Supplementary details on experiments

Olivia

2023-09-17

Literature review (TBRemoved)

EpiEstim

They considered two scenarios: constant $R_t = 2.5$ and piecewise-constant $R_t = 2.5, 0.7$ before and after day $t = 15$ to illustrate the effect of control measure. They simulated 100 epidemics with 50 days.
They used a SARS like serial interval distribution with mean 8.4 days and standard deviation of 3.8 days.
They compared the estimated effective Rt with the case Rt by the WT method.

APEestim (excluded since its advantage is prediction)

They considered four scenarios: constant Rt, periodically varying changes in Rt, exponentially rising and decaying Rt, and piecewise falling Rt due to various interventions. They demonstrated the Rt estimation and one-step-ahead prediction of incidence using APEestim.

EpiFilter

EpiFilter run three sets of experiments for low incidence, multiple waves, and empirical epidemics. They compared the Rt estimates and one-step-ahead incidence prediction with the corresponding credible intervals. All simulated epidemics are of length t=300. They compared *EpiFilter* against two competitors *EpiEstim* and *APEestim*, where *APEestim* optimized the sliding window in terms of one-step-ahead prediction. (The optimal parameters do rely on the objective of the study — understanding (Rt estimation) of the change of the epidemic or predicting the ongoing incidence cases). They used the serial interval of Ebola virus.

[The focus here is to illustrate estimators have robust estimation during low incidence cases which are difficult tasks for Rt estimation.] The low incidence (small or waning) epidemic curves that they considered were rapidly controlled epidemics (piecewise constant Rt from 2 to .5 at t=100), small outbreaks with exponentially rising and falling epidemics (cont. Rt starting from 2 with exp. rate 0.02 and -0.008 before and after t=30), and medium outbreaks that are initially controlled then resurge into large epidemics and then are suppressed (cont. piecewise-linear Rt from 4 to .6 at t=40, and to 2 at t=80, and then to .2 at t=150). They simulate 200 epidemics per case (either with low incidences or long tails of zeros) using the renewal model. In graphical demonstration, they plotted representative examples of the true case and three estimates for Rt estimation and prediction for each scenario, and density curves of MSE and one-step-ahead RMSE for Rt estimates and incidence prediction respectively for all estimators per scenario.

[The focus here is to demonstrate the estimators can predict a preceding peak from a current trough to provide evidence for interventions.] The multiple-wave epidemics are initially controlled but then resurged epidemics (Rt from 2.5 to 0.5 then to 2.5 at t=70 then t=230), periodic or seasonal transmission (sinusoidal Rt with magnitude 1.3 ± 1.2 and period of 120 time units), and then outbreaks with exponential rising and falling (exponent rates 0.03, -0.015, 0.02 changing at t=40, then t=190). These curves are of multiple peaks and troughs.

A unique type of figures they drew is the MSE/PMSE trajectories throughout all runs.

In summary, *EpiFilter* outperformed the other competitors in the accuracy of Rt estimation and was comparative to *APEestim* and outperformed *EpiEstim* in the accuracy of prediction.

EpiLPS

They compared EpiLPS with EpiEstim with three sliding windows (default: weekly, daily, three days). They assumed the incidence data to follow **negative binomial** distribution with certain overdispersion levels. They considered three serial interval distributions of influenza, SARS-CoV-1 and MERS-CoV like serial intervals.

Experimental design

Synthetic reproduction numbers

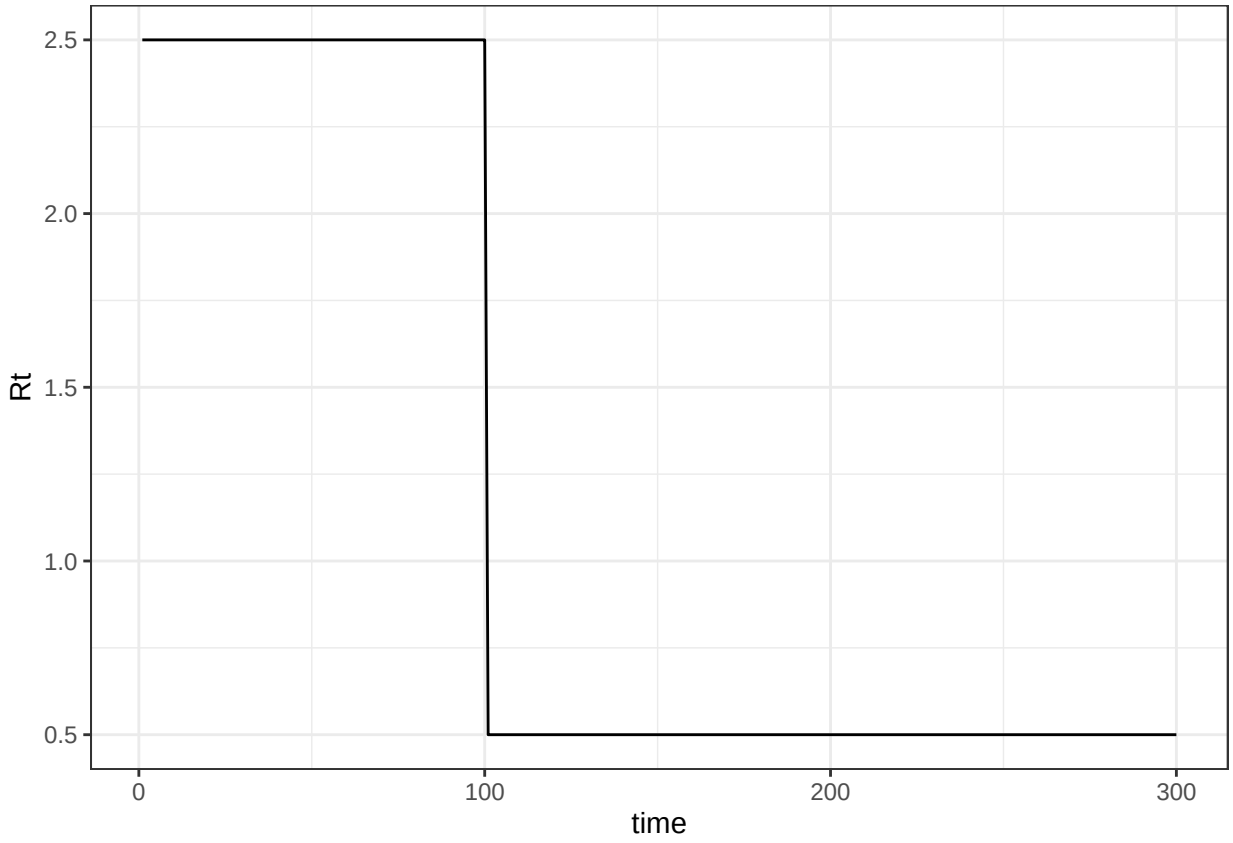
Overall, we argue our estimator is accurate, robust in model misspecification and computationally efficient. We can do a series of tests for each property. We may consider the following curvature of efficient reproduction numbers, and test the accuracy of our estimators compared to

