Modelling Club
Model Code

Rachel Sippy

19th February, 2023

R Code for Compartmental Models

SEIR

Remember for setting up compartmental models, there are several steps. First we load the `deSolve` package, which we use to solve equations in the model.

```
library(deSolve)
```

We need to set up the compartments and equations for the model inside a function, which we call `seir.model`. Let's start with a simple SEIR model, which has compartments S, E, I, R and parameters β, σ, and γ.

```
seir.model <- function (t, x, params) { # the function needs three arguments
    S <- x[1]                           # create local variable S
    E <- x[2]                           # create local variable E
    I <- x[3]                           # create local variable I
    R <- x[4]                           # create local variable R
    with(                               # code is simpler using "with"
        as.list(params),                # with takes the variables as a list
        {                               # the system of rate equations
            dS <- -beta*S*I
            dE <- beta*S*I - sigma*E
            dI <- sigma*E - gamma*I
            dR <- gamma*I
            dx <- c(dS,dE,dI,dR)        # combine results into vector dx
            list(dx)                    # return result as a list
        }
    )
}
```

Next we create three objects called `xstart`, `params`, and `times`. We set up the initial conditions for the compartments in `xstart` (the proportion of people in each compartment), the values of the model parameters β, σ, and γ in `params`, and the amount of time to run the model in `times`.

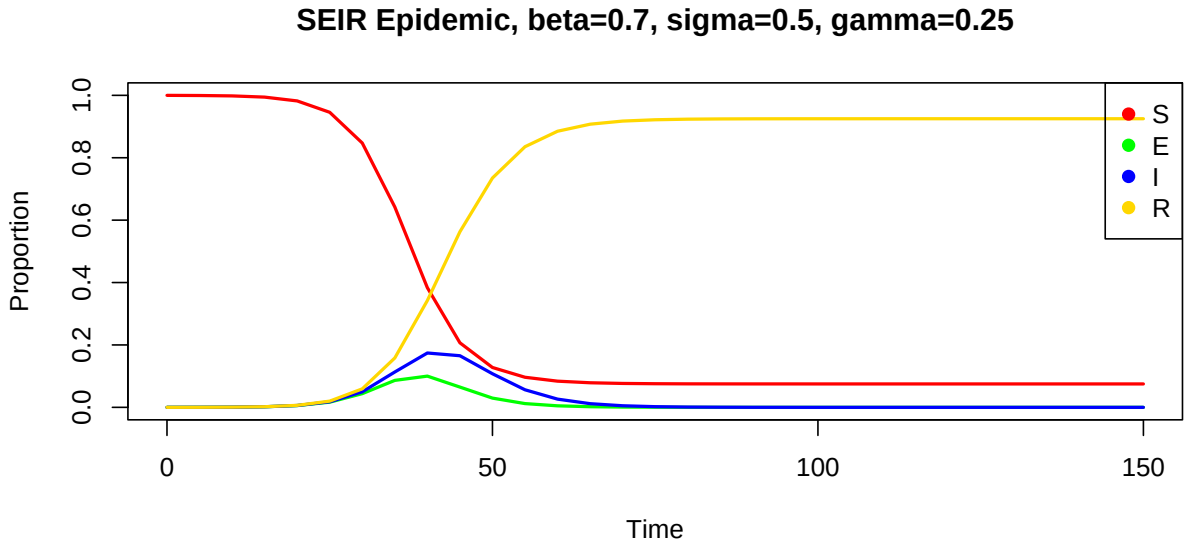
```
xstart <- c(S=9999/10000, E=0, I=1/10000, R=0)    # initial conditions
times <- seq(0, 150, by=5)                        # 150 days, 5 days at a time
params <- c(beta=0.7, sigma=1/2, gamma=1/4)       # parameter values
```

Then we can run the model with the numbers we've set, using the function `lsoda()` to solve the equations.

```
out <- as.data.frame(lsoda(xstart, times, seir.model, params))
```

