

Module 11: Lesson 1 Lab

Callum Arnold

21 July, 2021

Contents

Background	1
Preliminaries	1
Definitions	2
The threshold parameter R_0	2
The final size of an epidemic	2
The duration of the epidemic	2
Exercises	2
Exercise 1	2
SIR Markov	3
SIR Markov alt	3
SIR non-Markov constant	5
SIR non-Markov Gamma	6
Exercise 2	8
Exercise 3	9
SIR Markov	9
SIR Markov alt	11
SIR non-Markov constant	13
SIR non-Markov Gamma	16
Exercise 4	19
SIR Markov	19
SIR non-Markov constant	21
SIR non-Markov Gamma	22
Exercise 5	23
Exercise 6	24
Exercise 7	24

Background

Preliminaries

In the first lecture we introduced the concept of simulation by which we meant the procedure of *producing a realisation of the model*, or in other words a possible outcome. In this lab session we will be modifying existing R functions and by the end of this session, among other things, we will

- draw samples from the distributions of the *final size* and the *duration* of the epidemic (see below for definitions of these quantities);
- be able to simulate from an epidemic model where the infectious period follows a Weibull distribution;
- be able to simulate from a stochastic Susceptible-Exposed-Infective-Removed (SEIR) epidemic model.

Start by downloading the file `simulation.R` from here and save it in your workspace. This file `simulation.R` contains four different functions which will enable us to simulate realisations from various stochastic epidemic models:

- **simSIR.Markov**: The function produces realisations from a Markovian SIR model with infection and removal rate, β/N and γ respectively. The procedure (as described in Section 2 in the lecture) is the following:
 - i) first simulate the *time to the next event* and then
 - ii) decide the *type of the event* (infection or removal).
- **simSIR.Markov.alternative**: This function also produces realisations from a Markovian SIR model with infection and removal rate, β/N and γ respectively but with a slightly different algorithm to the one described above. The procedure here is as follows:
 - i) generate a possible time to the next infection and the possible time to the next removal; then
 - ii) the event which happens first determines the type of the next event.
- **simSIR.Non.Markov.constant** and **simSIR.Non.Markov.gamma**: These two functions allows us to simulate realisations from non-Markov SIR models; with constant and Gamma infectious period respectively. The procedure is similar to the one used in **simSIR.Markov.alternative** based on Section 5 in Lecture 1.

Definitions

In this section we give definitions of two quantities for which samples from their distribution will be drawn.

The threshold parameter R_0

An appealing feature of an epidemic model is that it embodies a threshold parameter which can be utilised as a severity measure of the outbreak. Stochastic models such as epidemics and branching processes typically generate bimodal realisations where an epidemic may or may not die out quickly, depending on the value of a threshold parameter R_0 often referred to as the basic reproduction number. This is the most important parameter in epidemic theory and it is defined as *the expected number of infections generated by a typical infective in an infinite susceptible population*. In the case of a homogeneously mixing stochastic epidemic, it holds that $R_0 = \beta \mathbb{E}[I]$ where $\mathbb{E}[I]$ denotes the expected infectious period. For instance, in the case of the Markovian SIR model, $R_0 = \beta/\gamma$, since the individuals remain infectious for an average period of length $1/\gamma$.

The final size of an epidemic

The final size of an epidemic is the total number of initial susceptibles who contracted the disease by the end of the outbreak.

The duration of the epidemic

Another quantity of interest is the duration of the epidemic and this is defined as the *time that elapsed from the initial infection until the time of the last removal*.

Exercises

Exercise 1

Go through the lines of each function and make sure that you follow the logic behind.

SIR Markov

```
simSIR.Markov <- function(N, beta, gamma) {  
  
  # initial number of infectives and susceptibles;  
  I <- 1  
  S <- N-1;  
  
  # recording time;  
  t <- 0;  
  times <- c(t);  
  
  # a vector which records the type of event (1=infection, 2=removal)  
  type <- c(1);  
  
  while (I > 0) {  
  
    # time to next event;  
    t <- t + rexp(1, (beta/N)*I*S + gamma*I);  
    times <- append(times, t);  
  
    if (runif(1) < beta*S/(beta*S + N*gamma)) {  
      # infection  
      I <- I+1;  
      S <- S-1;  
      type <- append(type, 1);  
    }  
    else {  
      #removal  
      I <- I-1  
      type <- append(type, 2);  
    }  
  }  
  
  # record the final size , i.e. the number of initially susceptibles who  
  # contracted the disease sometime during the epidemic.  
  #  
  #  
  #  
  
  # record the times of events (infections/removals) as well as the type  
  res <- list("t"=times, "type"=type);  
  res  
}
```

SIR Markov alt

```
simSIR.Markov.alternative <- function(N, beta, gamma) {  
  
  # initial number of infectives and susceptibles;  
  I <- 1  
  S <- N-1;
```

```

# recording time;
t <- 0;
times <- c(t);

# a vector which records the type of event (1=infection, 2=removal)
type <- c(1);

while (I > 0) {

#####
# simulate times to the next possible events
#####

# time to next infection
if (S > 0) {
  t.next.infection <- t + rexp(1, (beta/N)*I*S)
}
else {
  t.next.infection <- Inf;
}

# time to next removal
t.next.removal <- t + rexp(1, gamma*I)

# check which of the two events happens first
if (t.next.infection < t.next.removal) {
  # infection occurs
  I <- I+1;
  S <- S-1;
  type <- append(type, 1);
  times <- append(times, t.next.infection);
  t <- t.next.infection
}
else {
  #removal occurs
  I <- I-1
  times <- append(times, t.next.removal);
  type <- append(type, 2);
  t <- t.next.removal
}
}

# record the final size , i.e. the number of initially susceptibles who
# contracted the disease sometime during the epidemic.
#
#

# record the times of events (infections/removals) as well as the type
#
#
res <- list("t"=times, "type"=type);
res
}

