Supplementary details on experiments

Olivia

2023-09-19

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

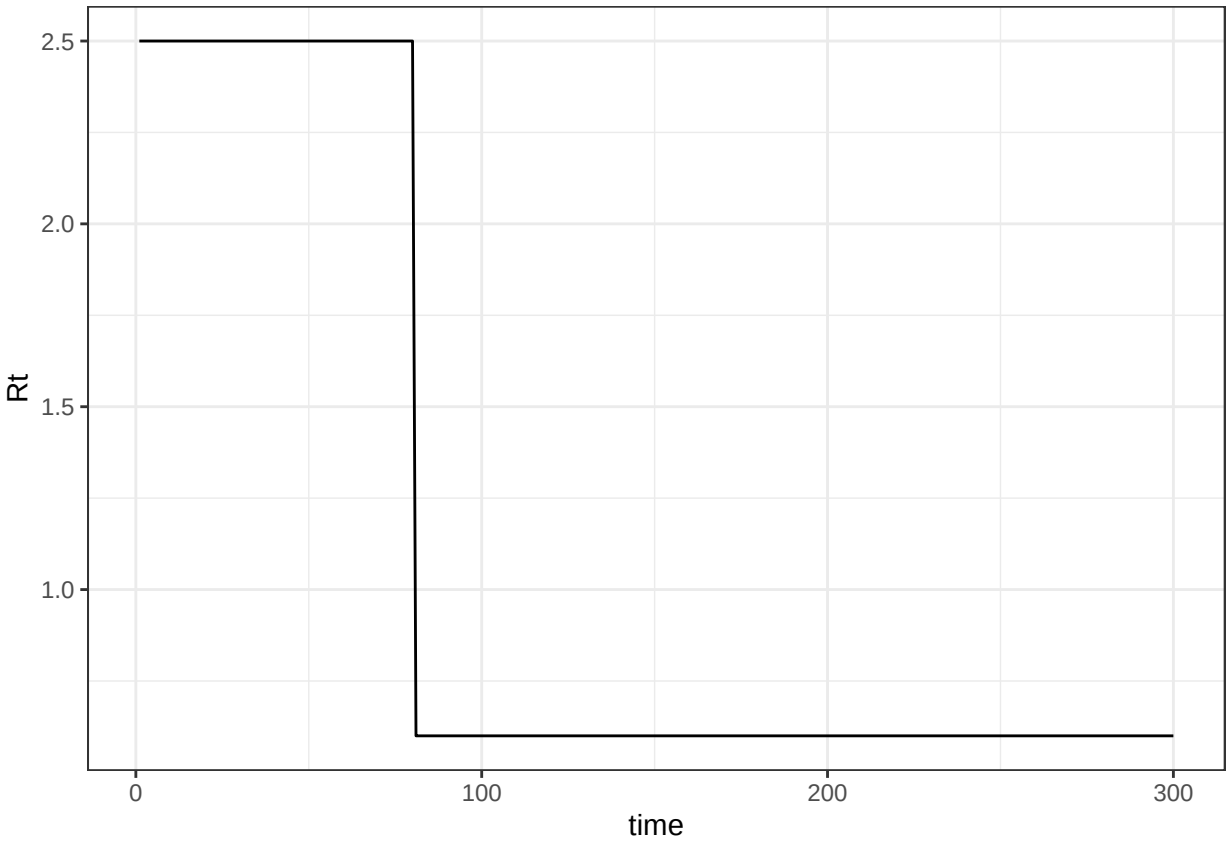
We may consider arbitrary reproduction numbers in a few scenarios: a) piecewise-constant epidemics with a drop at a certain time point to measure the effect of control measures, b) exponentially rising and falling epidemics with a change point, c) piecewise-constant with multiple segments to measure 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.

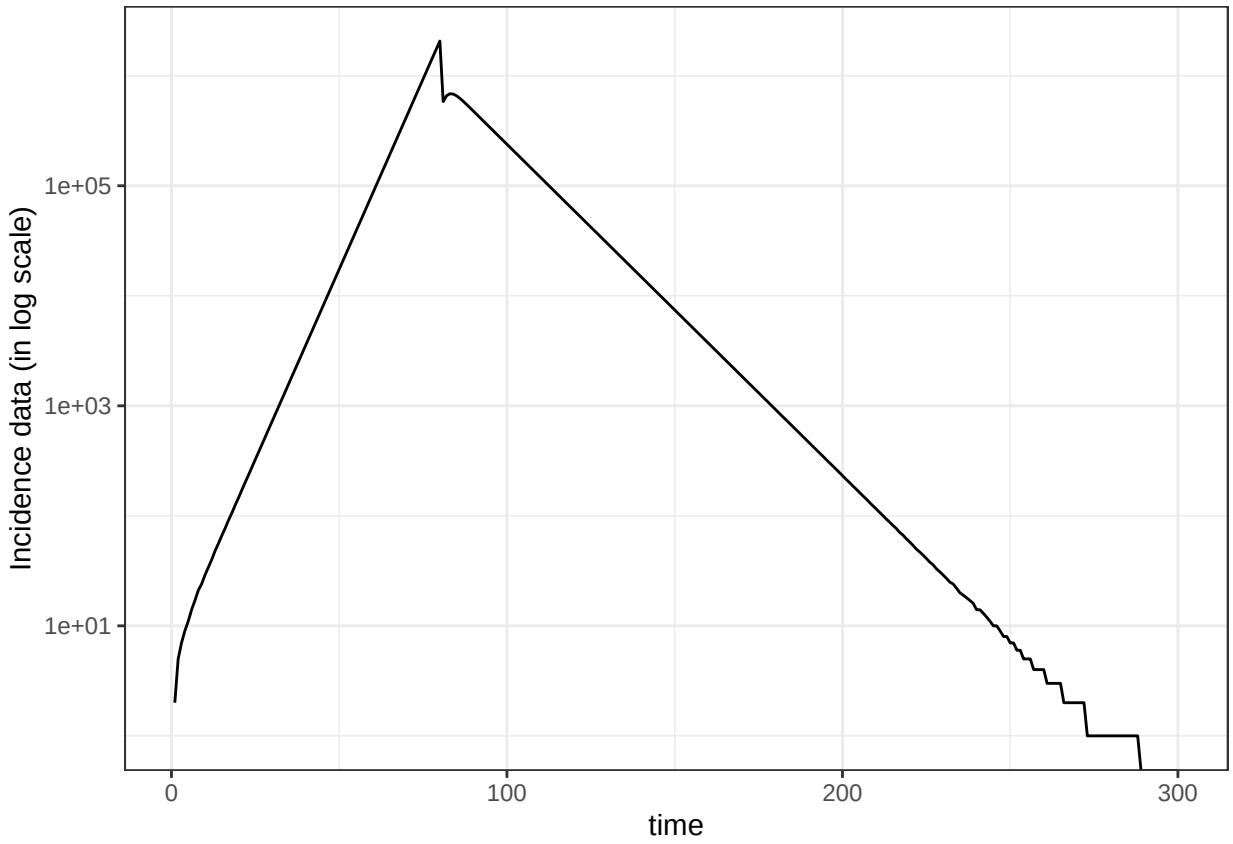
```r
# General settings:
N1 = 2 # first incidence data
len = 300 # number of evenly spaced time points
library(rtestim)
# Get incidence:
get_incidence <- function(N1, Rt, gamma_pars = c(2.5, 2.5)){
  len <- length(Rt)
  incidence <- numeric(len)
  incidence[1] <- N1
  for(t in 2:len){
    pi <- discretize_gamma(1:(t-1), 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)
}
# Display the synthetic data:
library(ggplot2)
display_dat <- function(incidence, Rt, counts = NULL){
  len <- length(Rt)
```

```r
  if(is.null(counts)){
    if(length(incidence) != length(Rt)) cli::cli_abort("Data lengths do not match.")
  } else{
    if(length(incidence) != length(Rt) || length(incidence) != length(counts) || length(counts) != leng
  }
  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 (in log scale)") +
    theme_bw()
  print(fig2)
  if(!is.null(counts)){
    dat <- data.frame(time = 1:len, count = counts, mean = incidence)
    fig3 <- dat %>%
    ggplot(aes(y = count, x = time)) +
    geom_point() +
    geom_line(aes(y = mean), col="orange") +
    #scale_y_log10() + # axis of incidence in log scale
    ylab("Observed indicence counts") +
    theme_bw()
  print(fig3)
  }
}
# Check data quality:
check_dat <- function(incidence, Rt){
  if(min(Rt) < 0) {cli::cli_abort("`Rt` must be non-negative.")}
  if(min(incidence) < 0) {cli::cli_abort("`incidence` cases must be ")}
  if(max(incidence) > 1e7L) {cli::cli_alert_warning("`incidence` cases are too large given reality.")}
  if(sum(incidence == 0) > 30) {cli::cli_warn("`incidence` data has more than 10% 0s.")}
}

# Scenario 1: two-stage piecewise constant with one dropping point (similar as in EpiFilter)
Rt1 <- c(rep(2.5, 80), rep(0.6, len-80)) # arbitrary sequence of Rt
gamma_pars1 <- c(2.5, 2.5) # serial interval distribution parameters
incidence1 <- get_incidence(N1, Rt1, gamma_pars1)
incidence1 <- round(incidence1, 0)
display_dat(incidence1, Rt1)
```

