

Debates about vaccines are [ongoing](https://www.bbc.com/news/blogs-echochambers-26722871) in [many](https://edition.cnn.com/2015/07/20/health/vaccines-fast-facts/index.html) [countries](https://afpa.org/obligation-vaccinale/) and the debate has  
reblossomed in Denmark after we’ve had five recent occurrences of  
measels. While that is nothing compared to the [measles outbreak  
currently ravaging Japan](https://www.theguardian.com/world/2019/feb/25/japan-battles-worst-measles-outbreak-in-a-decade), it is still enough to worry the health authorities that it might  
result in an epidemic. Here we’ll use Shiny to create an app that shows the  
impact of contagious diseases and the influence of  
vaccination. Wrapping the computations in a Shiny app will allow  
non-R-users to tweak the input parameters themselves and observe the  
consequences of an outbreak. Hopefully, this can lead to a more informed discussion about vaccination.

**The SIR compartmental model**

The **s**usceptible-**i**nfectious-**r**ecovered [(SIR)  
model](https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology)  
is one of the simplest compartmental models and it has previously been  
successfully used to describe outbreaks of measles, the flu, small  
pox, mumps, and rubella. The SIR model is easily extended  
to accommodate immunities due to earlier infections or  
vaccinations. Implementing the SIR model in R has so in this post we will extend this previous work adding an additional component to accommodate previously vaccinated individuals, and warp everything  
in a Shiny app.

We will use a 4-compartment model where each individual in the population initially can be classified into 4 categories: susceptible (S), the infected (I), the recovered (R), and a group of previously vaccinated/immune (V) individuals. The number of individuals starting in compartment I will be very small since that is just the initial number of infected individuals, the initial number of persons in R will be zero since no one has yet had the disease and recovered from it as part of the *current* outbreak. The remaining individuals will be in either S (have not had the disease before) or V (vaccinated / immune from earlier infection). The setup is sketched in the diagram below. Adding the group V to the SIR model reduces the spread of the disease since the disease cannot infect individuals that it comes into contact with if they are already immune or vaccinated.

This variant of the SIR model is useful for modeling airborne diseases and in the model we disregard individuals who die from other causes than the disease, new vaccinations, and demographic changes in the population.11 Consequently, we are also assuming that the total population size is constant over the time period, and we will also assume that the transition rates \(\beta\) and \(\gamma\) are constant over the period, and that anyone in the population can get in contact with anyone else.

To compute the consequences of an outbreak we need to set some initial parameters for model. The parameters directly influencing the model are

* \(\beta\) – the transition rate from compartment \(S\) to \(I\). This rate is defined as the basic reproductive number, \(R0\), divided by the infection period (i.e., the average number of individuals each infected person will infect in an unprotected population divided by the number of days that the person can pass on the disease)
* \(\gamma\) – the transition rate from \(I\) to \(R\). This is equal to the inverse of the disease period since once the disease period is over, a person automatically transfers to the \(R\) group.

Consequently we need to allow the user to set the following

* The reproductive number \(R0\) – the average number of individuals that each infected person may infect, i.e., how contagious is the disease,
* the infection period,
* the population size,
* the number of individuals initially infected,
* the proportion of individuals in the immune/vaccine group \(V\). This percentage should be multiplied by the vaccine effectiveness if it is not 100%.
* the time frame to consider.

An example of these initial parameters is shown in the code below

# Set parameters

timeframe <- 200 # Look at development over 200 days

pinf <- 5 # Initial number of infected persons

popsize <- 5700000 # Population size (5.7 mio in Denmark)

pvac <- .90 # Proportion vaccinated

vaceff <- .95 # Effectiveness of vaccine

connum <- 15 # R0, the reproductive number. 15 is roughly measles

infper <- 14 # Infection period, 14 days

# Compute the proportions in each of the compartments at

# the initial outbreak

init <- c(S = 1 - pinf / popsize - pvac \* vaceff,

I = pinf / popsize,

R = 0,

V = pvac \* vaceff

)

# First set the parameters for beta and gamma

parameters <- c(beta = connum / infper,

gamma = 1 / infper)

## Time frame

times <- seq(0, timeframe, by = .2)

Wonderful!