The output is the proportion of people in each compartment at each time step (every 5 days). We can plot this:

```
plot(NA,NA, xlab="Time", ylab="Proportion", main=paste0("SEIR Epidemic, beta=", params[1], ", sigma=", p
     ylim=c(0,1), xlim=c(0, max(times)))
lines(out$time, out$S, col="red", lwd=2)
lines(out$time, out$E, col="green", lwd=2)
lines(out$time, out$I, col="blue", lwd=2)
lines(out$time, out$R, col="gold", lwd=2)
legend("topright", pch=19, c("S","E","I","R"), col=c("red","green","blue","gold"))
```



We can find the time point with the highest proportion of infected cases, and the total proportion that were infected:

```
# time when I is highest
out$time[out$I==max(out$I)]
```

```
[1] 40
```

```
# proportion who were infected are those who recovered
max(out$R)
```

```
[1] 0.924984
```

SEIR with Hospitalization

The previous model assumed that everyone recovered from the infection, but we know that many with COVID-19 needed to be hospitalized, and we are interested in estimating hospitalizations for our models. We can add hospitalization to our model. We will modify the function we created, which we call `seir.hosp`, which has one additional compartment H, and parameters τ (the hospitalization rate) and γ_H (the recovery rate for the hospitalized).

```
seir.hosp <- function (t, x, params) { # the function needs three arguments
    S <- x[1]                          # create local variable S
    E <- x[2]                          # create local variable E
    I <- x[3]                          # create local variable I
    H <- x[4]                          # create local variable H
    R <- x[5]                          # create local variable R
    with(                              # code is simpler using "with"
        as.list(params),               # with takes the variables as a list
        {                              # the system of rate equations
            dS <- -beta*S*I
            dE <- beta*S*I - sigma*E
            dI <- sigma*E - gamma*I - tau*I
            dH <- tau*I - gammaH*H
            dR <- gamma*I + gammaH*H
            dx <- c(dS,dE,dI,dH,dR)    # combine results into vector dx
            list(dx)                   # return result as a list
        }
    )
}
```

We add the initial conditions for the hospitalized compartment H to `xstart`, and the values of τ and γ_H to `params`.

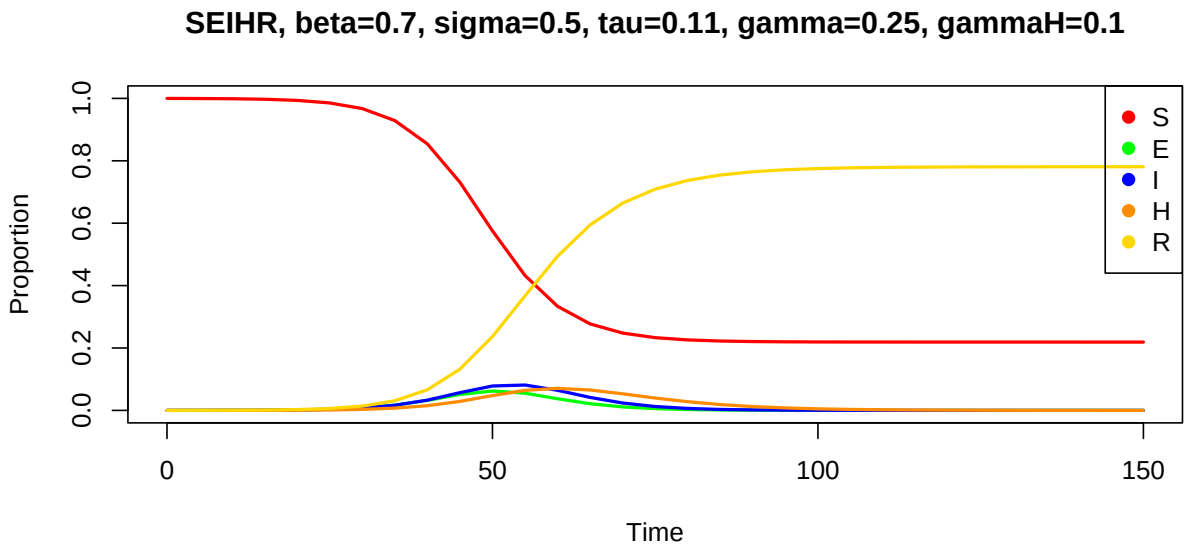
```
xstart <- c(S=9999/10000, E=0, I=1/10000, H=0, R=0) # initial conditions
times <- seq(0, 150, by=5)                          #
params <- c(beta=0.7, sigma=1/2, tau=0.11,          # add new parameters
            gamma=1/4, gammaH=1/10)
```