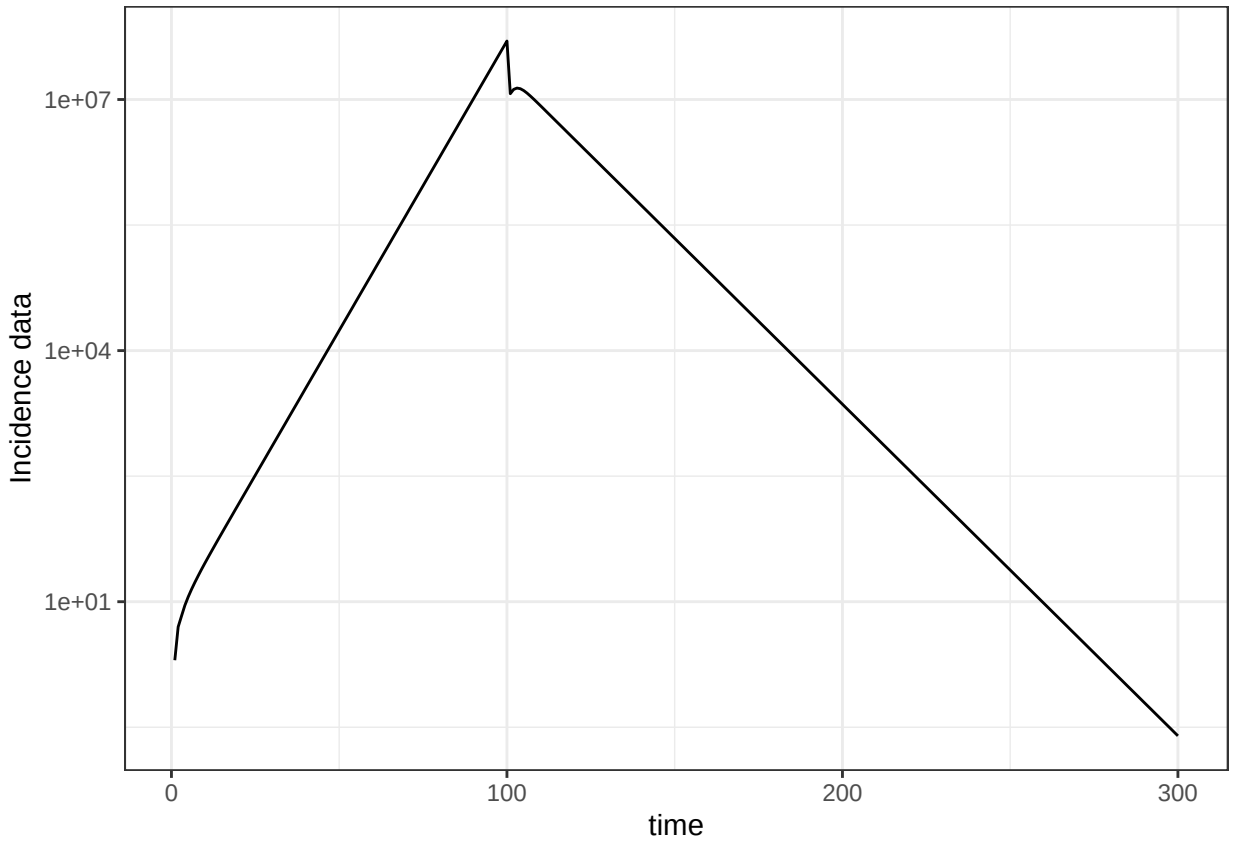
We may consider arbitrary reproduction numbers in a few scenarios: a) piecewise-constant epidemics with a drop at a certain time point to measure the effect of control measures, b) exponentially rising and falling epidemics with a change point, c) piecewise-constant with multiple segments to measure the initially controlled and resurged and the suppressed epidemics, d) periodic waves, and e) exponential rising and falling and then resurging. We may simulate the epidemics (with length T=300) 10 times for each scenario, estimate Rt, and compute the MSE. (Maybe do more replicates later.) (A few questions TBD. 1. Smoothed or non-smoothed piecewise-constant Rt? Is the smoothness assumption for instantaneous Rt even valid? 2. Low incidence v.s. Medium or large? *EpiFilter* make a) and b) to be low incidence cases and c) to be medium at the beginning and then large outbreaks. We may consider a) and b) to be simple cases with only one change point and c)-e) to be more complex cases with multiple waves. 3. Time comparisons for long sequences, say T=500? 4. Measures: RMSE? No. Do KL or log(lambda2/ lambda1).)

We may consider varying serial interval for different scenarios of Rt curvatures. For example, we may use influenza serial interval for periodic epidemics and Covid serial interval for epidemics with more non-periodic moves.

