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

John M. Drake

1/28/2020

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. This is the second of two models. The first model in this series was simulated using Gillespie's direct method. This version of the model use (binomial) tau-leaping to speed computation.

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. Additional, we recognize that the waiting time distributions are not exponential and therefore use the Linear Chain Trick (LCT) to represent realistic wait times in each compartment (Cite: https://link.springer.com/article/10.1007/s00285-019-01412-w) From (https://github.com/jabacker/nCoV-2019/blob/master/Incubationperiod_2019nCoV.pdf), the incubation period is gamma distributed with a mean of 5.6 days and shape parameter 6.2. For simplicity, we assume an Erlang distribution with integer shape parameter of 6. Thus, we require six exposed classes.

It is predicted that the time between onset of symptoms and isolation will decline as awareness and clinical capacity are increased (Cite: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0000020). Therefore, the traditional "recovery" parameter (γ) is set to a declining lineary function of time to recognize the effect increased awareness, case identification and isolation, social distancing, etc. Movement from the I to R is therefore assumed to be an Erlang distribution with integer shape parameter of 6, but time varying mean.

For now, the model assumes a frequency-dependent force of infection of $\lambda = \beta \times Y/N$. This is reasonable for a respiratory infection that requires relatively close contact for infection to occur (compare: https://science.sciencemag.org/content/362/6410/75) **Ultimately, we hope to transfer estimates among different locations, therefore a force of infection that is intermediate between density-dependent and frequency-dependent transmission should be considered a high priority for further development.**

