Numerical solution of stochastic epidemiological models: Solutions

John M. Drake & Pejman Rohani

Exercise 1. Simulate the stochastic SI model using Gillespie's direct method. Experiment with the initial number of infecteds (Y_0) and with the total population size (N). What effects do these have on the predictability of the epidemic? What effects do these have on the variability of the final outbreak size?

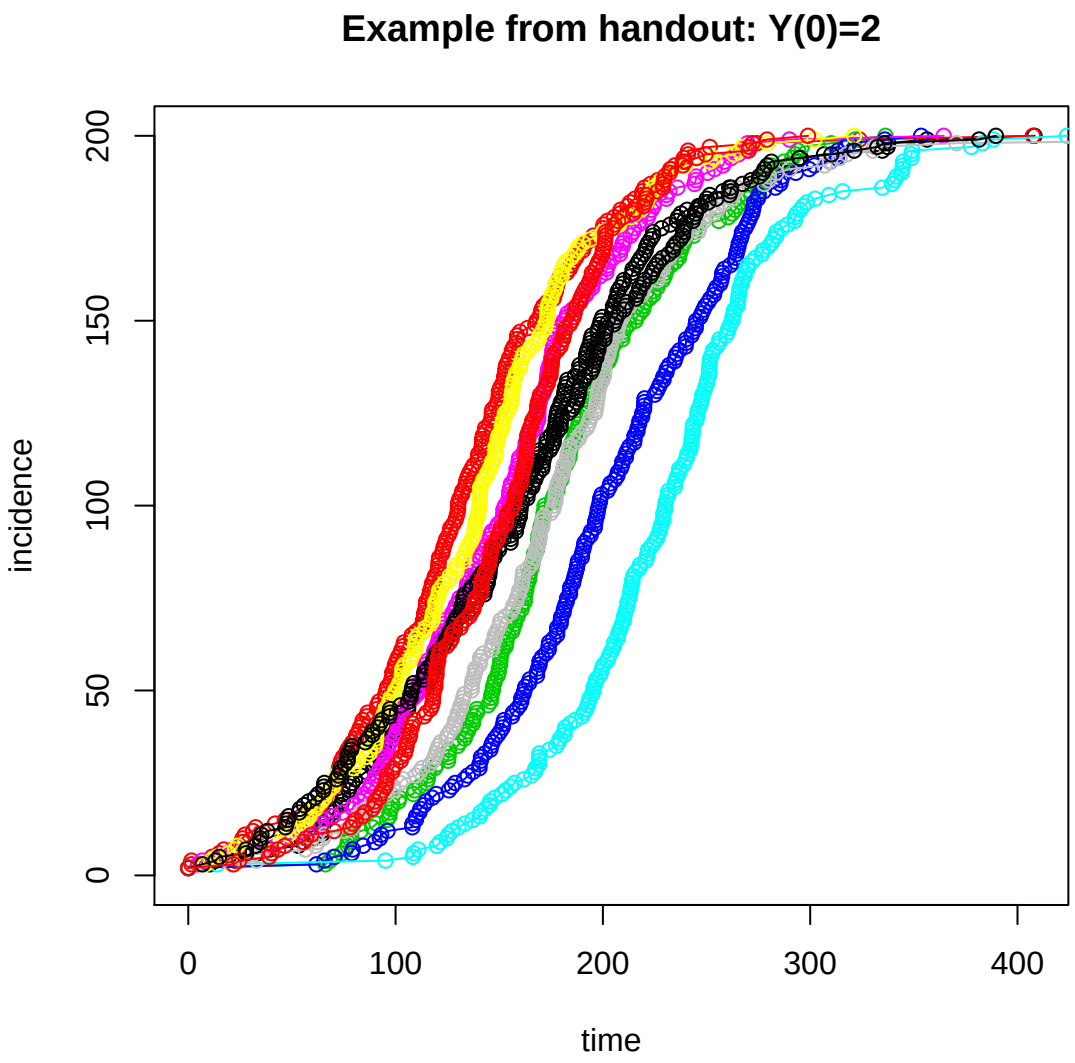
To solve this problem we require the functions introduced in the handout, namely `SI.onestep` and `SI.model`.

```
> SI.onestep <- function (x, beta) {     #function for one step of the stochastic SI epidemic
+    X <- x[2]                            #the second element of x is number of susceptibles X
+    Y <- x[3]                            #the third element of x is number of infecteds Y
+    new.Y <- Y+1                         #whenever an event occurs we increase infecteds by 1...
+    new.X <- X-1                         #and descrease susceptibles by 1
+    tau <- rexp(n=1,rate=beta*X*Y/(X+Y))  #exponential random time to next event
+    c(tau=tau,X=new.X,Y=new.Y)           #store result
+ }
> SI.model <- function (x, beta, nstep) { #function to iterate the stochastic SI for nstep events
+   output <- array(dim=c(nstep+1,3))       #set up an array to store all the results
+   colnames(output) <- c("time","X","Y")   #name the variables in the array
+   output[1,] <- x                         #the first record of the array is the initial condition
+   for (k in 1:nstep) {                    #iterate the model for nstep events
+     output[k+1,] <- x <- SI.onestep(x,beta) #update x and store result
+   }
+   output                                  #return output
+ }
```

Here, for comparison, we repeat the example in the handout.

```
> set.seed(38499583)                        #set the random seed so results are repeatable
> nsims <- 10                               #number of simulations to run
> pop.size <- 200                           #total size of the population
> Y0 <- 2                                   #initial number infected
> nstep <- pop.size-Y0                      #how many steps to run? until everyone infected
> xstart <- c(time=0,X=(pop.size-Y0),Y=Y0)  #initial conditions
> beta <- c(beta=3e-2)                     #transmission rate
> data <- vector(mode='list',length=nsims)  #create a list called ``data'' to store all runs
> for (k in 1:nsims) {                      #simulate k different runs
+   data[[k]] <- as.data.frame(SI.model(xstart,beta,nstep))  #main simulation step
+   data[[k]]$cum.time <- cumsum(data[[k]]$time) #calculates the running sum of inter-event intervals
+ }
```

