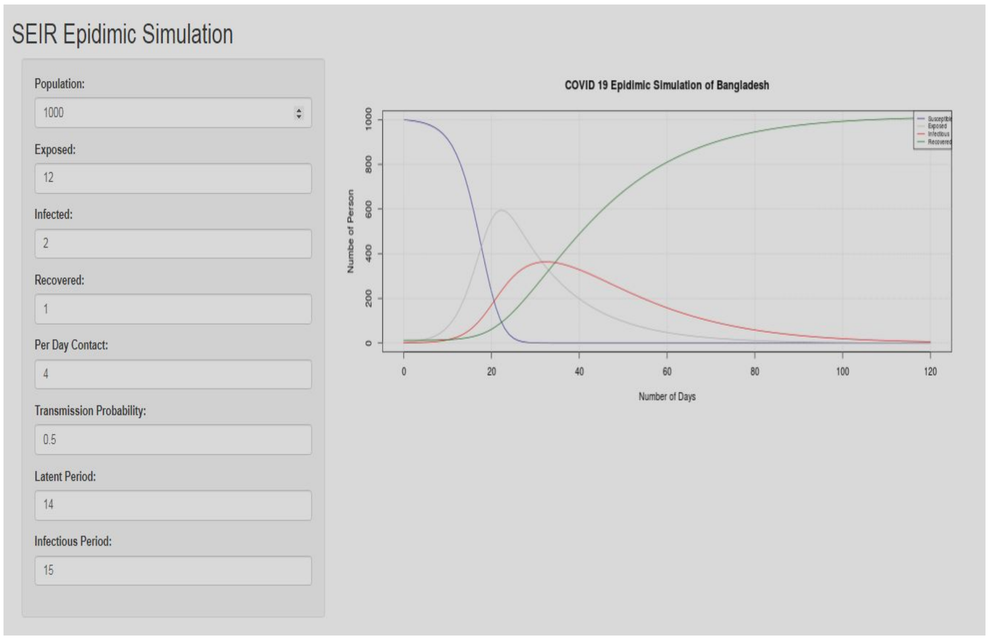
Documentation of SEIR Epidemiological Simulation


SEIR Epidimic Simulation

Population:
1000

Exposed:
12

Infected:
2

Recovered:
1

Per Day Contact:
4

Transmission Probability:
0.5

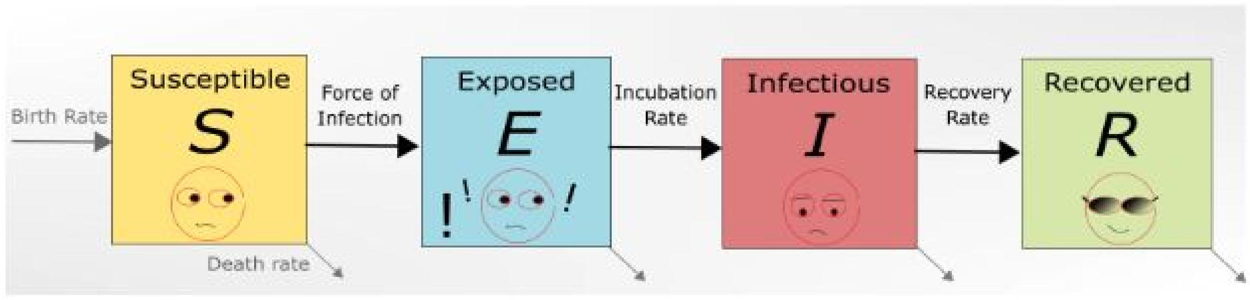
Latent Period:
14

Infectious Period:
15

Table of Content

1. Background of SEIR Model

The SEIR model covers four infectious disease stages: Susceptible (S), Exposed (E), Infectious (I), and Recovery (R). The host begins in the susceptible stage before being exposed to the disease and then becoming infectious. From the infectious stage, the host then reaches the recovery stage and is no longer susceptible to the disease because of developing immunity. As shown above, influenza is a good example of this model since almost all people are susceptible without vaccination, can be exposed to the disease from infected people, and become highly infectious for a short period of time before reaching the recovery stage.



