021/01/16' '2021/01/17' '2021/01/18' '2021/01/19'
'2021/01/20' '2021/01/21' '2021/01/22' '2021/01/23' '2021/01/24'
'2021/01/25' '2021/01/26' '2021/01/27' '2021/01/28' '2021/01/29'
'2021/01/30' '2021/01/31' '2021/02/01' '2021/02/02' '2021/02/03'
'2021/02/04' '2021/02/05' '2021/02/06' '2021/02/07' '2021/02/08'
'2021/02/09' '2021/02/10' '2021/02/11' '2021/02/12' '2021/02/13']
```

[9]:
```python
# Map of dates to t value
date_to_index = {}
for i in range(len(unique_dates)):
    date_to_index[unique_dates[i]] = i
```

[10]:
```python
date_to_index['2021/02/09']
```

[10]:
```
405
```

[18]:
```python
array_of_ints = [i for i in range(410)]
```

[6]:
```python
# initialize arrays for number in state at each time t
susceptible = [0] * 410
infected = [0] * 410
recovered = [0] * 410
deceased = [0] * 410
```

[24]:
```python
severe = 19
mild = 11
```

[13]:
```python
covid_data["infection_t"] = covid_data['cdc_case_earliest_dt'].map(date_to_index)
```

[25]:
```python
covid_data["deceased_t"] = covid_data.apply(lambda x: x['infection_t'] + severe␣
↪if x['death_yn'] == 'Yes' else -1, axis=1)
```

[27]:
```
covid_data["recovered_t"] = covid_data.apply(lambda x: x['infection_t'] + mild␣
↪if x['death_yn'] == 'No' else -1, axis=1)
```

[29]:
```
covid_data["infection_end_t"] = covid_data[["deceased_t", "recovered_t"]].
↪max(axis=1)
```

[45]:
```
len(state_by_day)
```

[45]: 166606

[9]:
```
# def soc_iter(TEAM,home,away,ftr):
#     df['Draws'] = 'No_Game'
#     df.loc[((home == TEAM) & (ftr == 'D')) | ((away == TEAM) & (ftr == 'D')),␣
↪'Draws'] = 'Draw'
#     df.loc[((home == TEAM) & (ftr != 'D')) | ((away == TEAM) & (ftr != 'D')),␣
↪'Draws'] = 'No_Draw'

details = details.apply(lambda x : True
            if x['College'] == "Geu" else False, axis = 1)

# Count number of True in the series
num_rows = len(details[details == True].index)

# function for mapping each row of covid data to the states_by_day dataframe
def get_counts_of_S_at_t(t):
    len(covid_data[(date_to_index[covid_data['cdc_case_earliest_dt']] > t)])
    counts = covid_data.apply(lambda x : True if covid)
```

[18]:
```
# Construct number of susceptible at time t
for i in range(len(susceptible)):
    susceptible[i] = len(covid_data[covid_data['infection_t'] > i])
    if i % 10 == 0:
        print(i)
```

```
0
10
20
30
40
50
60
70
80
90
100
110
120
```

```
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
```

[33]:

```
# Construct number of infected at time t
for i in range(len(infected)):
    recover = i + mild
    die = i + severe
    infected[i] = len(covid_data[(covid_data['infection_t'] <= i) &␣
↪(covid_data['infection_end_t'] > i)])
    if i % 10 == 0:
        print(i)
```

```
0
10
20
30
40
50
60
70
80
90
```

```
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
```

[35]:

```
# Construct number of recovered at time t
for i in range(len(recovered)):
    recovered[i] = len(covid_data[(covid_data['recovered_t'] >= 0) &␣
 ↪(covid_data['recovered_t'] <= i)])
    if i % 10 == 0:
        print(i)
```

```
0
10
20
30
40
50
60
70
80
```

```
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
```

[37]:
```python
# Construct number of recovered at time t
for i in range(len(deceased)):
    deceased[i] = len(covid_data[(covid_data['deceased_t'] >= 0) &␣
 ↪(covid_data['deceased_t'] <= i)])
    if i % 10 == 0:
        print(i)
```