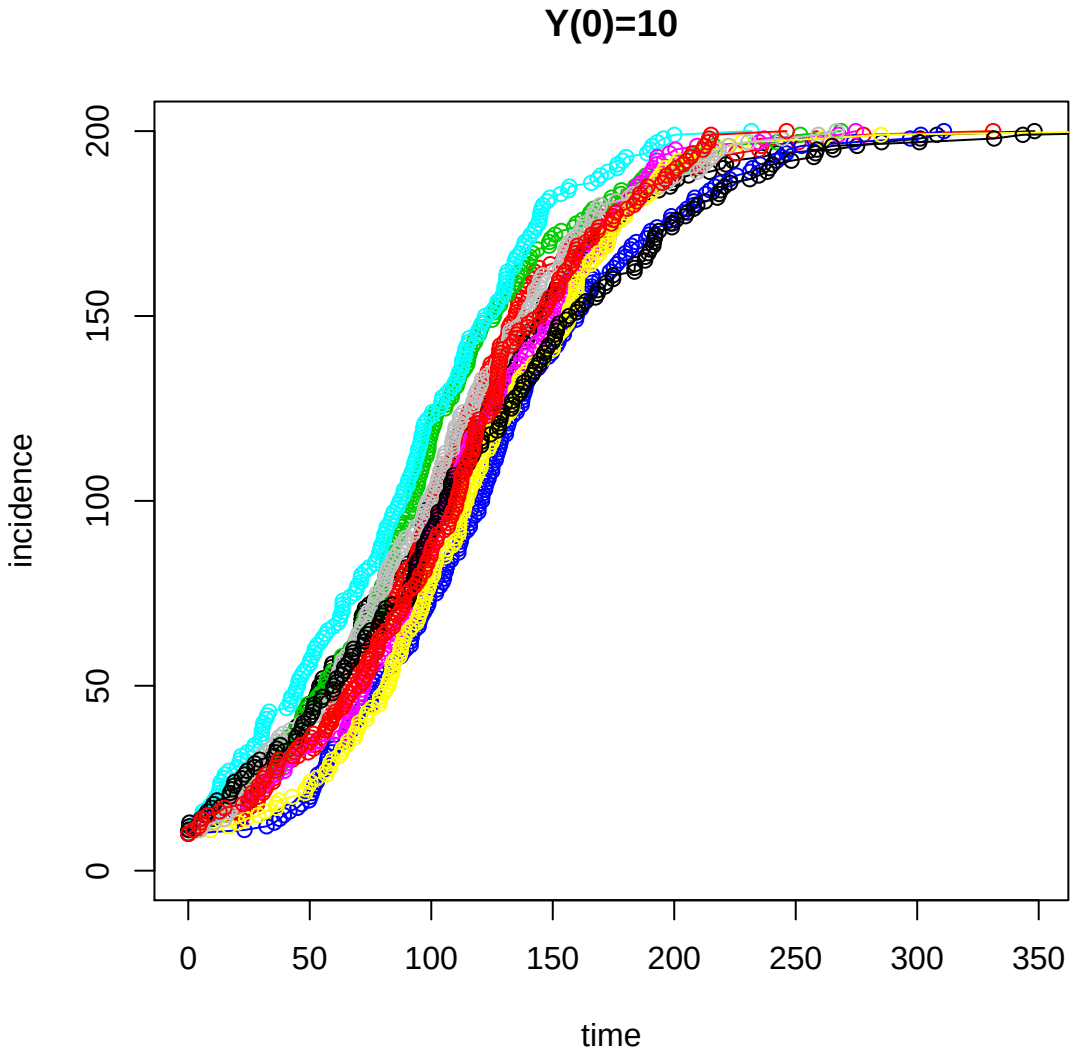
```
> max.y<-max(data[[1]]$cum.time)               #find the maximum time any simulation ran (to set x axis)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y),main='Example i
> for (k in 1:nsims) {                         #loop over each simulation...
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')  #to plot
+ }
```



Now, we change the initial number of infected individuals to $Y(0) = 10$.

```
> #Same parameter but 10 individuals initially infected
> set.seed(38499583)                           #set the random seed so results are repeatable
> nsims <- 10                                  #number of simulations to run
> pop.size <- 200                              #total size of the population
> Y0 <- 10                                      #initial number infected
> nstep <- pop.size-Y0                         #how many steps to run? until everyone infected
> xstart <- c(time=0,X=(pop.size-Y0),Y=Y0)  #initial conditions
```

```
> beta <- c(beta=3e-2)                    #transmission rate
> data <- vector(mode='list',length=nsims)  #create a list called ``data'' to store all runs
> for (k in 1:nsims) {                     #simulate k different runs
+   data[[k]] <- as.data.frame(SI.model(xstart,beta,nstep))  #main simulation step
+   data[[k]]$cum.time <- cumsum(data[[k]]$time) #calculates the running sum of inter-event intervals
+ }
> max.y<-max(data[[1]]$cum.time)              #find the maximum time any simulation ran (to set x axis)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y),main='Y(0)=10')
> for (k in 1:nsims) {                     #loop over each simulation...
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')  #to plot
+ }
```

