VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
HO CHI MINH UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MATH MODELING

Assignment

The SIR Model in COVID-19 Prediction

Advisors:	Nguyen An Khuong	
	Nguyen Tien Thinh	
Candidates:	Tran Hoang Long	– 1852545
	Pham Hoang Hai	– 1852020
	Tran Quoc Anh	– 1852247
	Nguyen Duy Thuan	– 1752526
	Nguyen Hoang Dung	– 1852305

Ho Chi Minh City, April 10, 2021

Contents

1 Introduction **2**

2 Modeling the Pandemic **2**
2.1 SIR Model . . . 2
2.1.1 Assumptions . . . 2
2.1.2 Example for SIR model . . . 3
2.2 Extensions of SIR model . . . 4
2.3 SEIRD Model . . . 5
2.3.1 Assumptions . . . 5
2.3.2 SEIRD for COVID-19 . . . 6

3 Euler's Method for Epidemic Model **8**
3.1 Theory . . . 8
3.2 Applying on SIR Model . . . 9
3.3 Euler's Method Extension . . . 12
3.3.1 Modified Euler Method . . . 12
3.3.2 4th Order Runge–Kutta method (RK4) . . . 13

4 SIR Model's Parameter Sampling **16**
4.1 Background Theory . . . 16
4.1.1 Measure Theory . . . 16
4.1.2 Probability Theory . . . 17
4.1.3 Stochastic Process . . . 18
4.1.4 Markov Chain . . . 18
4.2 Metropolis–Hasting Algorithm . . . 19
4.2.1 Theory . . . 20
4.2.2 Implementation Details . . . 20
4.3 Sampling the Prior Distribution . . . 21

5 Estimating Basic Reproduction Number **23**
5.1 Basic Reproduction Number . . . 23
5.2 Bayes Estimate of R_0 . . . 24
5.3 Monte Carlo Intergration . . . 25
5.3.1 Classical Monte Carlo Integration . . . 25
5.3.2 Monte Carlo Methods Based on Markov Chains . . . 26

6 Affect of Social Distancing in Vietnam **27**
6.1 Motivation . . . 27
6.2 Implementation Details . . . 27
6.2.1 Logarithmic Scale . . . 27
6.2.2 Efficent Jumping Rule . . . 28
6.2.3 Likelihood Function . . . 28
6.3 Sampling Parameters . . . 29
6.3.1 Random-Walk Metropolis Algorithm . . . 29
6.3.2 Code Implementation and Result . . . 30
6.4 Estimating R_0 . . . 34
6.5 Conclusion . . . 35
6.5.1 Changes . . . 35
6.5.2 Discussion . . . 35

7 Conclusion **35**

1 Introduction

The third corona virus outbreak in the past 20 years, the COVID-19 pandemic, has caused unprecedented morbidity, mortality, and economic disruption. Safe, effective, and deployable COVID-19 policies are urgently needed to mitigate the consequences of the pandemic and protect from future outbreaks.

In this assignment report, we dive into the process of modeling a pandemic and predicting its outcome depending on several different parameters. Using our gathered knowledge, we will try and model a country's COVID-19 pandemic during a moderate time period, in which social distancing policy is established. By calculating the disease's basic reproductive number before and after the policy is implemented, we can show how effective quarantine can be for such a disastrous pandemic.

All calculation will be done using the R programming language[13]. R is a programming language and free software environment for statistical computing and graphics. Its base packages include a wide range of functions suitable for common statistical operations.

2 Modeling the Pandemic

Epidemiological models are mathematical models that are used to project the progress of infectious deceases. These models help us to decide which intervention method we want to avoid or bring to trial, or predict the future growth of a potential pandemic. Models use basic assumptions or collected statistics along with mathematics to find parameters for various infectious diseases and use those parameters to calculate the effects of different interventions, like mass vaccination programs.

In this assignment, we will focus on a specific type of epidemiological model, which is the *compartment model*. In compartment models, the population is divided into different categories with label called compartment. And the model would simulate how people may progress from one group to another, giving us a overall look of the disease's process on the population. The origin of these models comes from the work of Kermack and McKendrick in 1927, which gave rise to our most basic of compartment models: The **SIR (Susceptible - Infectious - Recovered)** model. And later comes the extensions of the original SIR.

2.1 SIR Model

In the SIR model, there are three compartments:

- Susceptible: individuals who have no immunity to the infectious agent, so might become infected if exposed
- Infectious: individuals who are currently infected and can transmit the infection to susceptible individuals who they contact
- Removed: individuals who are immune to the infection, and consequently do not affect the transmission dynamics in any way when they contact other individuals

It is traditional to denote the number of individuals in each of these compartments as S, I and R, respectively. The total host population size is $N = S + I + R$.

2.1.1 Assumptions

Having compartmentalized the host population, we now need a set of equations that specify how the sizes of the compartments change over time. To formulate these equations, following assumptions will be made:

1. Every individuals in the population must be consider to have the same characteristics:
   - a is the disease contract rate when facing an infectious individual (transmission probability).
   - b is the number of people that this individual make contact with per unit time (contact rate).

   Then, let $\beta = ab$, this will be the transmission probability times the contact rate. So, each infective can infect $\beta S/N$ other people (knowing that S/N is the susceptible fraction of the population). And the total suceptibles infected by infectives per unit time is $\beta SI/N$.

2. The parameter γ is the mean recover/death rate of an infectious person. Of course, death here only account for ones that are caused by the disease, not natural death.
3. People that has recovered are also assumed to become immune, thus, not going back to the susceptible compartment.
4. Age, sex, social status, and race do not affect the probability of being infected.
5. Lastly, we assume that the total population size N does not change in time. Or more specifically, we assume that: Either, the natural birth/death are equal **or** the rate of infection/recovery is much faster than the rate of natural birth/death so that these natural factors are not considered in our model.

$$N(t) = S(t) + I(t) + R(t) = N = \text{const}$$

The simple visualization of the model can be seen in figure 1.

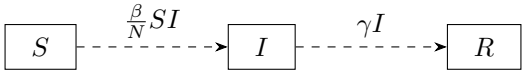

Figure 1: Transition between compartments in the SIR model

The numbers of individuals in each compartment must be integers, of course, but if the host population size N is sufficiently large we can treat S, I and R as continuous variables and express our model for how they change in terms of a system of ordinary differential equations (ODEs)

$$\begin{aligned} \frac{\mathrm{d}S}{\mathrm{d}t} &= -\frac{\beta}{N}IS, \\ \frac{\mathrm{d}I}{\mathrm{d}t} &= \frac{\beta}{N}IS - \gamma I, \\ \frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma I. \end{aligned} \tag{1}$$

When the time is discrete, we have following discrete model:

$$\begin{aligned} S_{n+1} &= S_N - \frac{\beta}{N}S_n I_n, \\ I_{n+1} &= I_n + \frac{\beta}{N}S_n I_n - \gamma I_n, \\ R_{n+1} &= R_n + \gamma I_n. \end{aligned}$$

2.1.2 Example for SIR model

In this example, we decide to choose COVID-19 statistic in Vietnam and the data that we use in this example is collected at 2020-07-13 01:46 GMT from:

1. Guidance for discharge and ending isolation in the context of widespread community transmission of COVID-19. [Link]
2. COVID-19 online data website: worldometers.info

Since the population was 97 365 708 people and the total infected individuals was 372, we can set the value for β/N which is equal to 3.82×10^{-6} (about 4 cases per 1 million residents). We also assume that:

- Population is isolated (constant), i.e., no one leaves or enters;
- The recovery time of an infected individual is exactly 2 weeks and it does not change in time, so recovery rate γ is equal to 0.5;
- Who recovered from COVID-19 will be immune to it in the future.

Substituting above values to system (1) gives us

$$\begin{aligned}\frac{\mathrm{d}S}{\mathrm{d}t} &= -3.82 \times 10^{-6} IS,\\ \frac{\mathrm{d}I}{\mathrm{d}t} &= 3.82 \times 10^{-6} IS - 0.5I,\\ \frac{\mathrm{d}R}{\mathrm{d}t} &= 0.5I.\end{aligned}$$

2.2 Extensions of SIR model

For this assignment, our motivation is the spread of the current pandemic that is causing havoc on all of modern civilization: the COVID-19 pandemic caused by SARS-CoV-2. And for this particular purpose, there's some aspects that our simple SIR model fails to project:

- The incubation (or exposed) period: The time in which a person is infected with the virus but does not show any symptoms. This is specifically important for COVID-19 since people are most infectious when symptoms appear.
- The fatality rate of the pandemic: The recovered compartment in SIR model is for both recovered and deceased people, which does not give us an overall look of the impact that COVID-19 been making.

There are many other extensions of the original SIR model that potentially help us with the downsides of SIR that we mentioned above. They deal with this by introducing more specific compartments to further categorize the population under effects of a disease, each compartment have their own behavior and give us more insight to the problem. This means that for each individual disease and depending on our assumptions and our goals—what we hope to observe from the model; there should be an extension of SIR that suits the need. Some of those extensions are:

1. **SIS:** Many diseases' infection does not confer complete immunity and individuals become susceptible again after recovering from these diseases. Typical examples of this model are the common cold and influenza. Additionally, for these diseases, fatality rate is low and can be neglected.

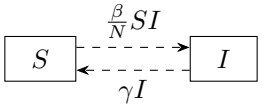

Figure 2: Transition between compartments in the SIS model

2. **MSIR:** For many infections, babies are basically immune for a short period since their birth, due to protection from maternal antibodies passed down from their mother (passive immunity). This is shown by introducing a M class (Maternally Derived Immunity) at the start of the SIR model.

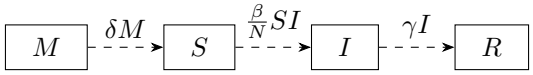

Figure 3: Transition between compartments in the MSIR model

3. **SIRD:** This model separated our previously "composite" class of Removed (include both immune and dead population) into more specific **R**ecovered (only people that got infected and recovered, becoming immune after) and **D**eceased (people that pass away from infection). This model helps to also measure the impact of a disease since it shows the death rate of infection.

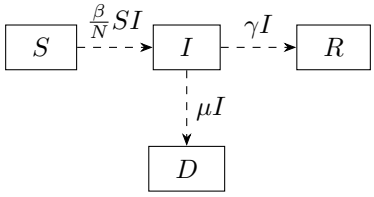

Figure 4: Transition between compartments in the SIRD model

4. **SEIR:** In some diseases, there is an incubation period where the host does not show any symptoms of infection. This model help model this period by introducing an **E**xposed class before the **I**nfected class. The people in the exposed class can be non-infectious or lesser infectious then people that's infectious, depending on our choice of constructing the model.

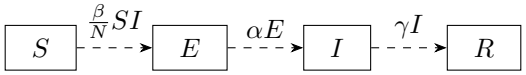

Figure 5: Transition between compartments in the SEIR model

In the list of popular extensions of SIR we mentioned here, the **SIRD** and **SEIR** models both elaborate more on our subject of COVID-19 (death and exposed period respectively). But we are settling on a more specific model, a combination of both SIRD and SEIR.

2.3 SEIRD Model

For modeling the spread and affect of COVID-19 on society, we decided on using the SEIRD model, a combination of both SIRD and SEIR to be able to model both the exposed period of a person and the death rate that this pandemic currently have. **SEIRD** (Susceptible - Exposed - Infectious - Recovered - Deceased) model is an extension of the compartment model SIR that we already introduce. This model consist of 5 compartments:

1. **S (Susceptible):** The number of susceptible individuals. When this group come in contact with infectious group, they can become infected, carrying the disease and move to the exposed group.
2. **E (Exposed):** The number of people that are infected but not infectious. They carry the disease but do not show any symptoms and do not infect susceptible people. After an incubation period, these individuals start to show symptoms and move to the infectious group.
3. **I (Infectious):** The number of infectious individuals. These are individuals who have been infected and are capable of infecting susceptible individuals. Here, two outcome could happen, either an individual here recover or, pass away from the disease.
4. **R (Recovered):** The number of people that were infectious then recovered from the disease and develop immunity.
5. **D (Deceased):** The people that pass away due to the affect of the disease

2.3.1 Assumptions

Before formulating the model for SEIRD, there are assumptions that we have to make. Most of these assumptions are already presented and well explained in the assumption for SIR model in section 2.1.1.

1. People must have the same transmission probability and same people contact rate. This help us formulate the transition rate between susceptible group and exposed group.

2. The incubation period is a random variable with parameter α (the mean incubation period is $1/\alpha$). So α can be understood as the incubation rate.
3. γ is the mean recovery rate of an infectious group.
4. People that recovered are also assumed to be immune to the disease.
5. μ is the mean death rate of the infectious group.
6. The total population N does not change in time (we will ignore vital dynamics).