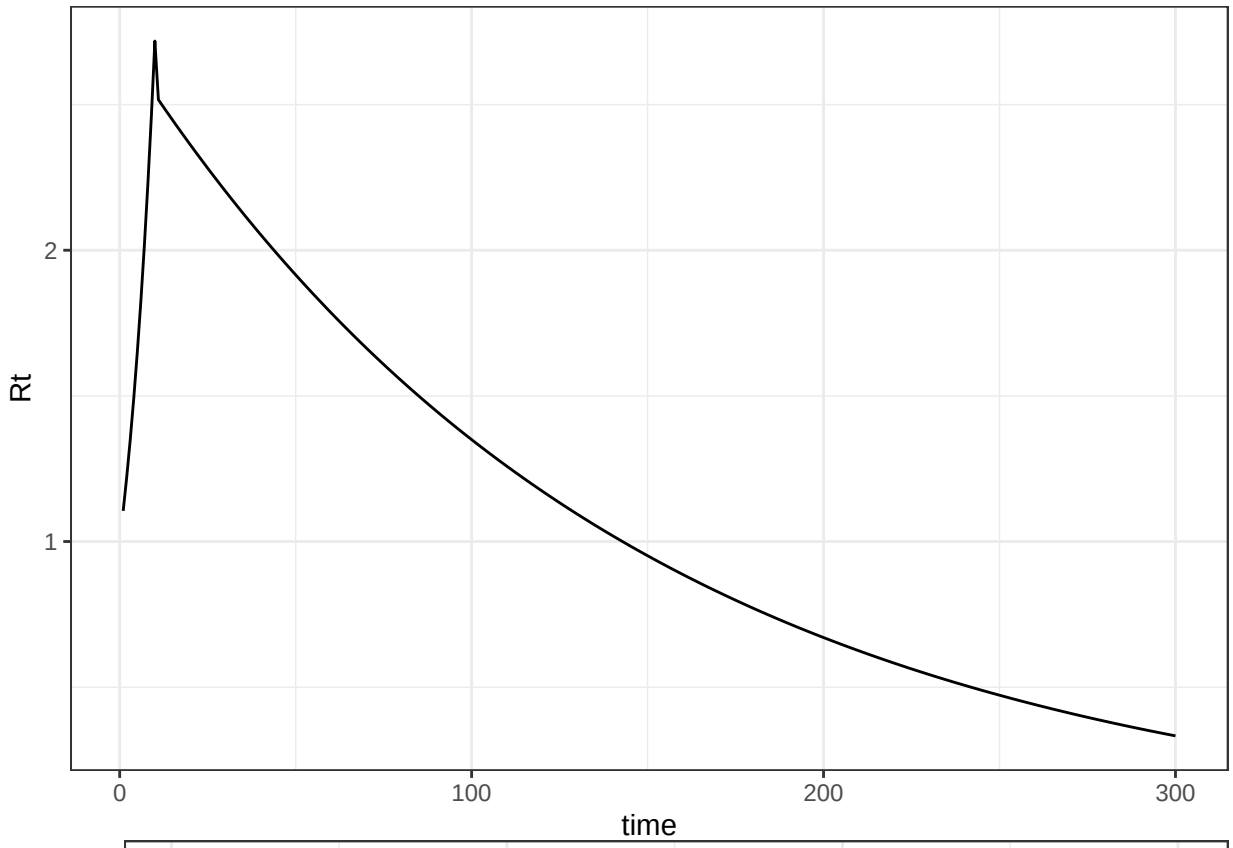


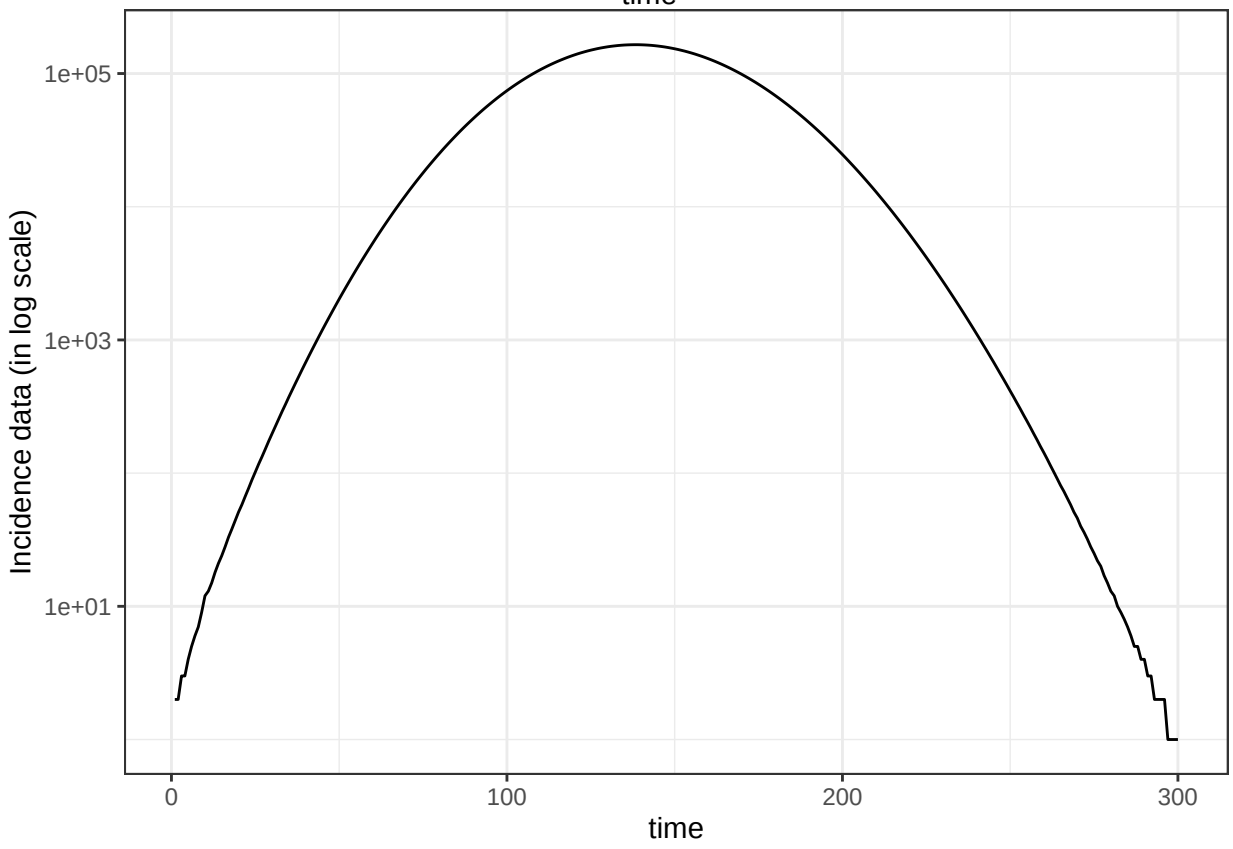

```
## Warning: Transformation introduced infinite values in continuous y-axis
```




```
check_dat(incidence1, Rt1)

# Scenario 2: two-stage exponential growth and decay (similar as in EpiFilter)
rate <- c(.1, -.007)
Rt2 <- numeric(len)
Rt2[1:10] <- exp(rate[1] * (1:10))
Rt2[11:len] <- exp((rate[2]) * (11:len))* Rt2[10]
gamma_pars2 <- c(2.5, 2.5) # serial interval distribution parameters
incidence2 <- get_incidence(N1, Rt2, gamma_pars2)
incidence2 <- round(incidence2)
display_dat(incidence2, Rt2)
```


Rt
2
1
0
100
200
300
time


Incidence data (in log scale)
1e+05
1e+03
1e+01
0
100
200
300
time

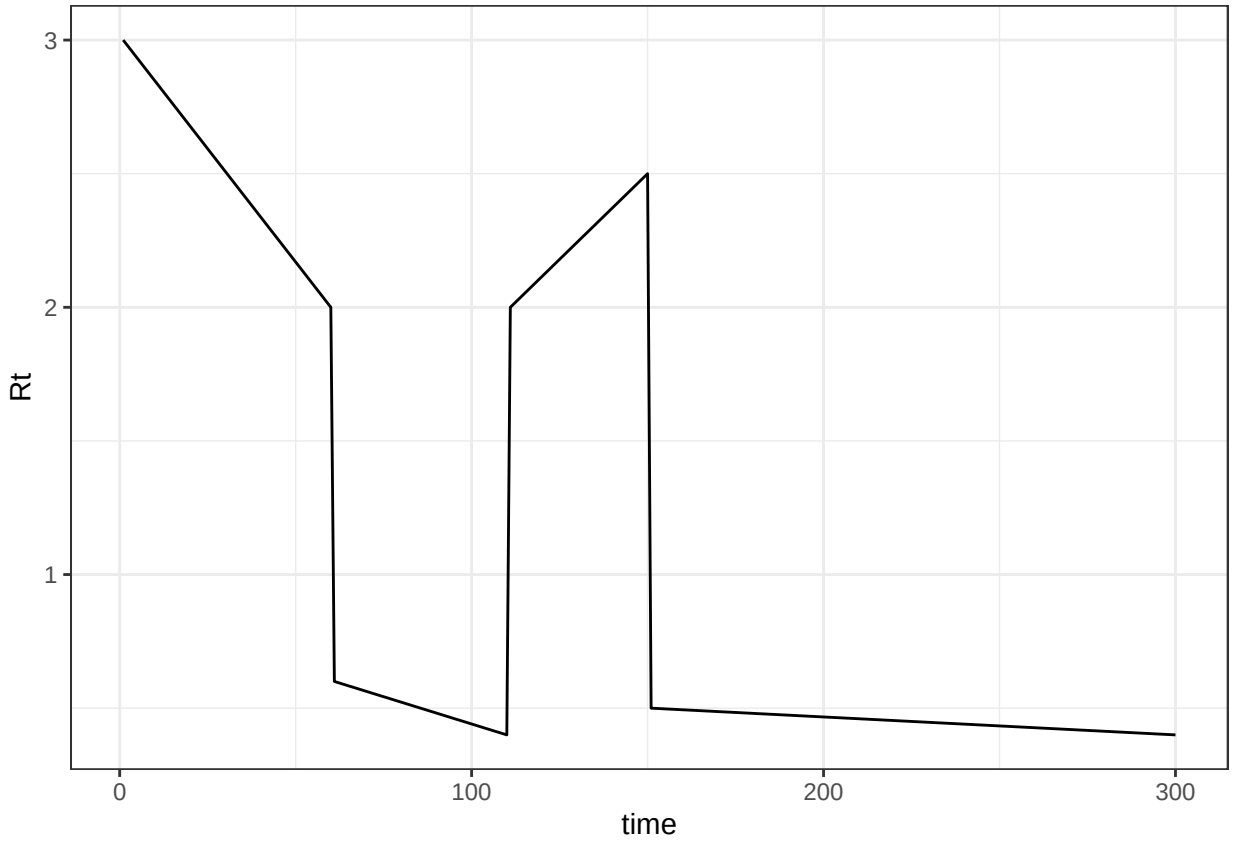
```r
check_dat(incidence2, Rt2)

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

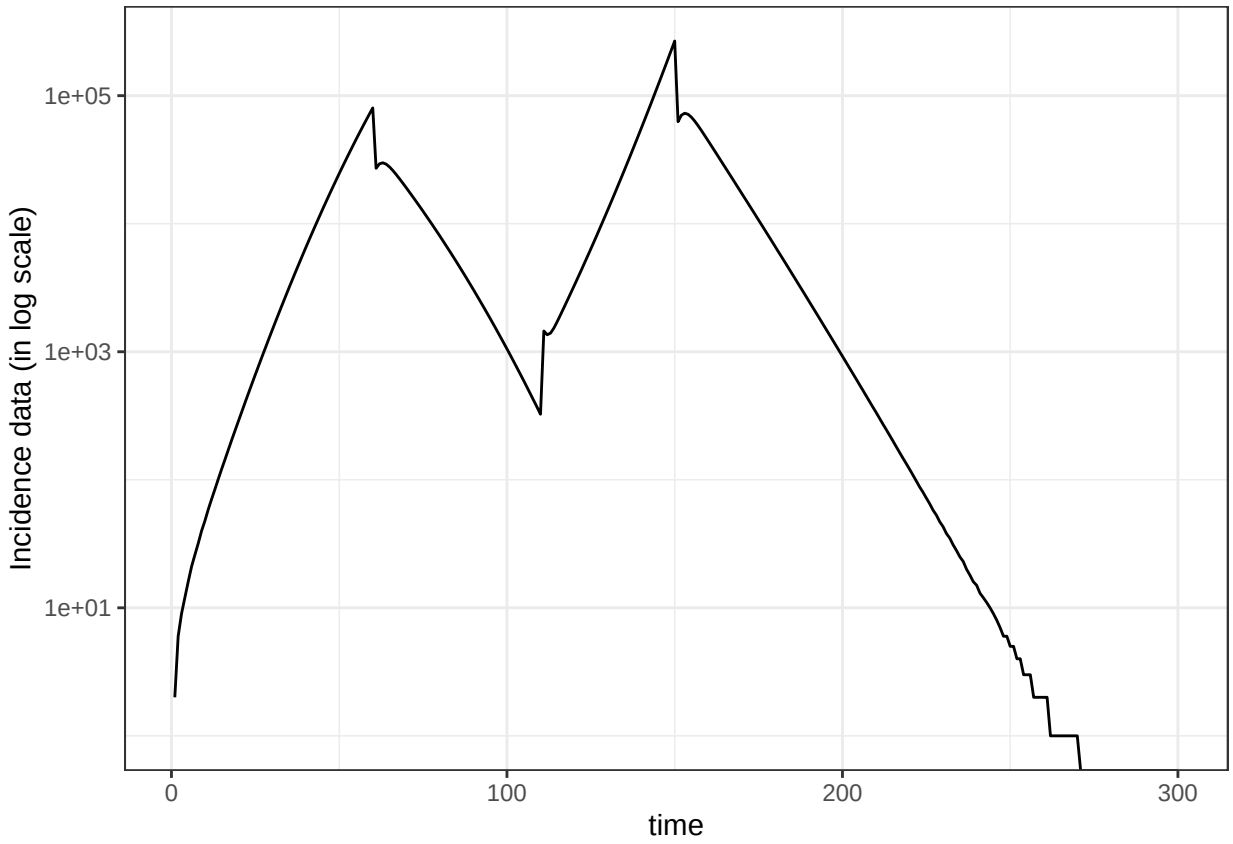
```
##
## Attaching package: 'pracma'
```

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

