Age-structured transmission dynamic models of RSV

Tutorial 1 for transition

Gigi (Zhe Zheng)[1] Dan (Daniel Weinberger) Ginny (Virginia Pitzer)

2022-12-22

[1]zhe.zheng@yale.edu; zhe.zheng@aya.yale.edu; gigi.zhe.zheng@gmail.com

References

This age structured transmission dynamic model of RSV was first developed by Ginny Pitzer in her publication: https://doi.org/10.1371/journal.ppat.1004591. With the help of Dan Weinberger, we translated Ginny's Matlab codes to R codes. We further developed/modified this model, resulting in several mechanistic models corresponding to a variety of scenarios. Please refer to:

- https://doi.org/10.1001/jamanetworkopen.2021.41779
- https://doi.org/10.1038/s41541-022-00550-5
- https://doi.org/10.1101/2022.11.10.22282132

The complete code example for this tutorial please refer to - https://github.com/weinbergerlab/RSV_metapop/tree/master/SimpleModel

The Compartmental Model Structure (states of transmission dynamics)

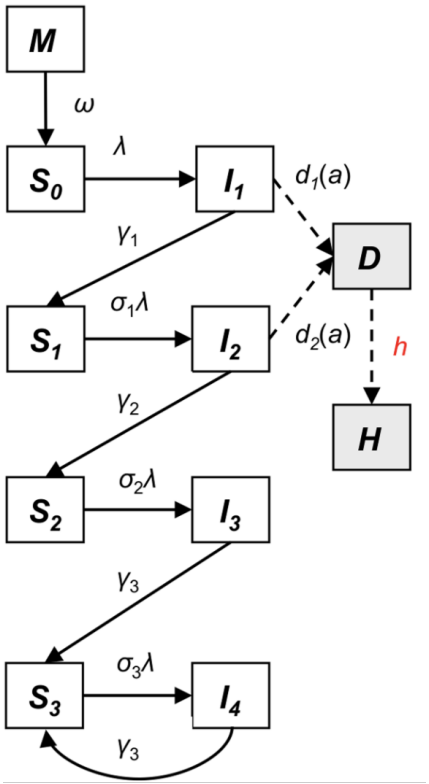
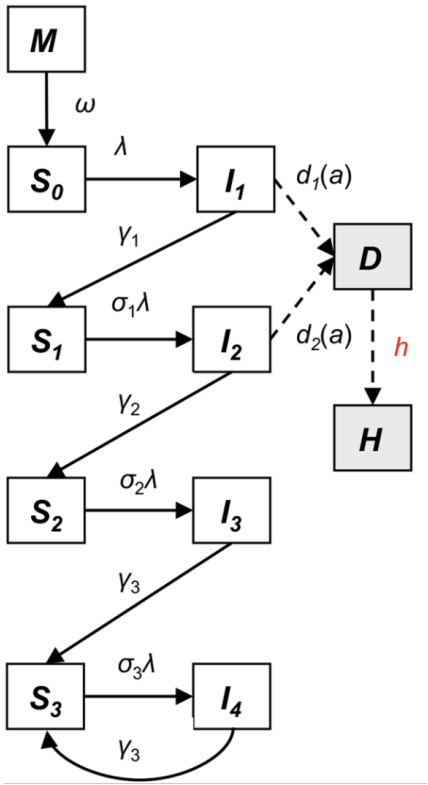


Figure 1: Ginny's model

- Each white compartment correspond to an underlying status in transmission dynamics. These statuses are **unobservable**.
  - **M** All infants born are born with protective maternal immunity, which wanes exponentially. The waning rate is ω.
  - $\boldsymbol{S_0}$ After protective maternal immunity wanes, infants are susceptible to RSV infection. Being exposed to infection, susceptible infants progress to infected status at rate λ.
  - $\boldsymbol{I_0}$ We ignored the exposed status (short and do not affect transmission dynamics) and assumed infected individuals are infectious. The first-time infected individuals recovered at rate γ_1
  - $\boldsymbol{S_n}$ Here n is the number of previous infections. Previous infection reduced the risk of re-infection. $\sigma_1 \sim \sigma_3$
  - $\boldsymbol{I_n}$ Here n is the number of infections. Infectious periods are shorter in subsequent infections. $\gamma_2 \sim \gamma_3$

The Compartmental Model Structure (rates)



Parameters in red are estimated for each state

Parameters in blue are from published literature (either as fixed value inputs or ranges as priors)

- Rates in this compartmental model.
  - ω is the waning rate of protective maternal immunity

$$\omega = \frac{1}{\text{duration of maternal immunity}}$$

  - λ is the force of infection

$$\lambda = \beta_0(1 + A\cos(2\pi\nu t - \phi))I$$

ϕ has a minus sign for moving towards a later time period.

$$\nu = \frac{1}{\text{period}}$$

$$I = I_1 + \rho_1 I_2 + \rho_2(I_3 + I_4)$$

here $\rho_1 \sim \rho_2$ are the relative infectiousness for 2nd and subsequent infections (probabilities)

  - $\gamma_1 \sim \gamma_3$ are the rates of recovery

$$\gamma_1 \sim \gamma_3 = \frac{1}{\text{duration of infectiousness}}$$

The Compartmental Model Structure (probabilities)

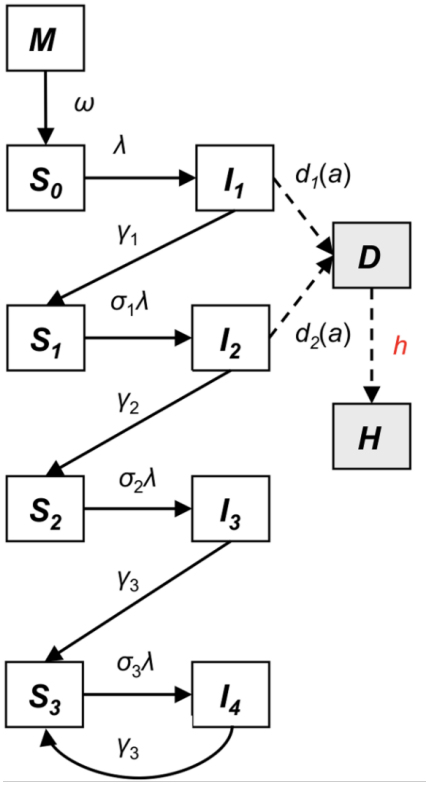


Figure 2: Ginny's model

- Probabilities in this compartmental model.
  - $\sigma_1 \sim \sigma_3$ are the relative risks of infection following 1st, 2nd, 3rd+ infections
  - $\rho_1 \sim \rho_2$ are the relative infectiousness for 2nd and subsequent infections
  - $d_1(a) \sim d_2(a)$ are the age- and infection-specific risks of developing lower respiratory tract illness given infections (LRTIs). In Ginny's model, only the first two infections will result in LRTIs. Later, we modified the model to assume subsequent infections will also lead to LRTIs $d_3(a) \sim d_4(a)$. This is very important for model calibration in older adult populations (For example, this will be needed in the cost-effectiveness of older adult RSV vaccine project).
  - $h_1(a) \sim h_4(a)$ are the probability of requiring inpatient care given LRTIs. These probabilities are also age- and infection-specific.
- From published literature (Either as fixed value inputs or ranges as priors)

The Compartmental Model Structure (observed states for calibration)

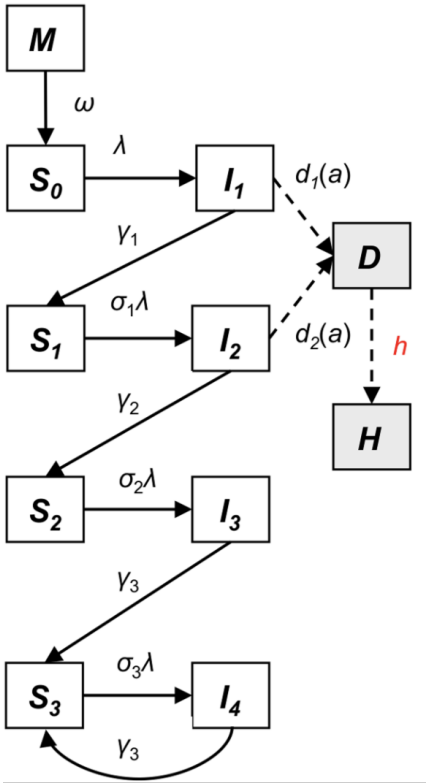


Figure 3: Ginny's model

- Each white compartment correspond to a disease status. These statuses are **observable**. These outputs of transmission dynamic models will be used for the model calibration.
  - **D** Individuals in each age group who develop lower respiratory tract illness. We can find this information in state emergency department visits and state inpatient visits.
  - **H** Individuals in each age group who are admitted to hospitals because of RSV infection. We can find this information in state inpatient visits. For my previous work, I calibrated the transmission model outputs to time-series generated from State Inpatient Databases.
  - Click to see information on State Emergency Department Databases and State Inpatient Databases.