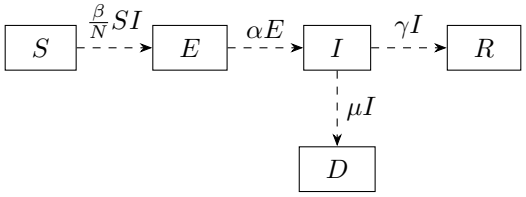

Figure 6: Transition between compartments in the SEIRD model

Now we can formulate our model simply as a system of ODEs

$$\begin{aligned}
\frac{\mathrm{d}S}{\mathrm{d}t} &= -\frac{\beta}{N}IS, \\
\frac{\mathrm{d}E}{\mathrm{d}t} &= \frac{\beta}{N}IS - \alpha E, \\
\frac{\mathrm{d}I}{\mathrm{d}t} &= \alpha E - \gamma I - \mu I, \\
\frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma I, \\
\frac{\mathrm{d}D}{\mathrm{d}t} &= \mu I.
\end{aligned} \tag{2}$$

Similar to the continuous model, when time is discrete interval instead of continuous, the model is as follows:

$$\begin{aligned}
S_{n+1} &= S_n - \frac{\beta}{N}I_nS_n, \\
E_{n+1} &= E_n + \frac{\beta}{N}I_nS_n - \alpha E_n, \\
I_{n+1} &= I_n + \alpha E_n - \gamma I_n - \mu I_n, \\
R_{n+1} &= R_n + \gamma I_n, \\
D_{n+1} &= D_n + \mu I_n.
\end{aligned}$$

2.3.2 SEIRD for COVID-19

In this section, we will form an example SEIRD model for the current pandemic of COVID-19.

Why SEIRD for COVID-19?

1. Frequently talked about as "The defining global health crisis of our time and the greatest challenge we have faced since World War Two", the affects of this disease upon our social, financial and physical lives is can be reflected upon its fatality. So we decided to include the **Deceased** group in our model, as a critical indicator of the pandemic's impact.

2. Referenced from COVID-19 situation report by WHO, the disease has an average incubation period of 5–6 days. This is the "pre-symptomatic" period where no symptoms of the disease is shown. Some people can be infectious here, but the amount case reported and studied for this is quite small. Thus, we can assume that people in this phase of COVID-19 is not infectious. We reflect this trait of the virus in our model as the **Exposed** group.

3. Another case of disease transmission is asymptomatic transmission where infected person does not show any symptoms during the whole cycle of carrying the virus and recover from it. Though, there has been no documented asymptomatic transmission so this is neglectable.

From the above reasons, we have decided to expand SIR model into SEIRD model and use it on COVID-19 as an example. We also choose USA's statistics for specific analysis in this area.

Model's assumptions and parameters:

Note that the statistics we use to make these assumptions are collected at 2020-07-11 22:37 GMT from:

1. WHO's coronavirus document: Situation Report–73
2. COVID-19 online data website: worldometers.info

We assume that

- Population is isolated (constant).
- Recovered people are immune to COVID-19.
- All the parameters does not change in time (β, α, γ, μ).
- A susceptible individual becomes an infected individual with the rate β equals 1 012 689 cases per 100 000 000 residents. It is calculated by the formula:

$$\text{Rate of infection} = k \times \frac{\text{Number of infections}}{\text{Population at risk}}.$$

  Constant k is usually an assigned value of 100, 1000, 10 000 or 100 000, which represents a standard population and time period for interpretation of the rate.

- The virus's incubation period α is 5.5 days, thus a persons incubation rate $\alpha = 1/5.5 = 2/11$.
- The death rate μ is around 4% (computing by the number of reported deaths divided by the reported cases).
- The recovery rate γ is around 96% since the death rate is around 4%.

COVID-19's Model

From those assumptions and parameters that we've established above, the equations for the final SEIRD model is

$$\begin{aligned}
\frac{\mathrm{d}S}{\mathrm{d}t} &= -0.01IS \\
\frac{\mathrm{d}E}{\mathrm{d}t} &= 0.01IS - 0.182E \\
\frac{\mathrm{d}I}{\mathrm{d}t} &= 0.182E - 0.96I - 0.04I \\
\frac{\mathrm{d}R}{\mathrm{d}t} &= 0.96I \\
\frac{\mathrm{d}D}{\mathrm{d}t} &= 0.04I
\end{aligned}$$

3 Euler's Method for Epidemic Model

3.1 Theory

The Euler's method is a first-order method that is used to solve ordinary differential equations (ODEs) with a given initial value. The method was first introduced by Leonhard Euler in Institutionum Calculi Integralis published in between 1768 and 1770.

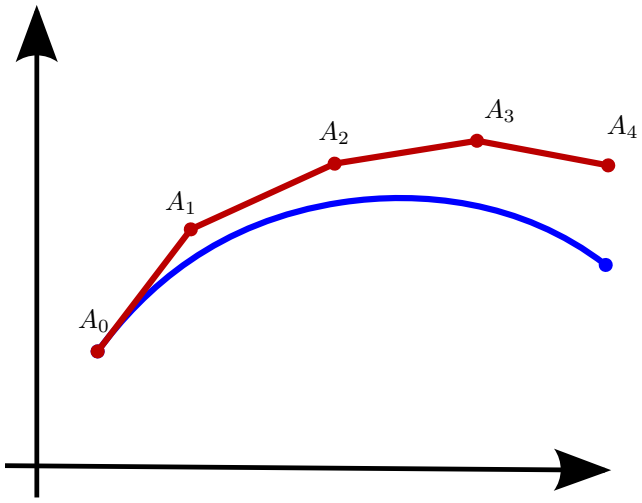

Figure 7: Illustration of Euler method

Euler's method can be understood geometrically by considering the problem of calculating the shape of an unknown curve which starts at a given point and satisfies a given differential equation. Here, a differential equation can be thought of as a formula by which the slope of the tangent line to the curve can be computed at any point on the curve, once the position of that point has been calculated.

The idea is that while the curve is initially unknown, its starting point, which we denote by A_0, is known (see figure 7). Then, from the differential equation, the slope to the curve at A_0 can be computed, and so, the tangent line.

Take a small step along that tangent line up to a point A_1. Along this small step, the slope does not change too much, so A_1 will be close to the curve. If we pretend that A_1 is still on the curve, the same reasoning as for the point A_0 above can be used. After several steps, a polygonal curve $A_0, A_1, A_2, A_3, \ldots$ is computed. In general, this curve does not diverge too far from the original unknown curve, and the error between the two curves can be made small if the step size is small enough and the interval of computation is finite.

Formally, consider the initial value problem

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \tag{3}$$

where we need to find a continuously differentiate function $y : \mathbb{R} \to \mathbb{R}$ that is a solution of the ODE $y'(t) = f(t, y(t))$ and satisfy the initial condition $y(t_0) = y_0$. The Euler method for the numerical solution of the initial value problem (3) constructs approximations y_n to the exact solution $y(t_n)$ at the equidistant grid points

$$t_n = t_0 + nh, \quad n = 1, 2, \ldots,$$

with step size h by

$$y_{n+1} = y_n + hf(t_n, y_n). \tag{4}$$

In general, we can use Euler's method to solve a system of ODEs

$$\begin{aligned} y_1'(t) &= f_1(t, y_1(t), \ldots, y_N(t)), \\ &\vdots \\ y_N'(t) &= f_N(t, y_1(t), \ldots, y_N(t)), \end{aligned} \tag{5}$$

by introducing another continuously differentiable function $Y : \mathbb{R} \to \mathbb{R}^N$ defined as

$$Y(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_N(t) \end{bmatrix}, \text{ for } t \in \mathbb{R}, \tag{6}$$

and a vector-valued function $F : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}^N$ defined as

$$F\left(t, \begin{bmatrix} y_1(t) \\ \vdots \\ y_N(t) \end{bmatrix}\right) = \begin{bmatrix} f_1(t, y_1(t), \ldots, y_N(t)) \\ \vdots \\ f_N(t, y_1(t), \ldots, y_N(t)) \end{bmatrix}, \text{ for } t \in \mathbb{R}, \tag{7}$$

our system of ODEs (5) will become

$$Y'(t) = F(t, Y(t)).$$

Substitute scalar function y by vector-valued fucntion Y and derivative function f by vector F of derivative functions in equation (4), we obtained the generalized version of Euler's method:

$$Y_{n+1} = Y_n + hF(t_n, Y_n).$$

```r
# Arguments:
#   y0: the initial values for the ODE system
#   times: times at which explicit estimates for y are desired
#   func: a function that computes the derivatives in the ODE system at time t
#   parms: vector or list of parameters used in `func`.
euler <- function(y0, times, func, parms=NULL) {
    h <- times[2] - times[1]
    tbl <- matrix(nrow=length(times), ncol=1+length(y0))
    colnames(tbl) <- c("time", names(y0))
    tbl[, 1] <- times
    tbl[1, -1] <- y0
    for (i in seq_len(nrow(tbl)-1)) {
        ti <- tbl[i, 1]
        yi <- tbl[i, -1]
        tbl[i+1, -1] <- yi + func(ti, yi, parms)
    }
    return(tbl)
}
```

Listing 1: R implementation of Euler's method

3.2 Applying on SIR Model

Consider S, I, R in the SIR model are three functions with independent variable t, we can represent these functions as a vector-valued function

$$Y = \begin{bmatrix} S \\ I \\ R \end{bmatrix} \tag{8}$$

and from equation (1), we have the derivative function

$$F\left(t, \begin{bmatrix} S \\ I \\ R \end{bmatrix}\right) = \begin{bmatrix} -\frac{\beta}{N} IS \\ \frac{\beta}{N} IS - \gamma I \\ \gamma I \end{bmatrix}. \tag{9}$$

The following R function can compute the value of F given the value of Y and parameters β, γ, N:

```
SIR.model <- function(t, y, parms) with(as.list(c(y, parms)), {
    dS.dt <- -beta / N * S * I
    dI.dt <- beta / N * I * S - gamma * I
    dR.dt <- gamma * I
    c(dS.dt, dI.dt, dR.dt)
})
```

Input for the program of finding the approximate solution to the SIR system using Euler's method are

- The initial value of the SIR model: the number of people in each compartment at initial time t_0

$$Y_0 = \begin{bmatrix} S_0 \\ I_0 \\ R_0 \end{bmatrix}$$

- A partition of time interval $[t_0, t]$ with step size h
- The parameters of SIR model:
  - The contact rate β
  - The recovery rate γ
  - The total number of the population N

Example

Here is an example with $S_0 = 995$, $I_0 = 5$, $R_0 = 0$, $t_0 = 0$, $t = 22$, $h = 0.5$, $\beta = 1.407$, $\gamma = 0.6$, $N = 1000$.

```
euler(
    y0 = c(S=995, I=5, R=0),
    times = seq(0, 22, by=.5),
    func = SIR.model,
    parms = list(beta=1.407, gamma=.6, N=pop)
)
```

	time	S	I	R
[1,]	0.0	995.00000	5.000000e+00	0.000000
[2,]	0.5	988.00018	8.999825e+00	3.000000
[3,]	1.0	975.48937	1.611073e+01	8.399895
[4,]	1.5	953.37717	2.855649e+01	18.066335
[5,]	2.0	915.07145	4.972832e+01	35.200230
[6,]	2.5	851.04595	8.391682e+01	65.037225
[7,]	3.0	750.56213	1.340505e+02	115.387317
[8,]	3.5	608.99927	1.951831e+02	195.817646
[9,]	4.0	441.75431	2.453182e+02	312.927496
[10,]	4.5	289.27720	2.506044e+02	460.118413
[11,]	5.0	187.27795	2.022410e+02	610.481046
[12,]	5.5	133.98743	1.341869e+02	731.825649
[13,]	6.0	108.69047	7.897173e+01	812.337801
[14,]	6.5	96.61352	4.366564e+01	859.720839
[15,]	7.0	90.67782	2.340195e+01	885.920223
[16,]	7.5	87.69211	1.234649e+01	899.961396
[17,]	8.0	86.16877	6.461940e+00	907.369289
[18,]	8.5	85.38533	3.368218e+00	911.246454
[19,]	9.0	84.98068	1.751936e+00	913.267385
[20,]	9.5	84.77120	9.102493e-01	914.318546
[21,]	10.0	84.66264	4.726680e-01	914.864696
[22,]	10.5	84.60633	2.453715e-01	915.148296
[23,]	11.0	84.57712	1.273579e-01	915.295519
[24,]	11.5	84.56197	6.609876e-02	915.371934
[25,]	12.0	84.55410	3.430385e-02	915.411593
[26,]	12.5	84.55002	1.780259e-02	915.432176
[27,]	13.0	84.54790	9.238864e-03	915.442857
[28,]	13.5	84.54680	4.794591e-03	915.448400
[29,]	14.0	84.54623	2.488188e-03	915.451277
[30,]	14.5	84.54594	1.291262e-03	915.452770
[31,]	15.0	84.54579	6.701081e-04	915.453545
[32,]	15.5	84.54571	3.477566e-04	915.453947
[33,]	16.0	84.54566	1.804703e-04	915.454156
[34,]	16.5	84.54564	9.365609e-05	915.454264
[35,]	17.0	84.54563	4.860337e-05	915.454320
[36,]	17.5	84.54563	2.522299e-05	915.454349
[37,]	18.0	84.54562	1.308962e-05	915.454364
[38,]	18.5	84.54562	6.792931e-06	915.454372
[39,]	19.0	84.54562	3.525230e-06	915.454376
[40,]	19.5	84.54562	1.829438e-06	915.454378
[41,]	20.0	84.54562	9.493974e-07	915.454380
[42,]	20.5	84.54562	4.926952e-07	915.454380
[43,]	21.0	84.54562	2.556870e-07	915.454380
[44,]	21.5	84.54562	1.326902e-07	915.454381
[45,]	22.0	84.54562	6.886033e-08	915.454381