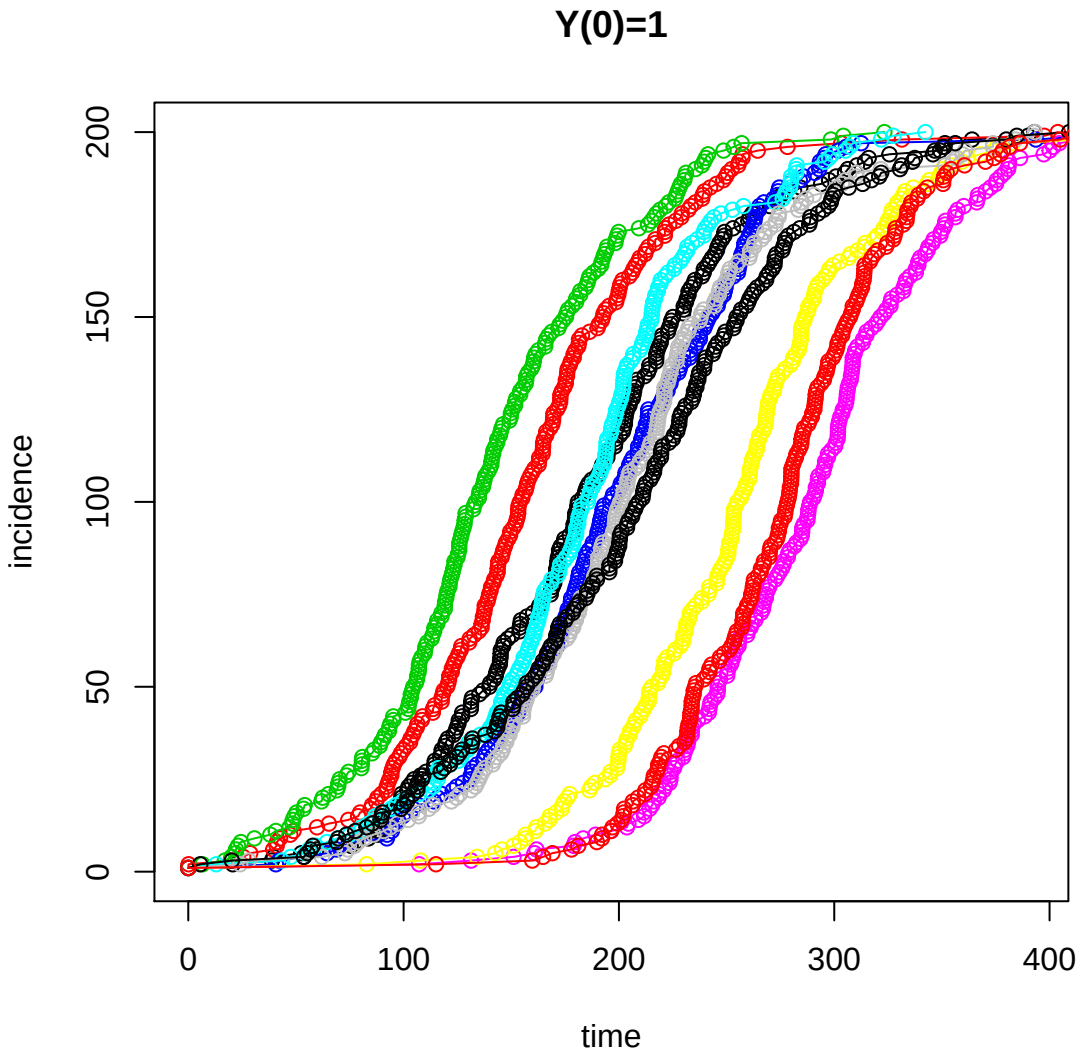


We observe that the total epidemic size does not change. After all, this is an SI epidemic. Since there is no recovery, everyone will be infected eventually. However, as the initial number infected gets larger the variation in the realized epidemic trajectories gets small. Thus, the trajectories for $Y(0) = 10$ fall

within a much tighter band than the trajectories for $Y(0) = 1$.

Now, for comparison, we run again with $Y(0) = 1$. As expected, we see that the variation in realized trajectories is greater than for $Y(0) = 10$ and even greater than for $Y(0) = 2$.

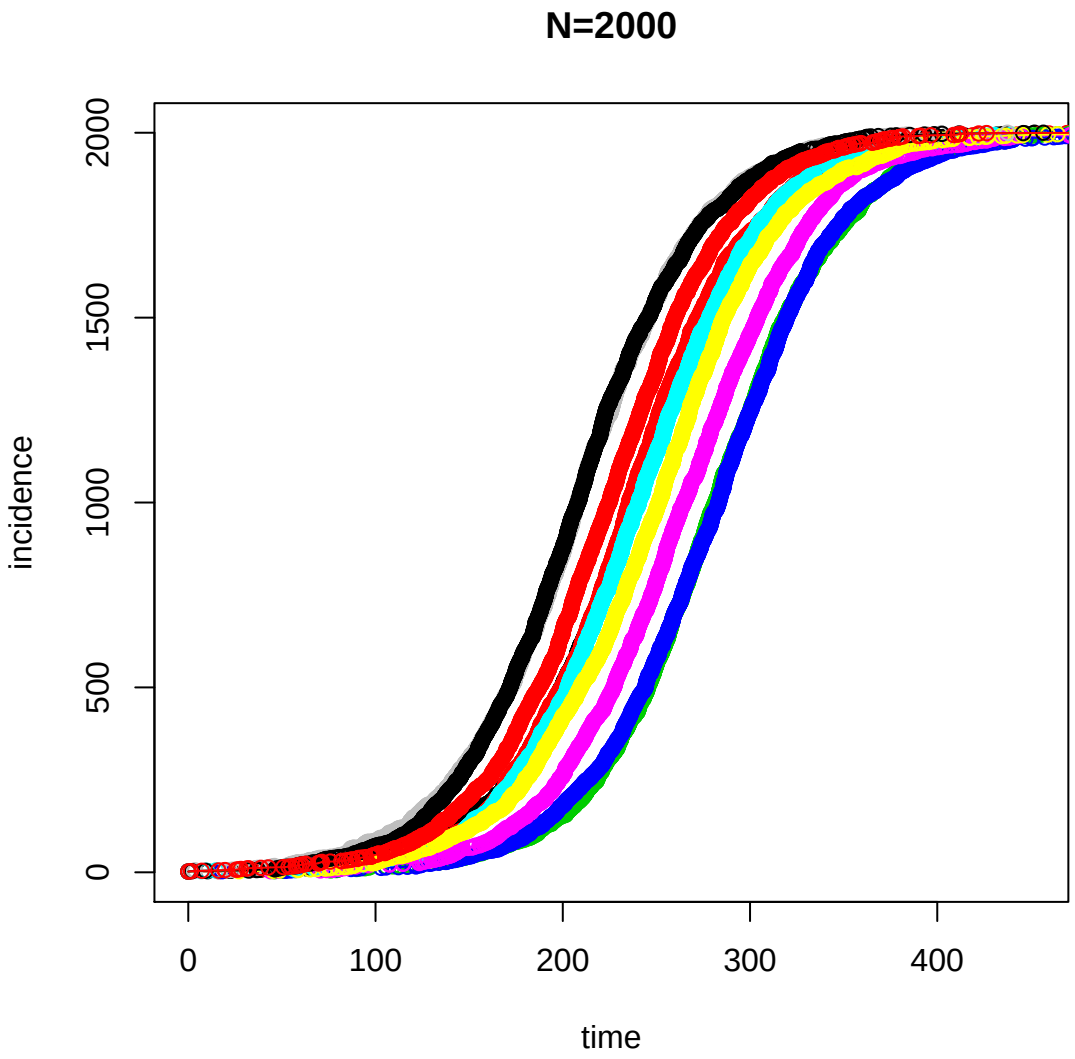
```
> #Same parameter but 1 individuals initially infected
> set.seed(38499583)                          #set the random seed so results are repeatable
> nsims <- 10                                 #number of simulations to run
> pop.size <- 200                             #total size of the population
> Y0 <- 1                                     #initial number infected
> nstep <- pop.size-Y0                        #how many steps to run? until everyone infected
> xstart <- c(time=0,X=(pop.size-Y0),Y=Y0)  #initial conditions
> beta <- c(beta=3e-2)                      #transmission rate
> data <- vector(mode='list',length=nsims)  #create a list called ``data'' to store all runs
> for (k in 1:nsims) {                        #simulate k different runs
+   data[[k]] <- as.data.frame(SI.model(xstart,beta,nstep))  #main simulation step
+   data[[k]]$cum.time <- cumsum(data[[k]]$time) #calculates the running sum of inter-event intervals
+ }
> max.y<-max(data[[1]]$cum.time)               #find the maximum time any simulation ran (to set x axis)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y), main='Y(0)=1')
> for (k in 1:nsims) {                        #loop over each simulation...
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')  #to plot
+ }
```