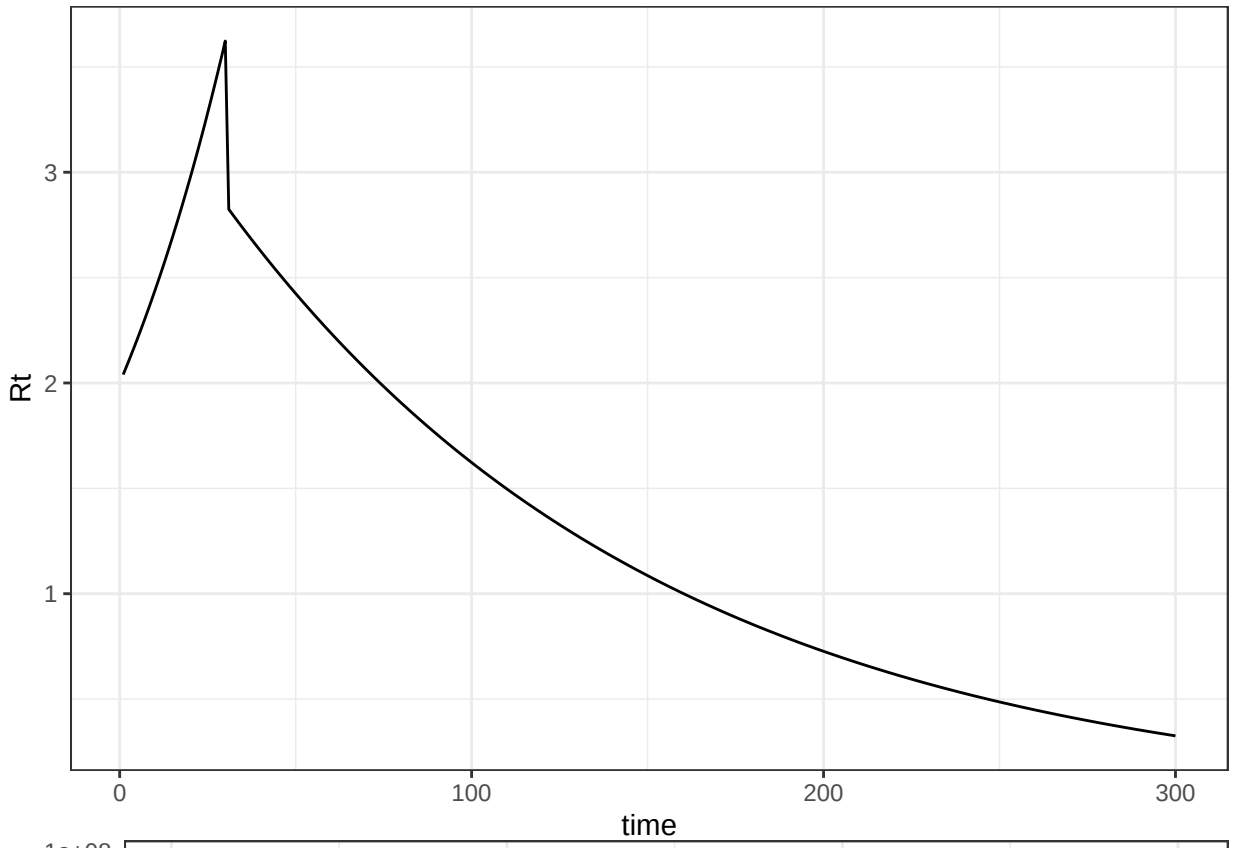
```r
# General settings:
N1 = 2 # first incidence data
length = 300 # number of evenly spaced time points
library(rtestim)
## Get incidence:
get_incidence <- function(N1, Rt, gamma_pars = c(2.5, 2.5)){
  len <- length(Rt)
  incidence <- numeric(len)
  incidence[1] <- N1
  for(t in 2:len){
    pi <- discretize_gamma(1:(t-1), gamma_pars[1], gamma_pars[2])
    incidence[t] <- Rt[t] * sum(rev(pi) * incidence[1:(t-1)])
    # alternative:
    # incidence[t] <- delay_calculator(incidence[1:t], dist_gamma = gamma_pars)[t] * Rt[t]

  }
  return(incidence)
}
# Check the synthetic data:
library(ggplot2)
display_dat <- function(incidence, Rt){
```

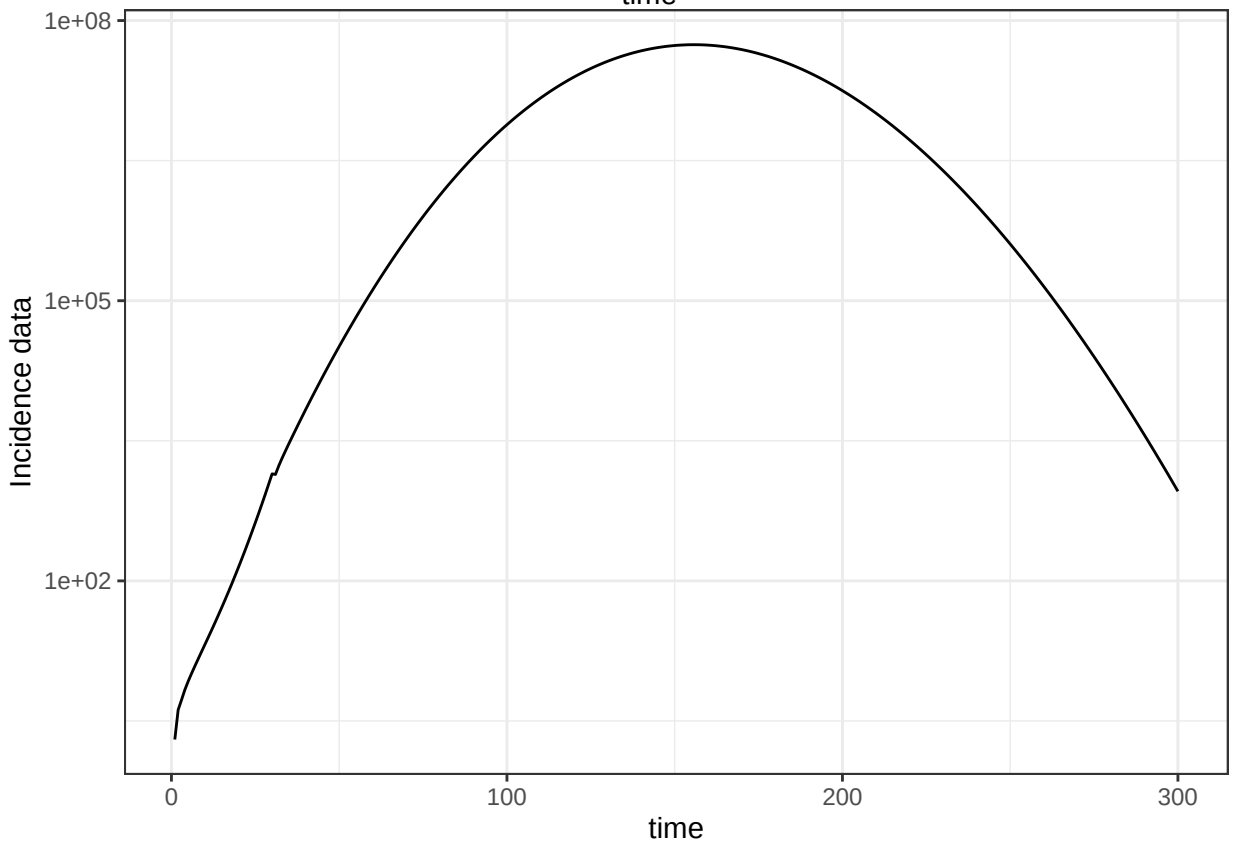
```
  len <- length(Rt)
  if(length(incidence) != length(Rt)) cli::cli_abort("Number of `incidence` does not match number of `Rt
  dat <- data.frame(time = 1:len, count = incidence, Rt = Rt)
  fig1 <- dat %>%
    ggplot(aes(y = Rt, x = time)) +
    geom_line() +
    theme_bw()
  print(fig1)
  fig2 <- dat %>%
    ggplot(aes(y = count, x = time)) +
    geom_line() +
    scale_y_log10() + # axis of incidence in log scale
    ylab("Incidence data") +
    theme_bw()
  print(fig2)
}

# Scenario 1: two-stage piecewise constant with one dropping point (similar as in EpiFilter)
Rt1 <- c(rep(2.5, 100), rep(0.5, length-100)) # arbitrary sequence of Rt
gamma_pars1 <- c(2.5, 2.5) # serial interval distribution parameters
incidence1 <- get_incidence(N1, Rt1, gamma_pars1)
display_dat(incidence1, Rt1)
```






```r
# Scenario 2: two-stage exponential growth and decay (similar as in EpiFilter)
rate <- c(.02, -.008)
Rt2 <- numeric(length)
Rt2[1:30] <- (1 + rate[1]) ^ (1:30) * 2
Rt2[31:300] <- (1 + rate[2]) ^ (31:300) * Rt2[30]
gamma_pars2 <- c(2.5, 2.5) # serial interval distribution parameters
incidence2 <- get_incidence(N1, Rt2, gamma_pars2)
display_dat(incidence2, Rt2)
```