```r
Rt3 <- c(seq(3, 2, length.out = 60), seq(0.6, 0.4, length.out = 50), seq(2, 2.5, length.out = 40), seq(
gamma_pars3 <- c(2.5, 2.5) # serial interval distribution parameters
incidence3 <- get_incidence(N1, Rt3, gamma_pars3)
incidence3 <- round(incidence3)
display_dat(incidence3, Rt3)
```

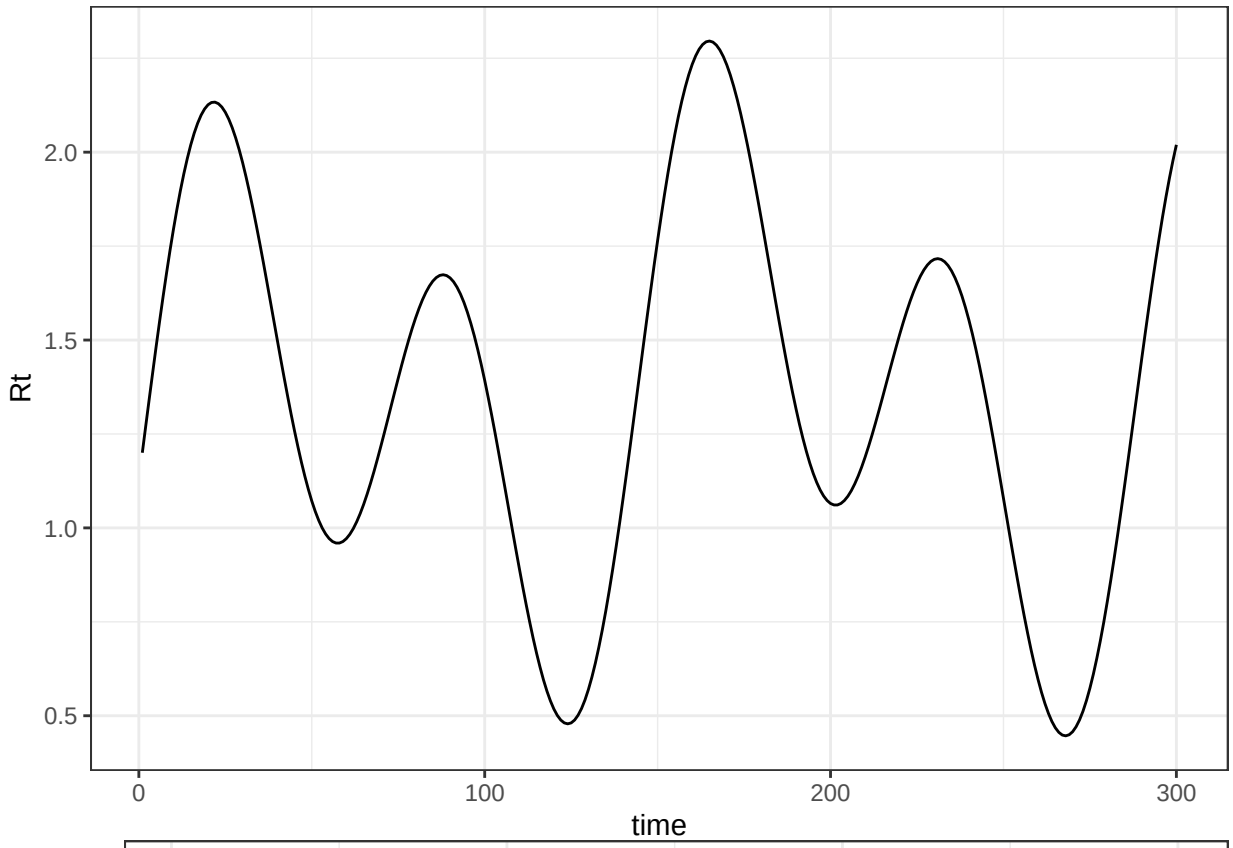


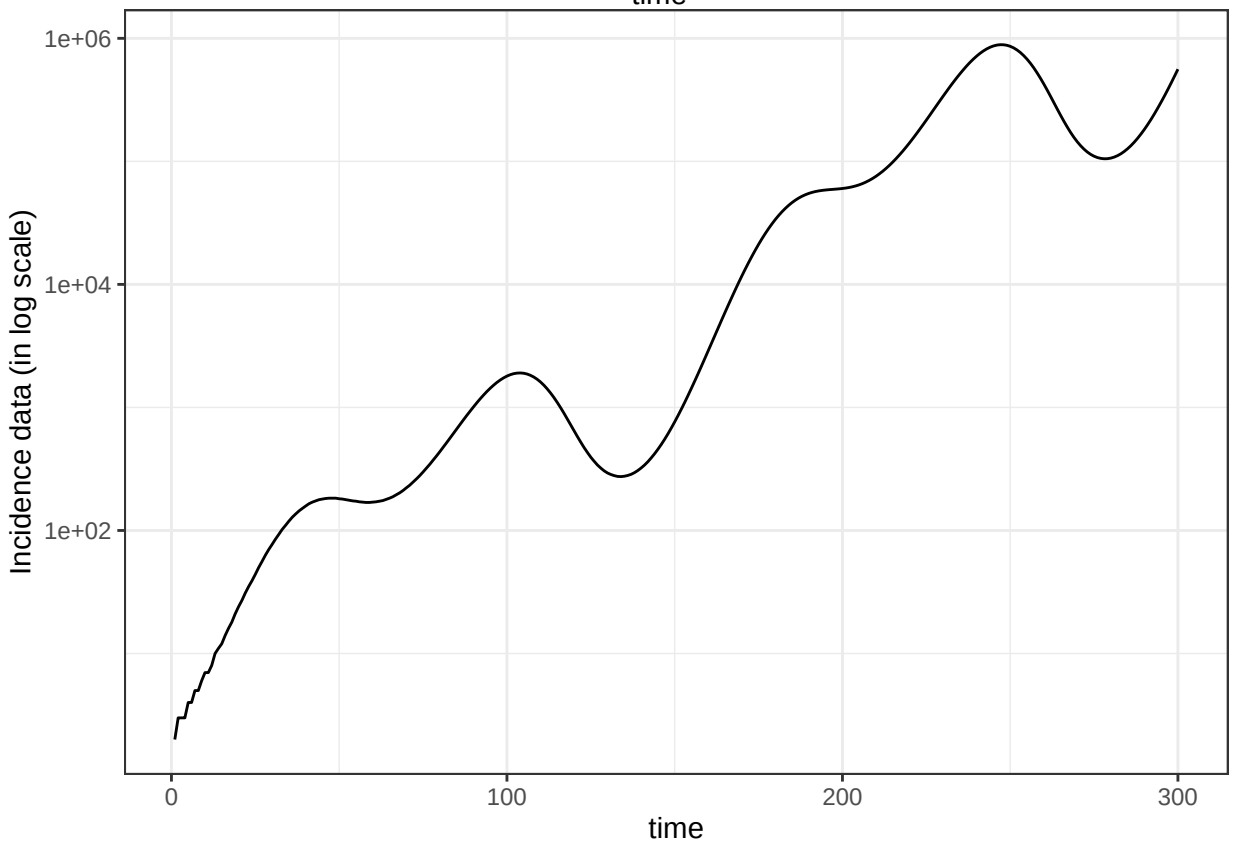

```
## Warning: Transformation introduced infinite values in continuous y-axis
```




```r
check_dat(incidence3, Rt3)

# Scenario 4: multi-stage with exponential-sinusoidal growth and decay (cited the function from EpiLPS)
x <- seq(0, 10, length.out = len)
Rt4 <- numeric(len)
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 + 0.2 * (component$amp * sin(pi * component$freq * x / 1.2) + + component$amp)
}
#Rt4 <- 1:len
#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)
incidence4 <- round(incidence4)
display_dat(incidence4, Rt4)
```

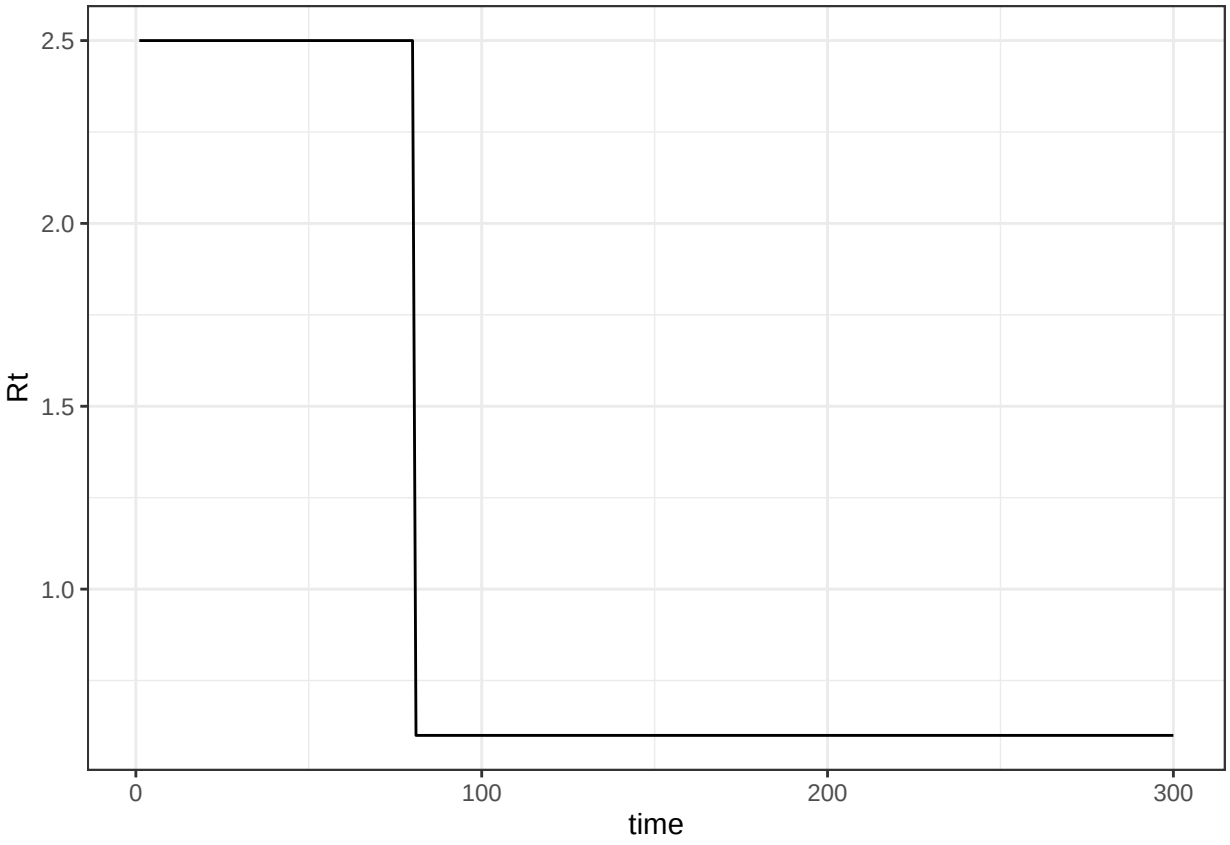

Rt
2.0
1.5
1.0
0.5
0
100
200
300
time


Incidence data (in log scale)
1e+06
1e+04
1e+02
0
100
200
300
time

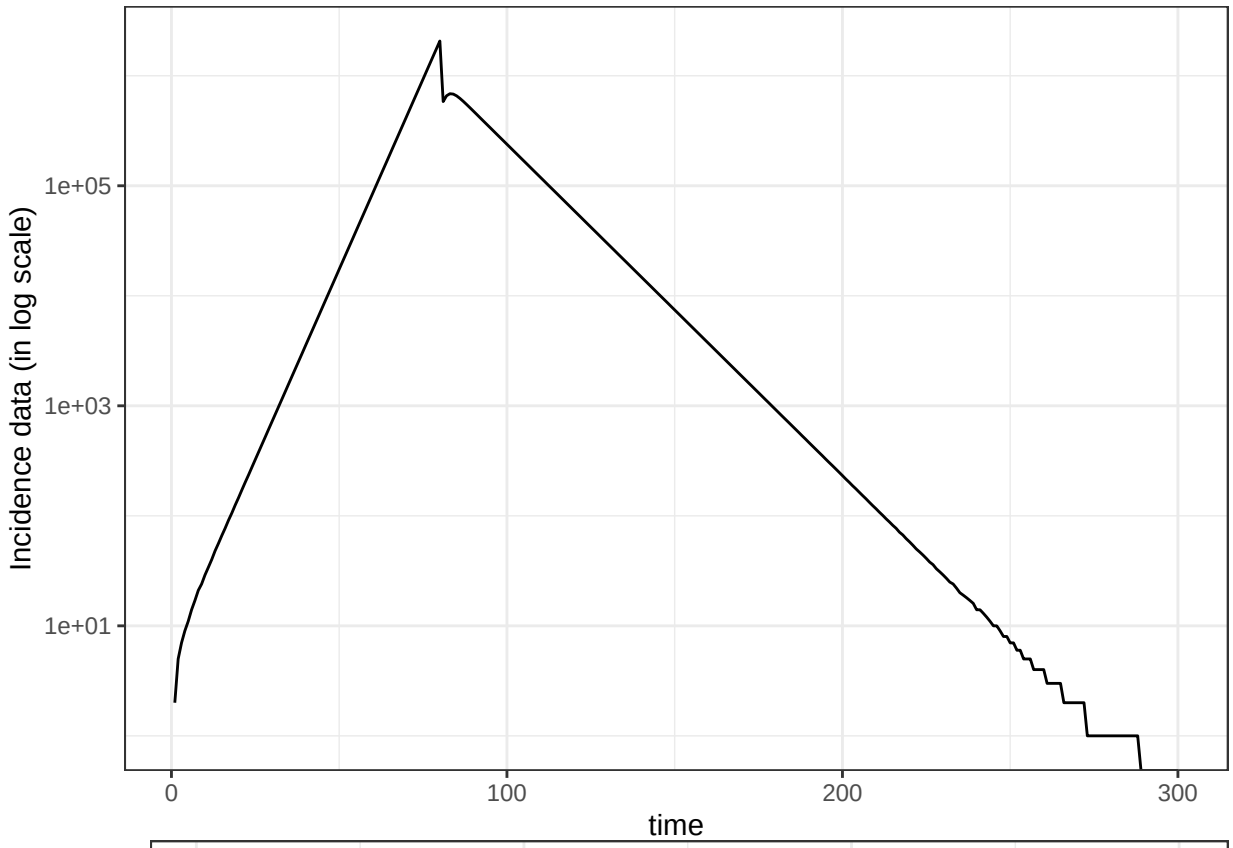
```r
check_dat(incidence4, Rt4)
```

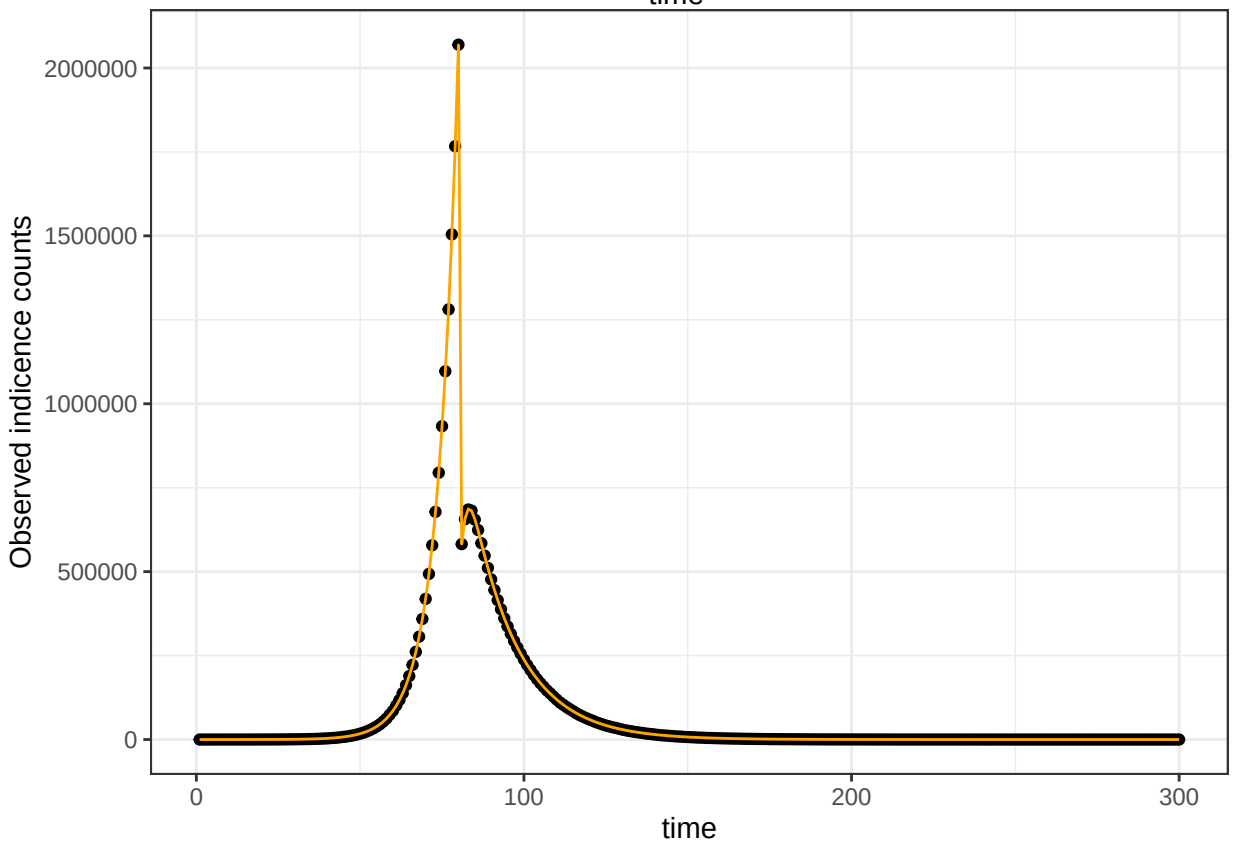
```r
# Generate incidence observations
set.seed(857)
count_data1 <- rpois(len, incidence1)
count_data2 <- rpois(len, incidence2)
count_data3 <- rpois(len, incidence3)
count_data4 <- rpois(len, incidence4)