```

SIR non-Markov constant

```
simSIR.Non.Markov.constant <- function(N, beta, k) {  
  
  # initial number of infectives and susceptibles;  
  I <- 1  
  S <- N-1;  
  
  # recording time;  
  t <- 0;  
  times <- c(t);  
  
  # create a vector containing the removal times of all the current infectives  
  r <- k  
  
  # a vector which records the type of event (1=infection, 2=removal)  
  type <- c(1);  
  
  # a counter for labelling the individuals  
  lambda <- 1;  
  
  # a vector to store the labels  
  labels <- c(1);  
  
  while (I > 0) {  
  
    #####  
    # simulate times to the next possible events  
    #####  
  
    # time to next infection  
    if (S > 0) {  
      T <- rexp(1, (beta/N)*I*S)  
    }  
    else {  
      T <- Inf;  
    }  
  
    # time to next removal  
    R <- min(r, na.rm=TRUE);  
  
    # check which of the two events happens first  
    if (t + T < R) {  
      # infection occurs  
      I <- I+1;  
      S <- S-1;  
      r <- append(r, t + T + k)  
      type <- append(type, 1);  
      times <- append(times, t + T);  
  
      lambda <- lambda + 1;  
      labels <- append(labels, lambda)  
      t <- t + T  
    }  
  }  
}
```

```

else {
  #removal occurs
  I <- I-1
  type <- append(type, 2);
  index.min.r <- which(min(r, na.rm=TRUE)==r)
  r[index.min.r] <- NA
  labels <- append(labels, index.min.r)
  times <- append(times, R);
  t <- R

  # update the vector of
}
}

# record the final size , i.e. the number of initially susceptibles who contracted the disease sometime
#
#

# record the times of events (infections/removals) as well as the type
#
#
res <- list("t"=times, "type"=type, "labels" = labels);
res
}

```

SIR non-Markov Gamma

```

simSIR.Non.Markov.gamma <- function(N, beta, gamma, delta) {

  # initial number of infectives and susceptibles;
  I <- 1
  S <- N-1;

  # recording time;
  t <- 0;
  times <- c(t);

  # create a vector containing the removal times of all the current infectives.
  k <- rgamma(1, gamma, delta)
  r <- k

  # a vector which records the type of event (1=infection, 2=removal)
  type <- c(1);

  # a counter for labelling the individuals
  lambda <- 1;

  # a vector to store the labels
  labels <- c(1);

  while (I > 0) {

    #####

```

```

# simulate times to the next possible events
#####

# time to next infection
if (S > 0) {
  T <- rexp(1, (beta/N)*I*S)
}
else {
  T <- Inf;
}

# time to next removal
R <- min(r, na.rm=TRUE);

# check which of the two events happens first
if (t + T < R) {
  # infection occurs
  I <- I+1;
  S <- S-1;
  k <- rgamma(1, gamma, delta)
  r <- append(r, t + T + k)

  lambda <- lambda + 1;
  labels <- append(labels, lambda)
  type <- append(type, 1);
  times <- append(times, t + T);
  t <- t + T
}
else {
  #removal occurs
  I <- I-1
  type <- append(type, 2);
  index.min.r <- which(min(r, na.rm=TRUE)==r)
  r[index.min.r] <- NA
  labels <- append(labels, index.min.r)
  times <- append(times, R);
  t <- R
}
}

# record the final size , i.e. the number of initially susceptibles who contracted the disease sometime
#
#

# record the times of events (infections/removals) as well as the type
#
#
res <- list("t"=times, "type"=type, "labels"=labels);
res
}

```

Exercise 2

Simulate realisations from a Markovian SIR using the function `simSIR.Markov` and make sure that you understand the output. You may assume that size of the population size is $N=21$ (i.e. 20 susceptibles and 1 initial infective). In addition, you could try different values for (β, γ) , e.g. $(0.9, 1)$, $(2, 1)$ and $(4, 1)$.

```
simSIR.Markov(N=21, beta = 0.9, gamma = 1)
```

```
## $t
## [1] 0.0000000 0.4108494
##
## $type
## [1] 1 2
```

```
library(purrr)
map2(
  .x = c(0.9, 2, 4),
  .y = c(1, 1, 1),
  .f = function(.x, .y){
    print(glue::glue("Beta = {.x}, Gamma = {.y}"))
    simSIR.Markov(N = 21, beta = .x, gamma = .y)
  }
)
```

```
## Beta = 0.9, Gamma = 1
## Beta = 2, Gamma = 1
## Beta = 4, Gamma = 1

## [[1]]
## [[1]]$t
## [1] 0.000000 1.576690 1.854074 3.491239 3.616663 4.001931
##
## [[1]]$type
## [1] 1 1 2 1 2 2
##
##
## [[2]]
## [[2]]$t
## [1] 0.00000000 0.04963427 0.39554197 0.40620545 0.67797842 0.75225774
## [7] 0.78308789 1.07379471 1.35515435 1.40323759 1.44783152 1.46976105
## [13] 1.51269905 1.58724185 1.63497240 1.73049709 1.80839420 1.98177915
## [19] 1.98184693 2.01117568 2.03039998 2.12833932 2.15555839 2.23027685
## [25] 2.29914157 2.46176777 2.76897859 2.85676806 2.92204267 2.97965896
## [31] 3.04305256 3.19384516 3.25769879 3.39381930 3.42548087 3.60342461
## [37] 4.50804640 5.50673847
##
## [[2]]$type
## [1] 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 2 1 2 2 2 2 2 2 2
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.3685143
##
## [[3]]$type
```



```
## [1] 1 2
```

