Stat 400 Presentation

12/7/2021

Novel approach for Monte Carlo simulation of the new COVID-19 spread dynamics

Original Authors: Stavros Maltezos, Angelika Georgapakopoulou

Resimulated by: Nathan Hemenway, Mark Hinds, Ian Hall

Introduction

- Aim to investigate the transmission and incubation random processes of COVID-19.
- Our focus will be on simulating daily infection case counts.
- The goal of the study is to generate epidemiological based on natural mechanism of transmission of this disease assuming random interactions of a large-finite number of individuals in very short distance ranges.

Assumptions

- The transmission of the virus occurs when the physical distance between individuals decreases to where the probability of transmission converges to one, asymptotmically.
- the step of movement of people is a randomly assigned process.
- the specific parameters of focus will be recovery and incubation time.

General Model

- This is approached through an SIR model: Susceptible, Infected, Recovered. All change based on time as the function iterates through a set number of days.
  - 0.5% of the initial subjects start out with the virus (infected)
  - 99.5% subjects are susceptible
  - 0% recovered

Variables

- Factors to consider in the transmission of a virus.
  - What is the relationship and distribution between the infection probability and the physical distance
  - High/low virus load, varying physical distances, ineffective vs effective mask wearing.
  - Environmental conditions: humidity, temperature, air ventilation.
- Based on real epidemiological data analysis, these encompassing factors will be distributed Exponential(rate = lambda)
- Recovery and Incubation times $\sim$ Gamma(alpha, beta), But in order to get whole number for days, we distribute them Possion(lambda)

What We are Trying to Simulate

- Step Function

Simulating Step Distance and Direction

```r
get_step_length <- function(df, x_dim, y_dim){
  step_lengths <- rnorm(n=nrow(df), mean=min(x_dim, y_dim)/4, sd=min(x_dim, y_dim)/12)
  return(step_lengths)
}

get_step_direction <- function(length_vec){
  direction <- runif(n=length(length_vec), min=0, max=2*pi)
}

get_delts <- function(length_vec, direc_vec){
  delta_x <- length_vec * cos(direc_vec)
  delta_y <- length_vec * sin(direc_vec)
  return(cbind(delta_x, delta_y))
}
```

Updating Coordinates and In-Bounds Check

```
update_coords <- function(og_df, delta_df){
  new_mx <- matrix(data=NA, nrow=nrow(og_df), ncol=5)
  new_df <- data.frame(new_mx)
  new_df[,1] <- as.numeric(og_df[,'x']) + delta_df[,'delta_x']
  new_df[,2] <- as.numeric(og_df[,'y']) + delta_df[,'delta_y']
  new_df[,3] <- og_df[,3]
  new_df[,4] <- og_df[,4]
  new_df[,5] <- og_df[,5]
  colnames(new_df) <- c('x', 'y', 'subjects', 'incubation', 'recovery')
  new_df
}

coord_checker <- function(og_df, new_df, x_dim, y_dim){
  baddies <- NULL
  for(i in 1:nrow(new_df)){
    if(as.numeric(new_df[i, 'x']) > x_dim | as.numeric(new_df[i, 'x']) < 0 | as.numeric(new_df[i, 'y']) >
      baddies <- append(baddies, i)
    }
  }

  if(is.null(baddies)){
    return(new_df)
  }else{
  new_step <- get_step_length(new_df[i,], x_dim, y_dim)
  new_direction <- get_step_direction(new_step)
  new_change <- get_delts(new_step, new_direction)
  new_df <- update_coords(og_df[baddies,], new_change)
  full_df <- rbind(new_df, og_df[-baddies,])
  return(coord_checker(og_df, full_df, x_dim, y_dim))
  }
}
```

