The SIR model

A model developed based on a template markdown file

Lubos Polerecky and Karline Soetaert

July 2021

Contents

Introduction

This document describes a simple SIR model, as specified in the COVID exercise:

```
require(RTM)
RTMexercise("COVID")
```

Model definition

```
# Initial conditions of the state variables (D=deceased)
yini <- c(S = 17.5e6, I = 1e3, R = 0, D = 0) # number of people

# Model parameters
pars <- c(
  b   = 0.00000002, # [/ind/d]  infection rate constant
  g   = 0.07,       # [/d]      recovery  rate constant
  m   = 0.007       # [/d]      mortality rate constant
)

# Model function: calculates time-derivatives and other output
SIRmodel <-function(t, state, pars) {
  # t: time, state: state variables, pars: model parameters
  with (as.list(c(state, pars)),{

  # rate expressions [ind/d]
    Infection <- b * I * S # infection rate
    Recovery  <- g * I          # recovery rate
    Mortality <- m * I          # mortality rate

  # Time-derivatives: dC/dt = production - consumption [ind/d]
    dSdt      <- -Infection
    dIdt      <-  Infection - Recovery - Mortality
    dRdt      <-  Recovery
    dDdt      <-  Mortality

  # return time-derivatives and ordinary variables as a list
    list(c(dSdt, dIdt, dRdt, dDdt), # vector with derivatives
                        # (the same order as state variables!)
      Infection = Infection,  # other output
      Mortality = Mortality)
  })
}
```
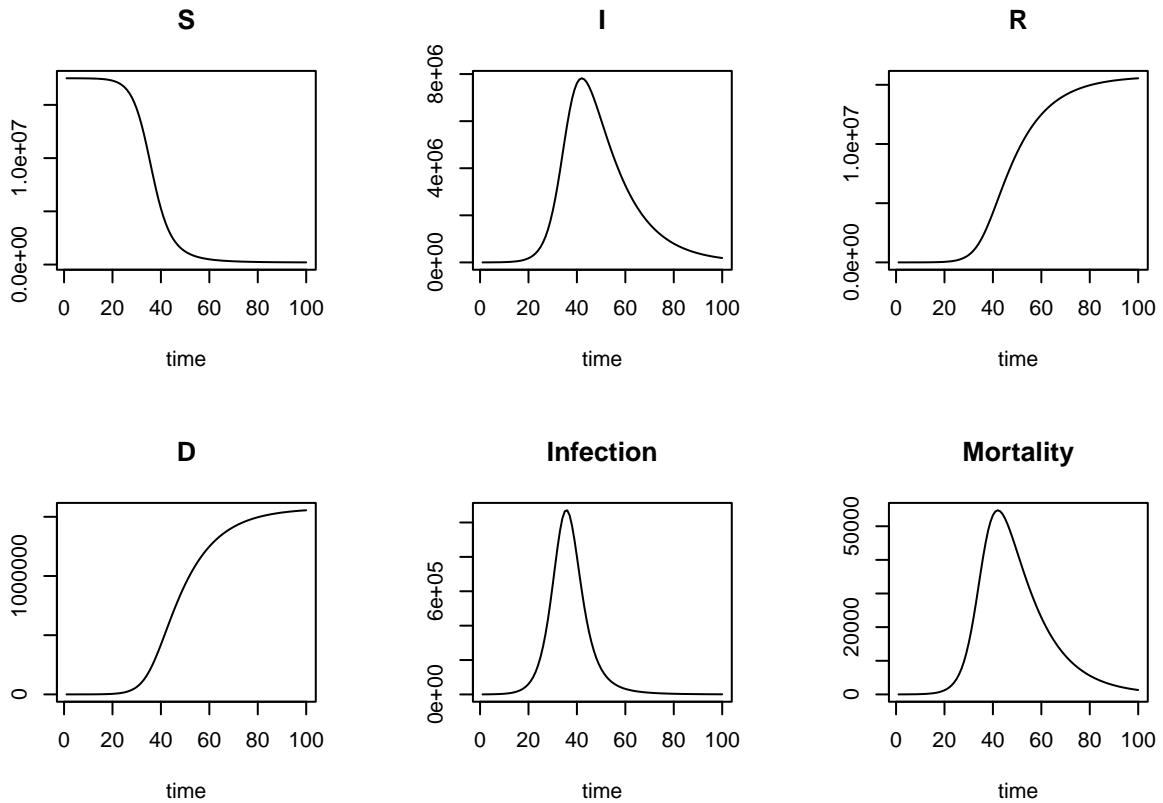
Model solution

