Module 11: Lesson 1 Lab

Callum Arnold

21 July, 2021

Contents

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:
  - first simulate the *time to the next event* and then
  - 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:
  - generate a possible time to the next infection and the possible time to the next removal; then
  - 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 R0 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 $R0 = \beta\mathbb{E}[I]$ where $\mathbb{E}[I]$ denotes the expected infectious period. For instance, in the case of the Markovian SIR model, $R0 = \beta\gamma$, 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

Question

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

Answer

```
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 susceptlbles 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

```
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 susceptlbles 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.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 susceptlbles who contracted the disease someti
  #
  #

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

SIR non-Markov constant

```
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 susceptlbles who contracted the disease someti
  #
  #

  # 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

Exercise 2

Question

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 (,), e.g. (0.9,1),(2,1) and (4,1).

Answer

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

```
## $t
## [1] 0.0000000 0.8063889 1.1268750 1.1887420
##
## $type
## [1] 1 1 2 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]]$t
## [1] 0.0000000 0.9196016
##
## [[1]]$type
```

```
## [1] 1 2
##
##
## [[2]]
## [[2]]$t
##  [1] 0.0000000 0.3539650 0.4549538 1.1945812 1.2167705 1.3659420 1.4028599
##  [8] 1.5060622 1.6071951 2.3665071 2.8940054 3.5639557 3.6540498 3.6807410
## [15] 3.9632611 4.2463123 4.3168766 4.3815925 4.4178741 4.4361455 4.6082054
## [22] 4.9893798 5.0867642 5.1343105 5.3558814 5.8666928 6.0415904 6.1898722
## [29] 6.3855138 7.2461630 7.4252563 7.7120195 7.8566200 7.8683249
##
## [[2]]$type
##  [1] 1 1 2 1 2 1 1 2 2 1 2 1 1 1 2 1 2 1 1 2 2 1 2 2 1 1 2 2 2 1 2 1 2 2
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.2243704
##
## [[3]]$type
## [1] 1 2
```

Exercise 3

Question

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

Answer

```r
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)
  }
)
```

SIR Markov

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

```
## [[1]]
## [[1]]$t
##  [1] 0.000000 1.070915 1.276735 1.482837 1.736198 1.795845 1.804960 2.009630
##  [9] 2.283072 2.434989 2.475673 2.624158 2.748824 2.784520 2.842192 3.060604
## [17] 3.159904 3.447671 3.526777 3.803531 3.806537 3.876266 3.946743 4.152213
## [25] 4.326037 4.374219 5.041184 5.147725 5.472746 6.686057 6.707192 7.498523
##
## [[1]]$type
##  [1] 1 1 1 1 1 2 2 2 1 1 2 2 1 1 1 1 2 2 2 1 1 2 1 2 2 2 1 1 2 2 2 2
##
## [[1]]$fin_size
## [1] 15
##
```

```
## [[1]]$duration
## [1] 7.498523
##
##
## [[2]]
## [[2]]$t
##  [1] 0.0000000 0.1815892 0.3555965 0.6119124 0.6660195 0.9190978 1.0214671
##  [8] 1.0797868 1.1815665 1.2467171 1.3471950 1.3475363 1.3610890 1.4750710
## [15] 1.5526955 1.5531746 1.5704048 1.7996873 1.9970180 1.9979943 2.0269940
## [22] 2.0935014 2.1224964 2.3112216 2.3541999 2.8555323 2.8626017 3.1910357
## [29] 3.3912398 3.4782033 3.6572902 3.9366556 3.9879257 4.0981680 4.2648071
## [36] 4.2871980 4.3705795 6.1765608
##
## [[2]]$type
##  [1] 1 1 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 2 2 1 2 2 2 2 1 2 2 2 2 1 2 2 1 2 2 2 2
##
## [[2]]$fin_size
## [1] 18
##
## [[2]]$duration
## [1] 6.176561
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.0774626
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 0.0774626
```

```
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)
  }
)
```

SIR Markov alt

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