The Compartmental Model Structure (observed statuses for calibration)

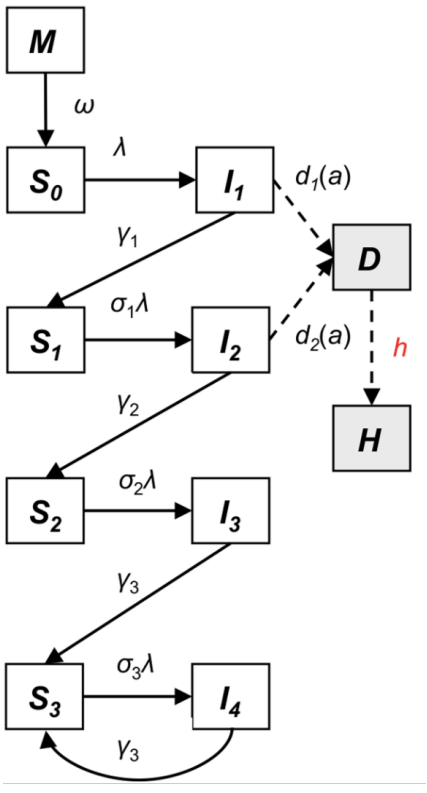


Figure 4: Ginny's model

- Each white compartment correspond to a disease status. These statuses are **observable**. These outputs of transmission dynamic models will be used for the model calibration.
  - Note that although the model outputs specify the number of previous infection, we are unlikely to observe this information in real world.
  - Note that State Emergency Department Databases and State Inpatient Databases only captured the RSV visits that are tested and recorded. This may not be a problem in pediatric populations but it needs to be corrected for the testing and recording ratios in adult populations.

R Code in Details (Basic Age-structure MSIS model)

Setting up the packages

```
## load required packages
library(deSolve)
# for solving the ordinary differential equations
library(RColorBrewer)
# color palettes for making plots
library(reshape2)
# for reshape data frames
```

Setting up inputs

```
# T is time points. T contains both the burn-in period
# and evaluation period.
# N_ages is number of age groups.

Pop1 <- readRDS('./SimpleModel/data_and_parms/pop1.rds')
#initial population, by age group.

B <-  readRDS('./SimpleModel/data_and_parms/Birth_rate.rds')
# birth rate in each age group
# a matrix with T rows and  N_ages columns
# The first column is the population birth rate
# They are born into the 0 month age group
# The rest columns are all zeros (1 months to 80+ age group)
# For U.S., you will find this information in CDC wonder
```

Click to see information on CDC wonder

Setting up inputs

```
c2 <- readRDS( './SimpleModel/data_and_parms/c2.rds')
# The contact patterns in each age group.
# The patterns affect the age-specific likelihood
# of a susceptible individual come into contact with
# an infectious individual.
# The exact values does not matter because we will
# estimate beta_0.
# This information can be found in published literature
# Then you may consider rearrange the matrix to
# reflect the age groups in your model
# I wrote an r script to rearrange the contact matrix
# see contactmatrix.r under code source folder
```

Age structure in this example

In Ginny's paper, the age groups are as follows: The <12 month olds, were divided into monthly age classes. The remaining population was divided into 6 classes: 1–4 years old, 5–9 years, 10–19 years, 20–39 years, 40–59-years, and 60+ years old. In this example, the 1-4 year old are divided into 12 month age classes as well. In under 1 year old, we calibrate the model to age category of every 3 month.

How to divide age depends on the goal of your project. It relates closely to the parameters for model calibration and the difficulty of model calibration.

We set names of age groups

```
N_ages <- length(Pop1)
agenames <- paste0('Agegrp', 1:N_ages)
#Could replace this with vector of actual age names
```

Initialize the compartments (states)

Please refer to page 3: The Compartmental Model Structure (states of transmission dynamics)

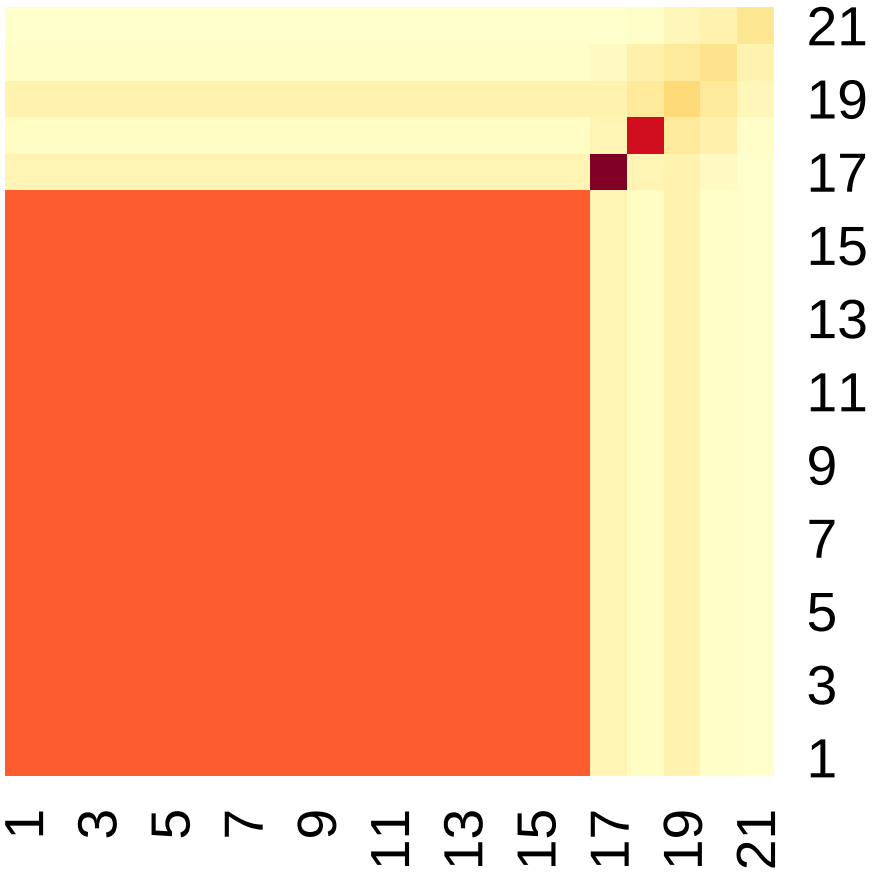
```
StateNames <- c('M','S0','I1','S1','I2','S2','I3','S3','I4')

# N age groups x K states
yinit.matrix <- array(NA, dim=c(N_ages, length(StateNames)))

# assign row names and column names
dimnames(yinit.matrix)[[1]] <- agenames
dimnames(yinit.matrix)[[2]] <- StateNames

# Initializes population with infants under 3 months
# are protected by maternal immunity and
# with 1 infected person per age group in other age groups
yinit.matrix[,c('S1','I2','S2','I3','S3','I4')]  = 0
yinit.matrix[,'M'] = c(Pop1[1:3], rep(0,N_ages-3))
yinit.matrix[,'S0'] = c(rep(0,3),Pop1[4:N_ages]-rep(N_ages-3))
yinit.matrix[,'I1'] = c(rep(0,3), rep(1,N_ages-3))
```

Vectorize the states for ODE input

```
#Vectorize the ynit matrix
yinit.vector <- as.vector(yinit.matrix)

# Create array that has the labels by age, state
# and use this to name the yinit.vector
name.array <- array(NA, dim=dim(yinit.matrix))
for(i in 1:dim(name.array)[1]){
 for(j in 1:dim(name.array)[2]){
   name.array[i,j] <- paste(dimnames(yinit.matrix)[[1]][i],
                            dimnames(yinit.matrix)[[2]][j])
    }
  }

name.vector <- as.vector(name.array)
names(yinit.vector) <- name.vector
```

Contact matrix in our case

```
n.cols=100
nice.cols <-  colorRampPalette(brewer.pal(9,
                                          "YlOrRd"))(n.cols)
heatmap(c2/sum(diag(c2)),
        Rowv=NA, Colv=NA, scale='none', col=nice.cols)
```



Section outlines

- In the first section, we will learn how to run transmission model with all parameters known.