display_dat(incidence1, Rt1, count_data1)
```

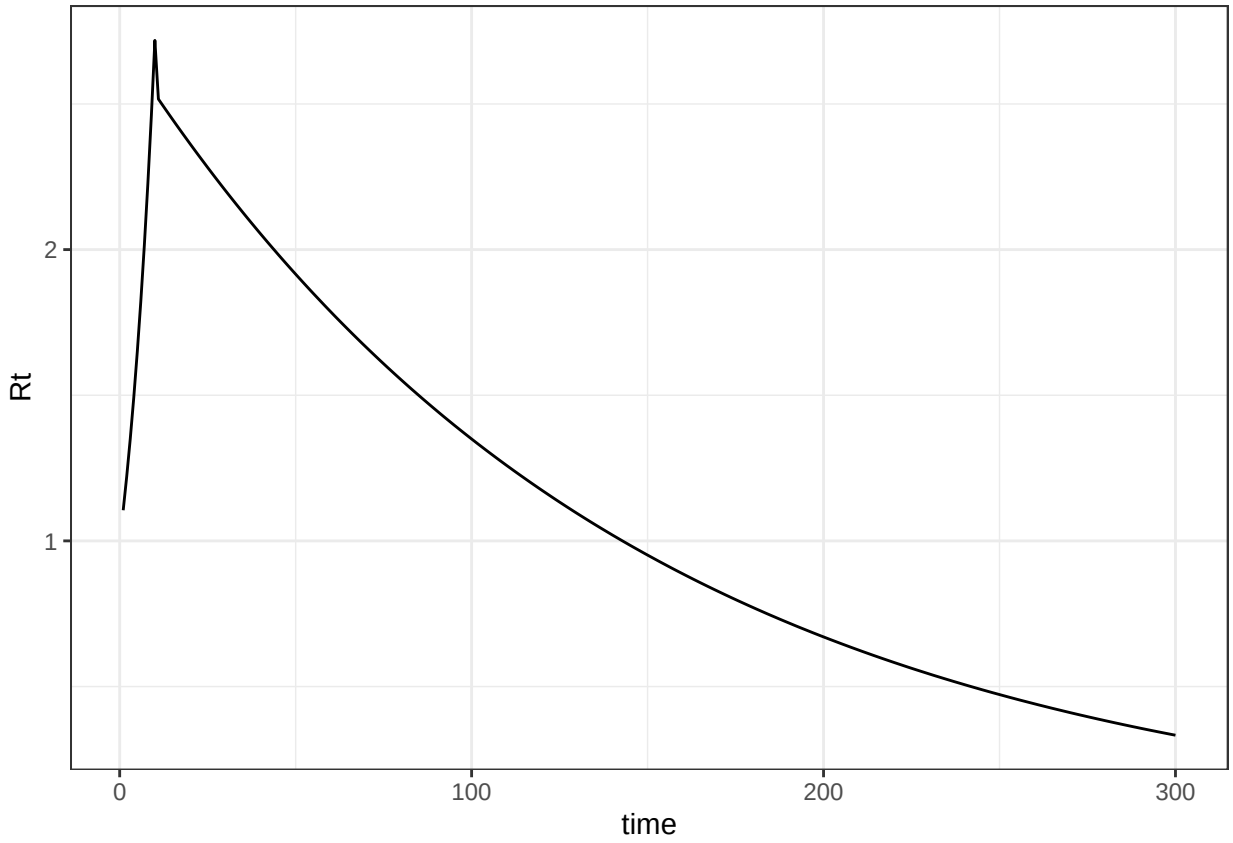


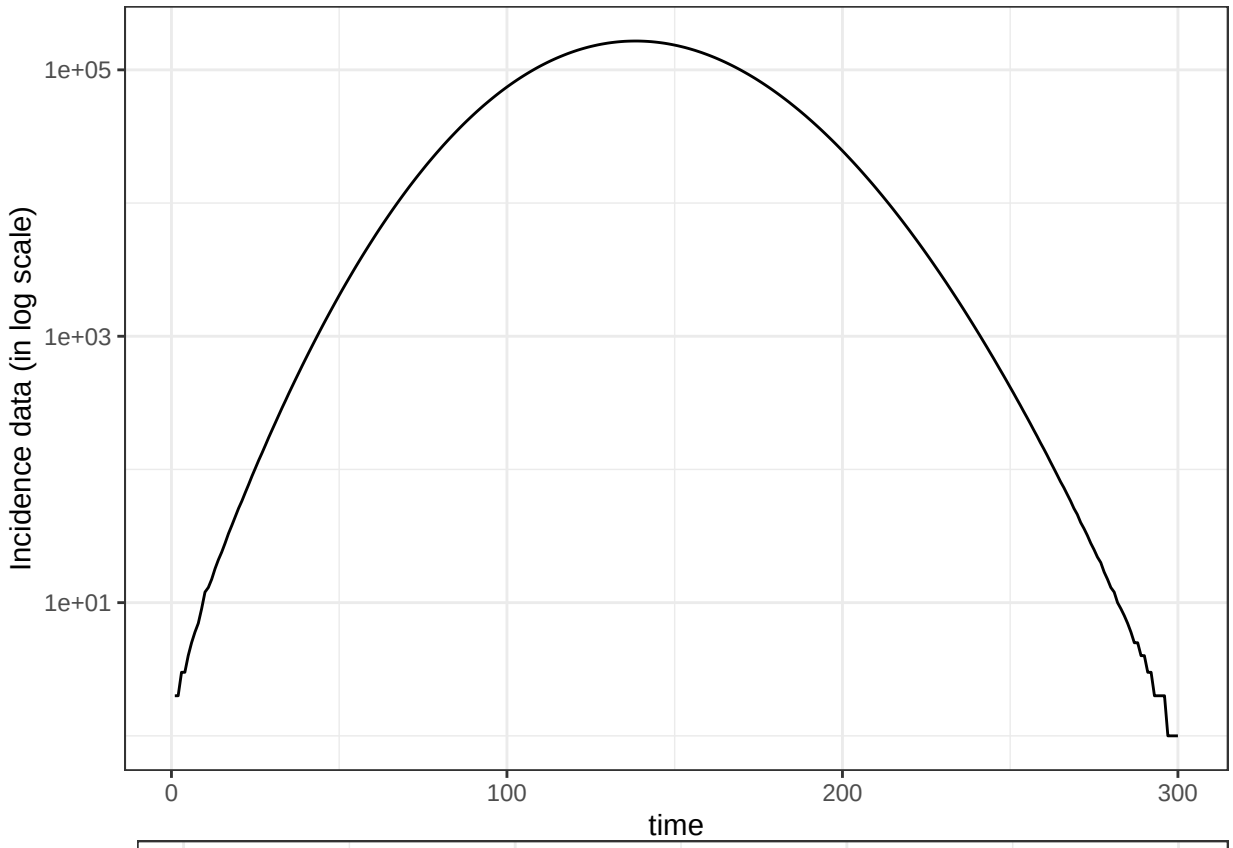

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

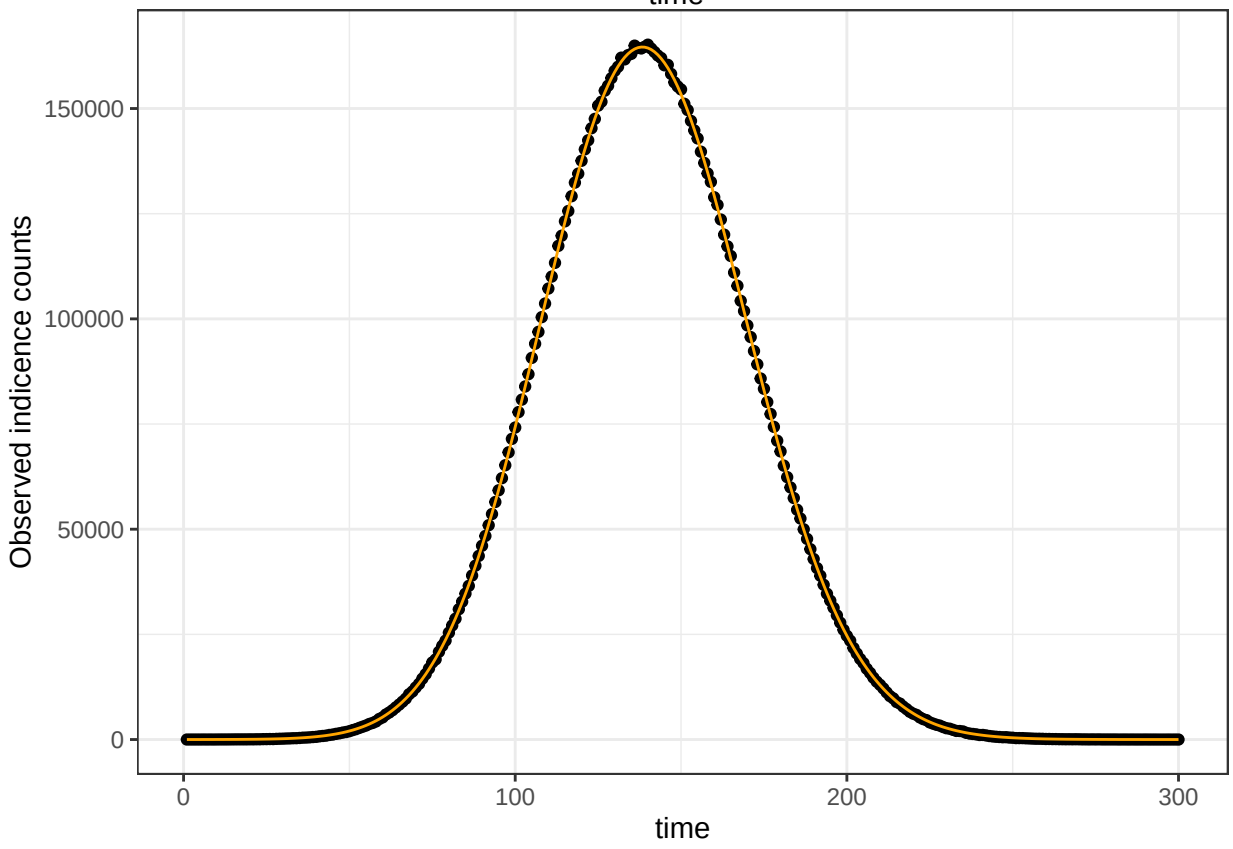

Incidence data (in log scale)
1e+05
1e+03
1e+01
0
100
200
300
time


Observed indicence counts
2000000
1500000
1000000
500000
0
0
100
200
300
time

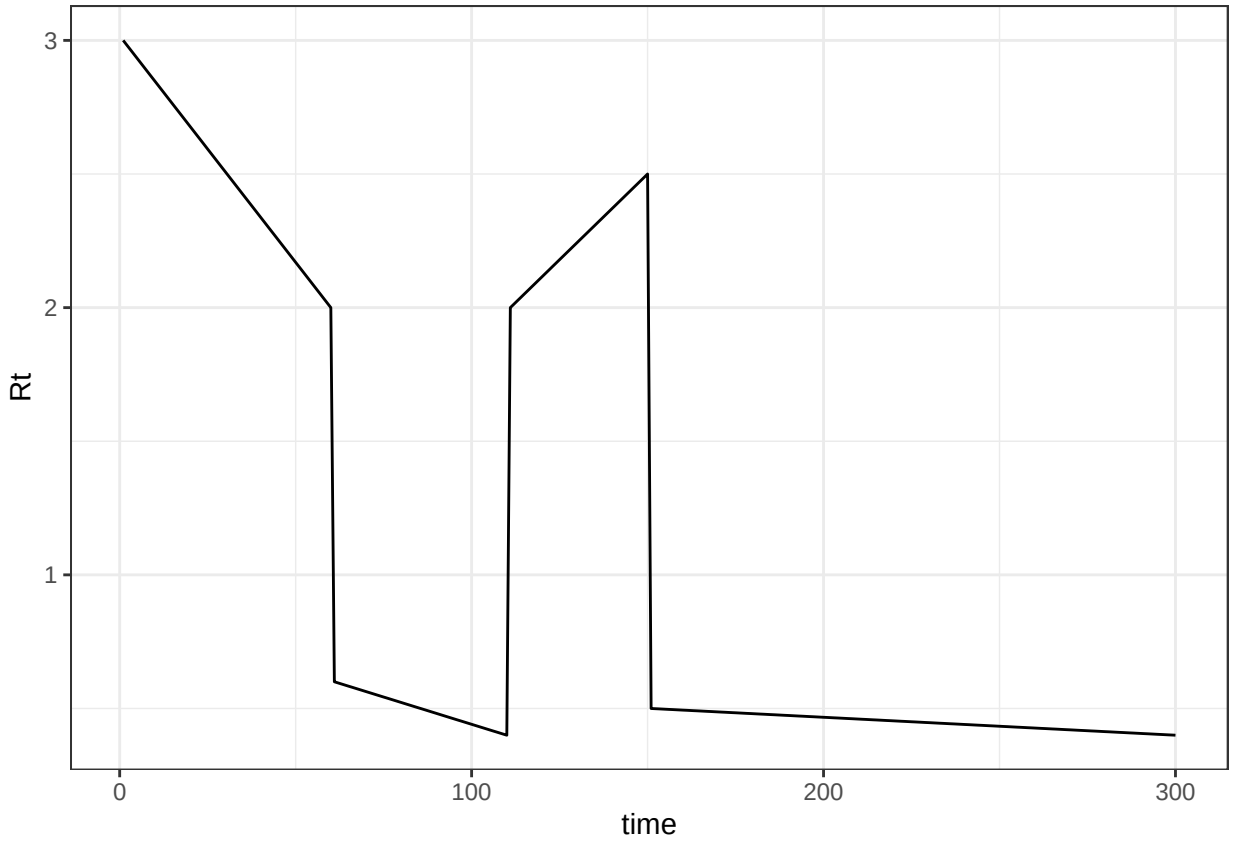
```
display_dat(incidence2, Rt2, count_data2)
```



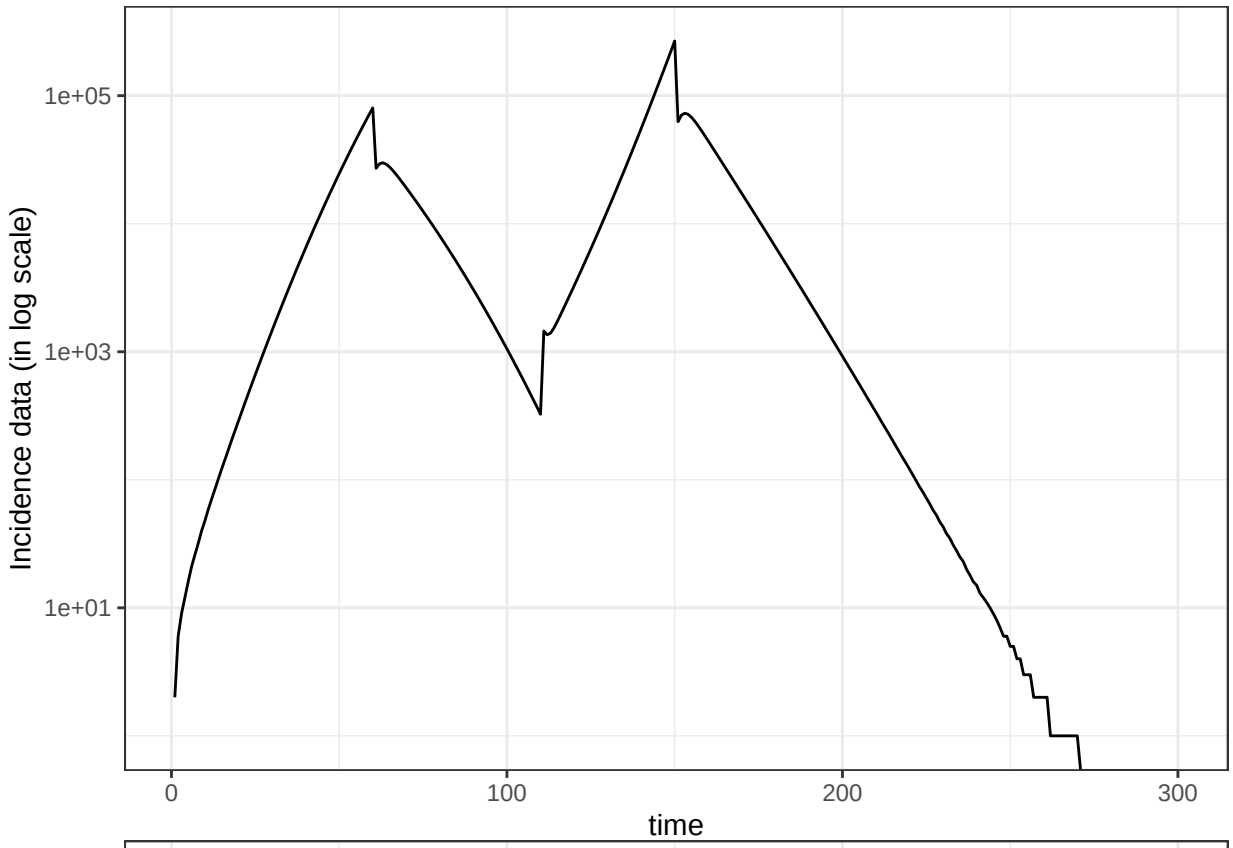

1e+05
1e+03
1e+01
Incidence data (in log scale)
0
100
200
300
time

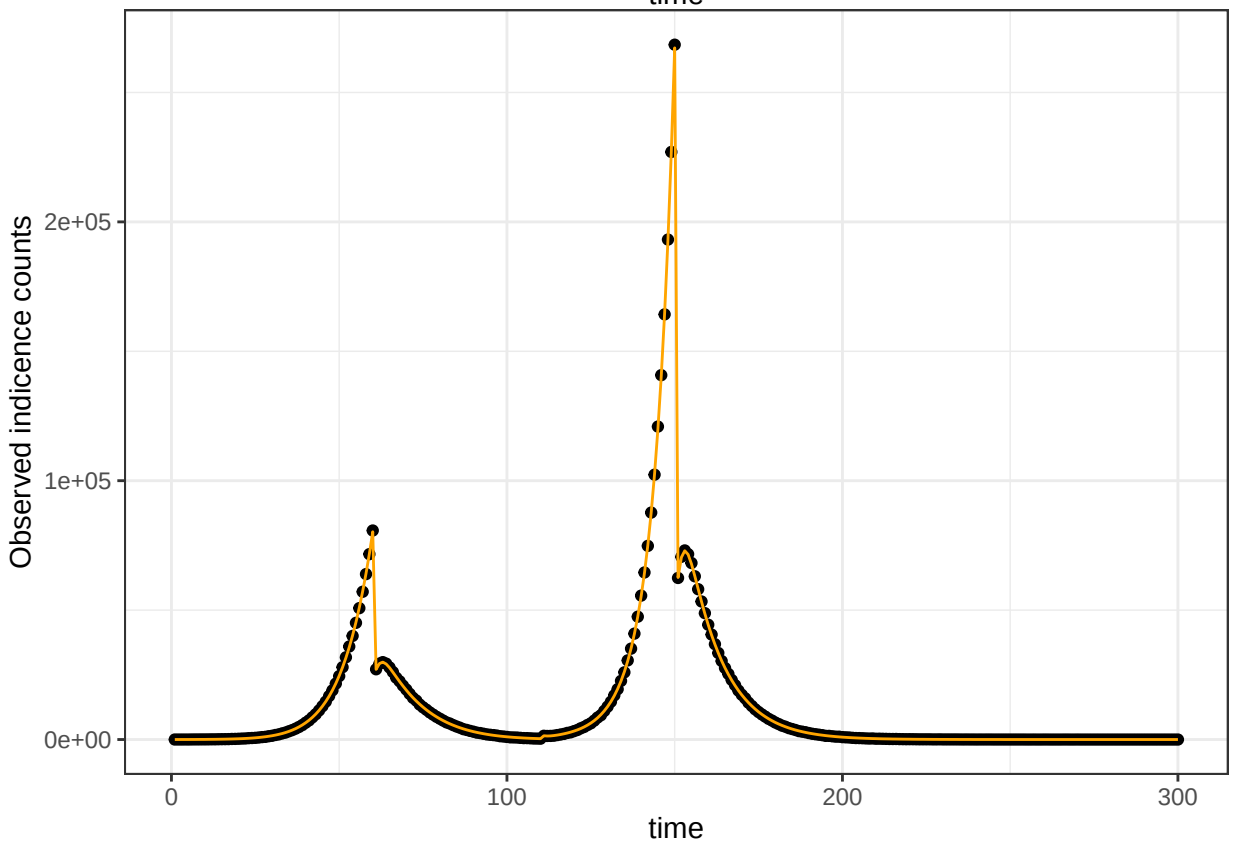

150000
100000
50000
0
Observed indicence counts
0
100
200
300
time