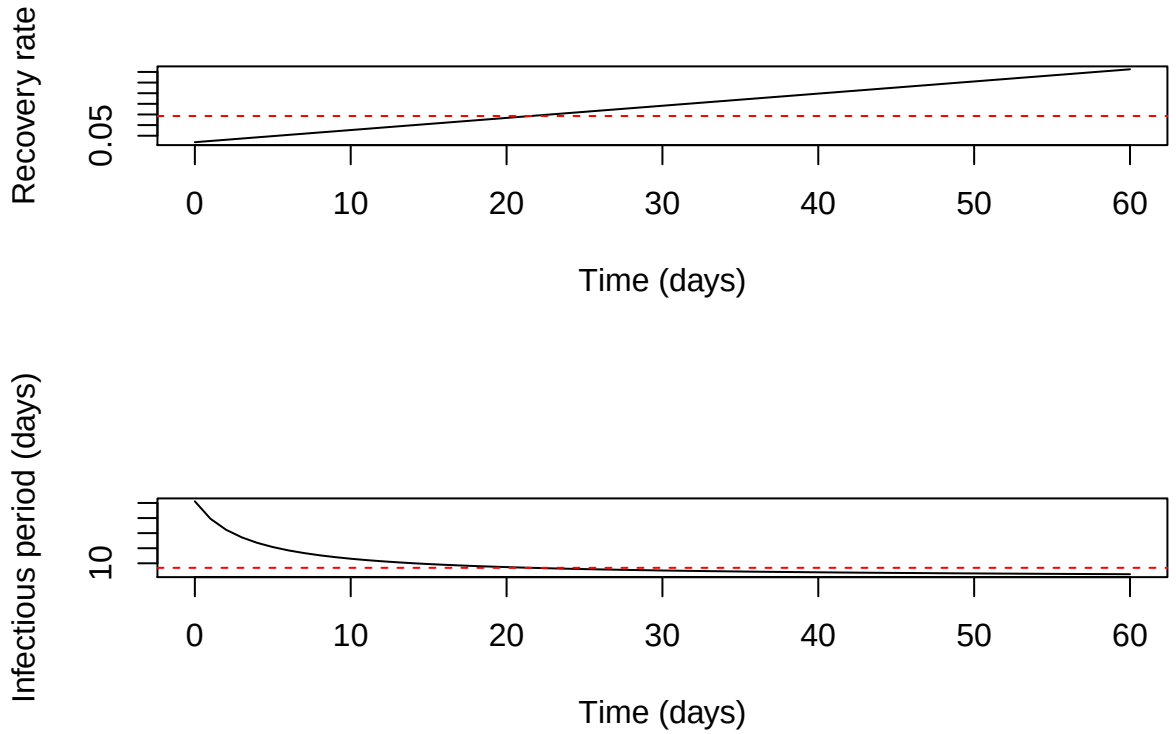
Support functions

The first function returns the time-dependent recovery rate as a function of time. The default parameter values are taken from estimates of the speed at which case isolation increased during a 2003 outbreak of SARS in Singapore (Cite: https://doi.org/10.1371/journal.pone.0000020)

```
gamma <- function(b=0.137/7, a0=0.04/7, t){
  # linear function
  # arguments (names chosen to follow Drake et al. (2006) https://doi.org/10.1371/journal.pone.0000020)
  # default parameters from "reduced model w/o week 8" estimates for SARS with units adjusted from week
  #    b: intercept (positive)
  #    a0: slope parameter (units of days, positive)
  #    t: time in the model

  gamma <- b + a0*t
  return(gamma)
}
```

Here we plot the notional recovery rate against time for the first 60 days of the outbreak and the implied mean infectious period. A horizontal line shows the assumed constant value of 7 day infectious period.

```
t <- seq(0, 60)
g <- gamma(t=t)
par(mfrow=c(2,1))
plot(t,g, type='l', xlab='Time (days)', ylab='Recovery rate')
abline(h=1/7, lty=2, col='red')
plot(t, 1/g, type='l', xlab='Time (days)', ylab='Infectious period (days)')
abline(h=7, lty=2, col='red')
```



The basic function simulates one event in the transmission process.

```
onestep <- function (x, params) {  #function to calculate one step of stochastic SIR

  S <- x[2]                              #local variable for susceptibles

  E1 <- x[3]                             #exposed classes
  E2 <- x[4]
  E3 <- x[5]
  E4 <- x[6]
  E5 <- x[7]
  E6 <- x[8]

  I1 <- x[9]                             #infectious classes
  I2 <- x[10]
  I3 <- x[11]
  I4 <- x[12]
  I5 <- x[13]
  I6 <- x[14]

  I <- I1+I2+I3+I4+I5+I6

  R <- x[15]                            #local variable for recovereds
  N <- S+E1+E2+E3+E4+E5+E6+I1+I2+I3+I4+I5+I6+R                          #total population size (subjec

  t <- x[16]                            #get current time


  with(                                  #use with as in deterministic model to simplify code
       as.list(params),
       {
         gammai <- 6*gamma(b=b, a0=a0, t=as.numeric(t))  # multiplier 6 for pseudo stage
