Module 11: Lesson 1 Lab

Callum Arnold

7/19/2021

Contents

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 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.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 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 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

```
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 someti
  #
  #

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

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

```
## $t
## [1] 0.0000000 0.9218053
##
## $type
## [1] 1 2
```

```r
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]]$t
## [1] 0.00000000 0.03562905
##
## [[1]]$type
## [1] 1 2
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.8519231
##
## [[2]]$type
## [1] 1 2
##
##
## [[3]]
## [[3]]$t
## [1] 0.00000000 0.02879908 0.18170781 1.10834107
##
## [[3]]$type
## [1] 1 1 2 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

```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)
  }
)
```

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

```
## [[1]]
## [[1]]$t
```

```
## [1] 0.0000000 0.3071778 0.3619325 0.6589776
##
## [[1]]$type
## [1] 1 1 2 2
##
## [[1]]$fin_size
## [1] 1
##
## [[1]]$duration
## [1] 0.6589776
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.2852330 0.4310578 0.5321694
##
## [[2]]$type
## [1] 1 1 2 2
##
## [[2]]$fin_size
## [1] 1
##
## [[2]]$duration
## [1] 0.5321694
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.3602461
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 0.3602461
```

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
}
```

```r
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]]$t
## [1] 0.0000000 0.3216997
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 0.3216997
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.1418588 0.3494013 0.4112981 0.6508129 0.6953594 0.7383631
## [8] 1.1471204
##
## [[2]]$type
## [1] 1 1 2 1 1 2 2 2
##
## [[2]]$fin_size
## [1] 3
##
## [[2]]$duration
## [1] 1.14712
##
##
## [[3]]
## [[3]]$t
## [1] 0.00000000 0.09355771
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 0.09355771
```

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]]$t
##  [1] 0.0000000 0.3104709 0.4429462 0.5558051 1.0000000 1.1195382 1.3104709
##  [8] 1.4389093 1.4429462 1.5558051 2.0963350 2.1195382 2.4389093 3.0963350
##
## [[1]]$type
##  [1] 1 1 1 1 2 1 2 1 2 2 1 2 2 2
##
## [[1]]$fin_size
## [1] 6
##
## [[1]]$duration
```

```
## [1] 3.096335
##
##
## [[2]]
## [[2]]$t
##  [1] 0.000000000 0.000940683 0.343607761 0.501369904 0.595134066 0.886328088
##  [7] 1.000000000 1.000940683 1.129994763 1.165303462 1.205933980 1.343607761
## [13] 1.416117986 1.501369904 1.533669953 1.595134066 1.739333819 1.789501270
## [19] 1.886328088 2.036873317 2.062873059 2.084992381 2.129994763 2.163765637
## [25] 2.165303462 2.205933980 2.401094745 2.416117986 2.533669953 2.739333819
## [31] 2.789501270 2.941688633 3.036873317 3.062873059 3.084992381 3.163765637
## [37] 3.401094745 3.941688633
##
## [[2]]$type
##  [1] 1 1 1 1 1 1 2 2 1 1 1 2 1 2 1 2 1 1 2 1 1 1 2 1 2 2 1 2 2 2 2 1 2 2 2 2 2 2
##
## [[2]]$fin_size
## [1] 18
##
## [[2]]$duration
## [1] 3.941689
##
##
## [[3]]
## [[3]]$t
##  [1] 0.0000000 0.7697254 0.8313263 0.9687958 1.0000000 1.0185548 1.0418781
##  [8] 1.0867795 1.1961851 1.2099351 1.2847982 1.3303147 1.3959691 1.4280721
## [15] 1.4289360 1.6054222 1.6413854 1.6852272 1.6932165 1.7697254 1.8313263
## [22] 1.9226784 1.9687958 1.9735873 2.0185548 2.0418781 2.0867795 2.1961851
## [29] 2.2099351 2.2847982 2.3303147 2.3959691 2.4280721 2.4289360 2.6054222
## [36] 2.6413854 2.6852272 2.6932165 2.9226784 2.9735873
##
## [[3]]$type
##  [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [39] 2 2
##
## [[3]]$fin_size
## [1] 19
##
## [[3]]$duration
## [1] 2.973587
```

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]]$t
## [1] 0.00000000 0.08501393
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 0.08501393
##
##
## [[2]]
## [[2]]$t
##  [1] 0.0000000 0.1371352 0.3206618 0.5679104 0.6830039 0.7813545 0.8357477
##  [8] 1.0371659 1.1125317 1.1724005 1.3315782 1.4757459 1.7535509 1.9030776
## [15] 2.1898297 2.2345545 2.2760838 2.2887180 2.3567192 2.4990449 2.5261065
```