Exercise 3

Modify the existing functions in `simulation.R` to record the *final size* and the *duration* of the epidemic as part of the functions' output.

SIR Markov

```
simSIR.Markov <- function(N, beta, gamma) {  
  
  # initial number of infectives and susceptibles;  
  I <- 1  
  S <- N-1;  
  
  # recording time;  
  t <- 0;  
  times <- c(t);  
  
  # a vector which records the type of event (1=infection, 2=removal)  
  type <- c(1);  
  
  while (I > 0) {  
  
    # time to next event;  
    t <- t + rexp(1, (beta/N)*I*S + gamma*I);  
    times <- append(times, t);  
  
    if (runif(1) < beta*S/(beta*S + N*gamma)) {  
      # infection  
      I <- I+1;  
      S <- S-1;  
      type <- append(type, 1);  
    }  
    else {  
      #removal  
      I <- I-1  
      type <- append(type, 2);  
    }  
  }  
  
  # record the final size , i.e. the number of initially susceptibles who  
  # contracted the disease sometime during the epidemic.  
  fin_size = sum(type == 1) - 1  
  duration = sum(t)  
  
  # record the times of events (infections/removals) as well as the type  
  #  
  #  
  res <- list(  
    "t" = times,  
    "type" = type,  
    "fin_size" = fin_size,  
    "duration" = duration  
  )  
}
```

```

    );
    res
  }

map2(
  .x = c(0.9, 2, 4),
  .y = c(1, 1, 1),
  .f = function(.x, .y){
    print(glue::glue("Beta = {.x}, Gamma = {.y}"))
    simSIR.Markov(N = 21, beta = .x, gamma = .y)
  }
)

## Beta = 0.9, Gamma = 1
## Beta = 2, Gamma = 1
## Beta = 4, Gamma = 1

## [[1]]
## [[1]]$t
## [1] 0.0000000 0.5203504 0.7289745 0.8974476 1.0003098 1.1307596 1.1364698
## [8] 1.1789895 1.3164688 1.3427939 1.4023596 1.4024002 1.4486334 1.4871425
## [15] 1.7986628 1.8108649 1.8277384 1.9245474 1.9371643 2.0141691 2.2547536
## [22] 2.2664327 2.4773961 2.5440287 2.5511692 2.8697302 2.9027064 3.0178179
## [29] 3.1166929 3.1236537 3.1408114 3.1923533 3.2474393 3.2579989 3.4813823
## [36] 3.7823947
##
## [[1]]$type
## [1] 1 1 1 1 1 1 2 1 1 1 1 1 2 2 1 2 2 1 2 1 1 2 2 2 2 2 1 1 2 2 2 2 1 2 2 2
##
## [[1]]$fin_size
## [1] 17
##
## [[1]]$duration
## [1] 3.782395
##
##
## [[2]]
## [[2]]$t
## [1] 0.000000 1.052645 1.758976 2.813210 3.355329 3.355374 3.501046 3.780013
## [9] 4.034143 4.072024 4.880634 4.895997 4.940857 4.941380 5.035757 5.040274
## [17] 5.045425 5.065696 5.311450 5.327109 5.468293 5.563947 5.669604 5.674912
## [25] 5.907557 6.008117 6.101621 6.383661 6.871402 7.000123 7.290030 7.662954
## [33] 8.840034 9.068278
##
## [[2]]$type
## [1] 1 1 2 1 1 2 1 2 1 1 1 1 1 1 2 2 1 2 1 2 1 2 1 2 2 2 2 2 2 2 1 2 1 2 2
##
## [[2]]$fin_size
## [1] 16
##
## [[2]]$duration
## [1] 9.068278
##
##
## [[3]]

```

```
## [[3]]$t
## [1] 0.0000000 0.1947303 0.2404458 0.3208013 0.4212387 0.5002690 0.5641201
## [8] 0.5808629 0.5830404 0.6232374 0.6957359 0.7581835 0.7630334 0.8117711
## [15] 0.8545760 0.8865498 0.9395313 0.9907267 1.0072777 1.0348162 1.0718357
## [22] 1.1492831 1.1775478 1.4107122 1.4921198 1.5162603 1.6389381 1.6647748
## [29] 1.7820130 1.8493429 1.8964181 2.0704935 2.0904142 2.1906466 2.3145982
## [36] 2.3243348 2.3834074 2.7103933 4.0006425 4.6442630
##
## [[3]]$type
## [1] 1 1 2 1 1 1 2 1 1 1 2 2 1 1 2 1 1 1 1 1 1 2 1 2 2 2 2 2 2 2 2 2
## [39] 2 2
##
## [[3]]$fin_size
## [1] 19
##
## [[3]]$duration
## [1] 4.644263
```