Rt
3
2
1
0
100
200
300
time

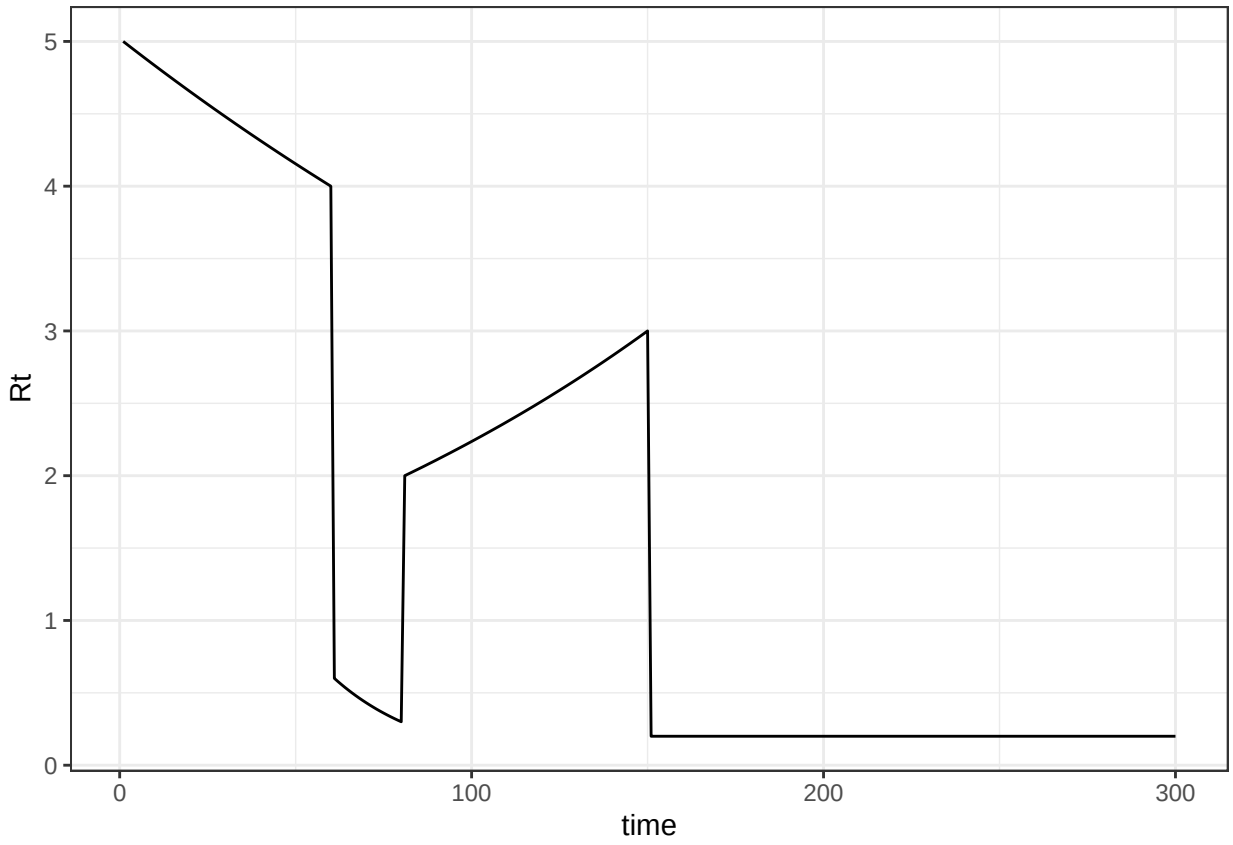

Incidence data
1e+08
1e+05
1e+02
0
100
200
300
time

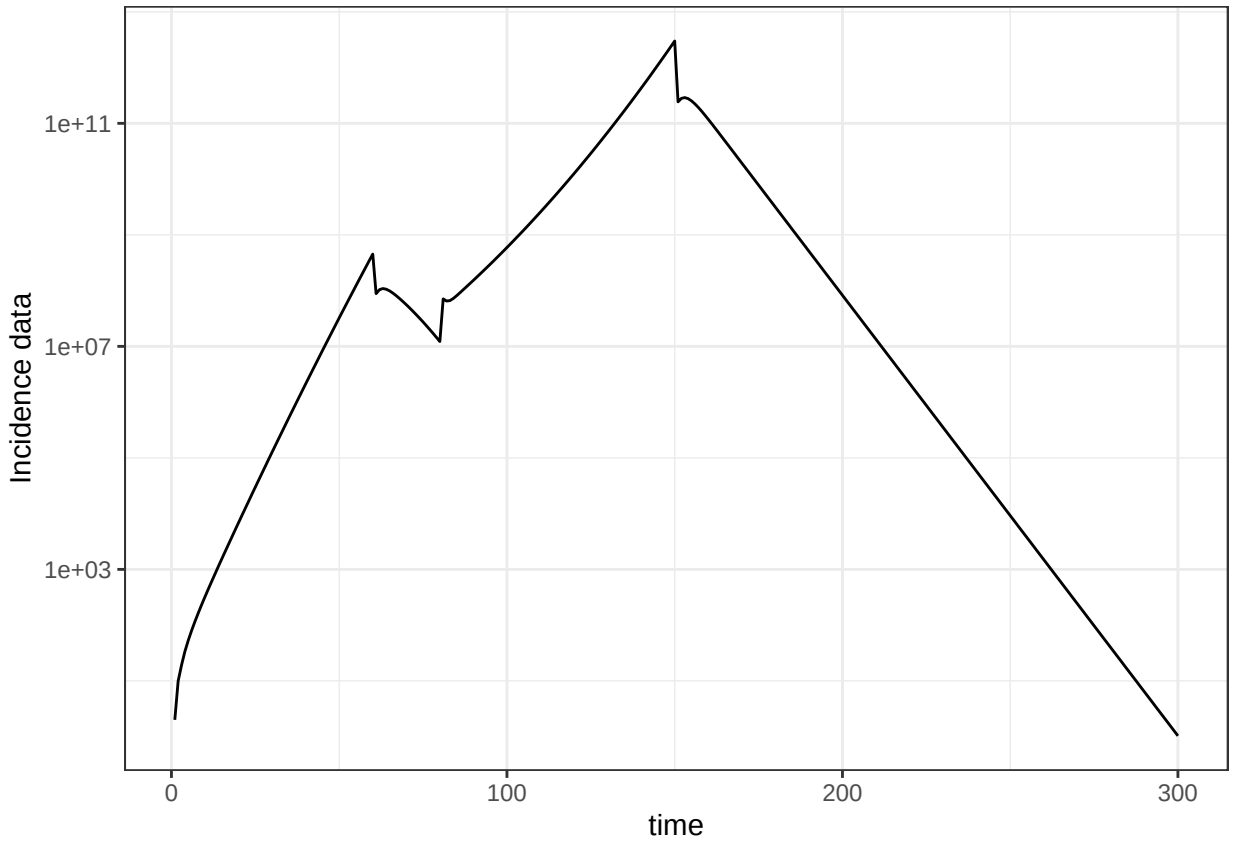
```r
# Scenario 3: multi-stage piecewise constant (similar as in EpiFilter)
library(pracma)
```

```
##
## Attaching package: 'pracma'
```