The WHO used SEIR models to characterize and forecast the early stages of the COVID-19 outbreak in Wuhan

$$\frac{dS}{dt} = -\frac{\beta SI}{N}$$
$$\frac{dE}{dt} = \frac{\beta SI}{N} - \sigma E$$
$$\frac{dI}{dt} = \sigma E - \gamma I$$
$$\frac{dR}{dt} = \gamma I$$

$$N = S + E + I + R$$

2. Description of Programming Parameters

2.1 Input Parameter

Parameter	Description
contact_rate	Number of contacts per day
transmission_probability	Transmission probability
infectious_period	Infectious period
latent_period	latent period
W	Susceptible hosts /Population
X	Infectious hosts
Y	Recovered hosts
Z	Exposed hosts

2.2 Calculating S,E,I,R

```
N = W + X + Y + Z
initial_values = c (S = W/N, E = Z/N, I = X/N, R = Y/N)
```

2.3 Calculating Coefficients

```
beta_value = contact_rate * transmission_probability
gamma_value = 1 / infectious_period
delta_value = 1 / latent_period
```

2.4 Calculating Derivatives

```
dS = (-beta * S * I)
dE = (beta * S * I) - (delta * E)
dI = (delta * E) - (gamma * I)
dR = (gamma * I)
```

2.5 Calculating R_0

```
Ro = beta_value / gamma_value
```

2.6 Defining time period

```
timepoints = seq (0, 120, by=1)
```

2.7 Executing the model

```
parameter_list = c (beta = beta_value, gamma = gamma_value, delta = delta_value)
output = lsoda (initial_values, timepoints, seir_model, parameter_list)
```

3. Basic R Code

```
library (deSolve)
seir_model = function (current_timepoint, state_values, parameters)
{
  # create state variables (local variables)
  S = state_values [1]        # susceptibles
  E = state_values [2]        # exposed
  I = state_values [3]        # infectious
  R = state_values [4]        # recovered

  with (
    as.list (parameters),     # variable names within parameters can be used
    {
      # compute derivatives
      dS = (-beta * S * I)
      dE = (beta * S * I) - (delta * E)
      dI = (delta * E) - (gamma * I)
      dR = (gamma * I)

      # combine results
      results = c (dS, dE, dI, dR)
      list (results)
    }
  )
}


contact_rate = 5                      # number of contacts per day
transmission_probability = 0.20       # transmission probability
infectious_period = 14                # infectious period
latent_period = 7                     # latent period

beta_value = contact_rate * transmission_probability
gamma_value = 1 / infectious_period
delta_value = 1 / latent_period

Ro = beta_value / gamma_value
parameter_list = c (beta = beta_value, gamma = gamma_value, delta = delta_value)


W = 1650      # susceptible hosts
X = 0.00020         # infectious hosts
Y = 0.00003         # recovered hosts
Z = 0.39         # exposed hosts
 N = W + X + Y + Z
```

```
   initial_values = c (S = W/N, E = Z/N, I = X/N, R = Y/N)

timepoints = seq (0, 120, by=1)
output = lsoda (initial_values, timepoints, seir_model, parameter_list)

# susceptible hosts over time
plot (S ~ time, data = output, type='l', ylim = c(0,1), col = 'blue', ylab = 'S, E, I, R', main = 'COVID19 SEIR Epidemic
                Simulation of Bangladesh')
# remain on same frame
par (new = TRUE)
# exposed hosts over time
plot (E ~ time, data = output, type='l', ylim = c(0,1), col = 'gray', ylab = '', axes = FALSE)
# remain on same frame
par (new = TRUE)
# infectious hosts over time
plot (I ~ time, data = output, type='l', ylim = c(0,1), col = 'red', ylab = '', axes = FALSE)
# remain on same frame
par (new = TRUE)
# recovered hosts over time
plot (R ~ time, data = output, type='l', ylim = c(0,1), col = 'green', ylab = '', axes = FALSE)
par (new = TRUE)
grid()
legend(100,0.7,legend=c("S","E","I","R"),col=c("blue","gray","red","green"), lty=1, cex=0.6)


time=output[,1]
s1=output[,2]*N
e1=output[,3]*N
i1=output[,4]*N
r1=output[,5]*N

transformed=cbind.data.frame(s1,e1,i1,r1)

#############
plot(time, s1, type = "l", frame = TRUE, pch = 19,axes = TRUE, col = "darkblue", xlab = "Time", ylab = "Person in
                Lac", main = "COVID 19 Epidimic Simulation of Bangladesh")
grid()
# Add a second line
lines(time, e1, pch = 22, col = "gray", type = "l")
# Add a second line
lines(time, i1, pch = 22, col = "red", type = "l")
# Add a second line
lines(time, r1, pch = 22, col = "darkgreen", type = "l")
legend("topright", legend=c("Susceptible", "Exposed","Infectious","Recovered"),col=c("darkblue",
                "gray","red","darkgreen"), lty = 1, cex=0.6)
```