Dynamic solution

We run the model dynamically over 100 days, using two different values of the reaeration rate constant:

```
require(deSolve)  # package with integration methods
# vector of output times
outtimes <- seq(from = 1, to = 100, length.out = 100)
# ode integrates the model
out <- ode(y=yini, parms=pars, func=SIRmodel, times=outtimes)
# plot the model output
plot(out)
```

```r
# change the value of the infection rate constant
pars2       <- pars              # copy the original parameter vector
pars2["b"] <- pars["b"]*0.6  # 60% of original

# integrate the model with the new parameters
out2 <- ode(y = yini, parms = pars2, func = SIRmodel,
            times = outtimes)
```

```r
# print summary of the solution
summary(out)
```

We plot both solutions in one graph:

```r
plot(out, out2, xlab="time (days)", las=1, lwd=2,
     ylab=list("ind","ind","ind", "ind", "ind/d","ind/d"))
legend("topright", legend = c("original", "60% reduced"),
       title="infection rate constant:",
       col=1:2, lwd=2, lty=1:2, bty="n")
```
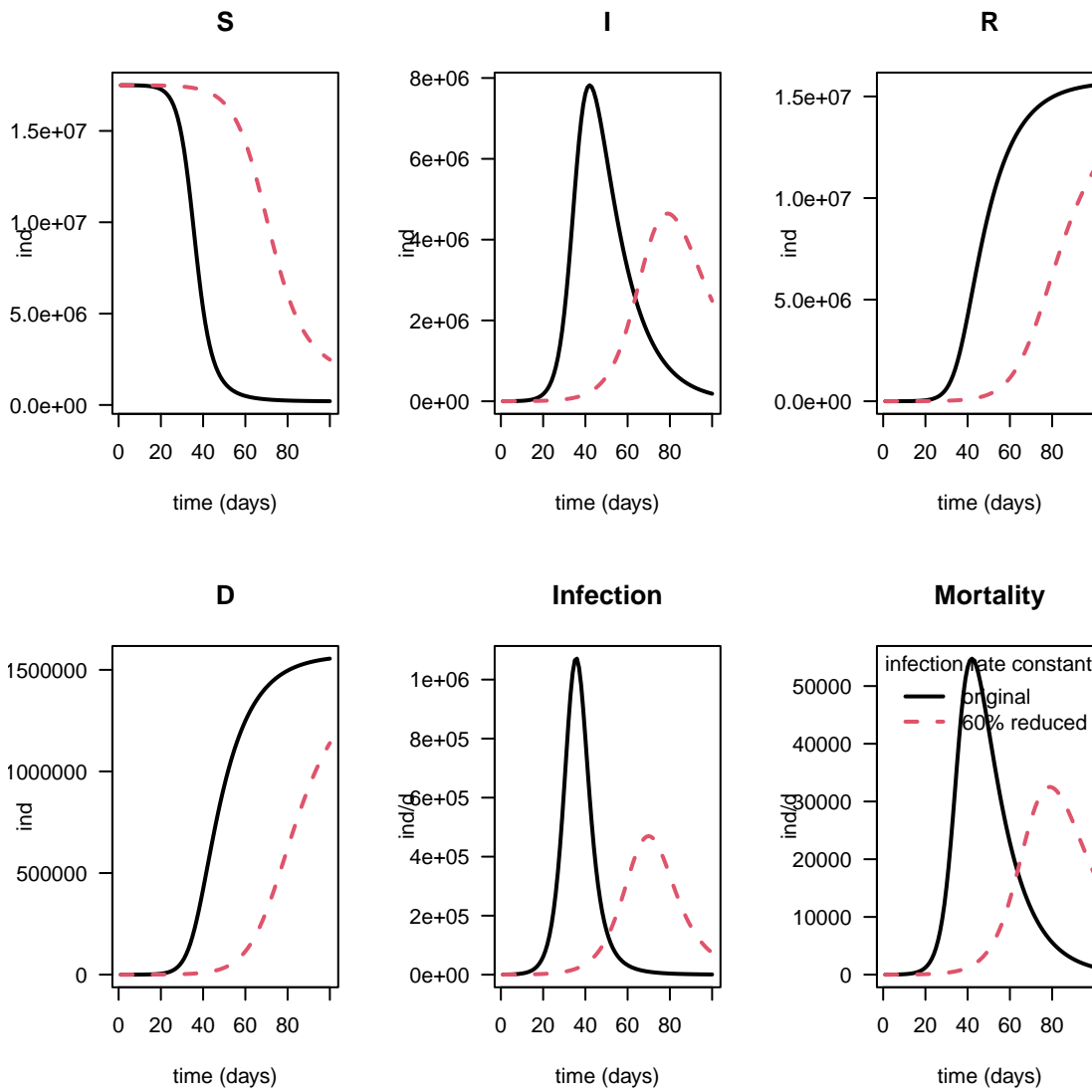
Steady-state solution

