SISMID Module 6: Stochastic Epidemic Models with Inference – Solutions to Exercise Session 1

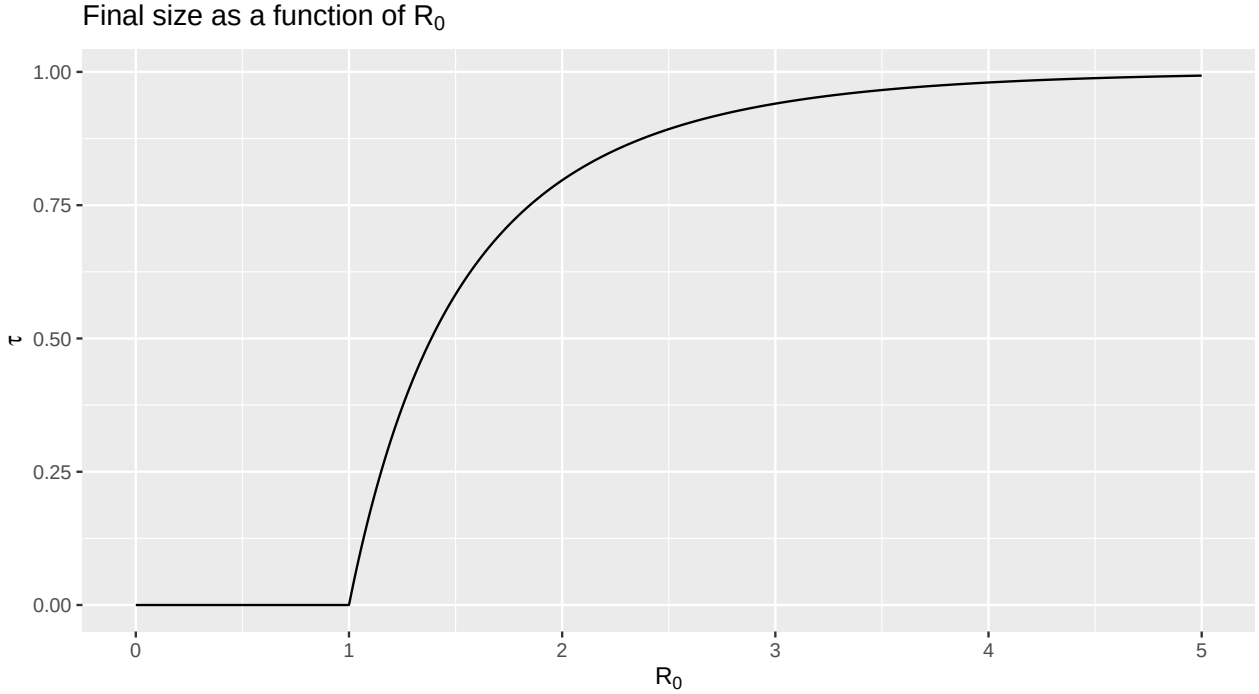
July 12, 2021

Exercise 1.1 (Final Outbreak Size)

(a) Solve the final size equation $1 - \tau = \exp(-R_0\tau)$ numerically (with the largest solution for $\tau \in [0, 1]$) as a function of R_0 and plot the function for $R_0 \in [0, 5]$.

```
################################################################################
# Function of R_0 solving for tau numerically
#  R_0 - reproduction number
#  tau - final size
#  Return: tau
################################################################################
tau <- function(R_0) {
  final_eq <- function(tau) {1 - tau - exp(-R_0*tau)}
  "
  Note: This final size equation is always a solution tau = 0.
  When R_0 <=1, it gives only one solution tau = 0;
  if R_0 > 1, then there are two solutions.
  "
  result1 <- uniroot(final_eq, lower=0, upper=1)$root
  result2 <- uniroot(final_eq, lower=1e-12, upper=1, extendInt = "yes")$root
  result <-max(result1,result2)
  return(result)
}
```

```
################################################################################
#  Compute tau for R_0 in [0,5]
#  Plot tau against R_0
################################################################################

# Create a vector of 10000 R0 values between 1.00001 and 5
R0_vec <- seq(from = 0, to = 5, length.out = 10000)
# Create a vector of the corresponding values of tau
tau_vec <- as.vector(10000)
for (i in 1:10000){
    R0 <- R0_vec[i]
    tau_vec[i] <- tau(R_0 = R0)
  }
# Create a plot of tau against R0
plot1a <- ggplot(mapping = aes(x = R0_vec, y = tau_vec)) +
  geom_line()  +
  xlab(expression('R'[0])) +
  ylab(expression(tau)) +
  ggtitle(expression("Final size as a function of R"[0]))
print(plot1a)
```