```
## [[1]]
## [[1]]$t
##  [1] 0.0000000 0.1098986 0.6154403 0.8914334 0.9062656 1.3695233 1.4693132
##  [8] 2.1875680 2.3421868 2.8554768
##
## [[1]]$type
##  [1] 1 1 2 1 1 2 2 1 2 2
##
## [[1]]$fin_size
## [1] 4
##
## [[1]]$duration
## [1] 2.855477
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.5173231 1.0663491 1.5134904
##
## [[2]]$type
## [1] 1 1 2 2
##
## [[2]]$fin_size
## [1] 1
##
## [[2]]$duration
## [1] 1.51349
##
##
## [[3]]
## [[3]]$t
##  [1] 0.00000000 0.03941333 0.22226472 0.55578328 0.58205460 0.76782828
##  [7] 0.77214806 0.77361383 0.85728735 0.88252877 0.89494462 0.93897681
## [13] 0.98403294 0.99591132 1.02580587 1.06436397 1.07758455 1.13683768
## [19] 1.25449560 1.26240418 1.41020208 1.75876696 1.77336117 1.81758749
## [25] 1.98216329 2.00801370 2.13927720 2.14800223 2.15641868 2.39862673
## [31] 2.86149768 2.89987219 2.95208115 3.21455568 3.21538792 3.25184803
## [37] 3.90297111 4.27493732 4.88770934 4.97852419 5.88802320 7.27762769
##
## [[3]]$type
##  [1] 1 1 2 1 1 1 1 1 2 1 1 1 2 2 2 2 1 2 2 1 2 1 1 1 1 1 1 2 2 2 2 1 1 1 2 2 2 2
## [39] 2 2 2 2
##
## [[3]]$fin_size
## [1] 20
```

```
##
## [[3]]$duration
## [1] 7.277628
```

```
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)
  }
)
```

SIR non-Markov constant

```
## 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]]$t
##  [1] 0.0000000 0.3333938 0.3338035 0.4168218 0.5730578 0.6060968 0.6705301
##  [8] 0.7109489 0.7974506 0.8981005 0.9970383 1.0000000 1.2919357 1.3270518
## [15] 1.3331582 1.3333938 1.3338035 1.3595117 1.4168218 1.5730578 1.6060968
## [22] 1.6705301 1.7109489 1.7974506 1.8981005 1.8987465 1.9970383 2.2919357
## [29] 2.3270518 2.3279226 2.3331582 2.3595117 2.6507342 2.8987465 3.3279226
## [36] 3.6507342
##
## [[2]]$type
##  [1] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2 2 2 2 2 2 2 1 2 2 2 1 2 2 1 2 2 2
##
## [[2]]$fin_size
## [1] 17
##
## [[2]]$duration
## [1] 3.650734
##
##
## [[3]]
## [[3]]$t
##  [1] 0.00000000 0.00979765 0.24617282 0.33860100 0.35034499 0.36636518
##  [7] 0.37105579 0.39355229 0.41541883 0.48379540 0.48986246 0.49199643
## [13] 0.55354634 0.75601502 0.78807413 0.80909971 0.81439557 0.85153734
## [19] 0.87903334 0.93460184 1.00000000 1.00979765 1.14841730 1.24617282
## [25] 1.33860100 1.35034499 1.36636518 1.37105579 1.39355229 1.41541883
## [31] 1.48379540 1.48986246 1.49199643 1.55354634 1.75601502 1.78807413
## [37] 1.80909971 1.81439557 1.85153734 1.87903334 1.93460184 2.14841730
##
## [[3]]$type
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2
##
## [[3]]$fin_size
## [1] 20
##
## [[3]]$duration
## [1] 2.148417
```

```
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)
  }
)
```

SIR non-Markov Gamma

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

```
## [[1]]
## [[1]]$t
## [1] 0.000000 1.508225 1.702805 2.861912 3.272522 4.026558
##
## [[1]]$type
## [1] 1 1 2 1 2 2
##
## [[1]]$fin_size
## [1] 2
##
## [[1]]$duration
## [1] 4.026558
```

```
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.5285462
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.5285462
##
##
## [[3]]
## [[3]]$t
##  [1] 0.00000000 0.06500064 0.12123139 0.33271222 0.33498466 0.40630578
##  [7] 0.48354598 0.49552085 0.53891163 0.63694895 0.70433156 0.72117087
## [13] 0.75925346 0.85806048 0.85838677 0.86874729 0.94092107 1.00986084
## [19] 1.01285487 1.05998930 1.10074428 1.11682997 1.14259753 1.25466621
## [25] 1.26120544 1.31291656 1.32306819 1.39759684 1.49021050 1.56621675
## [31] 1.79975242 2.02760203 2.27104655 2.34780648 2.39386523 2.41182271
## [37] 2.58045104 2.58289477 2.75678096 2.84057273 2.93744481 3.14603772
##
## [[3]]$type
##  [1] 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 1 1 1 1 1 2 2 1 1 1 2 2 1 2 2 2 1 2 2 2 2 2 2
## [39] 2 2 2 2
##
## [[3]]$fin_size
## [1] 20
##
## [[3]]$duration
## [1] 3.146038
```

Exercise 4

Question

Derive a simulation-based estimate of the distribution of the *final size* of a Markovian SIR model for different values of R0, e.g. R0=0.9, R0=1.5 and R0=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
3. 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.

Answer

Set up functions We can simulate once and plot the output, but let's write a function so that we can run as many simulations as we'd like, and plot all the values obtained. Exercise 5 asks us to rerun this process for duration instead of final size, so let's make sure we capture the duration as part of this function to avoid

unnecessary duplication.

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