SIR Markov alt

```
simSIR.Markov.alternative <- function(N, beta, gamma) {

  # initial number of infectives and susceptibles;
  I <- 1
  S <- N-1;

  # recording time;
  t <- 0;
  times <- c(t);

  # a vector which records the type of event (1=infection, 2=removal)
  type <- c(1);

  while (I > 0) {

    #####
    # simulate times to the next possible events
    #####

    # time to next infection
    if (S > 0) {
      t.next.infection <- t + rexp(1, (beta/N)*I*S)
    }
    else {
      t.next.infection <- Inf;
    }

    # time to next removal
    t.next.removal <- t + rexp(1, gamma*I)

    # check which of the two events happens first
    if (t.next.infection < t.next.removal) {
      # infection occurs
      I <- I+1;
    }
  }
}
```

```

    S <- S-1;
    type <- append(type, 1);
    times <- append(times, t.next.infection);
    t <- t.next.infection
  }
  else {
    #removal occurs
    I <- I-1
    times <- append(times, t.next.removal);
    type <- append(type, 2);
    t <- t.next.removal
  }
}

# record the final size , i.e. the number of initially susceptibles who
# contracted the disease sometime during the epidemic.
fin_size = sum(type == 1) - 1
duration = sum(t)

# record the times of events (infections/removals) as well as the type
#
#
res <- list(
  "t" = times,
  "type" = type,
  "fin_size" = fin_size,
  "duration" = duration
);
res
}

```

```

map2(
  .x = c(0.9, 2, 4),
  .y = c(1, 1, 1),
  .f = function(.x, .y){
    print(glue::glue("Beta = {.x}, Gamma = {.y}"))
    simSIR.Markov.alternative(N = 21, beta = .x, gamma = .y)
  }
)

```

```

## Beta = 0.9, Gamma = 1
## Beta = 2, Gamma = 1
## Beta = 4, Gamma = 1

## [[1]]
## [[1]]$t
## [1] 0.0000000 0.6087923 0.9698033 0.9902248
##
## [[1]]$type
## [1] 1 1 2 2
##
## [[1]]$fin_size
## [1] 1
##
## [[1]]$duration

```

```
## [1] 0.9902248
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.6194445
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.6194445
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.3008080 0.4002812 0.4299250 1.2091689 1.9355255 2.3435313
## [8] 2.3934274 2.7087461 2.8078603 2.8103265 2.9062412 2.9217959 2.9240532
## [15] 2.9828442 3.1614984 3.2117944 3.2187872 3.2376919 3.2409120 3.2440296
## [22] 3.2721492 3.5041782 3.5845066 3.5878537 3.7089915 3.7387701 3.7515665
## [29] 3.8160814 3.9510413 4.0257179 4.0418038 4.0620198 4.1154973 4.2136380
## [36] 4.3075743 4.6113416 4.6183060 4.7244538 4.8498910 5.2223065 5.7320913
##
## [[3]]$type
## [1] 1 1 1 2 2 1 1 2 1 1 1 1 2 2 1 1 2 1 1 1 2 2 1 1 2 2 1 1 1 2 2 2 2 2
## [39] 2 2 2 2
##
## [[3]]$fin_size
## [1] 20
##
## [[3]]$duration
## [1] 5.732091
```

SIR non-Markov constant

```
simSIR.Non.Markov.constant <- function(N, beta, k) {

  # initial number of infectives and susceptibles;
  I <- 1
  S <- N-1;

  # recording time;
  t <- 0;
  times <- c(t);

  # create a vector containing the removal times of all the current infectives
  r <- k

  # a vector which records the type of event (1=infection, 2=removal)
  type <- c(1);
```

```

# a counter for labelling the individuals
lambda <- 1;

# a vector to store the labels
labels <- c(1);

while (I > 0) {

#####
# simulate times to the next possible events
#####

# time to next infection
if (S > 0) {
  T <- rexp(1, (beta/N)*I*S)
}
else {
  T <- Inf;
}

# time to next removal
R <- min(r, na.rm=TRUE);

# check which of the two events happens first
if (t + T < R) {
  # infection occurs
  I <- I+1;
  S <- S-1;
  r <- append(r, t + T + k)
  type <- append(type, 1);
  times <- append(times, t + T);

  lambda <- lambda + 1;
  labels <- append(labels, lambda)
  t <- t + T
}
else {
  #removal occurs
  I <- I-1
  type <- append(type, 2);
  index.min.r <- which(min(r, na.rm=TRUE)==r)
  r[index.min.r] <- NA
  labels <- append(labels, index.min.r)
  times <- append(times, R);
  t <- R

  # update the vector of
}
}

# record the final size , i.e. the number of initially susceptibles who
# contracted the disease sometime during the epidemic.
fin_size = sum(type == 1) - 1