We find the steady-state solution:

```r
require(rootSolve) # package with solution methods
std <- steady(y = yini, parms = pars, func = SIRmodel,
              positive = TRUE) # to ensure that the solution is positive
```

```
## Warning in stode(y, times, func, parms = parms, ...): error during factorisation
## of matrix (dgefa); singular matrix
```

```
## diagonal element is zero
## [1] 4
```

```
## Warning in stode(y, times, func, parms = parms, ...): steady-state not reached
```

```r
std$y
```

4

```
##           S         I         R         D
## 17500000      1000         0         0
```

Model definition with a forcing function

```r
# Initial conditions of the state variables (D=deceased)
yini <- c(S = 17.5e6, I = 1e3, R = 0, D = 0) # number of people

# Model parameters
pars <- c(
  b    = 0.00000002, # [/ind/d]  infection rate constant
  g    = 0.07,       # [/d]      recovery  rate constant
  m    = 0.007,      # [/d]      mortality rate constant
  sInt = 1           # [-]       level of social interactions
)

# Model function: calculates time-derivatives and other output
SIRmodel2 <-function(t, state, pars, sIntDyn) {
  # t: time, state: state variables, pars: model parameters
  with (as.list(c(state, pars)),{

  # social interactions parameter sint is determined externally
    sInt <- sIntDyn(t)

  # rate expressions [ind/d]
    Infection <- sInt * b * I * S # infection rate
    Recovery  <- g * I            # recovery rate
    Mortality <- m * I            # mortality rate

  # Time-derivatives: dC/dt = production - consumption [ind/d]
    dSdt      <- -Infection
    dIdt      <-  Infection - Recovery - Mortality
    dRdt      <-  Recovery
    dDdt      <-  Mortality

  # return time-derivatives and ordinary variables as a list
    list(c(dSdt, dIdt, dRdt, dDdt), # vector with derivatives
                    # (the same order as state variables!)
      Infection = Infection,       # other output
      Mortality = Mortality)
  })
}
```

Now we define the forcing function:

```r
# define forcing function for the social interactions parameter
# sint = 1: no restrictions, sint = 0.1: restricted to 10% of normal
sIntDATA <- data.frame(time = c(0, 29,  30,  69, 70, 100),
                       sInt = c(1,  1, 0.1, 0.1,  1,   1))
sIntFUN  <- approxfun(x=sIntDATA)
```

Plot the forcing function:

```r
plot(outtimes, sIntFUN(outtimes), main="Restriction on social interactions",
     type = "l", col=2, xlab="time (d)", ylab="sInt", ylim=c(0, 1))
points(sIntDATA, col=1, pch=5)
```