Now we investigate the effect of changing population size. Here we start with a total population size of $N = 2000$ (retaining the initial number infected $Y(0) = 2$).

```
> #Same parameters but population size of 2000
> set.seed(38499583)                          #set the random seed so results are repeatable
> nsims <- 10                                 #number of simulations to run
> pop.size <- 2000                             #total size of the population
> Y0 <- 2                                     #initial number infected
> nstep <- pop.size-Y0                        #how many steps to run? until everyone infected
> xstart <- c(time=0,X=(pop.size-Y0),Y=Y0)  #initial conditions
> beta <- c(beta=3e-2)                       #transmission rate
> data <- vector(mode='list',length=nsims)  #create a list called ``data'' to store all runs
> for (k in 1:nsims) {                        #simulate k different runs
+   data[[k]] <- as.data.frame(SI.model(xstart,beta,nstep))  #main simulation step
+   data[[k]]$cum.time <- cumsum(data[[k]]$time) #calculates the running sum of inter-event intervals
```

```
+ }
> max.y<-max(data[[1]]$cum.time)                #find the maximum time any simulation ran (to set x axis)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y), main='N=2000')
> for (k in 1:nsims) {                          #loop over each simulation...
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')  #to plot
+ }
```

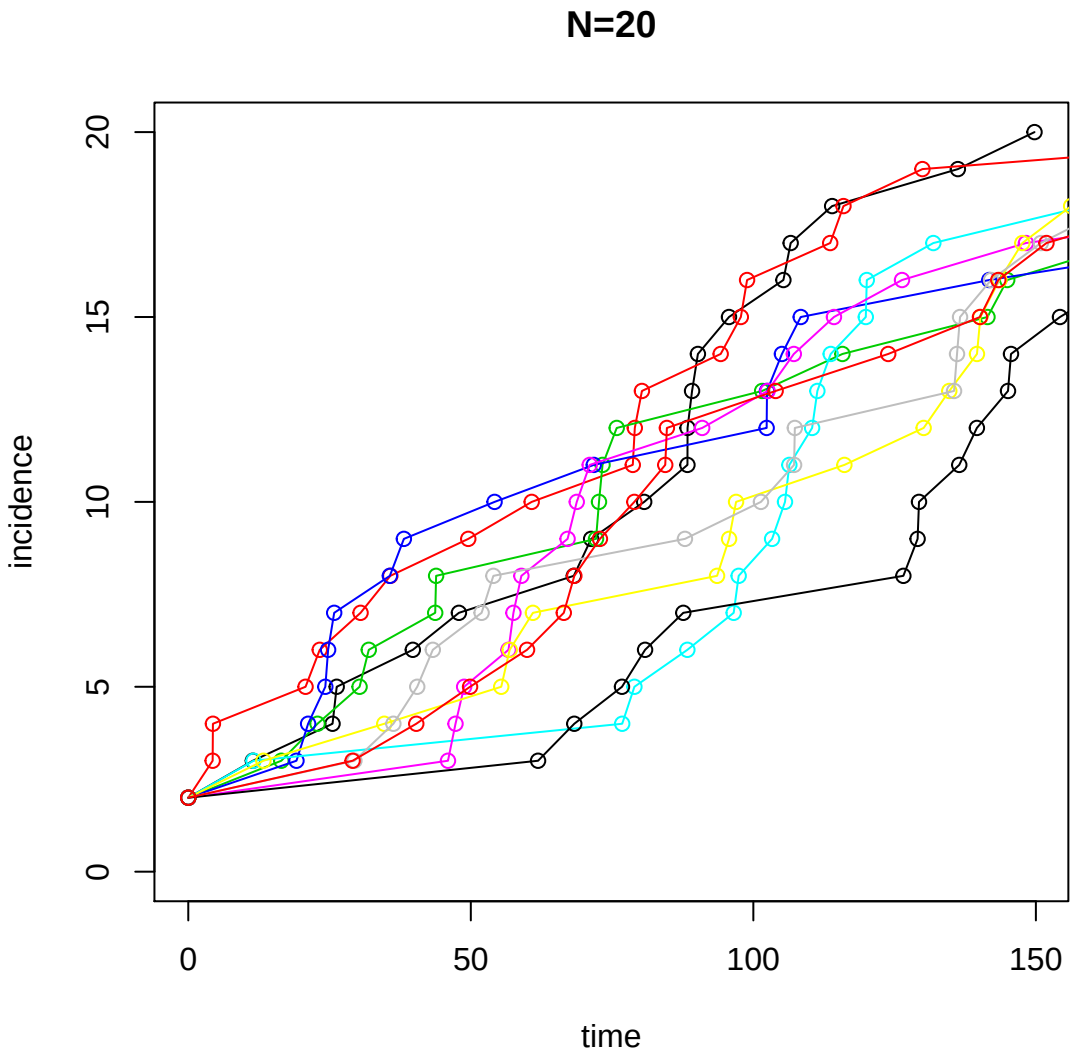


Evidently, increasing population size causes the epidemic to behave more like a deterministic model (the trajectories are relatively smooth). But, there remains some variation in the time at which the epidemic takes off.