```

```

duration = sum(t)

# record the times of events (infections/removals) as well as the type
#
#
res <- list(
  "t" = times,
  "type" = type,
  "fin_size" = fin_size,
  "duration" = duration
);
res
}

map(
  .x = c(0.9, 2, 4),
  .f = function(.x){
    print(glue::glue("Beta = {.x}"))
    simSIR.Non.Markov.constant(N = 21, beta = .x, k = 1)
  }
)

## Beta = 0.9
## Beta = 2
## Beta = 4

## [[1]]
## [[1]]$t
## [1] 0 1
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 1
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.4912412 0.5936591 0.9180667 1.0000000 1.1667022 1.3149190
## [8] 1.4739039 1.4912412 1.5936591 1.6712174 1.7992121 1.8370251 1.9180667
## [15] 1.9322643 1.9655278 2.1020639 2.1667022 2.1717137 2.3149190 2.3675786
## [22] 2.4739039 2.5590328 2.6712174 2.7022338 2.7992121 2.8370251 2.9322643
## [29] 2.9655278 3.1020639 3.1717137 3.3675786 3.5590328 3.7022338
##
## [[2]]$type
## [1] 1 1 1 1 2 1 1 1 2 2 1 1 1 2 1 1 1 2 1 2 1 2 1 2 2 2 2 2 2 2 2
##
## [[2]]$fin_size
## [1] 16
##
## [[2]]$duration

```

```
## [1] 3.702234
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.2045369 0.3874470 0.6728409 0.7293615 0.7719497 0.8106906
## [8] 0.8187663 0.8878815 0.9156871 0.9953932 1.0000000 1.0095170 1.0410671
## [15] 1.0430502 1.1232154 1.1281287 1.2045369 1.2237158 1.2589691 1.3874470
## [22] 1.3906319 1.5509849 1.6728409 1.7293615 1.7719497 1.8106906 1.8187663
## [29] 1.8878815 1.9156871 1.9953932 2.0095170 2.0410671 2.0430502 2.1232154
## [36] 2.1281287 2.2237158 2.2589691 2.3906319 2.5509849
##
## [[3]]$type
## [1] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2
##
## [[3]]$fin_size
## [1] 19
##
## [[3]]$duration
## [1] 2.550985
```

SIR non-Markov Gamma

```
simSIR.Non.Markov.gamma <- function(N, beta, gamma, delta) {

  # initial number of infectives and susceptibles;
  I <- 1
  S <- N-1;

  # recording time;
  t <- 0;
  times <- c(t);

  # create a vector containing the removal times of all the current infectives.
  k <- rgamma(1, gamma, delta)
  r <- k

  # a vector which records the type of event (1=infection, 2=removal)
  type <- c(1);

  # a counter for labelling the individuals
  lambda <- 1;

  # a vector to store the labels
  labels <- c(1);

  while (I > 0) {

    #####
    # simulate times to the next possible events
    #####

    # time to next infection
```



```

if (S > 0) {
  T <- rexp(1, (beta/N)*I*S)
}
else {
  T <- Inf;
}

# time to next removal
R <- min(r, na.rm=TRUE);

# check which of the two events happens first
if (t + T < R) {
  # infection occurs
  I <- I+1;
  S <- S-1;
  k <- rgamma(1, gamma, delta)
  r <- append(r, t + T + k)

  lambda <- lambda + 1;
  labels <- append(labels, lambda)
  type <- append(type, 1);
  times <- append(times, t + T);
  t <- t + T
}
else {
  #removal occurs
  I <- I-1
  type <- append(type, 2);
  index.min.r <- which(min(r, na.rm=TRUE)==r)
  r[index.min.r] <- NA
  labels <- append(labels, index.min.r)
  times <- append(times, R);
  t <- R
}
}

# record the final size , i.e. the number of initially susceptibles who
# contracted the disease sometime during the epidemic.
fin_size = sum(type == 1) - 1
duration = sum(t)

# record the times of events (infections/removals) as well as the type
#
#
res <- list(
  "t" = times,
  "type" = type,
  "fin_size" = fin_size,
  "duration" = duration
);
res
}

```

```

pmap(
  .l = list(
    beta = c(0.9, 2, 4),
    gamma = c(1, 1, 1),
    delta = c(1, 1, 1)
  ),
  .f = function(beta, gamma, delta){
    print(glue::glue("Beta = {beta}, Gamma = {gamma}, Delta = {delta}"))
    simSIR.Non.Markov.gamma(N = 21, beta = beta, gamma = gamma, delta = delta)
  }
)

## Beta = 0.9, Gamma = 1, Delta = 1
## Beta = 2, Gamma = 1, Delta = 1
## Beta = 4, Gamma = 1, Delta = 1

## [[1]]
## [[1]]$t
## [1] 0.000000 1.416205
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 1.416205
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.4182911 0.5197838 0.6033177 0.6283245 0.7200717 0.7662815
## [8] 0.8848257 0.9473238 0.9862121 1.0407569 1.1954537 1.2193320 1.2650826
## [15] 1.3005538 1.3153771 1.3338421 1.3552393 1.3818607 1.5056824 1.5386051
## [22] 1.6099339 1.6319517 1.6415823 1.7550168 1.7653795 1.8620657 1.9001488
## [29] 1.9170657 1.9532142 1.9623542 2.3473407 2.3615994 2.7190823 2.9436877
## [36] 2.9559842 4.1140507 8.0178438 8.1413773 8.4199845 8.8557039 9.1457019
##
## [[2]]$type
## [1] 1 1 1 1 1 2 1 1 2 1 1 2 1 2 1 2 1 1 1 1 1 1 2 2 2 1 2 2 2 2 2 2 2 2 1
## [39] 2 1 2 2
##
## [[2]]$fin_size
## [1] 20
##
## [[2]]$duration
## [1] 9.145702
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000000 0.0005253654 0.0336353964 0.2916104932 0.3741138272
## [6] 0.5099704022 0.5718715521 0.5897533584 0.6303139370 0.6566093641