```
0
10
20
30
40
50
60
70
```

```
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
370
380
390
400
```

[44]:
```
states = [susceptible, infected, recovered, deceased]
state_counts_by_day = pd.DataFrame(states)
```

[99]:
```
state_counts_by_day.head()
```

[99]:

	S	I	R	D
0	0.999999	8.050600e-07	0.0	0.0
1	0.999999	1.403539e-06	0.0	0.0
2	0.999998	1.892651e-06	0.0	0.0
3	0.999997	2.916444e-06	0.0	0.0
4	0.999996	3.627327e-06	0.0	0.0

[54]:
```
state_counts_by_day['S'] = state_counts_by_day['S'].add(329168006 -
↪len(covid_data))
```

[89]:
```
state_counts_by_day = state_counts_by_day/329168006
```

[120]:
```
plt.plot(state_counts_by_day['S'], label="S")
plt.plot(state_counts_by_day['I'], label="I")
plt.plot(state_counts_by_day['R'], label="R")
plt.plot(state_counts_by_day['D'], label="D")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
```

[121]:
```
plt.plot(state_counts_by_day['I'], label="I")
plt.plot(state_counts_by_day['R'], label="R")
plt.plot(state_counts_by_day['D'], label="D")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
```

[122]:
```
plt.plot(state_counts_by_day['S'], label="S")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
```

[58]:
```
state_counts_by_day.to_pickle('sird_df')
```

[4]:
```
state_counts_by_day = pd.read_pickle('sird_df')
```

[5]:
```
s_array = state_counts_by_day.loc[:,'S']
i_array = state_counts_by_day.loc[:,'I']
r_array = state_counts_by_day.loc[:,'R']
```

[6]:
```
d_array = state_counts_by_day.loc[:,'D']
```

[13]:
```
s_array[0]
```

[13]: 329167741

[7]:
```
def params_at_t(s, i, r, k):
    M = np.array([[-s[k-1]*i[k-1], 0, 0],
              [s[k-1]*i[k-1], -i[k-1], -i[k-1]],
              [0, i[k-1], 0]])
    dots = np.array([[s[k]-s[k-1]],
                    [i[k]-i[k-1]],
                    [r[k]-r[k-1]]])
    M = np.linalg.inv(M)
```

```
    params = M.dot(dots)
    return params[0][0], params[1][0], params[2][0]
```

[8]:
```
betas = []
gammas = []
deltas = []

for t in range(1, len(s_array)):
    b, g, d = params_at_t(s_array, i_array, r_array, t)
    betas.append(b)
    gammas.append(g)
    deltas.append(d)
```

[9]:
```
def mse_of_arrays(array1, array2):
    diff = np.subtract(array1, array2)
    square = np.power(diff, 2)
    return np.average(square)

def mse_of_params_at_t(b, g, delta, s, i, r, d, k):
    s_new = [s[0]]
    i_new = [i[0]]
    r_new = [r[0]]
    d_new = [d[0]]
    mse_array = []
    for i in range(1, len(s)):
        s_next = s_new[i-1] - b*s_new[i-1]*i_new[i-1]
        i_next = i_new[i-1] + b*s_new[i-1]*i_new[i-1] - g*i_new[i-1] -␣
 ↪delta*i_new[i-1]
        r_next = r_new[i-1] + g*i_new[i-1]
        d_next = d_new[i-1] + delta*i_new[i-1]
        s_new.append(s_next)
        i_new.append(i_next)
        r_new.append(r_next)
        d_new.append(d_next)
    mse_array.append(mse_of_arrays(s, s_new))
    mse_array.append(mse_of_arrays(i, i_new))
    mse_array.append(mse_of_arrays(r, r_new))
    mse_array.append(mse_of_arrays(d, d_new))
    return np.average(mse_array)
```

[10]:
```
mse_values = []
for i in range(len(s_array)-1):
    this_mse = mse_of_params_at_t(betas[i], gammas[i], deltas[i], s_array,␣
 ↪i_array, r_array, d_array, i)
    mse_values.append(this_mse)
```

[11]:
```
np.argmin(mse_values)
```

[11]: 277

[12]:
```
print("The values at 277 are: beta {}, gamma {}, delta {}".format(betas[277],␣
↪gammas[277], deltas[277]))
```

```
The values at 277 are: beta 3.2431585128849923e-10, gamma 0.07951436225486924,
delta 0.001203607485723554
```

[24]:
```
def simulate_params(b, g, delta, s_0, i_0, r_0, d_0, t_max):
    s_new = [s_0]
    i_new = [i_0]
    r_new = [r_0]
    d_new = [d_0]
    for i in range(1, t_max):
        s_next = s_new[i-1] - b*s_new[i-1]*i_new[i-1]
        i_next = i_new[i-1] + b*s_new[i-1]*i_new[i-1] - g*i_new[i-1] -␣
↪delta*i_new[i-1]
        r_next = r_new[i-1] + g*i_new[i-1]
        d_next = d_new[i-1] + delta*i_new[i-1]
        s_new.append(s_next)
        i_new.append(i_next)
        r_new.append(r_next)
        d_new.append(d_next)
    return s_new, i_new, r_new, d_new
```

[29]:
```
s_simulated, i_simulated, r_simulated, d_simulated = simulate_params(betas[277],␣
↪gammas[277], deltas[277], s_array[0], i_array[0], r_array[0], d_array[0],␣
↪10000)
```

[30]:
```
plt.plot(s_simulated, label="S")
plt.plot(i_simulated, label="I")
plt.plot(r_simulated, label="R")
plt.plot(d_simulated, label="D")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
```

[31]:
```
s_simulated[2000]
```

[31]: 182386243.03023988

[17]:
```
plt.plot(s_array, label="actual")
plt.plot(s_simulated, label="predicted")
plt.title('S Predicted vs Actual')
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
```

