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

John M. Drake & Pejman Rohani

February 15, 2020

Introduction

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

Key features of this model include:

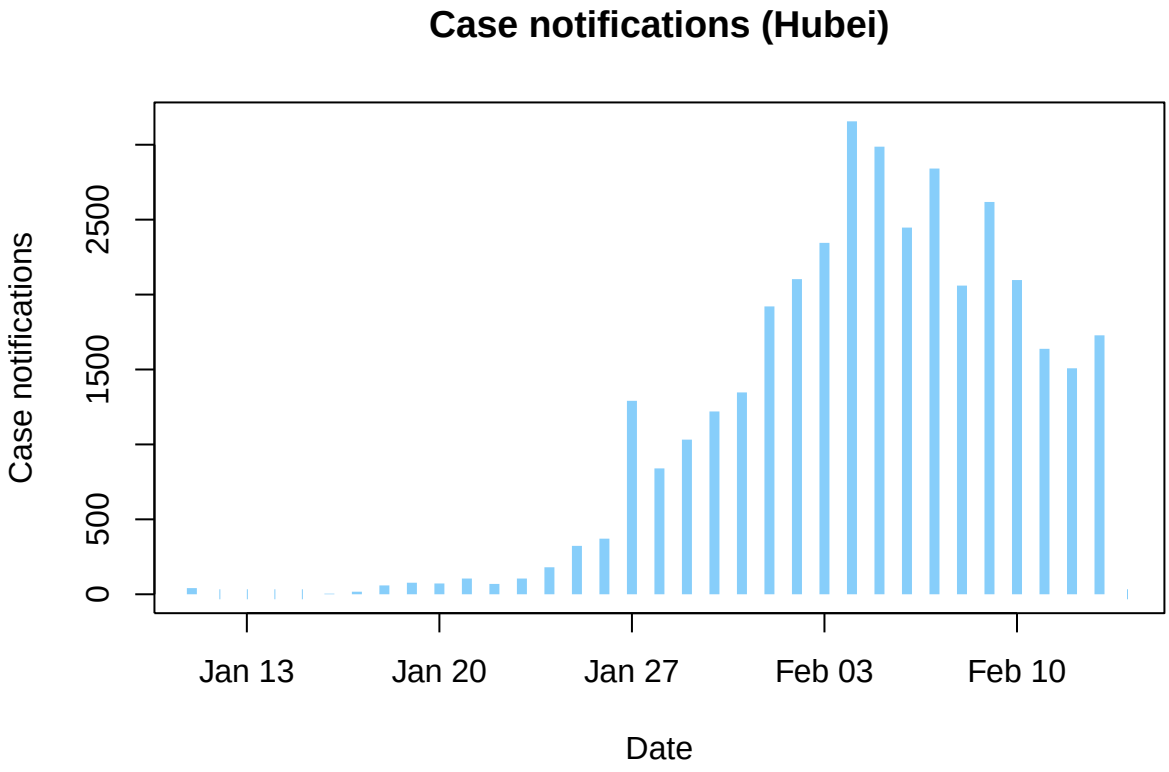
1. Stochastic transmission process
2. Realistic wait time distributions
3. Time varying rates of case detection, isolation, and case notification

This model was parameterized using clinical outcome reports and has not been calibrated by fitting to case notification data. The epidemic of 2019-nCov is changing rapidly and information that was used in the construction of this model may be incomplete or contain errors. Accordingly, these results are preliminary, provisional, and subject to change. These results have not been peer-reviewed, but have been prepared to a professional standard with the intention of providing useful interpretation of a rapidly developing event.

Data

Here we investigate case reports to the current date for comparison with the model. For this project, we look at case notifications in the province of Hubei.

```
today <- Sys.Date()
start <- as.Date('12/01/2019',format='%m/%d/%Y')
today.day <- today - start + 1
#data <- read.csv('china-province-data - cases.csv')
data <- read.csv('china-province-data - wikipedia-cases.csv')
  data[,1] <- as.Date(as.character(data[,1]), format='%Y-%m-%d')
  names(data)[1] <- 'date'


data.province <- data #head(data,-3)
#data.province$date <- as.Date(data.province$Date, format='%m-%d-%Y')
data.province[is.na(data.province)] <- 0
#data.province$total <- rowSums(data.province[,3:35])
data.province$total <- rowSums(data.province[,c(2,4:33)])
data.province$day <- data.province$date - start + 1  # Day 1 is given by "start"
data.province$cum.cases <- cumsum(data.province$total)

plot(data.province$date, data.province$Hubei, type='h', lwd=5, col='lightskyblue',
     lend='butt', xlab='Date', ylab='Case notifications', main='Case notifications (Hubei)')
```



Model details

Up to approximately January 25 the outbreak remained small (hundreds of cases) implying that a stochastic model is recommended. We assume that the primary model compartments are Susceptible (S), Exposed (latent) infections (E), Infected (I), Hospitalized (H), and Recovered (R) individuals. Infected individuals are distinguished according to their eventual detection, i.e. $I = I_d + I_u$ where I_d is detected individuals and I_u is undetected individuals. Note that whether an individual belongs to class I_d or I_u is determined at the time of infection, not at the time of detection. (This is required to ensure that progression through to recovery or isolation follows the desired statistical distributions.) Recovered individuals are also distinguished between those that recovered after treatment and natural recoveries (i.e. recoveries of undetected infections) and are designated R_u. It is assumed that case notifications are based on hospitalization. Recoveries of treated individuals are therefore not tracked because they do not contribute further to transmission or case notification. Case notifications (C), which arise from hospitalization after a lag, are also tracked for comparison with data. Because 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 ignoring demographic fluctuations due to births and deaths. Preliminary work indicates that key intervals distributions (infection to presentation of symptoms, presentation of symptoms to hospital admission, and hospital admission to case notification) are not exponential. Thus, we 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 an earlier version of work by Backer et al (https://github.com/jabacker/nCoV-2019/blob/master/Incubationperiod_2019nCoV.pdf), we decided to assume the incubation period to be gamma distributed with a shape parameter 6.4. We assume an Erlang distribution with integer shape parameter of 6. Thus, we require six exposed classes. It is expected 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 ($\gamma(t)$) is set to a declining linear function of time to recognize the effect of increased awareness, case identification and isolation, social distancing, etc. Movement from I to H is modeled as an Erlang distribution with estimated integer shape parameter of 4, but time varying mean. The force of infection is assumed to be frequency-dependent $\lambda = \beta \times S/N$, which is reasonable for a respiratory infection that requires

relatively close contact for infection to occur (compare: https://science.sciencemag.org/content/362/6410/75) Exposed cases are assumed to be detectable at time-varying rate $q(t)$. Importantly, undetected cases go through the infectious period at the base rate γ_0. Undetected cases may be undetected either because case detection is poor or because the case is asymptomatic. For now, detected and undetected (possibly asymptomatic) cases are assumed to be equally infectious, although it is straightforward to relax this assumption.

Simulation functions

Here we provide functions for evaluating the model.