- In the second section, we will learn to use maximum a posteriori / maximum likelihood estimation (MLE) to estimate the unknown parameters (one value for each unknown parameter).

- In the third section, we will learn to use STAN, a probabilistic programming language for statistical inference[2], to get the 95% credible intervals from the posterior samples of these unknown parameters.

[2] https://en.wikipedia.org/wiki/Stan_(software)

Section 1: run transmission model with all parameters known

We first assign values to all required parameters, including rates and probabilities. Most of these are described in Table 2 of Pitzer et al, PLOS Pathogens[3]

```
##########################################
# omega = 1/duration of maternal immunity
DurationMatImmunityDays= 30 #days
##########################################

##########################################
# gamma1~gamma3 = 1/duration of infectiousness
# Duration in days
dur.days1 <- 10 #days
dur.days2 <- 7 #days
dur.days3 <- 5 #days
##########################################
```

[3] https://journals.plos.org/plospathogens/article?id=10.1371/journal.ppat.1004591

Parameters related to the force of transmission

$$\lambda = \beta_0(1 + A\cos(2\pi\nu t - \phi))I$$

β_0 is the effective contact rate, meaning the number of infections per unit time per susceptible per infected.

β_0 = per capita transmission probability * total contact rate

```
#These parameters will likely be estimated in practice
#############################################
#Seasonal components--
Amp = 0.2 #Seasonal amplitude
phi = 3.327749 #Seasonal phase shift
# (depends on the starting month;here 0=peak @ July 1)

#############################################
# per capita transmission probability
baseline.txn.rate <- 9
# this is now for the entire infectious period
# it needs to divide the length of infectious period
#############################################
```

Parameters related to the force of transmission (continued)

β_0 = total contact rate * per capita transmission probability

Note, In Density Dependent transmission, the contact rate (c) depends on the population density. In Frequent Dependent transmission, the contact rate (c') does not depend on the population density. (In this example, we assumed frequency-dependent)

```
##############################################
c=c2 # contact rate

q=1 # use this to switch between density (q=0) vs
# frequency-dependent (q=1) transmission
##############################################
```

To better understand the different in Frequent Dependent transmission and Density Dependent transmission, please check out: https://parasiteecology.wordpress.com/2013/10/17/density-dependent-vs-frequency-dependent-disease-transmission/

Probabilities related to the number of infections

Please refer to page 5: The Compartmental Model Structure (probabilities)

```
##################################################
#Relative infectiousness for 2nd and subsequent infections
rho1 = 0.75
rho2 = 0.51

##################################################
#Relative risk of infection following 1st, 2nd, 3rd+ infections
sigma1=0.76
sigma2=0.6
sigma3=0.4
##################################################
```

Age-specific probabilities related to the number of infections

In this example, the LRI and hospitalization probabilities are from several cohort studies. These studies reported the probabilities in aggregated age groups (every 3 month). In later section, if you would like to fit the model outputs to self-defined age groups, you will need to come up with the age-specific probabilities first. This can be done by creating a polynomial regression fit to the reported probabilities of the cohort studies.

```
##########################################
# LRI probability
##########################################

#proportion of first infections that are LRI (by age)
delta1=c(rep(.40,3),rep(.39,3),rep(.21,3),
         rep(.20,3),0.16,rep(.14,3),rep(0.05,N_ages-16))
#proportion of second infections that are LRI
delta2=.5*delta1
#proportion of third infections that are LRI
delta3=.7*delta2
```

Age-specific probabilities related to the number of infections (continued)

```
#########################################
# Hospitalization probability
#########################################

#proportion of first infection that are hospitalized
hosp1=c(.18*rep(.40,3),0.08*rep(.39,3),
        0.07*rep(.21,3),0.06*rep(.20,3),0.06*0.16,
        0.05*rep(.14,3),0.02*rep(0.05,N_ages-16))
# = hosp prob given LRI * LRI prob given infection

#proportion of second infection that are hospitalized
hosp2=.4*hosp1

#proportion of subsequent infection that are hospitalized
#(The last two probabilities come from the previous
#fitting of the transmission dynamic model)
hosp3=c(rep(0,N_ages-2),0.00001,0.00004)
```

Parameters governing population dynamics

```
##############################################
# birth rate
# Matrix: T rows, N_ages columns; columns 2:N_ages all 0s
PerCapitaBirthsYear=B
##############################################
# net rate of crude deaths (+) and immigration (-)
# You should calibrate this parameter
# so we can reproduce the population growth
um= -0.0002227 #(from all age groups)

##############################################
# Aging rate = 1/width age class (months)
# Vector of long N_age
WidthAgeClassMonth = c(rep(1,times=12),
      rep(12,times=4),  60, 120, 240, 240, 240)
##############################################
```

Save parameters in a list

```
parms<-list(PerCapitaBirthsYear=PerCapitaBirthsYear,
            DurationMatImmunityDays=DurationMatImmunityDays,
            WidthAgeClassMonth=WidthAgeClassMonth,
            um=um, # net growth rate
            Amp=Amp, # seasonal amplitude
            phi=phi, # seasonal peak timing
            rho1=rho1, # Relative infectiousness (2nd)
            rho2=rho2, # Relative infectiousness (3rd+)
            dur.days1=dur.days1,# Duration of infectiousness
            dur.days2=dur.days2,# Duration of infectiousness
            dur.days3=dur.days3,# Duration of infectiousness
            yinit.matrix=yinit.matrix, # initial states
            baseline.txn.rate = baseline.txn.rate,
            q=q, # Frequency or Density dependent
            contact=c2, # contact matrix
            sigma1=sigma1, # Relative risk of infection (2nd)
            sigma2=sigma2, # Relative risk of infection (3rd)
            sigma3=sigma3, # Relative risk of infection (4th+)
            time.step='month')
```

Read in the model

```
#Read in the model
source('./SimpleModel/data_and_parms/simple_model.R')
```

In the following slides, we will show the source function of ordinary differential equations. You can modify this function to reflect different assumptions.

Note, please modify the r script **simple_model.R** in the folder **data_and_parms** under **SimpleModel**. In this PDF, the function of transmission dynamic model is separated into several chunks for the tutorial purpose.

Source function of the transmission dynamic model

```
simple_model <- function(t,y,parms,time.step='month'){

  # read in initial states and their names
  States<-array(y, dim=dim(parms$yinit.matrix))
  dimnames(States) <- dimnames(parms$yinit.matrix)

  # unify the time unit of parameter inputs
  if(parms$time.step=='month'){
    period=12
    length.step=30.44 #days
  }else if(parms$time.step=='week'){
    period=52.1775
    length.step=7 #days
  }
```

Source function of the transmission dynamic model (continued)

Note: Here we need to convert all the rates from 1/days to 1/length.step

```
# waning rate of maternal immunity (by time step)
omega = 1/(parms$DurationMatImmunityDays/length.step)

# aging rate (by time step)
mu = 1/parms$WidthAgeClassMonth
if (parms$time.step == "week") {
    mu = 1/(WidthAgeClassMonth * 4.345)
}

# rate of recovery of first infection
gamma1 = 1/(parms$dur.days1/length.step)
# rate of recovery of second infection
gamma2 = 1/(parms$dur.days2/length.step)
# rate of recovery of third infection
gamma3 = 1/(parms$dur.days3/length.step)
gamma4 = gamma3
# gamma3 stands for rate of recovery from subsequent
# infection
```

Source function of the transmission dynamic model (continued)

```
# Relative risk of infection (2nd)
sigma1 = parms$sigma1
# Relative risk of infection (3rd)
sigma2 = parms$sigma2
# Relative risk of infection (4th+)
sigma3 = parms$sigma3

# Relative infectiousness (2nd)
rho1 = parms$rho1
# Relative infectiousness (3rd+)
rho2 = parms$rho2
```