```
SIR_model_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$fin_size,
              duration = model_output$duration
            )

          } 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$fin_size,
              duration = model_output$duration
            )
          }

        })
    })

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

}
```

```
SIR_Markov_alt_reps <- SIR_model_reps(
  reps = 1:10,
  r0 = seq(1, 10, 0.1),
  N = 100,
  beta = 1,
  constant = FALSE,
  model = simSIR.Markov.alternative,
  combine_lists = TRUE
)
```

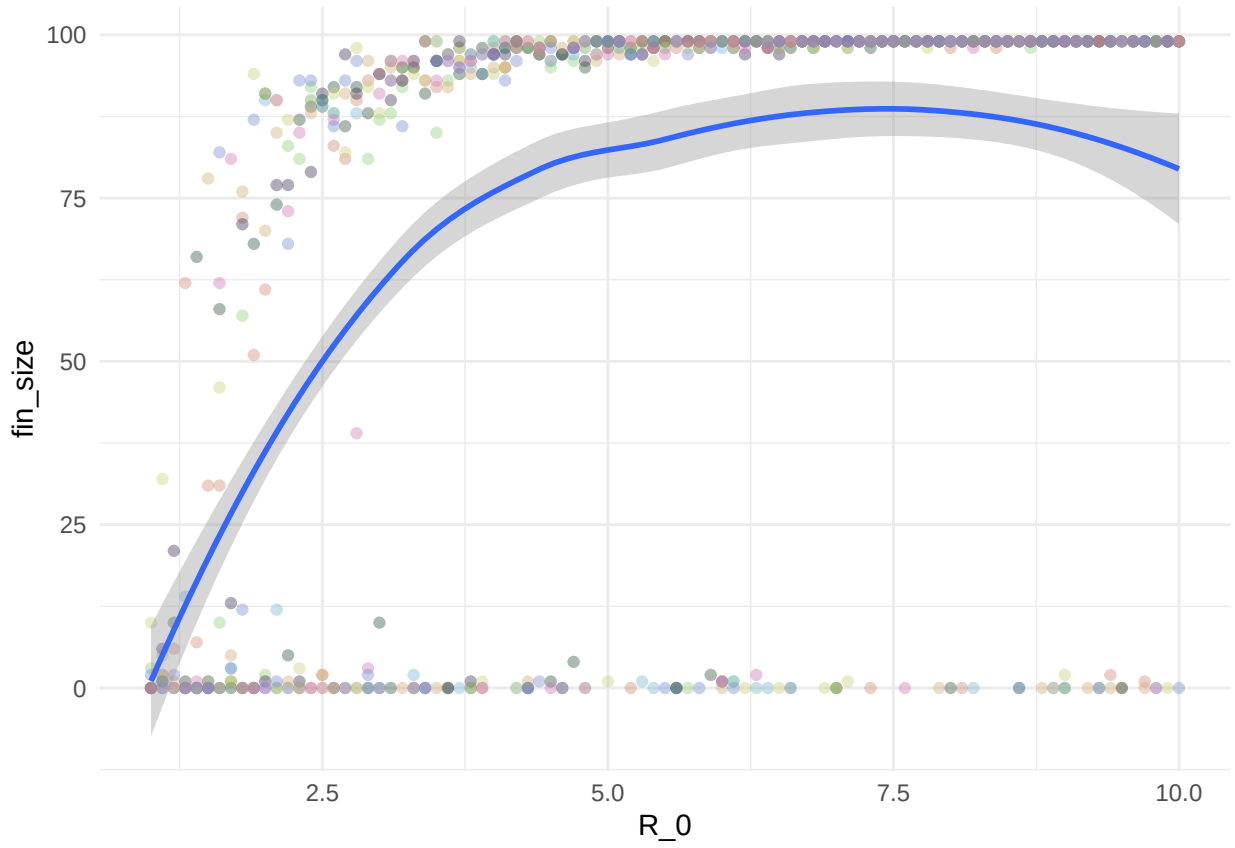
```
SIR_Markov_alt_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = fin_size)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()
```

SIR Markov alt