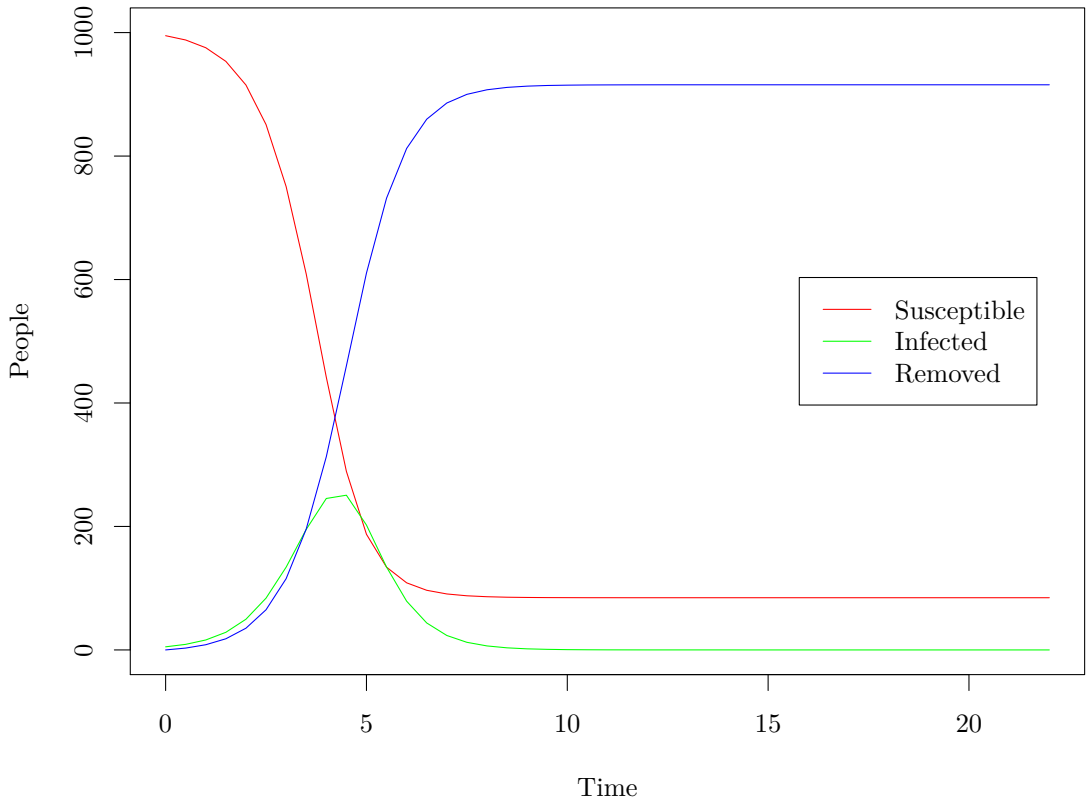

Figure 8: Line plot for the solution of this example

3.3 Euler's Method Extension

3.3.1 Modified Euler Method

In the Euler's method, the tangent is drawn at a point and slope is calculated for a given step size. Thus this method works best with linear functions, but for other cases, there remains a truncation error. To solve this problem, the modified Euler method is introduced. In this method, instead of a point, the arithmetic average of the slope over an interval $[t_n, t_{n+1}]$ is used. Thus in the Modified Euler method for each step, the slope k_1 at the point (t_n, y_n) is calculated first, then the predicted value of y_{n+1} is calculated using Euler's method, $y_{n+1} \approx y_n + hk_1$, Next, the slope k_2 at the point $(t_{n+1}, y_n + hk_1)$ is calculated. Finally, the arithmetic average of these slopes are added to y_n to calculate the corrected value of y_{n+1}.

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_{n+1}, y_n + hk_1), \\ y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2), \\ t_{n+1} &= t_n + h. \end{aligned}$$

The Modified Euler's method is a second-order method, meaning that the local truncation error is on the order $O(h^2)$.

3.3.2 4th Order Runge–Kutta method (RK4)

In numerical analysis, the Runge–Kutta methods are a family of implicit and explicit iterative methods, which include the well-known routine called the Euler Method, used in temporal discretization for the approximate solutions of ordinary differential equations. These methods were developed around 1900 by the German mathematicians Carl Runge and Wilhelm Kutta.

Consider the initial value problem (3), the iterative formula of RK4 method is

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \\ k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2), \\ k_4 &= f(t_n + h, y_n + hk_3), \\ t_{n+1} &= t_n + h. \end{aligned}$$

Here y_{n+1} is the RK4 approximation of $y(t_{n+1})$, and the next value y_{n+1} is determined by the present value y_n plus the weighted average of four increments, where each increment is the product of step size h and an estimated slope specified by function f on the right-hand side of the differential equation.

- k_1 is the slope at the beginning of the interval, using y (Euler's method);
- k_2 is the slope at the midpoint of the interval, using y and k_1;
- k_3 is again the slope at the midpoint, but now using y and k_2;
- k_4 is the slope at the end of the interval, using y and k_3.

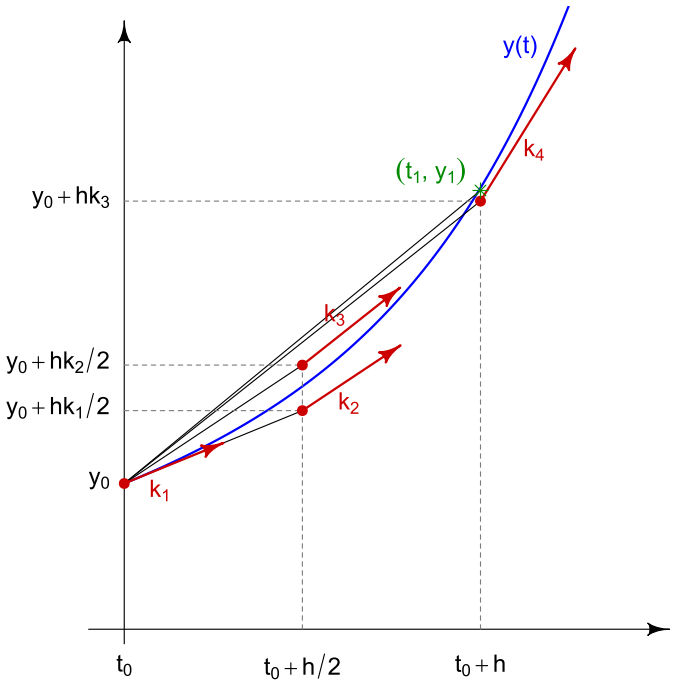

Figure 9: Slopes used by the Runge–Kutta method

The RK4 method is a fourth-order method, meaning that the local truncation error is on the order of $O(h^5)$, while the global truncation error is on the order of $O(h^4)$.

The Runge–Kutta method can also be generalized to solve a system of ODEs (5) by introducing vector-valued function Y and F as defined by equation (6) and (7) respectively.

```
rk4 <- function(y0, times, func, parms=NULL) {
    h <- times[2] - times[1]
    tbl <- matrix(nrow=length(times), ncol=1+length(y0))
    colnames(tbl) <- c("time", names(y0))
    tbl[, 1] <- times
    tbl[1, -1] <- y0
    for (i in seq_len(nrow(tbl)-1)) {
        ti <- tbl[i, 1]
        yi <- tbl[i, -1]
        k1 <- func(ti, yi, parms)
        k2 <- func(ti + h/2, yi + h/2 * k1, parms)
        k3 <- func(ti + h/2, yi + h/2 * k2, parms)
        k4 <- func(ti + h, yi + h * k3, parms)
        tbl[i+1, -1] <- yi + h/6 * (k1 + 2 * k2 + 2 * k3 + k4)
    }
    return(tbl)
}
```

Listing 2: Implementation of RK4 method in the R programming language

Applying on SEIRD Model

Consider S, E, I, R, D in the SIR model are five functions with independent variable t, we have

$$Y = \begin{bmatrix} S \\ E \\ I \\ R \\ D \end{bmatrix},$$

and from equation (2), we have the derivative function

$$F\left(t, \begin{bmatrix} S \\ E \\ I \\ R \\ D \end{bmatrix}\right) = \begin{bmatrix} -\frac{\beta}{N} IS \\ \frac{\beta}{N} IS - \alpha E \\ \alpha E - \gamma I - \mu I \\ \gamma I \\ \mu I \end{bmatrix}.$$

The following R function can compute the value of F given the value of Y and parameters α, β, γ, μ, N.

```
SEIRD.model <- function(t, y, parms) with(as.list(c(y, parms)), {
    dS.dt <- -beta / N * S * I
    dE.dt <- beta / N * S * I - alpha * E
    dI.dt <- alpha * E - gamma * I
    dR.dt <- gamma * I
    dD.dt <- mu * I
    c(dS.dt, dE.dt, dI.dt, dR.dt, dD.dt)
})
```

Input for the program of finding the approximate solution to the SEIRD system using Euler's method is the same as SIR model above and consists of further variables:

- The mean incubation rate α
- The mean death rate μ
- The initial number of exposed people E_0
- The initial number of death people D_0

Here is an example with $S_0 = 999$, $I_0 = 1$, $E_0 = R_0 = D_0 = 0$, $t_0 = 0$, $t = 80$, $h = 0.1$, $\beta = 1.407$, $\gamma = 0.6$, $N = 1000, \alpha = 0.2, \mu = 0.4$.

```
sol <- rk4(
    y0 = c(S=999, E=0, I=1, R=0, D=0),
    times = seq(0, 80, by=.1),
    func = SEIRD.model,
    parms = list(beta=1.407, gamma=.6, mu=.4, alpha=.2, N=1000)
)
```

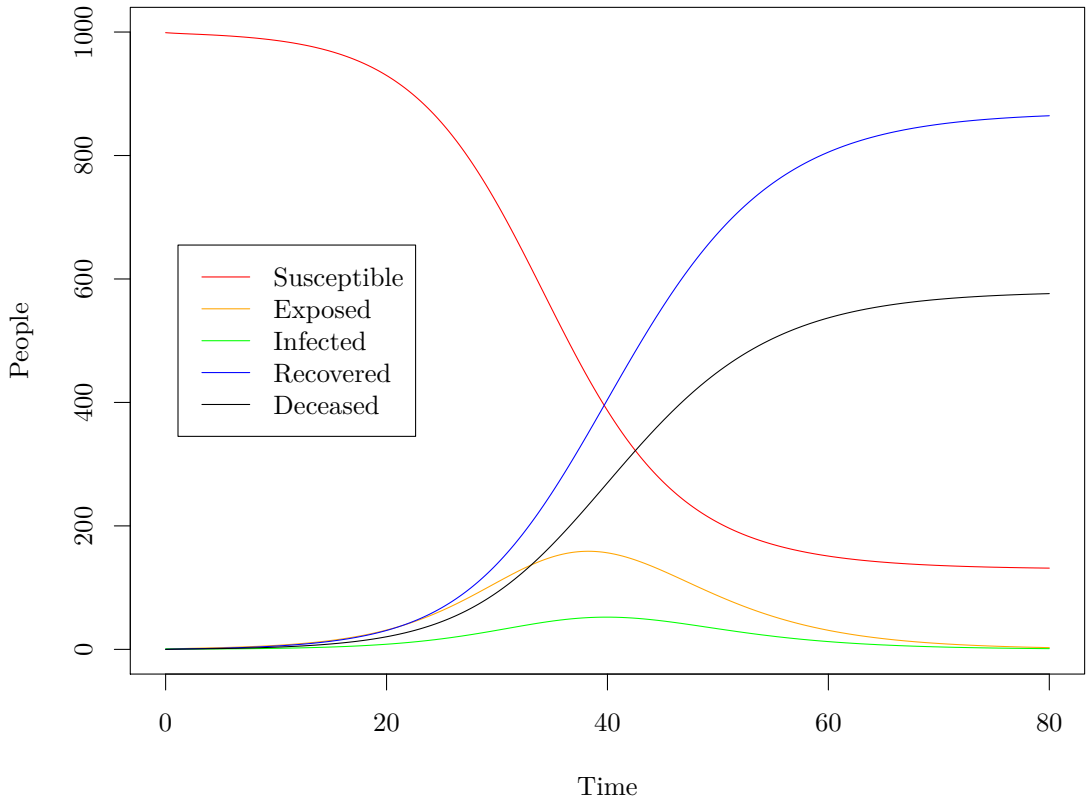

Figure 10: Plot for SEIRD example with Runge–Kutte method

Note: We decided on using the 4th order Runge–Kutta method (RK4) for further calculations (later in the report) but not standard Euler's or modified Euler's method for following reasons:

- The Euler method is only first order convergent, i.e., the error of the computed solution is $O(h)$, where h is the time step. This is unacceptably poor, and requires a too small step size to achieve some serious accuracy. For example, if you need one hundred times more accuracy, you will have to take one hundred times as many steps to achieve it.
- Modified Euler's method, by contrast, is second order convergent, with an error $O(h^2)$. The improvement is dramatic, and one can almost obtain reasonably accurate results with Modified Euler's method. Using the same example as above, if you need one hundred times more accuracy, you will only have to take ten times as many steps to achieve it, so efficiency is dramatically improved.
- 4th order Runge–Kutta method is 4th order convergent, with an error $O(h^4)$, and can be used for accurate computations. Reduce the step size by a factor of ten, and you get 10,000 times improved accuracy.

4 SIR Model's Parameter Sampling

4.1 Background Theory

4.1.1 Measure Theory

The development of probability theory require measure theory, so we will present some basic concepts in measure theory.

Definition 4.1 (σ-algebra)**.** Suppose X is a set and $\mathcal{S}$ is a set of subsets of X. Then $\mathcal{S}$ is called a σ-algebra on X if the following three conditions are satisfied:

- $\varnothing \in \mathcal{S}$;
- if $E \in \mathcal{S}$, then $X/E \in \mathcal{S}$;
- if $E_1, E_2, \ldots$ is a sequence of elements of $\mathcal{S}$, then $\bigcup_{k=1}^{\infty} E_k \in \mathcal{S}$.

Example 4.2. Some typical σ-algebras are:

- Suppose X is a set. Then clearly $\{\varnothing, X\}$ is a σ-algebra on X.
- Suppose X is a set. Then clearly the set of all subsets of X is a σ-algebra on X.

Now we come to the crucial definition of this section. A measure assign a non-negative value to many subsets of X, which can be understood as generalized version of length, area or volume.

Definition 4.3 (measure)**.** Suppose X is a set and $\mathcal{S}$ is a σ-algebra on X. A measure on $(X, \mathcal{S})$ is a function $\mu : \mathcal{S} \to [0, \infty)$ such that $\mu(\varnothing) = 0$ and

$$\mu\left(\bigcup_{k=1}^{\infty} E_k\right) = \sum_{k=1}^{\infty} \mu(E_k)$$

for every disjoint sequence $E_1, E_2, \ldots$ of sets in $\mathcal{S}$.

Example 4.4 (measure)**.** If X is a set, then *counting measure* is the measure μ defined on the σ-algebra of all subsets of X by setting $\mu(E) = n$ if E is a finite set containing exactly n elements and $\mu(E) = \infty$ if E is not a finite set.

Definition 4.5 (measure space)**.** A measure space is an ordered triple $(X, \mathcal{S}, \mu)$, where X is a set, $\mathcal{S}$ is a σ-algebra on X, and μ is a measure on $(X, \mathcal{S})$.

4.1.2 Probability Theory

Probability space is a measure space characterized by a three-tuple (also called probability triple) of $(\Omega, \mathcal{F}, P)$ that defines a model for a random experiment, such as the experiment of rolling a dice.

A probability space consists of three elements:

1. The sample space Ω is the set of all possible outcomes. Each outcome ω is the result of a single performance of the experiment.
2. The event space $\mathcal{F}$ is a σ-algebra on Ω, a collection of events that we are interested in observing. An event is a set of outcomes in the sample space.
3. A probability measure $P : \mathcal{F} \to [0, 1]$ that assign each event in the event space a probability.

Example 4.6. Considering rolling a six-sided fair dice:

- The sample space consists of outcome that a side is up: $\Omega = \{1, 2, 3, 4, 5, 6\}$
- Event space $\mathcal{F} = 2^{\Omega}$, which is all 2^6 subsets of Ω
- Because the dice is fair, the probability of each side is the same and equal 1/6. Then the probability measure is
$$P(E) = \frac{1}{6} \times \text{Number of elements of } E, \text{ for any event } E \in \mathcal{F}.$$
- Take an example event $E = \{2, 4, 6\}$, this event has 3 elements so its probability of happening is $P(E) = 3/6 = 0.5$.

Random Variables

A function $X : \Omega \to \mathbb{R}$ is called a *random variable*. The probability measure P_X defined by

$$P_X(A) = P(\{\omega \in \Omega : X(\omega) \in A\}) := P(X \in A)$$

for $A \subset \mathbb{R}$ is called the *distribution* (also called *rule* or *law*) of X. The notation

$$X \sim Q$$

is used to indicate that X has distribution Q; that is, $P_X = Q$. If the function Q is a conditional distribution for Y given X, then it can be written as

$$Y|X = x \ \sim \ Q_x.$$

For clarity, sometimes we will write Q_x as $Q_{Y|x}$.

Random Vectors

If $X_1, \ldots, X_n$ are random variables, then the function $X : \Omega \to \mathbb{R}^n$ defined by

$$X(\omega) = \begin{bmatrix} X_1(\omega) \\ \vdots \\ X_n(\omega) \end{bmatrix}, \quad \omega \in \Omega,$$

is called a *random vector*. Much of the notation and definitions presented above for random variables extend naturally and directly to random vectors.

4.1.3 Stochastic Process

A stochastic process is any collection of random variables $\{X_\theta : \theta \in \Theta\}$ defined on a common probability space $(\Omega, \mathcal{F}, P)$. These random variables map the sample space Ω to a state space S and are indexed by a parameter θ, where θ belongs to some index set Θ. In modeling real problems, stochastic process usually represent the numerical value of some system randomly changing over time.

More details on it's main components:

1. The state space S: also called the mathematical space, is the codomain of all X_θ. This represents the different values our system can take. S can be integers, real line, n-dimensional or even more abstract spaces.
2. The index set Θ: the parameter that indexes our stochastic process, determines the type of stochastic process that we currently working on. For example:
   - If Θ is the set of natural numbers, it can represent specific time point and our model is a discrete time stochastic process. The general subscript θ can change to n instead.
   - If Θ is an interval of the real line, we have a stochastic process of continuous time and θ can be replaced by t.
   - Moreover, Θ can be n-dimensional mathematical sets such as $\mathbb{R}^2$ where $\theta \in \Theta$ represents a point in space.

In general, for a discrete time process, the random variable X_n will depend on earlier values of the process, X_{n-1}, $X_{n-2}, \ldots$ Similarly, in continuous time, X_t will generally depends on values X_u for $u < t$. Therefore, we are often interested in the conditional distributions of X_{t_k} given the values of previous states $X_{t_{k-1}}, \ldots, X_{t_0}$, or

$$P(X_{t_k} | X_{t_{k-1}} = x_{t_{k-1}}, \ldots, X_{t_0} = x_{t_0}) \tag{10}$$

for some set $t_k > t_{k-1} > \cdots > t_0$.

4.1.4 Markov Chain

Markov chains are among the most important stochastic processes. They are stochastic processes for which the description of the current state fully captures all the information that could influence the future evolution of the process, rather than the previous steps that led up to the present. The usefulness of Markov chain arise broadly in statistical and information-theoretical contexts and are widely employed in economics, game theory, queuing (communication) theory, genetics, and finance.

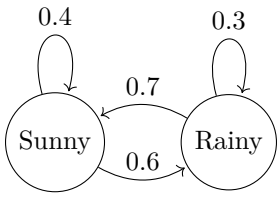

Figure 11: Transition diagram of a Markov chain

The Markov Property

A stochastic process is a Markov chain if it satisfies the Markov property. This assumes that prediction of future requires solely on the knowledge of the present alone. In other terms, a stochastic process is a Markov chain if for arbitrary time (index set) $t_k > t_{k-1} > \cdots > t_1$, the conditional distribution (10) reduces to the transition probability between the last two states, that is,

$$P(X_{t_k} | X_{t_{k-1}} = x_{t_{k-1}}, \ldots, X_{t_1} = x_{t_1}) = P(X_{t_k} | X_{t_{k-1}} = x_{t_{k-1}}).$$

This is also called the "memorylessness" property.

Transitions

Change between states in the system is called transition and this is associated with the transition probability for each possible state change. And a transition matrix Q with transition probability

$$Q_{ij} = P(X_{n+1} = j | X_n = i)$$

help describe probability of transitions and initial state (distribution) between the state in S. For example, the transition matrix of the transition diagram in figure 11 is

$$\mathrm{Q} = \begin{matrix} \\ Sunny \\ Rainy \end{matrix} \begin{matrix} \begin{matrix} Sunny & Rainy \end{matrix} \\ \begin{pmatrix} 0.4 & 0.6 \\ 0.7 & 0.3 \end{pmatrix} \end{matrix}.$$

Notice that total of transition probability from a state i to all other states (a row in the matrix) must be 1:

$$\sum_j Q_{ij} = 1$$

In Meteropolis–Hastings algorithm (next section), the state space is usually continuous so the transition probability is described by transition kernel (also called Markov kernel), which plays the same role that the transition matrix does for Markov chain with finite state space. Simply put, a transtional kernel Q is just the conditional distribution for the next state X_{n+1} given current state X_n:

$$X_{n+1} | X_n = x_n \ \sim \ Q_{x_n}.$$

Note: Now we can sum up our Markov chain process, which is characterized by a state space S, a transition matrix (or a transition kernel for infinite state space) Q and an initial distribution, or state X_0.

Since the system changes randomly, it is generally impossible to predict with certainty the state of a Markov chain at a given point in the future. However, the statistical properties of the system's future can be predicted. In many applications, it is these statistical properties that are important.

Stationary Distribution

The most interesting thing to us about Markov chain is its *stationary distribution* (also called *equilibrium distribution*). This is the distribution of our random variable that remains unchanged as time progresses.

For the case of finite state space, the stationary distribution is usually represented as a row vector π (sum of all entries is 1). Let Q be the transition matrix, we have

$$\pi = \pi Q.$$

This property is the basis for the general stochastic simulation methods known as Markov chain Monte Carlo, which are used to simulate sampling from complex probability distributions. The simple idea is to construct a Markov chain that has our desired distribution as its stationary distribution, thus allowing us to obtain a sample of the desire distributions by observing and recording the states from the chain as it progresses in time. The more steps that are included, the more closely the distribution of the sample matches the actual desired distribution. Our assignment will focus on the use of Metropolis–Hasting Algorithm (an algorithm from the class of Markov chain Monte Carlo methods) to achieve such result.

4.2 Metropolis–Hasting Algorithm

The initial version of this algorithm was discovered by Nicholas Metropolis[12] for the case of symmetrical proposal distributions. It was not until 1970 that its extension researched by Hasting[8] appeared in the statistical science community and become popular with the name "Metropolis–Hasting algorithm".

4.2.1 Theory

The Metropolis–Hastings algorithm is a simulation method that allows approximate sampling from arbitrary posterior distribution without computing the normalizing constant. Specifically, the algorithm gives a Markov chain that has the target law as its stationary distribution.