4. R Code for Web App

```
  #
# This is a Shiny web application. You can run the application by clicking
# the 'Run App' button above.
#
# Find out more about building applications with Shiny here:
#
#    http://shiny.rstudio.com/
#
#install.packages('rsconnect')
library(DT)
library(rsconnect)
library(shiny)

# Define UI for application that draws a histogram
ui <- fluidPage(

  # Application title
  titlePanel("SEIR Epidimic Simulation [BETA Version]"),

  # Sidebar with a slider input for number of bins
  sidebarPanel(
    numericInput("pop", "Number of Susceptible Population", 1000),
    numericInput("exp", "Number of Exposed Person", 12),
    numericInput("inf", "Number of Infected Person", 2),
    numericInput("rec", "Number of Recovered Person", 1),
    numericInput("cnt", "Per Day Contact", 4),
    numericInput("pro", "Transmission Probability", 0.50),
    numericInput("ltn", "Latent Period", 14),
    numericInput("infp", "Infectious Period", 15),
    numericInput("days", "Number of Simulation Days", 120),
    tags$a(href="www.rstudio.com", "Source Code & Documentation1")
  ),




  # Show a plot of the generated distribution
  mainPanel(
    plotOutput("distPlot"),
    tabsetPanel(
      id = 'dataset',
      tabPanel("Predicted Values", DT::dataTableOutput("mytable1"))
    )


  )
```