```
display_dat(incidence3, Rt3, count_data3)
```

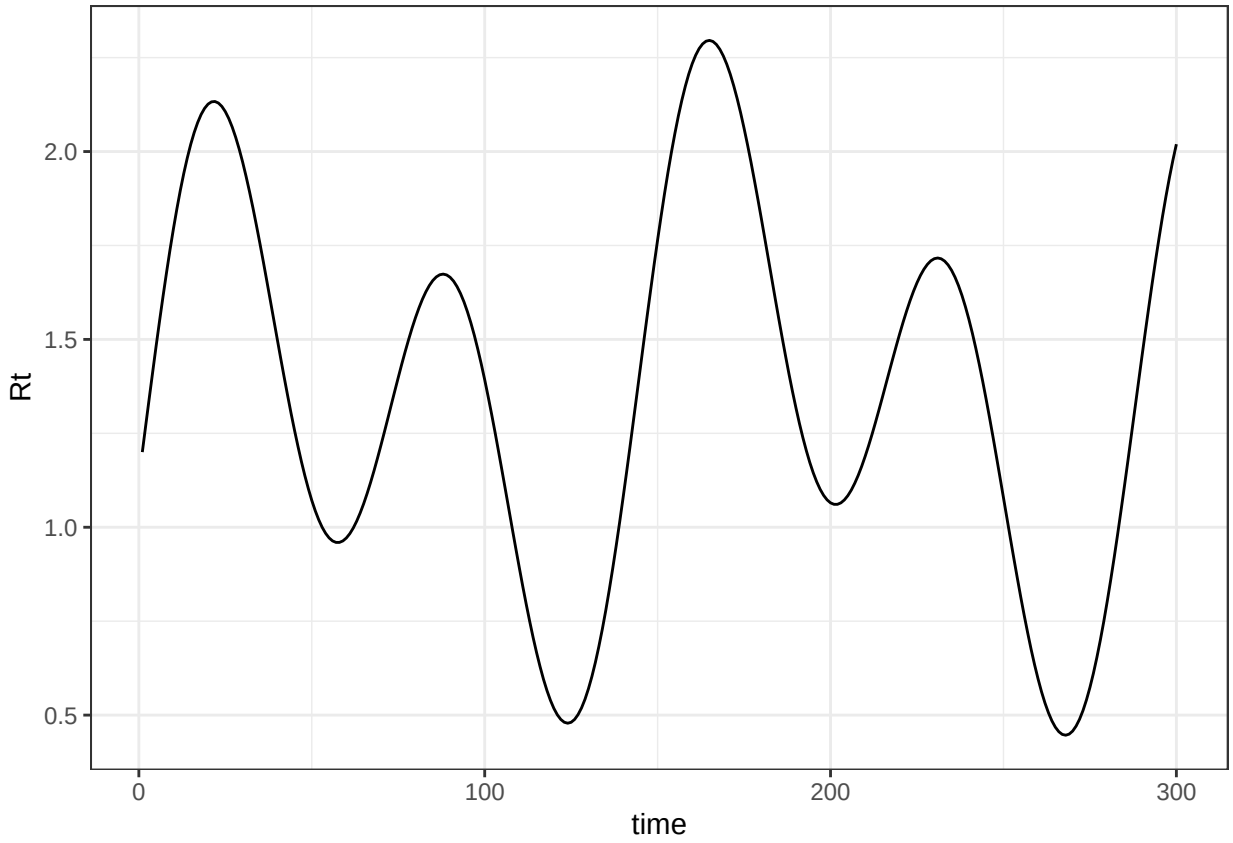


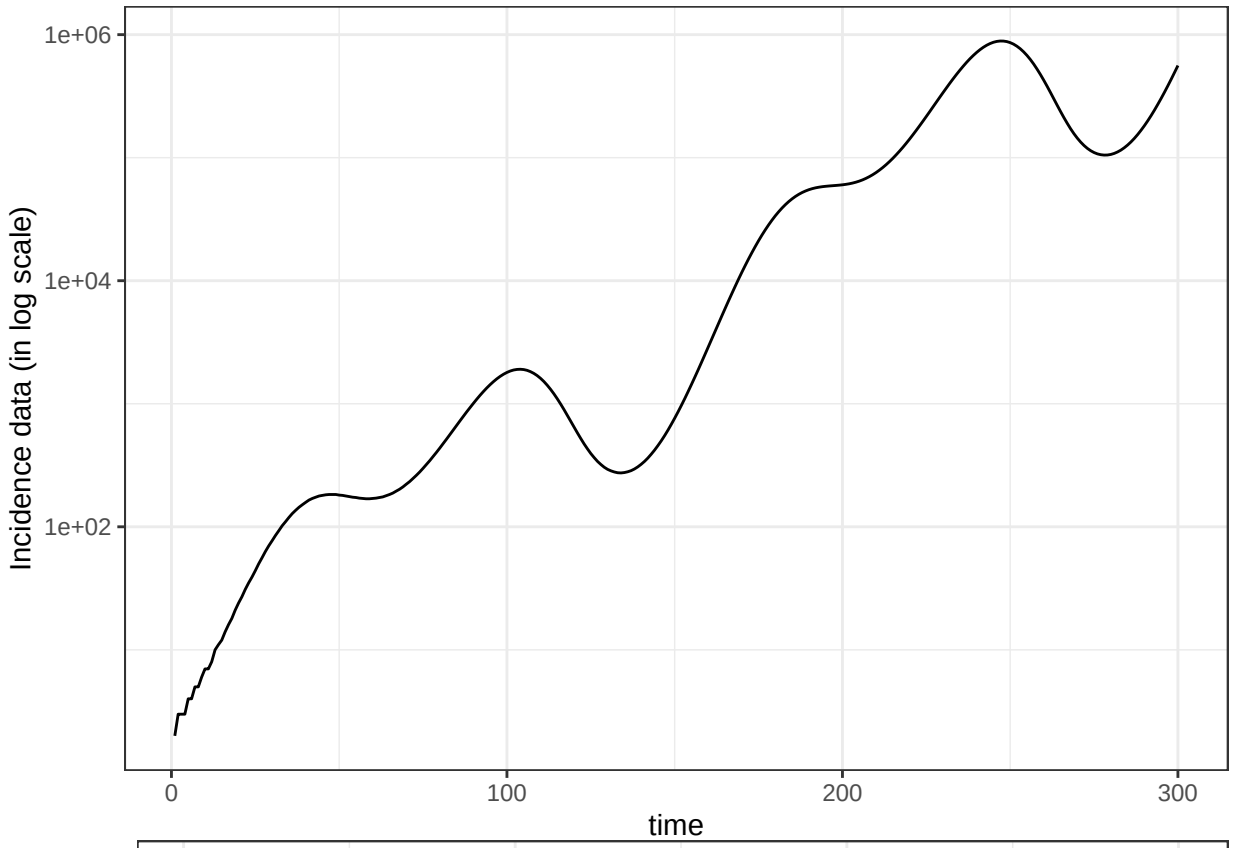

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

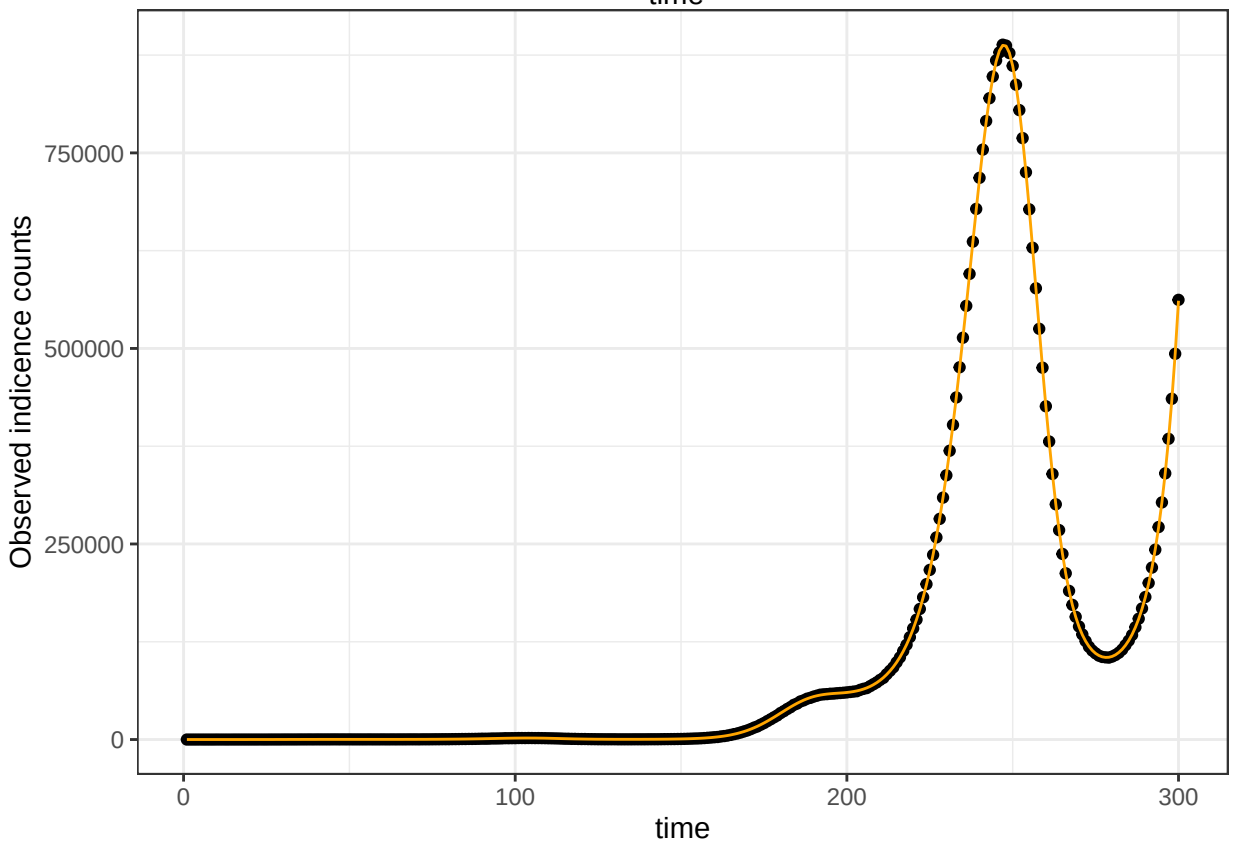

1e+05
1e+03
1e+01
Incidence data (in log scale)
0
100
200
300
time


2e+05
1e+05
0e+00
Observed indicence counts
0
100
200
300
time