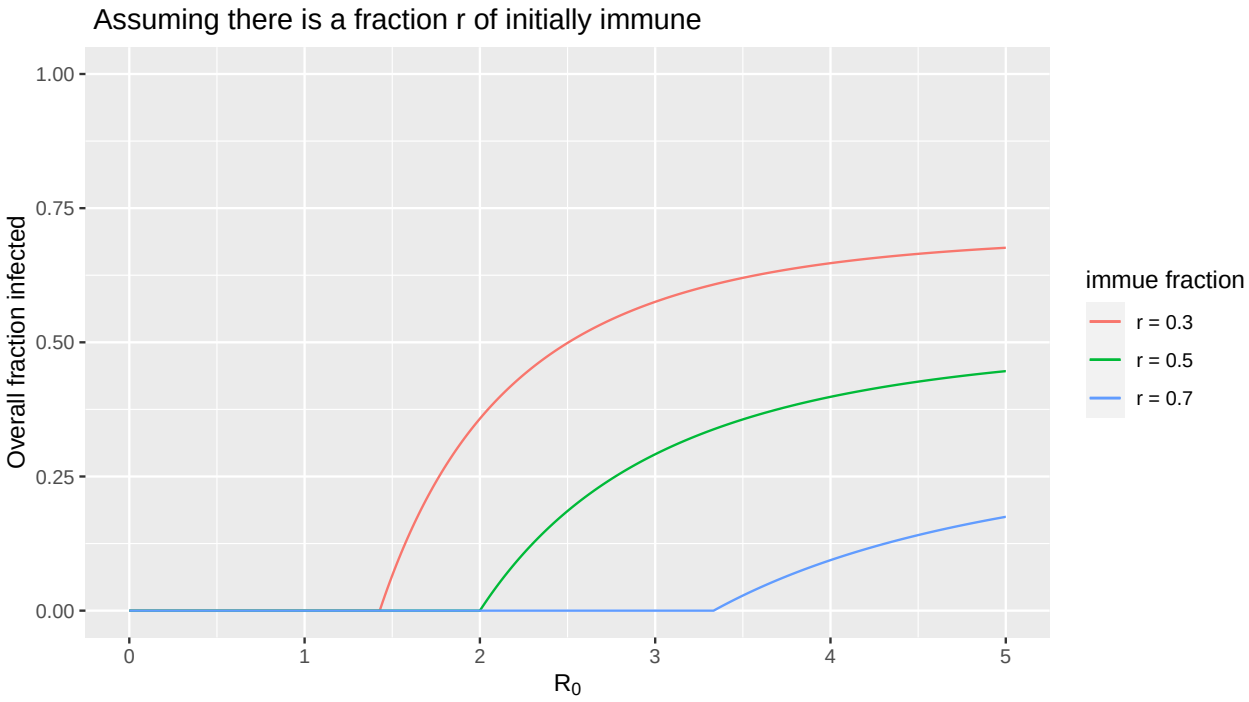


(b) Now suppose there is a fraction r of initially immune, then the final fraction infected among the initially susceptible, solves $1-\tau = \exp(-R_0(1-r)\tau)$. As in part (a), plot the **overall** fraction infected, as the function of R_0 in $[0,5]$ for $r = 30\%, 50\%, 70\%$.

```
################################################################################
# Function of R_0 solving for tau numerically, when there is initial immune
#  R_0 - reproduction number
#  r - fraction of initially immune
#  tau - overall final size
#  Return: tau
################################################################################
tau_overall <- function(R_0,r) {
  final_eq <- function(tau) {1 - tau - exp(-R_0*(1-r)*tau)}
  "
  Note: This final size equation only gives the fraction infected among
  those who are initially susceptible. The overall fraction infected shall
  be the solution*(1-r).
  "
  result1 <- uniroot(final_eq, lower=0, upper=1)$root
  result2 <- uniroot(final_eq, lower=1e-12, upper=1, extendInt = "yes")$root
  tau <-max(result1,result2)

  return(tau*(1-r))
}
```

```
################################################################################
# Plot the overall final size of R_0 in [0,5]
# in three cases: r = 0.3, 0.5, 0.7
################################################################################
# Create a vector of 10000 R0 values between 1.00001 and 5
R0_vec <- seq(from = 0, to = 5, length.out = 10000)
df_tau_r=data.frame(R0=R0_vec)
```

```r
# Create a vector of the corresponding values of tau for r = 0.3
tau_vec <- as.vector(10000)
r=0.3;
for (i in 1:10000){
    R0 <- R0_vec[i]
    tau_vec[i] <- tau_overall(R0,r)
}
df_tau_r= cbind(df_tau_r,overall_infected.3=tau_vec)
# Create a vector of the corresponding values of tau for r = 0.5
tau_vec <- as.vector(10000)
r=0.5;
for (i in 1:10000){
    R0 <- R0_vec[i]
    tau_vec[i] <- tau_overall(R0,r)
}
df_tau_r= cbind(df_tau_r,overall_infected.5=tau_vec)
# Create a vector of the corresponding values of tau for r = 0.7
tau_vec <- as.vector(10000)
r=0.7;
for (i in 1:10000){
    R0 <- R0_vec[i]
    tau_vec[i] <- tau_overall(R0,r)
}
df_tau_r= cbind(df_tau_r,overall_infected.7=tau_vec)
# Create a plot of tau against R0 in three cases
plot1b <- ggplot(df_tau_r)+
geom_line(aes(x=R0, y=overall_infected.3, color="r = 0.3")) +
geom_line(aes(x=R0, y=overall_infected.5, color="r = 0.5")) +
geom_line(aes(x=R0, y=overall_infected.7, color="r = 0.7")) +
scale_color_discrete(name="immue fraction")+
labs(title = " Assuming there is a fraction r of initially immune ") +
ylab("Overall fraction infected")+
xlab(expression('R'[0])) + ylim(0,1)

print(plot1b)
```

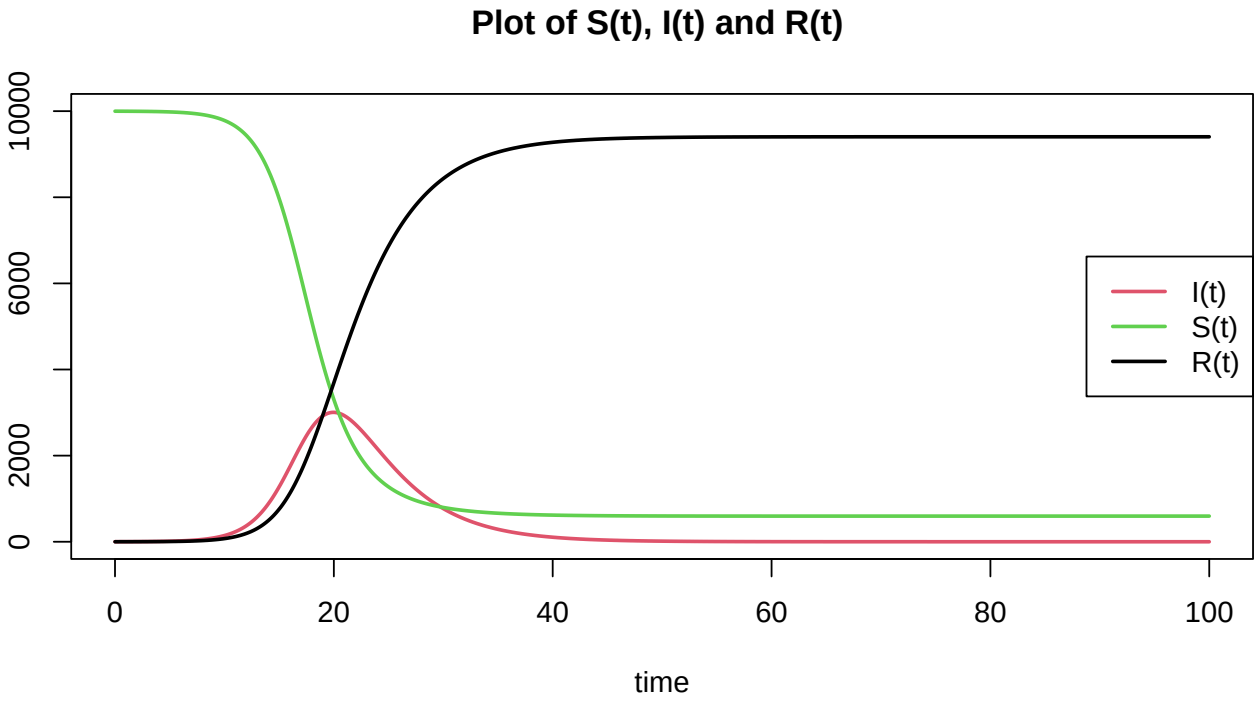


Exercise 1.2 (Determinstic and stochastic SIR Model)

(a) Consider a continuous-time deterministic SIR model with parameter values $\beta = 0.75$ and $\gamma = 0.25$. We assume a closed population of size $N = 10000$. Solve the SIR differential equation system with initial conditions: $S(0) = N - 1, I(0) = 1$ using R package *deSolve*. Plot the curves of $I(t)$ and $S(t)$ over time.

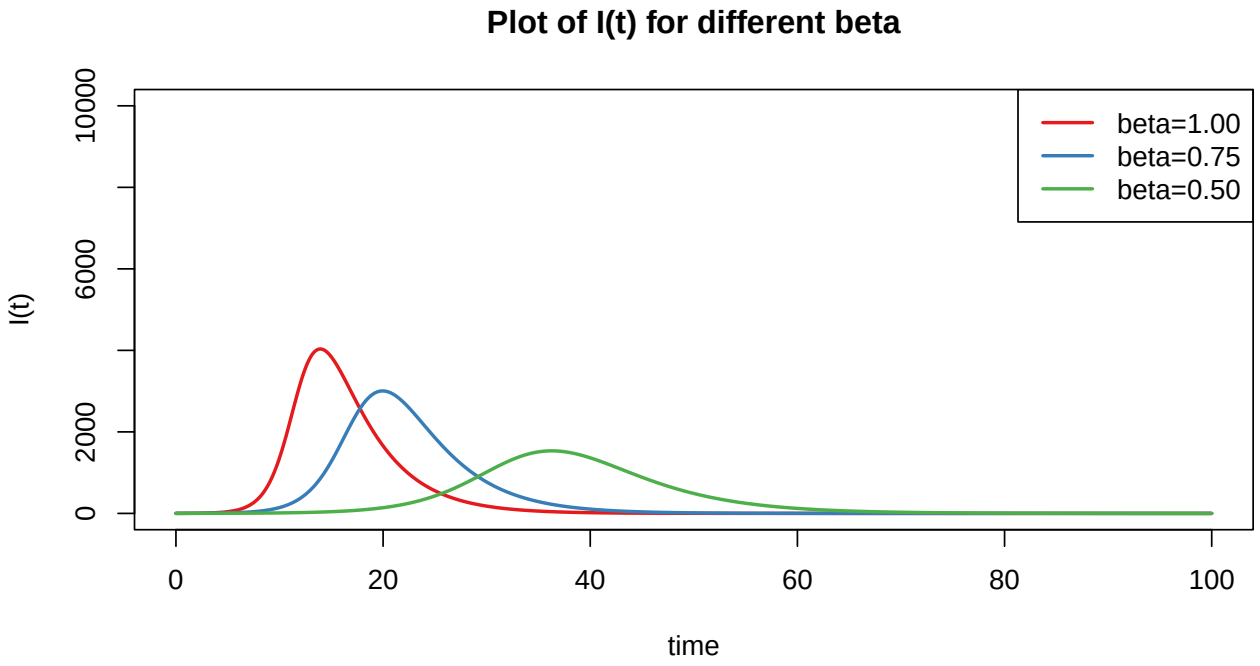
```
##############################################################################
# Function to compute the derivative of the ODE system for SIR Model:
# S(t): number of susceptible
# I(t): number of infectives
# ODE:
# dS(t)/dt = -beta/N*S(t)*I(t)
# dI(t)/dt = beta/N*S(t)*I(t)-gamma*I(t)
#
#  t - time
#  y - current state vector of the ODE at time t
#  parms - Parameter vector used by the ODE system
#
# Returns:
#  list containing dS(t)/dt and dI(t)/dt
##############################################################################
gamma <- 0.25
beta.grid <- c(1, 0.75, 0.5)
N <- 10000
times <- seq(0,100,length=1000)

deter_sir <- function(t,y, parms) {
  beta <- parms[1]
  gamma <- parms[2]
  S <- y[1]
  I <- y[2]
```

```
  return(list(c(S=-(beta*S*I)/N,I=(beta*S*I)/N-gamma*I)))
}

sim <- lsoda(y=c(N-1,1), times=times, func=deter_sir,
             parms=c(beta.grid[2],gamma))

plot(times, sim[,3],type="l", lwd=2, col=2,xlab="time",ylab="",
     xlim = c(0,100),ylim = c(0,N))
lines(times, sim[,2], col=3,lwd=2)
lines(times, N-sim[,3]-sim[,2], col=1,lwd=2)
legend(x="right",c("I(t)", "S(t)", "R(t)"), col=c(2,3,1),lty=1,lwd=2)
title("Plot of S(t), I(t) and R(t) ")
```