```
## `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_const_reps <- SIR_model_reps(
  reps = 1:10,
  r0 = seq(1, 10, 0.1),
  N = 100,
  beta = 1,
  constant = TRUE,
  model = simSIR.Non.Markov.constant,
  combine_lists = TRUE
)
```

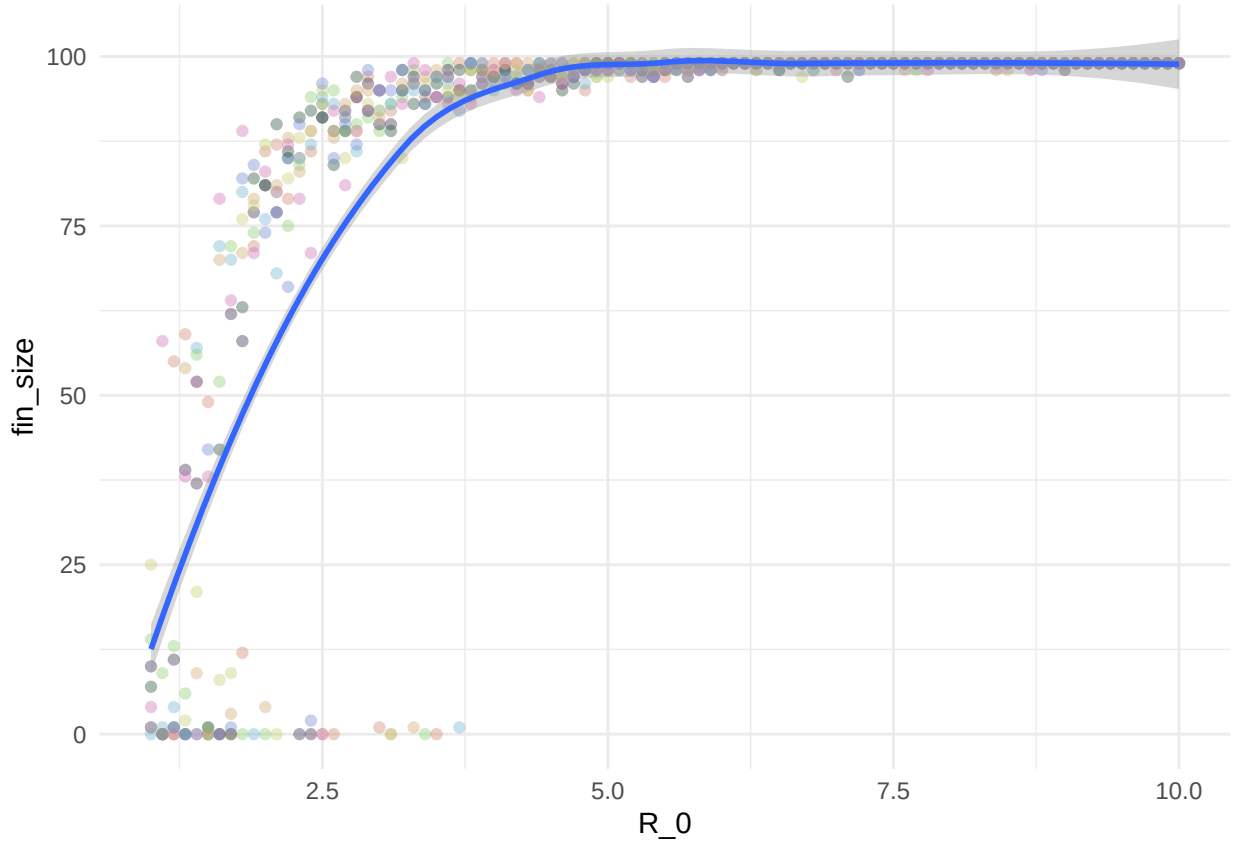
```
SIR_non_Markov_const_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = fin_size)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()
```

SIR non-Markov constant