The function `onestep` simulates one time step 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]                             #detected infectious classes
  I2 <- x[10]
  I3 <- x[11]
  I4 <- x[12]

  Iu1 <- x[13]                             #undetected infectious classes
  Iu2 <- x[14]
  Iu3 <- x[15]
  Iu4 <- x[16]

  I.detected <-I1+I2+I3+I4
  I.undetected <- Iu1+Iu2+Iu3+Iu4
  I <- I1+I2+I3+I4+Iu1+Iu2+Iu3+Iu4     #total infections

  H <- x[17]                             #local variable for hospitalized
  Ru <- x[18]                            #local variable for undetected recovereds

  C <- x[19]    # local variable for notifications

  N <- S+E1+E2+E3+E4+E5+E6+I1+I2+I3+I4+Iu1+Iu2+Iu3+Iu4+H+Ru          # total size of population

  t <- x[20]                            #get current time


  with(                                  #use with to simplify code
       as.list(params),
       {
         gammai <- 4*gamma(z=z, b=b, a0=a0, t=as.numeric(t))  # multiplier 4 for pseudo stages
         sigmai <- 6*sigma  # multiplier 6 for pseudo stages
```

```
        etat <- eta(t)
       # betat <- beta(beta0, t)

        rates <- as.numeric(c(beta0*I.detected/N+beta0*c*I.undetected/N,                          # m
                  sigmai, sigmai, sigmai, sigmai, sigmai, sigmai,   # movements out of E
                  gammai, gammai, gammai, gammai,                   # movements out of I (detected)
                  b, b, b, b,                                       # movements out of I (undetected)
                  etat))

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

        # transition probabilities

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

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

        p[2,]  <- c(0, exp(-rates[2]*dt), 1-exp(-rates[2]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        p[3,]  <- c(0, 0, exp(-rates[3]*dt), 1-exp(-rates[3]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        p[4,]  <- c(0, 0, 0, exp(-rates[4]*dt), 1-exp(-rates[4]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        p[5,]  <- c(0, 0, 0, 0, exp(-rates[5]*dt), 1-exp(-rates[5]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        p[6,]  <- c(0, 0, 0, 0, 0, exp(-rates[6]*dt), 1-exp(-rates[6]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
        p[7,]  <- c(0, 0, 0, 0, 0, 0, exp(-rates[7]*dt), (1-exp(-rates[7]*dt))*q(t,w), 0, 0, 0, (1-exp

        p[8,]  <- c(0, 0, 0, 0, 0, 0, 0, exp(-rates[8]*dt), 1-exp(-rates[8]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0
        p[9,]  <- c(0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[9]*dt), 1-exp(-rates[9]*dt), 0, 0, 0, 0, 0, 0, 0, 0
        p[10,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[10]*dt), 1-exp(-rates[10]*dt), 0, 0, 0, 0, 0, 0
        p[11,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[11]*dt), 0, 0, 0, 0, 1-exp(-rates[11]*dt)

        p[12,]  <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  exp(-rates[12]*dt), 1-exp(-rates[12]*dt), 0, 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), 0,
        p[14,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  exp(-rates[14]*dt), 1-exp(-rates[14]*dt), 0
        p[15,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  exp(-rates[15]*dt), 0, 1-exp(-rates[15]


        p[16,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[16]*dt), 0, 1-exp(-rates[16

        # 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)
        states1[17] <- states1[17]+Ru  #add formerly Recovered undetected cases
        states1[18] <- states1[18]+C  #add formerly notified cases

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

The function `model` iteratively applies `onestep` to generate a solution of the stochastic 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", "Iu1", "Iu2", "Iu3", "Iu4",
                        "H", "Ru", "C", "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
}
```

The function `evaluate.model` simulates an arbitrary number of realizations and returns the result.

```
evaluate.model <- function(params=list(beta0=0.657, sigma=1/6.4, z=45, b=0.143, a0=0.0446, w=40, c=1,  
                                       init = list(S=59002000, E1=0, E2=0, E3=0, E4=0, E5=6, E6=0,
                                                   I1 = 1, I2= 0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=
                                                   H=0, Ru=0, C=0),
                           nsims=2, nstep=NULL, start=as.Date("2019-12-01"),today=Sys.Date()){

  #popn size of Wuhan 11081000
  #popn size of Hubei 59002000
  if(is.null(nstep)) nstep <- (as.numeric(today-start)+1+14)/params$dt #run simulation from start to cu

  xstart <- c(time=0, unlist(init), 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)

  }

  return(data)
}
```

The function `plot.model` provides automated visualization of model simulations.

```
plot.model <- function(data, log='y'){

  # process data
  nsims <- length(data)

  for(i in 1:nsims) data[[i]]$I <- data[[i]]$I1 + data[[i]]$I2 + data[[i]]$I3 +
      data[[i]]$I4
  for(i in 1:nsims) data[[i]]$Iu <- data[[i]]$Iu1 + data[[i]]$Iu2 + data[[i]]$Iu3 +
      data[[i]]$Iu4
  for(i in 1:nsims) data[[i]]$E <- data[[i]]$E1 + data[[i]]$E2 + data[[i]]$E3 +
      data[[i]]$E4 + data[[i]]$E5 + data[[i]]$E6

  max.time<-data[[1]]$cum.time[max(which(data[[1]]$I>0))] #maximum time in first simulation
  max.y<-max(data[[1]]$C)       #find max total confirmed cases for plotting range
```

```
 # calculate means
 m1 <- m2 <- m3 <- m4 <- m5 <- matrix(nrow=length(data[[1]]$I), ncol=nsims)
 for(i in 1:nsims){
   m1[,i] <- data[[i]]$E
   m2[,i] <- data[[i]]$I+data[[i]]$Iu
  # m3[,i] <- data[[i]]$Iu
   m4[,i] <- data[[i]]$H
   m5[,i] <- data[[i]]$C
 }
 E.mean <- rowMeans(m1)
 I.mean <- rowMeans(m2)
# Iu.mean <- rowMeans(m3)
 H.mean <- rowMeans(m4)
 C.mean <- rowMeans(m5)

 # colors
 E.col <- rgb(0,1,0,.25)
 I.col <- rgb(1,0,0,.25)
 Iu.col <- rgb(0.5, 0.5, 0, 0.25)
 H.col <- rgb(0,0,1,.25)
 C.col <- rgb(0,0,0,.25)
 E.mean.col <- rgb(0,1,0,1)
 I.mean.col <- rgb(1,0,0,1)
 Iu.mean.col <- rgb(0.5,0.5,0,1)
 H.mean.col <- rgb(0,0,1,1)
 C.mean.col <- rgb(0,0,0,1)

 #set up plot
 plot(I~cum.time,data=data[[1]],xlab='',ylab='Cases',col=1,
      xlim=c(0,max.time),ylim=c(1,max.y), type='n', lty=1, log=log, axes=FALSE) # set up plot

 # add data to plot
 day <- data.province$date - start
 lines(day, cumsum(data.province$Hubei), type='h', col='lightskyblue', lwd=3, lend='butt' )

 # plot spaghetti
 lines(E~cum.time,data=data[[1]], col=E.col, lty=1)
 lines(I+Iu~cum.time,data=data[[1]], col=I.col, lty=1)
# lines(Iu~cum.time,data=data[[1]], col=Iu.col, lty=1)
 lines(H~cum.time,data=data[[1]], col=H.col, lty=1)
 lines(C~cum.time,data=data[[1]], col=C.col, lty=1, lwd=3)


 axis(1, at=seq(0,max.time,5), labels=format(start+seq(0,max.time,5), format= '%b %d'))
 axis(2)
 box()

 if(nsims > 1){
 for (k in 2:min(100,nsims)) {              #add multiple epidemics to plot
   lines(E~cum.time, data=data[[k]], col=E.col, type='l', lty=1)
   lines(I+Iu~cum.time, data=data[[k]], col=I.col, type='l', lty=1)
#   lines(Iu~cum.time, data=data[[k]], col=Iu.col, type='l', lty=1)
   lines(H~cum.time, data=data[[k]], col=H.col, type='l', lty=1)
```

```
    lines(C~cum.time, data=data[[k]], col=C.col, type='l', lty=1, lwd=3)
  }

  # plot means
  lines(E.mean~cum.time, data=data[[k]], col=E.mean.col, lty=1)
  lines(I.mean~cum.time, data=data[[k]], col=I.mean.col, lty=1)
#  lines(Iu.mean~cum.time, data=data[[k]], col=Iu.mean.col, lty=1)
  lines(H.mean~cum.time, data=data[[k]], col=H.mean.col, lty=1)
  lines(C.mean~cum.time, data=data[[k]], col=C.mean.col, lty=1)
  }

   legend('topleft', lty=c(1,1,1,1,1,1), lwd=c(1,1,1,1,3,3), bty='n', cex=0.75,
         col=c(E.col, I.col, H.col, C.col, 'lightskyblue'),
         legend=c('Latent cases', 'Infectious cases', 'Hospitalized',
                  'Cumulative reported cases (Model)', 'Cumulative reported cases (Data)'))
}
```

A second function plots just the observed and unobserved cases over time.