```
## [22] 2.5629638 2.6421902 2.6988165 2.7169132 2.7325309 2.7494072 2.8874681
## [29] 2.8996825 2.9811898 3.0696235 3.1587920 3.1758942 3.2934252 3.4660598
## [36] 3.5907499 3.6605012 3.6845525 3.8596179 4.0364570
##
## [[2]]$type
##  [1] 1 1 1 1 2 1 1 1 2 1 2 2 1 1 1 2 1 1 1 2 2 1 1 1 1 2 1 2 2 2 2 2 1 2 2 2 2 2
## [39] 2 2
##
## [[2]]$fin_size
## [1] 19
##
## [[2]]$duration
## [1] 4.036457
##
##
## [[3]]
## [[3]]$t
##  [1] 0.00000000 0.02925213 0.04571428 0.13722158 0.16985610 0.19196097
##  [7] 0.21019728 0.24030045 0.32613946 0.43732612 0.47262253 0.52659553
## [13] 0.55316047 0.55526771 0.58275710 0.59070431 0.60121756 0.65518136
## [19] 0.75673893 0.79072543 0.80908147 0.82520665 0.86029918 0.90986412
## [25] 0.91114280 0.93368792 1.07219610 1.08995505 1.11990504 1.12994105
## [31] 1.19317028 1.56918051 1.64636743 1.75631378 1.79976401 1.87898537
## [37] 1.93336774 1.98098174 2.31383241 2.33932467 2.67095971 5.00207016
##
## [[3]]$type
##  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 1 2 2 1 2 2 1 2 1 2 2 2 2 2 2 2 2 2
## [39] 2 2 2 2
##
## [[3]]$fin_size
## [1] 20
##
## [[3]]$duration
## [1] 5.00207
```

Exercise 4

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

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

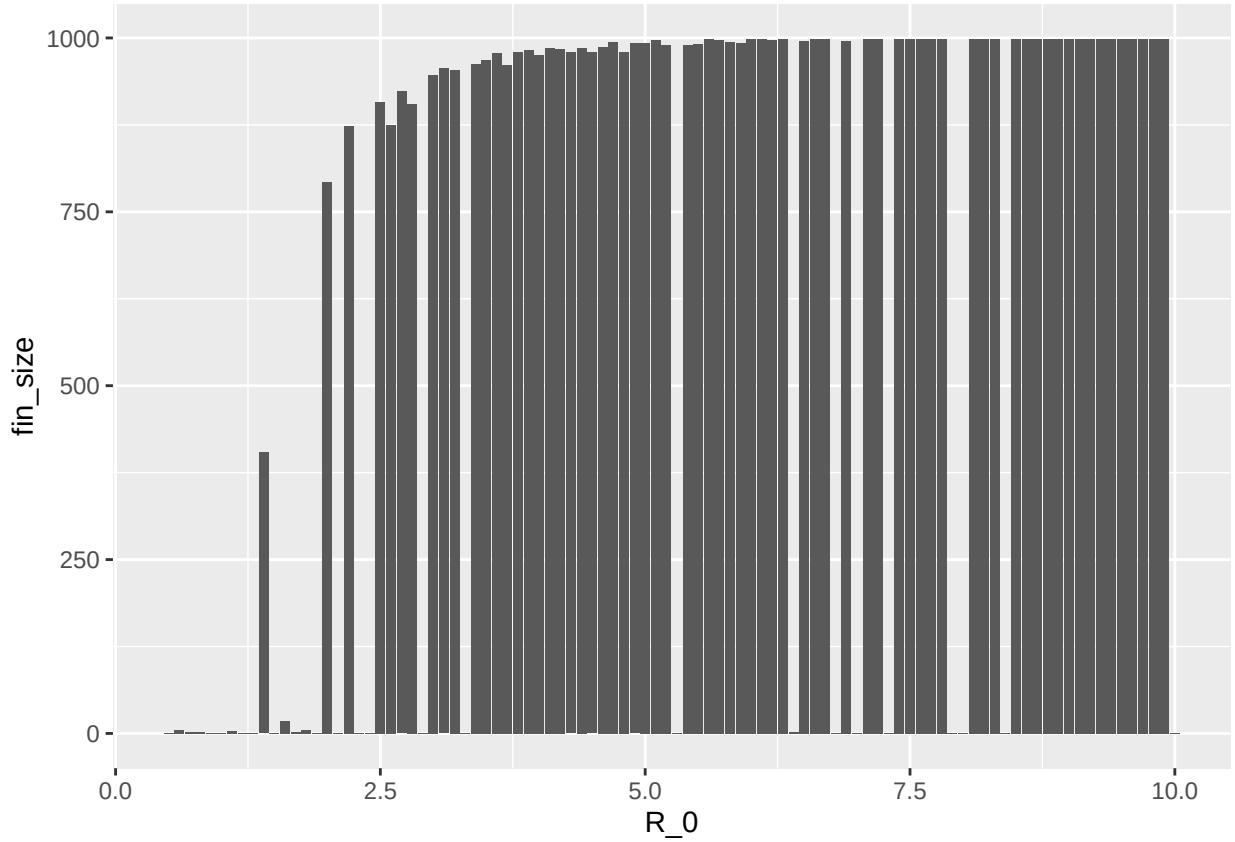
```
library(ggplot2)
```

```
test <- map_df(
  .x = seq(0.5, 10, by = 0.1),
```

```
  .f = function(.x){
    gamma <- 1 / .x
    N <- 1000
    fin_size <- simSIR.Markov.alternative(
        N = N, beta = 1, gamma = gamma
        )$fin_size
    fin_prop = fin_size / N

    return(data.frame(
      R_0 = .x,
      N = N,
      fin_size = simSIR.Markov.alternative(
        N = 1000, beta = 1, gamma = gamma
        )$fin_size,
      fin_prop = fin_prop
    ))
  }
)

ggplot(test, aes(x = R_0, y = fin_size)) +
  geom_bar(stat = "identity")
```



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:

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

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

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.