```
display_dat(incidence4, Rt4, count_data4)
```




1e+06
1e+04
1e+02
Incidence data (in log scale)
0
100
200
300
time


750000
500000
250000
0
Observed indicence counts
0
100
200
300
time

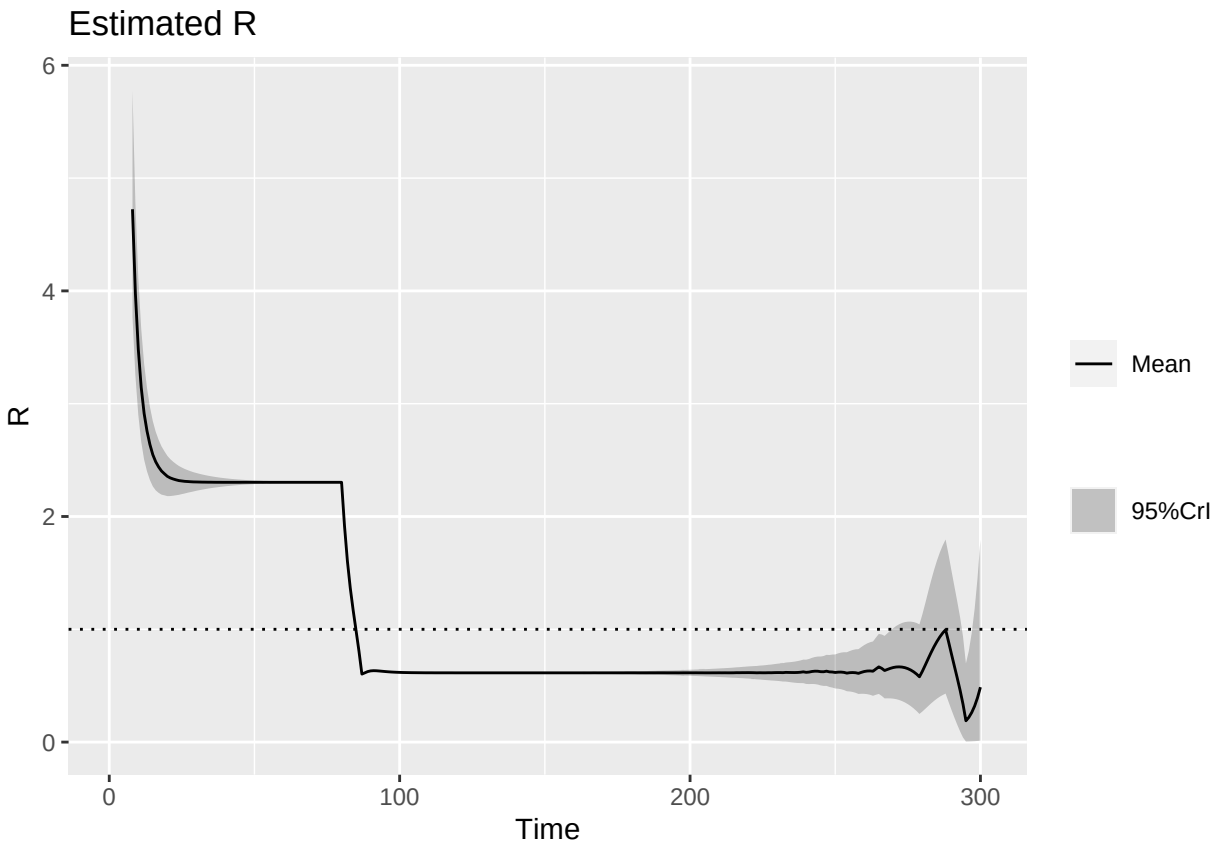
```
# Note: since both of the incidence data and range of data can get very large, the noises varying by ra
```

Reproduction number estimation

```
# EpiEstim with 12-day sliding windows
library(EpiEstim)
mean_si <- 6.3
std_si <- 4.2
method <- "parametric_si"
config <- make_config(list(mean_si = mean_si, std_si = std_si))#t_start=2, t_end=14
mod_epiEstim1 <- EpiEstim::estimate_R(incid = incidence1,
            #dt = 7L, # smoothing window is 7 days
            #recon_opt = "naive",
            config = config,
            method = method)
```