Based on the model outlined above we can now set up the following set of coupled differential equations22 Without loss of generality we have divided by the total number of individuals in the population, \(N\), to obtain the *proportion* of individuals in each compartment.

\[\begin{array}{l}  
\frac{dS}{dt} = -\beta I(t)S(t) \\  
\frac{dI}{dt} = \beta I(t)S(t) – \gamma I(t) = [\beta S(t) – \gamma] I(t)\\  
\frac{dR}{dt} = \gamma I(t) \\  
\frac{dV}{dt} = 0  
\end{array}\]

If \(\beta S(t) – \gamma>0\) then the number of infected individuals will increase and an epidemic will ensure. If \(\beta S(t) – \gamma<0\) then the number of infected will decrease and the disease will die out by itself. Note that since \(S(t)\) decreases with time the outbreak will eventually die out by itself since the majority of the current population will at one point be in the recovered group, where they cannot get the disease again.

Now we are ready to define the coupled differential equations that describe the transitions between the compartments. This is formulated as a function that takes the time points, time, the current state (i.e., the distribution of the individuals in the 4 compartments), and the set of parameters.

## Create a SIR function with an extra V component

sirv <- function(time, state, parameters) {

with(as.list(c(state, parameters)), {

dS <- -beta \* S \* I

dI <- beta \* S \* I - gamma \* I

dR <- gamma \* I

dV <- 0

return(list(c(dS, dI, dR, dV)))

})

}

Once we have the sirv() function we can use the ode() function from the deSolve package to solve the coupled differential equations.

library("deSolve")

## Solve the set of coupled differential equations

out <- ode(y = init,

times = times,

func = sirv,

parms = parameters

)

head(as.data.frame(out))

time S I R V

1 0.0 0.1449991 8.771930e-07 0.000000e+00 0.855

2 0.2 0.1449991 8.920417e-07 1.263739e-08 0.855

3 0.4 0.1449991 9.071419e-07 2.548870e-08 0.855

4 0.6 0.1449990 9.224976e-07 3.855756e-08 0.855

5 0.8 0.1449990 9.381132e-07 5.184763e-08 0.855

6 1.0 0.1449990 9.539932e-07 6.536268e-08 0.855

The value for the basic reproductive number, \(R0\), depends on the type of disease and typical numbers can be seen in the table below.33 It is not easy to provide good estimates of the reproductive number, \(R0\), since it is difficult to gauge how many individuals a person has been in contact with and since we hardly have a population containing *only* susceptible individuals. A large value of \(R0\) means that the disease is highly contagious.

⊕Tabel 1: R0 for common diseases that can be used with the SIR-model. The estimates for \(R0\) for the different diseases are obtained from [Wikipedia](https://en.wikipedia.org/wiki/Basic_reproduction_number) and from [these guidelines](https://www.hse.ie/eng/health/immunisation/hcpinfo/guidelines/chapter23.pdf).

| **Sygdom** | **R0** |
| --- | --- |
| Measels | 12-18 |
| Chickenpox | 10-12 |
| Smallpox | 5-7 |
| Rubella | 5-7 |
| Mumps | 4-7 |
| SARS | 2-5 |
| Flu (Spanish flu) | 2-3 |

Now we have the full functionality and we just need to wrap it in a shiny app to make it easily accessible to anyone and to allow non-R-users to try out different parameter values and see their consequences. One way to see the results is to plot the proportion of the population in each of the four compartments:

library("ggplot2")

library("tidyverse")

ldata <- as.data.frame(out) %>% gather(key, value, -time)

head(ldata)

time key value

1 0.0 S 0.1449991

2 0.2 S 0.1449991

3 0.4 S 0.1449991

4 0.6 S 0.1449990

5 0.8 S 0.1449990

6 1.0 S 0.1449990

ggplot(data=ldata,

aes(x = time,

y = value,

group = key,

col = key

)) +

ylab("Proportion of full population") + xlab("Time (days)") +

geom\_line(size = 2) +

scale\_colour\_manual(values = c("red", "green4", "black", "blue")) +

scale\_y\_continuous(labels = scales::percent, limits = c(0, 1))

The graph shows the result of the model and the distribution of individuals in the four groups over time. The interesting part is how large (and how quickly) a drop in the proportion of the susceptible that is observed. The blue line shows the actual proportion of vaccinated / previously immune individuals and it will be constant in this model. The new individuals in the recovered group will be added to the immune group by the time of the next outbreak.44 The reason for why the recovered and vaccinated groups are kept separate is that it might be interesting to add a societal cost from the current outbreak to end up in the recovered group so we need to be able to distinguish the two groups for this outbreak. The black curve shows how the susceptible individuals are moved to the infected (and subsequently to the recovered) group.

It is also possible to compute simple statistics that show the impact of the outbreak.