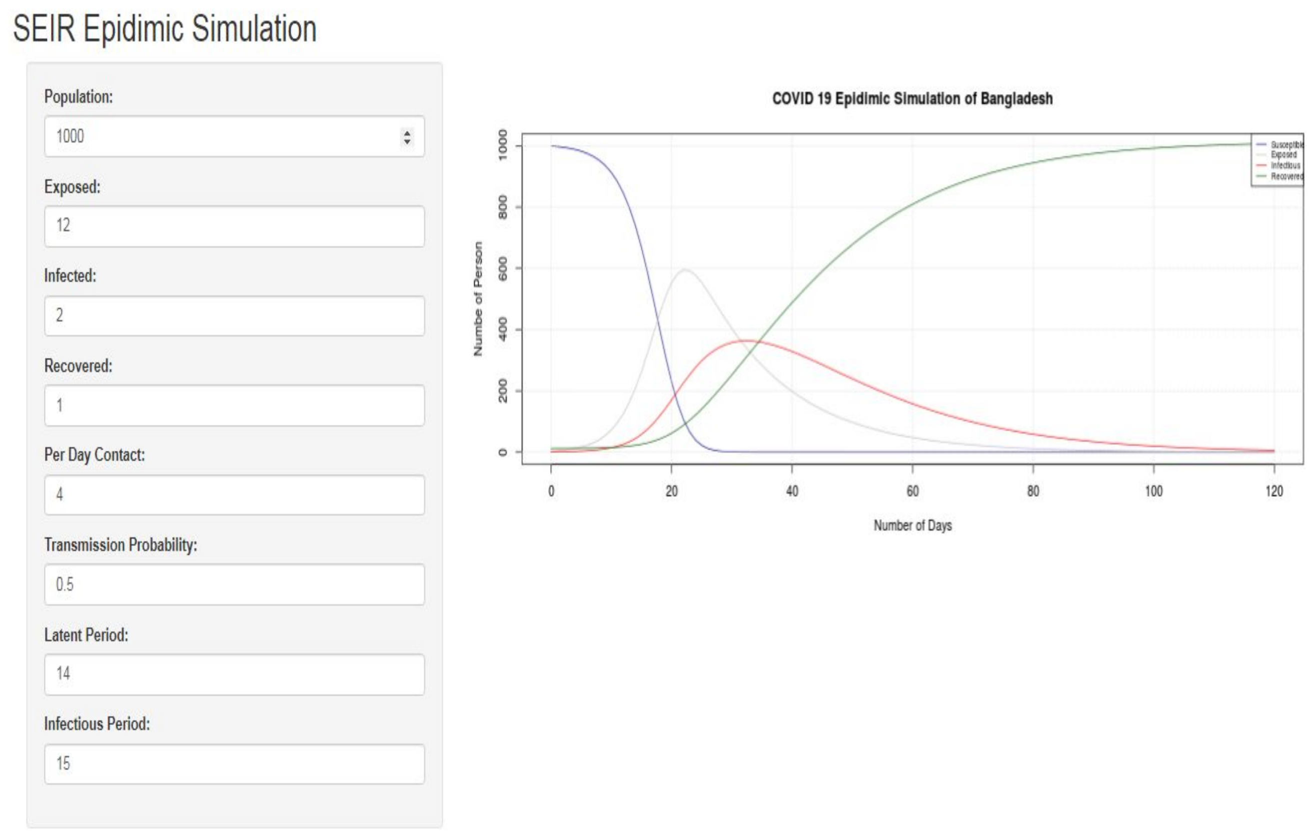
```
)


# Define server logic required to draw a histogram
server <- function(input, output) {

  output$distPlot <- renderPlot({
    # generate bins based on input$bins from ui.R

    x1    <- faithful[, 2]

    library (deSolve)
    seir_model = function (current_timepoint, state_values, parameters)
    {
      # create state variables (local variables)
      S = state_values [1]        # susceptibles
      E = state_values [2]        # exposed
      I = state_values [3]        # infectious
      R = state_values [4]        # recovered

      with (
        as.list (parameters),     # variable names within parameters can be used
        {
          # compute derivatives
          dS = (-beta * S * I)
          dE = (beta * S * I) - (delta * E)
          dI = (delta * E) - (gamma * I)
          dR = (gamma * I)

          # combine results
          results = c (dS, dE, dI, dR)
          list (results)
        }
      )
    }


    contact_rate = input$cnt                     # number of contacts per day
    transmission_probability = input$pro       # transmission probability
    infectious_period = input$infp                # infectious period
    latent_period = input$ltn                     # latent period

    beta_value = contact_rate * transmission_probability
    gamma_value = 1 / infectious_period
    delta_value = 1 / latent_period

    Ro = beta_value / gamma_value
    parameter_list = c (beta = beta_value, gamma = gamma_value, delta = delta_value)
    W = input$pop      # susceptible hosts
    X = input$inf        # infectious hosts
    Y = input$rec        # recovered hosts
    Z = input$exp        # exposed hosts

    N = W + X + Y + Z

    initial_values = c (S = W/N, E = Z/N, I = X/N, R = Y/N)


    d = input$days        # simulation periods
    timepoints = seq (0, d, by=1)
    output = lsoda (initial_values, timepoints, seir_model, parameter_list)


    time=output[,1]
    s1=output[,2]*N
    e1=output[,3]*N
    i1=output[,4]*N
    r1=output[,5]*N

    transformed=cbind.data.frame(s1,e1,i1,r1)

    #############
    plot(time, s1, type = "l", frame = TRUE, pch = 19,axes = TRUE, col = "darkblue", xlab = "Number of Days", ylab =
"Numbe of Person", main = "Epidimic Simulation Graph")
    grid()
    # Add a second line
    lines(time, e1, pch = 22, col = "gray", type = "l")
    # Add a second line
    lines(time, i1, pch = 22, col = "red", type = "l")
    # Add a second line
    lines(time, r1, pch = 22, col = "darkgreen", type = "l")
    legend("topright", legend=c("Susceptible", "Exposed","Infectious","Recovered"),col=c("darkblue",
"gray","red","darkgreen"), lty = 1, cex=0.75)

    ########################################################
    #bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    # hist(x, breaks = bins, col = 'darkgray', border = 'white')

  })

  ####################################
  output$mytable1 <- DT::renderDataTable({
```

```
seir_model = function (current_timepoint, state_values, parameters)
{
  # create state variables (local variables)
  S = state_values [1]        # susceptibles
  E = state_values [2]        # exposed
  I = state_values [3]        # infectious
  R = state_values [4]        # recovered

  with (
    as.list (parameters),     # variable names within parameters can be used
    {
      # compute derivatives
      dS = (-beta * S * I)
      dE = (beta * S * I) - (delta * E)
      dI = (delta * E) - (gamma * I)
      dR = (gamma * I)

      # combine results
      results = c (dS, dE, dI, dR)
      list (results)
    }
  )
}


contact_rate = input$cnt                    # number of contacts per day
transmission_probability = input$pro       # transmission probability
infectious_period = input$infp                # infectious period
latent_period = input$ltn                     # latent period

beta_value = contact_rate * transmission_probability
gamma_value = 1 / infectious_period
delta_value = 1 / latent_period

Ro = beta_value / gamma_value
parameter_list = c (beta = beta_value, gamma = gamma_value, delta = delta_value)


W = input$pop      # susceptible hosts
X = input$inf        # infectious hosts
Y = input$rec         # recovered hosts
Z = input$exp         # exposed hosts
N = W + X + Y + Z

initial_values = c (S = W/N, E = X/N, I = Y/N, R = Z/N)
```

```
    d = input$days         # simulation periods
    timepoints = seq (0, d, by=1)
    output = lsoda (initial_values, timepoints, seir_model, parameter_list)


    DAYS=output[,1]
    SUSCEPTIBLE=output[,2]*N
    EXPOSED=output[,3]*N
    INFECTED=output[,4]*N
    RECOVERED=output[,5]*N

    transformed2=cbind.data.frame(DAYS,SUSCEPTIBLE,EXPOSED,INFECTED,RECOVERED)
    #tra=cbind.data.frame(a,b)
    DT::datatable(transformed2)
  })
}

# Run the application
shinyApp(ui = ui, server = server)
```