         sigmai <- 6*sigma  # multiplier 6 for pseudo stages

         rates <- as.numeric(c(beta*I/N,
                    sigmai, sigmai, sigmai, sigmai, sigmai, sigmai,
                    gammai, gammai, gammai, gammai, gammai, gammai))

         propensity <- sum(rates)

         states0 <- x[2:(length(x)-1)]

         # transition probabilities

         p <- matrix(0, nrow=length(rates),ncol=length(states0))                                    # I

         p[1,]  <- c(exp(-rates[1])*dt,1-exp(-rates[1]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)   # S->

         p[2,]  <- c(0, exp(-rates[2])*dt, 1-exp(-rates[2]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)    # T
         p[3,]  <- c(0, 0, exp(-rates[3])*dt, 1-exp(-rates[3]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0)    # T
         p[4,]  <- c(0, 0, 0, exp(-rates[4])*dt, 1-exp(-rates[4]*dt), 0, 0, 0, 0, 0, 0, 0, 0)    # T
         p[5,]  <- c(0, 0, 0, 0, exp(-rates[5])*dt, 1-exp(-rates[5]*dt), 0, 0, 0, 0, 0, 0, 0)    # T
         p[6,]  <- c(0, 0, 0, 0, 0, exp(-rates[6])*dt, 1-exp(-rates[6]*dt), 0, 0, 0, 0, 0, 0)    # T
         p[7,]  <- c(0, 0, 0, 0, 0, 0, exp(-rates[7])*dt, 1-exp(-rates[7]*dt), 0, 0, 0, 0, 0)    # T
```

```
        p[8,]  <- c(0, 0, 0, 0, 0, 0, 0, exp(-rates[8])*dt, 1-exp(-rates[8]*dt), 0, 0, 0, 0, 0)    # T
        p[9,]  <- c(0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[9])*dt, 1-exp(-rates[9]*dt), 0, 0, 0, 0)    # T
        p[10,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[10])*dt, 1-exp(-rates[10]*dt), 0, 0, 0)    #
        p[11,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[11])*dt, 1-exp(-rates[11]*dt), 0, 0)    #
        p[12,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[12])*dt, 1-exp(-rates[12]*dt), 0)    #
        p[13,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[13])*dt, 1-exp(-rates[13]*dt))    #

        # update states

        states1 <- matrix(0, nrow=length(rates),ncol=length(states0))                                  #

        for(i in 1:length(rates)){
          states1[i,] <- t(rmultinom(1, states0[i], p[i,]) )
        }

        states1 <- colSums(states1)

        return(x <- c(dt, states1, tail(x,1)+dt))
      }
      )
}
```