```
plt.show()
```

[19]:
```
plt.plot(i_array, label="actual")
plt.plot(i_simulated, label="predicted")
plt.title('I Predicted vs Actual')
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
plt.show()
```

[20]:
```
plt.plot(r_array, label="actual")
plt.plot(r_simulated, label="predicted")
plt.title('R Predicted vs Actual')
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
plt.show()
```

[21]:
```
plt.plot(d_array, label="actual")
plt.plot(d_simulated, label="predicted")
plt.title('D Predicted vs Actual')
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
plt.show()
```

[12]:
```
plt.plot(mse_values)
plt.show()
```

[6]:
```
plt.plot(betas, label="beta")
plt.plot(gammas, label="gamma")
plt.plot(deltas, label="delta")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
plt.show()
```

[130]:
```
plt.plot(deltas, "r-", label="delta")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
plt.show()
```

[131]:
```
plt.plot(gammas, "y-", label="gamma")
plt.legend()
plt.xlabel('t days since 1/1/2020')
plt.figure(figsize=(7, 7))
```

```
plt.show()
```

[132]:
```
for t in range(len(gammas)):
    converge_criteria = -gammas[t] - deltas[t]
    if converge_criteria >= 0:
        print(unique_dates[t])
```

```
2020/01/03
2020/01/05
2020/01/06
2020/01/08
```

[ ]:
```

```

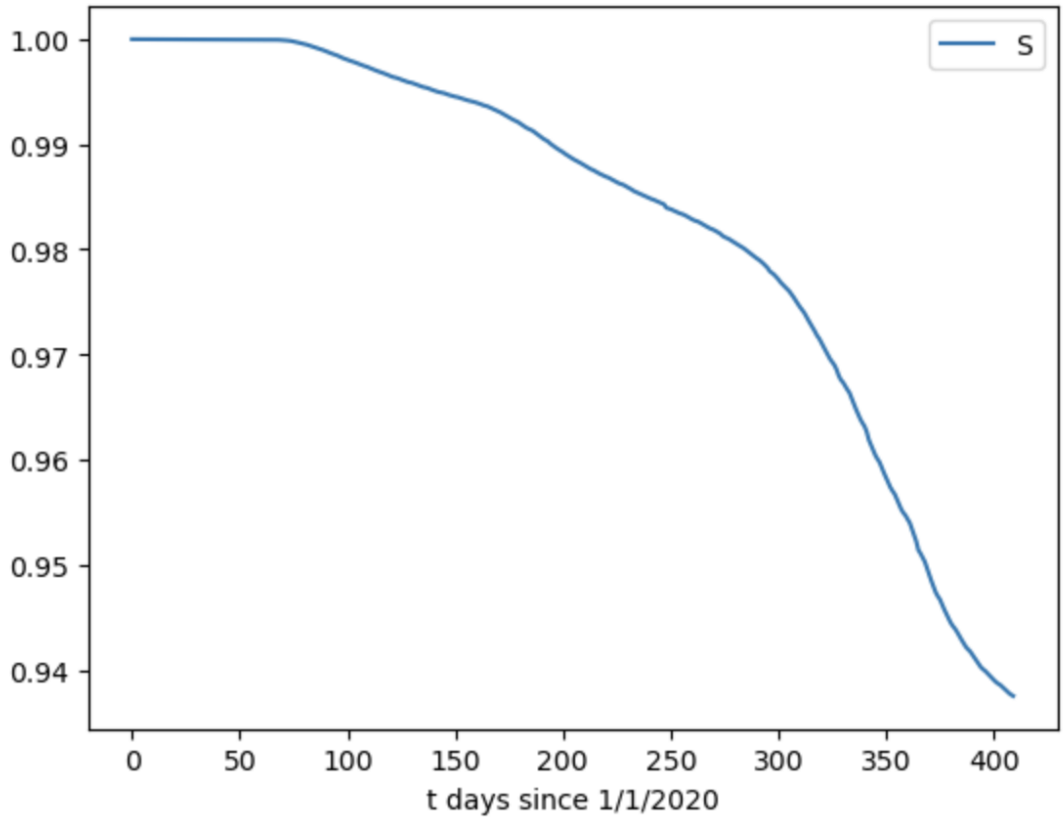

Fig. 4: Actual S Values

References

[1] D. Khullar, "Inside india's covid-19 surge." [Online]. Available: https://www.newyorker.com/science/medical-dispatch/inside-indias-covid-19-surge

