Modeling the Effect of School Closures in a Pandemic Scenario: Exploring Two Different Contact Matrices

Isaac Chun-Hai Fung,* Department of Epidemiology, Jiann-Ping Hsu College of Public Health, Georgia Southern University, Statesboro, Georgia, USA; Centers for Disease Control and Prevention, Atlanta, Georgia, USA;

Manoj Gambhir, Epidemiology Modelling Unit, School of Public Health and Preventive Medicine, Monash University, Melbourne, Australia; Centers for Disease Control and Prevention, Atlanta, Georgia, USA; Department of Infectious Disease Epidemiology, School of Public Health, Imperial College London, London, UK.

John W. Glasser, Centers for Disease Control and Prevention, Atlanta, Georgia, USA

Hongjiang Gao, Centers for Disease Control and Prevention, Atlanta, Georgia, USA

Michael L. Washington, Centers for Disease Control and Prevention, Atlanta, Georgia, USA

Amra Uzicanin, Centers for Disease Control and Prevention, Atlanta, Georgia, USA

Martin I. Meltzer, Centers for Disease Control and Prevention, Atlanta, Georgia, USA

*Corresponding author.

Current address: Dr. Isaac Chun-Hai Fung, PO Box 8015, Jiann-Ping Hsu College of Public Health, Georgia Southern University, Statesboro, GA 30460-8015, USA. Email: cfung@georgiasouthern.edu

R codes

Introduction

As supplementary materials to the article, this PDF file contains four R codes (.r files):

1. "Flu_School_Closure_CID.r": This is the main R code for ordinary differential equation model.
2. "Flu_School_Closure_CID_Data.r": This is the R code that creates the summary result file.
3. "Flu_School_Closure_CID_Plot.r": This is the R code that plots the figures.
4. "Flu_School_Closure_CID_Difference_eqn.r": This is the R code for the difference equation model.

Note: These codes are provided as examples for readers to use in their own studies. These sample codes are by no means the most efficient ways of writing R codes. There are likely to be more than one way to program in R to achieve the same goal and readers may re-write the codes in a more efficient manner (with fewer lines).