(b) Now fix $\gamma = 0.25$, but choose different values of $\beta = 1, 0.75$ and 0.25. In each case, solve the SIR differential equation system with initial condition $(S(0) = N - 1, I(0) = 1)$. Plot the curves of $I(t)$ and compare them.

```
I <- sapply(beta.grid, function(beta) {
  lsoda(y=c(N-1,1), times=times, func=deter_sir,parms=c(beta,gamma))[,3]
})
pal <- brewer.pal(length(beta.grid),"Set1")
matplot(times,I,type="l",lwd=2,lty=1,col=pal,xlab="time",ylab="I(t)", ylim=c(0, N))
legend(x="topright",paste("beta=",sprintf("%.2f",beta.grid),sep=""),lty=1,lwd=2,col=pal)
title("Plot of I(t) for different beta")
```



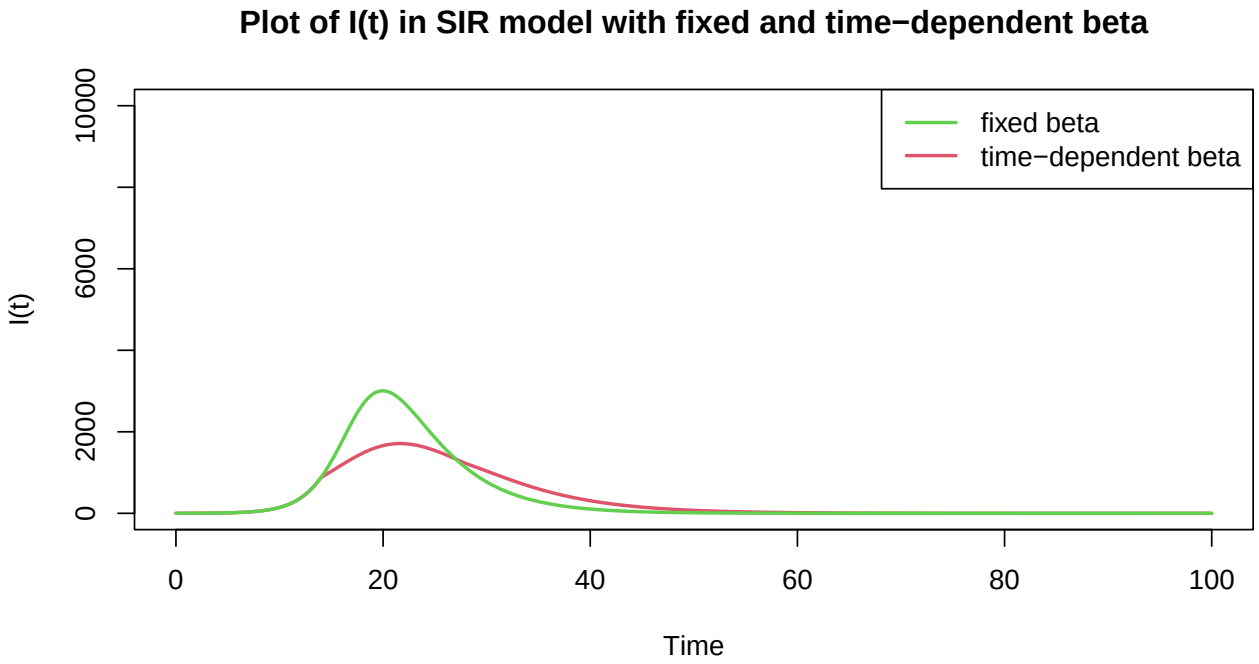
(c) Take $\beta = 0.75$ and $\gamma = 0.25$.(implying that $R_0 = 3$). There is one way to reduce R_0, reducing the number of contacts made by individuals, i.e. reducing β. We pursue the simple strategy, where the rate β depends on time when different measures take place. Within some time between t_1 and t_2, there are large reduction of contacts, and then the control measures(e.g. social distancing interventions) are slightly relaxed. To be more precise, we have

$$\beta(t) = \begin{cases} \beta_0 \text{ ,if } t \leq t_1, \\ \beta_1 \text{ ,if } t_1 < t \leq t_2, \\ \beta_2 \text{ ,if } t > t_2, \end{cases}$$

with β_0 the ordinary contact rate, $\beta_1 < \beta_2 < \beta_0$. Here we use $\beta_1 = r_1\beta_0$ and $\beta_2 = r_2\beta_0$ with $r_1 \leq r_2$. Take $r_1 = 0.65, r_2 = 0.75, t_1 = 14, t_2 = 28$. Assuming that size $N = 10000$ and there is one initial infective, plot the curve of $I(t)$.

```
N <- 10000
sir_change <- function(t, y, parms) {
  beta0 <- parms[1]
  beta1 <- parms[2]
  beta2 <- parms[3]
  t1 <- parms[4]
  t2 <- parms[5]
  gamma <- parms[6]

  S <- y[1]
  I <- y[2]


  # time-dependent rate \beta(t):

  beta <- function(t) {ifelse( t <= t1, beta0, ifelse(t > t2, beta2,
                                    beta1))
  }

  return(list(c(S=-(beta(t)*S*I)/N, I=(beta(t)*S*I)/N - gamma*I)))
```

```
}
beta0 <- 0.75
beta1 <- 0.65*0.75
beta2 <- 0.75*0.75
t1 <- 14
t2 <- 28
gamma <- 0.25
N <- 10000
I0 <- 1
time<-seq(0,100,length=1000)
sol_SIRchange <- lsoda(y=c(N-I0,I0), times=time, func=sir_change,
                       parms=c(beta0, beta1, beta2, t1, t2, gamma))

plot(time, sol_SIRchange [,3],type="l", lwd=2, col=2,xlab="Time",
     ylab="I(t)", xlim = c(0,100),ylim = c(0,N))
lines(times, sim[,3], col=3,lwd=2)
legend(x="topright", c("fixed beta","time-dependent beta"), col=c(3,2), lty=1,lwd=2)
title("Plot of I(t) in SIR model with fixed and time-dependent beta")
```