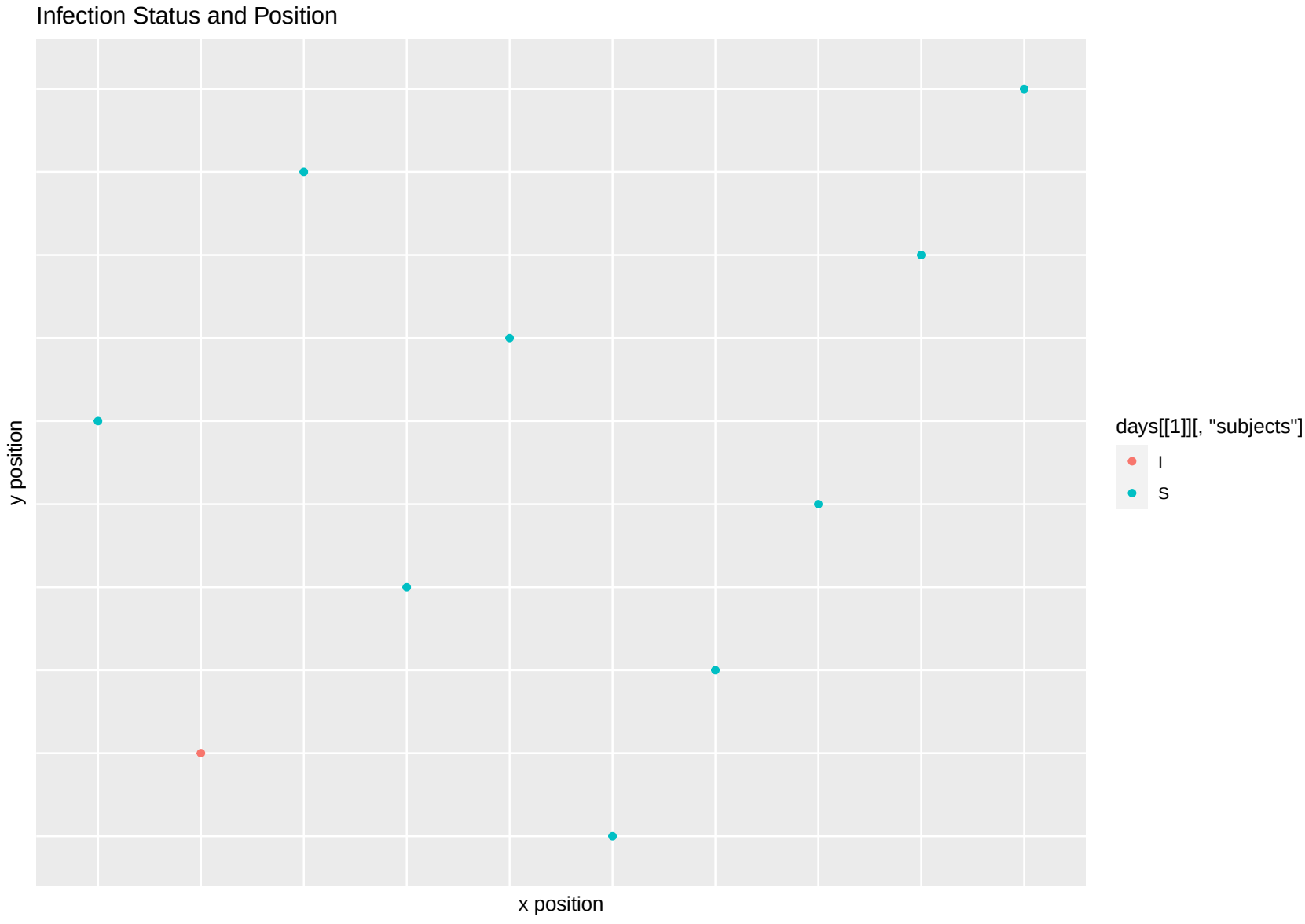
Changing SIR Status

```
infection_status <- function(state, mean_crit_dist){

  current_infected <- state[state[,3]=='I' & state[,4] <= 0,]
  current_suceptable <- state[state[,3]=='S',]
  current_removed <- state[state[,3]=='R',]
  current_non_contagious <- state[state[,3]=='I' & state[,4] > 0,]

  newly_infected <- NULL

  if(nrow(current_suceptable) != 0 & nrow(current_infected) != 0){
  for(i in 1:nrow(current_infected)){
    for(j in 1:nrow(current_suceptable)){
      x_i <- as.numeric(current_infected[i, 'x'])
      y_i <- as.numeric(current_infected[i, 'y'])
      x_s <- as.numeric(current_suceptable[j, 'x'])
      y_s <- as.numeric(current_suceptable[j, 'y'])

      crucial_distance <- rexp(1, 1/mean_crit_dist)

      d <- sqrt((x_i - x_s)^2 + (y_i - y_s)^2)
      if(d <= crucial_distance){
        newly_infected <- c(newly_infected, j)
      }
    }
  }
  current_suceptable[newly_infected, 3] <- 'I'
  current_suceptable[newly_infected, 4] <- rpois(n=length(newly_infected), lambda = 5)
  current_suceptable[newly_infected, 5] <- rpois(n=length(newly_infected), lambda = 14)

  new_state <- rbind(current_infected, current_suceptable, current_removed, current_non_contagious)
  return(new_state)
  }else{return(state)}
}
```

