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 0.0861099 0.1936956 0.5074140
##
## $type
## [1] 1 1 2 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]]$t
## [1] 0.000000 2.007769 2.027867 2.719262 2.968399 3.043663
##
## [[1]]$type
## [1] 1 1 2 1 2 2
##
##
## [[2]]
## [[2]]$t
##  [1] 0.0000000 0.1559387 0.2037172 0.4517394 1.1865950 1.3822409 1.4657535
##  [8] 1.5215977 1.5528893 1.6414918 1.6871305 1.8765129 1.8818203 2.0963757
## [15] 2.2664996 2.3461854 2.5639618 2.6863619 2.8598353 2.8907087 2.8984595
## [22] 3.0594020 3.2483682 3.4249099 3.5230241 3.5552390 3.8952912 4.1036558
## [29] 4.5119515 4.8479709 5.0158671 5.2710623 5.6321701 6.1053189
##
## [[2]]$type
##  [1] 1 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 2 1 2 2 1 2 1 2 2 2 2 1 2 2 1 2 2
##
##
## [[3]]
## [[3]]$t
##  [1] 0.00000000 0.07614066 0.29079856 0.44579716 0.44876092 0.48231912
##  [7] 0.51696523 0.57265609 0.59985299 0.60423727 0.63005505 0.63432628
## [13] 0.66696749 0.68498501 0.68727336 0.69990666 0.70701518 0.78637302
## [19] 0.81154968 0.84518907 0.89927634 0.91827384 1.02961806 1.33512938
## [25] 1.34414291 1.35100366 1.51108628 1.52303394 1.56039531 1.89985152
## [31] 1.91681974 1.96915466 1.98390594 2.00580277 2.06534171 2.10600826
## [37] 2.15390920 2.67713432 3.40187262 3.42046503 3.68723032 4.43341549
##
## [[3]]$type
##  [1] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 2 2 1 2 2 2 2 2 2 2 2 2
## [39] 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]]$t
##  [1] 0.0000000 0.2570441 0.2732894 0.3020570 0.3920984 0.6582735 0.7892389
##  [8] 0.8243430 1.4509961 1.7236880 1.7434203 1.9513908 2.2894684 2.9813545
##
## [[1]]$type
##  [1] 1 1 1 2 1 1 2 2 2 1 1 2 2 2
##
## [[1]]$fin_size
## [1] 6
##
## [[1]]$duration
## [1] 2.981355
##
##
## [[2]]
## [[2]]$t
## [1] 0.00000000 0.02546873
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.02546873
##
##
## [[3]]
## [[3]]$t
## [1] 0.0000000 0.0344854
##
## [[3]]$type
## [1] 1 2
##
## [[3]]$fin_size
## [1] 0
##
## [[3]]$duration
## [1] 0.0344854
```

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]]$t
## [1] 0.0000000 0.4227784
##
## [[1]]$type
## [1] 1 2
##
## [[1]]$fin_size
## [1] 0
##
## [[1]]$duration
## [1] 0.4227784
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.5523803
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.5523803
##
##
## [[3]]
```

```
## [[3]]$t
##  [1] 0.0000000 0.3168388 0.3564816 0.4375324 0.4961179 0.6212696 0.6511777
##  [8] 0.8972769 0.9475616 0.9557703 0.9805943 0.9864134 1.0273386 1.0488412
## [15] 1.1294333 1.1774534 1.1843915 1.2974171 1.3597301 1.4218458 1.4998845
## [22] 1.5407296 1.5889200 1.5910159 1.6102848 1.6164982 1.6483446 1.6523404
## [29] 1.7367227 1.7826118 1.8144429 1.9503442 1.9970648 2.3080219 2.3355654
## [36] 2.4464764 2.9755613 4.2558864
##
## [[3]]$type
##  [1] 1 1 1 1 1 1 1 1 2 1 1 2 2 1 2 1 1 1 1 2 2 1 2 1 2 2 2 2 2 2 1 2 2 2 1 2 2 2
##
## [[3]]$fin_size
## [1] 18
##
## [[3]]$duration
## [1] 4.255886
```

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]]$t
##  [1] 0.00000000 0.01709666 0.03983325 0.07205201 0.47515789 0.56628426
##  [7] 0.58678738 0.68727509 0.74730999 0.79405414 0.84286953 1.00000000
## [13] 1.01709666 1.03983325 1.07205201 1.44514042 1.47515789 1.56502267
## [19] 1.56628426 1.58678738 1.68727509 1.74730999 1.79405414 1.84286953
## [25] 1.86309448 2.38773011 2.44514042 2.56502267 2.86309448 2.91641317
## [31] 3.38773011 3.91641317
##
## [[2]]$type
##  [1] 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 1 2 2 2 2 2 2 1 1 2 2 2 1 2 2
##
## [[2]]$fin_size
## [1] 15
##
## [[2]]$duration
## [1] 3.916413
##
##
## [[3]]
## [[3]]$t
##  [1] 0.0000000 0.2732585 0.7201237 0.7667432 0.9357718 0.9528798 0.9581583
##  [8] 0.9628932 0.9699123 0.9838368 0.9840826 1.0000000 1.0166302 1.0450916
## [15] 1.1424543 1.2089498 1.2250952 1.2732585 1.2873060 1.5860652 1.7201237
## [22] 1.7667432 1.8389427 1.9357718 1.9528798 1.9581583 1.9628932 1.9699123
## [29] 1.9838368 1.9840826 2.0166302 2.0450916 2.1424543 2.2089498 2.2250952
## [36] 2.2873060 2.5860652 2.8389427
##
## [[3]]$type
##  [1] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
##
## [[3]]$fin_size
## [1] 18
```