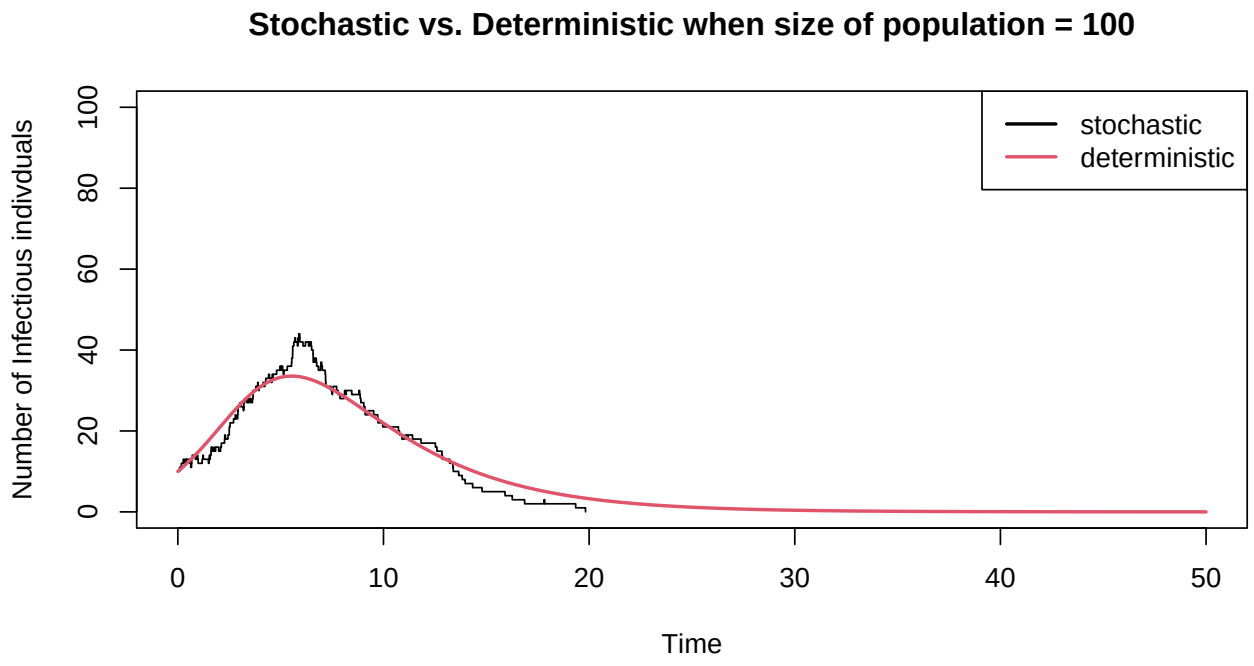


(d) Here we turn our focus to stochastic SIR model in continuous time. Always Assume that there are fraction $c = 10\%$ of initial infectives. Take $\beta = 0.75$ and $\gamma = 0.25$.(implying that $R_0 = 3$). Plot $I(t)$ for one (typical) simulated stochastic epidemic and deterministic limit for different size of population $N = 100, 1000$ and 10000.

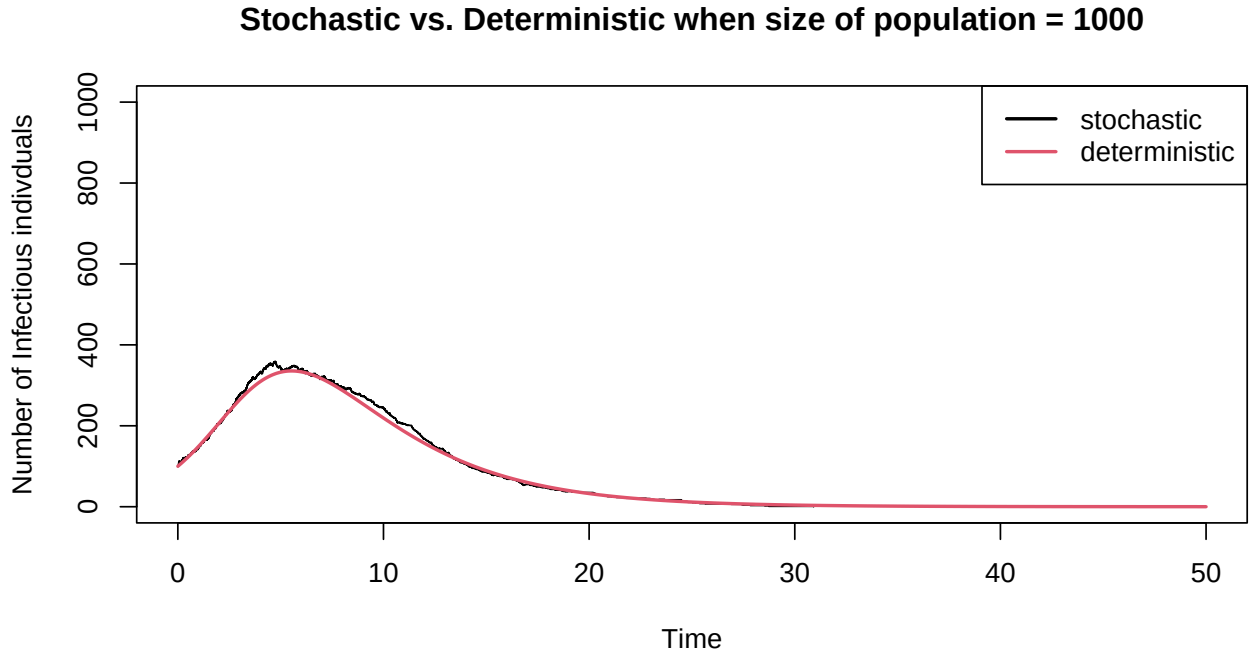
```
#############################################################################
# Simulate simple SIR model using the Gillespie-Algorithm
#
# Params:
#  [0,T] - time horizon
#  beta - infection rate
#  gamma - recovery rate
#  n - initial number of susceptibles.
#  m - initial number of infectives
```

```
#########################################################################

stoch_SIR <- function( T, beta, gamma, n, m) {
  #Initialize (x= number of S, y=number of I, t=event time)
  x <- n
  y <- m
  z <- 1
  t <- 0
  N <- n+m
  # Possible events: Infection(S->I) and Removal(I->R)
  eventLevels <- c("S->I","I->R")
  # Initialize result
  df_SIR <- data.frame(t=t,x=x,y=y,totalinfected=z,event=NA)
  # Loop until  time T or the epidemic stops(there is no infectives)
  while (t < T & (y>0)) {
    # Draw the waiting type for each possible event
    wait <-  rexp(2,c("S->I"=beta/N*x*y,"I->R"=gamma*y))
    # Determine which event occurs first
    i <- which.min(wait)
    # Record Event Time
    t <- t+wait[i]
    # Update the number of S and I according to the event type
    if (eventLevels[i]=="S->I") { x <- x-1 ; y <- y+1; z <- z+1} # if infection
    if (eventLevels[i]=="I->R") { y <- y-1 }# if recovery
    # Store result
    df_SIR <- rbind(df_SIR,c(t,x,y,z,i))
  }
  # Re-code event type and return
  df_SIR$event <- factor(eventLevels[df_SIR$event], levels=eventLevels)

  return(df_SIR)
}

# set parameter values:
beta <- 0.75
gamma <- 0.25
Tmax <- 50
c <- 0.1
compare_DS <- function(N) {

  traj <- stoch_SIR(T=Tmax, beta=beta, gamma=gamma, n=N*(1-c), m=N*c)

    plot(traj$t,traj$y,type="s",ylim=c(0,N),xlab="Time",
    ylab="Number of Infectious indivduals",xlim=c(0,Tmax),col=1)

  # Solve the deterministic ODE
  solution <- lsoda(y=c(N*(1-c),N*c), times=seq(0,Tmax,length=1000),
                    func=deter_sir,parms=c(beta,gamma))
  lines(solution[,"time"], solution[,3], col=2, lwd=2)

}
## Compare deterministic and stochastic for different N
N <- 100
```