```
plot.model2 <- function(data, log='y'){

  # process data
  nsims <- length(data)

  for(i in 1:nsims) data[[i]]$I <- data[[i]]$I1 + data[[i]]$I2 + data[[i]]$I3 +
      data[[i]]$I4
  for(i in 1:nsims) data[[i]]$Iu <- data[[i]]$Iu1 + data[[i]]$Iu2 + data[[i]]$Iu3 +
      data[[i]]$Iu4
  for(i in 1:nsims) data[[i]]$E <- data[[i]]$E1 + data[[i]]$E2 + data[[i]]$E3 +
      data[[i]]$E4 + data[[i]]$E5 + data[[i]]$E6

  max.time<-data[[1]]$cum.time[max(which(data[[1]]$I>0))] #maximum time in first simulation
  #max.y<-max(data[[1]]$Q)       #find max total confirmed cases for plotting range
  max.y <- max(unlist(lapply(data, FUN=function(x) max(x$C))))

  # calculate means
  m1 <- m2 <- m3 <- m4 <- m5 <- matrix(nrow=length(data[[1]]$I), ncol=nsims)
  for(i in 1:nsims){
    m1[,i] <- data[[i]]$E
    m2[,i] <- data[[i]]$I
    m3[,i] <- data[[i]]$Iu
    m4[,i] <- data[[i]]$H
    m5[,i] <- data[[i]]$C
  }

  observed <- m5
  unobserved <- m1 + m2 + m3 + m4

  obs.mean <- rowMeans(observed)
  unobs.mean <- rowMeans(unobserved)
  cum.time <- data[[1]]$cum.time

  # colors
  E.col <- rgb(0,1,0,.25)
```

```
  I.col <- rgb(1,0,0,.25)
  Iu.col <- rgb(0.5, 0.5, 0, 0.25)
  H.col <- rgb(0,0,1,.25)
  C.col <- rgb(0,0,0,.25)
  E.mean.col <- rgb(0,1,0,1)
  I.mean.col <- rgb(1,0,0,1)
  Iu.mean.col <- rgb(0.5,0.5,0,1)
  H.mean.col <- rgb(0,0,1,1)
  C.mean.col <- rgb(0,0,0,1)

  #set up plot
  plot(obs.mean~cum.time, xlab='',ylab='Cases',col=1,
       xlim=c(0,max.time),ylim=c(1,max.y), type='n', lty=1, log=log, axes=FALSE) # set up plot

  # add data to plot
  day <- data.province$date - start
  lines(day, cumsum(data.province$Hubei), type='h', col='lightskyblue', lwd=3, lend='butt' )

  # plot spaghetti
  lines(unobserved[,1]~cum.time, col=E.col, lty=1)
  lines(observed[,1]~cum.time, col=C.col, lty=1, lwd=1)

  axis(1, at=seq(0,max.time,5), labels=format(start+seq(0,max.time,5), format= '%b %d'))
  axis(2)
  box()

  if(nsims > 1){
  for (k in 2:min(100,nsims)) {              #add multiple epidemics to plot
    lines(unobserved[,k]~cum.time,  col=E.col, type='l', lty=1)
    lines(observed[,k]~cum.time, col=C.col, type='l', lty=1, lwd=1)
  }

  # plot means
  lines(unobs.mean~cum.time, col=E.mean.col, lty=1)
  lines(obs.mean~cum.time, col=C.mean.col, lty=1)
  }

   legend('topleft', lty=c(1,1,1), lwd=c(1,1,3), bty='n', cex=0.75,
         col=c(E.col, C.col, 'lightskyblue'),
         legend=c('Unobserved cases', 'Cumulative reported cases (Model)', 'Cumulative reported cases (
}
```

A third plotting function summarizes the model-based distribution of cases and notifications at the current time.

```
plot.summary <- function(data){
   # process data
  nsims <- length(data)

  for(i in 1:nsims) data[[i]]$I <- data[[i]]$I1 + data[[i]]$I2 + data[[i]]$I3 +
      data[[i]]$I4
  for(i in 1:nsims) data[[i]]$Iu <- data[[i]]$Iu1 + data[[i]]$Iu2 + data[[i]]$Iu3 +
      data[[i]]$Iu4
  for(i in 1:nsims) data[[i]]$E <- data[[i]]$E1 + data[[i]]$E2 + data[[i]]$E3 +
```

```
      data[[i]]$E4 + data[[i]]$E5 + data[[i]]$E6

  max.time<-data[[1]]$cum.time[max(which(data[[1]]$I>0))] #maximum time in first simulation
  max.y<-max(data[[1]]$I)       #find max total confirmed cases for plotting range

  # calculate means
  m1 <- m2 <- m3 <- m4 <- m5 <- matrix(nrow=length(data[[1]]$I), ncol=nsims)
  for(i in 1:nsims){
    m1[,i] <- data[[i]]$E
    m2[,i] <- data[[i]]$I
    m3[,i] <- data[[i]]$Iu
    m4[,i] <- data[[i]]$H
    m5[,i] <- data[[i]]$C
  }
  E.mean <- rowMeans(m1)
  I.mean <- rowMeans(m2)
  Iu.mean <- rowMeans(m3)
  H.mean <- rowMeans(m4)
  C.mean <- rowMeans(m5)

  # colors
  E.col <- rgb(0,1,0,.25)
  I.col <- rgb(1,0,0,.25)
  Iu.col <- rgb(0.5, 0.5, 0, 0.25)
  H.col <- rgb(0,0,1,.25)
  C.col <- rgb(0,0,0,.25)
  E.mean.col <- rgb(0,1,0,1)
  I.mean.col <- rgb(1,0,0,1)
  Iu.mean.col <- rgb(0.5,0.5,0,1)
  H.mean.col <- rgb(0,0,1,1)
  C.mean.col <- rgb(0,0,0,1)