Then we can run the hospitalization model, again using the function `lsoda()` to solve the equations.

```
out.h <- as.data.frame(lsoda(xstart, times, seir.hosp, params))
```

We can plot the output from this model:

```
plot(NA,NA,xlab="Time",ylab="Proportion",main=paste0("SEIHR, beta=", params[1], ", sigma=", params[2], "
     ylim=c(0,1), xlim=c(0, max(times)))
lines(out.h$time, out.h$S, col="red", lwd=2)
lines(out.h$time, out.h$E, col="green", lwd=2)
lines(out.h$time, out.h$I, col="blue", lwd=2)
lines(out.h$time, out.h$H, col="darkorange", lwd=2)
lines(out.h$time, out.h$R, col="gold", lwd=2)
legend("topright", pch=19, c("S", "E", "I", "H", "R"), col=c("red", "green", "blue", "darkorange", "gol
```



We can find the time point with the highest proportion of infected cases, and the highest proportion of hospitalizations:

```
# time when I is highest
out.h$time[out.h$I==max(out.h$I)]
```

```
[1] 55
```

```
# time when H is highest
out.h$time[out.h$H==max(out.h$H)]
```

```
[1] 60
```

However, if we want to know the total proportion who were hospitalized, we need to add an additional compartment to our model, to capture the hospitalized people who recovered *separately* from those who recovered that were never hospitalized. We add R_H to the model:

```
seir.hosp2 <- function (t, x, params) { # the function needs three arguments
    S <- x[1]                           # create local variable S
    E <- x[2]                           # create local variable E
    I <- x[3]                           # create local variable I
    H <- x[4]                           # create local variable H
    R <- x[5]                           # create local variable R
    RH <- x[6]                          # create local variable RH
    with(                               # code is simpler using "with"
        as.list(params),                # with takes the variables as a list
        {                               # the system of rate equations
            dS <- -beta*S*I
            dE <- beta*S*I - sigma*E
            dI <- sigma*E - gamma*I - tau*I
            dH <- tau*I - gammaH*H
            dR <- gamma*I
```

```
            dRH <- gammaH*H
            dx <- c(dS,dE,dI,dH,dR,dRH)    # combine results into vector dx
            list(dx)                       # return result as a list
        }
    )
}
```

We update `xstart` to include the initial conditions for the recovered hospitalizations R_H.

```
xstart <- c(S=9999/10000, E=0, I=1/10000, H=0, R=0, RH=0) # initial conditions
```

We run this updated model:

```
out.h2 <- as.data.frame(lsoda(xstart, times, seir.hosp2, params))
```

Now we can see the total proportion infected without hospitalization, the total proportion hospitalized, and add them together for the total infected:

```
# total I without hospitalization
max(out.h2$R)
```

```
[1] 0.5423657
```

```
# total hospitalized
max(out.h2$RH)
```

```
[1] 0.2386016
```

```
# total infected
max(out.h2$R) + max(out.h2$RH)
```

```
[1] 0.7809673
```

For our models, we are also interested in modeling the number of deaths. This would require us to add another compartment D and another parameter δ (death rate). We would need to modify the code in a similar way as before to accomplish this.