Source function of the transmission dynamic model (continued)

```
# Pull out the states for the model as vectors
M <- States[, "M"]  # protected by maternal immunity
S0 <- States[, "S0"]  # purely susceptible population
I1 <- States[, "I1"]  # first time infection (infectious)

S1 <- States[, "S1"]
# susceptible population with build-up immunity
I2 <- States[, "I2"]  # second time infection

S2 <- States[, "S2"]
# susceptible population with lower risk of re-infection
I3 <- States[, "I3"]  # third time infection

S3 <- States[, "S3"]
# susceptible population with lowest risk of re-infection
I4 <- States[, "I4"]  # subsequent time infection

N_ages <- length(M)  # the number of age groups
```

Source function of the transmission dynamic model (continued)

```
## parameter related to force of infection ################
# per capita transmission probability
baseline.txn.rate=parms$baseline.txn.rate
# transmission probability per unit time
b <- baseline.txn.rate/ (parms$dur.days1/length.step)
q=parms$q # q depends on transmission type
# (whether depends on population density or not)
contact=parms$contact # c2 is the contact matrix
# transmission probability per unit time in each age group
beta <- (b/100)/(sum(yinit.matrix)^(1-q))*contact
# 100 is a scaling factor for the contact matrix we choose
# (see Ginny's paper and Matlab code for details)
# this does not matter because most likely
# you will need to estimate baseline.txn.rate
```

Source function of the transmission dynamic model (continued)

```
## parameter related to force of infection ################
Amp=parms$Amp # seasonal amplitude
phi=parms$phi # seasonal phase shift
#seasonality
seasonal.txn <- (1+Amp*cos(2*pi*(t-phi*period)/period))

# seasonal transmission probability
beta_a_i <- seasonal.txn * beta
infectiousN <- (I1+rho1*I2+rho2*I3+rho2*I4)/sum(States)
# for frequency dependent transmission

lambda <- infectiousN %*% beta_a_i # force of transmission
lambda <- as.vector(lambda) # vectorize force of transmission
```

Source function of the transmission dynamic model (continued)

```
# create a matrix to record the changing variables
dy <- matrix(NA, nrow=N_ages, ncol=ncol(States))
colnames(dy) <- colnames(States)

period.birth.rate <-
  log(parms$PerCapitaBirthsYear[t,]+1)/period
# get period birth rate from annual birth rate
# see the following page for birth rate calculation

#um is death rate
um=parms$um
#mu represents aging to the next class
Aging.Prop <- c(0,mu[1:(N.ages-1)])
```

Period birth rate calculation

For birth rate[4]:

The weekly per capita birth rate B_w is equal to log(1+B)/52.18 because the data on the birth rate is annual and the differential equation model inherently assumes that growth is occurring exponentially. So if the annual birth rate is equal to 12 per 1000 per year (B=0.012), for example, then we are assuming that:

$$N_1 = N_0 e^{(B_w * 52.10)}$$

where N_1 is the population after 1 year and N_0 is the baseline population, and $N_1 = N_0 * (1 + B)$.

$$1.012 = 1 * e^{(B_w * 52.18)}$$

$$B_w = log(1.012)/52.18$$

. The same goes for getting monthly birth rate.

[4] https://journals.plos.org/plospathogens/article/file?id=10.1371/journal.ppat.1004591.s016&type=supplementary

Source function of the transmission dynamic model (continued)

For the equations, please refer to the supplementary document of Ginny's paper

```
# ordinary differential equations
  dy[,'M'] <- period.birth.rate*sum(States) -
    (omega+(mu+um))*M +
    Aging.Prop*c(0,M[1:(N_ages-1)])

  dy[,'S0'] <- omega*M -
    lambda*S0 -
    (mu + um)*S0 +
    Aging.Prop*c(0,S0[1:(N_ages-1)])

  dy[,'I1'] <-   lambda*S0 -
    (gamma1 + mu + um)*I1 +
    Aging.Prop*c(0,I1[1:(N_ages-1)])
```

Source function of the transmission dynamic model (continued)

```
dy[,'S1'] <- gamma1*I1 -
  sigma1*lambda*S1 -
  (mu+um)*S1 +
  Aging.Prop*c(0,S1[1:(N_ages-1)])

dy[,'I2'] <- sigma1*lambda*S1 -
  gamma2*I2-(mu + um)*I2 +
  Aging.Prop*c(0,I2[1:(N_ages-1)])

dy[,'S2'] <- gamma2*I2 -
  sigma2*lambda*S2 -
  (mu+um)*S2 +
  Aging.Prop*c(0,S2[1:(N_ages-1)])

dy[,'I3'] <- sigma2*lambda*S2 -
  (gamma3 + mu+um)*I3 +
  Aging.Prop*c(0,I3[1:(N_ages-1)])
```

Source function of the transmission dynamic model (continued)

```
  dy[,'S3'] <- gamma3*I3 +
    gamma4*I4 -
    sigma3*lambda*S3 -
    (mu + um)*S3 +
    Aging.Prop*c(0,S3[1:(N_ages-1)])

  dy[,'I4'] <- sigma3*lambda*S3 -
    gamma4*I4 -
    (mu + um)*I4 +
    Aging.Prop*c(0,I4[1:(N_ages-1)])

  derivs <- as.vector(dy)

  res <- list(derivs)

  return(res)
}
```

Run the model

NOTE: time step here is in months–you need to adjust seasonality accordingly for the time step you choose

```
 start_time = 1 # start date (years)
 tmax = nrow(B)
# end_time = 25 # end date (years)
my_times <- seq(start_time, tmax, by = 1)
# gives a sequence from start to end in increments of 1
```

Using the ODE function, we get the results of the transmission dynamic model[5].

```
results <- ode(y=yinit.vector, t=my_times,
               func=simple_model,
             parms=parms)
# y is the initial population in each state and age
# t is the time steps of evaluation
# func specify the ordinary differential equations
# parms are all the parameter inputs
```

[5]NOTE: here we are just simulating with set parameters, not fitting to data

Extract population growth in each age group and the entire population

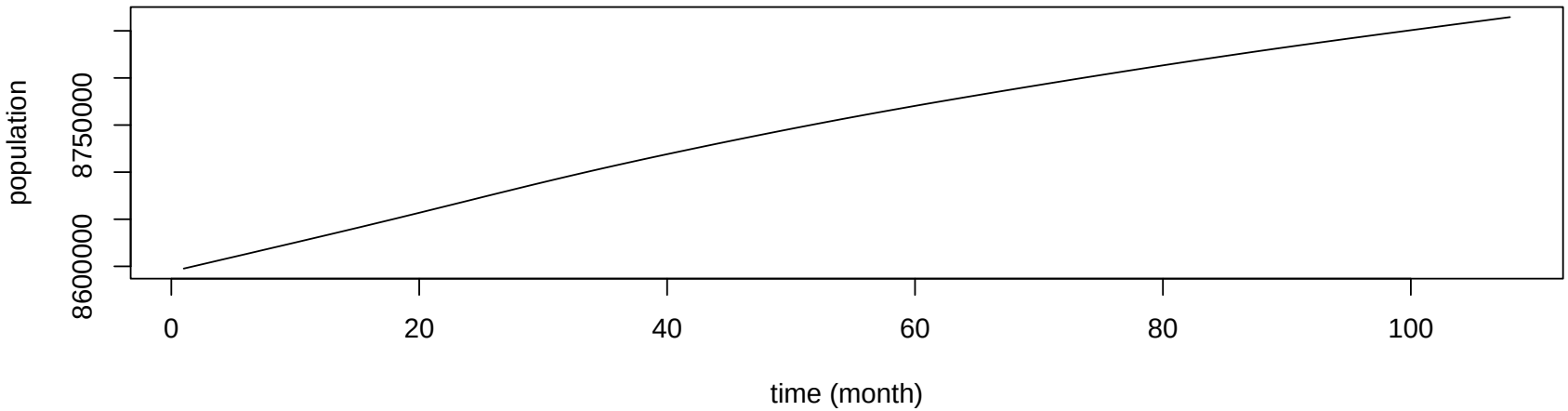
```
#Ginny Pitzer used a 40-50 YEAR burn in period
burnN <- 25*12 # you will need to evaluate
# whether after burn-in the model reach quasi-equilibrium
results.burned <- results[-c(1:burnN),]

# first get the total population in each time point
pop.all <- rowSums(results.burned[,-1])

# Then check the population growth in each age group
# from wide format to long format
all.m <- melt(results.burned[,
              grep('Agegrp',colnames(results.burned))])
# then we get the name of age group of each state
all.m$agegrp <- sub(" .*", "",all.m$Var2)
# sum within age groups to get the age-specific
# population in each time point
all.c <- dcast(all.m, Var1~agegrp, fun.aggregate = sum)
```