Let π denote a target distribution on some state space S with density f. Usually the function f is only known up to a proportionality constant, $f \propto g$. The chain runs by accepting or rejecting potential states generated using a conditional distribution J (also called *jumping distribution* or *proposal distribution*) with densities j_x. Let X_n denote the current state of the chain. Given $X_n = x_n$, a variable X^* is drawn from J_{x_n}, so

$$X^*|X_n = x_n \ \sim \ J_{x_n}.$$

The chances for accepting or rejecting a new value are based on a function r given by

$$r(x_n, x^*) = \frac{f(x^*)/j_{x_n}(x^*)}{f(x_n)/j_{x^*}(x_n)}.$$

Note that r can be computed if f is only known up to a proportionality constant; i.e., $f \propto g$. Therefore, we can replace in function f in formula above by a function g that we can compute.

$$r(x_n, x^*) = \frac{g(x^*)/j_{x_n}(x^*)}{g(x_n)/j_{x^*}(x_n)}.$$

If the proposal distribution is symmetric, i.e., $j_{x_n}(x^*) = j_{x^*}(x_n)$, the algorithm simply become Metropolis algorithm and r become

$$r(x_n, x^*) = \frac{g(x^*)}{g(x_n)}. \tag{11}$$

The next state for the Markov chain, X_{n+1}, will be either X_n or X^*, with

$$P(X_{n+1} = X^*|X_n = x_n, X^* = x^*) = \min\{1, r(x_n, x^*)\}.$$

At this stage, we have two choices: accept or reject X^* with acception probability $\alpha = \min\{1, r(x_n, x^*)\}$. To make decision follow this probability, we will use inverse transformation method. Simply put, generate a random number u from the standard uniform distribution $\mathcal{U}(0, 1)$. Because CDF of $\mathcal{U}(0, 1)$ is the identity function on $[0, 1]$, we have

$$P(u < \alpha) = F_u(\alpha) = \alpha.$$

Therefore, if we accept proposed value X^* when $u < \alpha$, then the propbality that we accept X^* will equal to the probability that $u < \alpha$, or

$$P(X_{n+1} = X^*|X_n = x_n, X^* = x^*) = P(u < \alpha) = \alpha.$$

4.2.2 Implementation Details

Choice of Proposal Distribution

The study of independent Metropolis–Hastings algorithms is certainly interesting, but their practical implementation is more problematic in that they are delicate to use in complex settings because the construction of the proposal is complicated. A more natural approach for the practical construction of a Metropolis–Hastings proposal is thus to take into account the value previously simulated to generate the following value; that is, to consider a *local* exploration of the neighborhood of the current value of the Markov chain. The implementation of this idea is to generate X^* according to

$$X^* = X_n + \epsilon$$

where ϵ is a random *perturbation* with distribution P_ϵ and density p_ϵ independent of X_n. In this assignment, we choose ϵ to be a normal random vector with mean $\mathbf{0}$:

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \Sigma). \tag{12}$$

Then the proposal distribution will be

$$X^*|X_n = x_n \sim \mathcal{N}(x_n, \Sigma).$$

In terms of the general Metropolis–Hastings algorithm, the proposal density $j_{x_n}(x^*)$ is now of the form $p_\epsilon(x^* - x_n)$. As p_ϵ is symmetric around $\mathbf{0}$, we have $p_\epsilon(x^* - x_n) = p_\epsilon(x_n - x^*)$, therefore this proposal distribution is symmetric. The Markov chain associated with this proposal distribution is a *random walk*. But, due to the additional Metropolis–Hastings acceptance step, the Metropolis–Hastings Markov chain X_n is not a random walk. Nevertheless, this variant of Metroplis–Hastings algorithm is called *random-walk Metropolis algorithm*.

Acceptance Rate

A Metropolis–Hastings algorithm can be characterized by the proportion of jumps that are accepted. The optimal jumping rule has acceptance rate around 0.44 in one dimension, declining to about 0.23 in high dimensions (roughly more than 5 paramters) (see [16, p. 306]). For our problems with 2 parameters, the optimal acceptance rate is about 30–40%.

4.3 Sampling the Prior Distribution

The model's parameters β and γ must be positive, from their definition in section 2.1.1. It makes sense to transform these parameters using logarithms, so they are defined on the whole real line.

For prior distribution, we will assume that β and γ are independent and their logarithms have normal distribution with mean 0. Because we are not certain about the value of parameter, we will choose a large variance, about 100^2.

$$\begin{aligned} \ln \beta &\sim \mathcal{N}(0, 100^2) \\ \ln \gamma &\sim \mathcal{N}(0, 100^2) \end{aligned} \tag{13}$$

We choose the Monte Carlo sample size to be 20 000. After some test run, we found that the best covariance matrix for the proposal distribution is the covariance of the prior distribution scaled by 3, which yield the acceptance rate of 34.8367%.

```
prior <- function(theta) prod(dnorm(theta, sd=100))
sample.size <- 20000

theta <- log(c(beta=1, gamma=1))
ptheta <- prior(theta)
# Generate matrix containing the sample
sample <- matrix(nrow=sample.size, ncol=2, dimnames=list(NULL, names(theta)))
# Initialize first sample with the starting value
sample[1, ] <- theta
for (i in seq_len(sample.size - 1)) {
    # Sample a candidate
    candidate <- theta + rnorm(2, sd=sqrt(3) * 100)
    # Calculate prior of candidate and store it in case it gets accepted
    r <- prior(candidate) / ptheta
    if (runif(1) < r) {
        theta <- candidate
        ptheta <- prior(theta)
    }
    sample[i+1, ] <- theta
}
```

The trace of parameters and the sampling process is shown in figure 12 and 13. Note that the value of β and γ is displayed in logarithmic scale.

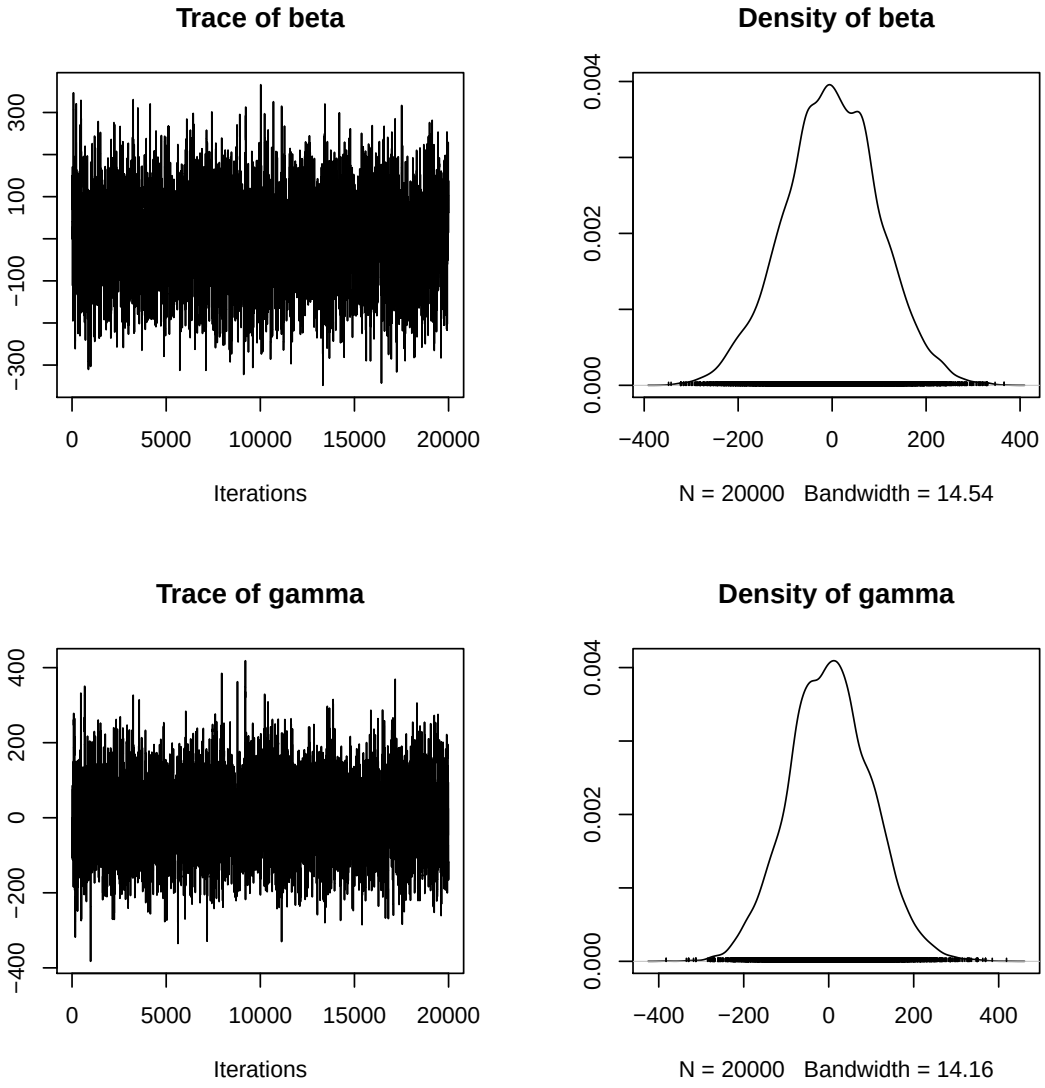

Figure 12: Trace of $\ln \beta$ and $\ln \gamma$

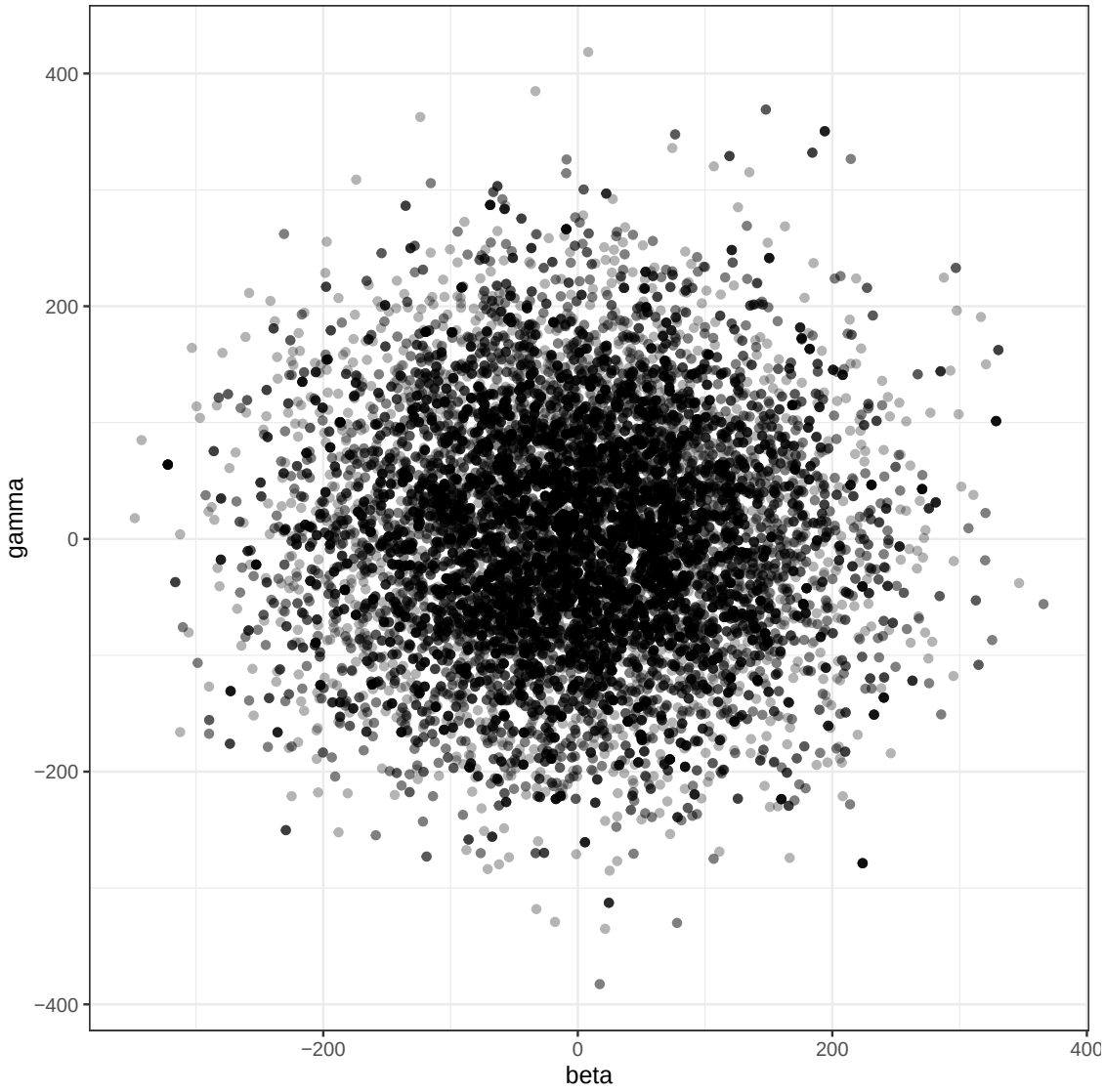

Figure 13: Sampling process of $\ln \beta$ and $\ln \gamma$

5 Estimating Basic Reproduction Number

5.1 Basic Reproduction Number

In epidemiology, the *basic reproduction number* (as known as *basic reproductive number*, *basic reproduction ratio* or *basic reproductive rate*), denoted as R_0, is defined as the expected number of secondary cases produced by a single (typical) infection in a completely susceptible population. The definition describes the state where no other individuals are infected or immunized (naturally or through vaccination), or occurrence of any deliberate intervention in disease transmission. It is important to note that R_0 is actually a dimensionless number and not a rate.