5. WEB Application

5.1 WEB Application Interface



SEIR Epidimic Simulation

Population: 1000

Exposed: 12

Infected: 2

Recovered: 1

Per Day Contact: 4

Transmission Probability: 0.5

Latent Period: 14

Infectious Period: 15

5.2 URL - https://fazlyrabbikhan.shinyapps.io/SERIsim/

5.3 R Shiny App

Shiny is an R package that makes it easy to build interactive web apps straight from R. We can host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. We can also extend your Shiny apps with CSS themes, html widgets, and JavaScript actions.

We used R Shiny App Technology to make web interface of this program and hosted into their cloud for internet access.

References

1. Influenza SERI Model in R- https://rstudio-pubs-static.s3.amazonaws.com/108550_a60980210a6f4f41a49a376e82ba7bbf.html
2. deSolve R Package use guideline-https://cran.r-project.org/web/packages/deSolve/deSolve.pdf
3. deSolve R Package Documentation- https://www.rdocumentation.org/packages/deSolve/versions/1.27.1
4. WHO Modeling & Other References (How COVID-19 and Other Infectious Diseases Spread: Mathematical Modeling)- https://triplebyte.com/blog/modeling-infectious-diseases
5. SERI Descriptions- https://www.idmod.org/docs/hiv/model-seir.html#seir-model
6. Model Video- https://www.youtube.com/watch?v=Mrx-kSjOHAY