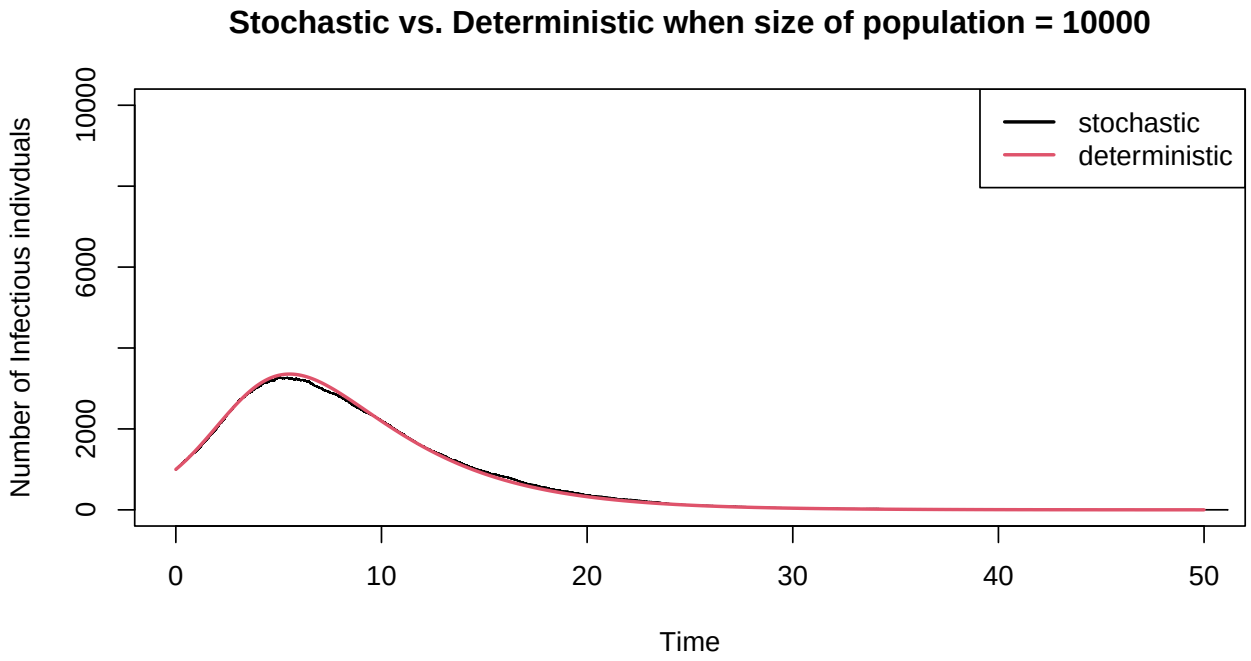
```
compare_DS(N=100)
legend(x="topright", c("stochastic", "deterministic"), col=c(1,2), lty=1,lwd=2)
title("Stochastic vs. Deterministic when size of population = 100")
```




```
N <- 1000
compare_DS(N=1000)
legend(x="topright", c("stochastic", "deterministic"), col=c(1,2), lty=1,lwd=2)
title("Stochastic vs. Deterministic when size of population = 1000")
```



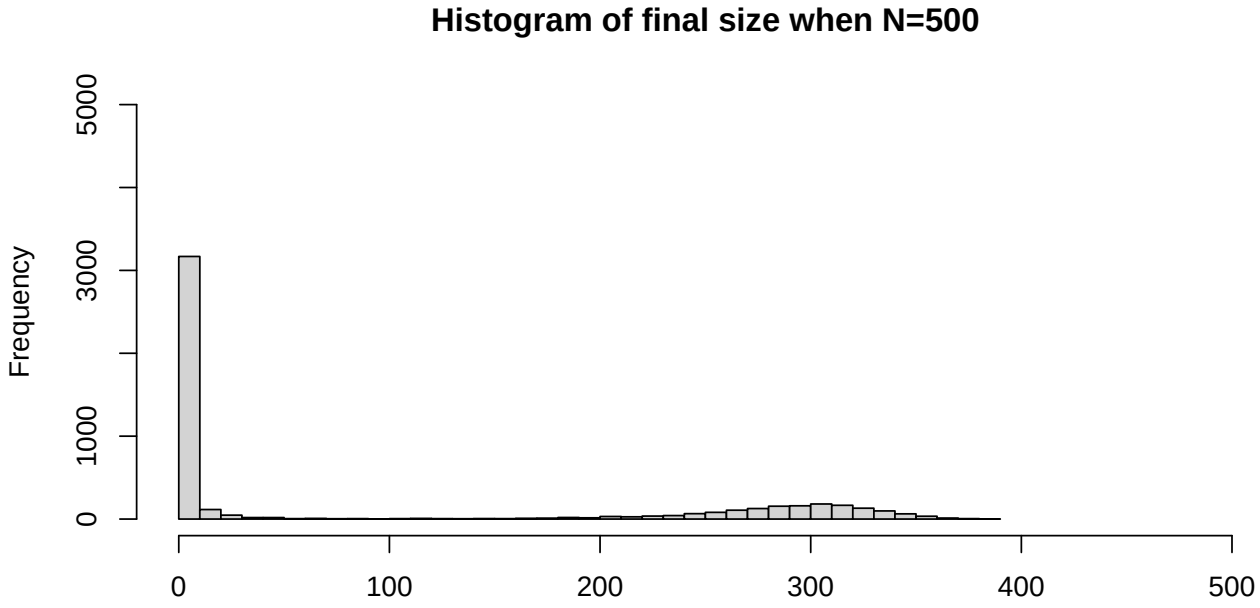
```
N <- 10000
compare_DS(N=10000)
legend(x="topright", c("stochastic", "deterministic"), col=c(1,2), lty=1,lwd=2)
title("Stochastic vs. Deterministic when size of population = 10000")
```



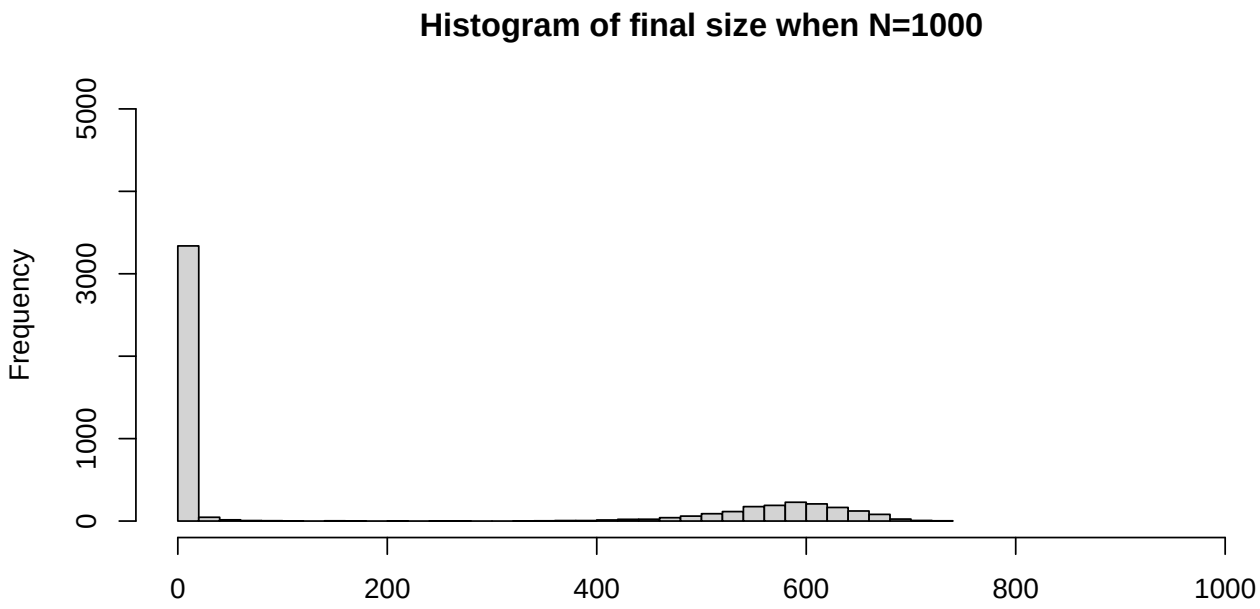
(e) Let $\beta = 0.375$ and $\gamma = 0.25$.(implying that $R_0 = 1.5$), do 5000 simulations in three cases when the size of population $N = 500, 1000$ and 5000 with one initial infective. Make a histogram of the final size distribution in each case. Give your comments.

```
# total infected:
fsize_SIR <- function( T, beta, gamma, n, m) {
  #Initialize (x= number of S, y=number of I, t=event time)
  x <- n
  y <- m
  z <- 0
  t <- 0
  N <- n+m
  # Possible events: Infection(S->I) and Removal(I->R)
  eventLevels <- c("S->I","I->R")
  # Loop until  time T or the epidemic stops(there is no infectives)
  while (t < T & (y>0)) {
    # Draw the waiting type for each possible event
    wait <-  rexp(2,c("S->I"=beta/N*x*y,"I->R"=gamma*y))
    # Determine which event occurs first
    i <- which.min(wait)
    # Record Event Time
    t <- t+wait[i]
    # Update the number of S and I according to the event type
    if (eventLevels[i]=="S->I") { x <- x-1 ; y <- y+1; z <- z+1} # if infection
    if (eventLevels[i]=="I->R") { y <- y-1 }# if recovery
  }
  return(z)
```