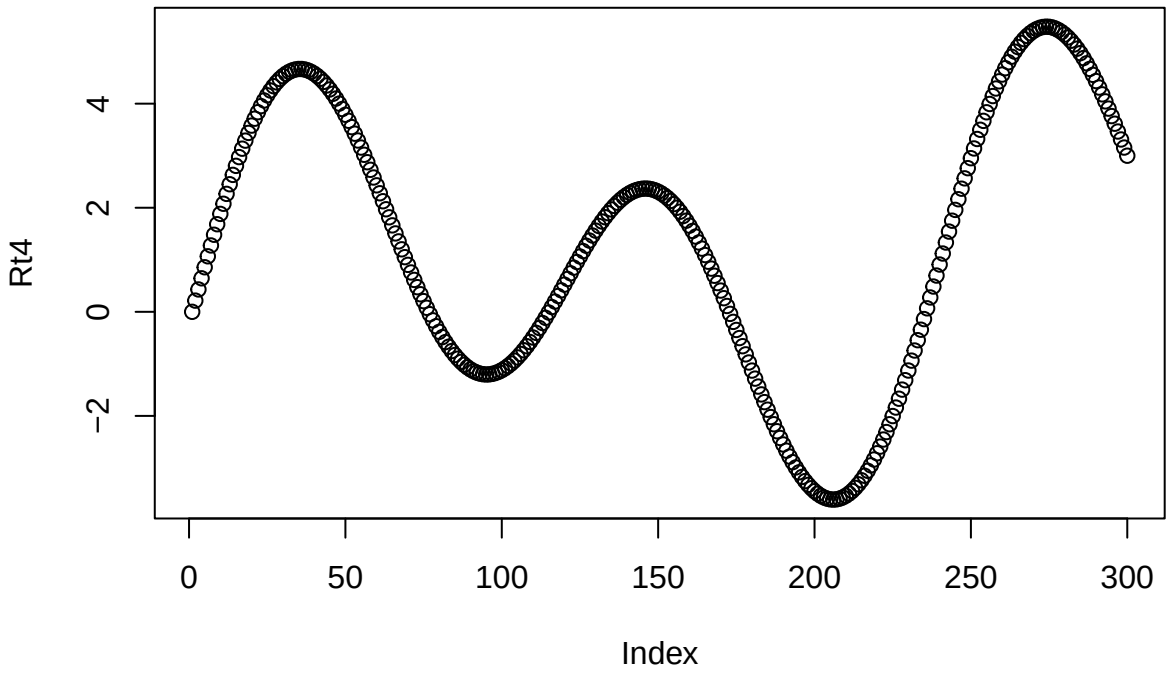
```
## The following objects are masked from 'package:Matrix':
##
##     expm, lu, tril, triu
```

```r
Rt3 <- c(logseq(5, 4, n=60), logseq(0.6, 0.3, 20), logseq(2, 3, 70), rep(0.2, 150))
gamma_pars3 <- c(2.5, 2.5) # serial interval distribution parameters
incidence3 <- get_incidence(N1, Rt3, gamma_pars3)
display_dat(incidence3, Rt3)
```