[2] A. H. Amiri Mehra, M. Shafieirad, Z. Abbasi, and I. Zamani, "Parameter Estimation and Prediction of COVID-19 Epidemic Turning Point and Ending Time of a Case Study on SIR/SQAIR Epidemic Models," *Computational and Mathematical Methods in Medicine*, vol. 2020, p. 1465923, 2020. [Online]. Available: https://doi.org/10.1155/2020/1465923

[3] A. R. Hota, J. Godbole, P. Bhariya, and P. E. Paré, "A closed-loop framework for inference, prediction and control of sir epidemics on networks," 2020.

[4] L. Bao, W. Deng, H. Gao, C. Xiao, J. Liu, J. Xue, Q. Lv, J. Liu, P. Yu, Y. Xu, F. Qi, Y. Qu, F. Li, Z. Xiang, H. Yu, S. Gong, M. Liu, G. Wang, S. Wang, Z. Song, Y. Liu, W. Zhao, Y. Han, L. Zhao, X. Liu, Q. Wei, and C. Qin, "Lack of reinfection in rhesus macaques infected with sars-cov-2," *bioRxiv*, 2020. [Online]. Available: https://www.biorxiv.org/content/early/2020/05/01/2020.03.13.990226

[5] CDC, "Covid-19 case surveillance public use data," 2021, data retrieved from COVID-19 Case Surveillance Public Use Data, https://catalog.data.gov/tr/dataset/covid-19-case-surveillance-public-use-data-0354e.

[6] D. K. Bagal, A. Rath, A. Barua, and D. Patnaik, "Estimating the parameters of susceptible-infected-recovered model of covid-19 cases in india during lockdown periods," *Chaos, Solitons Fractals*, vol. 140, p. 110154, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960077920305506