Table of Contents

R code 1: “Flu_School_Closure_CID.r”

```
  #####################################################################################
  ## This model is written by Dr. Isaac Chun Hai Fung
  ## With input from Dr. Manoj Gambhir and Dr. John Glasser on the mixing matrix
  ##
  ## The model is based on a previously published model:
  ##     Fung IC-H, Antia R, Handel A (2012)
  ##     How to Minimize the Attack Rate during Multiple Influenza Outbreaks in a
  ##         Heterogeneous Population.
  ##     PLoS ONE 7(6): e36573. doi:10.1371/journal.pone.0036573
  ##
  ## This R code was based on the R code previously written for Fung, Antia and Handel
  ## (2012). The model is now expanded to have 4 age groups instead of 2.
  ##
  ## Final edits were made on November 21, 2014 for the purpose of publication.
  #####################################################################################


rm(list=ls())
# This clears the workspace to make sure no leftover variables are
# floating around. It is not strictly needed but it is often a good idea.

graphics.off();
# Close all graphics windows, in case there are still
# some open from previous work that we did

library(deSolve)
# Load ODE solver package. You need to have the package installed first.

###################################################################################
## First, we specify the function that describes the differential equation model
## for the simulated virus infection.
##
## This function is called by lsoda (the ODE solver) in the main program
###################################################################################

odeequations=function(t,y,parameters)
## The function has to be written in a certain form, dictated by lsoda
{
    S1=y[1]; S2=y[2];  S3=y[3];  S4=y[4];   # Susceptible
    I1=y[5]; I2=y[6];  I3=y[7];  I4=y[8];   # Infected (Infectious)
    R1=y[9]; R2=y[10]; R3=y[11]; R4=y[12];  # Recovered


    ## Model parameters, passed into function by main program
    beta11 = parameters[1];
    beta12 = parameters[2];
    beta13 = parameters[3];
    beta14 = parameters[4];
    beta21 = parameters[5];
    beta22 = parameters[6];
    beta23 = parameters[7];
    beta24 = parameters[8];
    beta31 = parameters[9];
    beta32 = parameters[10];
    beta33 = parameters[11];
    beta34 = parameters[12];
    beta41 = parameters[13];
    beta42 = parameters[14];
    beta43 = parameters[15];
    beta44 = parameters[16];
    gamma1 = parameters[17];
    gamma2 = parameters[18];
    gamma3 = parameters[19];
    gamma4 = parameters[20];



    ## These are the differential equations which describe an S-I-R model
    ## comprising 4 age groups (in proportion) so all add up to 1.
    ## Group 1: Age 0 to 4 years
    ## Group 2: Age 5 to 19 years
    ## Group 3: Age 20 to 64 years
    ## Group 4: Age 65+ years

    N1 <- 0.06440
    N2 <- 0.20204
    N3 <- 0.60074
    N4 <- 0.13282

    dS1dt = - beta11*S1*(I1/N1) - beta12*S1*(I2/N2) - beta13*S1*(I3/N3) - beta14*S1*(I4/N4)
    dS2dt = - beta21*S2*(I1/N1) - beta22*S2*(I2/N2) - beta23*S2*(I3/N3) - beta24*S2*(I4/N4)
    dS3dt = - beta31*S3*(I1/N1) - beta32*S3*(I2/N2) - beta33*S3*(I3/N3) - beta34*S3*(I4/N4)
    dS4dt = - beta41*S4*(I1/N1) - beta42*S4*(I2/N2) - beta43*S4*(I3/N3) - beta44*S4*(I4/N4)

    dI1dt =   beta11*S1*(I1/N1) + beta12*S1*(I2/N2) + beta13*S1*(I3/N3) + beta14*S1*(I4/N4) -
gamma1*I1;
    dI2dt =   beta21*S2*(I1/N1) + beta22*S2*(I2/N2) + beta23*S2*(I3/N3) + beta24*S2*(I4/N4) -
gamma2*I2;
    dI3dt =   beta31*S3*(I1/N1) + beta32*S3*(I2/N2) + beta33*S3*(I3/N3) + beta34*S3*(I4/N4) -
gamma3*I3;
    dI4dt =   beta41*S4*(I1/N1) + beta42*S4*(I2/N2) + beta43*S4*(I3/N3) + beta44*S4*(I4/N4) -
gamma4*I4;

    dR1dt =   gamma1 * I1
    dR2dt =   gamma2 * I2
    dR3dt =   gamma3 * I3
    dR4dt =   gamma4 * I4

    return(list(c(dS1dt,dS2dt,dS3dt,dS4dt,dI1dt,dI2dt,dI3dt,dI4dt,dR1dt,dR2dt,dR3dt,dR4dt)));
    ## This command returns the result, which is the right side of the ODEs as a list,
    ## back to the solver (i.e. lsoda).
} ## End function specifying the ODEs

################################################################################
## Global variable
##
## Assign numerical values to the parameters and initial conditions of the model
##
## Initial conditions
##
## Demographics proportion of the US Population
## Source: US Census Bureau, ACS Demographic and Housing Estimates,
##         2011 American Community Survey 1-Year Estimates:
##
http://factfinder2.census.gov/faces/tableservices/jsf/pages/productview.xhtml?pid=ACS_11_1YR_DP05
&prodType=table
################################################################################

Total_pop <- 310000000 # Approximation of total US population
  S1 <- 0.06440  # Proportion of Group 1 in the population   (0-4)
  S2 <- 0.20204  # Proportion of Group 2 in the population   (5-19)
  S3 <- 0.60074  # Proportion of Group 3 in the population   (20-64)
  S4 <- 0.13282  # Proportion of Group 4 in the population   (64+)

  ## Number of infected persons introduced into the USA
  seed = 10
  ## 10 infected person are introduced into the USA (310 million people)

  infection_introduction = seed/Total_pop

  ## Initial proportion of susceptible, infected and recovered populations
  ## by Group (1, 2, 3, and 4)
  S1_entry <- S1
  S2_entry <- S2
  S3_entry <- S3
  S4_entry <- S4
  I1_entry <- 0
  I2_entry <- 0
  I3_entry <- infection_introduction  ## Assumption the infected person ("seed")
                                      ## is a working adult (Group 3)
  I4_entry <- 0
  R1_entry <- 0
  R2_entry <- 0
  R3_entry <- 0
  R4_entry <- 0

 ## Combine initial conditions into a vector
  Y0=c(S1_entry, S2_entry, S3_entry, S4_entry, I1_entry, I2_entry, I3_entry, I4_entry, R1_entry,
R2_entry, R3_entry, R4_entry);


  #############################################################################################
  ## Recovery rate
  ## Values for model parameters, units are assumed to be 1/days
  ## Recovery rate = 0.25; i.e. length of infectiousness = 4 days
  gamma_matrix <- c(0.25,0.25,0.25,0.25)
  gamma1 <- 0.25;
  gamma2 <- 0.25;