# Proportion of full population that got the

# disease by end of time frame

ldata %>% filter(time == max(time), key=="R") %>%

mutate(prop = round(100 \* value, 2))

time key value prop

1 200 R 0.1137451 11.37

Here, 11.37% of the population will have had the disease as part of this outbreak. We can also compute the proportion of the susceptible, that have had the disease at the end of the time frame.

# Proportion of susceptible that will get

# the disease by end of time frame

as.data.frame(out) %>% filter(row\_number() == n()) %>%

mutate(res = round(100\*(R + I) / (S + I + R), 2)) %>% pull(res)

[1] 81.84

**What does it take to prevent an epidemic?**

\(R0\) denotes the number of persons that an infected person will infect. If this number is less that 1 then the outbreak will die out by itself and if \(R0>1\) then an epidemic starts. Herd immunity is the percentage of vaccinated/immune that is necessary to keep the effective \(R0\) less than one from the initial outbreak. We can compute these numbers directly

# Proportion of population that need to be

# immune for herd immunity

round(100 \* (1 - 1 / (connum)), 2)

[1] 93.33

So 93.33% immunity if needed for herd  
immunity from day 1 when \(R0\)=15. The effective \(R0\) at day 1 is the \(R0\) from the  
disease scaled down according to the number of people already immune.

# Effective R0 (for populationen at outbreak,

# when prior immunity is taken into account)

round(connum \* (1 - pvac \* vaceff), 2)

[1] 2.18

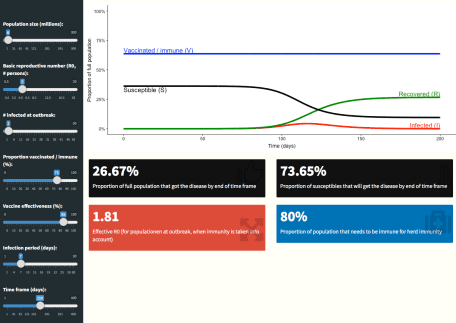
Even though the basic reproductive number here was $R0=$15 then the effective number of individuals that an infected person will infect is only 2.18 due to the number of immune and vaccinated individuals. However, since this number is larger than 1 it will still turn into an outbreak.

When evaluating the impact of vaccination it is necessary to take into account the frequencies, risks and costs of getting the disease compared to the costs and risks of vaccination. For example, roughly [1-2 out of 1000 persons infected with measles](https://www.cdc.gov/measles/parent-infographic.html) will **die** – even with the best treatment, and 1 out of 1000 persons infected with measles are under risk on swelling of the brain. If just 1% of the 5.7 million peopple in Denmark are infected with measles then that corresponds to around. 57000 persons. Consequently, 80 people are likely to die because of measles and 60 will be under risk of swelling of the brain.

It should be stressed that we have used ordinary differential equations for all of the computations. Thus, we have not taken the stochastic fluctuations that might occur in a real situation into account. The numbers that we see here should therefore be regarded as average effects whereas the effect in practice could be milder or worse than what we see here.

**Wrapping it in a Shiny app**

Now we have all the machinery in place and we need to wrap it inside a Shiny app. The final layout is shown below



#

# Shiny app for SIR model with vaccinations

#

#

library("shiny")

library("deSolve")

library("cowplot")

library("ggplot2")

library("tidyverse")

library("ggrepel")

library("shinydashboard")

## Create an SIR function

sir <- function(time, state, parameters) {

with(as.list(c(state, parameters)), {

dS <- -beta \* S \* I

dI <- beta \* S \* I - gamma \* I

dR <- gamma \* I

dV <- 0

return(list(c(dS, dI, dR, dV)))

})

}

#

# Define UI

#

ui <- dashboardPage(

dashboardHeader(disable = TRUE),

dashboardSidebar(

sliderInput("popsize",

"Population size (millions):",

min = 1, max = 300, value = 6

),

sliderInput("connum",

"Basic reproductive number (R0, # persons):",

min = .5, max = 20, value = 5

),

sliderInput("pinf",

"# infected at outbreak:",

min = 1, max = 50, value = 2

),

sliderInput("pvac",

"Proportion vaccinated / immune (%):",

min = 0, max = 100, value = 75

),

sliderInput("vaceff",

"Vaccine effectiveness (%):",

min = 0, max = 100, value = 85

),

sliderInput("infper",

"Infection period (days):",

min = 1, max = 30, value = 7

),

sliderInput("timeframe",

"Time frame (days):",

min = 1, max = 400, value = 200

)