We can use the fact that R_0 is a dimensionless number to help us get the intuitive equation

$$R_0 \propto \frac{\text{infection}}{\text{contact}} \times \frac{\text{contact}}{\text{time}} \times \frac{\text{time}}{\text{infection}}$$

or more specifically

$$R_0 = abd$$

where a is the transmissibility (the probability of infection given contact between a susceptible and infected individual), b is the average rate of contact between susceptible and infected individuals, and d is the duration of infectiousness.

The most important uses of R_0 are determining if an emerging infectious disease can spread in a population and determining what proportion of the population should be immunized through vaccination to eradicate a disease.

In order to prove the uses of R_0 in SIR model, we know that an epidemic occurs when the number of infected individuals increases (the infectious rate $\mathrm{d}I/\mathrm{d}t$ is positive) and it cannot be spread when the infectious rate is negative since the number of infected individuals decrease. Moreover, since $\beta = ab$ (mentioned in section 2.1.1) and by the assumption that all rates are constant, the expected duration of infection is simply the inverse of the removal rate, $d = \gamma^{-1}$, we have

$$R_0 = \frac{\beta}{\gamma} \tag{14}$$

Now the aim is to find the relationship between the number of infected cases and basic reproduction number. From derivative formula of $\mathrm{d}I/\mathrm{d}t$ in system (1), we have

$$\begin{aligned}\frac{\mathrm{d}I}{\mathrm{d}t} &= \frac{\beta}{N} IS - \gamma I \\ &= \left(\beta \frac{S}{N} - \gamma\right) I \\ &= \gamma \left(R_0 \frac{S}{N} - 1\right) I\end{aligned}$$

At the outset of an epidemic, nearly everyone (except the index case/first case) is susceptible, so we can say that $S/N \approx 1$. Substituting $S/N = 1$ to above equation, we have

$$\frac{\mathrm{d}I}{\mathrm{d}t} = \gamma(R_0 - 1)I$$

Since I and γ is always positive, the sign of infectious rate is followed by the sign of $R_0 - 1$. From the above equation as well as in commonly used infection models, if $R_0 > 1$, new outbreaks of the virus might occur in the future, but not if $R_0 < 1$. Moreover, the higher R_0 reaches, the harder it is to control the epidemic.

Disease	R_0
Measles	12–18
Chickenpox (varicella)	10–12
SARS	0.19–1.08
Influenza	1.4–2.8
Ebola	1.5–1.9
Influenza (2009 pandemic strain)	1.4–1.6
MERS	0.3–0.8
COVID-19	3.28

Table 1: Values of R_0 of well-known infectious diseases

5.2 Bayes Estimate of R_0

In Bayesian interference, both the unknown parameter and data are viewed as random vectors. For notation, Θ is the random parameter with θ is a possible value for Θ and X is the random data with x is a possible value. The unknown parameter of our SIR model is $\Theta = (\beta, \gamma)^\top$.

Theorem 5.1 (Bayes' theorem)**.** *Bayes' theorem for probability densities is stated mathematically as*

$$p_{\Theta|x}(\theta) = \frac{p_{X|\theta}(x)p_\Theta(\theta)}{p_X(x)}.$$

where

- $p_\Theta(\theta)$*, the* prior density*, represents probabilities before data are observed;*
- $p_{\Theta|x}(\theta)$ *is the* posterior density *of* Θ *given* $X = x$*;*
- $p_{X|\theta}(x)$*, the* likelihood function*, is the conditional density of* X *given* $\Theta = \theta$*;*
- $p_X(x)$ *is called the* normalizing constant *because it does not depend on* Θ*.*

The denominator in Bayes' theorem is usually dropped and Bayes' theorem can be stated in another way: The posterior density is proportional to the product of the likelihood function and the prior density, or

$$p_{\Theta|x}(\theta) \propto p_{X|\theta}(x)p_\Theta(\theta).$$

This product is called the *unnormalized* posterior density.

From (14), the basic reproduction number is a function of the parameter, i.e., $R_0 = R_0(\Theta)$. We want the Bayes estimator $\hat{R}_0$ of R_0 to minimize the mean square error

$$\mathrm{E}[(\hat{R}_0 - R_0(\Theta))^2 | X = x]$$

where the expectation is taken over the posterior distribution of Θ. Then $\hat{R}_0$ is simply the posterior mean (see [10, p. 117])

$$\hat{R}_0 = \mathrm{E}[R_0(\Theta)|X = x] = \int R_0(\theta)p_{\Theta|x}(\theta)\,\mathrm{d}\theta \tag{15}$$

To compute the posterior density $p_{\Theta|x}(\theta)$, we have to compute the normalizing constant, $p_X(x)$. However, $p_X(x)$ is unknown since we cannot assume data X to follow any type of distribution. An alternative method is to use Metropolis–Hastings algorithm, which does not require computing the normalizing constant. This will be discussed in later section.

5.3 Monte Carlo Intergration

5.3.1 Classical Monte Carlo Integration

In mathematics, Monte Carlo integration uses random sampling of a function to numerically compute an estimate of its integral.

Consider a distribution P_X with density p_X. Suppose we want to compute the expected value (or expectation) of an random variable $Y = f(X)$:

$$\mu = \mathrm{E}(Y) = \int f(x)p_X(x)\,\mathrm{d}x.$$

The principle of the Monte Carlo method for approximating above integration is to generate a random sample $(X_1, \ldots, X_n)$ from the distribution P_X and propose as an approximation the sample mean (see [3])

$$\hat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} f(X_i) = \frac{1}{n}\sum_{i=1}^{n} Y_i. \tag{16}$$

The primary justification for classical Monte Carlo is through the strong law of large number. It states that the sample mean converges almost surely to the expected value, that is, $\hat{\mu}_n \to \mu$ almost surely as $n \to \infty$.

Also according to central limit theorem, as $n \to \infty$, the sample mean $\hat{\mu}_n$ will converge in distribution to a normal distribution, or

$$\hat{\mu} \implies \mathcal{N}\left(\mu, \frac{\sigma^2}{n}\right) \quad \text{when } n \to \infty, \tag{17}$$

where σ^2 is the variance of Y, which can be estimated by the sample variance

$$\hat{\sigma}_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \hat{\mu}_n)^2. \tag{18}$$

The only tricky issue is that the randomness involved is the pseudorandomness of computer simulation, rather than randomness of real-world phenomena. Thus it is a good idea to use terminology that emphasizes the difference:

- $\hat{\mu}_n$ is called the *Monte Carlo approximation* of μ, rather than the "point estimator" of μ.
- n is called the *Monte Carlo sample size*, rather than just the "sample size".
- $\hat{\sigma}_n/\sqrt{n}$ is called the *Monte Carlo standard error* (MCSE), rather than just the "standard error".

5.3.2 Monte Carlo Methods Based on Markov Chains

It not the case that we can always sample directly from distribution P_X. But by using an Markov chain with stationary distribution P_X, it is possible to obtain a sample $X_1, \ldots, X_n$ approximately distributed from P_X without directly simulating from P_X. But because each state in the Markov chain is depend on previous state through transition probability, a chain $(X_1, \ldots, X_m)$ resulting from a Markov chain Monte Carlo algorithm is not a random sample. Nevertheless, we can guarantee that the sample mean from (16) with sample from a Markov chain will converge to μ. Although there are many MCMC methods, the Metropolis–Hastings algorithms have the advantage of imposing minimal requirements on the target distribution π and allowing for a wide choice of possible implementation.

Central limit theorem still holds for Markov chain, but due to correlation between states in the Markov chain, σ^2 in (17) now becomes

$$\sigma^2 = \sigma_{Y_1}^2 + 2\sum_{i=1}^{\infty}\operatorname{cov}(Y_1, Y_{i+1}).$$

Therefore, the sample variance estimator in (18) is no longer accurate. Many methods of variance estimation for Monte Carlo Markov chain methods have been proposed, but the batch means estimator is usually a good choice.

Batch Means Estimator

A batch is simply a subsequence of consecutive iterates of the Markov chain $X_{k+1}, \ldots, X_{k+b}$. The number b is called the batch length. Suppose b divides n evenly. Divide the whole run into a nonoverlapping batches of length b. Average these batches:

$$\hat{\mu}_{b,k} = \frac{1}{b}\sum_{i=b(k-1)+1}^{bk} Y_i. \tag{19}$$

Then the batch means estimator is

$$\hat{\sigma}_{b,n}^2 = \frac{b}{a-1}\sum_{k=1}^{a}(\hat{\mu}_{b,k} - \hat{\mu}_n)^2. \tag{20}$$

A batch length of $b = \lfloor\sqrt{n}\rfloor$, suggested by [7], is often used in practice.

6 Affect of Social Distancing in Vietnam

6.1 Motivation

In this section, we will focus on determining the affect of social distancing policy of a country on COVID-19. To do this, we calculate the Bayes estimate of the basic reproduction number R_0 one week before and one week after the social distancing policy took place and draw conclusion based on their difference.

We are performing the test on Vietnam. The reason being the size of Vietnam and the big affect that COVID-19 had upon Vietnam. Most importantly, this country has a nation wide social distancing period, which means the "uniform" social distancing policy will be better reflected in our calculated R_0, thus making our result more reliable for interpretation.

For calculation method, we will be estimating the Bayes estimate of R_0 by using Monte Carlo integration (see (15) and (16))

$$\hat{R}_0 = \mathrm{E}[R_0(\Theta)|X = x] = \int R_0(\theta) p_{\Theta|x}(\theta)\,\mathrm{d}\theta \approx \frac{1}{n}\sum_{i=1}^{n}\frac{\beta_i}{\gamma_i} \tag{21}$$

The sample for this integration follows the posterior distribution of our parameter $\Theta = (\beta, \gamma)^\top$, given observed data X (number of Susceptible, Infected and Recovered people). However, this posterior distribution is hard to sample from, so we will use the Metropolis–Hasting algorithm to do that, knowing that our posterior density is proportional to the likelihood function multiplied by the prior density.

The dataset will be taken from COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (link). The time series data is located in directory `csse_covid_19_data/csse_covid_19_time_series`, which contains following files:

- `time_series_covid19_confirmed_global.csv`: Global cumulative confirmed cases
- `time_series_covid19_recovered_global.csv`: Global recovered cases
- `time_series_covid19_deaths_global.csv`: Global deaths

For this assignment, we will only interested in data of Vietnam.

6.2 Implementation Details

6.2.1 Logarithmic Scale

To avoid underflow and overflow in our program, we will use log probability, which represent probability in logarithmic scale. Representing probabilities in this way has several practical advantages:

- Speed: Since multiplication is more expensive than addition, taking the product of a high number of probabilities is often faster if they are represented in log form. (The conversion to log form is expensive, but is only incurred once.) Multiplication arises from calculating the probability that multiple independent events occur: the probability that all independent events of interest occur is the product of all these events' probabilities.
- Accuracy: The use of log probabilities improves numerical stability, when the probabilities are very small, because of the way in which computers approximate real numbers.
- Simplicity: Many probability distributions have an exponential form. Taking the log of these distributions eliminates the exponential function, unwrapping the exponent. For example, the log probability of the normal distribution's probability density function is

  $$-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 - \ln\sigma - \frac{1}{2}\ln(2\pi)$$

  instead of

  $$\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

  Log probabilities make some mathematical manipulations easier to perform.

6.2.2 Efficent Jumping Rule

In this section, we will find the ideal covariance matrix Σ in (12). There are two main concerns when choosing covariance matrix Σ:

- It must large enough so that each jump goes a reasonable distance in the parameter space (otherwise the random walk moves too slowly).
- It must small enough so that the jumps are not rejected too frequently (otherwise the random walk wastes too much time standing still).

Thus each problem require a different suitable covariance matrix. [15] and [16, p. 314] suggests us an general approach to find a suitable covariance matrix for a lot of problems.

