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] 0.00000000 0.08697621 0.10579427 0.37179637 0.38802261 0.51587110 0.88690279
## [8] 1.16020347
##
## $type
```

```
## [1] 1 1 2 1 1 2 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]]$t
##  [1] 0.0000000 0.1818400 0.3308484 0.4362792 0.5916895 0.6662371 0.6707047
##  [8] 0.9485065 1.0508407 1.1764378 1.3163988 1.6190625 1.6235813 1.8890227
## [15] 2.8102770 3.0299530 3.9076077 4.2953817 4.4655506 5.2969704 5.3985067
## [22] 6.8562759
##
## [[1]]$type
##  [1] 1 1 1 1 1 1 2 2 1 2 2 2 1 2 2 1 1 1 2 2 2 2
##
##
## [[2]]
## [[2]]$t
##  [1] 0.0000000 0.2375322 0.2416139 0.4467502 0.4607004 0.5816184 0.7507140
##  [8] 1.0744142 1.6519851 1.8987841 2.0308412 2.0465496 2.4438812 3.5770719
## [15] 3.6034071 4.5308510 4.5650748 4.6420037 4.7046675 4.9319559 5.7998100
## [22] 7.0189431
##
## [[2]]$type
##  [1] 1 1 2 1 1 1 2 2 1 1 2 2 2 1 2 1 1 1 2 2 2 2
##
##
## [[3]]
## [[3]]$t
##  [1] 0.0000000 0.0111027 0.1714496 0.2644277 0.3108499 0.3683509 0.4047614
##  [8] 0.5092617 0.5265813 0.5485927 0.5877426 0.6720203 0.7402149 0.7875260
## [15] 0.8681136 0.9029319 0.9304495 0.9953701 1.0677790 1.2010213 1.2717459
## [22] 1.3455708 1.4231631 1.4910150 1.6807629 1.8488663 1.9419984 2.0183928
## [29] 2.0614779 2.1840255 2.3197748 2.5457592 2.8552647 2.9785852 3.0945900
## [36] 3.2311738 3.2615772 5.1568389
##
## [[3]]$type
##  [1] 1 1 1 1 1 1 2 2 2 2 1 1 1 2 1 1 1 1 2 1 1 2 2 2 2 1 2 1 1 1 2 2 2 2 2 2 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

```
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]]$t
## [1] 0.0000000 0.1293153
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 0.1293153
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.5075077
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.5075077
##
##
## [[3]]
## [[3]]$t
##  [1] 0.0000000 0.2975222 0.3214975 0.3372744 0.3506775 0.4642021 0.4725561
##  [8] 0.5637513 0.6271180 0.6600227 0.6693584 0.6855022 0.6931189 0.7147626
## [15] 0.7434401 1.0422677 1.0816525 1.0963983 1.1095201 1.1963991 1.2110731
## [22] 1.2232821 1.2269242 1.2279576 1.3845657 1.5479827 1.6306986 1.7629290
## [29] 1.9659379 2.2286915 2.4705342 2.4803180 2.7894426 2.8836481 3.2582716
## [36] 3.6974431 4.1409431 5.7450648
##
## [[3]]$type
##  [1] 1 1 1 1 2 2 1 1 1 1 2 2 2 2 1 1 1 2 1 2 1 2 2 1 1 1 2 2 2 2 1 1 2 2 1 2 2 2
##
## [[3]]$fin_size
## [1] 18
##
## [[3]]$duration
```

```
## [1] 5.745065
```

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]]$t
## [1] 0.0000000 0.8737453
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 0.8737453
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.3312462
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.3312462
```

```
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.4217254
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 0.4217254
```

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]]$t
##  [1] 0.0000000 0.5657469 0.9418995 1.0000000 1.1141397 1.2155015 1.3272238
##  [8] 1.4860250 1.5657469 1.7257782 1.8095648 1.8277163 1.8331844 1.9418995
## [15] 1.9857132 2.1081613 2.1141397 2.2121838 2.2155015 2.2874188 2.3272238
## [22] 2.4860250 2.7257782 2.7339106 2.8095648 2.8277163 2.8331844 2.9615026
## [29] 2.9857132 3.1081613 3.2121838 3.2874188 3.7339106 3.9615026
##
## [[2]]$type
##  [1] 1 1 1 2 1 1 1 1 2 1 1 1 1 2 1 1 2 1 2 1 2 2 2 1 2 2 2 1 2 2 2 2 2 2
##
## [[2]]$fin_size
## [1] 16
##
## [[2]]$duration
## [1] 3.961503
##
##
## [[3]]
## [[3]]$t
## [1] 0 1
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 1
```