```
## Default config will estimate R on weekly sliding windows.
##     To change this change the t_start and t_end arguments.
```

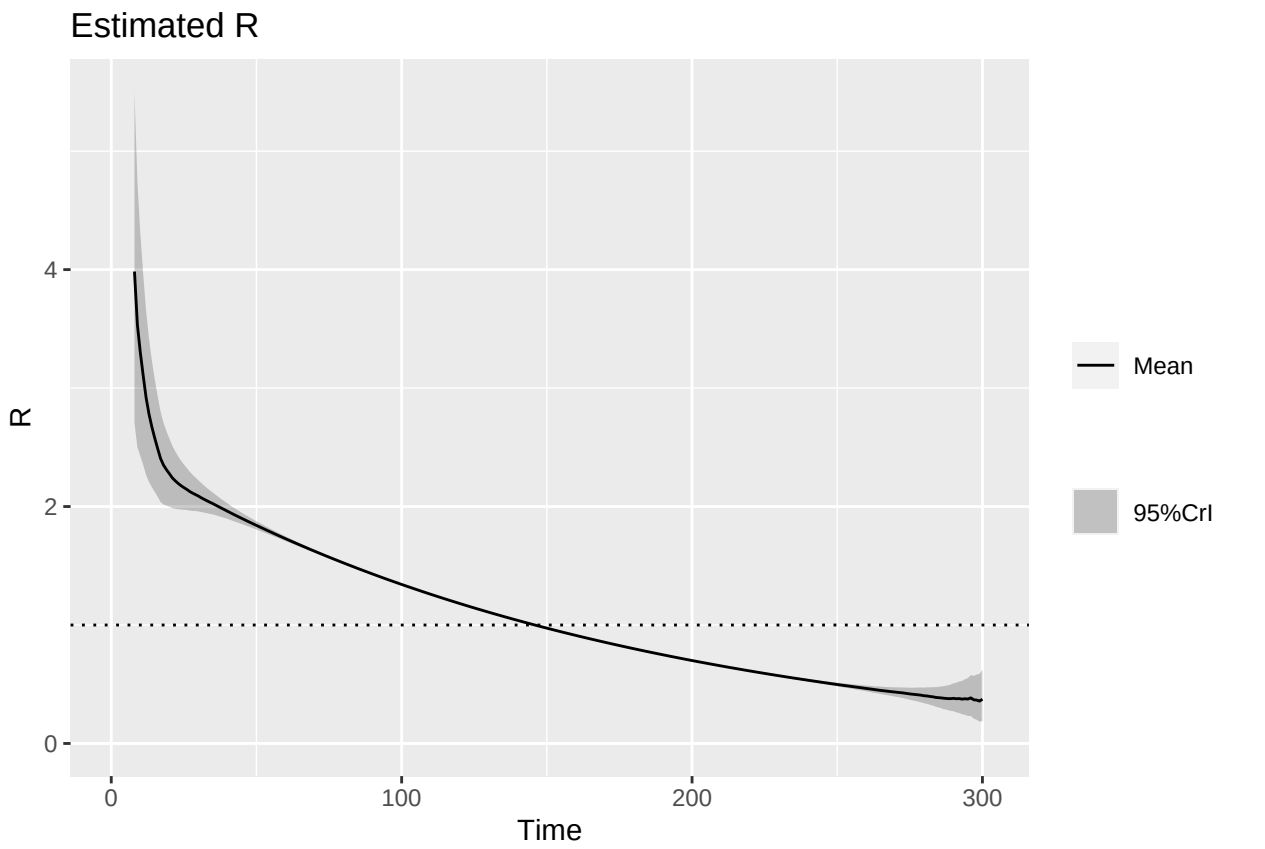
```
#head(mod_epiEstim1$R)
plot(mod_epiEstim1, "R")
```




```
mod_epiEstim2 <- EpiEstim::estimate_R(incid = incidence2, config = config, method = method)
```

```
## Default config will estimate R on weekly sliding windows.
##     To change this change the t_start and t_end arguments.
```

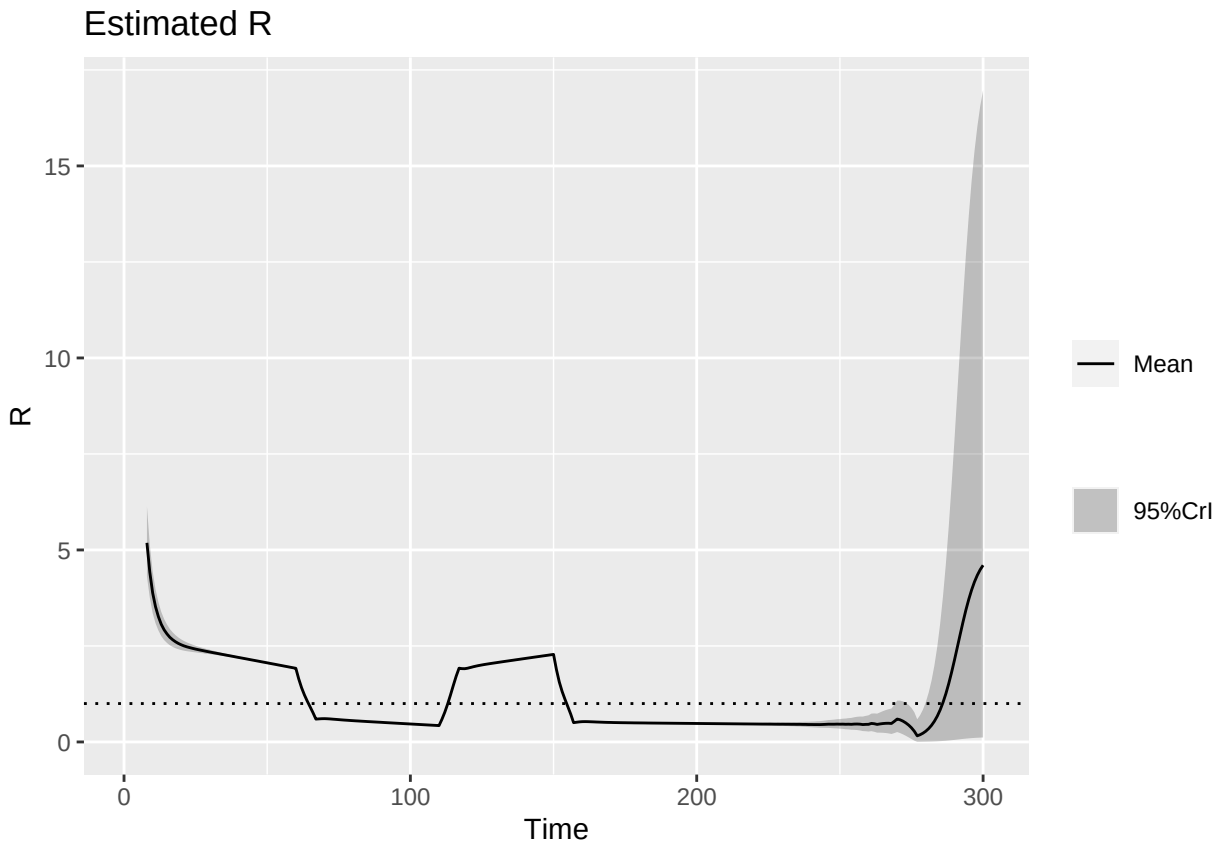
```
plot(mod_epiEstim2, "R")
```




```
mod_epiEstim3 <- EpiEstim::estimate_R(incid = incidence3, config = config, method = method)
```

```
## Default config will estimate R on weekly sliding windows.
##     To change this change the t_start and t_end arguments.
```

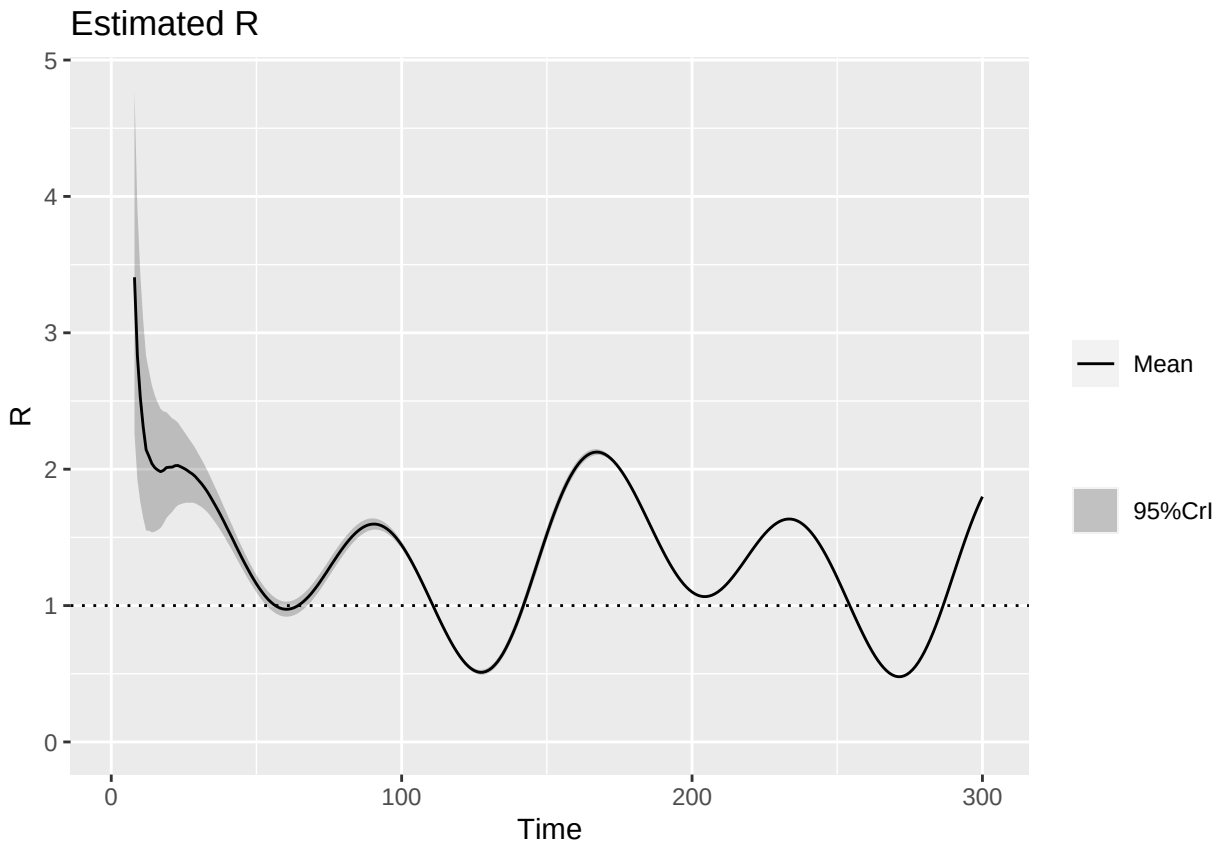
```
plot(mod_epiEstim3, "R")
```




```
mod_epiEstim4 <- EpiEstim::estimate_R(incid = incidence4, config = config, method = method)
```

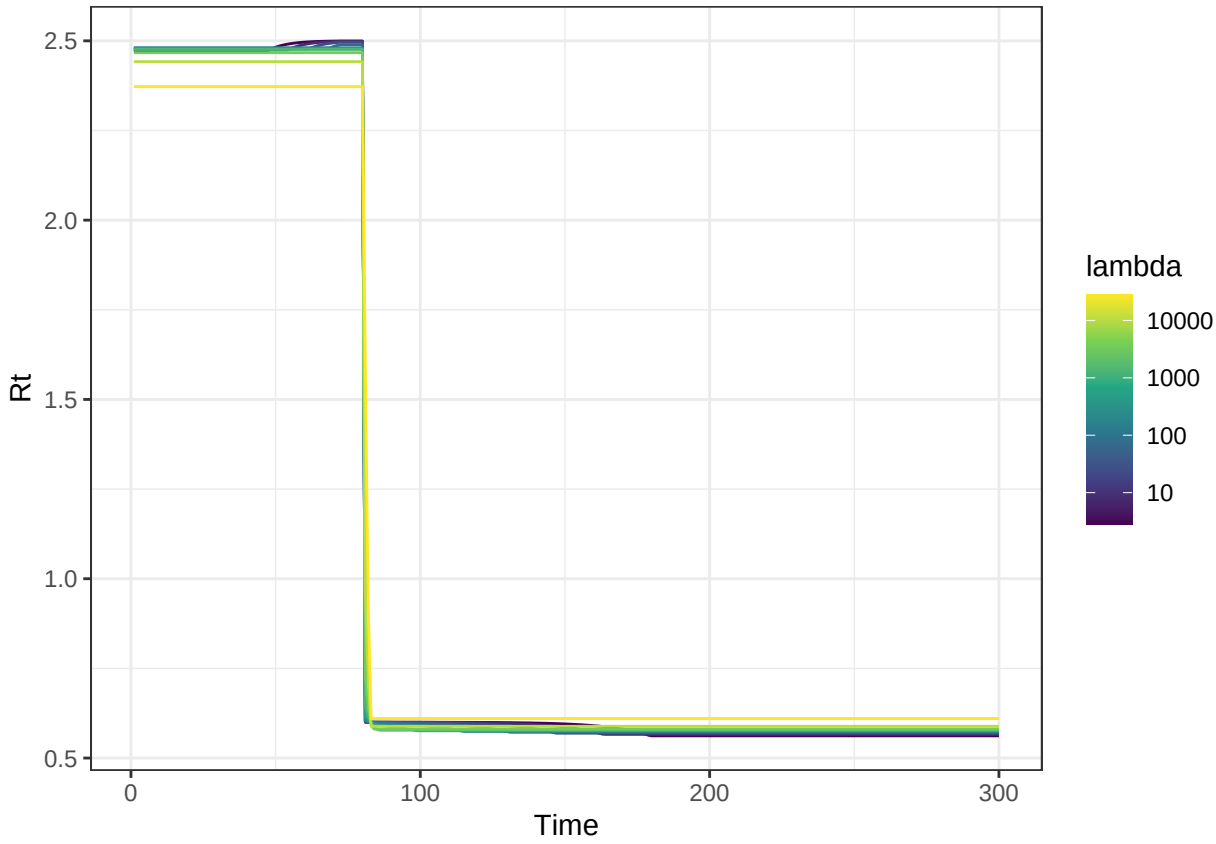
```
## Default config will estimate R on weekly sliding windows.
##     To change this change the t_start and t_end arguments.
```