Based on this, we predict that as we reduce population size we will find the solutions to show more variation compared with the example. For instance, if we use the original parameters but reduce the total population size from the original example by a factor of 10 to $N = 20$.

```
> #Same parameters but population size of 20
```

```
> set.seed(38499583)                          #set the random seed so results are repeatable
> nsims <- 10                                 #number of simulations to run
> pop.size <- 20                             #total size of the population
> Y0 <- 2                                     #initial number infected
> nstep <- pop.size-Y0                        #how many steps to run? until everyone infected
> xstart <- c(time=0,X=(pop.size-Y0),Y=Y0)  #initial conditions
> beta <- c(beta=3e-2)                      #transmission rate
> data <- vector(mode='list',length=nsims)  #create a list called ``data'' to store all runs
> for (k in 1:nsims) {                        #simulate k different runs
+   data[[k]] <- as.data.frame(SI.model(xstart,beta,nstep))  #main simulation step
+   data[[k]]$cum.time <- cumsum(data[[k]]$time) #calculates the running sum of inter-event intervals
+ }
> max.y<-max(data[[1]]$cum.time)               #find the maximum time any simulation ran (to set x axis)
> plot(c(0,pop.size),c(0,pop.size),type='n',xlab='time',ylab='incidence',xlim=c(0,max.y), main='N=20') #
> for (k in 1:nsims) {                        #loop over each simulation...
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')  #to plot
+ }
```



As predicted, epidemics in very small populations are indeed "noisy" and relatively unpredictable.

Exercise 2. Simulate the stochastic SIR model using Gillespie's direct method. As before, experiment with the initial number of infecteds (Y_0) and with the total population size (N). What effects do these have on the predictability of the epidemic?

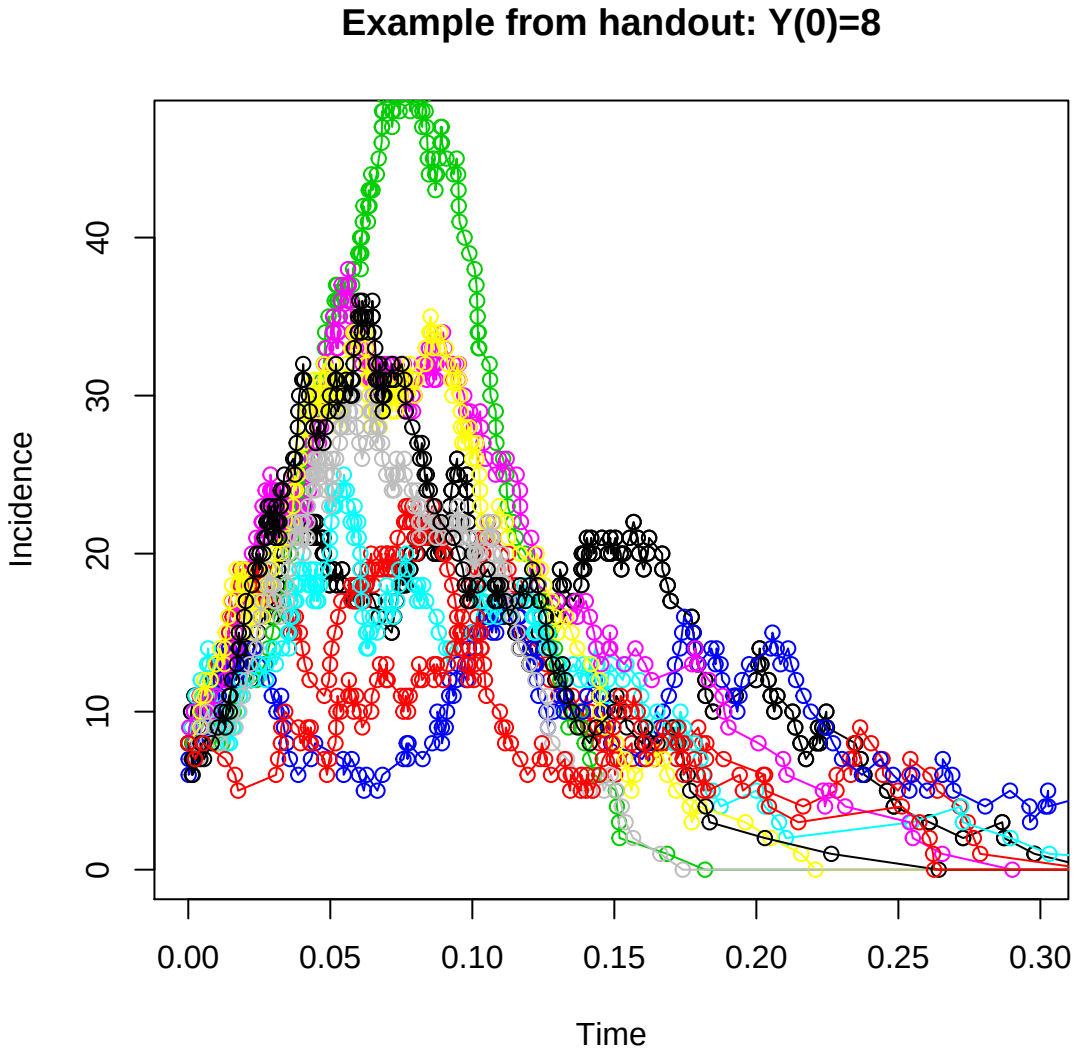
For this exercise, we require the functions `SIR.onestep` and `SIR.model` from the handout.

```
> SIR.onestep <- function (x, params) {  #function to calculate one step of stochastic SIR
+   X <- x[2]                            #local variable for susceptibles
+   Y <- x[3]                            #local variable for infecteds
+   Z <- x[4]                            #local variable for recovereds
+   N <- X+Y+Z                           #total population size (subject to demographic change)
+   with(                                #use with as in deterministic model to simplify code
+        as.list(params),
+        {
```

```
+          total.rate <- mu*N+beta*X*Y/N+mu*X+mu*Y+gamma*Y+mu*Z #calculate ``total rate''
+          tau <- rexp(n=1,rate=total.rate)                     #inter-event time
+          new.xyz <- c(X,Y,Z) #initialize a local variable at previous state variable values
+          U <- runif(1)       #uniform random deviate
+          new.xyz<-c(X,Y,Z-1) #death of recovered id ``default''
+          if (U<=(mu*N+beta*X*Y/N+mu*X+gamma*Y+mu*Y)/total.rate) new.xyz<-c(X,Y-1,Z)
+             #death of infected
+          if (U<=(mu*N+beta*X*Y/N+mu*X+gamma*Y)/total.rate) new.xyz<-c(X,Y-1,Z+1)
+                         #recovery of infected
+          if (U<=(mu*N+beta*X*Y/N+mu*X)/total.rate) new.xyz<-c(X-1,Y,Z)    #death of a susceptible
+          if (U<=(mu*N+beta*X*Y/N)/total.rate) new.xyz<-c(X-1,Y+1,Z)       #transmission event
+          if (U<=(mu*N/total.rate)) new.xyz<-c(X+1, Y, Z)                  #birth of susceptible
+          c(tau,new.xyz) #store result
+        }
+        )
+ }
> SIR.model <- function (x, params, nstep) {  #function to simulate stochastic SIR
+   output <- array(dim=c(nstep+1,4))         #set up array to store results
+   colnames(output) <- c("time","X","Y","Z") #name variables
+   output[1,] <- x                           #first record of output is initial condition
+   for (k in 1:nstep) {                      #iterate for nstep steps
+     output[k+1,] <- x <- SIR.onestep(x,params)
+   }
+   output                                    #return output
+ }
```