  gamma3 <- 0.25;
  gamma4 <- 0.25;

  ############################################################
  timevec_interval <- 1 ## "lsoda" output results every day (unit = day)
  ############################################################

  ######################################
  ## Total length of simulation
  total_length_simulation <- 365

  ######################################
  ## The time when the infection was introduced into the USA
  ## (Number of days since the beginning of the simulation)
  time_of_introduction <- 14

  ######################################
  ## The beginning of the summer break or school closure
  ## We assume that summer vacation (or school closure) begins 5 days
  ## after the introduction of the infection
  from_intro_to_summer_break <-5
  summer_break_start <- time_of_introduction + from_intro_to_summer_break

  ###################################################
  ## Transmission Matrix
  ###################################################

  ## Multiplier to adjust the attack rate and therefore R0
  ## Epidemiologically, it reflects the overall transmission probability of
  ## the virus per person-to-person contact
  ## Assuming 50% of the cases are asymptomatic
  ## If clinical attack rate ~ 30%, we have "real" AR ~ 60%
  ## If clinical attack rate ~ 15%, we have "real" AR ~ 30%

  for (ARScenario in 1:2){
  if (ARScenario == 1){ #15% Clinical AR
    multiplier <- 0.011047  # CInf == 0.1500055
        CARlabel <- c("CAR15")
    }
  if (ARScenario == 2){ #30% Clinical AR
    multiplier <- 0.016487  # CInf == 0.3000068
        CARlabel <- c("CAR30")
    }
      ## Term time conversational contact matrix obtained from
      ## Eames KTD et al. (2012) Measured Dynamic Social Contact
      ## Patterns Explain the Spread of H1N1v Influenza.
      School_matrix <- matrix(nrow = 4, ncol = 4)
        School_matrix[1,] <- c(4.0196, 1.8137, 7.8039, 0.1373)
        School_matrix[2,] <- c(1.4139, 27.6762, 11.1639, 0.9795)
        School_matrix[3,] <- c(0.8472, 3.8457, 14.7942, 1.0078)
        School_matrix[4,] <- c(0.1048, 0.2857, 6.5673, 2.0980)
      Vacation_matrix <- matrix(nrow = 4, ncol = 4)
        Vacation_matrix[1,] <- c(6.5227, 1.7500, 7.0227, 0.0909)
        Vacation_matrix[2,] <- c(0.9783, 11.5761, 11.7174, 0.5761)
        Vacation_matrix[3,] <- c(1.3087, 2.2781, 14.9680, 1.0525)
        Vacation_matrix[4,] <- c(0.1442, 0.4279, 5.6512, 1.4326)

      n_people <- matrix(nrow = 1, ncol = 4)

B_School_matrix <- matrix(nrow = 4, ncol = 4)
B_Vacation_matrix <- matrix(nrow = 4, ncol = 4)
Contact_matrix <- matrix(nrow = 4, ncol = 4)
Vacation_Contact_matrix <- matrix(nrow = 4, ncol = 4)

experiment <- 3

if (experiment == 1){

      ## Method provided by Dr. John Glasser (part 1) and adapted
      ## from Ken Eames' paper.
      ## Dr. John Glasser's method was applied to convert it to be symmetrical
      Contact_matrix <- sqrt(School_matrix*t(School_matrix))

      # Contact matrix is the square root of A %*% t(A);
      # t(A) is the transpose of A
      # A, i.e. Polymod matrix
      #             [,1]       [,2]       [,3]       [,4]
      #  [1,] 4.0196000  1.601371  2.571277 0.1199543
      #  [2,] 1.6013714 27.676200  6.552329 0.5290020
      #  [3,] 2.5712767  6.552329 14.794200 2.5726494
      #  [4,] 0.1199543  0.529002  2.572649 2.0980000

      #> Row_sum  # Row sums
      #            [,1]
      #  [1,]  8.312202
      #  [2,] 36.358902
      #  [3,] 26.490455
      #  [4,]  5.319606

      #> Mixing_matrix
      #            [,1]        [,2]       [,3]        [,4]
      #  [1,] 0.48357821 0.19265308 0.3093376 0.01443111
      #  [2,] 0.04404345 0.76119460 0.1802125 0.01454945
      #  [3,] 0.09706427 0.24734678 0.5584729 0.09711609
      #  [4,] 0.02254948 0.09944384 0.4836166 0.39439013

      #> beta_matrix
      #             [,1]         [,2]        [,3]         [,4]
      # [1,] 0.066271145 0.026401811 0.04239264 0.001977687
      # [2,] 0.026401811 0.456297509 0.10802824 0.008721657
      # [3,] 0.042392640 0.108028242 0.24391198 0.042415271
      # [4,] 0.001977687 0.008721657 0.04241527 0.034589726

      Vacation_Contact_matrix<-sqrt(Vacation_matrix*t(Vacation_matrix))

      #  > Vacation_Contact_matrix
      #          [,1]        [,2]       [,3]       [,4]
      #[1,] 6.5227000  1.3084437  3.031601 0.1144892
      #[2,] 1.3084437 11.5761000  5.166566 0.4965009
      #[3,] 3.0316015  5.1665665 14.968000 2.4388292
      #[4,] 0.1144892  0.4965009  2.438829 1.4326000

      #  > Vacation_Row_sum
      #          [,1]
      #[1,] 10.977234
      #[2,] 18.547611
      #[3,] 25.604997
      #[4,]  4.482419

      #  > Vacation_Mixing_matrix
      #            [,1]       [,2]       [,3]        [,4]
      #[1,] 0.59420249 0.1191961 0.2761717 0.01042970
      #[2,] 0.07054514 0.6241289 0.2785570 0.02676900
      #[3,] 0.11839882 0.2017796 0.5845734 0.09524817
      #[4,] 0.02554183 0.1107663 0.5440877 0.31960419

      # > Vacation_beta_matrix
      #            [,1]        [,2]       [,3]        [,4]
      #[1,] 0.107539755 0.021572312 0.04998201 0.001887584
      #[2,] 0.021572312 0.190855161 0.08518118 0.008185811
      #[3,] 0.049982013 0.085181181 0.24677742 0.040208977
      #[4,] 0.001887584 0.008185811 0.04020898 0.023619276
}

if (experiment == 3){  # Alternative matrix

      n_people[1,] <- c(20067828, 62953784, 187185281, 41385026) # Million. US population.
      for (i in 1:4){
        for (j in 1:4){
           B_School_matrix[i,j] <- n_people[1,i] * School_matrix[i,j] + n_people[1,j] *
School_matrix[j,i]
        }
      }

      for (i in 1:4){
        for (j in 1:4){
           B_Vacation_matrix[i,j] <- n_people[1,i] * Vacation_matrix[i,j] + n_people[1,j] *
Vacation_matrix[j,i]
        }
      }
       for (i in 1:4){
        for (j in 1:4){
           Contact_matrix[i,j] <- B_School_matrix[i,j] / (2*n_people[1,i])
        }
      }
      for (i in 1:4){
        for (j in 1:4){
           Vacation_Contact_matrix[i,j] <- B_Vacation_matrix[i,j] / (2*n_people[1,i])
        }
      }

    #> Contact_matrix
    #           [,1]       [,2]      [,3]      [,4]