```
## `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 `delta`

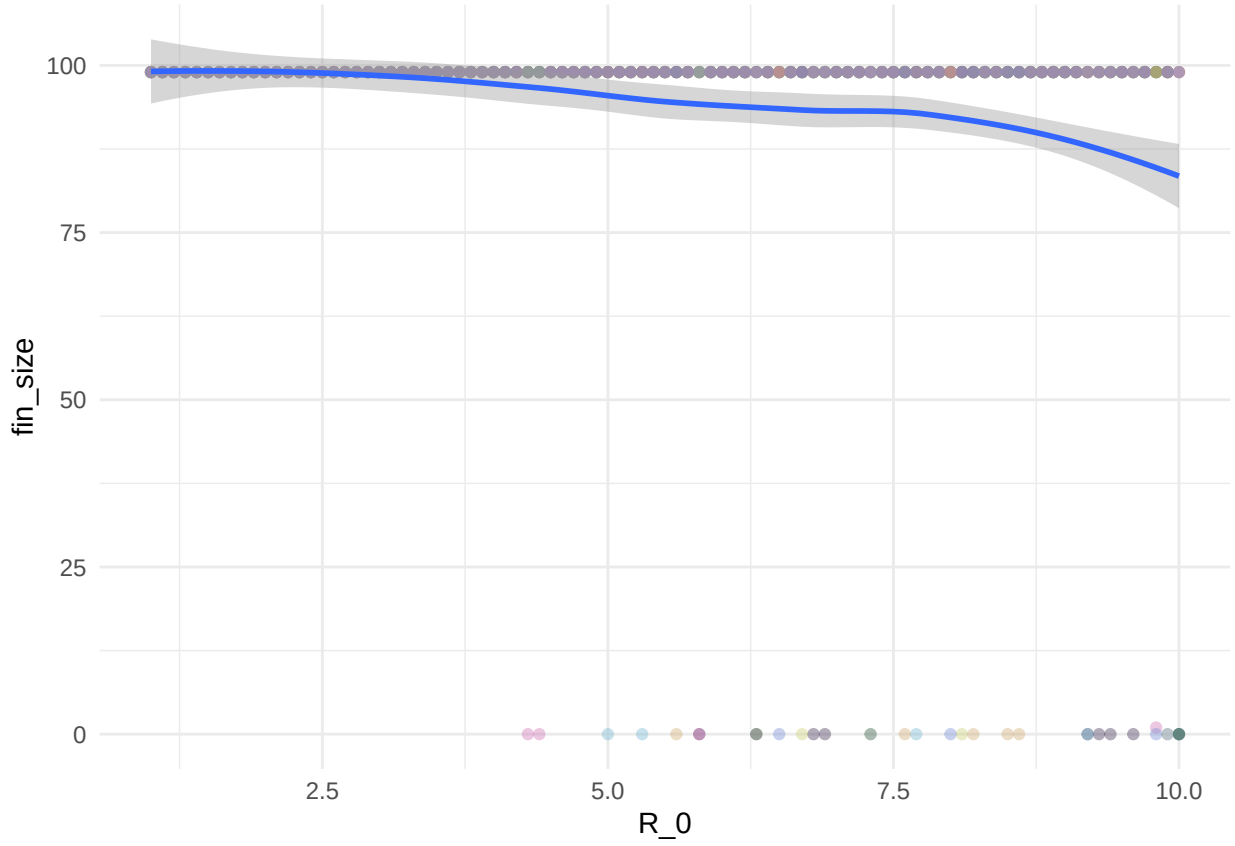
```
SIR_non_Markov_gamma_reps <- SIR_model_reps(
  reps = 1:10,
  r0 = seq(1, 10, 0.1),
  N = 100,
  beta = 5,
  model = simSIR.Non.Markov.gamma,
  delta = 0.1,
  combine_lists = TRUE
)
```

```
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

Question

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

Answer

As we set up our function in exercise 4 to capture both final size and duration, we don't need to calculate anything - just create the plots.

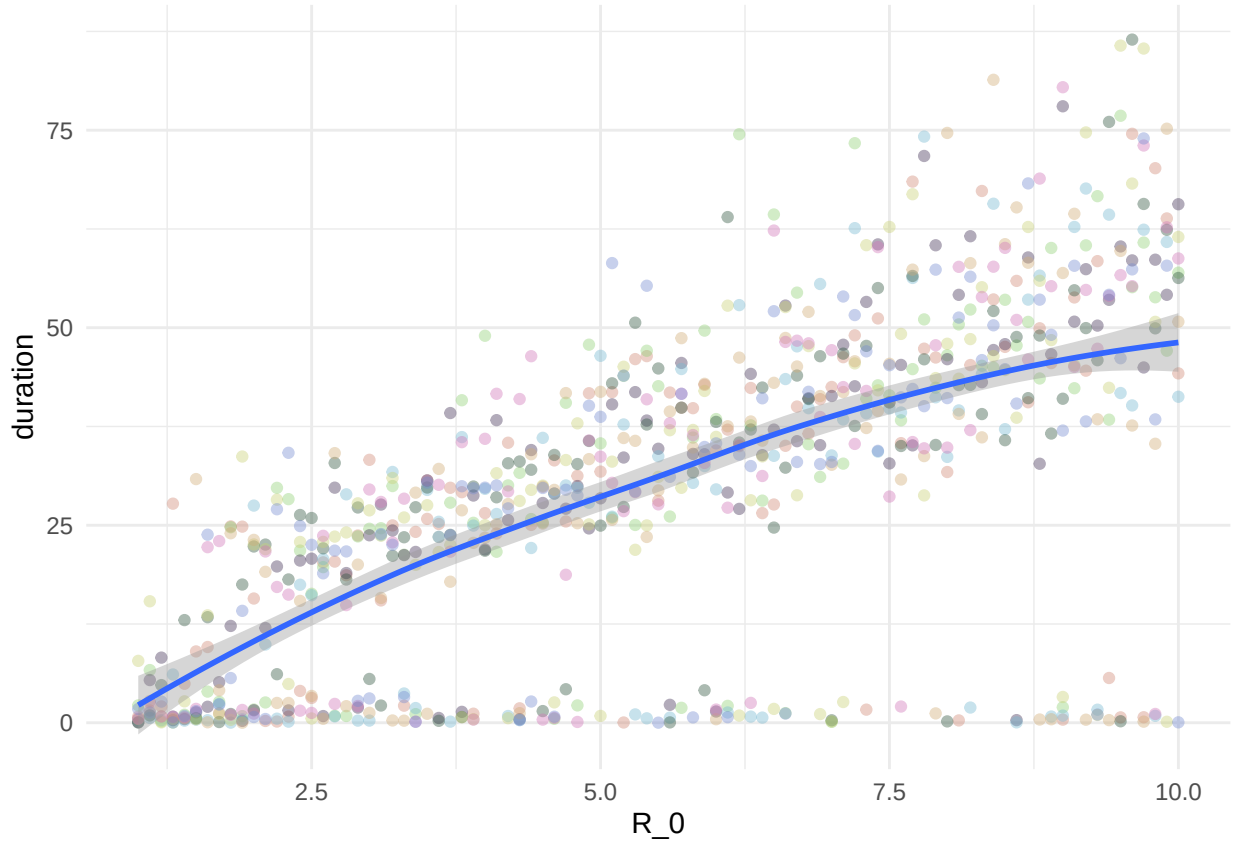
```
SIR_Markov_alt_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = duration)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()
```

SIR Markov alt