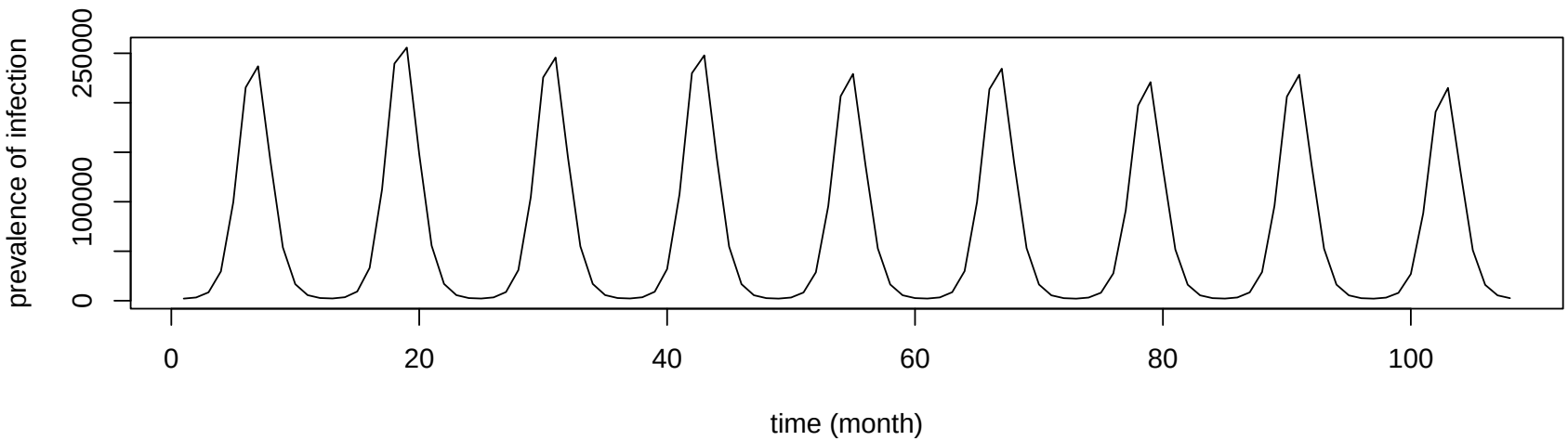
Exact the prevalence of infection from the results

```
##Any infected person
infected.cols <- results.burned[,
                 c(grep('I1', colnames(results.burned)),
                  grep('I2', colnames(results.burned)),
                  grep('I3', colnames(results.burned)),
                  grep('I4', colnames(results.burned)))]

## Sum up the infected individuals at each time point
infected.all <- apply(infected.cols,1,sum)# = rowsum

## this chunk of code get the prevalence of infection
## in each age group at each time point.
infected.cols.m <- melt(infected.cols)
infected.cols.m$agegrp <- sub(" .*", "",infected.cols.m$Var2 )
infected.cols.c <- dcast(infected.cols.m,
                         Var1~agegrp,
                         fun.aggregate = sum)
```

Visualize population growth and prevalence of infection

```
par(mfrow = c(2, 1))
plot(pop.all,type="l",xlab="time (month)",ylab="population")
plot(infected.all, type='l',xlab="time (month)",
     ylab="prevalence of infection")
```





Calculate the number of hospitalizations

Take out all parameters needed to calculate hospitalizations[6]

```
q=1 # frequency dependent
# transmission probability per unit time
b= parms$baseline.txn.rate/(parms$dur.days1/30.44)
contact=parms$contact # c2 is the contact matrix
#transmission probability per unit time in each age group
beta <- (b/100)/(sum(yinit.matrix)^(1-q))*contact
Amp=parms$Amp # seasonal amplitude
phi=parms$phi # seasonal phase shift

rho1=parms$rho1 # Relative infectiousness (2nd)
rho2=parms$rho2 # Relative infectiousness (3rd+)
sigma1=parms$sigma1 # Relative risk of infection (2nd)
sigma2=parms$sigma2 # Relative risk of infection (3rd)
sigma3=parms$sigma3 # Relative risk of infection (4th+)
t0=nrow(results.burned) # length of time for evaluation
```

[6] This is not needed in our case since we already save them in the environment.

Calculate the number of hospitalizations (continued)

Take out all states needed to calculate hospitalizations (S and I)

```
I1 <- results.burned[,grep('I1', colnames(results.burned))]
I2 <- results.burned[,grep('I2', colnames(results.burned))]
I3 <- results.burned[,grep('I3', colnames(results.burned))]
I4 <- results.burned[,grep('I4', colnames(results.burned))]
S0 <- results.burned[,grep('S0', colnames(results.burned))]
S1 <- results.burned[,grep('S1', colnames(results.burned))]
S2 <- results.burned[,grep('S2', colnames(results.burned))]
S3 <- results.burned[,grep('S3', colnames(results.burned))]
```

calculate the force of infection

```
##################Force of infection#################
lambda1=matrix(0,nrow=t0,ncol=N_ages)
 for (t in 1:t0)
 {lambda1[t,]<-as.vector((1+Amp*cos(2*pi*(t-phi*12)/12))*
                ((I1[t,]+rho1*I2[t,]+rho2*I3[t,]+rho2*I4[t,])
                 %*%beta)/sum(results.burned[t,]))}
## (see pages 4, 29, and 30 for detailed explanation)
```

Calculate the number of hospitalizations (continued)

λS_0 = number of new first infection

$\sigma_1 \lambda S_1$ = number of new second infection

$\sigma_2 \lambda S_2$ = number of new third infection

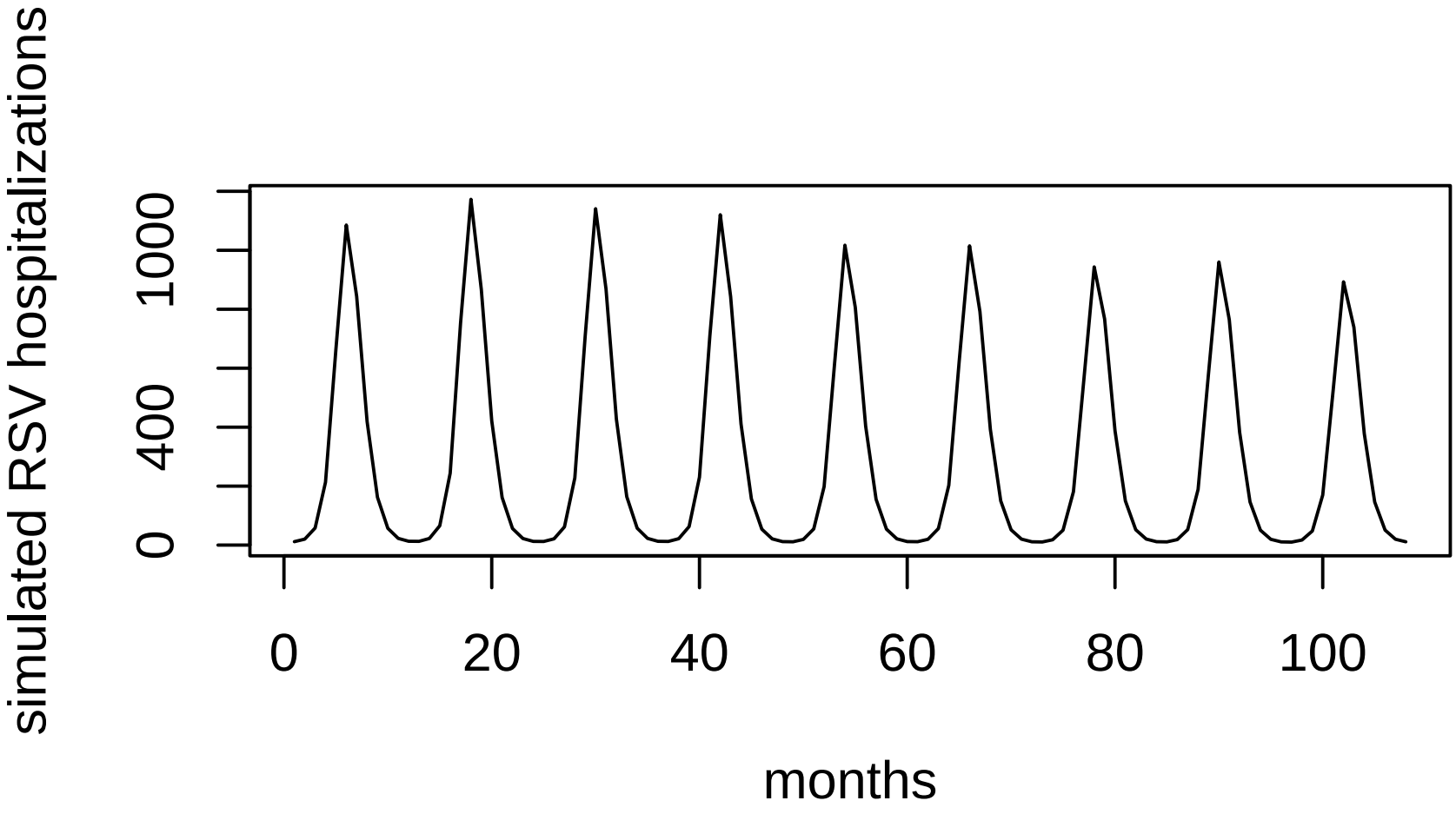
$\sigma_3 \lambda S_3$ = number of new subsequent infection