```

```
## [11] 0.7832972821 0.7954173507 0.7965474883 0.8143036988 0.8289670504
## [16] 0.8882218437 0.9352640255 1.0776218070 1.0944703734 1.1050718945
## [21] 1.1136700238 1.1316342618 1.1705416454 1.2092808431 1.2751540240
## [26] 1.3036742976 1.3146516902 1.3461363595 1.3587868018 1.3657160008
## [31] 1.4350742996 1.7363021321 1.8716014431 2.1177088194 2.4103563436
## [36] 2.6618066607 2.6884738982 3.0947141647 3.1891617285 3.6877084163
##
## [[3]]$type
## [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 2 1 2 2 2 2 2 2 2 2 1 1 2 2 2
## [39] 2 2
##
## [[3]]$fin_size
## [1] 19
##
## [[3]]$duration
## [1] 3.687708
```

Exercise 4

Derive a simulation-based estimate of the distribution of the *final size* of a Markovian SIR model for different values of R_0 , e.g. $R_0=0.9$, $R_0=1.5$ and $R_0=4$. Furthermore, do the same for the non-Markovian models, e.g. for a constant and a Gamma infectious period. **Hint:** You may find it useful to write a loop which will iterate the following steps for a number of times:

1. Simulate a realisation from the epidemic model;
2. Store the final size

At the end you should have a collection of *final sizes* for which then you can plot a histogram as your estimate of the true distribution of the final size.

SIR Markov

```
library(tidyverse)
theme_set(theme_minimal())
```

```
SIR_final_size_reps <- function(reps, r0, N, beta, constant = FALSE, ..., model, combine_lists = FALSE){
```

```
  # Run a number of simulations
```

```
  list_of_tibbles <- pmap(
```

```
    .l = list(reps),
```

```
    .f = function(reps){
```

```
      # Create the table of final size values for each R0 (within a single
      # simulation)
```

```
      pmap_dfr(
```

```
        .l = list(r0 = r0),
```

```
        .f = function(r0){
```

```
          rep <- reps
```

```
          model <- model
```

```
          if (isFALSE(constant)){
```

```
            if (hasArg(delta)){
```

```
              gamma <- beta / r0
```

```
            }
```

```

gamma <- beta / r0

# Can't input model as an argument to map/functions in general, as it
# is a function itself, so instead need to use rlang::exec() to
# execute it!
model_output <- rlang::exec(
  "model",
  N = N, beta = beta, gamma = gamma, ...
)

# Create a dataframe of the important
tibble(
  id = rep,
  R_0 = r0,
  beta = beta,
  gamma = gamma,
  fin_size = model_output$fin_size
)

} else {
  k <- r0 / beta

  # Can't input model as an argument to map/functions in general, as it
  # is a function itself, so instead need to use rlang::exec() to
  # execute it!
  model_output <- rlang::exec(
    "model",
    N = N, beta = beta, k = k
  )

  # Create a dataframe of the important
  tibble(
    id = rep,
    R_0 = r0,
    beta = beta,
    k = k,
    fin_size = model_output$fin_size
  )
}

})

}))

if (combine_lists == TRUE){
  bind_rows(list_of_tibbles, .id = "id")
}

}

SIR_Markov_alt_reps <- SIR_final_size_reps(
  reps = 1:10,
  r0 = seq(1, 10, 0.1),
  N = 100,
  beta = 1,

```

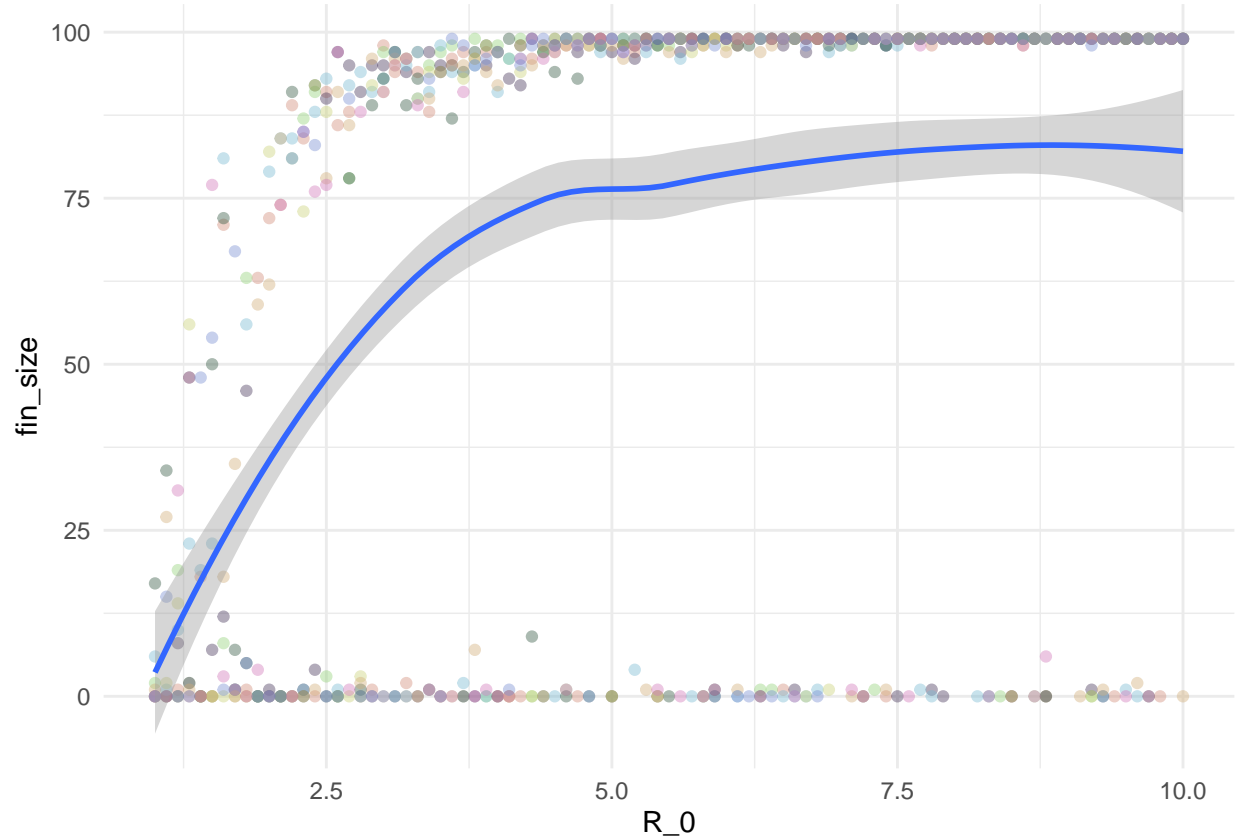
```

constant = FALSE,
model = simSIR.Markov.alternative,
combine_lists = TRUE
)

SIR_Markov_alt_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = fin_size)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()

## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## Warning: This manual palette can handle a maximum of 9 values. You have supplied
## 10.
## Warning: Removed 91 rows containing missing values (geom_point).