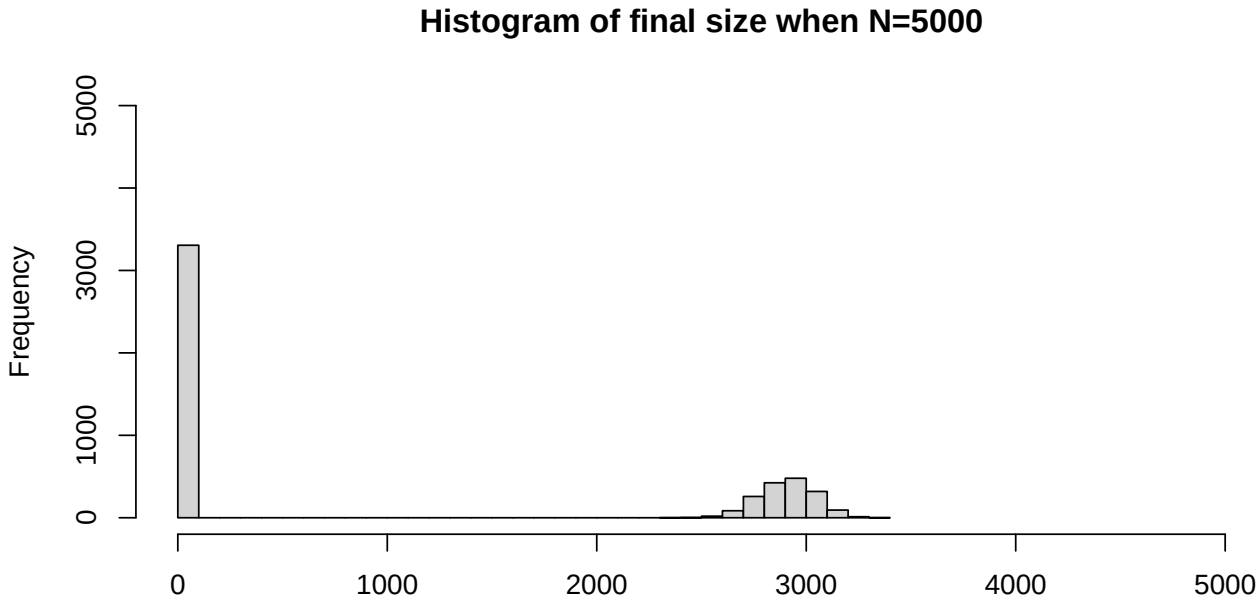
```
}
# set parameter values:
beta <- 0.375
gamma <- 0.25
Tmax <- 500
I0 <- 1
# Do 1000 simulations with different size of community N:
compare_hist <- function(N) {
  nSim <- 5000
  df_fsize <- data.frame(sim=NA, totalinfected=NA)
  for (i in 1:nSim) {

  z <- fsize_SIR(T=Tmax, beta=beta, gamma=gamma, n=N-I0, m=I0)

  df_fsize <- rbind(df_fsize,c(i,z))

  }
 return(df_fsize$totalinfected)
}
N <- 500
hist(compare_hist(N=500),breaks=50,main="Histogram of final size when N=500",
xlab="", xlim=c(0,500),ylim=c(0,5000))
```




```
N <- 1000
hist(compare_hist(N=1000),breaks=50,main="Histogram of final size when N=1000",
xlab="",xlim=c(0,1000),ylim=c(0,5000))
```




```
N <- 5000
hist(compare_hist(N=5000),breaks=45,main="Histogram of final size when N=5000",
xlab="",xlim=c(0,5000),ylim=c(0,5000))
```



Exercise 1.3 (SEIR Model)

(a) Consider the SEIR (Susceptible->Exposed->Infectious->Recovered) model in a closed population with size $N = 100$, rate of infection $\beta = 0.4$, rate of recovery $\gamma = 1/7$ and the rate for the E $\rightarrow$ I transition is $\rho = 1/5$, implying a latency period with mean 5 days. Write up the ordinary differential equation

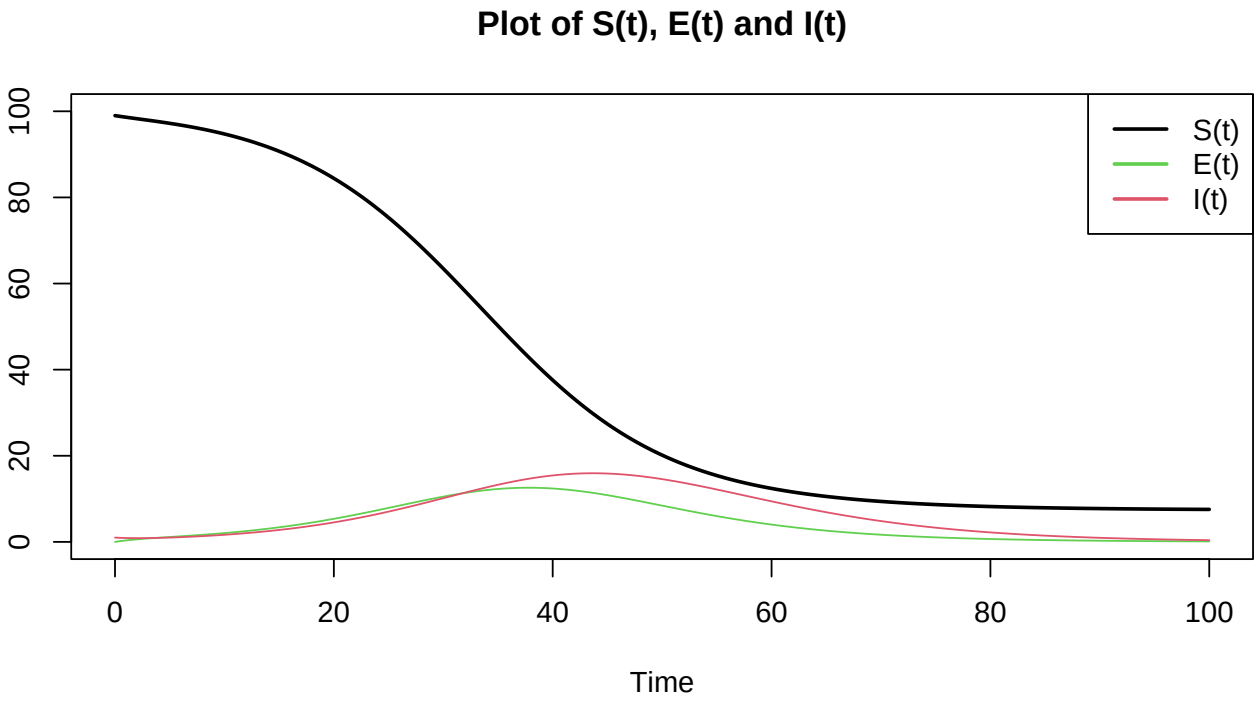
system for the above SEIR model:

$$\begin{aligned}
\frac{\partial S(t)}{\partial t} &= -\frac{\beta}{N} S(t)I(t) = -0.004 S(t)I(t), \\
\frac{\partial E(t)}{\partial t} &= \frac{\beta}{N} S(t)I(t) - \rho E(t) = 0.004 S(t)I(t) - \frac{1}{5} E(t), \\
\frac{\partial I(t)}{\partial t} &= \rho E(t) - \gamma I(t) = \frac{1}{5} E(t) - \frac{1}{7} I(t), \\
\frac{\partial R(t)}{\partial t} &= \gamma I(t) = \frac{1}{7} I(t).
\end{aligned}$$

```
################################################################################
# Function to compute the derivative of the ODE system for SEIR Model
#
#  t - time
#  y - current state vector of the ODE at time t
#  parms - Parameter vector used by the ODE system
#
# Returns:
#  list containing dS(t)/dt, dE(t)/dt, and dI(t)/dt
################################################################################
N <- 100
seir <- function(t, y, parms) {
  beta <- parms[1]
  rho <- parms[2]
  gamma <- parms[3]
  S <- y[1]
  E <- y[2]
  I <- y[3]
  return(list(c(S=-beta/N*S*I, E=beta/N*S*I-rho*E, I=rho*E - gamma*I)))
}
```