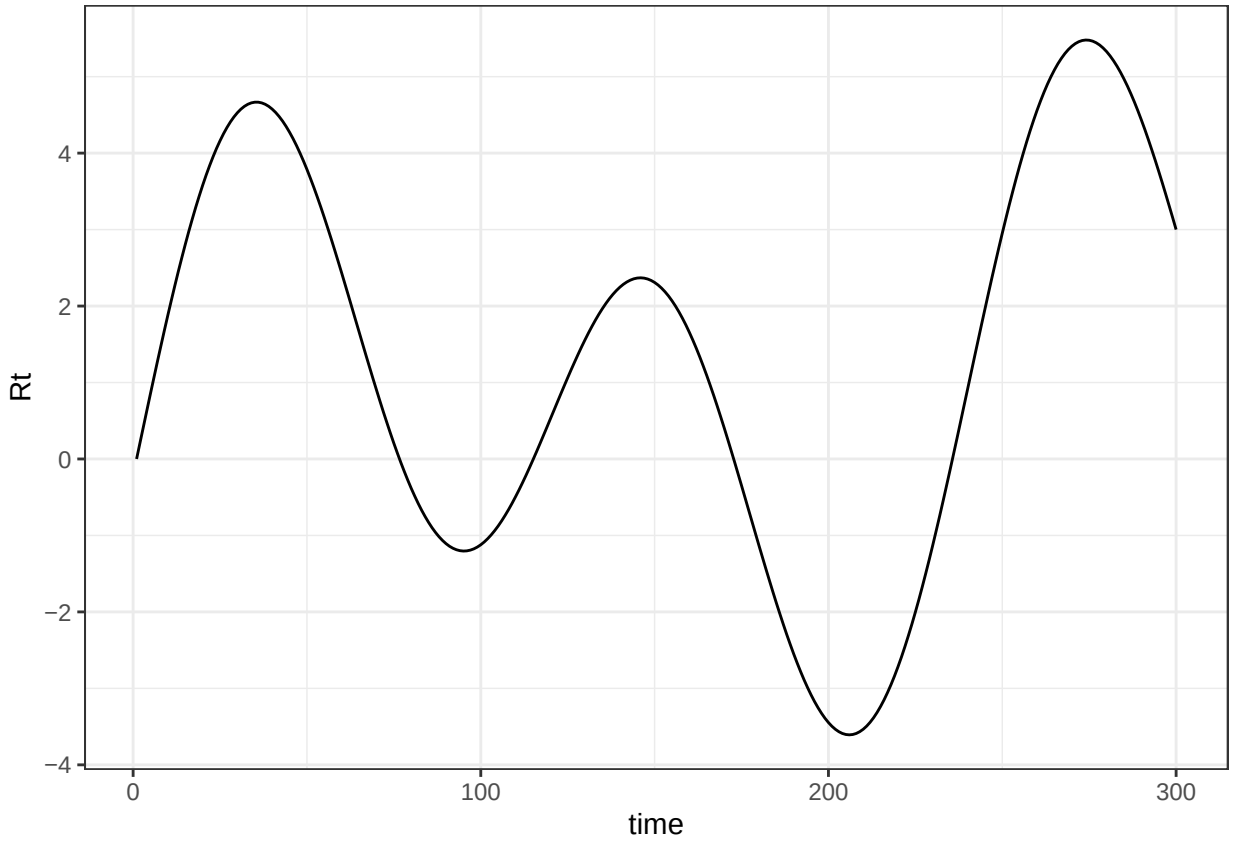




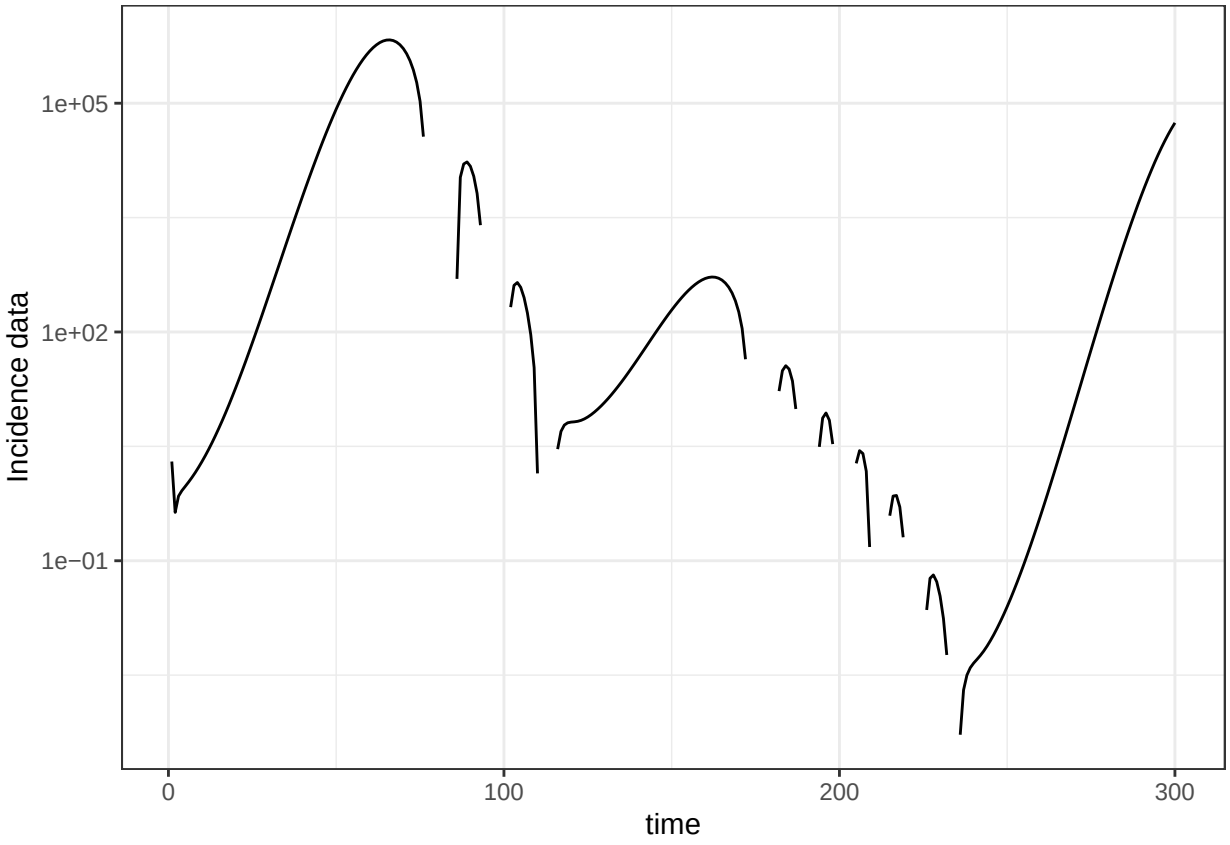

```r
# Scenario 4: multi-stage with exponential-sinusoidal growth and decay (cited the function from EpiLPS)
x <- seq(0, 10, length.out = 300)
Rt4 <- numeric(300)
components <- list(
  list(freq = 0.1, amp = 1),
  list(freq = 0.5, amp = 2),
  list(freq = 1.0, amp = 3)
)
for (component in components) {
  Rt4 <- Rt4 + component$amp * sin(pi * component$freq * x/2)
}
plot(Rt4)
```




```
#Rt4 <- 1:300
#Rt4 <- sapply(Rt4, function(t) {0.5 * (exp(sin(pi * t / 36)) + 1.5 * exp(cos(4 / t)) )})
gamma_pars4 <- c(2.5, 2.5) # serial interval distribution parameters
incidence4 <- get_incidence(N1, Rt4, gamma_pars4)
display_dat(incidence4, Rt4)
```



```
## Warning in self$trans$transform(x): NaNs produced
## Warning: Transformation introduced infinite values in continuous y-axis
```



Reproduction numbers using SIR

We generate the real-world transmission mechanism using a simple compartmental model, SIR, and compute the reproduction numbers by assuming fixed transmission parameters.