```
##
## [[3]]$duration
## [1] 2.838943
```

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] 0.0000000 0.8015317 0.8714036 0.9439642 1.1108979 2.6537274
```

```
##
## [[1]]$type
## [1] 1 1 2 1 2 2
##
## [[1]]$fin_size
## [1] 2
##
## [[1]]$duration
## [1] 2.653727
##
##
## [[2]]
## [[2]]$t
## [1] 0.0000000 0.1629416
##
## [[2]]$type
## [1] 1 2
##
## [[2]]$fin_size
## [1] 0
##
## [[2]]$duration
## [1] 0.1629416
##
##
## [[3]]
## [[3]]$t
##  [1] 0.00000000 0.07045346 0.17787903 0.25806636 0.29061107 0.30689438
##  [7] 0.36683937 0.37412001 0.42379166 0.50977404 0.59393927 0.59708694
## [13] 0.61954184 0.62135337 0.67501358 0.67909382 0.74199002 0.88735669
## [19] 0.88828745 0.94364317 1.05408451 1.08687932 1.11195924 1.13224278
## [25] 1.13834661 1.15532009 1.24908186 1.24953126 1.30151912 1.45113289
## [31] 1.45203762 1.57876693 1.57971625 1.58901438 1.96657358 2.32125086
## [37] 2.52007624 3.00539174
##
## [[3]]$type
##  [1] 1 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 2 2 2 2 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
##
## [[3]]$fin_size
## [1] 18
##
## [[3]]$duration
## [1] 3.005392
```

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.

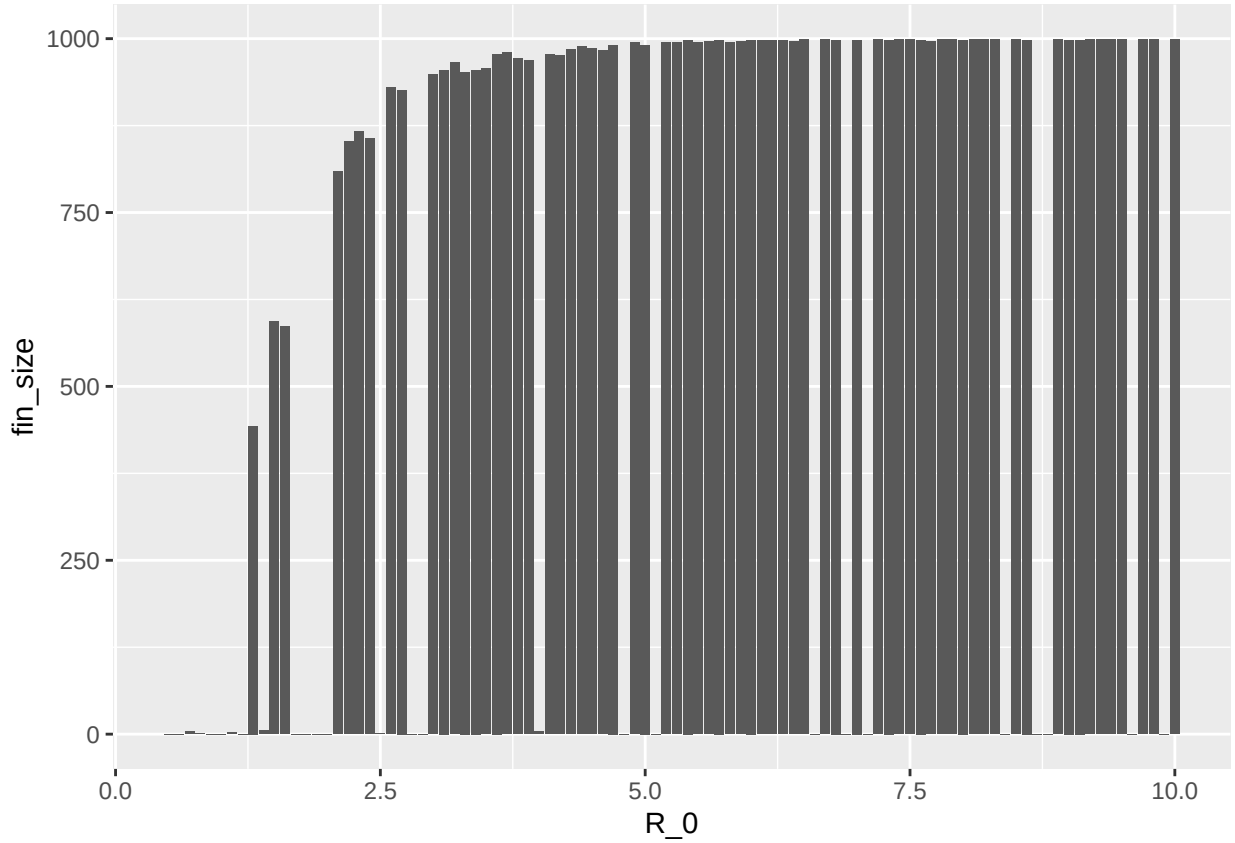
SIR Markov

```
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.