[7] S. B. Bastos and D. O. Cajueiro, "Modeling and forecasting the early evolution of the covid-19 pandemic in brazil," 2020.

[8] N. Slighton, J. M. Rico, E. Kallfelz, J. Qi, and C. G. Brinton, "A network-driven approach to modeling the spread of ebola-type epidemics," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, 2018, pp. 1–6.

[9] J. Wu, W. Li, X. Shi, Z. Chen, B. Jiang, J. Liu, D. Wang, C. Liu, Y. Meng, L. Cui, J. Yu, H. Cao, and L. Li, "Early antiviral treatment contributes to alleviate the severity and improve the prognosis of patients with novel coronavirus disease (covid-19)," *Journal of Internal Medicine*, vol. 288, no. 1, pp. 128–138, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/joim.13063

[10] "U.s. and world population clock." [Online]. Available: https://www.census.gov/popclock/

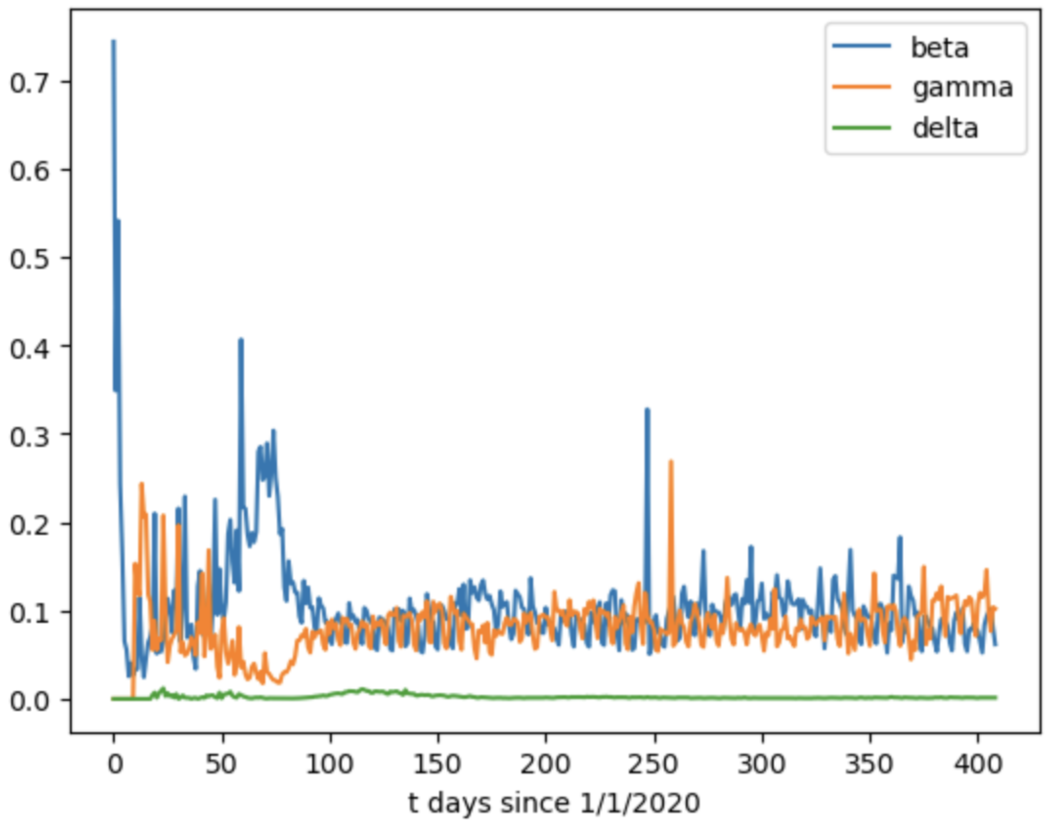

Fig. 5: Actual parameters β, γ, & δ over time

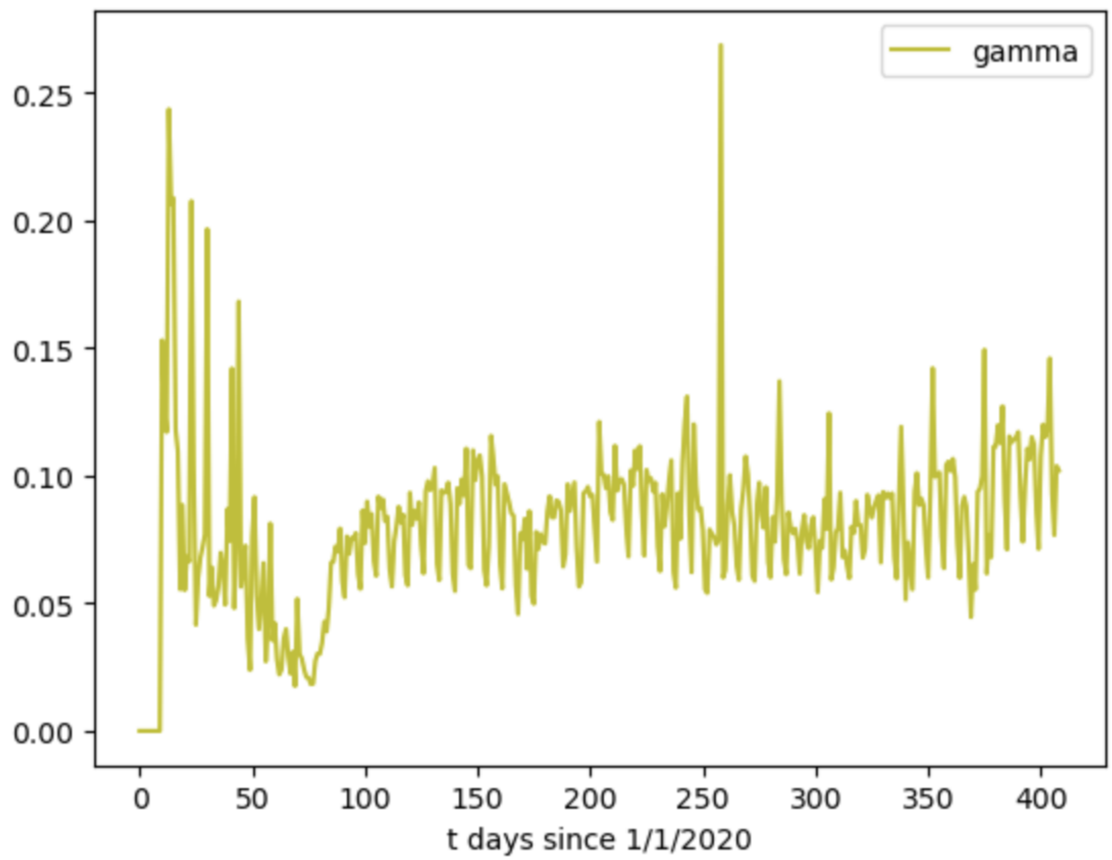

Fig. 6: Actual γ over time

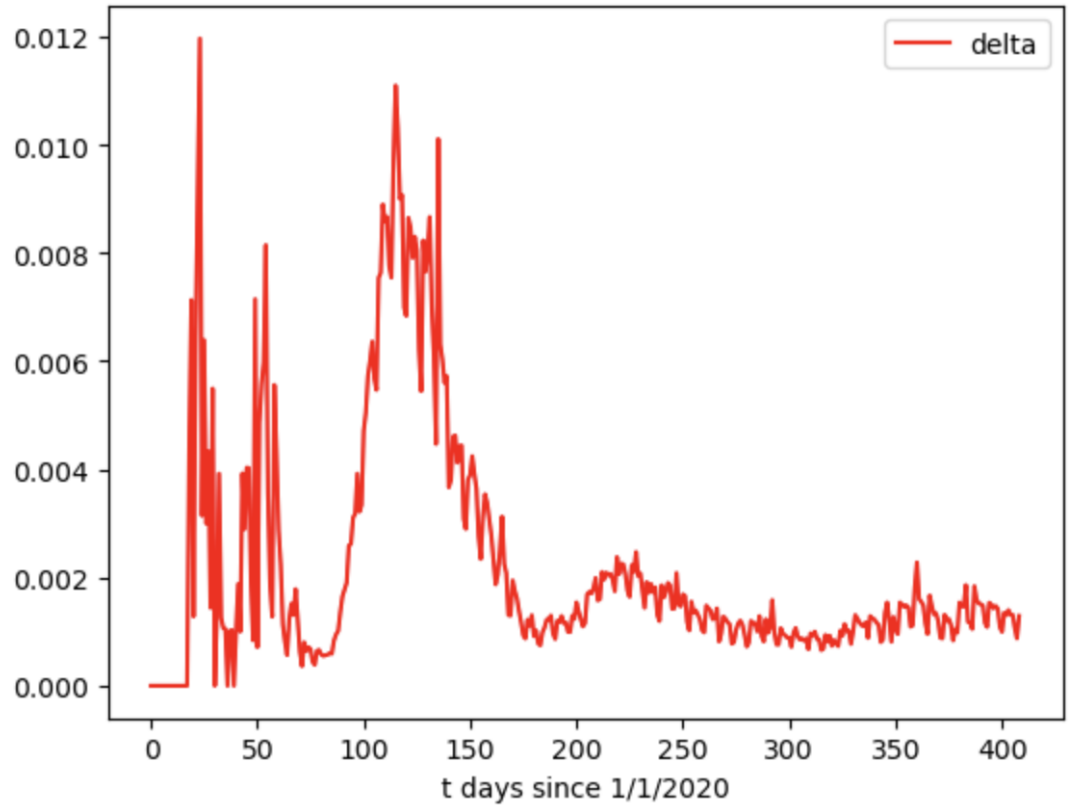

Fig. 7: Actual δ over time

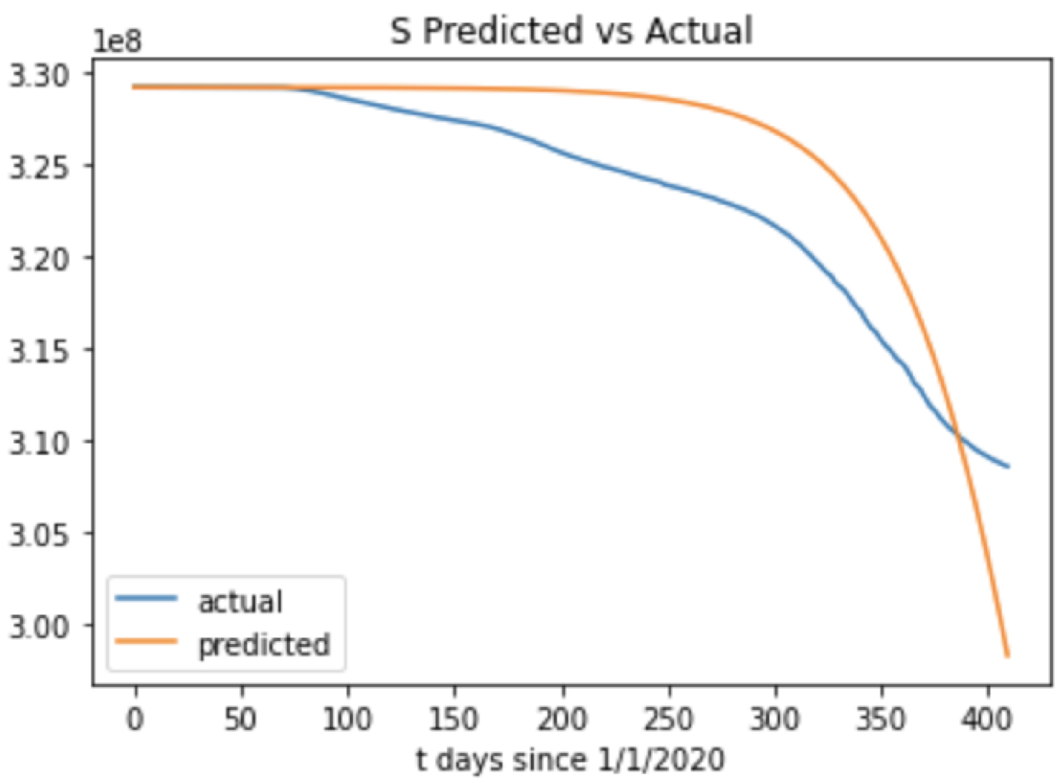

Fig. 8: Simulated S vs Actual S

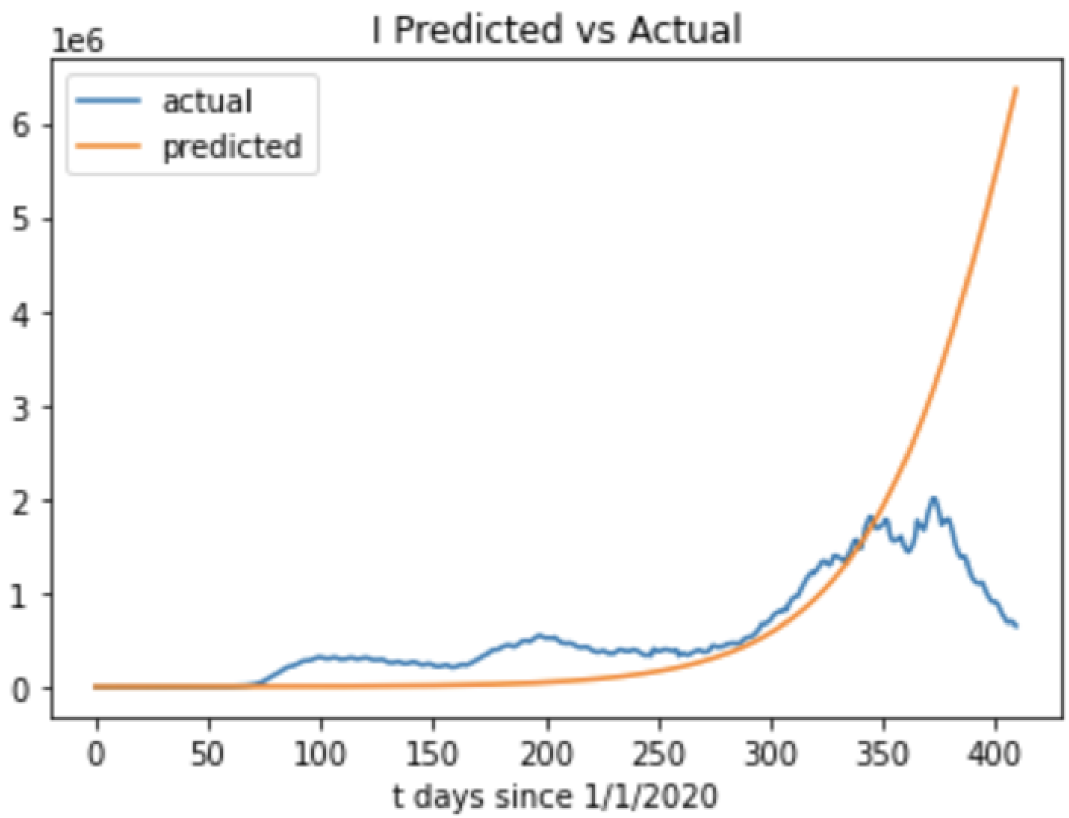

Fig. 9: Simulated I vs Actual I

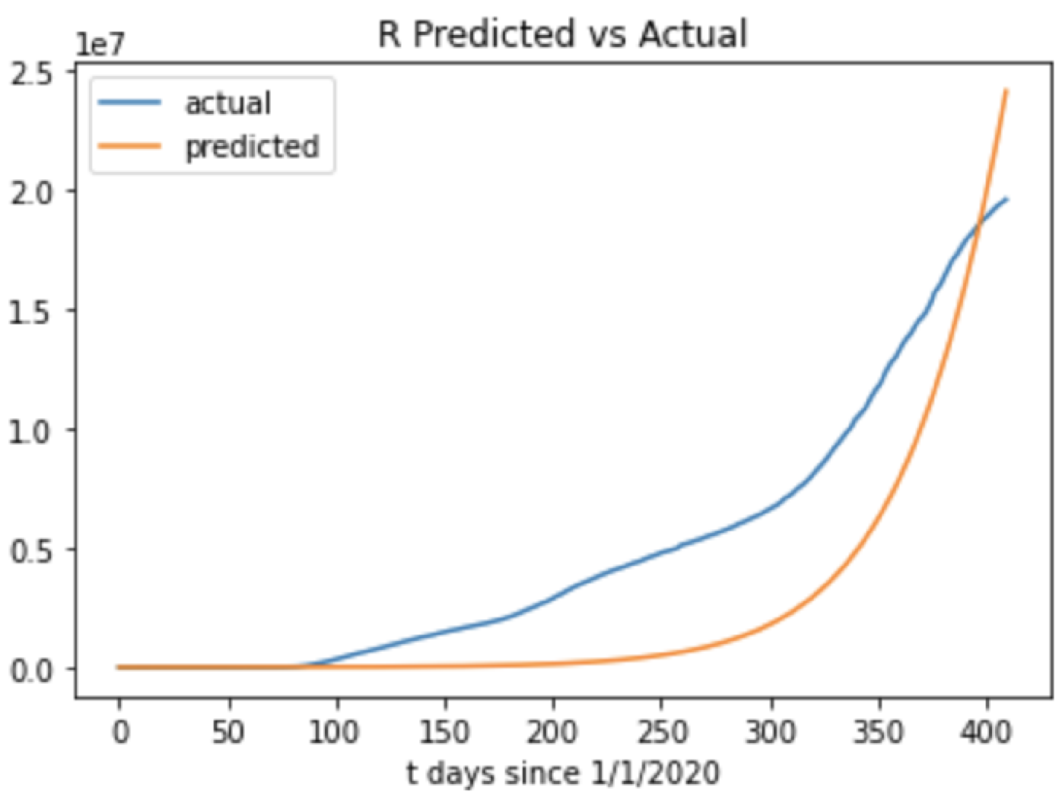

Fig. 10: Simulated R vs Actual R

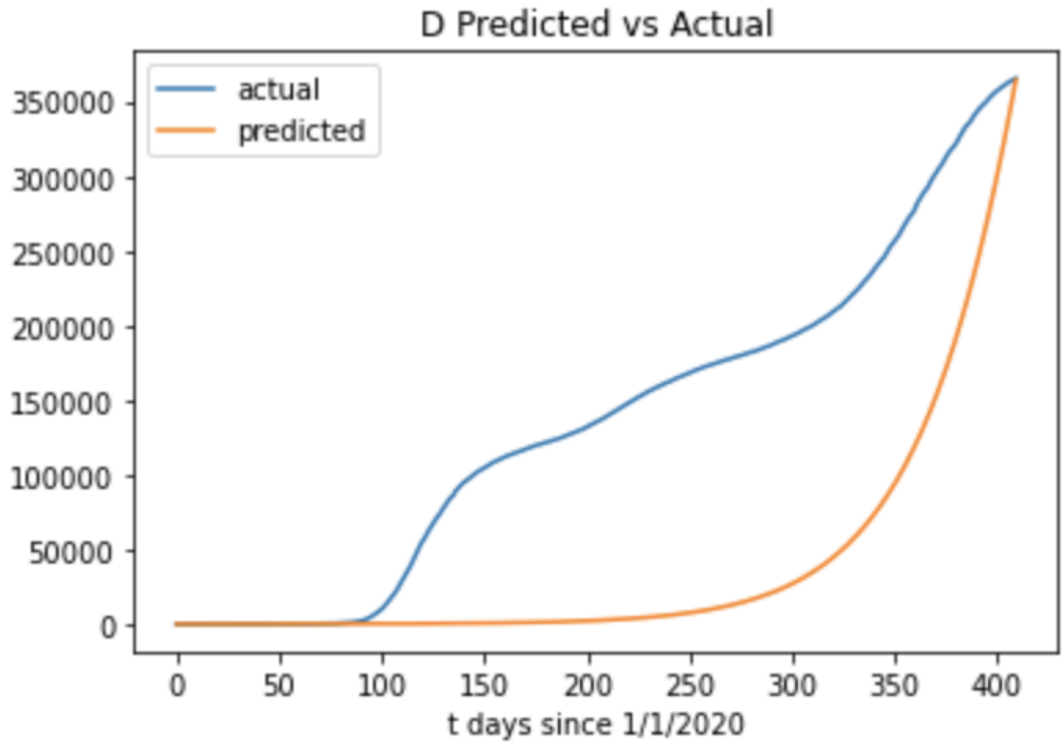

Fig. 11: Simulated D vs Actual D

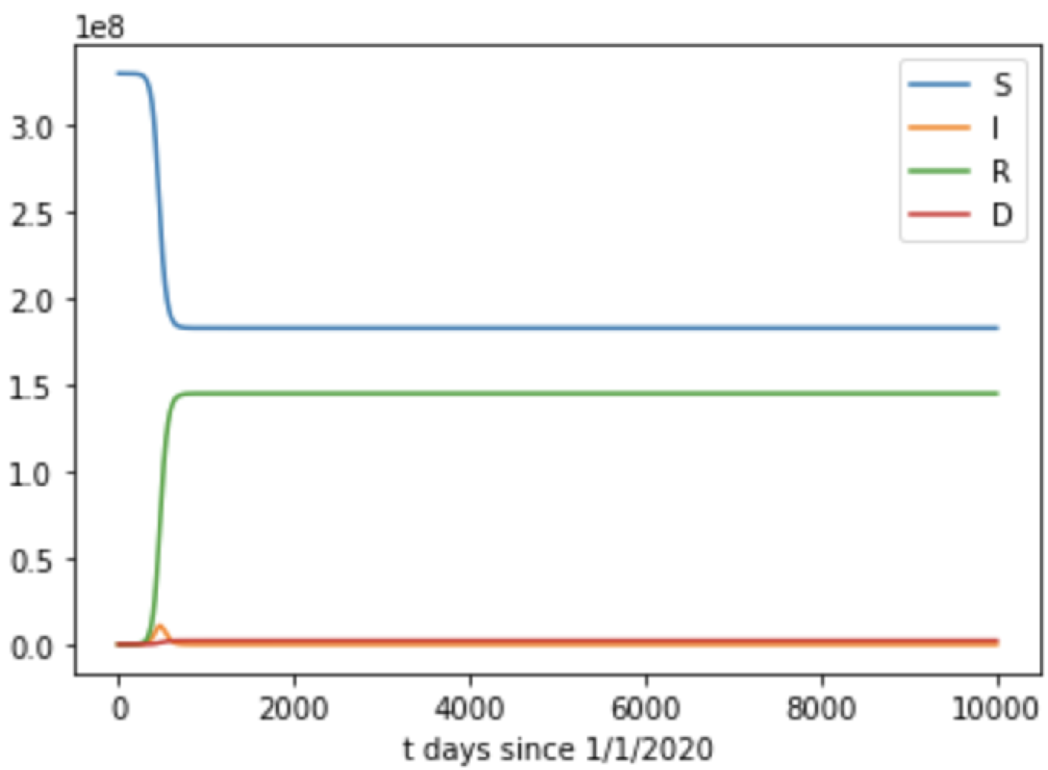

Fig. 12: Simulated SIRD over $t \in [0, 10000]$