number of hospitalizations = the probability of hospitalization given infection * the number of new infection

```
####Number of hospitalizations by age###############
   H1=matrix(0,nrow=t0,ncol=N_ages)
   for (i in 1:N_ages){
     H1[,i]=hosp1[i]*S0[,i]*lambda1[,i]+
       hosp2[i]*sigma1*S1[,i]*lambda1[,i]+
       hosp3[i]*sigma2*S2[,i]*lambda1[,i]+
       hosp3[i]*sigma3*S3[,i]*lambda1[,i]}
```

Plot the number of hospitalizations over time

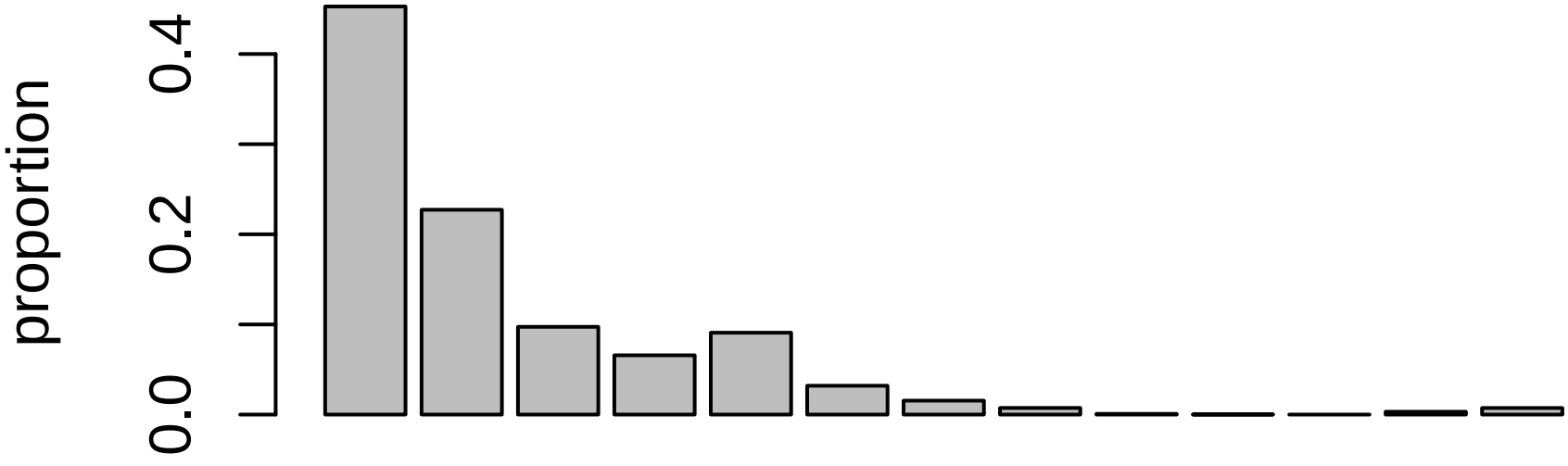
```
## time-series of number of hospitalizations ##
plot(1:108,rowSums(H1),type="l",xlab="months",
     ylab="simulated RSV hospitalizations")
```




```
## the decrease at the end is likely caused by
## the drop of birth rate 0.0115 vs 0.013 (in 1980)
```

Plot the age distribution of hospitalizations

```
## age distribution of hospitalizations ##
barplot(c(sum(colSums(H1)[1:3]),# 0-2 months
          sum(colSums(H1)[4:6]),# 3-5 months
          sum(colSums(H1)[7:9]),# 6-8 months
          sum(colSums(H1)[10:12]),# 9-11 months
          colSums(H1)[13:21])/sum(H1),
        # 1,2,3,4 years old, 5-9 years,
        # 10-19 years, 20-39 years,
        # 40-59-years, and 60+ years old.
        xlab="age group",ylab="proportion")
```



Section 2: use maximum a posteriori (variation of MLE) to estimate the unknown parameters

In this example, we will use maximum a posteriori to estimate the (1) per capita transmission probability: baseline.txn.rate, (2) seasonal amplitude: Amp, (3) seasonal peak timing: phi, (4) duration of maternal immunity: DurationMatImmunityDays.

We will treat the time-series and age distribution of RSV hospitalizations that we just simulated as observed status and calibrate our transmission dynamic models using MLE to estimate these "unknown" parameters.

```
###### save simulated time-series and age distribution
Hosp_sim <- as.integer(rowSums(H1)) # time-series

agedist_Sim <- c(sum(colSums(H1)[1:3]),
                 sum(colSums(H1)[4:6]),
                 sum(colSums(H1)[7:9]),
                 sum(colSums(H1)[10:12]),
                 colSums(H1)[13:21])/sum(H1)
                 # age distribution

sim <- list(Hosp_sim=Hosp_sim,agedist_Sim=agedist_Sim)
```

Create a list of known parameters

```
parmset<-list(PerCapitaBirthsYear=PerCapitaBirthsYear,
              WidthAgeClassMonth=WidthAgeClassMonth,
              um=um, # net growth rate
              rho1=rho1, # Relative infectiousness (2nd)
              rho2=rho2, # Relative infectiousness (3rd+)
              dur.days1=dur.days1,# Duration of infectiousness
              dur.days2=dur.days2,# Duration of infectiousness
              dur.days3=dur.days3,# Duration of infectiousness
              yinit.matrix=yinit.matrix, # initial states
              q=q, # Frequency or Density dependent
              contact=c2, # contact matrix
              sigma1=sigma1, # Relative risk of infection (2nd)
              sigma2=sigma2, # Relative risk of infection (3rd)
              sigma3=sigma3, # Relative risk of infection (4th+)
              time.step='month',
              t0=t0, # length of time for evaluation
              N_ages=N_ages) # number of age group
```

Create a function to calculate the log likelihood of given values

Read in the function

```
source('./code source/Loglikelihood_MSIS.R')
```

View log likelihood calculation function

```
fitmodel <-  function(parameters,dat) {
# takes the parameter values and dataset as inputs

# parameter related to R0 (baseline transmission rate)
  protrans <- parameters[1]
# parameter related to seasonal amplitude
  b1 <- parameters[2]
# parameter related to seasonal peak timing
  trans <- parameters[3]
# parameter related to the duration of maternal immunity
  DMD <- parameters[4]
```

Log likelihood calculation function (continued)

```
  Amp <- exp(b1) #ensure positive
  baseline.txn.rate <- exp(protrans) #ensure positive
# transform to its scale in the model
  phi <-  (2*pi*(exp(trans))) / (1+exp(trans))
  durx <- exp(DMD) #ensure positive

# we need this for the function and model
# to recognize these parameters
  durx <<- durx
  baseline.txn.rate <<- baseline.txn.rate
  Amp <<- Amp
  phi <<- phi

# Run transmission model with initial conditions
# and time steps defined above,
# and parameter values from function call
  results <- ode(y=yinit.vector, t=my_times,
                 func=simple_model,
                 parms=c(parmset,
                 baseline.txn.rate=baseline.txn.rate,
                 Amp=Amp,
```