  #set up plot
  j <- max(which(data[[1]]$time < today.day))

  h1 <- hist(log10(m1[j,]+m2[j,]+m3[j,]+m4[j,]), plot=FALSE)  # total unnotifed cases in the community
  h2 <- hist(log10(m5[j,]), plot=FALSE)          # total case notifications

  xlim <- c(floor(min(c(h1$mids, h2$mids))),ceiling(max(c(h1$mids, h2$mids))))
  ylim <- c(0, ceiling(max(c(h1$density, h2$density))))

  plot(xlim, ylim, type='n', axes=FALSE, xlab='Outbreak size', ylab='Probability density', main=paste('
  lines(h1$mids, h1$density, type='l', lwd=3, lend='butt', col=rgb(0.5,0.5,0, 0.5))
  lines(h2$mids, h2$density, type='l', lwd=3, lend='butt', col=C.col)
  legend('topleft', lty=1, col = c(rgb(0.5,0.5,0, 0.5), C.col), legend=c('Unobserved cases', 'Case noti
  axis(1, seq(min(xlim), max(xlim), by=1), labels=formatC(10^seq(min(xlim), max(xlim), by=1), format='d
  box()
}
```

Parameters

Parameter summary

The model depends on the following parameters

- Population size of Hubei: $N \approx 59002000$ (Wikipedia)
- Basic reproduction number: $R_0 = 2.3$
- Natural infectious period ($1/\gamma_0$): 7 days
- Rate of increase in isolation rate: $a_0 = 0.0446$
- Time at which case detection rate increased: January 9, 2020 (Day 40)
- Time at which increase in isolation rate initiated: January 15, 2020 (Day 45)

Parameter details

All rates are expressed in terms of days. The epidemic is assumed to have originated on December 1, 2019 with one case.

We have estimated R_0 and R_{eff} to range from 1.4 to 4.6, depending on data and modeling assumptions (see: *Estimating R_0 and other parameters for the 2019-nCov epidemic*). For this study, we assume $R_0 \approx 4.6$, which is higher than most estimates by other groups, but not unreasonable.

The rate of progression from symptomatic illness to hospitalization (γ) is assumed to be piecewide linear with an average infectious period of $\frac{1}{0.143} \approx 7$ days prior to an arbitrary intervention day d, followed by a linear increase in average recovery rate increasing in time at rate a_0. The default assumption is that $d = 45$, which (assuming an epidemic start date of December 1) corresponds to a signficant change on January 15 as found by statistical analysis, and four days before testing was expanded in Wuhan.

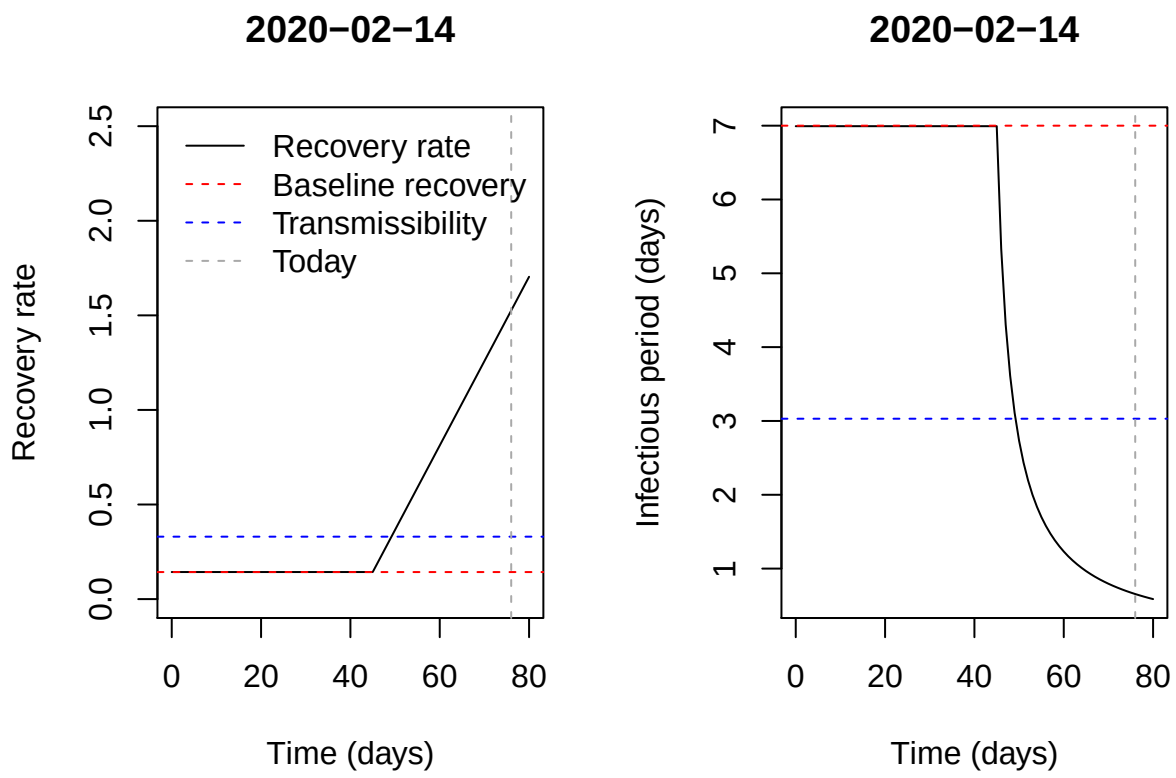
$$\gamma(t) = \begin{cases} \frac{1}{7}, & \text{if } t < d \\ \frac{1}{7} + a_0(t-d), & \text{otherwise} \end{cases} \tag{1}$$

```
gamma <- function(z = 45, b=0.143, a0=0.0446, t){
  # linear function
  # default parameters z = 50, b=1/7, a0=0.3 from Paige Miller analysis; revised 2/1/20 based on update
  #    z: time at start of intervention (notionally Jan 19, 50 days after notional start on Dec 1 and t
  #    b: intercept (positive)
  #    a0: slope parameter (units of days, positive) -- original analysis had set to 0.03, but this doe
  #    t: time in the model

  gamma <- ifelse(t<=z, gamma <- b, gamma <- b + a0*(t-z))
  return(gamma)
}
```

For illustration, we plot the recovery rate against time for the first 60 days of the outbreak and the implied mean infectious period. The horizontal read line shows the baseline ≈ 7 day infectious period, as fit by our statistical model. The horizontal blue line shows the assumed constant $\beta \approx 0.657$. Subcritical transmission, required for containment, is achieved when the recovery rate exceeds the transmissibility. Note that use of the Linear Chain Trick to induce realistic waiting times requires the introduction of pseudo-stages with their own rates. The mean of a Erlang distributed random variable (e.g. incubation period, infectious period) is given by $\mu = k/r$ where k is the shape parameter and r is the rate parameter. We have assumed $k = 6$ for the incubation period (following CITE - PENDING UPDATE WITH OUR OWN ANALYSIS). We have estimated $k = 4$ for the the infectious period. Thus, the rate of progression out of pseudo-stage I_i is

$\gamma_i = 4 \times \gamma^*(t)$, where $\gamma^*(t) = \gamma(t)$ for infectious individuals that will eventually be detected and $\gamma^*(t) = \gamma_0$ for undetected cases. The current time as of this writing is indicated by a vertical grey line.