),

dashboardBody(

tags$head(tags$style(HTML('

/\* body \*/

.content-wrapper, .right-side {

background-color: #fffff8;

}

'))),

# mainPanel(

fluidRow(plotOutput("distPlot")),

br(),

fluidRow(

# Dynamic valueBoxes

valueBoxOutput("progressBox", width = 6),

valueBoxOutput("approvalBox", width = 6),

valueBoxOutput("BRRBox", width = 6),

valueBoxOutput("HIBox", width = 6)

),

br(),

br()

)

)

#

# Define server

#

server <- function(input, output) {

# Create reactive input

dataInput <- reactive({

init <-

c(

S = 1 - input$pinf / (input$popsize\*1000000) - input$pvac / 100 \* input$vaceff / 100,

I = input$pinf / (input$popsize\*1000000),

R = 0,

V = input$pvac / 100 \* input$vaceff / 100

)

## beta: infection parameter; gamma: recovery parameter

parameters <-

c(beta = input$connum \* 1 / input$infper,

# \* (1 - input$pvac/100\*input$vaceff/100),

gamma = 1 / input$infper)

## Time frame

times <- seq(0, input$timeframe, by = .2)

## Solve using ode (General Solver for Ordinary Differential Equations)

out <- ode(

y = init,

times = times,

func = sir,

parms = parameters

)

# out

as.data.frame(out)

})

output$distPlot <- renderPlot({

out <-

dataInput() %>%

gather(key, value, -time) %>%

mutate(

id = row\_number(),

key2 = recode(

key,

S = "Susceptible (S)",

I = "Infected (I)",

R = "Recovered (R)",

V = "Vaccinated / immune (V)"

),

keyleft = recode(

key,

S = "Susceptible (S)",

I = "",

R = "",

V = "Vaccinated / immune (V)"

),

keyright = recode(

key,

S = "",

I = "Infected (I)",

R = "Recovered (R)",

V = ""

)

)

ggplot(data = out,

aes(

x = time,

y = value,

group = key2,

col = key2,

label = key2,

data\_id = id

)) + # ylim(0, 1) +

ylab("Proportion of full population") + xlab("Time (days)") +

geom\_line(size = 2) +

geom\_text\_repel(

data = subset(out, time == max(time)),

aes(label = keyright),

size = 6,

segment.size = 0.2,

segment.color = "grey50",

nudge\_x = 0,

hjust = 1,

direction = "y"

) +

geom\_text\_repel(

data = subset(out, time == min(time)),

aes(label = keyleft),

size = 6,

segment.size = 0.2,

segment.color = "grey50",

nudge\_x = 0,

hjust = 0,

direction = "y"

) +

theme(legend.position = "none") +

scale\_colour\_manual(values = c("red", "green4", "black", "blue")) +

scale\_y\_continuous(labels = scales::percent, limits = c(0, 1)) +

theme(

rect=element\_rect(size=0),

legend.position="none",

panel.background=element\_rect(fill="transparent", colour=NA),

plot.background=element\_rect(fill="transparent", colour=NA),

legend.key = element\_rect(fill = "transparent", colour = "transparent")

)

})

output$progressBox <- renderValueBox({

valueBox(

dataInput() %>% filter(time == max(time)) %>% select(R) %>% mutate(R = round(100 \* R, 2)) %>% paste0("%"),

"Proportion of full population that got the disease by end of time frame",

icon = icon("thumbs-up", lib = "glyphicon"),

color = "black"

)

})

output$approvalBox <- renderValueBox({

valueBox(

paste0(round(

100 \* (dataInput() %>% filter(row\_number() == n()) %>% mutate(res = (R + I) / (S + I + R)) %>% pull("res")), 2), "%"),

"Proportion of susceptibles that will get the disease by end of time frame",

icon = icon("thermometer-full"),

color = "black"

)

})

output$BRRBox <- renderValueBox({

valueBox(

paste0(round(input$connum \*

(1 - input$pvac / 100 \* input$vaceff / 100), 2), ""),

"Effective R0 (for populationen at outbreak, when immunity is taken into account)",

icon = icon("arrows-alt"),

color = "red"

)

})

output$HIBox <- renderValueBox({

valueBox(

paste0(round(100 \* (1 - 1 / (input$connum)), 2), "%"),

"Proportion of population that needs to be immune for herd immunity",

icon = icon("medkit"),

color = "blue"

)

})

}

# Run the application

shinyApp(ui = ui, server = server)