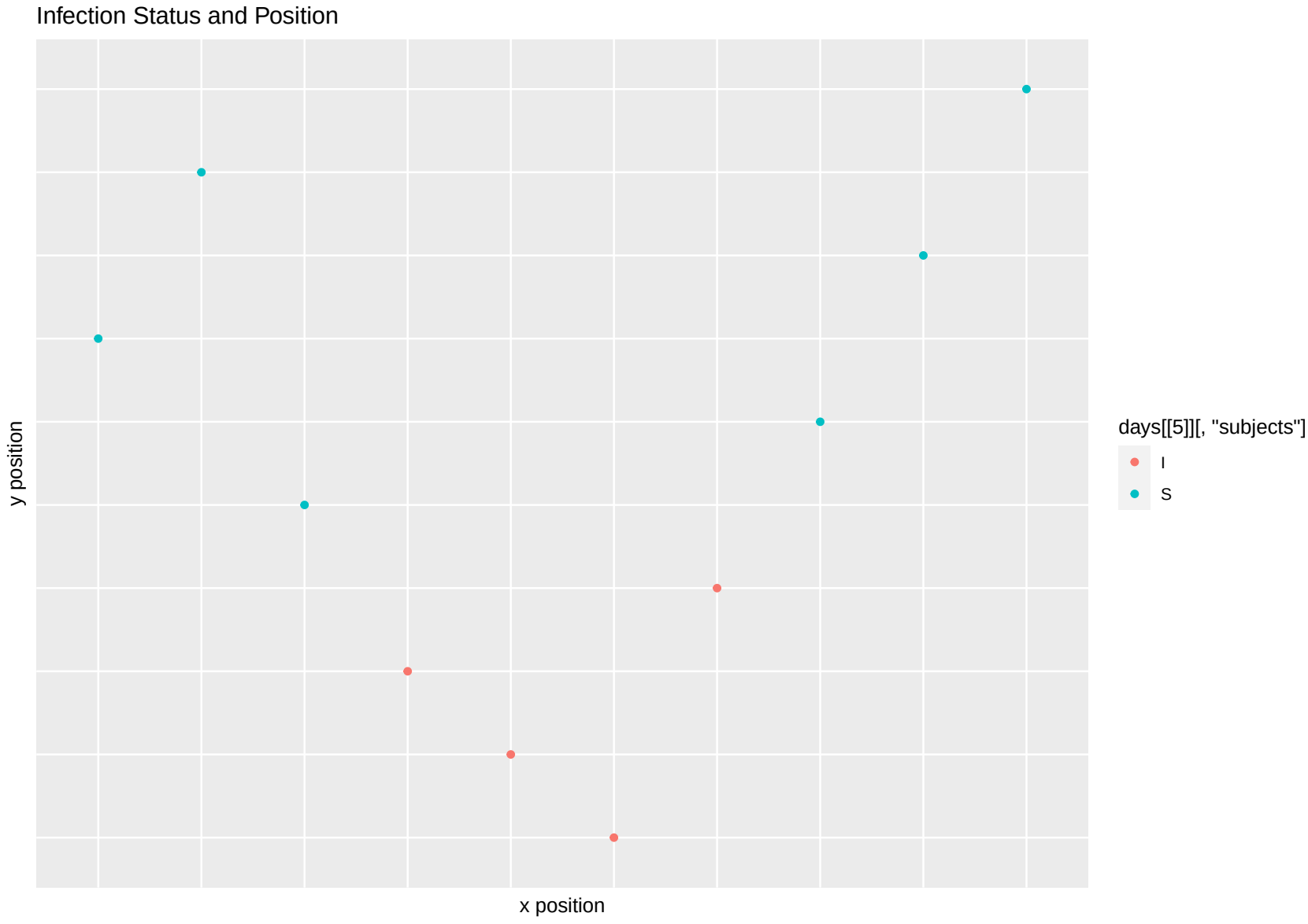
Simulation

```
dummy_simulator <- function(days, num_subjects, x_dim, y_dim, I_0, inf_dist){
  initial_data <- starting_vals(num_subjects = num_subjects, x_dim=x_dim, y_dim=y_dim, I_0=I_0)
  end_day <- list()
  num_cases <- rep(NA, days)
  num_cases <- c(I_0, num_cases)
  num_new_cases <- rep(NA, days)
  for(i in 1:days){
    if(i == 1){
      step_size <- get_step_length(initial_data, x_dim, y_dim)
      direction <- get_step_direction(step_size)
      change <- get_delts(step_size, direction)
      updated_positions <- update_coords(initial_data, change)
      checked_positions <- coord_checker(og_df=initial_data, new_df=updated_positions, x_dim=x_dim, y_dim=
      new_status <- infection_status(checked_positions, inf_dist)

      not_recovered <- new_status[as.numeric(new_status[, 5]) > 0, ]
      new_time <- as.numeric(not_recovered[, 5]) - 1
      not_recovered[, 5] <- new_time
      infected_0 <- new_status[as.numeric(new_status[, 5]) == 0, ]
      infected_0[, 3] <- rep('R', nrow(infected_0))
      not_infected = new_status[as.numeric(new_status[, 5]) < 0, ]
      new_status = rbind(not_recovered, infected_0, not_infected)
      infected_people <- new_status[new_status[,3] == 'I',]
      new_inc_time <- as.numeric(infected_people[,4]) - 1
      infected_people[,4] <- new_inc_time
      other_people <- new_status[new_status[,3] != 'I',]
      new_status <- rbind(infected_people, other_people)
      end_day[[i]] <- new_status
      num_cases[i+1] <- sum(new_status[,3]=='I')
      num_new_cases[i] <- sum(new_status[,3]=='I') - I_0
    }
    }
  return(num_cases)
}
```