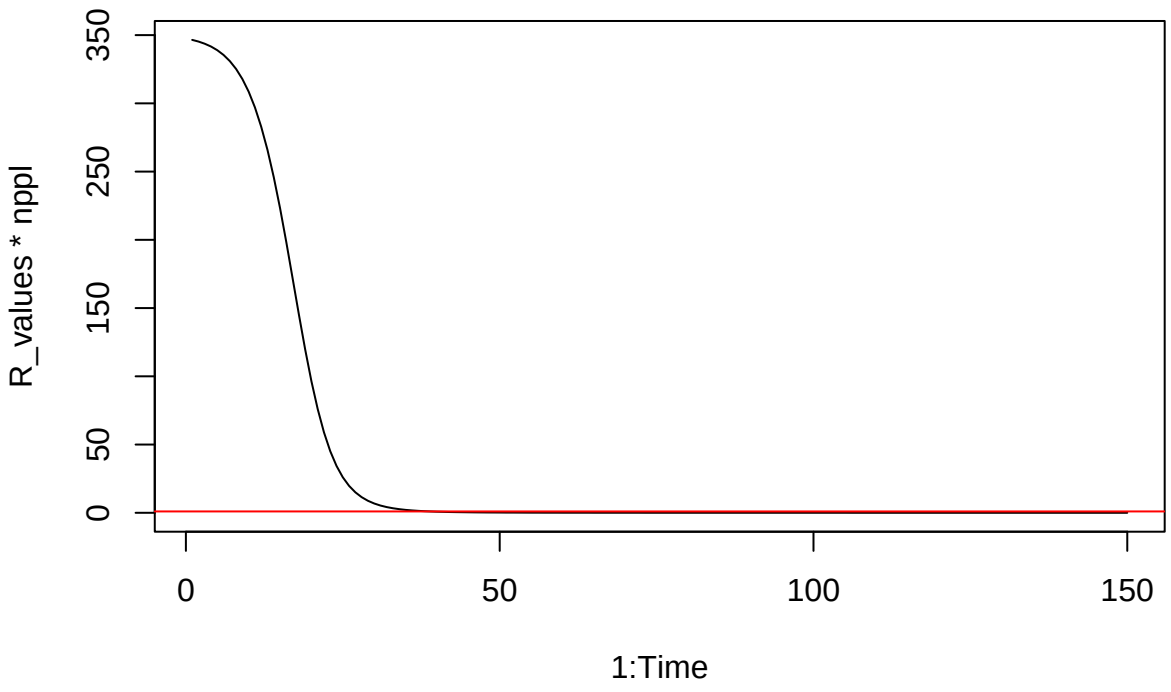
This model assumes a closed and well-mixed population with size N, a constant contact rate β and removal rate γ. The population is divided into three compartments, susceptible, infectious and removed (recovered) compartments, denoted by S, I, R respectively. We start with the initial proportions $S = 0.99, I = 0.01, R = 0.0$ with sum 1 and fix $\beta = 0.35, \gamma = 0.035$. Generate the compartmental proportions for the following $Time = 100$ time points and compute the reproduction number for each time point. The reproduction numbers are plotted in the figure below.

The basic reproduction number is widely used to be $\mathcal{R}_0 = \beta/\gamma$, but the effective reproduction number varies by time. By definition, the effective reproduction number is the number of secondary infection caused by a primary infection at a specific time. At time t, $S_{t-1} - S_t$ individuals are infected and moved from the susceptible compartment to the infectious compartment. There are I_{t-1} primary individuals at time $t-1$. Thus, the reproduction number at time t is $\mathcal{R}_t = (S_{t-1} - S_t)/I_{t-1}$.

Figure for one single set of parameters.

```
# Parameters
beta = 0.35  # Transmission rate
gamma = 0.035 # Recovery rate

S0 = 0.99  # Initial proportion of susceptible individuals
I0 = 0.01  # Initial proportion of infectious individuals
```

```r
R0 = 0.0  # Initial proportion of recovered individuals
Time = 150  # Total number of time points

# Store S, I, and R values
S = numeric(Time + 1); S[1] = S0
I = numeric(Time + 1); I[1] = I0
R = numeric(Time + 1); R[1] = R0
R_values = numeric(Time)

# Simulate the SIR model
for(t in 1:Time){
    dS = -beta * S[t] * I[t]
    dI = (beta * S[t] * I[t]) - (gamma * I[t])
    dR = gamma * I[t]

    S[t+1] = S[t] + dS
    I[t+1] = I[t] + dI
    R[t+1] = R[t] + dR

    # Calculate R for each time point
    R_values[t] = -dS / I[t]
}

# Set a number of population
nppl = 1000
# Plot R over time
plot(R_values*nppl, x=1:Time, type="l")
abline(h = 1, col="red")
```




```r
plot((1-S)*nppl, col="red", type="l") # cumulative infection proportions
lines(R*nppl, col="orange")
lines(I*nppl, col="blue")
```

