########################################

### MIDAS-CCDD Outreach Conference ###

### March 4th 2024 ###

### Breakout session #1 : SIR Basics ###

########################################

# Adapted from material by Mathew Kiang and Stephen Kissler

### Install required package

# You only need to do this once on any computer.

install.packages("deSolve", dep = TRUE)

## Load deSolve package (so we can use the lsoda() command below)

# You'll need to do this very time you open a new R session.

library(deSolve)

########################################

### TUTORIAL ###

### 'BASIC SIR MODEL' ###

### (Questions 1, 2 and 3) ###

########################################

### Set time steps, initial values, and parameters based on information provided in the question

# Hint: Make sure all the parameters are coded in days.

# Also make sure that the time steps are in the same time units as your parameters (days).

dt <- seq(from = , to= , by= ) ## this is a vector with the time steps of the simulation

# Hint: If you don't know what this function does in R, type '?seq()' in the console below.

# A help page describing this function and the inputs required will appear on the right.

inits <- c(S= , I= , R= ) ## this is a vector with the number of people starting in each compartment

# S = Number of Susceptible individuals at the first time point

# I = Number of Infected individuals at the first time point

# R = Number of Recovered individuals at the first time point

parms <- c(b= , k= , r= ) ## this is a vector with the parameters of the model

# b = probability of transmission given infectious contact

# k = average number of contacts per time step

# r = recovery rate (rate at which individuals transition out of the I compartment into the R compartment)

### Create an ODE model

# The solver needs your model written as a function that takes in a vector of times, initial values, and parameters

# (in that order) and returns a list with derivatives of your compartments relative to time.

SIR <- function(t, x, parms){ # do not change the order of these inputs

 # t is the vector of time-steps;

 # x is the current state of the model;

 # parms is the vector of parameters

 with(as.list(c(parms,x)),{ # "with" allows us to refer to parms and x by shorthand

 N <- S+I+R # N : total number of individuals in the population at each time step. Here, it remains constant because we make the simplifying assumptions that there are no births and no deaths.

 dS <- - (b\*k\*S\*I)/N # dS : the difference in the number of individuals in the 'Susceptible' compartment at each time point.

 dI <- + (b\*k\*S\*I)/N - r\*I # dI : the difference in the number of individuals in the 'Infected' compartment at each time point.

 dR <- r\*I # dS : the difference in the number of individuals in the 'Recovered' compartment at each time point.

 der <- c(dS, dI,dR)

 list(der) # the output must be returned as a list

 }) # end of 'with'

} # end of function definition

### Run the ode model

simulation <- as.data.frame(lsoda(y = inits,

 times = dt,

 func = SIR,

 parms = parms))

# Check first few rows of the simulation results

head(simulation, 10)

# Check final values of the simulation results

tail(simulation, 10)

## Plot results

matplot(x = simulation[,1], y = simulation[,2:4],

 type= "l", lty = 1,

 xlab = "Time", ylab = "People (count)",

 main = "Simulation results")

# Add a legend

legend(x = "right", legend = c('S', 'I', 'R'),

 col = 1:3, lty = 1)

########################################

### TUTORIAL ###

### 'SEIR MODEL' ###

### (Question 4) ###

########################################

# Only move on to this part after completing questions 1,2 and 3

### Set time steps, initial values, and parameters based on information provided in the question

dt <- seq(from = , to= , by= )

inits <- c(S= , E= , I= , R= ) # E: new 'Exposed' compartment

parms <- c(b= , k= , a=, r= ) # a = latency period

### Create an ODE model

SEIR <- function(t, x, parms){ # do not change the order of these inputs

 # t is the vector of time-steps;

 # x is the current state of the model;

 # parms is the vector of parameters

 with(as.list(c(parms,x)),{ # "with" allows us to refer to parms and x by shorthand

 N <- S+E+I+R

 dS <- - (b\*k\*S\*I)/N

 dE <- + (b\*k\*S\*I)/N - a\*E # New compartment

 dI <- + (a\*E) - r\*I

 dR <- r\*I

 der <- c(dS, dE, dI, dR)

 list(der) # the output must be returned as a list

 }) # end of 'with'

} # end of function definition

### Run the ode model

simulation <- as.data.frame(lsoda(y = inits,

 times = dt,

 func = SEIR,

 parms = parms))

## Plot results

matplot(x = simulation[,1], y = simulation[,2:5],

 type= "l", lty = 1,

 xlab = "Time", ylab = "People (count)",

 main = "Simulation results",

 col = c(1,4,2,3))

# Add a legend

legend(x = "right", legend = c('S', 'E', 'I', 'R'),

 col = c(1,4,2,3), lty = 1)

########################################

### TUTORIAL ###

### 'Adding birth and deaths' ###

### (Question 5) ###

########################################

# Only move on to this part after completing questions 4

### Set time steps, initial values, and parameters based on information provided in the question

dt <- seq(from = , to= , by= )

inits <- c(S= , I= , R= )

parms <- c(b= , k= , r= , birth = , death = ) # Add birth and death rates

### Create an ODE model

SIR\_steadystate <- function(t, x, parms){

 with(as.list(c(parms,x)),{

 N <- S+I+R

 dS <- - (b\*k\*S\*I)/N + (birth\*N) - (death\*S) # Add births and deaths

 dI <- + (b\*k\*S\*I)/N - r\*I - (death\*I) # Add deaths

 dR <- r\*I - (death\*R) # Add deaths

 der <- c(dS, dI,dR)

 list(der)

 })

}

### Run the ode model

simulation <- as.data.frame(lsoda(y = inits,

 times = dt,

 func = SIR\_steadystate,

 parms = parms))

## Plot results

matplot(x = simulation[,1], y = simulation[,2:4],

 type= "l", lty = 1,

 xlab = "Time", ylab = "People (count)",

 main = "Simulation results")

# Add a legend

legend(x = "right", legend = c('S', 'I', 'R'),

 col = 1:3, lty = 1)