Log likelihood calculation function (continued)

```
   t0 <- parmset$t0
  # make sure the evaluation period are the same
  results<- tail(results,t0) # get rid of burn-in period
  St <- results[,-1]

  I1 <- St[,grep('I1', colnames(St))]
  I2 <- St[,grep('I2', colnames(St))]
  I3 <- St[,grep('I3', colnames(St))]
  I4 <- St[,grep('I4', colnames(St))]
  S0 <- St[,grep('S0', colnames(St))]
  S1 <- St[,grep('S1', colnames(St))]
  S2 <- St[,grep('S2', colnames(St))]
  S3 <- St[,grep('S3', colnames(St))]

b=baseline.txn.rate/(parmset$dur.days1/30.44)
beta=(b/100)/(sum(yinit.matrix)^(1-parmset$q))*parmset$contact
```

Outcomes of interests given parameter inputs (function continued)

```
  #Force of infection
  lambda1=matrix(0,nrow=t0,ncol=parmset$N_ages)
  for (t in 1:t0)
  {lambda1[t,]<-as.vector((1+Amp*cos(2*pi*(t-phi*12)/12))*
                  ((I1[t,]+rho1*I2[t,]+rho2*I3[t,]+rho2*I4[t,])
                   %*%beta)/sum(St[t,]))}

 H1=matrix(0,nrow=t0,ncol=parmset$N_ages)
  for (i in 1:parmset$N_ages){
    H1[,i]=hosp1[i]*S0[,i]*lambda1[,i]+
      hosp2[i]*sigma1*S1[,i]*lambda1[,i]+
      hosp3[i]*sigma2*S2[,i]*lambda1[,i]+
      hosp3[i]*sigma3*S3[,i]*lambda1[,i]}
  H <- rowSums(H1)

  agedist <- c(sum(colSums(H1)[1:3]),sum(colSums(H1)[4:6]),
                 sum(colSums(H1)[7:9]),
                 sum(colSums(H1)[10:12]),
                 colSums(H1)[13:21])/sum(H1)
```

Calculate the log likelihood of model outputs with given parameters (function continued)

```
LLall<-sum(dpois(x=sim$Hosp_sim,lambda=H,log=T))
# fit to timeseries
# (number of cases follows poisson distribution)
Lmulti<-dmultinom(x=sim$agedist_Sim,
                  prob=agedist_Sim,log = T)
# fit to age distribution
# proportion (sum to 1) follows mutinomial distribution

  #prior
  durprior <- dgamma(x=durx,22,5,log=T)
  # give a prior for the duration of maternal immunity

  #total Loglikelihood (because of log, we sum up)
  LL <- LLall+LLmulti+durprior

  return(LL)
}
```

Use Optim function to maximum a posteriori (calibrate outputs of the transmission model to observed status)

```
fitLL <- optim(par=c(2.2,-1.6,0.1,3.5),
  # remember to exponential these values
  # to get the parameter estimates
  # starting values for beta and gamma
  # - you should get the similar/same result
  # no matter which values you choose here
      fn = fitmodel, # the distance function to optimise
      dat = sim,  # the dataset we fit to
# ("dat" argument is passed to the function specified in fn)
    control = list(fnscale=-1))# negative log likelihood
# here we minimize the negative log likelihood
```

"optimizers in statistical packages usually work by minimizing the result of a function... since the log likelihood and likelihood function have the same increasing or decreasing trend, you can minimize the negative log likelihood in order to actually perform the maximum likelihood estimate of the function you are testing."[7]

[7] https://stats.stackexchange.com/questions/141087/why-do-we-minimize-the-negative-likelihood-if-it-is-equivalent-to-maximization-o

Compare the estimations and real values

```
fitLL # (check if the convergence = 0)
# if not run for several times to make sure
# the function did not stuck at local minimum

exp(fitLL$par)[c(1:2)] # trans prob and Amp est
phi_est = (2*pi*(exp(fitLL$par[3]))) / (1+exp(fitLL$par[3]))
dur_est=exp(fitLL$par)[4]
# output: 8.99  0.20
# 3.33 (same as+/-pi)
# 23.4 days

# Model input:
# 9, 0.2, 3.33, 30 days

# estimations are very similar to model inputs
```

Section 3: Stan model to estimate posterior samples

Before we start, please learn the Rstan basic from the separated tutorials in the same file path.

You will find three code documents inside code_source/stan_code file. age-structureMSIS_deterministic.stan is the stan code contains the ODE function, input data, input parameters, priors and the distribution of posterior distributions. ageMSIS_deterministic_annual.R is the R code that create the interface between stan and R to sample from posterior distribution. msis_wa_D.sh is a file to run the model on cluster. We will start with the stan code.

I suggest you first run the model in local for short iterations for debug purpose. If the stan model works, you can move to HPC by using the msis_wa_D.sh script.

Stan code in details (page 55 to 68)

Function: page 55 to 60.

```
// function is the similar as our simpleModel.R
// except for the syntax.
functions {
  real[] msis (real t, real[] y, real[] theta,
               real[] x_r, int[] x_i) {
// t for time

// state, the volumes in each compartment, y;

// theta, variables used to compute function
// which depend on the model parameters;

// x_r, real variables used to evaluate f
// which only depend on fixed data;

// x_i, integer values used to evaluate f
// which only depend on fixed data.

      int agegroups = x_i[1];
```

You need to specify the variable type and locations of the parameter for each value

Note, remember to add ; after each line of code.

```
real birthrate[agegroups] = x_r[1:agegroups];
real um = x_r[agegroups+1]; // net growth
real rho1 = x_r[agegroups+2];
real rho2 = x_r[agegroups+3];
real gamma1 = x_r[agegroups+4];
real gamma2 = x_r[agegroups+5];
real gamma3 = x_r[agegroups+6];
real sigma1 = x_r[agegroups+7];
real sigma2 = x_r[agegroups+8];
real sigma3 = x_r[agegroups+9];
real u[agegroups]= x_r[(agegroups+10):(9+2*agegroups)];
//aging rate
real c2[agegroups*agegroups]=
      x_r[(10+2*agegroups):(9+2*agegroups+agegroups*agegroups)];
      // contact vector
// (I have not yet figured out how to input matrix)
```

When create a new variable, you also need to specify the length of the variable

```
// parameters to estimate
real beta = theta[1];
real b1 = theta[2];
real phi = theta[3];
real omega = theta[4];

// outcomes of interests that need to be recorded
real dydt[9*agegroups]; // state change
real lambda[agegroups]; // force of infection
real InfectN[agegroups];// number of infectious individuals

// seasonal component
real season_txn = (1+b1*cos(2*pi()*(t-phi*12)/12));
```

Write down ordinary differential equations

```
for (k in 1:agegroups) {
      InfectN[k] = (y[k+2*agegroups]+
      rho1*y[k+4*agegroups]+
      rho2*y[k+6*agegroups]+
      rho2*y[k+8*agegroups]);
      } // (I1+rho1*I2+rho2*I3+rho2*I4)

for (a in 1:agegroups) {
  // force of infection
  lambda[a] = season_txn*beta*gamma1/sum(y)*
  sum(to_vector(c2[((a-1)*agegroups+1):(a*agegroups)])
  .*to_vector(InfectN));

   //correspond to M.
  dydt[a] =  log(birthrate[a]+1)/12*sum(y) -
  (omega+u[a]+um)*y[a];
```

Ordinary differential equations (continued)

corresponding R code: Source function of the transmission dynamic model (continued) except for the aging in process (include in the next page).

```
//S0
  dydt[a+agegroups] =  omega*y[a] -
  lambda[a]*y[a+agegroups] -
  (um+u[a])*y[a+agegroups];

//I1
 dydt[a+2*agegroups] =  lambda[a]*y[a+agegroups] -
 gamma1*y[a+2*agegroups] -
 (um+u[a])*y[a+2*agegroups];

//S1
dydt[a+3*agegroups] =  gamma1*y[a+2*agegroups] -
sigma1*lambda[a]*y[a+3*agegroups] -
(um+u[a])*y[a+3*agegroups];
```

We will skip explaining the rest of the stan code of ordinary differential equations in this tutorial as they are the direct translation of the corresponding R codes.