Again, for comparison, we repeat the example simulation from the handout.

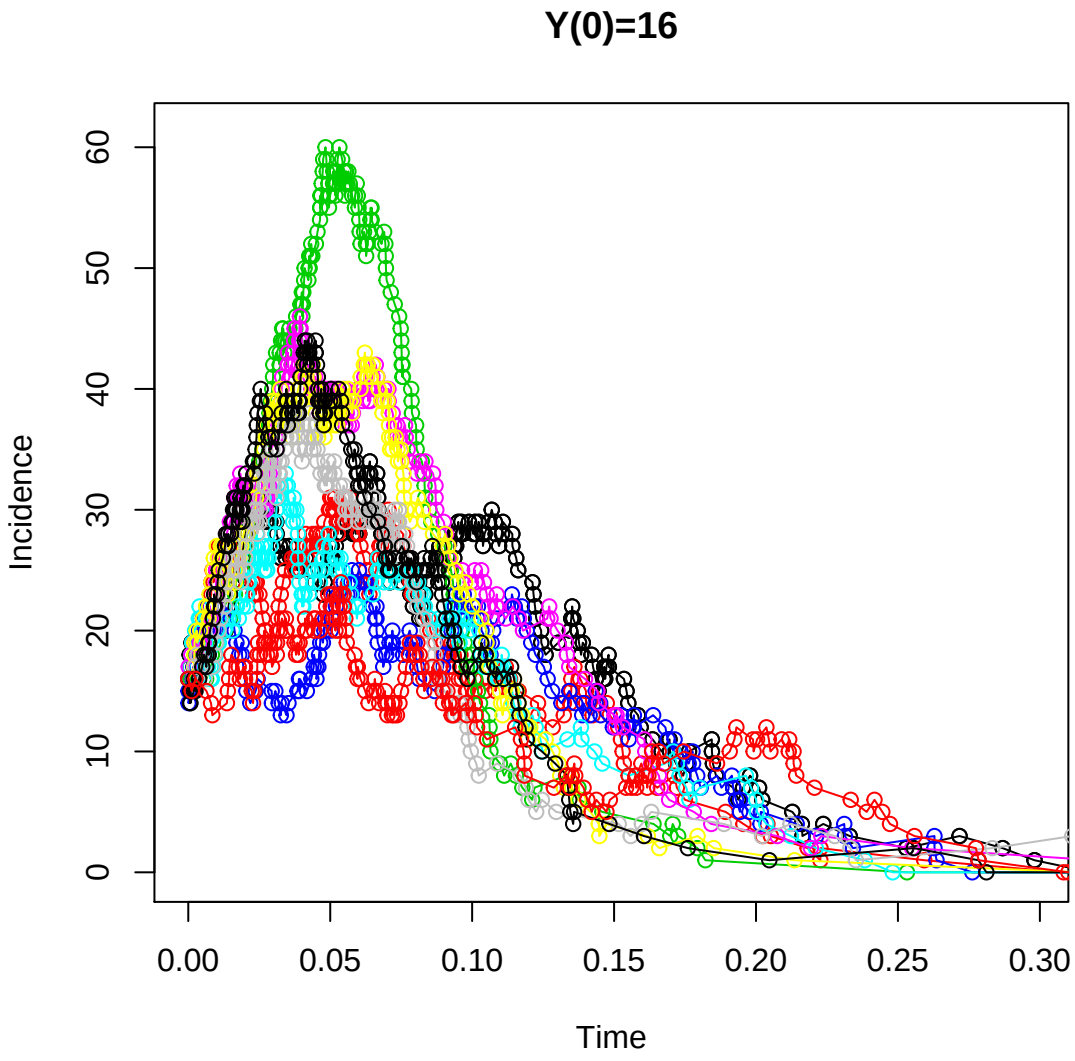
```
> set.seed(38499583)               #set seed
> nsims <- 10                      #number of simulations
> pop.size <- 100                  #total population size
> Y0 <- 8                          #initial number infected
> X0 <- round(0.98*pop.size)       #initial number suscepitlble (~98% of population)
> nstep <- 1600                    #number of events to simulate
> xstart <- c(time=0,X=X0,Y=Y0,Z=pop.size-X0-Y0) #initial conditions
> params <- list(mu=0.00001,beta=60,gamma=365/13) #parameters
> data <- vector(mode='list',length=nsims) #initialize list to store the output
> for (k in 1:nsims) {             #simulate nsims times
+   data[[k]] <- as.data.frame(SIR.model(xstart,params,nstep))
+   data[[k]]$cum.time <- cumsum(data[[k]]$time)
+ }
> max.time<-data[[1]]$cum.time[max(which(data[[1]]$Y>0))] #maximum time in first simulation
> max.y<-1.8*max(data[[1]]$Y)      #find max infected in run 1 and increase by 80% for plot
> plot(Y~cum.time,data=data[[1]],xlab='Time',ylab='Incidence',col=1,xlim=c(0,max.time),ylim=c(0,max.y),
> for (k in 1:nsims) {             #add multiple epidemics to plot
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')
+ }
```