First find the posterior mode (parameter value that maximize posterior density) by an optimization algorithm. Denote the mode of target distribution by $\tilde{\theta}$. Once the mode have been found, we can construct an approximation based on the multivariate normal distribution. We will fit a normal distribution to the first two derivatives of the log posterior density function at $\tilde{\theta}$:

$$p_{\text{normal approx}}(\theta) \approx \mathcal{N}(\theta|\mu = \tilde{\theta}, \Sigma = V).$$

The covariance matrix of this distribution is the inverse of the curvature (Hessian) of the log posterior density at the mode:

$$V = -H(\tilde{\theta})$$

The Hessian matrix is simply a matrix of second order partial derivatives, for example the Hessian of a function $f(x, y)$ is

$$\begin{bmatrix} \dfrac{\partial^2 f}{\partial x^2} & \dfrac{\partial f}{\partial y \partial x} \\ \dfrac{\partial f}{\partial x \partial y} & \dfrac{\partial^2 f}{\partial y^2} \end{bmatrix}.$$

It is guaranteed that the Hessian matrix at posterior mode is negative-definite. Then calculate Hessian $H(\tilde{\theta})$ of log posterior density at posterior mode $\tilde{\theta}$.

The jumping distribution will have the same shape as this normal distribution, that is, $J_\theta = \mathcal{N}(\theta, -c^2 V_\theta)$. This allows for an efficient exploration of the posterior distribution, at least in the neighborhood of the mode. Among this class of jumping rules, the most efficient has scale $c = 2.4/\sqrt{d}$, where where d is the dimension of parameter space. Finally, the covariance matrix in (12) will be

$$\Sigma = -\frac{2.4^2}{d}[H(\tilde{\theta})]^{-1} \tag{22}$$

6.2.3 Likelihood Function

Our time series data is not a direct measure of the number of disease cases present in a particular population at a given time, but rather the cumulative of the number of new cases from daily reports. The number of confirmed cases in the dataset the cumulative number of infected cases, include the people who have recovered of deceased:

$$C = I + R.$$

Because we only consider a week interval, we can assume the total population N is constant. Thus we have following modification of system (1):

$$\begin{aligned} \frac{\mathrm{d}C}{\mathrm{d}t} &= \frac{\beta}{N}(N - C)(C - R) \\ \frac{\mathrm{d}R}{\mathrm{d}t} &= \gamma(C - R) \end{aligned} \tag{23}$$

Consider a day n and next day in the dataset with data $X_n = (C_n, R_n)$ and $X_{n+1} = (C_{n+1}, R_{n+1})$. Then the number of people move from S compartment to I compartment (new infected cases) is

$$\Delta N_{SI} = C_{n+1} - C_n$$

and the number of people move from I compartmet to R compartment (new recovered cases or deaths) is

$$\Delta N_{IR} = R_{n+1} - R_n.$$

Using Runge–Kutta method for system (23), we can estimate the number of cases on the next day given number of cases in day n is $\hat{X}_{n+1} = (\hat{C}_{n+1}, \hat{R}_{n+1})$. Therefore we can get an estimted number for above quantities are

$$\Delta \hat{N}_{SI} = \hat{C}_{n+1} - C_n$$

and

$$\Delta \hat{N}_{IR} = \hat{R}_{n+1} - R_n.$$

We assume that the number of new cases and the number of recovered cases and deaths are independent and the number of cases reporte in each day interval is independent. Because the error of reporting and testing COVID-19, the data will tend to follow Poisson distribution with mean is our estimated value.

$$\Delta N_{SI} \sim \text{Pois}(\Delta \hat{N}_{SI})$$

$$\Delta N_{IR} \sim \text{Pois}(\Delta \hat{N}_{IR})$$

Then we have the likelihood function between each day is

$$\begin{aligned} p(x_{n+1}|\theta, x_n, \ldots, x_1) &= p(x_{n+1}|\theta, x_n) \\ &= \text{Pois}(C_{n+1} - C_n|\lambda = \hat{C}_{n+1} - C_n)\,\text{Pois}(R_{n+1} - R_n|\lambda = \hat{R}_{n+1} - R_n) \end{aligned}$$

and the likelihood function for the entire week is

$$\begin{aligned} p(x_7, \ldots, x_1|\theta) &= \prod_{i=1}^{6} p(x_{i+1}|\theta, x_i, \ldots, x_1) \\ &= \prod_{i=1}^{6} \text{Pois}(C_{i+1} - C_i|\lambda = \hat{C}_{i+1} - C_i)\,\text{Pois}(R_{i+1} - R_i|\lambda = \hat{R}_{i+1} - R_i) \end{aligned} \tag{24}$$

6.3 Sampling Parameters

6.3.1 Random-Walk Metropolis Algorithm

1. Use a numerical optimization routine to maximize log posterior density. Denote the posterior mode by $\tilde{\theta}$.
2. Let $\tilde{\Sigma}$ be scaled inverse of the Hessian of log posterior density computed at posterior mode $\tilde{\theta}$ using formula (22).
3. Draw initial state θ_0 from $\mathcal{N}(\tilde{\theta}, \tilde{\Sigma})$ and set $i = 1$.
4. Draw a step size ϵ from $\mathcal{N}(\mathbf{0}, \tilde{\Sigma})$. Given a new value θ^*.

   $$\theta^* = \theta_i + \varepsilon$$

   Because our proposal distribution is symmetric, the Metropolis–Hastings acceptance ratio is calculated using formula (11) expressed in logarithms

   $$\ln r(\theta^*, \theta_i) = \ln \pi(\theta^*) - \ln \pi(\theta_i)$$

5. Generate a uniform random number from $\mathcal{U}(0, 1)$. If $\ln u < \ln r$, then accept θ^* as next state. Otherwise, reject the new state and copy the old state θ forward. Note that we do not need a separate case for $r > 1$ because it is already included in case $u < r$, as u is always less than or equal 1.
6. If $i < m$, set $i = i + 1$ and repeat from step 4.

6.3.2 Code Implementation and Result

System (23) can be computed by following R function:

```
CR.model <- function(t, y, parms) with(as.list(c(y, parms)), {
    dC.dt <- beta / N * (N - C) * (C - R)
    dR.dt <- gamma * (C - R)
    c(dC.dt, dR.dt)
})
```

To sample from multivariate normal distribution, we will use package `MASS`[18]. Because the step sizes of the random walk and random numbers u are independent to the state of the Markov chain, we can compute those values in advance to improve performance of our program.

```
library(MASS)

rw.metro <- function(start, func, sample.size, pcov.scale=1, ...) {
    # Dimension of state space
    dim <- length(start)

    opt <- optim(start, func, control=list(fnscale=-1), hessian=TRUE, ...)
    if (opt$convergence != 0) stop(opt$message)
    prop.cov <- -2.4^2 / dim * solve(opt$hessian)
    # Current state of the Markov chain
    theta <- mvrnorm(n=1, mu=opt$par, Sigma=prop.cov)
    ptheta <- func(theta, ...)
    # Generate matrix containing the sample. Initialize first sample with the starting value
    sample <- matrix(nrow=sample.size, ncol=dim)
    colnames(sample) <- names(start)
    sample[1, ] <- theta
    # Generate uniform random numbers in advance, to save computation.
    log.u <- log(runif(sample.size - 1))
    # Proposal is a multivariate standard normal distribution. Generate sample and
    # later on use linearity property of Gaussian distribution
    normal.shift <- mvrnorm(n=sample.size-1, mu=rep(0, dim), Sigma=prop.cov)
    for (i in seq_len(sample.size - 1)) {
        # Sample a candidate
        candidate <- theta + normal.shift[i, ]
        # Calculate func of candidate and store it in case it gets accepted
        log.r <- func(candidate, ...) - ptheta
        if (log.u[i] < log.r) {
            theta <- candidate
            ptheta <- func(theta, ...)
        }
        sample[i+1, ] <- theta
    }
    return(sample)
}
```

From (13), prior distribution for $\ln\beta$ and $\ln\gamma$ are independent normal distributions $\mathcal{N}(0, 100^2)$.

```
log.prior <- function(theta) sum(dnorm(theta, sd=100, log=TRUE))
```

Likelihood function can be computed from equation (24).

```
log.likelihood <- function(theta, data, pop) {
    n <- 20
    ninterval <- nrow(data) - 1
    change <- data[1:ninterval, c("S.I", "I.R")]
    estimated <- t(apply(
        data[1:ninterval, c("C", "R")],
        MARGIN = 1,
        FUN = function(x) {
            sol <- rk4(
                y0 = x,
                times = seq(0, 1, by=1/n),
                func = CR.model,
                parms=c(exp(theta), N=pop)
            )
            sol[n+1, -1] - sol[1, -1]
        }
    ))
    sum(dpois(change, estimated, log=TRUE))
}
```

Then the log unnormalized posterior density is the sum of log prior density and log likelihood.

```
log.posterior <- function(theta, data, pop) {
    log.prior(theta) + log.likelihood(theta, data, pop)
}
```

To analyze the dataset, first we need to import it.

```
dataset <- paste(
    "https://github.com/CSSEGISandData/COVID-19",
    "raw/master/csse_covid_19_data/csse_covid_19_time_series",
    "time_series_covid19_%s_global.csv",
    sep = "/"
)
confirmed <- read.csv(sprintf(dataset, "confirmed"), check.names=FALSE)
recovered <- read.csv(sprintf(dataset, "recovered"), check.names=FALSE)
deaths <- read.csv(sprintf(dataset, "deaths"), check.names=FALSE)
report.date <- as.Date(names(confirmed[, -(1:4)]), "%m/%d/%y")
```

We will restrict to only the data of Vietnam.

```
confirmed.VN <- unlist(subset(confirmed, `Country/Region` == "Vietnam", select=-(1:4)))
recovered.VN <- unlist(subset(recovered, `Country/Region` == "Vietnam", select=-(1:4)))
deaths.VN <- unlist(subset(deaths, `Country/Region` == "Vietnam", select=-(1:4)))
observed <- data.frame(
    date = report.date,
    C = confirmed.VN,
    R = recovered.VN + deaths.VN,
    row.names = NULL
)
observed$I.R <- c(diff(observed$R), NA)
observed$S.I <- c(diff(observed$C), NA)
# Population of Vietnam
pop <- 97338579
```

Now we will analyze the data for 1-week period started at 2020-03-25.

```
interval1 <- seq(as.Date("2020-03-25"), by = "day", length.out = 7)
obs1 <- subset(observed, date %in% interval1)
```

We start off with crude estimates of β and γ. We have finite-difference approximation of system (23)

$$\frac{C(2) - C(1)}{1} \approx \frac{\beta}{N}(N - C(1))(C(1) - R(1))$$
$$\frac{R(2) - R(1)}{1} \approx \gamma(C(1) - R(1))$$

Then β and γ can be approximated by

$$\beta \approx \frac{(C(2) - C(1))N}{(N - C(1))(C(1) - R(1))}$$
$$\gamma \approx \frac{C(2) - C(1)}{C(1) - R(1)}$$

```
init.beta1 <- (obs1$C[2] - obs1$C[1]) * pop / (pop - obs1$C[1]) / (obs1$C[1] - obs1$R[1])
init.gamma1 <- (obs1$R[2] - obs1$R[1]) / (obs1$C[1] - obs1$R[1])
```

Then we take sample from posterior distribution with Monte Carlo sample size $n = 50\,000$.

```
sample.size <- 50000
sample1 <- rw.metro(
    start = log(c(beta=init.beta1, gamma=init.gamma1)),
    func = log.posterior,
    sample.size = sample.size,
    data = data.matrix(obs1[, -1]),
    pop = pop
)
```