    #[1,] 4.01960000  3.1245876  7.853134 0.1767123
    #[2,] 0.99602730 27.6762000 11.299306 0.5836578
    #[3,] 0.84192168  3.8001604 14.794200 1.2298863
    #[4,] 0.08568876  0.8878445  5.562800 2.0980000
    #> Vacation_Contact_matrix
    #           [,1]       [,2]      [,3]      [,4]
    #[1,] 6.52270000  2.4094881  9.614885 0.1941388
    #[2,] 0.76807445 11.5761000  9.245524 0.4286980
    #[3,] 1.03079609  3.1094364 14.968000 1.1509654
    #[4,] 0.09413896  0.6521239  5.205839 1.4326000
    #> beta_matrix
    #            [,1]       [,2]       [,3]        [,4]
    #[1,] 0.066271145 0.05151508 0.12947462 0.002913455
    #[2,] 0.016421502 0.45629751 0.18629166 0.009622766
    #[3,] 0.013880763 0.06265324 0.24391198 0.020277135
    #[4,] 0.001412751 0.01463789 0.09171388 0.034589726
    #> Vacation_beta_matrix
    #            [,1]       [,2]       [,3]        [,4]
    #[1,] 0.107539755 0.03972523 0.15852061 0.003200766
    #[2,] 0.012663243 0.19085516 0.15243096 0.007067945
    #[3,] 0.016994735 0.05126528 0.24677742 0.018975967
    #[4,] 0.001552069 0.01075157 0.08582867 0.023619276
}


    ##########################
    ## School open - matrix ##
    ##########################

      Row_sum <- matrix(nrow=4, ncol=1)
      for(i in 1:4){
```

```
    Row_sum[i,1]=sum(Contact_matrix[i,1:4])
   }


   Mixing_matrix <- matrix(nrow=4, ncol=4)
   for (i in 1:4){
    Mixing_matrix[i,1:4]<- Contact_matrix[i,1:4]/Row_sum[i,1]
   }

   beta_matrix <- matrix(nrow=4, ncol=4)
   for (i in 1:4){
    for (j in 1:4){
     beta_matrix[i,j] <- Row_sum[i,1] * Mixing_matrix[i,j] * multiplier
    }
   }

  print(beta_matrix)

  ##############################################
  ## Vacation matrix = school closure matrix ##
  ##############################################

    Vacation_Row_sum <- matrix(nrow=4, ncol=1)
   for(i in 1:4){
    Vacation_Row_sum[i,1]=sum(Vacation_Contact_matrix[i,1:4])
   }

    Vacation_Mixing_matrix <- matrix(nrow=4, ncol=4)
   for (i in 1:4){
    Vacation_Mixing_matrix[i,1:4]<- Vacation_Contact_matrix[i,1:4]/Vacation_Row_sum[i,1]
   }

    Vacation_beta_matrix <- matrix(nrow=4, ncol=4)
   for (i in 1:4){
    for (j in 1:4){
     Vacation_beta_matrix[i,j] <- Vacation_Row_sum[i,1] * Vacation_Mixing_matrix[i,j] *
multiplier
    }
   }


 #################################
 ## Scenarios of School closure ##
 #################################
for (school_scenario in 1:8){

 ## if baseline, beta_matrix2 == beta_matrix
 if (school_scenario == 1){
  beta_matrix2 <- beta_matrix
  Label <- c("Baseline")
  intervention_length <- 28; # But actually school remains open these 28 d
 }

 ## if we close school, beta_matrix2 == Vacation_beta_matrix
 if (school_scenario > 1){
  beta_matrix2 <- Vacation_beta_matrix
  Label <- c("Closure")
 }

 if (school_scenario == 2) {
   intervention_length <- 7;
   # 7 days; Duration of school closure (1 week)
  }
 if (school_scenario == 3) {
   intervention_length <- 14;
   # 14 days; Duration of school closure (2 weeks)
  }
 if (school_scenario == 4) {
   intervention_length <- 21;
   # 21 days; Duration of school closure (3 weeks)
  }
  if (school_scenario == 5) {
      intervention_length <- 28;
      # 28 days; Duration of school closure (4 weeks)
    }
  if (school_scenario == 6) {
      intervention_length <- 56;
      # 56 days; Duration of school closure (8 weeks)
    }
  if (school_scenario == 7) {
      intervention_length <- 84;
      # 84 days; Duration of school closure (12 weeks)
    }
  if (school_scenario == 8) {
      intervention_length <- 140;
      # 140 days; Duration of school closure (20 weeks)
    }

  intervention_length_reset <- intervention_length
  end_of_summer_break = summer_break_start + intervention_length
  #reset the time at which intervention ceases


####################################################################
## Create a time series before the introduction of imported cases ##
####################################################################
  output <- matrix(0, nrow= (time_of_introduction + 1), ncol=13,dimnames =
list(c(0:time_of_introduction),c("time", "1", "2","3","4","5","6","7","8","9","10","11","12")))
  for (n in 1 : (time_of_introduction+1)) {
    output[n, ] <-c((n-1), S1, S2, S3, S4, 0,0,0,0,0,0,0,0)
  }


    ScenLabel <- paste("P_",Label,sep="")

#########################################################
## Introduction of imported cases. Before summer break ##
#########################################################
  timevec=seq(time_of_introduction,summer_break_start,by=1);
  #this creates a vector of times for which integration is evaluated

  parvec=c(beta_matrix[1,1], beta_matrix[1,2], beta_matrix[1,3], beta_matrix[1,4],
beta_matrix[2,1], beta_matrix[2,2], beta_matrix[2,3], beta_matrix[2,4], beta_matrix[3,1],
beta_matrix[3,2], beta_matrix[3,3], beta_matrix[3,4], beta_matrix[4,1], beta_matrix[4,2],
beta_matrix[4,3], beta_matrix[4,4], gamma1, gamma2, gamma3, gamma4);
  # This combines all parameters into a vector called parvec
  # which is sent to the ODE function


  odeoutput=ode(y=Y0, times=timevec, func=odeequations, parms=parvec);
  pre_intervention = odeoutput[length(odeoutput[,1]),1]

  # Variable value at the beginning of summer break (or school closure)
  S1_break <- odeoutput[length(odeoutput[,1]),2];
  S2_break <- odeoutput[length(odeoutput[,1]),3];
  S3_break <- odeoutput[length(odeoutput[,1]),4];
  S4_break <- odeoutput[length(odeoutput[,1]),5];
  I1_break <- odeoutput[length(odeoutput[,1]),6];
  I2_break <- odeoutput[length(odeoutput[,1]),7];
  I3_break <- odeoutput[length(odeoutput[,1]),8];
  I4_break <- odeoutput[length(odeoutput[,1]),9];
  R1_break <- odeoutput[length(odeoutput[,1]),10];
  R2_break <- odeoutput[length(odeoutput[,1]),11];
  R3_break <- odeoutput[length(odeoutput[,1]),12];
  R4_break <- odeoutput[length(odeoutput[,1]),13];

  ## CInf stands for cumulative number of infections,
  ## i.e. cumulative attack rates, at the beginning of summer break
  CInf1_break <- S1_entry - S1_break
  CInf2_break <- S2_entry - S2_break
  CInf3_break <- S3_entry - S3_break
  CInf4_break <- S4_entry - S4_break