```
## `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).
```



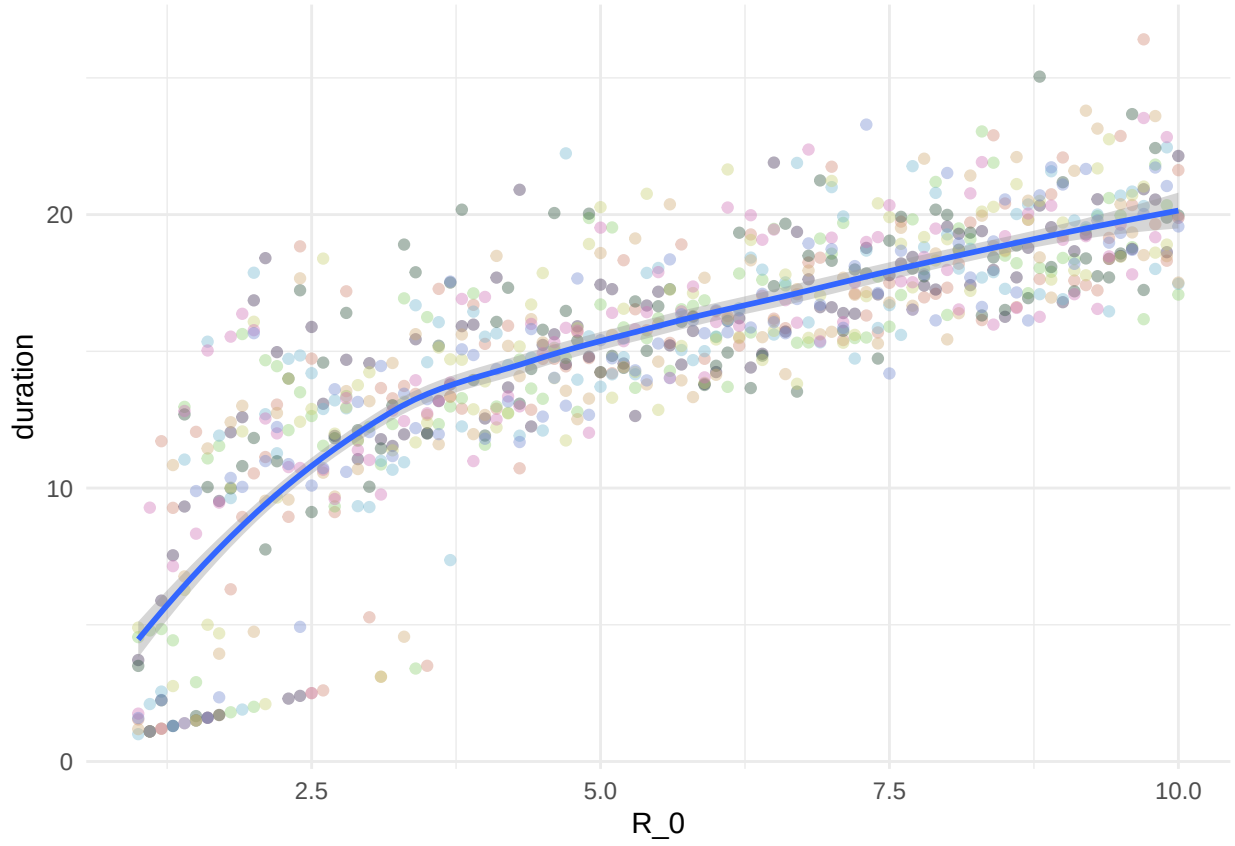