Assume that there is one initial infective, i.e. $I(0) = 1$ and find a numerical approximation for $I(t), S(t)$ and $E(t)$. Show a plot of $S(t), E(t)$ and $I(t)$ in $[0, 100]$.

```
# Parameter Values:
times <- seq(0,100,length=1000)
gamma <- 1/7
beta <- 0.4
rho <- 1/5
I0 <- 1
N <- 100
# Solve the ODE and plot
sol_seir <- lsoda(y=c(N-I0, 0, I0), times=times, func=seir,parms=c(beta,rho,gamma))
plot(times, sol_seir[,2],type="l", lwd=2, col=1,xlab="Time",
     ylab="", xlim = c(0,100),ylim = c(0,N))
lines(times, sol_seir[,3], col=3)
lines(times, sol_seir[,4], col=2)
legend(x="topright", c("S(t)", "E(t)", "I(t)"), col=c(1,3,2), lty=1,lwd=2)
title("Plot of S(t), E(t) and I(t)")
```



(b) Modify the SEIR equations such that $\beta(t)$ becomes a time dependent function, which is β_0 until time $t_1 - w$, is β_1 after time $t_1 + w$, and changes linearly from β_0 to β_1 between $t_1 - w$ and $t_1 + w$. Write the full $\beta(t)$. Hint: Determine what "??" should be in the equation below.

$$\beta(t) = \begin{cases} \beta_0 \text{ ,if } t \leq t_1 - w, \\ ?? \text{ ,if } t_1 - w < t \leq t_1 + w, \\ \beta_1 \text{ ,if } t > t_1 + w. \end{cases}$$

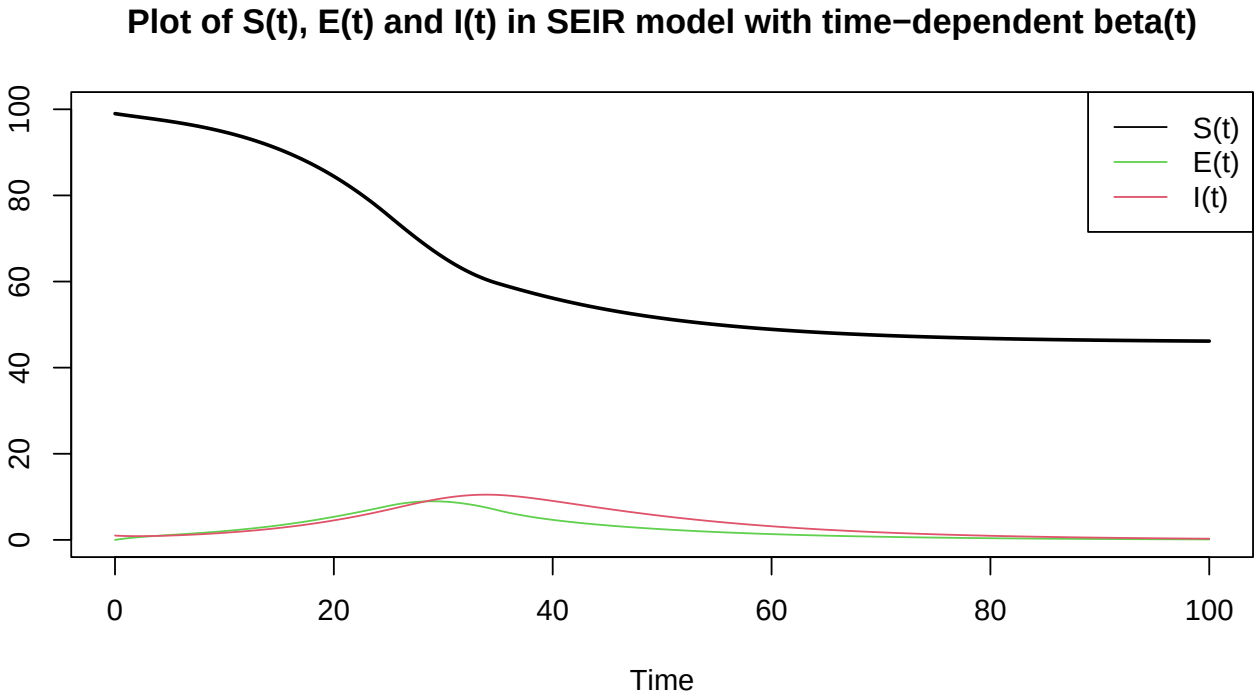
We know that between the times $t_1 - w$ and $t_1 + w$, the $\beta(t)$ function changes linearly between β_0 and β_1. So, we can solve "??" is by using the simple linear equation. Then, we can describe "??" as $\beta_0 + \frac{\beta_1 - \beta_0}{2w} t - \frac{\beta_1 - \beta_0}{2w}(t_1 - w)$.

```
#############################################################################################
# Function to compute the derivative of the ODE system for SEIR Model with changing point
#
#  t - time
#  y - current state vector of the ODE at time t
#  parms - Parameter vector used by the ODE system
#
# Returns:
#  list containing dS(t)/dt, dE(t)/dt, and dI(t)/dt
#############################################################################################
N <- 100
seir_change <- function(t, y, parms) {
  beta0 <- parms[1]
  beta1 <- parms[2]
  t1 <- parms[3]
  w <- parms[4]
  rho <- 1/5
  gamma <- parms[5]

  S <- y[1]
```

```
  E <- y[2]
  I <- y[3]

  # time-dependent rate \beta(t):

  beta <- function(t) {ifelse( t <= t1-w, beta0, ifelse(t > t1+w, beta1,
                                    beta0 + (beta1-beta0)/(2*w)*(t-(t1-w))))
  }

  return(list(c(S=-(beta(t)*S*I)/N, E=(beta(t)*S*I)/N-rho*E, I=rho*E - gamma*I)))
}
# plot \beta(t):
#plot(time, b, type="l",xlab="Time",ylab="beta(t)")
#title("Plot of beta(t)")
```

(c) let $t_1 = 30, w = 5, \beta_0 = 0.4$ and $\beta_1 = 0.12$, solve the ODE system for the SEIR model on time $[0, 100]$ with time-varying transmission rate numerically and plot $S(t), E(t)$ and $I(t)$ for $t \in [0, 100]$.

```
beta0 <- 0.4
beta1 <- 0.12
t1 <- 30
w <- 5
gamma <- 1/7
N <- 100
I0 <- 1
time<-seq(0,100,length=1000)
sol_SEIRchange <- lsoda(y=c(N-I0, 0, I0), times=time, func=seir_change,
                        parms=c(beta0, beta1, t1, w, gamma))
plot(times, sol_SEIRchange [,2],type="l", lwd=2, col=1,xlab="Time",
     ylab="", xlim = c(0,100),ylim = c(0,N))
lines(times, sol_SEIRchange [,3], col=3)
lines(times, sol_SEIRchange [,4], col=2)
legend(x="topright", c("S(t)", "E(t)", "I(t)"), col=c(1,3,2), lty=1)
title("Plot of S(t), E(t) and I(t) in SEIR model with time-dependent beta(t)")
```