```

```
#############################################################
## Summer break (School closure)

  Y1=c(S1_break, S2_break, S3_break, S4_break, I1_break, I2_break, I3_break, I4_break, R1_break,
R2_break, R3_break, R4_break);
  intervention_length = intervention_length_reset

  timevec1=seq( pre_intervention, end_of_summer_break, by=timevec_interval);
  ## This creates a vector of times for which integration is evaluated

  parvec1=c(beta_matrix2[1,1], beta_matrix2[1,2], beta_matrix2[1,3], beta_matrix2[1,4],
beta_matrix2[2,1], beta_matrix2[2,2], beta_matrix2[2,3], beta_matrix2[2,4], beta_matrix2[3,1],
beta_matrix2[3,2], beta_matrix2[3,3], beta_matrix2[3,4], beta_matrix2[4,1], beta_matrix2[4,2],
beta_matrix2[4,3], beta_matrix2[4,4], gamma1, gamma2, gamma3, gamma4);
  ## This combines all parameters into a vector called parvec which is sent to the ODE function


  odeoutput1=ode(y=Y1, times=timevec1, func=odeequations, parms=parvec1);

  intervention_end = odeoutput1[length(odeoutput1[,1]),1];

  S1_break_end <- odeoutput1[length(odeoutput1[,1]),2];
  S2_break_end <- odeoutput1[length(odeoutput1[,1]),3];
  S3_break_end <- odeoutput1[length(odeoutput1[,1]),4];
  S4_break_end <- odeoutput1[length(odeoutput1[,1]),5];
  I1_break_end <- odeoutput1[length(odeoutput1[,1]),6];
  I2_break_end <- odeoutput1[length(odeoutput1[,1]),7];
  I3_break_end <- odeoutput1[length(odeoutput1[,1]),8];
  I4_break_end <- odeoutput1[length(odeoutput1[,1]),9];
  R1_break_end <- odeoutput1[length(odeoutput1[,1]),10];
  R2_break_end <- odeoutput1[length(odeoutput1[,1]),11];
  R3_break_end <- odeoutput1[length(odeoutput1[,1]),12];
  R4_break_end <- odeoutput1[length(odeoutput1[,1]),13];

  ## CInf stands for cumulative number of infections, i.e. cumulative attack rates, at the
beginning of summer break
  CInf1_break_end <- S1_entry - S1_break_end
  CInf2_break_end <- S2_entry - S2_break_end
  CInf3_break_end <- S3_entry - S3_break_end
  CInf4_break_end <- S4_entry - S4_break_end

##################################################
## School starts in Fall (when schools re-open) ##
##################################################

  Y2=c(S1_break_end, S2_break_end, S3_break_end, S4_break_end, I1_break_end, I2_break_end,
I3_break_end, I4_break_end, R1_break_end, R2_break_end, R3_break_end, R4_break_end);

  timevec2=seq(end_of_summer_break, total_length_simulation, by=timevec_interval);
  # This creates a vector of times for which integration is evaluated

  parvec2=c(beta_matrix[1,1], beta_matrix[1,2], beta_matrix[1,3], beta_matrix[1,4],
beta_matrix[2,1], beta_matrix[2,2], beta_matrix[2,3], beta_matrix[2,4], beta_matrix[3,1],
beta_matrix[3,2], beta_matrix[3,3], beta_matrix[3,4], beta_matrix[4,1], beta_matrix[4,2],
beta_matrix[4,3], beta_matrix[4,4], gamma1, gamma2, gamma3, gamma4);
  # This combines all parameters into a vector called parvec which is sent to the ODE function

  odeoutput2=ode(y=Y2, times=timevec2, func=odeequations, parms=parvec2);

  intervention_end = odeoutput2[length(odeoutput2[,1]),1];

  S1_end <- odeoutput2[length(odeoutput2[,1]),2];
  S2_end <- odeoutput2[length(odeoutput2[,1]),3];
  S3_end <- odeoutput2[length(odeoutput2[,1]),4];
  S4_end <- odeoutput2[length(odeoutput2[,1]),5];
  I1_end <- odeoutput2[length(odeoutput2[,1]),6];
  I2_end <- odeoutput2[length(odeoutput2[,1]),7];
  I3_end <- odeoutput2[length(odeoutput2[,1]),8];
  I4_end <- odeoutput2[length(odeoutput2[,1]),9];
  R1_end <- odeoutput2[length(odeoutput2[,1]),10];
  R2_end <- odeoutput2[length(odeoutput2[,1]),11];
  R3_end <- odeoutput2[length(odeoutput2[,1]),12];
  R4_end <- odeoutput2[length(odeoutput2[,1]),13];

  outputall <- matrix(0,nrow=366,ncol=13)
  outputall[1:(time_of_introduction+1),] <- output[1:length(output[,1]),]
  outputall[(time_of_introduction+1):(summer_break_start+1),] <-
odeoutput[1:length(odeoutput[,1]),]
  outputall[(summer_break_start+1):(end_of_summer_break+1),] <-
odeoutput1[1:length(odeoutput1[,1]),]
  outputall[(end_of_summer_break+1):366,] <- odeoutput2[1:length(odeoutput2[,1]),]
  ## Convert to numbers
  newoutputall <- outputall
  newoutputall[,2:13] <- Total_pop*outputall[,2:13]

############################################################################
## Assumption of symptomatic to asymptomatic ratio
## If we want to count all cases (symptomatic and asymptomatic), or
## if we assume that all cases are symptomatic, then sym_ratio = 1
## If we assume that only 1 in 2 cases are symptomatic, then sym_ratio = 0.5
## If we assume that only 1 in 3 cases are symptomatic, then sym_ratio = 0.33
sym_ratio <- 0.5

  countall            <- matrix(0,nrow=366,ncol=7)
  countall[,1]        <- newoutputall[,1]
  countall[,2:5]      <- newoutputall[,6:9]*sym_ratio
    # Divided by sym_ratio
  for (n in 1:366) {
    countall[n,6]  <- (sum(newoutputall[n,6:9]))*sym_ratio
    # Total incidence on a given day
    # Divided by sym_ratio
    countall[n,7]  <- (Total_pop - sum(newoutputall[n,2:5]))*sym_ratio
    # Count cumulative incidence
    # i.e. Total population - Total Susceptible population
    # Divided by sym_ratio
  }


    filename_countall <- paste(experiment, CARlabel, ScenLabel, seed, intervention_length,
"incidence.csv", sep="_")
    write.csv(countall, filename_countall)

  ## CInf stands for cumulative number of infections, i.e. cumulative attack rates
  ## Divided by 2 (sym_ratio = 0.5): Assumption: 1 in 2 infected persons is asymptomatic

  CInf1 <- (S1_entry - S1_end)* sym_ratio
  CInf2 <- (S2_entry - S2_end)* sym_ratio
  CInf3 <- (S3_entry - S3_end)* sym_ratio
  CInf4 <- (S4_entry - S4_end)* sym_ratio
  CInf <- CInf1 + CInf2 + CInf3 + CInf4

  ## Write data file
  Data <- matrix(0, nrow=3, ncol=1)
  peak <- which.max(countall[,6])
  print(sprintf("Peak: Day %s",countall[peak,1]))
  Data[1,1] <- countall[peak,1]
  print(sprintf("Daily number of new cases at peak time: %s",countall[peak,6]))
  Data[2,1] <- countall[peak,6]
  Data[3,1] <- CInf
  print(CInf)
  filename_csv <- paste(experiment,CARlabel,ScenLabel,seed,intervention_length,"data.csv",sep="_")
  write.csv(Data, filename_csv)

}
}
```