SEIHR with Deaths

We add death to the model, allowing for deaths to occur among the hospitalized and infected people in the community. We set up this model function, which we call `seir.hospdeath`. It has compartments S, E, I, D, H, DH, R and RH and parameters β, σ, τ, δ_H, δ, γ_H and γ.

```
seir.hospdeath <- function (t, x, params) {
  S <- x[1]                            # create local variable S
  E <- x[2]                            # create local variable E
  I <- x[3]                            # create local variable I
  H <- x[4]                            # create local variable H
  RH <- x[5]                           # create local variable RH
```

```
  DH <- x[6]                              # create local variable DH
  R <- x[7]                               # create local variable R
  D <- x[8]                               # create local variable D
  with(                                   # code is simpler using "with"
    as.list(params),                    # with takes the variables as a list
    {                                   # the system of rate equations
      dS <- -beta*S*I
      dE <- beta*S*I - sigma*E
      dI <- sigma*E - (gamma)*I - (tau)*I - delta*I
      dH <- (tau)*I - (gammaH)*H - deltaH*H
      dRH <- gammaH*H
      dDH <- deltaH*H
      dR <- (gamma)*I
      dD <- delta*I
      dx <- c(dS,dE,dI,dH,dRH,dDH,dR,dD)       # combine results
      list(dx)                          # return result as a list
    }
  )
}
```

We add the initial conditions for the additional compartments DH and D to `xstart`, and the values of δ and δ_H to `params`.

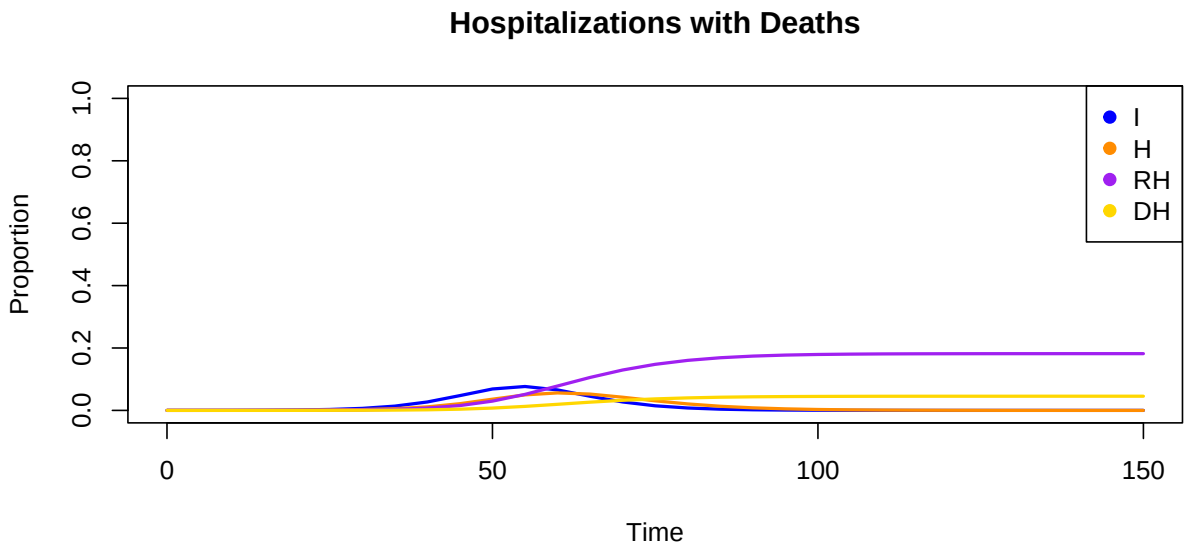
```
xstart <- c(S=9999/10000, E=0, I=1/10000, H=0, RH=0, DH=0, R=0, D=0)
times <- seq(0, 150, by=5)                              #
params <- c(beta=0.7, sigma=1/2, tau=0.11, delta=0.01, gamma=1/4, gammaH=1/10,
            deltaH=0.025)          # parameter values
```

We run this updated model:

```
out.hd <- as.data.frame(lsoda(xstart, times, seir.hospdeath, params))
```