Here we simulate the system with $Y(0) = 16$.

```
> set.seed(38499583)                 #set seed
> nsims <- 10                        #number of simulations
> pop.size <- 100                    #total population size
> Y0 <- 16                            #initial number infected
> X0 <- round(0.98*pop.size)         #initial number suscepitlble (~98% of population)
> nstep <- 1600                      #number of events to simulate
> xstart <- c(time=0,X=X0,Y=Y0,Z=pop.size-X0-Y0) #initial conditions
> params <- list(mu=0.00001,beta=60,gamma=365/13) #parameters
> data <- vector(mode='list',length=nsims) #initialize list to store the output
> for (k in 1:nsims) {               #simulate nsims times
+   data[[k]] <- as.data.frame(SIR.model(xstart,params,nstep))
+   data[[k]]$cum.time <- cumsum(data[[k]]$time)
+ }
```

```
> max.time<-data[[1]]$cum.time[max(which(data[[1]]$Y>0))] #maximum time in first simulation
> max.y<-1.8*max(data[[1]]$Y)       #find max infected in run 1 and increase by 80% for plot
> plot(Y~cum.time,data=data[[1]],xlab='Time',ylab='Incidence',col=1,xlim=c(0,max.time),ylim=c(0,max.y),
> for (k in 1:nsims) {              #add multiple epidemics to plot
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')
+ }
```

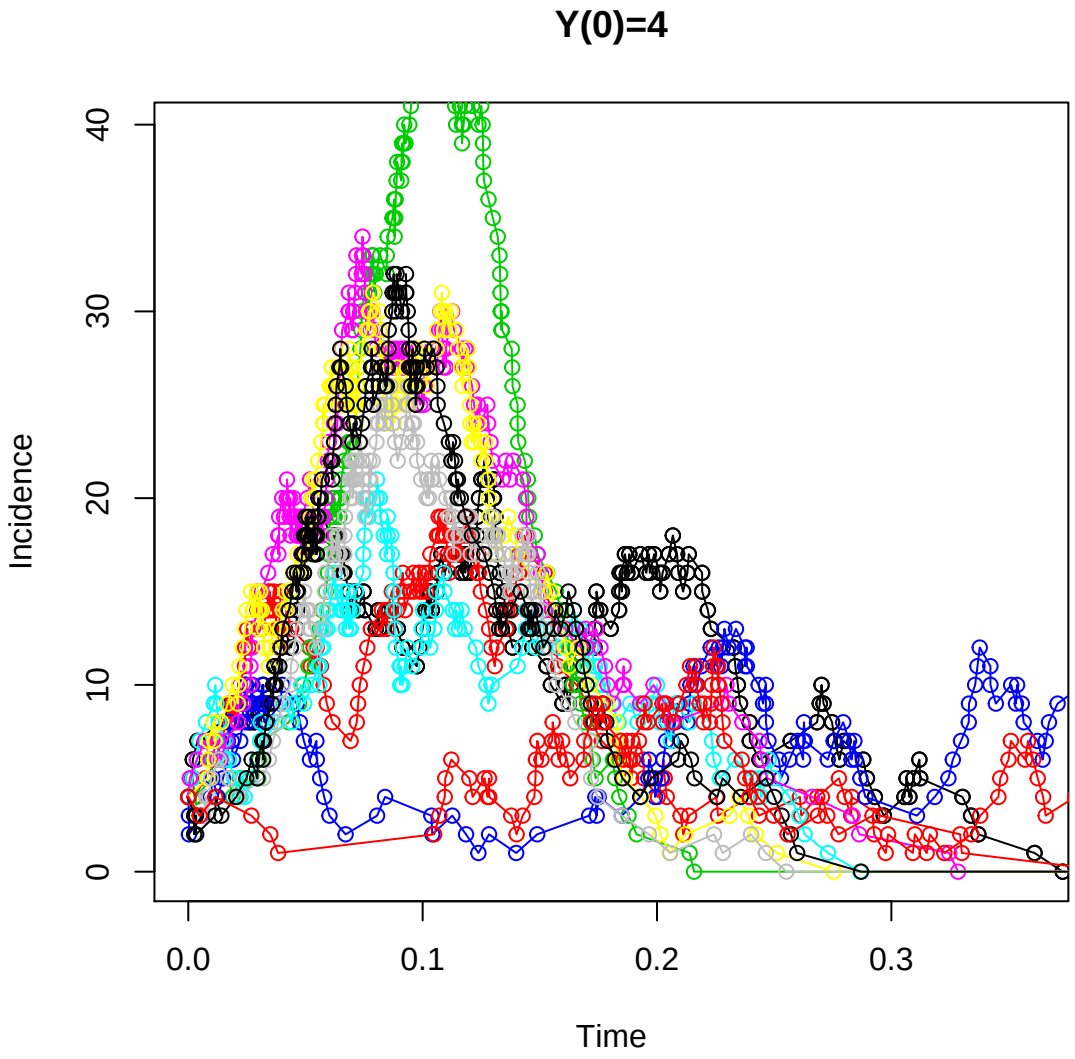


Evidently, as in the stochastic SI epidemic, increasing the initial number infected causes the epidemic to behave more like the deterministic solution. We notice, however, that unlike the SI epidemic, the total epidemic size is a random variable. Increasing the initial population size decreases the variance in the total epidemic size.