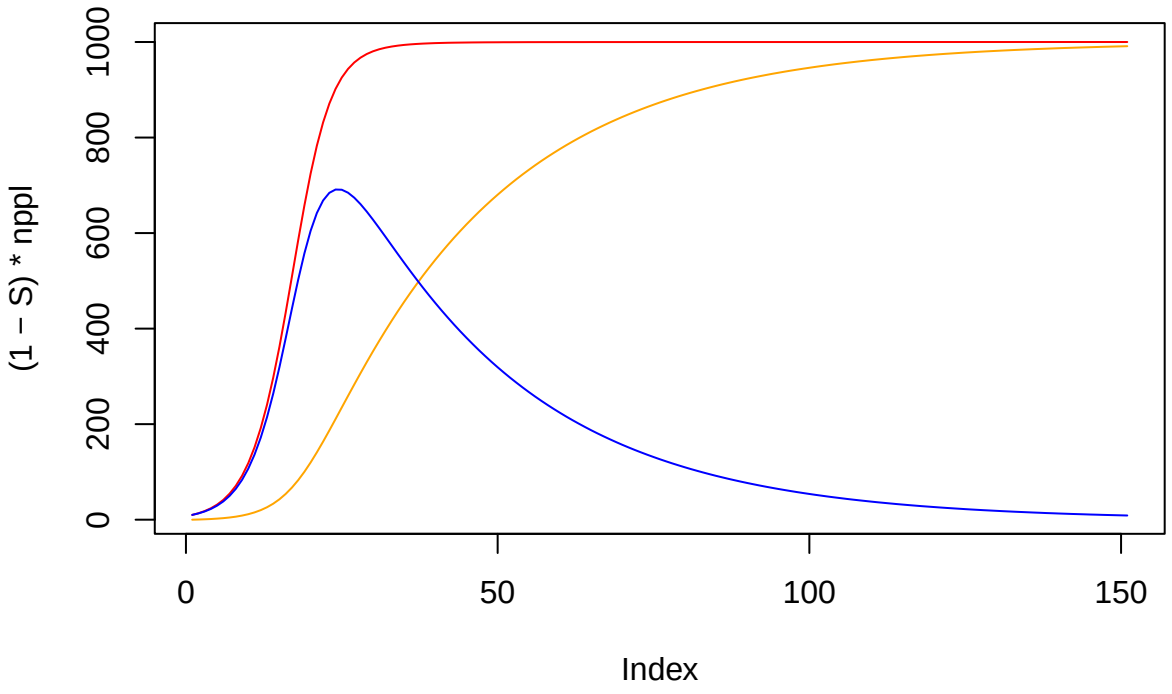


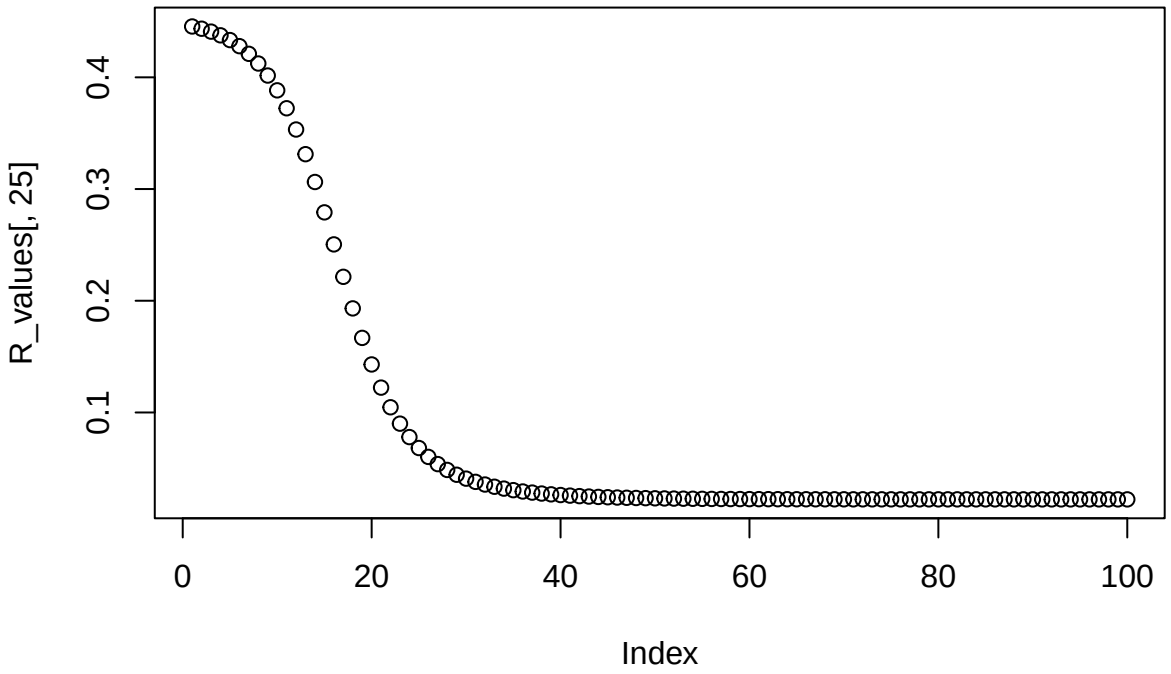
Figure for multiple sets of parameters.

```
# Parameters
beta = seq(0.35, 0.95, 0.05)  # Transmission rate
gamma = seq(0.05, 0.55, 0.05)  # Recovery rate
para_table <- data.table::CJ(beta = beta, gamma = gamma)
npar <- nrow(para_table)
print(para_table)
```

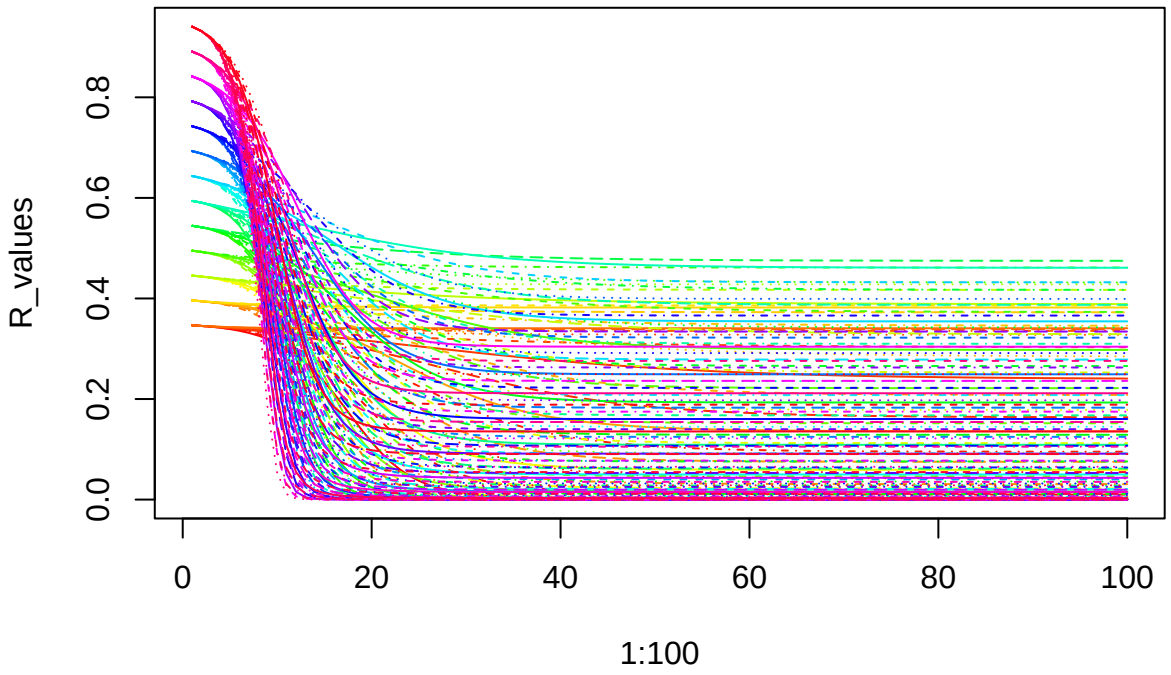
```
##       beta gamma
##    1: 0.35  0.05
##    2: 0.35  0.10
##    3: 0.35  0.15
##    4: 0.35  0.20
##    5: 0.35  0.25
##   ---
## 139: 0.95  0.35
## 140: 0.95  0.40
## 141: 0.95  0.45
## 142: 0.95  0.50
## 143: 0.95  0.55
```

```
S0 = 0.99  # Initial proportion of susceptible individuals
I0 = 0.01  # Initial proportion of infectious individuals
R0 = 0.0  # Initial proportion of recovered individuals
Time = 100  # Total number of time points

# Initialize arrays to store S, I, and R values
S = numeric(Time + 1); S[1] = S0
I = numeric(Time + 1); I[1] = I0
R = numeric(Time + 1); R[1] = R0
R_values = matrix(nrow = Time, ncol = npar)
```

```r
# Simulate the SIR model
for(l in 1:npar){
  for(t in 1:Time){
    dS = -para_table$beta[l] * S[t] * I[t]
    dI = (para_table$beta[l] * S[t] * I[t]) - (para_table$gamma[l] * I[t])
    dR = para_table$gamma[l] * I[t]

    S[t+1] = S[t] + dS
    I[t+1] = I[t] + dI
    R[t+1] = R[t] + dR

    # Calculate R for each time point
    R_values[t,l] = -dS / I[t]
  }
}
colnames(R_values) <- paste("Para",1:npar,sep="")

plot(R_values[,25])
```




```r
library(RColorBrewer)
palette <- rainbow(ncol(R_values))
matplot(1:100, R_values, type="l", col = palette)
```




```
#legend("topright", legend = colnames(R_values), fill = palette, title = "Legend", cex=.3)
```

An alternative approach to generate reproduction numbers is to assume specific graphical smoothness, e.g., piecewise-constant curve.