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,length(x)))         #set up array to store results
  colnames(output) <- c("time","S",
                        "E1", "E2", "E3", "E4", "E5", "E6",
                        "I1", "I2", "I3", "I4", "I5", "I6",
                        "R", "cum.time") #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 <- as.numeric(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. Note that the mean of a gamma distributed random variable (incubation period, infectious period) is given by $\mu = k/r$ where k is the shape parameter and r is the rate. Thus, since we have assumed $k = 6$ for both the incubation period and infectious period, these are considered in the parameterization below.

Parameters are as follows:

Population size of Wuhan: $N \approx 11081000$ (Wikipedia)

We assume $R_0 = 2.6$ as a notional value (Imperial College estimate)

Constant recovery rate (assuming seven day infectious period): $\gamma = 6/7$ Note that this is a notional value and the current model includes a time-varying recovery rate.

Transmissibility: $\beta = R_0 \times \gamma \approx 3.7$ **Note: A better estimate of β might be obtained by measuring the takeoff rate and calculating according to Wearing & Rohani (https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.0020174#pmed-0020174-e004)**

Transition through incubation period (Becker): $\sigma = 6/5.6$

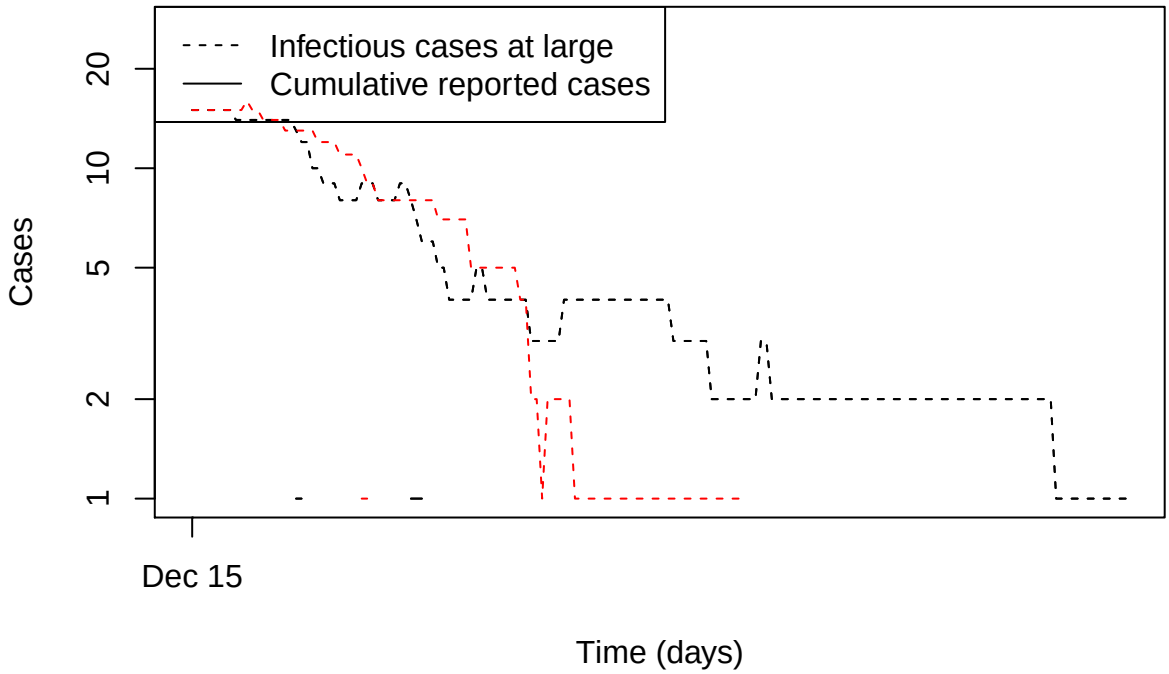
Simulations

Here we evaluate the model. Note that this model assumes that all infectious cases are eventually reported. The difference between infectious cases in the community and reported cases simply reflect the lag between infection and isolation. For this simulation it is assumed that the initial number of infectious cases is 15 (the reported size of the initial cluster) distributed over the first three infectious pseudo-classes. The model is presumed to start on December 15.

```
start <- as.Date('12/15/2019',format='%m/%d/%Y')

set.seed(1252019)                #set seed

evaluate.model <- function(params=list(beta=3.7, sigma=1/5.6, b=0.137/7, a0=0.04/7, dt=0.001), nsims=2,

  E10 <- 0
  E20 <- 0
  E30 <- 0
  E40 <- 0
  E50 <- 0
  E60 <- 0

  E0 <- E10+E20+E30+E40+E50+E60

  I10 <- 3
  I20 <- 3
  I30 <- 3
  I40 <- 3
  I50 <- 3
  I60 <- 0

  I0 <- I10+I20+I30+I40+I50+I60

  S0 <- round(pop.size-E0-I0)        #initial number susceptible
  xstart <- c(time=0, S=S0,
              E1=E10, E2=E20, E3=E30, E4=E40, E5=E50, E6=E60,
              I1=I10, I2=I20, I3=I30, I4=I40, I5=I50, I6=I60,
              R=pop.size-E10-E20-E30-E40-E50-E60-S0-I10-I20-I30-I40-I50-I60,
              cum.time = 0) #initial conditions
  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)
  }

  for(i in 1:nsims) data[[i]]$I <- data[[i]]$I1 + data[[i]]$I2 + data[[i]]$I3 + data[[i]]$I4 + data[[i]]

  max.time<-data[[1]]$cum.time[max(which(data[[1]]$I>0))] #maximum time in first simulation
  max.y<-1.8*max(data[[1]]$I)        #find max infected in run 1 and increase by 80% for plot
  plot(I~cum.time,data=data[[1]],xlab='Time (days)',ylab='Cases',col=1,xlim=c(0,max.time),ylim=c(1,max.y
  lines(R~cum.time,data=data[[1]], col=1,xlim=c(0,max.time),ylim=c(0,max.y), lty=1)
  legend('topleft', lty=c(2,1), legend=c('Infectious cases at large','Cumulative reported cases'))
  axis(1, at=seq(0,max.time,5), labels=format(start+seq(0,max.time,5), format= '%b %d'))
  axis(2)
  box()
```

```
  for (k in 1:nsims) {              #add multiple epidemics to plot
    lines(I~cum.time,data=data[[k]],col=k,type='l', lty=2)
    lines(R~cum.time,data=data[[k]],col=k, type='l', lty=1)

  }
}

evaluate.model()
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 9829 y values <= 0 omitted
## from logarithmic plot
```



This model predicts a somewhat faster epidemic growth than observed.