Restriction on social interactions
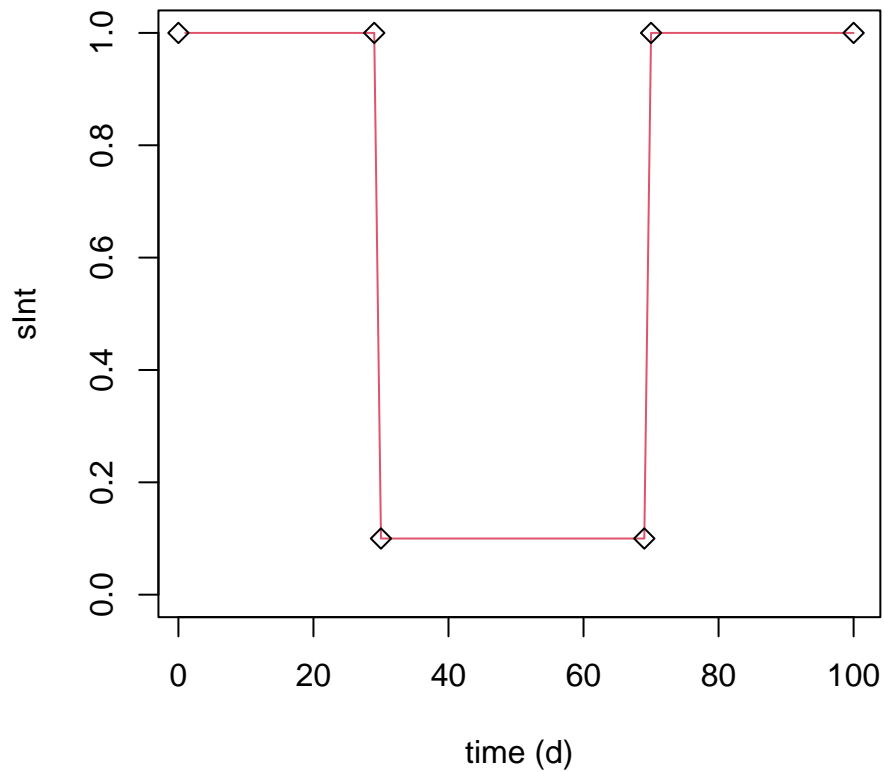
Model solution with a forcing function

```
require(deSolve)
# ode integrates the model
out2 <- ode(y=yini, parms=pars, func=SIRmodel2, times=outtimes,
            sIntDyn=sIntFUN) # forcing function included
# plot the model output
plot(out, out2)
```
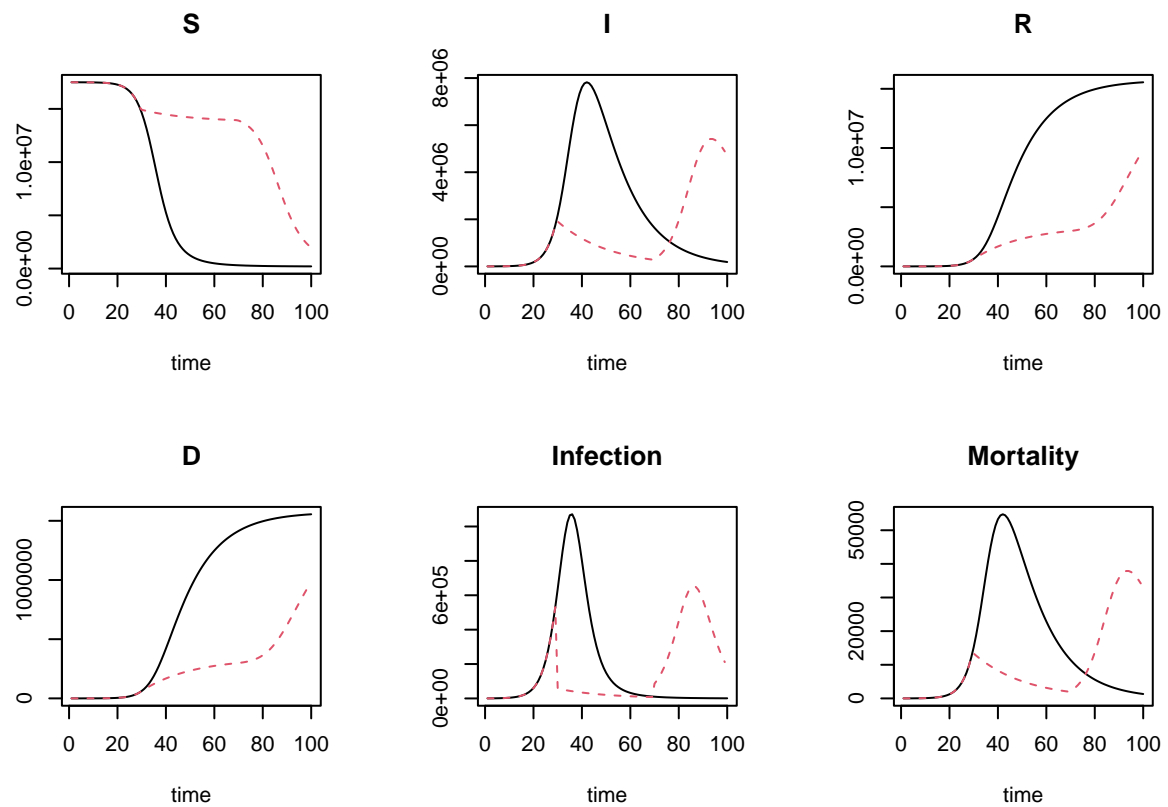
We plot both solutions in one graph:

```
plot(out, out2, xlab="time (days)", las=1, lwd=2,
     ylab=list("ind","ind","ind", "ind", "ind/d","ind/d"))
legend("topright", legend = c("original", "forced"),
