A stochastic model for the transmission of 2019-nCov in Wuhan

John M. Drake

1/25/2020

Introduction

The epidemiology of the 2019-nCov in China is poorly understood. Here we seek to develop a model for the transmission of 2019-nCov during the early stages of transmission in Wuhan. This model may be useful for inference, forecasting or scenario analysis.

Strategy

1. Develop a generative model based on clinical reports and outbreak investigations reported in the media and scientific literature.
2. Estimate unknown parameters by fitting to data.
3. Application of model to specific problems such as characterizing key unknown quantities (e.g. reporting rate) and forecasting.

Model construction

Up to approximately January 25 the outbreak has remained small (hundreds of cases). Therefore, we adopt a stochastic modeling approach that reflects intrinsic noise. The model will be constructed for simulation via Gillespie's direct method. Additionally, we assume that the primary model compartments are Susceptible, Infected, and Recovered individuals. Given that the epidemic is an acute infection with dynamics that occur on time scales much faster than the demography of the population, we assume a *closed* population.

The basic function simulates one event in the transmission process.

```
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),
       {
         rates <- c(beta*X*Y/N, gamma*Y)
         changes <- matrix(c(-1, 1, 0,
                              0,-1, 1),
                           ncol=3, byrow=TRUE)
         tau <- -log(runif(1)) / sum(rates)   # exponential waiting time
         U <- runif(1)       #uniform random deviate
         m <- min(which(cumsum(rates)>=U*sum(rates)))
         x <- x[2:4] + changes[m,]
         return(out <- c(tau, x))
       }
       )
}
```

A second function iteratively applies `onestep` to evaluate a solution of the model.

```
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 <- onestep(x,params)
  }
  output                                    #return output
}
```

A third function simulates an arbitrary number of realizations, stores and plots the result.

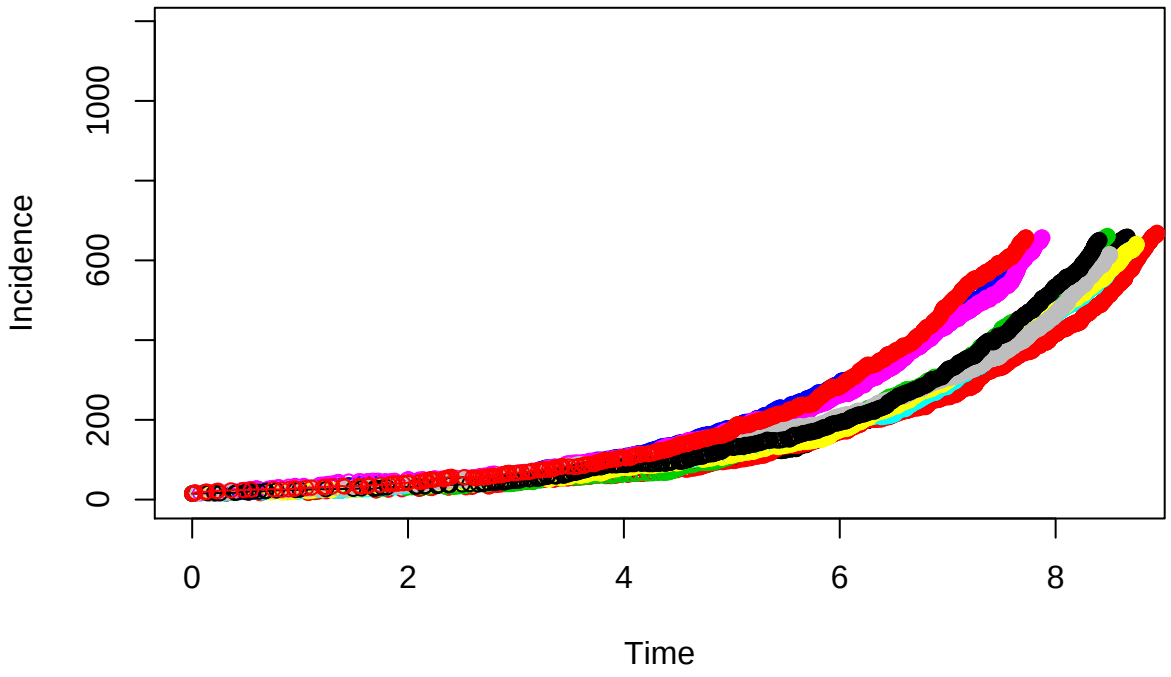
All units are expressed in terms of days. Parameters are as follows

Recovery rate (assuming one week infectious period): $\gamma = 1/7$

```
set.seed(1252019)                #set seed

evaluate <- function(){
  nsims <- 10                       #number of simulations
  pop.size <- 11081000                   #total population size
  Y0 <- 15                           #initial number infected
  X0 <- round(pop.size)         #initial number susceptible
  nstep <- 1000                     #number of events to simulate
  xstart <- c(time=0,X=X0,Y=Y0,Z=pop.size-X0-Y0) #initial conditions
  params <- list(beta=0.6,gamma=1/7) #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(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')
  }
}

evaluate()
```



Initial conditions

TBD