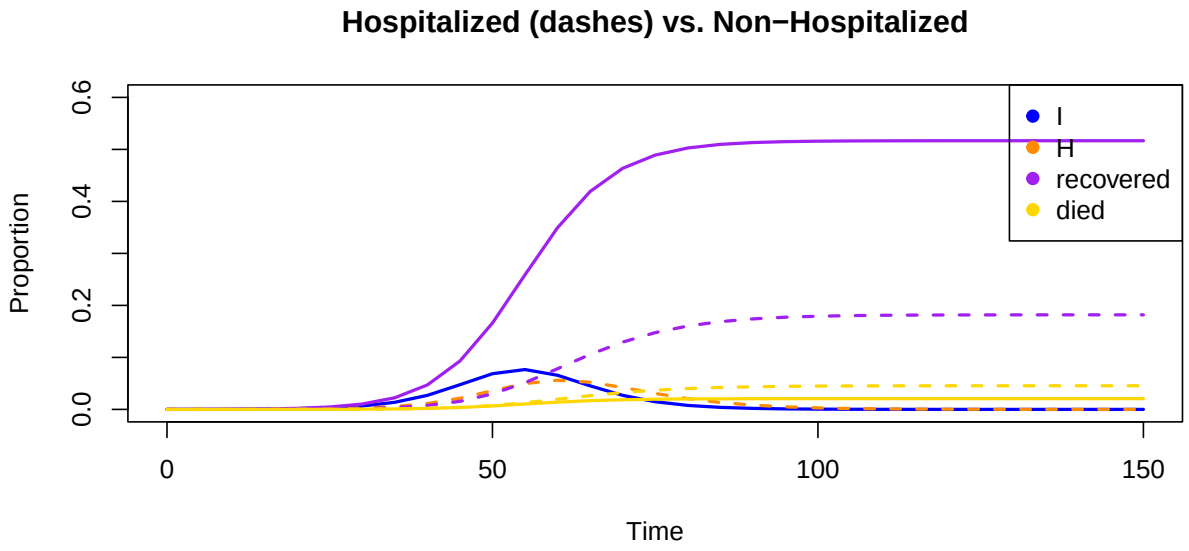
We can plot the outcomes for those who were hospitalized:

```
plot(NA,NA,xlab="Time",ylab="Proportion",main="Hospitalizations with Deaths",
     ylim=c(0,1), xlim=c(0, max(times)))
lines(out.hd$time, out.hd$I, col="blue", lwd=2)
lines(out.hd$time, out.hd$H, col="darkorange", lwd=2)
lines(out.hd$time, out.hd$RH, col="purple", lwd=2)
lines(out.hd$time, out.hd$DH, col="gold", lwd=2)
legend("topright", pch=19, c("I", "H", "RH", "DH"),
       col=c("blue", "darkorange", "purple", "gold"))
```



We can compare hospitalized and non-hospitalized outcomes:

```
plot(NA,NA,xlab="Time",ylab="Proportion",main="Hospitalized (dashes) vs. Non-Hospitalized",
     ylim=c(0,0.6), xlim=c(0, max(times)))
lines(out.hd$time, out.hd$I, col="blue", lwd=2)
lines(out.hd$time, out.hd$H, col="darkorange", lwd=2, lty=2)
lines(out.hd$time, out.hd$RH, col="purple", lwd=2, lty=2)
lines(out.hd$time, out.hd$R, col="purple", lwd=2)
lines(out.hd$time, out.hd$DH, col="gold", lwd=2, lty=2)
lines(out.hd$time, out.hd$D, col="gold", lwd=2)
legend("topright", pch=19, c("I", "H", "recovered", "died"),
       col=c("blue", "darkorange", "purple", "gold"))
```



We can calculate total cases, total hospitalizations, total deaths, and compare outcomes among the hospitalized/non-hospitalized:

```
# total cases:
max(out.hd$D) + max(out.hd$DH) + max(out.hd$R) + max(out.hd$RH)
```

```
[1] 0.7646625
```

```
# total hospitalized:
max(out.hd$RH) + max(out.hd$DH)
```

```
[1] 0.2273256
```

```
# total deaths:
max(out.hd$D) + max(out.hd$DH)
```

```
[1] 0.06613193
```

```
# non hosp who died
max(out.hd$D)/(max(out.hd$R) + max(out.hd$D))
```

```
[1] 0.03846154
```

```
# hosp who died
max(out.hd$DH)/(max(out.hd$RH) + max(out.hd$DH))
```

```
[1] 0.2
```