```
SIR_non_Markov_const_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = duration)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()
```

SIR non-Markov constant

```
## `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).
```



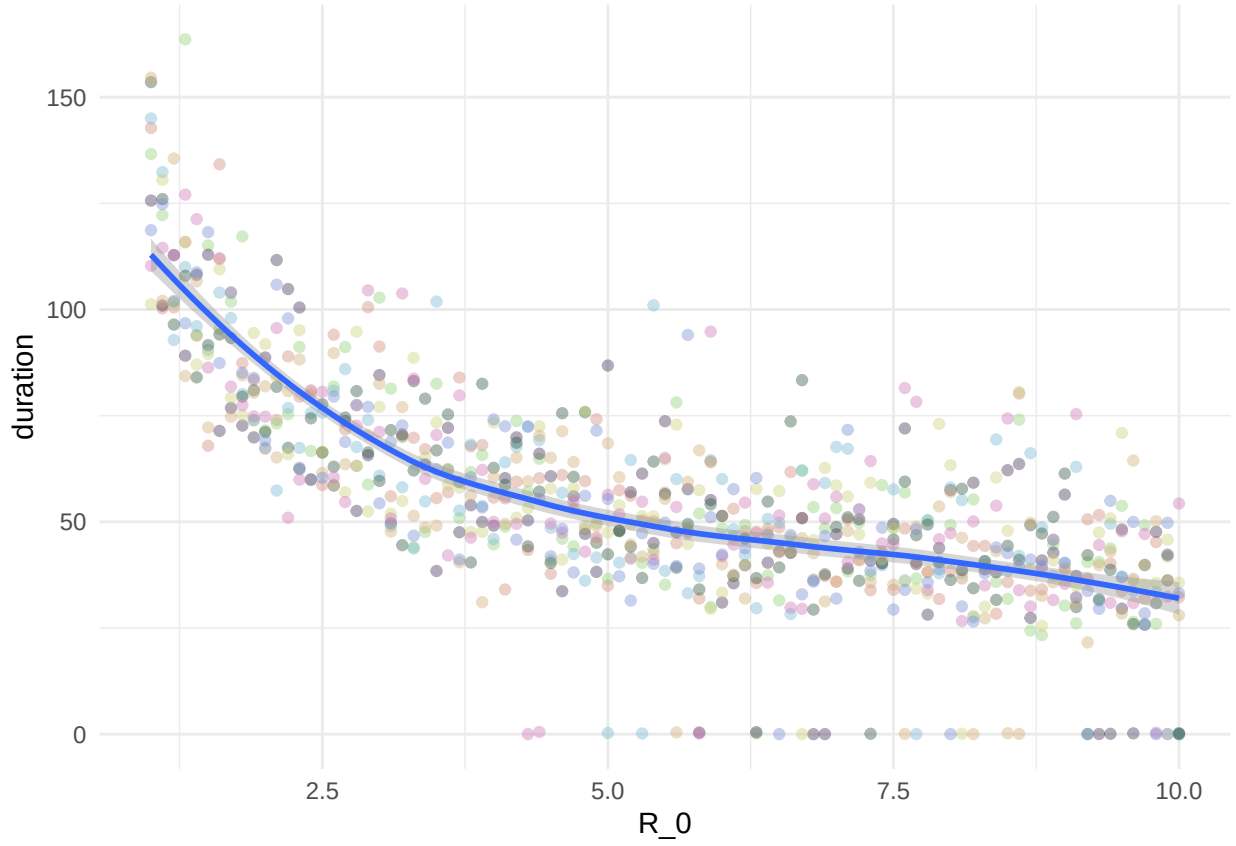