R code 2: “Flu_School_Closure_CID_Data.r”

```r
## This R code creates the data summary files. This is optional for our modeling purpose.

experiment <- 3 ## Type 1 for Main analysis; 3 for Alternative matrix

filename_M1 <- paste(experiment,"_CAR30_P_Baseline_10_28_data.csv", sep = "")
filename_M2 <- paste(experiment,"_CAR30_P_Closure_10_7_data.csv", sep = "")
filename_M3 <- paste(experiment,"_CAR30_P_Closure_10_14_data.csv", sep = "")
filename_M4 <- paste(experiment,"_CAR30_P_Closure_10_21_data.csv", sep = "")
filename_M5 <- paste(experiment,"_CAR30_P_Closure_10_28_data.csv", sep = "")
filename_M6 <- paste(experiment,"_CAR30_P_Closure_10_56_data.csv", sep = "")
filename_M7 <- paste(experiment,"_CAR30_P_Closure_10_84_data.csv", sep = "")
filename_M8 <- paste(experiment,"_CAR30_P_Closure_10_140_data.csv", sep = "")

filename_M9 <- paste(experiment,"_CAR15_P_Baseline_10_28_data.csv", sep = "")
filename_M10 <- paste(experiment,"_CAR15_P_Closure_10_7_data.csv", sep = "")
filename_M11 <- paste(experiment,"_CAR15_P_Closure_10_14_data.csv", sep = "")
filename_M12 <- paste(experiment,"_CAR15_P_Closure_10_21_data.csv", sep = "")
filename_M13 <- paste(experiment,"_CAR15_P_Closure_10_28_data.csv", sep = "")
filename_M14 <- paste(experiment,"_CAR15_P_Closure_10_56_data.csv", sep = "")
filename_M15 <- paste(experiment,"_CAR15_P_Closure_10_84_data.csv", sep = "")
filename_M16 <- paste(experiment,"_CAR15_P_Closure_10_140_data.csv", sep = "")

M1 <- read.csv(filename_M1)
M2 <- read.csv(filename_M2)
M3 <- read.csv(filename_M3)
M4 <- read.csv(filename_M4)
M5 <- read.csv(filename_M5)
M6 <- read.csv(filename_M6)
M7 <- read.csv(filename_M7)
M8 <- read.csv(filename_M8)

M9 <- read.csv(filename_M9)
M10 <- read.csv(filename_M10)
M11 <- read.csv(filename_M11)
M12 <- read.csv(filename_M12)
M13 <- read.csv(filename_M13)
M14 <- read.csv(filename_M14)
M15 <- read.csv(filename_M15)
M16 <- read.csv(filename_M16)

DataMatrix <- matrix(nrow=length(M8[,1]), ncol=16)
DataMatrix[,1] <- M1[,2]
DataMatrix[,2] <- M2[,2]
DataMatrix[,3] <- M3[,2]
DataMatrix[,4] <- M4[,2]
DataMatrix[,5] <- M5[,2]
DataMatrix[,6] <- M6[,2]
DataMatrix[,7] <- M7[,2]
DataMatrix[,8] <- M8[,2]
DataMatrix[,9] <- M9[,2]
DataMatrix[,10] <- M10[,2]
DataMatrix[,11] <- M11[,2]
DataMatrix[,12] <- M12[,2]
DataMatrix[,13] <- M13[,2]
DataMatrix[,14] <- M14[,2]
DataMatrix[,15] <- M15[,2]
DataMatrix[,16] <- M16[,2]

Report <- matrix(nrow=3, ncol=16)
Report[1,] <- DataMatrix[1,]
Report[2,] <- DataMatrix[2,]
Report[3,] <- DataMatrix[3,]

# Write datafile
filename_csv <- paste(experiment,"CAR30_report_data.csv",sep="_")
write.csv(Report[,1:8], filename_csv)
filename_csv2 <- paste(experiment,"CAR15_report_data.csv",sep="_")
write.csv(Report[,9:16], filename_csv2)
```