```
t <- seq(0, 80)
g <- gamma(t=t)
par(mfrow=c(1,2))
plot(t,g, type='l', xlab='Time (days)', ylab='Recovery rate', ylim=c(0,2.5), main=today)
abline(h=1/7, lty=2, col='red')
abline(h=0.33, lty=2, col='blue')
abline(v=today.day, lty=2, col='darkgrey')
legend('topleft', col=c('black','red','blue','darkgrey'), lty=c(1,2,2,2), bty='n',
       legend=c('Recovery rate','Baseline recovery','Transmissibility','Today'))
plot(t, 1/g, type='l', xlab='Time (days)', ylab='Infectious period (days)', main=today)
abline(h=7, lty=2, col='red')
abline(h=1/0.33, lty=2, col='blue')
abline(v=today.day, lty=2, col='darkgrey')
```

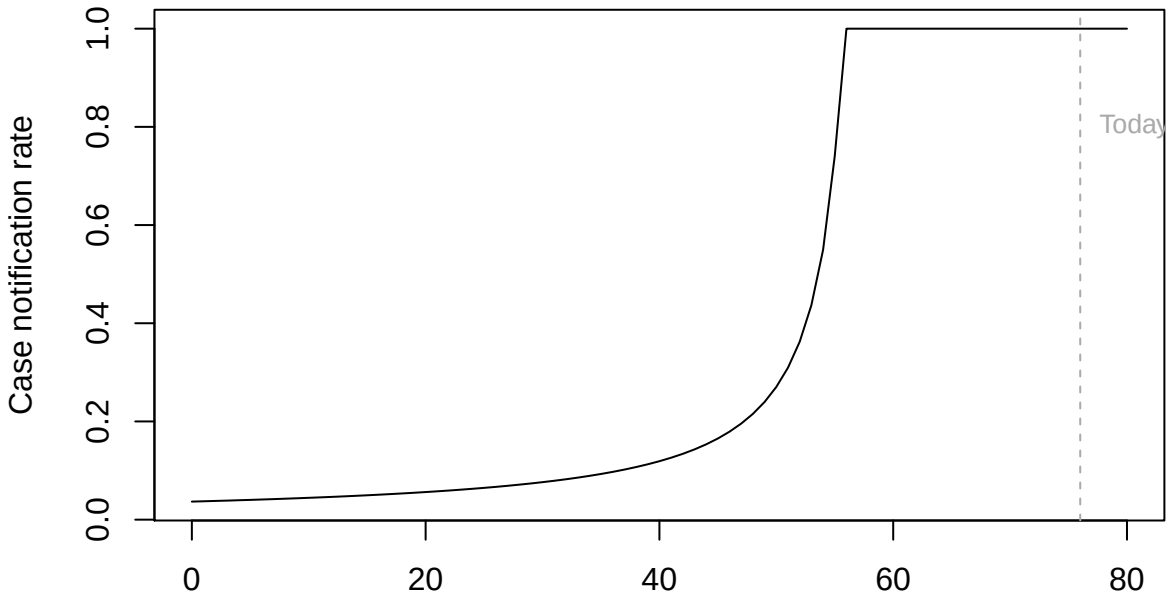


Notification is assumed to be time dependent. Analysis of clinical outcomes suggests that the *notification rate* may be represented by

$$\eta(t) = \begin{cases} (-0.47t + 27.2)^{-1}, & \text{if } t \leq 55 \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

The vertical line shows the case notification rate at the time of this report.

```
eta <- function(t) ifelse(t<=55,1/(-0.47*t+27.2),1)
plot(t, eta(t), type='l', xlab='', ylab='Case notification rate')
abline(v=today.day, lty=2, col='darkgrey')
text(today.day, 0.8, labels = paste('Today:',today), pos=4, col='darkgrey', cex=0.8)
```



Finally, case detection probability $q(t)$ is also assumed to be time-dependent. We assume that case detection was initially quite rare (e.g. $q(t) = q_0 << 1$), but at time w becomes much higher (e.g., after opening fever clinics on 9 January, Day 40, or the expansion of testing on Jan 19, Day 50). We have roughly estimated the baseline case detection rate to be $q_0 = 0.11$, although this calculation is laden with many assumptions.

$$q(t) = \begin{cases} q_0, & \text{if } t \leq w \\ q_1, & \text{otherwise.} \end{cases} \tag{3}$$

```
fever.clinic <- as.Date('2020-01-09')
w <- fever.clinic-start+1
q <- function(t, w=40, q0=0.11, q1=0.98) ifelse(t<=w,q0,q1)
```

In this analysis, we assume a value for $\beta = 0.657$ based on our maximal estimate $R_0 = 4.6$ (i.e. $4.6/7 \approx 0.657$) for Hubei. (Another value we have worked with is $based on \beta = 2.3/7 = 0.329$.)

The average duration of the incubation period has been estimated to be 6.4 days (Becker), yielding a progression rate of: $\sigma = 1/6.4$.

Scenarios

Here we examine some scenarios.

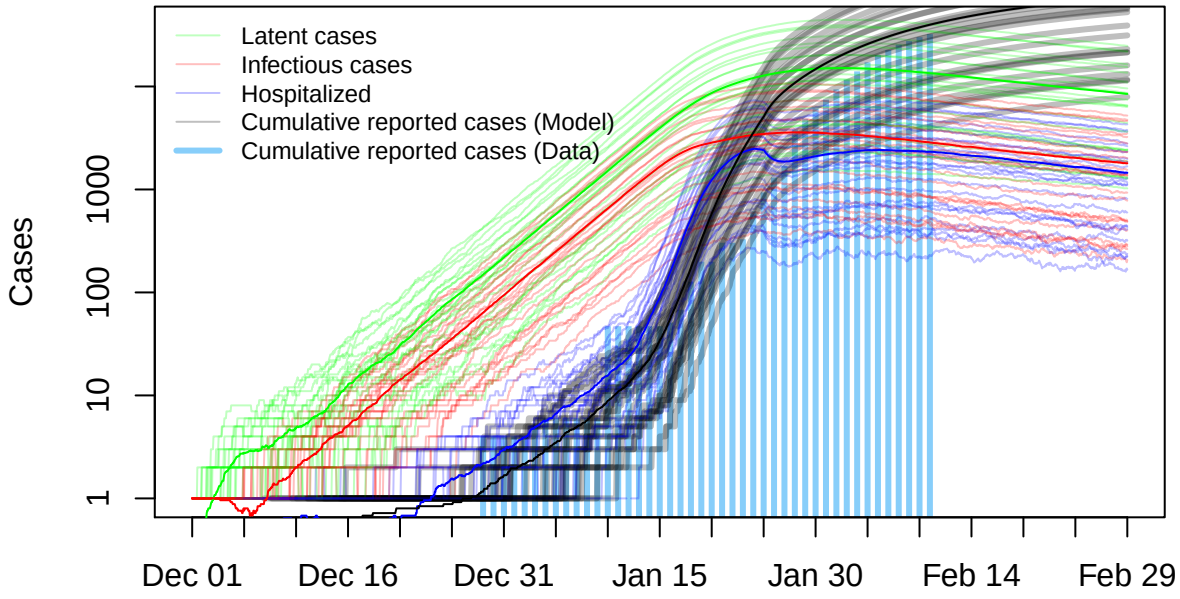
Scenario 1. Most likely scenario

We consider this our most likely scenario. In this scenario, the outbreak starts from one case around December 1. Rationale for this initial condition is as follows. First, we assume that initial cases (prior to the reporting of a cluster) would only be recognized if disease was severe and that ~20% of infections result in severe disease (http://en.nhc.gov.cn/2020-01/30/c_76048.htm). Four such severe cases were reported on Dec. 29 implying the existence of an additional 16 unnoticed cases at that time (20 total). Given the ~7 day doubling time (REF) and working backwards, we conclude there would have been 10 cases on December 22, 5 cases on December 15, ~2 cases on December 8, and ~1 case on December 1. Similarly, there is a least one case reported on December 6 by (REF), so by the same logic, there would have been 5 cases at that time. Working backwards, 2-3 cases around Dec 1 and 1 case around Nov 24. These dates are consistent with the Nextstrain estimates for the most recent common ancestor.

Note: we have experimented with allowing asymptomatic cases to infect during their last stage (approximately one day prior to becoming symptomatic). Although presymptomatic transmission is almost certain to be a key factor in eventual containment, we find that it contributes negligibly to transmission during the takeoff and exponential phases of the epidemic.

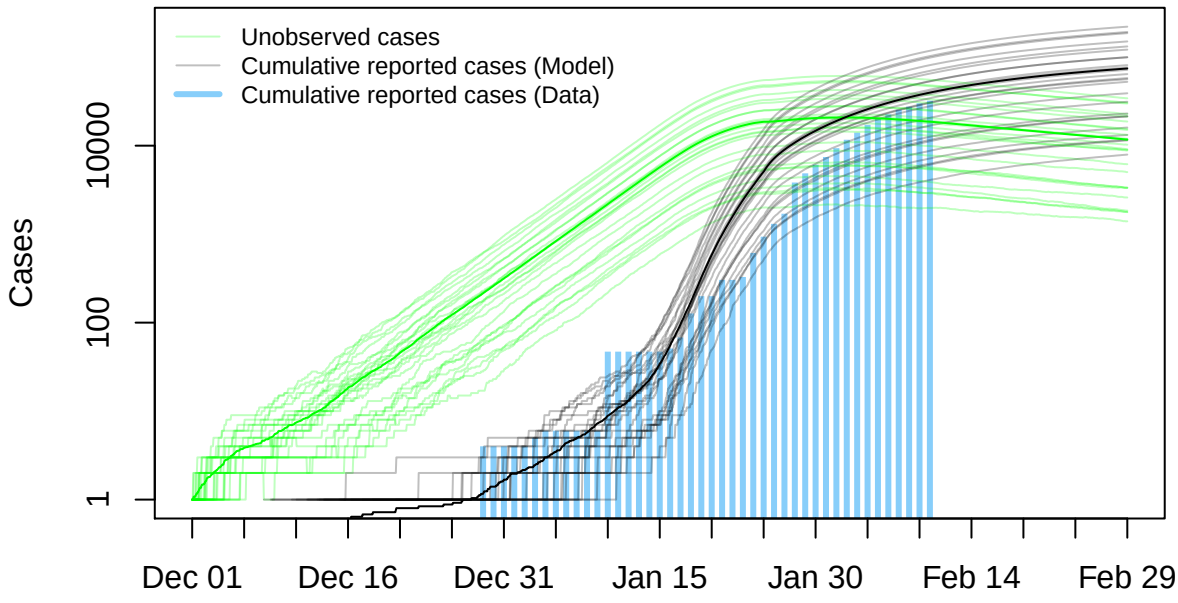
```
start <- as.Date('12/01/2019',format='%m/%d/%Y')
set.seed(1292020)                    #set seed
out1 <- evaluate.model(init = list(S=59002000,
                                   E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
                                   I1=1, I2=0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
                                   R=0, Ru=0, Q=0),
                       nsims=25, nstep=NULL, start=start)
plot.model(out1, log='y')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 203 y values <= 0 omitted
## from logarithmic plot
```