```
SIR_non_Markov_gamma_reps %>%
  mutate(id = as.factor(as.numeric(id))) %>%
  ggplot(aes(x = R_0, y = duration)) +
  geom_point(aes(color = id), alpha = 0.4) +
  geom_smooth() +
  theme(legend.position = "none") +
  hrbrthemes::scale_colour_ipsum()
```

SIR non-Markov Gamma

```
## `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 6

Question

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:

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

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

Answer

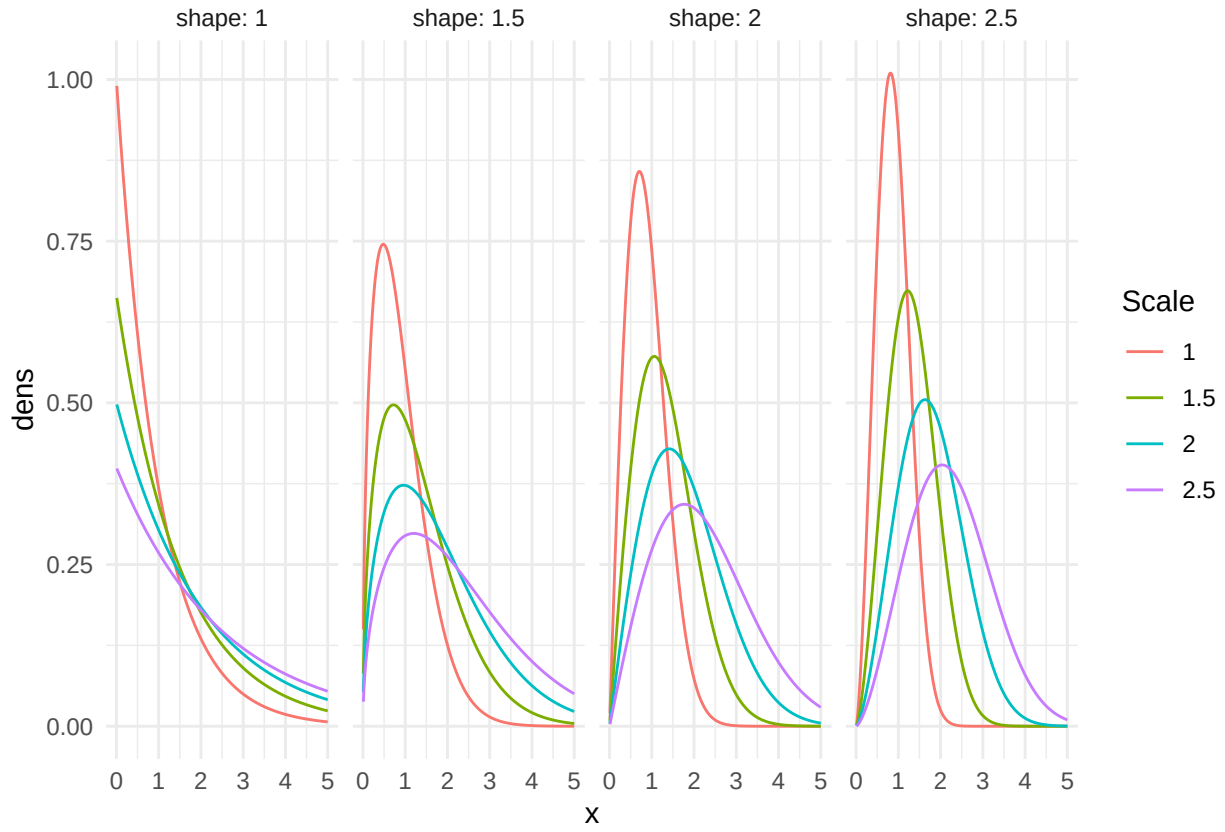
Let's first remind ourselves what a Weibull distribution looks like

```
cross_df(
  list(
    shape = seq(1, 2.5, 0.5),
    scale = seq(1, 2.5, 0.5)
  )) %>%
  pmap_df(
    .f = function(shape, scale){

      pmap_df(
        .l = list(x = seq(0.01, 5, 0.01)),
```

```
        .f = function(x){
         tibble(
           shape = shape,
           scale = scale,
           x = x,
           dens = dweibull(x, shape, scale)
          )
        }
      )

  }
) %>%
ggplot(aes(x, dens, color = factor(scale))) +
geom_line() +
facet_grid(.~shape, labeller = label_both) +
labs(color = "Scale")
```




```
simSIR.Non.Markov.weibull <- 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 <- rweibull(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 susceptlbles who contracted the disease someti
  #
  #

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

Exercise 7

Question

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.

Answer