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]]$t
##  [1] 0.000000 1.198901 1.626200 1.869265 2.514323 2.588635 2.612117 2.798913
##  [9] 2.872842 3.082408 3.375285 3.377022 4.086184 4.134096 4.228905 4.229776
## [17] 4.406769 4.430478 5.560920 6.795280
##
## [[1]]$type
##  [1] 1 1 1 2 1 1 1 1 2 2 2 2 1 1 2 1 2 2 2 2
##
## [[1]]$fin_size
## [1] 9
##
```

```
## [[1]]$duration
## [1] 6.79528
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.6445293 1.0280877 1.2957421
##
## [[2]]$type
## [1] 1 1 2 2
##
## [[2]]$fin_size
## [1] 1
##
## [[2]]$duration
## [1] 1.295742
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.1903319
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 0.1903319
```

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

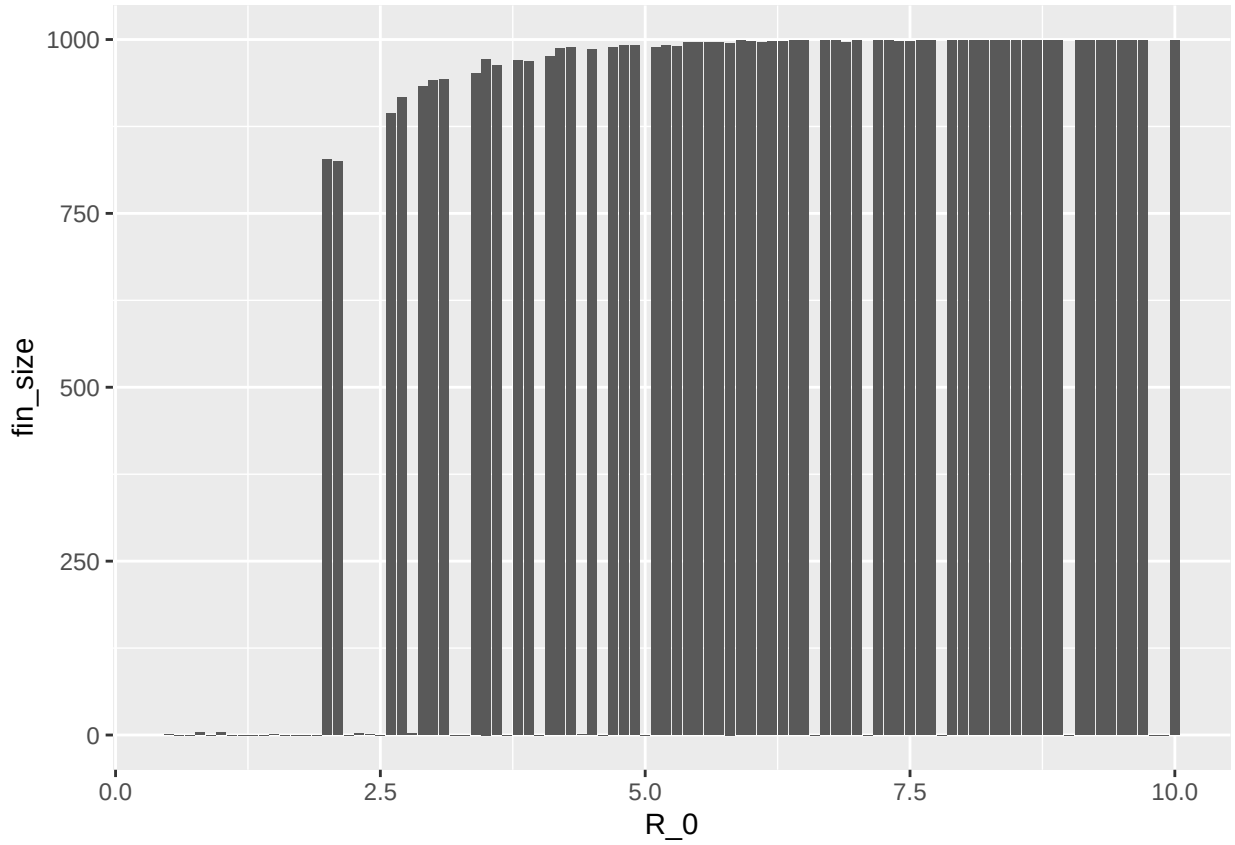
```
library(ggplot2)
```

```
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.