```
plot(mod_epiEstim4, "R")
```

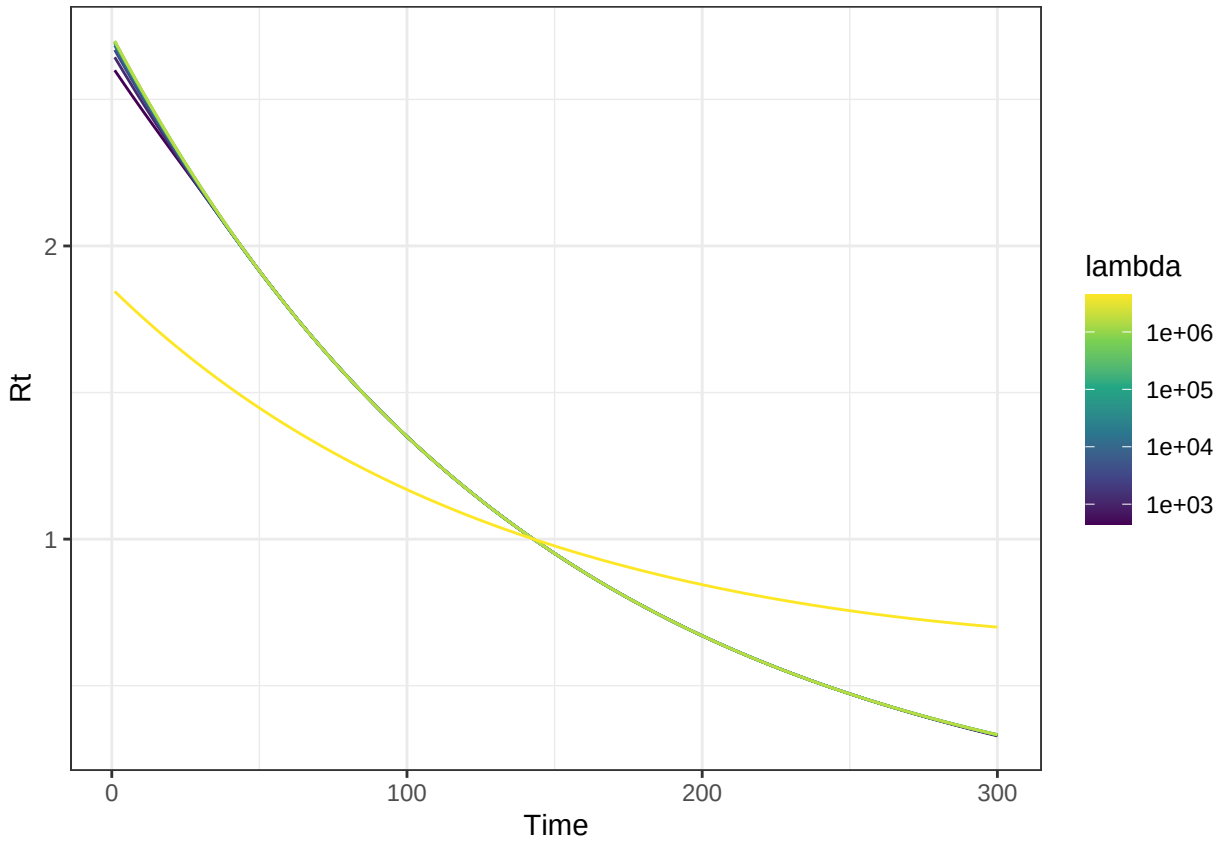



```
# EpiFilter; refer to https://github.com/kpzoo/EpiFilter/blob/master/R%20files/vignette.R
source("EpiFilter/epiFilter.R")
## Setup grid and noise parameters
#Rmin = 0.01; Rmax = 10; eta = 0.1
## Uniform prior over grid of size m
#m = 200; pR0 = (1/m)*rep(1, m)
## Delimited grid defining space of R
#Rgrid = seq(Rmin, Rmax, length.out = m)
#a = 0.025
#
#Iflu = read.csv('EpiFilter/Iflu.csv'); Iflu = Iflu[[1]]
#Lflu = read.csv('EpiFilter/Lflu.csv'); Lflu = Lflu[[1]]
#tflu <- 8:length(Iflu)
#epfilter1 <- epiFilter(Rgrid, m, eta, pR0, len, Lflu[tflu], #Iflu[tflu], a)
#nflu <- length(tflu)
##source("EpiFilter/plotEpiFilter.R")
##plotEpiFilter(Rt2[[3]][2:nflu], Rt2[[2]][, 2:nflu],
##              'EpiFilter_flu', folres, eta)
#plot(epfilter1[[3]][2:nflu])
```

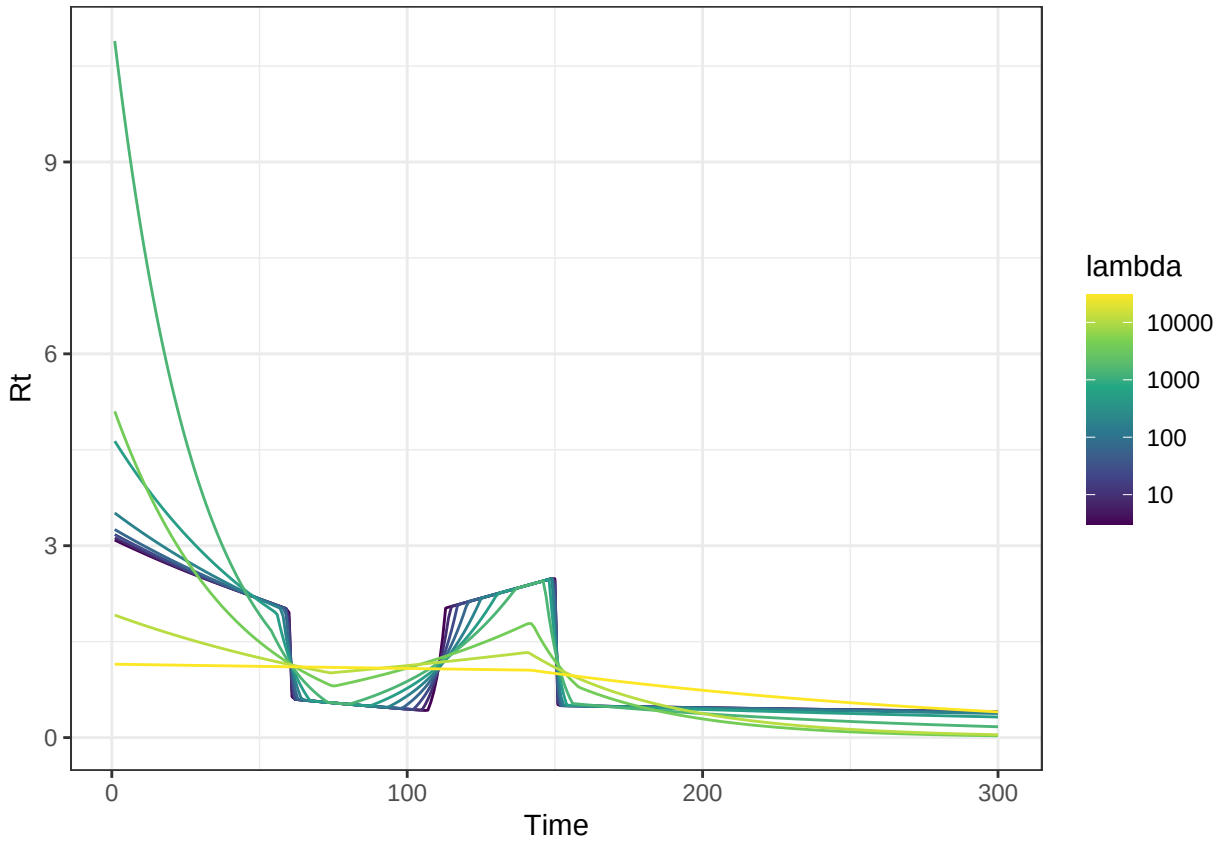
```
rtestim_mod1 <- rtestim::estimate_rt(incidence1, korder=0, nsol=10)
plot(rtestim_mod1)
```



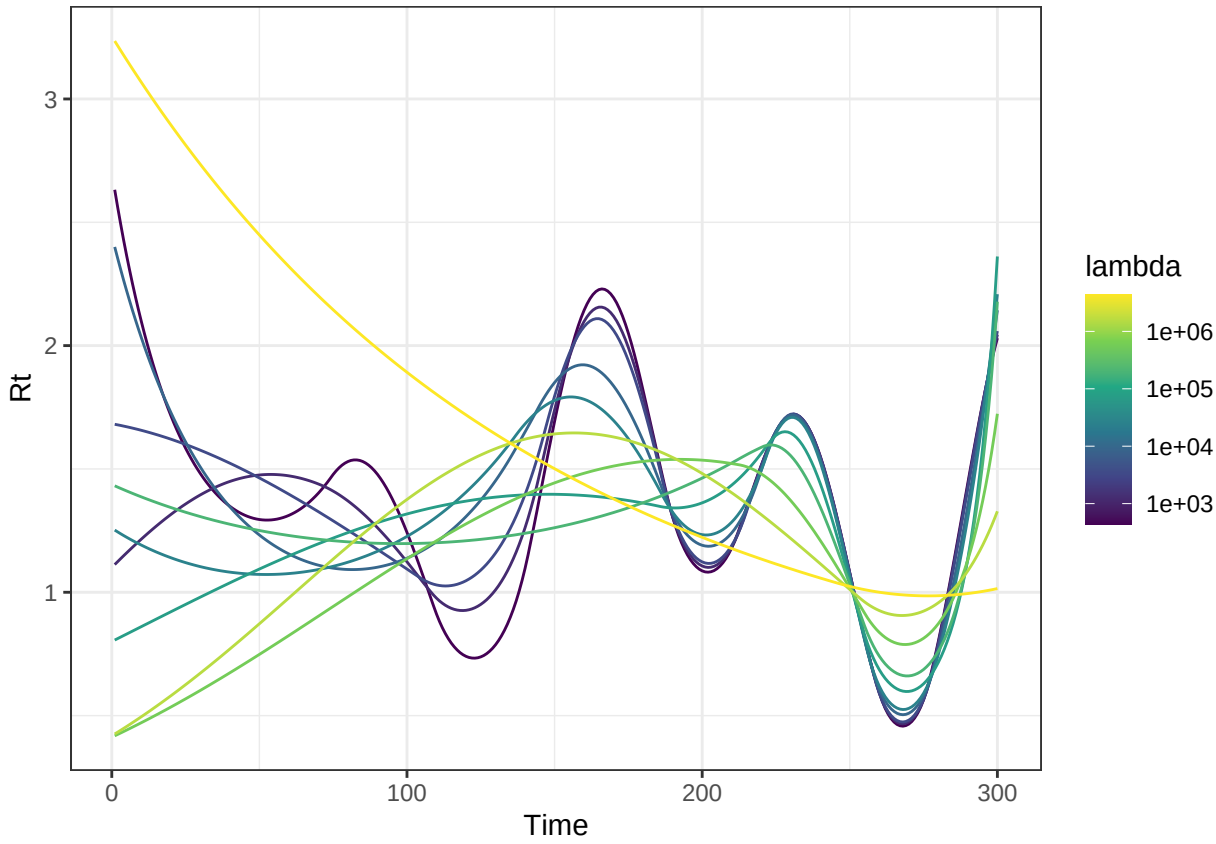

```
rtestim_mod2 <- rtestim::estimate_rt(incidence2, korder=2, nsol=10)
plot(rtestim_mod2)
```




```
rtestim_mod3 <- rtestim::estimate_rt(incidence3, korder=1, nsol=10)
plot(rtestim_mod3)
```




```
rtestim_mod4 <- rtestim::estimate_rt(incidence4, korder=2, nsol=10)
plot(rtestim_mod4)
```


Rt
3
2
1
0
100
200
300
Time
lambda
1e+06
1e+05
1e+04
1e+03