Now we simulate with half the original number of infected individuals.

```
> set.seed(38499583)                #set seed
> nsims <- 10                       #number of simulations
```

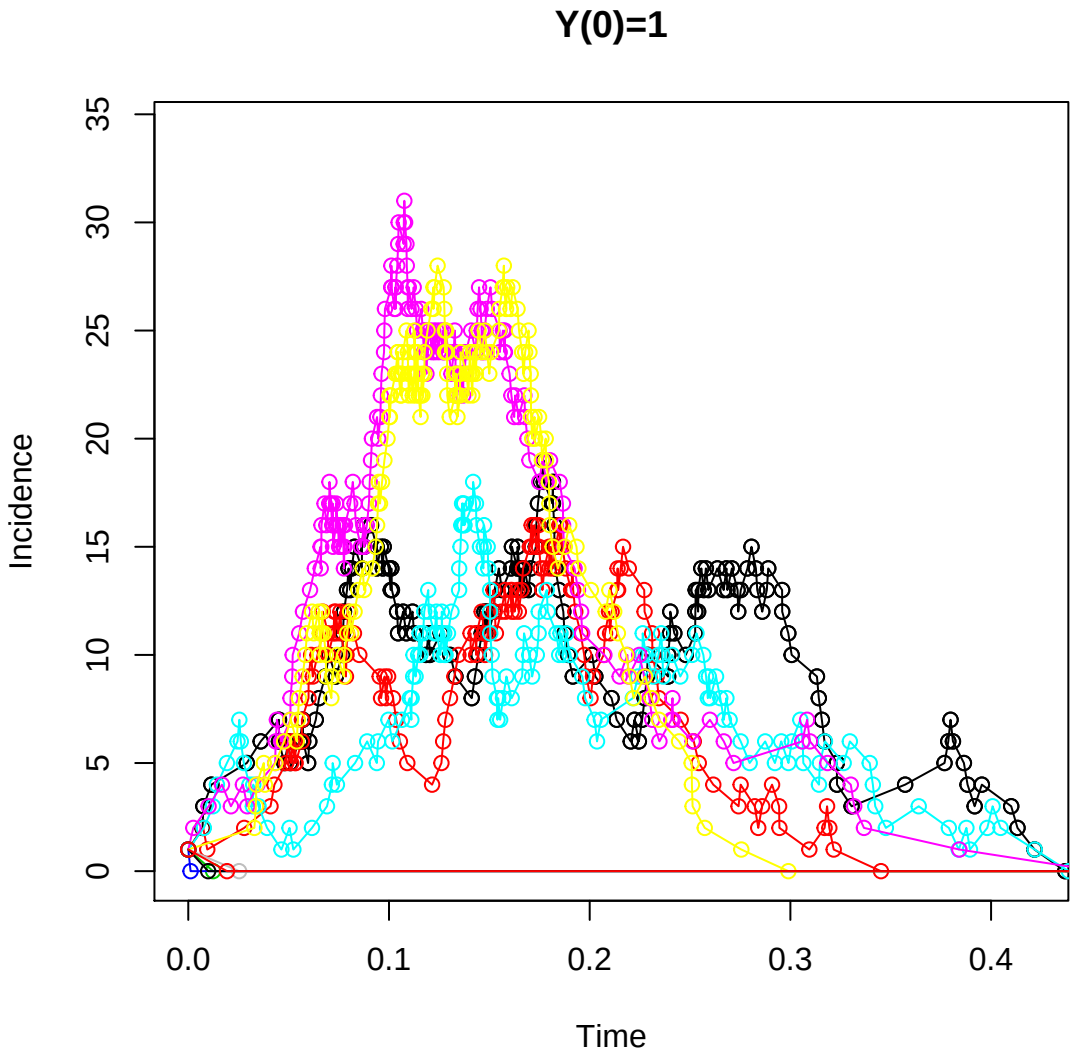
```
> pop.size <- 100                   #total population size
> Y0 <- 4                           #initial number infected
> X0 <- round(0.98*pop.size)        #initial number suscepitlble (~98% of population)
> nstep <- 1600                     #number of events to simulate
> xstart <- c(time=0,X=X0,Y=Y0,Z=pop.size-X0-Y0) #initial conditions
> params <- list(mu=0.00001,beta=60,gamma=365/13) #parameters
> data <- vector(mode='list',length=nsims) #initialize list to store the output
> for (k in 1:nsims) {              #simulate nsims times
+   data[[k]] <- as.data.frame(SIR.model(xstart,params,nstep))
+   data[[k]]$cum.time <- cumsum(data[[k]]$time)
+ }
> max.time<-data[[1]]$cum.time[max(which(data[[1]]$Y>0))] #maximum time in first simulation
> max.y<-1.8*max(data[[1]]$Y)       #find max infected in run 1 and increase by 80% for plot
> plot(Y~cum.time,data=data[[1]],xlab='Time',ylab='Incidence',col=1,xlim=c(0,max.time),ylim=c(0,max.y),
> for (k in 1:nsims) {              #add multiple epidemics to plot
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')
+ }
```



As expected, there is now tremendous variation in the shape of the epidemics that occur. We wonder, what would happen if we started with a single index case (i.e., $Y(0) = 1$).

```
> set.seed(38499583)                    #set seed
> nsims <- 10                           #number of simulations
> pop.size <- 100                       #total population size
> Y0 <- 1                               #initial number infected
> X0 <- round(0.98*pop.size)            #initial number suscepitlble (~98% of population)
> nstep <- 1600                         #number of events to simulate
> xstart <- c(time=0,X=X0,Y=Y0,Z=pop.size-X0-Y0) #initial conditions
> params <- list(mu=0.00001,beta=60,gamma=365/13) #parameters
> data <- vector(mode='list',length=nsims) #initialize list to store the output
> for (k in 1:nsims) {                  #simulate nsims times
+   data[[k]] <- as.data.frame(SIR.model(xstart,params,nstep))
+   data[[k]]$cum.time <- cumsum(data[[k]]$time)
```

```
+ }
> max.time<-data[[1]]$cum.time[max(which(data[[1]]$Y>0))] #maximum time in first simulation
> max.y<-1.8*max(data[[1]]$Y)       #find max infected in run 1 and increase by 80% for plot
> plot(Y~cum.time,data=data[[1]],xlab='Time',ylab='Incidence',col=1,xlim=c(0,max.time),ylim=c(0,max.y),
> for (k in 1:nsims) {              #add multiple epidemics to plot
+   lines(Y~cum.time,data=data[[k]],col=k,type='o')
+ }
```



Here we observe a new phenonenon: there might be no epidemic at all. Here we have an epidemic in only 5 out of 10 (50%) of simulations. In the others the initial single infected individual recovered prior to infecting a secondary case. Clearly, as the initial number of infected individuals increases the change that no secondary infections will occur declines.