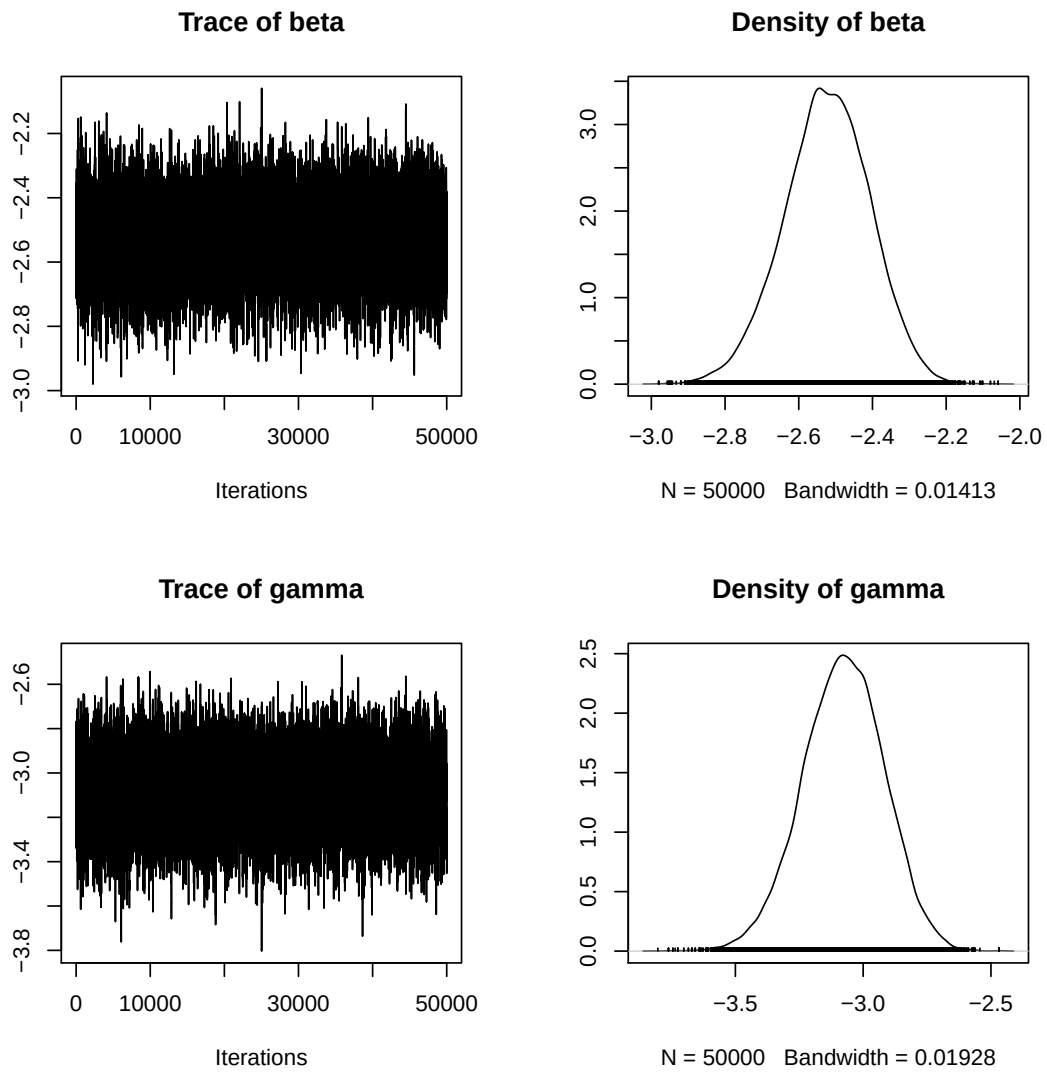

Figure 14: Trace of $\ln \beta$ and $\ln \gamma$ for 1-week interval from 2020-03-25 to 2020-03-31

The procedure is the same for the week interval from 2020-04-08 to 2020-04-14.

```
interval2 <- seq(as.Date("2020-04-08"), by = "day", length.out = 7)
obs2 <- subset(observed, date %in% interval1)
init.beta2 <- (obs2$C[2] - obs2$C[1]) * pop / (pop - obs2$C[1]) / (obs2$C[1] - obs2$R[1])
init.gamma2 <- (obs2$R[2] - obs2$R[1]) / (obs2$C[1] - obs2$R[1])
sample2 <- rw.metro(
    start = log(c(beta=init.beta2, gamma=init.gamma2)),
    func = log.posterior,
    sample.size = sample.size,
    data = data.matrix(obs2[, -1]),
    pop = pop
)
```

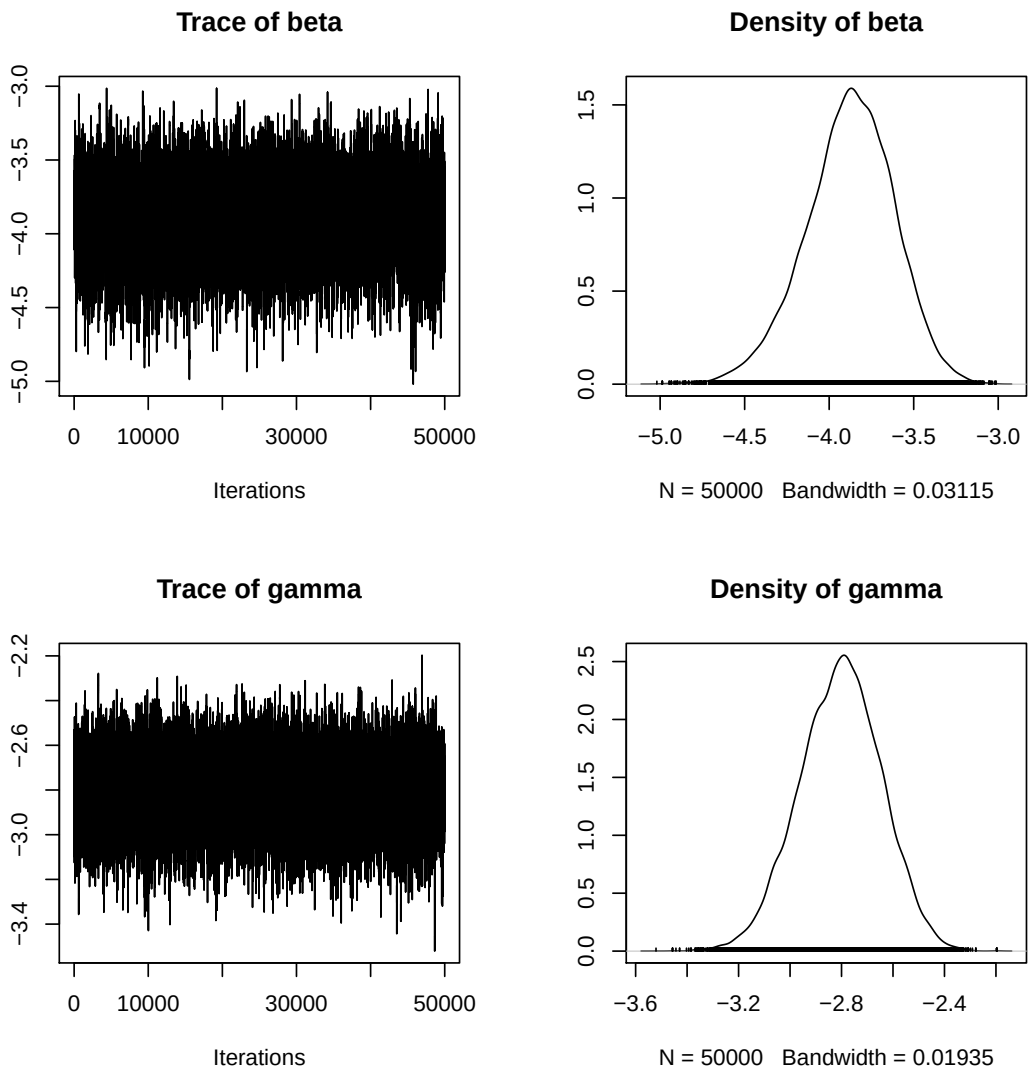

Figure 15: Trace of $\ln \beta$ and $\ln \gamma$ for 1-week interval from 2020-04-08 to 2020-04-16

6.4 Estimating R_0

After obtained the sample, we will calculate the Bayes estimate of R_0 using formula (21). But because we sample from logarithm of parameters, the formula must been expressed in logarithms:

$$\hat{R}_0 \approx \frac{1}{n} \sum_{i=1}^{n} \exp(\ln \beta_i - \ln \gamma_i).$$

```
sample.R0 <- exp(sample1[, "beta"] - sample1[, "gamma"])
R0 <- mean(sample.R0)
```

Then we use batch means estimator to estimate the MCSE, using equation (19) and (20).

```
blen <- floor(sqrt(sample.size))
nbatch <- sample.size %/% blen
batch.means <- sapply(
    1:nbatch,
    FUN = function(k) mean(sample.R0[((k-1)*blen + 1):(k * blen)])
)
var.hat <- blen / nbatch * sum((batch.means - R0)^2)
sqrt(var.hat / sample.size)
```

Time interval	Estimate	MCSE	95% confidence interval
2020-03-25–2020-03-31	1.7743	0.0045	1.7743 ± 0.0088
2020-04-08–2020-04-16	0.3583	0.0013	0.3583 ± 0.0025

Table 2: Monte Carlo approximation of posterior mean of R_0 before and during social distancing

6.5 Conclusion

In this section, we focused on an observation period that began after the lock-down was set on COVID-19 epidemic outbreak in Vietnam using the number of daily infected cases.

6.5.1 Changes

There has been some changes during our experimentation period. We initially intended on sampling β and γ with a gamma prior distribution but eventually backed down due to the complexity and the fact that we did not find resources backing up that choice of distribution. Additionally, since we have never worked with Gamma distribution, we could not really check if the sample we produce were "beautiful" enough to continue working with it. Thus, we decided on switching over to a log-normal prior distribution.

Other changes that we made to the algorithm and methodology was also mentioned in section 6.2, we already explained them in detail there.

6.5.2 Discussion

According to the estimation above, after one week of social distancing policy (from 2020-04-08 to 2020-04-16), we obtained an effective reproduction number $R_0 = 0.3583$ that was divided by a factor 5, compared to the estimation of the R_0 carried out in Vietnam at the early stage of the epidemic—before the country went into lock-down ($R_0 = 1.7743$). This indicates that the restriction policies were very efficient in decreasing the contact rate and therefore the number of infectious cases. In particular, the value $R_0 = 0.3583$ is significantly below the threshold value 1 where the epidemic starts dying out.

7 Conclusion

In this assignment, we have presented all of our findings regarding pandemic system modeling, specifically the COVID-19 pandemic that has been around for nearly a year and months to come. We hope that we have included all the necessary knowledge in a logical and easy to follow reading manner in the report. Our work is designed and shown in levels, increasing in difficulty. First, we went through the process of constructing a mathematical model for a pandemic. Next, we next take a step further, alter its parameters and then apply the system for modeling real world data, finally drawing meaningful conclusions as the end result.

Through this report, we have learnt much about our major. It serves at a big leap for our group into the field of mathematical modeling and statistics. And later as the basis of machine learning.

It is also important to point out that our report was done, and constructed, in a hasty manner. It is just not a good time for us to work on such a big assignment right now. You can imagine our group making

progress day by day and continuously meetup for idea exchange just to meet the deadline. Nevertheless, we as a group hope that you find something useful, or just any interesting ideas while skimming through this report.

To anyone reading this. Thank you for your time.

References

[1] Sheldon Axler. Measures. In *Measure, Integration & Real Analysis*, Graduate Texts in Mathematics, chapter 2. Springer International Publishing, 2020.

[2] F. Baudoin. Stochastic processes. In Penelope Peterson, Eva Baker, and Barry McGaw, editors, *International Encyclopedia of Education*, pages 451–452. Elsevier, third edition, 2010.

[3] Christian P. Robert; George Casella. *Monte Carlo Statistical Methods*, page 75. Springer Texts in Statistics. Springer, 2004.

[4] Christian P. Robert; George Casella. *Introducing Monte Carlo Methods with R*. Use R! Springer, 2010.

[5] Johan Dahlin, Fredrik Lindsten, and Thomas B. Schön. Particle Metropolis–Hastings using gradient and Hessian information. *Statistics and Computing*, 25(1):81–92, Jan 2015.

[6] James M. Flegal and Lei Gong. Relative fixed-width stopping rules for Markov chain Monte Carlo simulations, 2013.

[7] James M. Flegal and Galin L. Jones. Batch means and spectral variance estimators in Markov chain Monte Carlo. *Annals of Statistics*, 38(2):1034–1070, 04 2010.

[8] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 04 1970.

[9] James Holland Jones. Notes on R_0, 2007.

[10] Robert W. Keener. *Theoretical Statistics: Topics for a Core Course*. Springer Texts in Statistics. Springer, 2010.

[11] Raimer Kress. Initial Value Problems. In *Numerical Analysis*, Graduate Texts in Mathematics, chapter 10. Springer, 1998.

[12] N. Metropolis et al. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 1953.

[13] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2020.

[14] Christian P. Robert. The Metropolis–Hastings algorithm, 2015.

[15] Frank Schorfheide. Loss function-based evaluation of DSGE models. *Journal of Applied Econometrics*, 15(6):645–670, 2000.

[16] Andrew Gelman; John Carlin; Hal Stern; and Donald Rubin. *Bayesian Data Analysis*. Chapman & Hall, 2nd edition, 2004.

[17] Fred Brauer; Pauline van den Driessche; Jianhong Wu. *Mathematical Epidemiology*. Lecture Notes in Mathematics. Springer, 2008.

[18] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002.