       title="infection rate constant:",
       col=1:2, lwd=2, lty=1:2, bty="n")
```
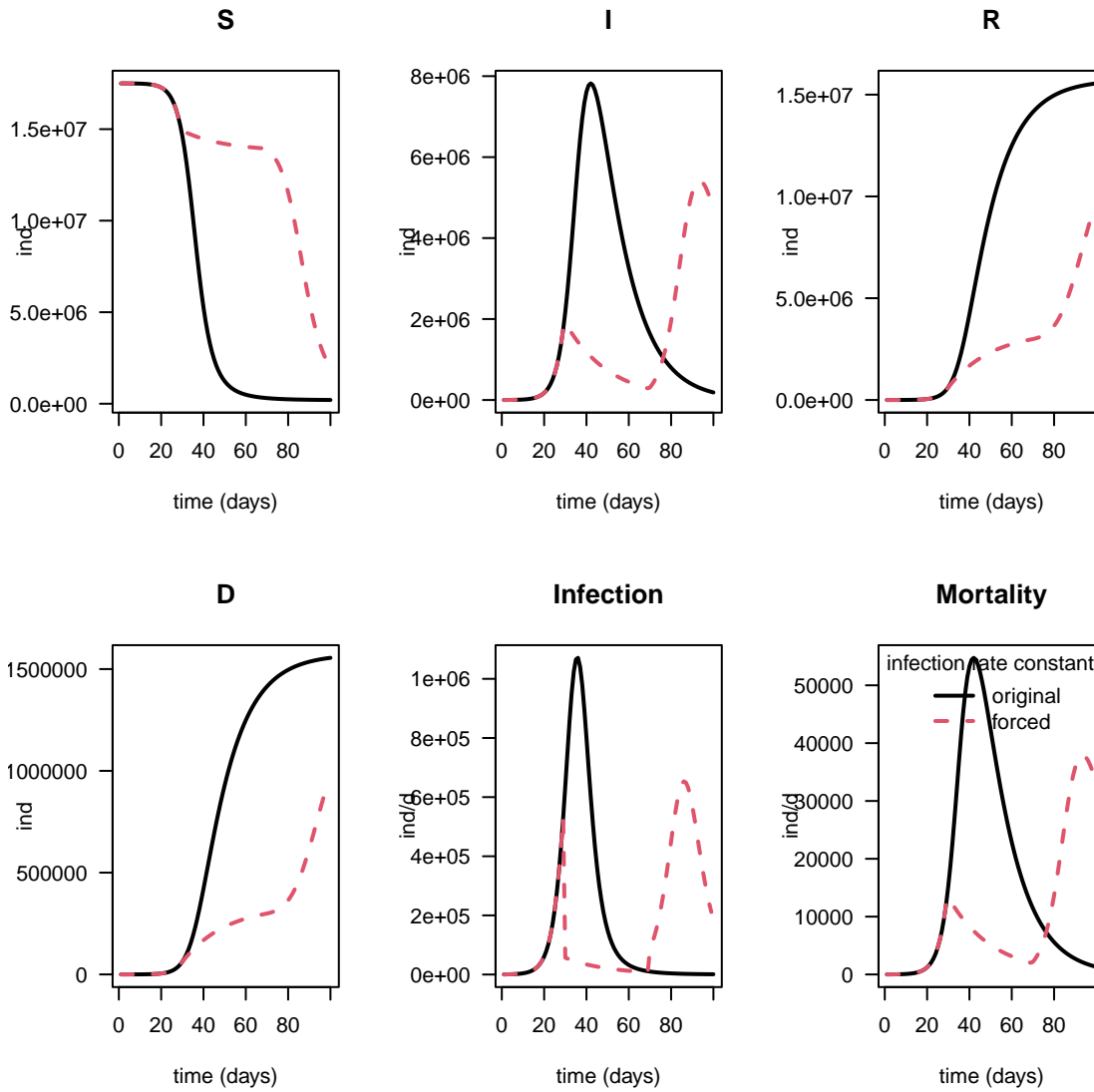
References

R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/.

Soetaert Karline (2009). rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations. R-package version 1.6

Soetaert Karline, Thomas Petzoldt, R. Woodrow Setzer (2010). Solving Differential Equations in R: Package deSolve. Journal of Statistical Software, 33(9), 1–25. http://www.jstatsoft.org/v33/i09/ DOI: 10.18637/jss.v033.i09