```
plot.model2(out1, log='y')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 138 y values <= 0 omitted
## from logarithmic plot
```




```
#plot.summary(out1)
```

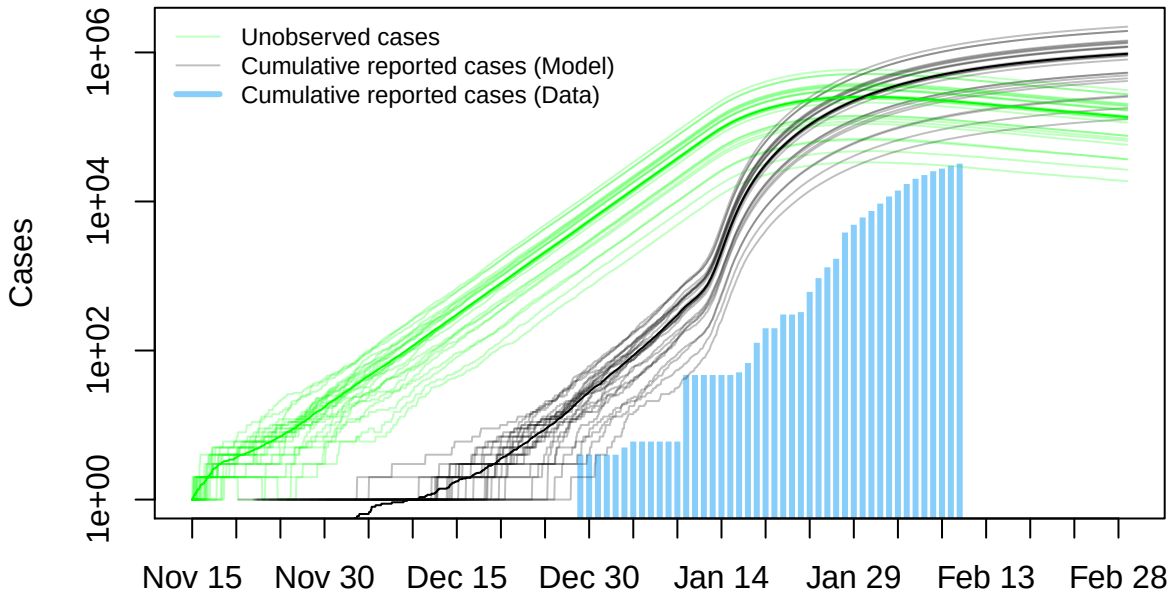
Scenario 2: Earlier spillover event

For comparison, we also show the results of an outbreak starting from one case around November 15. This scenario was prompted by initial reports that the time to most recent common ancestor was "mid-November" and recognition (from the fequency of imported cases elsewhere) that the outbreak was already quite large by mid-January. This model shows a large discrepancy between predicted and observed cases, supporting the later date of emergence envisioned in Scenario 1.

NOTE: Parameters set to account for different intervention date.

```
start <- as.Date('11/15/2019',format='%m/%d/%Y')
set.seed(1292020)                  #set seed
out2 <- evaluate.model(params=list(beta0=0.657, sigma=1/6.4, z=56, b=0.143, a0=0.0446, w=56, dt=0.05),
                       init = list(S=59002000,
                                   E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
                                   I1=1, I2=0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
                                   R=0, Ru=0, Q=0),
                     nsims=25, nstep=NULL, start=start, today=today)
plot.model2(out2)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 104 y values <= 0 omitted
## from logarithmic plot
```




```
#plot.summary(out3)
```

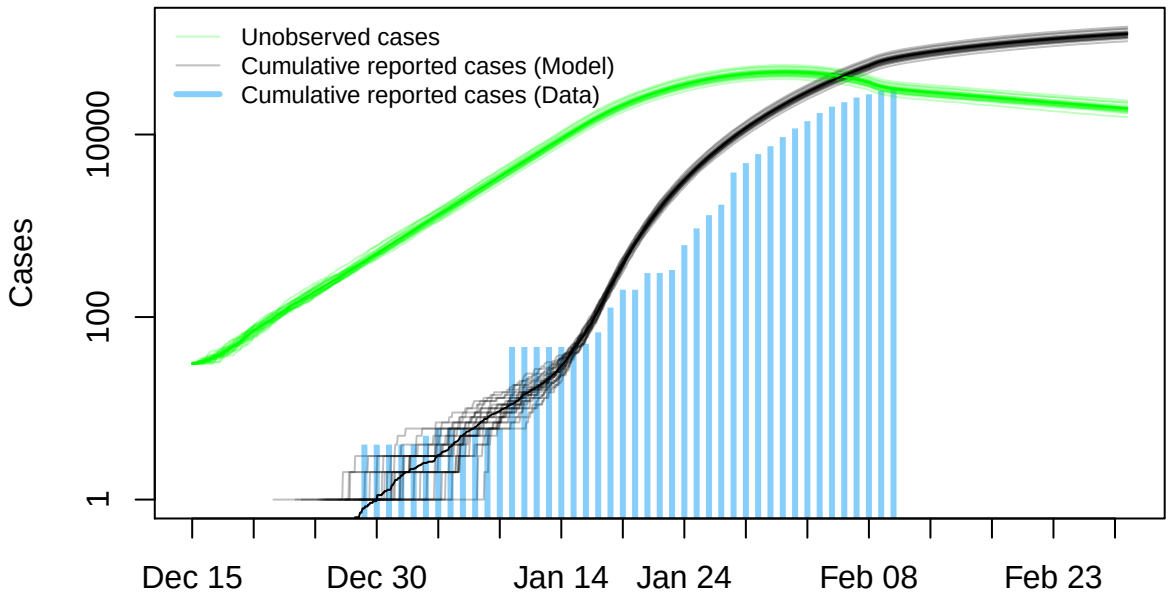
Scenario 3: Multiple simultaneous exposures

A third scenario consider an outbreak starting from 15 cases around December 15. This scenario is prompted by the report that the initial cluster was associated with the Huanan Seafood market and represents the possibility of multiple exposures in a single spillover event. Multiple exposures is based on 47 cases with longstanding exposure to the market distributed aming the E and I classes.

NOTE: Need to specify parameters here to adjust time of recovery rate increase since different start date.