Aging in process

In age group >= 1 month, they receive the inflows from the younger groups at rate u.

```
if (a >1 ){
dydt[a] = dydt[a]+ u[a-1]*y[a-1];

dydt[a+agegroups] = dydt[a+agegroups]+
u[a-1]*y[a+agegroups-1];

dydt[a+2*agegroups] = dydt[a+2*agegroups] +
u[a-1]*y[a+2*agegroups-1];

//...//

dydt[a+8*agegroups] = dydt[a+8*agegroups]+
u[a-1]*y[a+8*agegroups-1];}

 }
    return dydt;
  }
} // This is the end of the ODE function
```

Data block

Compared to R, the data block in stan code contains both fixed parameter values and also input observed data.

```
// Fixed data is declared in the data block:
data {
  int<lower=1> n_months;
  int<lower=1> agegroups;
  int q;
  real y0[9*agegroups];
  real t0;
  real ts[n_months];
  int N;
  real birthrate[agegroups];
  real um;
//...omit some parameters and data to fit into slides//
  int<lower=1> hosp_cases[n_months];
  int hosp_age[8];
  real hosp1[agegroups];
  real c2[agegroups*agegroups];
}
```

Transformed data block

Link data block to function inputs: real[ ] x_r, int[ ] x_i

```
transformed data {
  real x_r[204];
  int x_i[2];
  x_i[1] = agegroups;
// the first value of x_i is the number of age groups
  x_i[2] = q;
// the second value of x_i is parameter q

  x_r[1:agegroups]= birthrate;
// the first value of x_r is birth rate
  x_r[agegroups+1]=um;
//likewise, omit some parameters and data to fit into slides//
  x_r[agegroups+9]=sigma3;
  x_r[(agegroups+10):(9+2*agegroups)]=u;
  x_r[(10+2*agegroups):(9+2*agegroups+agegroups*agegroups)]=c2;
}
```

Parameter block

Specify the parameters we would like to estimate and their ranges.

```
parameters {
  real<lower=0> beta; // transmission prob
  real<lower=0,upper=1> b1; // Amp
  real<lower=0,upper=2*pi()> phi; // phase
  real<lower=0> omega;
  // waning rate of maternal immunity
  real<lower=0,upper=1> report_ratio;
  // report ratio of RSV hospitalizations
  }
```

Transformed parameters clock

This part of stan codes corresponds to using ode() function in r to get the transmission states of each age group in each time step and then calculate the number of RSV hospitalizations and age distribution.

```
transformed parameters{
real y[n_months, 9*agegroups];
// y is the states matrix that has row length=n_months and colum
  real rel_inf[n_months,agegroups];
// number of infectious individuals
  real lambda[n_months, agegroups]; // force of infection
  // outcomes
  // Number of hospitalizations by age
  matrix[n_months,agegroups] HOSP_AGE;// corresponds to H1
  vector[8] output_hosp_age;
  // age distribution of RSV hosp in children
  vector[n_months] total_children;
  // total hospitalizations in children <5 years
  vector<lower = 0>[n_months] output_hosp;
  //number of RSV hospitalizations by month
```

Transformed parameters clock (continued)

```
real theta[4]; // parameters that need to be estimate
{
  theta[1] = beta;
  theta[2] = b1;
  theta[3] = phi;
  theta[4] = omega;

  y = integrate_ode_bdf(msis, y0, t0, ts, theta, x_r, x_i,
  1.0E-10, 1.0E-10, 1.0E3);
  // relative tolerance for the ODE solver = 1.0E-10
  // absolute tolerance for the ODE solver = 1.0E-10
  // maximum number of steps to take in the ODE solver
  // = 1.0E3
  // these prevent stan takes too long to run
  }
```

Calculate force of infection (continued transformed parameters)

```
for(i in 1:n_months){
  for(j in 1:9*agegroups){
    if (y[i,j] <= 0.0) y[i,j] = 1e-12;}
// this prevent negative values in states

      for (k in 1:agegroups) {
      rel_inf[i,k] = (y[i,k+2*agegroups]+
      rho1*y[i,k+4*agegroups]+
      rho2*y[i,k+6*agegroups]+
      rho2*y[i,k+8*agegroups]);
      }// (I1+rho1*I2+rho2*I3+rho2*I4)

// calculate force of infection; same as inside function
      for (a in 1:agegroups) {
       lambda[i,a] = (1+b1*cos(2*pi()*(i-phi*12)/12))*
       beta*gamma1/sum(y[i,])*
       sum(to_vector(c2[((a-1)*agegroups+1):(a*agegroups)])
       .*to_vector(rel_inf[i,]));
```

Calculate hospitalizations (continued transformed parameters)

```
HOSP_AGE[i,a] = hosp1[a]*lambda[i,a]*y[i,a+agegroups]
              +hosp2[a]*lambda[i,a]*sigma1*y[i,a+3*agegroups]
              +hosp3[a]*lambda[i,a]*sigma2*y[i,a+5*agegroups]
              +hosp3[a]*lambda[i,a]*sigma3*y[i,a+7*agegroups];
        } // same as H1 calculation

  output_hosp[i] = report_ratio*sum(HOSP_AGE[i,]);
  // total reported RSV hospitalizations each month

  total_children[i]=HOSP_AGE[i,1]+HOSP_AGE[i,2]+
  HOSP_AGE[i,3]+HOSP_AGE[i,4]+HOSP_AGE[i,5]+HOSP_AGE[i,6]+
  HOSP_AGE[i,7]+HOSP_AGE[i,8];
  // total hospitalizations in children <5 years
    }
for (b in 1:8) {
 output_hosp_age[b] = sum(HOSP_AGE[,b])/sum(total_children);}}
 // age distribution of hospitalizations in children <5 years
```

Model block

(1) specify the priors of the parameters we would like to estimate and their ranges. / /
(2) specify the sample distributions of the posteriors.

```
model {
  //priors
  beta ~ normal(3,1) T[1,5]; //truncated at 0
  b1 ~ normal(0.2,0.05) T[0.05,1]; //truncated at 0,1
  phi ~ normal(1.4,0.5) T[0,2*pi()];
  omega ~ lognormal(-1,0.6) T[0,5];
  report_ratio ~ beta(2,2); //truncated at 0,1

  //sampling distribution
  for(i in 1:n_months){
  target += poisson_lpmf(hosp_cases[i]| output_hosp[i]);}

  target += multinomial_lpmf (hosp_age|output_hosp_age);
}
```

R function to call stan code

```
library(rstan) # function to link stan code and R code
rstan_options (auto_write = TRUE)
# this code is recommended for running stan in HPC

setwd("~/project/Rstan/WA")
# this is where I save the data and all these codes
data_msis<- readRDS("data_msis_wa_annual.rds")
# read in input data.
# This would come from the inpatient data sets
# and known parameters

print(paste("number of cores:", future::availableCores()))
# this is to check the cores you can use in HPC/local

# call the stan code
model <- stan_model("age-structureMSIS_deterministic.stan")

# set the initial values as the output of MLE function
# This helps the running time
init_fun <- function() list(beta=3.17,b1=0.198,
                            phi=1.46,omega=0.61,hosp_ratio=0.76)
```

R function to sample the posteriors from stan code

```
## this code samples the posteriors of parameters of interests
fit_msis <- sampling(model, # the stan model
            data = data_msis,
          # inpatient data and known parameters
            iter = 2000,# total iterations
                  warmup = 1000, #iterations discard
            chains = 4, # total chains
            seed = 112, # for reproducible purpose
            sample_file = file.path(getwd(),
              'WA_MSIS_12_deterministic.csv'),
          # this is where the posterior samples are saved
            cores=4,#parallel: one core for one chain
            open_progress=T, # to see immediate output
                  init= init_fun, # initial values
            control = list(stepsize=0.1)) # sample step size
```

Shell script (bash codes, Sample SLURM Scripts) to run R and STAN on high performance computer (HPC)

```
# indicate the script uses the bash shell.
#!/bin/bash

#SBATCH --time=30-
# This tells the cluster the running time for your script.
# Don't set it too high or your job will get low priority.
#SBATCH --mail-type=ALL
# This tells the cluster what results should send to your email.
#SBATCH --mail-user=zhe.zheng@yale.edu
# This is your email address
#SBATCH --cpus-per-task=4
# This is for running in parallel.
# Set the cores to the number of chains you have
#SBATCH --mem-per-cpu=30000
# This is for memory for each core

module load R/4.1.0-foss-2020b
# you first need to check the r version available on cluster
Rscript ageMSIS_deterministric_annual.R # R script to run
```