```



SIR non-Markov constant

```

SIR_non_Markov_const_reps <- SIR_final_size_reps(
  reps = 1:10,
  r0 = seq(1, 10, 0.1),
  N = 100,
  beta = 1,

```

```

constant = TRUE,
model = simSIR.Non.Markov.constant,
combine_lists = TRUE
)

```

```

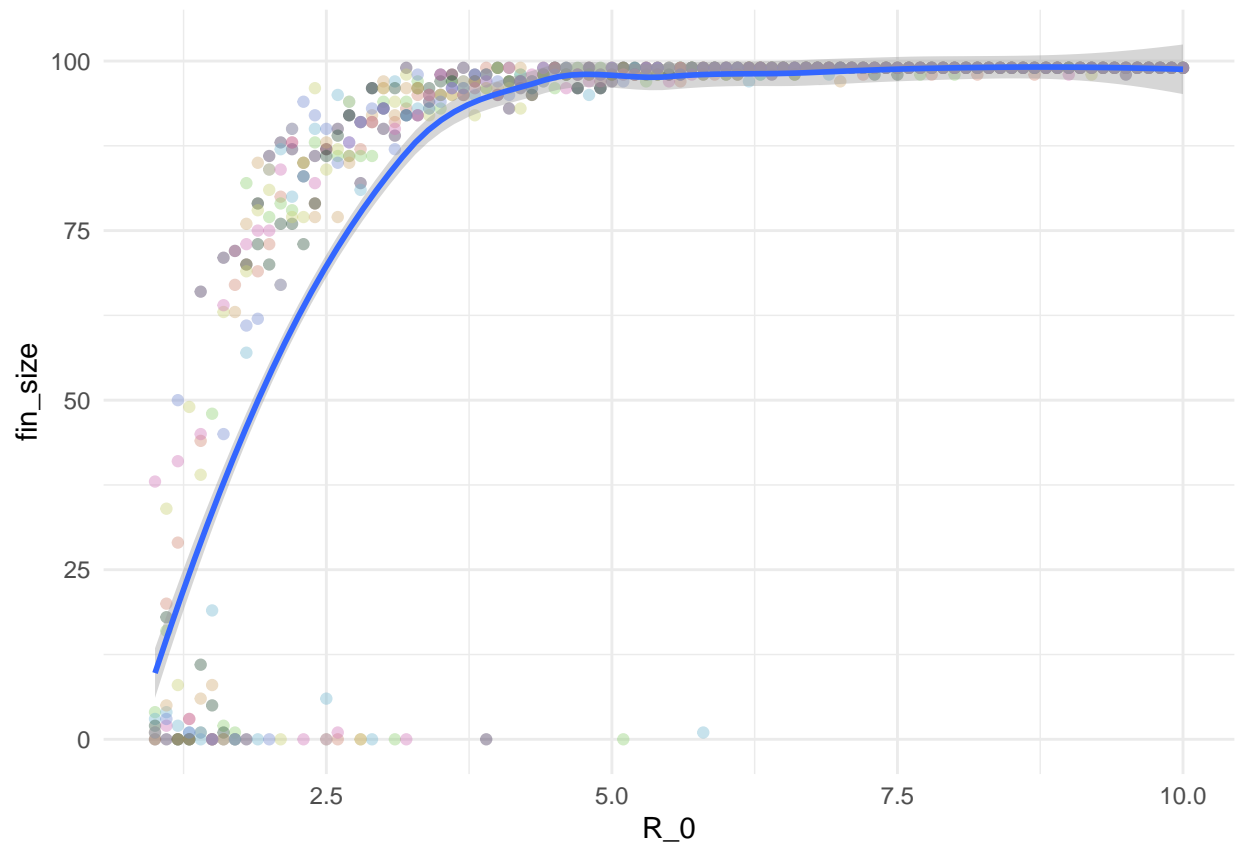
SIR_non_Markov_const_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = fin_size)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()