R code 3: “Flu_School_Closure_CID_Plot.r”

```
## This R code was used to create the figures in the article.

AR <- c("_CAR15")
## Type "_CAR15" for cumulative attack rate 15% scenario
## Type "_CAR30" for cumulative attack rate 30% scenario

## Read the data files
filename_M1 <- paste(experiment,AR,"_P_Baseline_10_28_incidence.csv", sep = "")
filename_M2 <- paste(experiment,AR,"_P_Closure_10_7_incidence.csv", sep = "")
filename_M3 <- paste(experiment,AR,"_P_Closure_10_14_incidence.csv", sep = "")
filename_M4 <- paste(experiment,AR,"_P_Closure_10_21_incidence.csv", sep = "")
filename_M5 <- paste(experiment,AR,"_P_Closure_10_28_incidence.csv", sep = "")
filename_M6 <- paste(experiment,AR,"_P_Closure_10_56_incidence.csv", sep = "")
filename_M7 <- paste(experiment,AR,"_P_Closure_10_84_incidence.csv", sep = "")
filename_M8 <- paste(experiment,AR,"_P_Closure_10_140_incidence.csv", sep = "")
M1 <- read.csv(filename_M1)
M2 <- read.csv(filename_M2)
M3 <- read.csv(filename_M3)
M4 <- read.csv(filename_M4)
M5 <- read.csv(filename_M5)
M6 <- read.csv(filename_M6)
M7 <- read.csv(filename_M7)
M8 <- read.csv(filename_M8)

## Plot the figures
    plot(M1[,2],M1[ ,7],type="l",xlab="Time (days)",ylab="Cases
(Million)",col="black",lwd=3,lty=1,xlim=c(0,365),ylim=c(0,25000000), yaxt='n')
        ## Suppress the default y-axis with yaxt='n'
    lines(M2[,2],M2[ ,7],type="l",col="green",lwd=3,lty=2)
    lines(M3[,2],M3[ ,7],type="l",col="light blue",lwd=3,lty=3)
    lines(M4[,2],M4[ ,7],type="l",col="gold",lwd=3,lty=1)
    lines(M5[,2],M5[ ,7],type="l",col="red",lwd=3,lty=2)
    lines(M6[,2],M6[ ,7],type="l",col="blue",lwd=3,lty=3)
    lines(M7[,2],M7[ ,7],type="l",col="magenta",lwd=3,lty=4)
    lines(M8[,2],M8[ ,7],type="l",col="grey",lwd=3,lty=5)

## Add the tailor-made y-axis
if (experiment == 1){ axis(2, at = c(0,4000000,10000000,16000000), labels=c("0","4","10","16"))}
if (experiment == 3){ axis(2, at = c(0,3000000,10000000,14000000), labels=c("0","3","10","14"))}

## Add arrows
    arrows(summer_break_start,11000000,(summer_break_start+7),11000000,code=3,col=c("green"))
    arrows(summer_break_start,10000000,(summer_break_start+14),10000000,code=3,col=c("light
blue"))
    arrows(summer_break_start,9000000,(summer_break_start+21),9000000,code=3,col=c("gold"))
    arrows(summer_break_start,8000000,(summer_break_start+28),8000000,code=3,col=c("red"))
    arrows(summer_break_start,7000000,(summer_break_start+56),7000000,code=3,col=c("blue"))
    arrows(summer_break_start,6000000,(summer_break_start+84),6000000,code=3,col=c("magenta"))
    arrows(summer_break_start,5000000,(summer_break_start+140),5000000,code=3,col=c("grey"))
    arrows(time_of_introduction,5000000,time_of_introduction,0,code=2,col=c("purple"))

## Add legend
    legend(0,25000000,c("Baseline","Closure 7d", "Closure 14d","Closure 21d","Closure
28d","Closure 56d","Closure 84d","Closure 140d"), col = c("black","green","light
blue","gold","red","blue","magenta","grey"), lty=c(1,2,3,1,2,3,4,5), lwd = 3, ncol=3)
```