```
start <- as.Date('12/15/2019',format='%m/%d/%Y')
set.seed(1252020)                    #set seed
out3 <- evaluate.model(params=list(beta0=0.657, sigma=1/6.4, z=30, b=0.143, a0=0.0446, w=26, dt=0.05),
                       init = list(S=59002000,
                                   E1=5, E2=5, E3=5, E4=5, E5=5, E6=5,
                                   I1=1, I2=0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
                                   R=0, Ru=0, Q=0),
                     nsims=25, nstep=NULL, start=start, today=today)
plot.model2(out3)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 132 y values <= 0 omitted
## from logarithmic plot
```

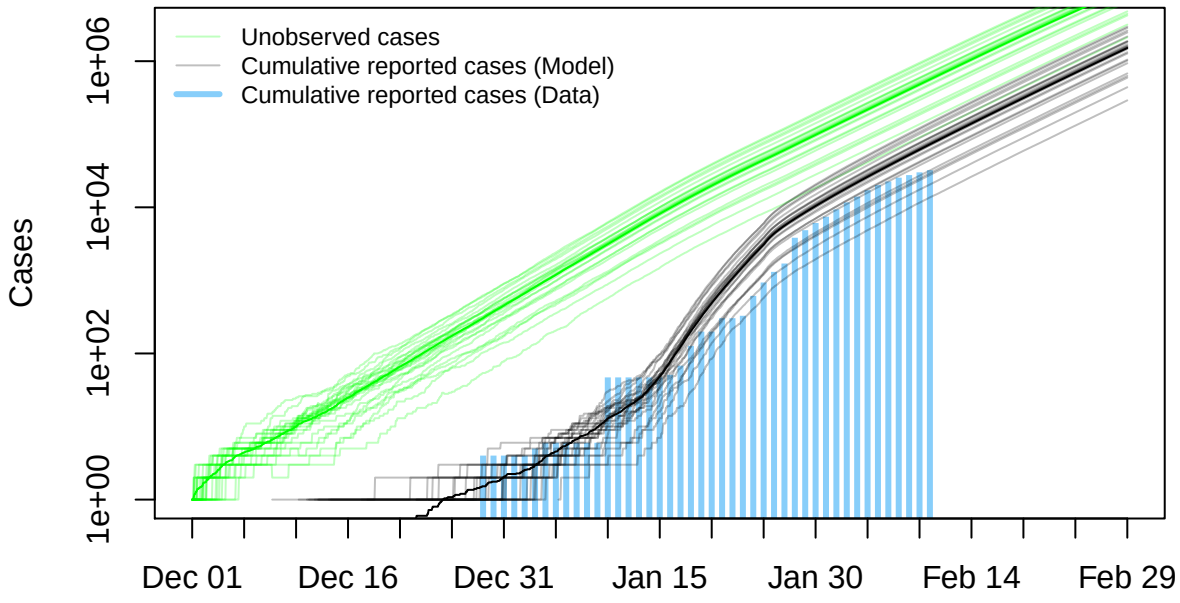


Scenario 4: Ineffectiveness of containment

For contrast, a fourth scenario supposes a mass exposure in the seafood market and (hypothetically) that interventions have had no effect on the isolation rate. This scenario is not considered to be epidemiologically plausible, but is included here for comparion. What is most interesting about this scenario is that the expected case notification curve is very consistent with the observed data. This suggests that existing case notification data may not be not sufficient to detect whether interventions have been effective.

```
start <- as.Date('12/01/2019',format='%m/%d/%Y')
set.seed(1252020)                    #set seed
out4 <- evaluate.model(params=list(beta0=0.657, sigma=1/6.4, z=3000, b=0.143, a0=0.0446, w=40, dt=0.05)
                       init = list(S=59002000,
                                   E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
                                   I1=1, I2=0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
                                   R=0, Ru=0, Q=0),
                     nsims=25, nstep=NULL, start=start, today=today)
plot.model2(out4)
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 154 y values <= 0 omitted
## from logarithmic plot
```

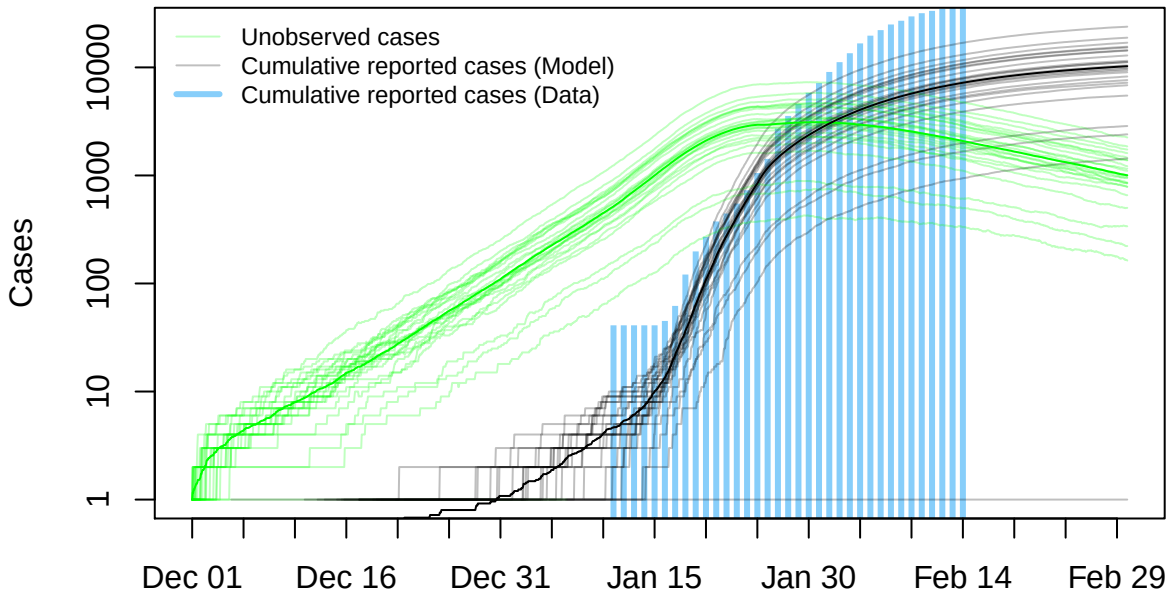


Scenario 5: Reduced transmissibility in undetected (weakly symptomatic) cases

Shaman et al. (personal communication) have estimated that transmissibility in undetected cases is approximatey 52% of that in detected cases. Here we return to our baseline scenario, but reduce transmissibility of undetected cases asccording to a coefficient $c = 0.52$. If the other features of this scenario (e.g. case isolation rate, etc.) are taken as given, such a reduction is inconsistent with the observed epidemic curve. Presumably, our model could be made consistent with this estimate if a different time-varying function for case isolation was chosen.