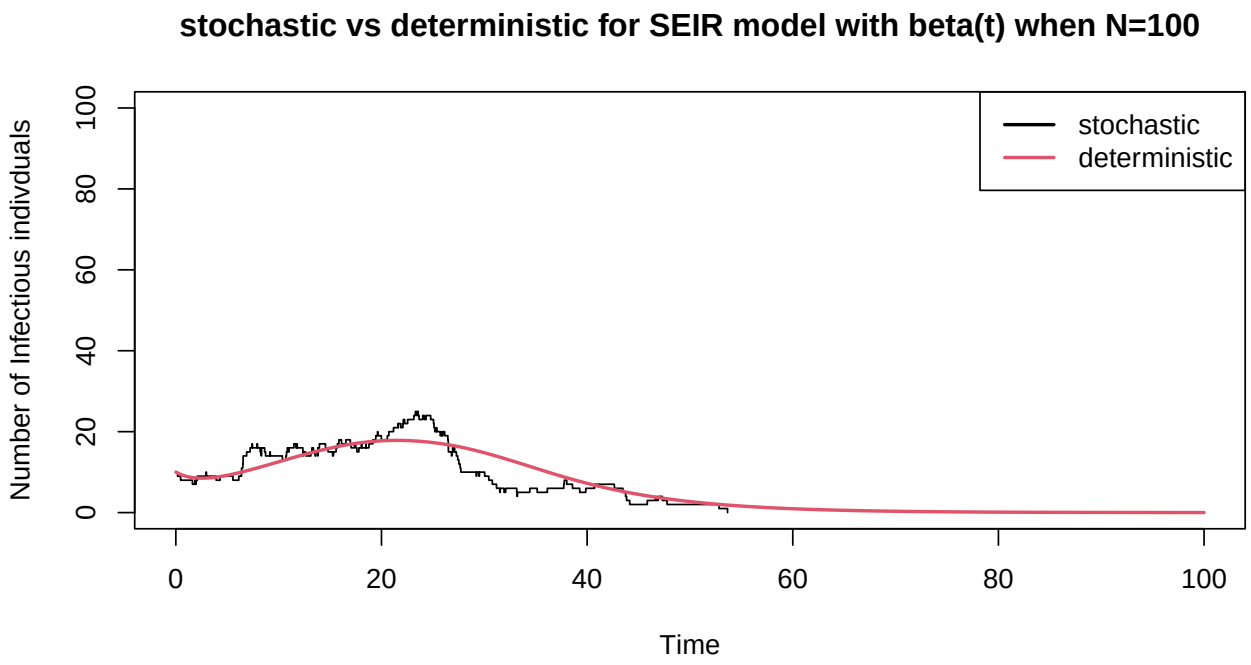


(d) Now for $N = 100, 1000$ and 10000, do one simulation of the stochastic SEIR epidemic starting from fraction infected with exponentially distributed incubation period with mean 5 days and the above time-changing $\beta(t)$. Overlay it on the plot of the deterministic curve as done in Exercise 2.

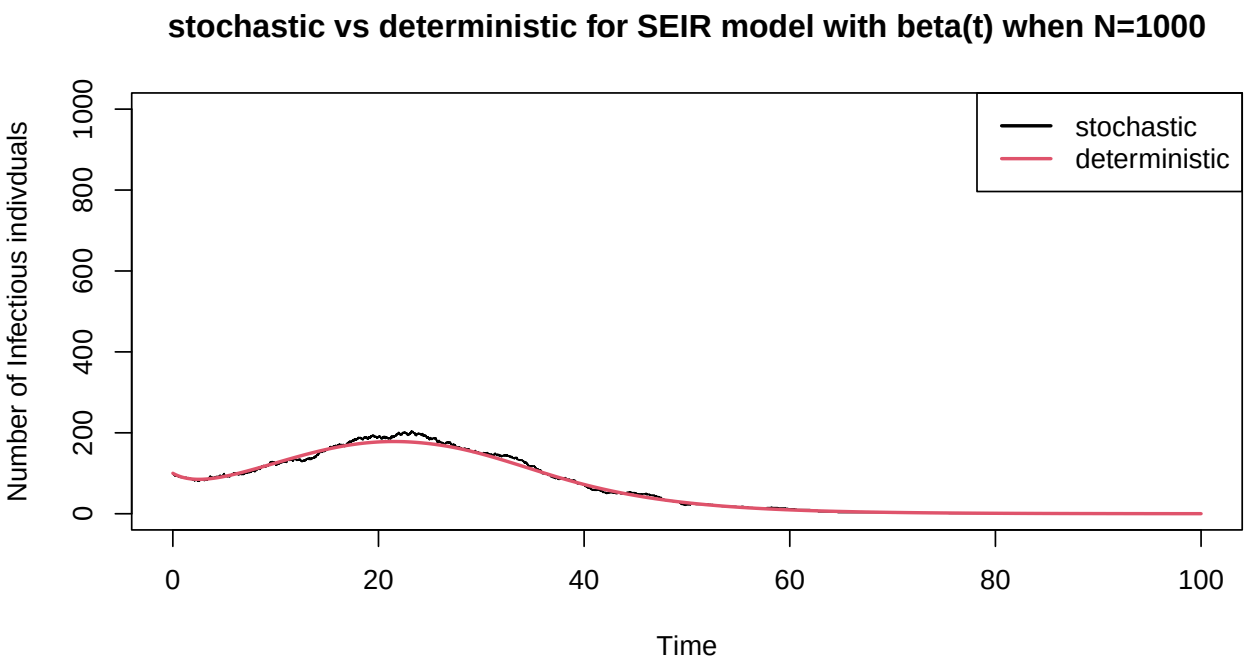
```
#########################################################################
# Simulate stochastic SEIR epidemic model
#
# Params:
#  [0,T] - time horizon
#  beta - infection rate
#  gamma - recovery rate
#  rho - rate of (E -> I)
#  n - initial number of susceptibles.
#  m - initial number of infectives, here m=1
#########################################################################


stoch_SEIR <- function( T, n, m, parms) {

  beta0 <- parms[1]
  beta1 <- parms[2]
  t1 <- parms[3]
  w <- parms[4]
  gamma <- parms[5]
  rho <- parms[6]

  beta <- function(t) {ifelse( t <= t1-w, beta0, ifelse(t > t1+w, beta1,
                                      beta0 + (beta1-beta0)/(2*w)*(t-(t1-w))))
}
  #Initialize (x= number of S, y=number of I, z=number of E, t=event time)
  x <- n
  y <- m
```

```
  z <- 0
  time <- 0
  # Possible events:
  eventLevels <- c("S->E","E->I","I->R")
  # Initialize result
  df_SEIR <- data.frame(time=time,x=x,y=y,z=z,event=NA)
  # Loop until  time T or the epidemic stops(there is no infectives)
  while (time < T & (y>0)) {
    # Draw the waiting type for each possible event
    b <- beta(time)
    wait <-  rexp(3,c("S->E"=(b*x*y)/N,"E->I"=rho*z,"I->R"=gamma*y))
    # Determine which event occurs first
    i <- which.min(wait)
    # Record Event Time
    time <- time+wait[i]
    # Update the number of S, I, E according to the event type
    if (eventLevels[i]=="S->E") { x <- x-1 ; z <- z+1}
    if (eventLevels[i]=="E->I") { z <- z-1 ; y <- y+1}
    if (eventLevels[i]=="I->R") { y <- y-1 }
    # Store result
    df_SEIR <- rbind(df_SEIR,c(time,x,y,z,i))
  }
  # Re-code event type and return
  df_SEIR$event <- factor(eventLevels[df_SEIR$event], levels=eventLevels)

  return(df_SEIR)
}

beta0 <- 0.4
beta1 <- 0.12
t1 <- 30
w <- 5
gamma <- 1/7
rho <- 1/5
c<- 0.1

compare_ds_seir <- function(N){

  beta <- function(t) {ifelse( t <= t1-w, beta0, ifelse(t > t1+w, beta1,
                                    beta0 + (beta1-beta0)/(2*w)*(t-(t1-w))))
  }
  # Do  stochastic simulations

  traj <- stoch_SEIR(T=100, n=N*(1-c), m=N*c,parms=c(beta0, beta1, t1, w, gamma, rho))
    plot(traj$time,traj$y,type="s",ylim=c(0,N),xlab="Time",
    ylab="Number of Infectious indivduals",xlim=c(0,100),col=1)

  # Solve the deterministic ODE

  solution <- lsoda(y=c(N*(1-c), 0, N*c), times=seq(0,100,length=1000),
                    func=seir_change,parms=c(beta0, beta1, t1, w, gamma))
  lines(solution[,"time"], solution[,4], col=2, lwd=2)
  invisible()
```

```
}

  # plot stochastic and deterministic
  N <- 100
  compare_ds_seir(N=100)
  title("stochastic vs deterministic for SEIR model with beta(t) when N=100")
  legend(x="topright", c("stochastic", "deterministic"), col=c(1,2), lty=1,lwd=2)
```




```
  N <- 1000
  compare_ds_seir(N=1000)
  title("stochastic vs deterministic for SEIR model with beta(t) when N=1000")
  legend(x="topright", c("stochastic", "deterministic"), col=c(1,2), lty=1,lwd=2)
```




```
N <- 10000
compare_ds_seir(N=10000)
title("stochastic vs deterministic for SEIR model with beta(t) when N=10000")
legend(x="topright", c("stochastic", "deterministic"), col=c(1,2), lty=1,lwd=2)
```