R code 4: “Flu_School_Closure_CID_Difference_eqn.r”

```r
###############################################################################
## Difference Equation SIR Model for School Closure in an Influenza Pandemic ##
###############################################################################
## Corresponding to the Ordinary Differential Equation SIR Model             ##
## for School Closure in an Influenza Pandemic                               ##
###############################################################################
## Written by Isaac Chun-Hai FUNG, PhD                                       ##
## June 11, 2013; revised on Sep 12, 2013;                                   ##
## edited for publication on Nov 21, 2014                                    ##
###############################################################################

###############
## Time step ##
###############
for (experiment in 1:4){

# Time step size for difference equation model
if (experiment == 1) {Timestep_size <- 1}
if (experiment == 2) {Timestep_size <- 0.1}
if (experiment == 3) {Timestep_size <- 0.01}
if (experiment == 4) {Timestep_size <- 0.001}

# Duration (days) of simulation
Simulation_length <- 365

Total_timesteps <-  Simulation_length / Timestep_size

###############################
## School closure time frame ##
###############################

school_close <- 5 / Timestep_size
school_closure_length <- 28 / Timestep_size
school_reopen <- school_close + school_closure_length
print(school_close)
print(school_reopen)

################################################################################
## Assumption of symptomatic to asymptomatic ratio                            ##
## If we want to count all cases (symptomatic and asymptomatic), or           ##
## if we assume that all cases are symptomatic, then sym_ratio = 1            ##
## If we assume that only 1 in 2 cases are symptomatic, then sym_ratio = 0.5  ##
## If we assume that only 1 in 3 cases are symptomatic, then sym_ratio = 0.33 ##
################################################################################
sym_ratio <- 0.5

###########################
## Transmission matrices ##
###########################
    multiplier <- 0.016487  # CInf == 0.2999887
    CARlabel <- c("CAR30")

## Rate of Recovery
  gamma_matrix <- c(0.25,0.25,0.25,0.25)

    ##########################
    ## School open - matrix ##
    ##########################
      ## School matrix (term time) obtained from Eames KTD et al. (2012) Measured
      ## Dynamic Social Contact Patterns Explain the Spread of H1N1v Influenza.
      School_matrix <- matrix(nrow = 4, ncol = 4)
        School_matrix[1,] <- c(4.0196, 1.8137, 7.8039, 0.1373)
        School_matrix[2,] <- c(1.4139, 27.6762, 11.1639, 0.9795)
        School_matrix[3,] <- c(0.8472, 3.8457, 14.7942, 1.0078)
        School_matrix[4,] <- c(0.1048, 0.2857, 6.5673, 2.0980)

      Contact_matrix <- matrix(nrow = 4, ncol = 4)
      Contact_matrix <- sqrt(School_matrix*t(School_matrix))
      # Contact matrix is the square root of A %*% t(A); t(A) is the transpose of A
      # A, i.e. Polymod matrix
      #             [,1]      [,2]      [,3]      [,4]
      #  [1,] 4.0196000  1.601371  2.571277 0.1199543
      #  [2,] 1.6013714 27.676200  6.552329 0.5290020
      #  [3,] 2.5712767  6.552329 14.794200 2.5726494
      #  [4,] 0.1199543  0.529002  2.572649 2.0980000

      Row_sum <- matrix(nrow=4, ncol=1)
      for(i in 1:4){
        Row_sum[i,1]=sum(Contact_matrix[i,1:4])
      }
      #> Row_sum  # Row sums
      #            [,1]
      #  [1,]  8.312202
      #  [2,] 36.358902
      #  [3,] 26.490455
      #  [4,]  5.319606

      Mixing_matrix <- matrix(nrow=4, ncol=4)
      for (i in 1:4){
        Mixing_matrix[i,1:4]<- Contact_matrix[i,1:4]/Row_sum[i,1]
      }
      #>    Mixing_matrix
      #            [,1]       [,2]      [,3]       [,4]
      #  [1,] 0.48357821 0.19265308 0.3093376 0.01443111
      #  [2,] 0.04404345 0.76119460 0.1802125 0.01454945
      #  [3,] 0.09706427 0.24734678 0.5584729 0.09711609
      #  [4,] 0.02254948 0.09944384 0.4836166 0.39439013
      beta_matrix <- matrix(nrow=4, ncol=4)
      for (i in 1:4){
        for (j in 1:4){
          beta_matrix[i,j] <- Row_sum[i,1] * Mixing_matrix[i,j] * multiplier
        }
      }
      R0_matrix <- matrix(nrow=1, ncol=4)
      for (j in 1:4) {
        R0_matrix[,j] <- sum(beta_matrix[,j]) / gamma_matrix[j]
      }

    #############################################
    ## Vacation matrix = school closure matrix ##
    #############################################
      Vacation_matrix <- matrix(nrow = 4, ncol = 4)
        Vacation_matrix[1,] <- c(6.5227, 1.7500, 7.0227, 0.0909)
        Vacation_matrix[2,] <- c(0.9783, 11.5761, 11.7174, 0.5761)
        Vacation_matrix[3,] <- c(1.3087, 2.2781, 14.9680, 1.0525)
        Vacation_matrix[4,] <- c(0.1442, 0.4279, 5.6512, 1.4326)

      Vacation_Contact_matrix <- matrix(nrow = 4, ncol = 4)
      Vacation_Contact_matrix <- sqrt(Vacation_matrix*t(Vacation_matrix))
      #  > Vacation_Contact_matrix
      #          [,1]       [,2]      [,3]      [,4]
      #[1,] 6.5227000  1.3084437  3.031601 0.1144892
      #[2,] 1.3084437 11.5761000  5.166566 0.4965009
      #[3,] 3.0316015  5.1665665 14.968000 2.4388292
      #[4,] 0.1144892  0.4965009  2.438829 1.4326000

      Vacation_Row_sum <- matrix(nrow=4, ncol=1)
      for(i in 1:4){
        Vacation_Row_sum[i,1]=sum(Vacation_Contact_matrix[i,1:4])
      }
      #  > Vacation_Row_sum
      #          [,1]
      #[1,] 10.977234
      #[2,] 18.547611
      #[3,] 25.604997
      #[4,]  4.482419

      Vacation_Mixing_matrix <- matrix(nrow=4, ncol=4)
      for (i in 1:4){
        Vacation_Mixing_matrix[i,1:4]<- Vacation_Contact_matrix[i,1:4]/Vacation_Row_sum[i,1]
      }
      #  > Vacation_Mixing_matrix
      #           [,1]      [,2]      [,3]       [,4]
      #[1,] 0.59420249 0.1191961 0.2761717 0.01042970
      #[2,] 0.07054514 0.6241289 0.2785570 0.02676900
      #[3,] 0.11839882 0.2017796 0.5845734 0.09524817
      #[4,] 0.02554183 0.1107663 0.5440877 0.31960419

      Vacation_beta_matrix <- matrix(nrow=4, ncol=4)
      for (i in 1:4){
        for (j in 1:4){
          Vacation_beta_matrix[i,j] <- Vacation_Row_sum[i,1] * Vacation_Mixing_matrix[i,j] *
multiplier
        }
      }
      Vacation_R0_matrix <- matrix(nrow = 1, ncol = 4)
      for (j in 1:4) {
        Vacation_R0_matrix[,j] <- sum(Vacation_beta_matrix[,j]) / gamma_matrix[j]
      }

###############
## Equations ##
###############

    ## These are the difference equations which describe an S-I-R model
    ## comprising children and adults as separated groups
    ## Group 1: Age 0 - 4  years
    ## Group 2: Age 5 - 19 years
    ## Group 3: Age 20 - 64 years
    ## Group 4: Age 65+ years

  Total_pop <- 310000000      ## Approximation of total US population
  Pop1 <- 0.06440 * Total_pop ## Proportion of Group 1 in the population   (0-4)
  Pop2 <- 0.20204 * Total_pop ## Proportion of Group 2 in the population   (5-19)
  Pop3 <- 0.60074 * Total_pop ## Proportion of Group 3 in the population   (20-64)
  Pop4 <- 0.13282 * Total_pop ## Proportion of Group 4 in the population   (64+)

  ## Initial number of infected people coming from overseas
  seed = 10

  ## Create a matrix for the variables
  y <- matrix(nrow = Total_timesteps+1, ncol = 13)

  ## Initial proportion of susceptible, infected and recovered populations
  ## by Group (1, 2, 3, and 4)
  y_initial <- matrix(0,1,13)
  y_initial[1,1:13] <- c(0,Pop1,Pop2,Pop3,Pop4,0,0,seed,0,0,0,0,0)
  ## Assumption: the infected person ("seed") is a working adult (Group 3)

  y[1,] <- y_initial[1,]
  S1 <- y[1,2]
  S2 <- y[1,3]
  S3 <- y[1,4]
  S4 <- y[1,5]
  I1 <- y[1,6]
  I2 <- y[1,7]
  I3 <- y[1,8]
  I4 <- y[1,9]
  R1 <- y[1,10]
  R2 <- y[1,11]
  R3 <- y[1,12]
  R4 <- y[1,13]

    N1 <- 0.06440 * Total_pop
    N2 <- 0.20204 * Total_pop
    N3 <- 0.60074 * Total_pop
    N4 <- 0.13282 * Total_pop

      gamma1 = gamma_matrix[1]
      gamma2 = gamma_matrix[2]
    gamma3 = gamma_matrix[3]
    gamma4 = gamma_matrix[4]

b <- beta_matrix

for (timestep in 1:Total_timesteps){

      ## Apply the appropriate beta matrix at the right time
      if (timestep >= school_close && timestep < school_reopen) { b <- Vacation_beta_matrix }
       else { b <- beta_matrix }

      ## These are the equations for the S-I-R model
      ## Susceptible populations
          S1_update = S1 - Timestep_size * (b[1,1]*S1*(I1/N1) + b[1,2]*S1*(I2/N2) +
      b[1,3]*S1*(I3/N3) + b[1,4]*S1*(I4/N4))
          S2_update = S2 - Timestep_size * (b[2,1]*S2*(I1/N1) + b[2,2]*S2*(I2/N2) +
      b[2,3]*S2*(I3/N3) + b[2,4]*S2*(I4/N4))
          S3_update = S3 - Timestep_size * (b[3,1]*S3*(I1/N1) + b[3,2]*S3*(I2/N2) +
      b[3,3]*S3*(I3/N3) + b[3,4]*S3*(I4/N4))
          S4_update = S4 - Timestep_size * (b[4,1]*S4*(I1/N1) + b[4,2]*S4*(I2/N2) +
      b[4,3]*S4*(I3/N3) + b[4,4]*S4*(I4/N4))

      ## Infected(Infectious) populations
          I1_update = I1 + Timestep_size * (b[1,1]*S1*(I1/N1) + b[1,2]*S1*(I2/N2) +
      b[1,3]*S1*(I3/N3) + b[1,4]*S1*(I4/N4) - gamma1*I1)
          I2_update = I2 + Timestep_size * (b[2,1]*S2*(I1/N1) + b[2,2]*S2*(I2/N2) +
      b[2,3]*S2*(I3/N3) + b[2,4]*S2*(I4/N4) - gamma2*I2)
          I3_update = I3 + Timestep_size * (b[3,1]*S3*(I1/N1) + b[3,2]*S3*(I2/N2) +
      b[3,3]*S3*(I3/N3) + b[3,4]*S3*(I4/N4) - gamma3*I3)
          I4_update = I4 + Timestep_size * (b[4,1]*S4*(I1/N1) + b[4,2]*S4*(I2/N2) +
      b[4,3]*S4*(I3/N3) + b[4,4]*S4*(I4/N4) - gamma4*I4)

      ## Recovered populations
          R1_update = R1 + Timestep_size * gamma1 * I1
          R2_update = R2 + Timestep_size * gamma2 * I2
          R3_update = R3 + Timestep_size * gamma3 * I3
          R4_update = R4 + Timestep_size * gamma4 * I4

      ## Update the matrix
           y[timestep+1,1]<- y[timestep,1] + Timestep_size

          y[timestep+1,2]<- S1_update;
          y[timestep+1,3]<- S2_update;
          y[timestep+1,4]<- S3_update;
          y[timestep+1,5]<- S4_update;   # Susceptible

          y[timestep+1,6]<- I1_update;
          y[timestep+1,7]<- I2_update;
          y[timestep+1,8]<- I3_update;
          y[timestep+1,9]<- I4_update;   # Infected (Infectious)

          y[timestep+1,10]<- R1_update;
          y[timestep+1,11]<- R2_update;
          y[timestep+1,12]<- R3_update;
          y[timestep+1,13]<- R4_update;  # Recovered

      ## Update the state variable for the next time step
          S1 <- S1_update;
          S2 <- S2_update;
          S3 <- S3_update;
          S4 <- S4_update;   # Susceptible

          I1 <- I1_update;
          I2 <- I2_update;
          I3 <- I3_update;
          I4 <- I4_update;   # Infected (Infectious)

          R1 <- R1_update;
          R2 <- R2_update;
          R3 <- R3_update;
          R4 <- R4_update;   # Recovered
```

```
        }
newoutputall <- y

newmatrix <- matrix(ncol = 2, nrow = length(newoutputall[,1]) )
newmatrix[,1] <- y[,1]
for (n in 1:length(newoutputall[,1])){
        newmatrix[n,2] <- sum(newoutputall[n,6:9])*sym_ratio
        }

## Plot figures
plot(newmatrix[,1], newmatrix[,2], type="l", xlab="Time (days)", ylab="Cases (million)",
col="red", lwd=3, lty=1, xlim=c(0,100), ylim=c(0,15000000), yaxt='n')

## Add y-axis, arrows and text
axis(2, at = c(0,13000000), labels=c("0","13"))
arrows(0,5000000,0,0,code=2,col=c("black"))
arrows(5,8000000,5,0,code=2,col=c("dark grey"))
arrows(5,8000000,(5+28),8000000,code=2,col=c("dark grey"))
text(10,6000000, "outbreak begins")
text(18,9000000,"school closure (28d)")
print(max(newmatrix[,2]))
}
```