```
start <- as.Date('12/01/2019',format='%m/%d/%Y')
set.seed(1292020)                    #set seed
out5 <- evaluate.model(params=list(beta0=0.657, sigma=1/6.4, z=45, b=0.143, a0=0.0446, w=40, c=0.52,  d
                                   E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
                                   I1=1, I2=0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
                                   R=0, Ru=0, Q=0),
                       nsims=25, nstep=NULL, start=start)
#plot.model(out5, log='y')
plot.model2(out5, log='y')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 76 y values <= 0 omitted
## from logarithmic plot
```



Scenario 6: Short serial interval

Nishiura et al (https://www.medrxiv.org/content/10.1101/2020.02.03.20019497v1) have estimated the serial interval to be 2.6 days, suggesting that a considerable amount of secondary transmission may be due to presymptomatic cases. To approximate this scenario, this scenario allow the latent class to be infectious. This scenario is starkly inconsistent with the oberved case notification curve.

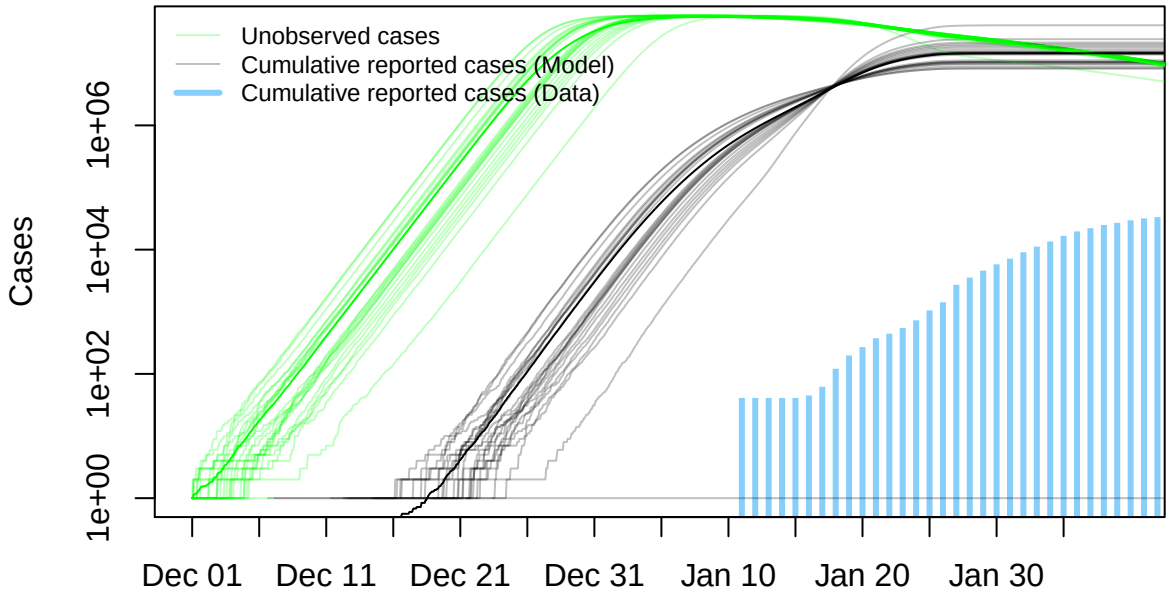
```
#this version of the model allows latent infections to transmit
onestep <- function (x, params) {  #function to calculate one step of stochastic SIR

  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]

  E <- E1+E2+E3+E4+E5+E6

  I1 <- x[9]                             #detected infectious classes
  I2 <- x[10]
  I3 <- x[11]
  I4 <- x[12]

  Iu1 <- x[13]                             #undetected infectious classes
  Iu2 <- x[14]
  Iu3 <- x[15]
  Iu4 <- x[16]

  I.detected <-I1+I2+I3+I4
  I.undetected <- Iu1+Iu2+Iu3+Iu4
  I <- I1+I2+I3+I4+Iu1+Iu2+Iu3+Iu4     #total infections
```

```
H <- x[17]                          #local variable for hospitalized
Ru <- x[18]                         #local variable for undetected recovereds

C <- x[19]    # local variable for notifications

N <- S+E1+E2+E3+E4+E5+E6+I1+I2+I3+I4+Iu1+Iu2+Iu3+Iu4+H+Ru          # total size of population

t <- x[20]                          #get current time


with(                               #use with to simplify code
     as.list(params),
     {
       gammai <- 4*gamma(z=z, b=b, a0=a0, t=as.numeric(t))  # multiplier 4 for pseudo stages
       sigmai <- 6*sigma  # multiplier 6 for pseudo stages
       etat <- eta(t)
      # betat <- beta(beta0, t)

       rates <- as.numeric(c(beta0*I.detected/N+beta0*c*I.undetected/N+beta0*c*E/N,
                 sigmai, sigmai, sigmai, sigmai, sigmai, sigmai,   # movements out of E
                 gammai, gammai, gammai, gammai,                   # movements out of I (detected)
                 b, b, b, b,                                       # movements out of I (undetected)
                 etat))

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

       # transition probabilities

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

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

       p[2,]  <- c(0, exp(-rates[2]*dt), 1-exp(-rates[2]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       p[3,]  <- c(0, 0, exp(-rates[3]*dt), 1-exp(-rates[3]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       p[4,]  <- c(0, 0, 0, exp(-rates[4]*dt), 1-exp(-rates[4]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       p[5,]  <- c(0, 0, 0, 0, exp(-rates[5]*dt), 1-exp(-rates[5]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       p[6,]  <- c(0, 0, 0, 0, 0, exp(-rates[6]*dt), 1-exp(-rates[6]*dt), 0, 0, 0, 0, 0, 0, 0, 0, 0,
       p[7,]  <- c(0, 0, 0, 0, 0, 0, exp(-rates[7]*dt), (1-exp(-rates[7]*dt))*q(t,w), 0, 0, 0, (1-exp

       p[8,]  <- c(0, 0, 0, 0, 0, 0, 0, exp(-rates[8]*dt), 1-exp(-rates[8]*dt), 0, 0, 0, 0, 0, 0, 0,
       p[9,]  <- c(0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[9]*dt), 1-exp(-rates[9]*dt), 0, 0, 0, 0, 0, 0,
       p[10,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[10]*dt), 1-exp(-rates[10]*dt), 0, 0, 0, 0, 0
       p[11,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[11]*dt), 0, 0, 0, 0, 1-exp(-rates[11]*dt)

       p[12,]  <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  exp(-rates[12]*dt), 1-exp(-rates[12]*dt), 0, 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), 0,
       p[14,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  exp(-rates[14]*dt), 1-exp(-rates[14]*dt),
       p[15,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  exp(-rates[15]*dt), 0, 1-exp(-rates[15]


       p[16,] <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, exp(-rates[16]*dt), 0, 1-exp(-rates[1

       # 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)
        states1[17] <- states1[17]+Ru  #add formerly Recovered undetected cases
        states1[18] <- states1[18]+C  #add formerly notified cases

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

```
start <- as.Date('12/01/2019',format='%m/%d/%Y')
set.seed(1292020)                  #set seed
out6 <- evaluate.model(init = list(S=59002000,
                                   E1=0, E2=0, E3=0, E4=0, E5=0, E6=0,
                                   I1=1, I2=0, I3=0, I4=0, Iu1=0, Iu2=0, Iu3=0, Iu4=0,
                                   R=0, Ru=0, Q=0),
                       nsims=25, nstep=NULL, start=start)
#plot.model(out5, log='y')
plot.model2(out6, log='y')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 113 y values <= 0 omitted
## from logarithmic plot
```



To Do

- Add references
- Update R_0 calculations