```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: This manual palette can handle a maximum of 9 values. You have supplied
## 10.
```

```
## Warning: Removed 91 rows containing missing values (geom_point).
```



SIR non-Markov Gamma

This isn't correct: need to figure out how to calculate R_0 when infection times include a draw from a Gamma distribution i.e. have the parameter δ

```

SIR_non_Markov_gamma_reps <- SIR_final_size_reps(
  reps = 1:10,
  r0 = seq(1, 10, 0.1),

```

```

N = 100,
beta = 5,
model = simSIR.Non.Markov.gamma,
delta = 0.1,
combine_lists = TRUE
)

```

```

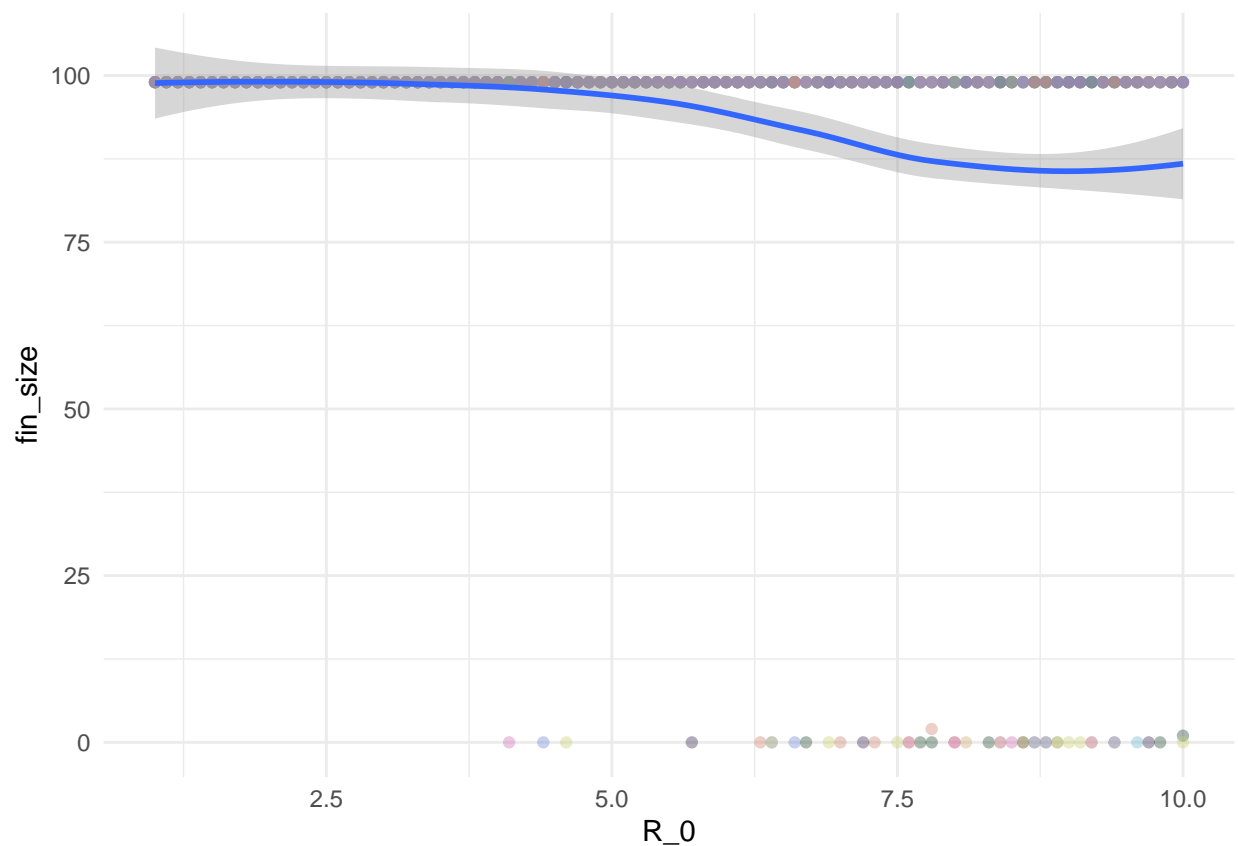
SIR_non_Markov_gamma_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = fin_size)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()

```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: This manual palette can handle a maximum of 9 values. You have supplied
## 10.
```

```
## Warning: Removed 91 rows containing missing values (geom_point).
```



Exercise 5

Repeat the above exercise but derive, by simulation, the distribution of the *duration* of the epidemic instead of the *final size*.

Exercise 6

Write a function in R to simulate from a non-Markovian stochastic epidemic model where the infectious period is assumed to follow a Weibull distribution. **Hint:** The probability density function (pdf) of the Weibull distribution is as follows:

$$f(x) = \frac{a}{b} \frac{x^{a-1}}{b^a} \exp(-(x/b)^a), x > 0, a > 0, b > 0$$

Type `?Weibull` to find out how to simulate from a Weibull distribution.

Exercise 7

Write a function to simulate from an epidemic model which involves a fixed latent period, i.e. write a function to simulate from a stochastic SEIR model.