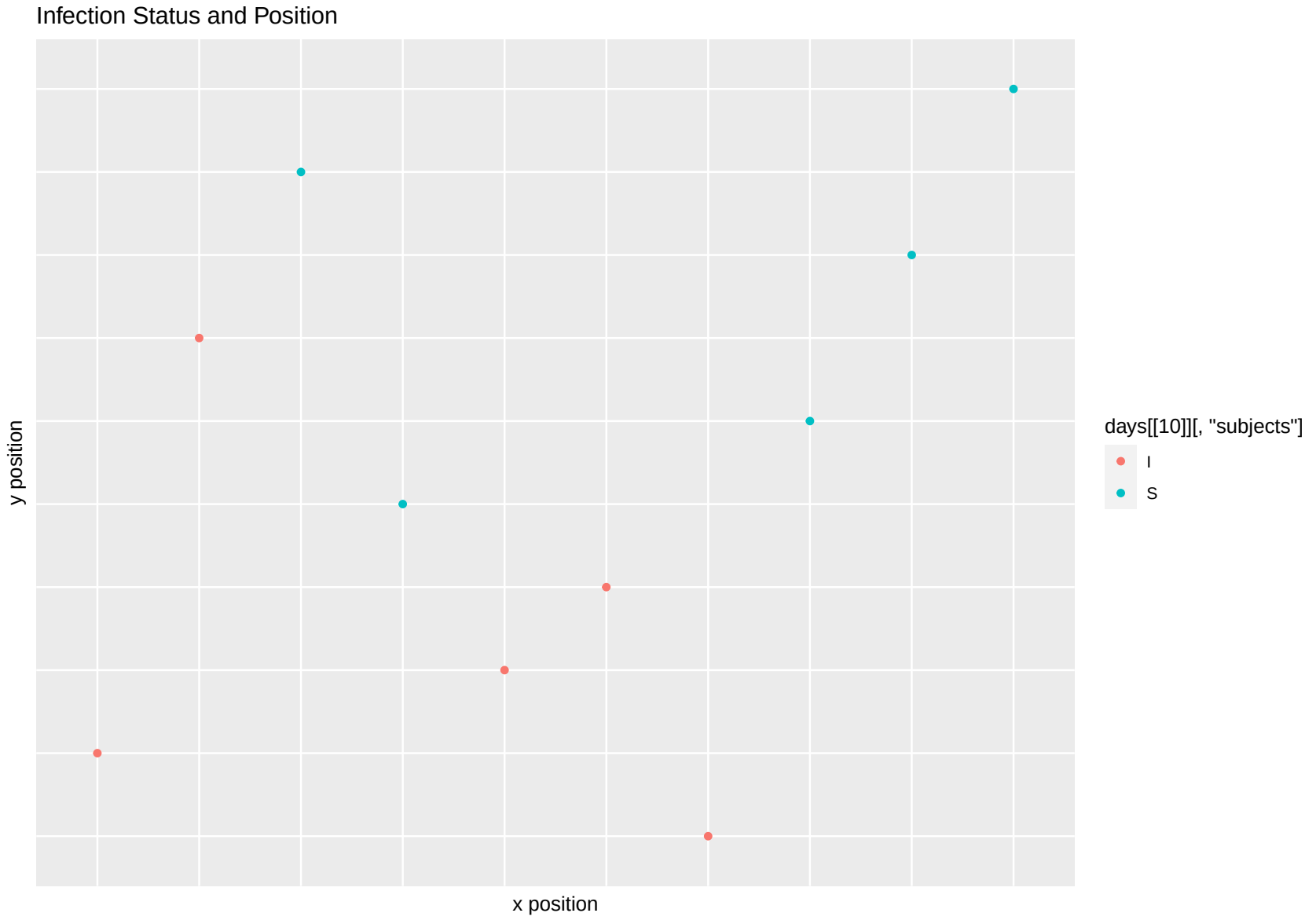
Demonstration Day 1



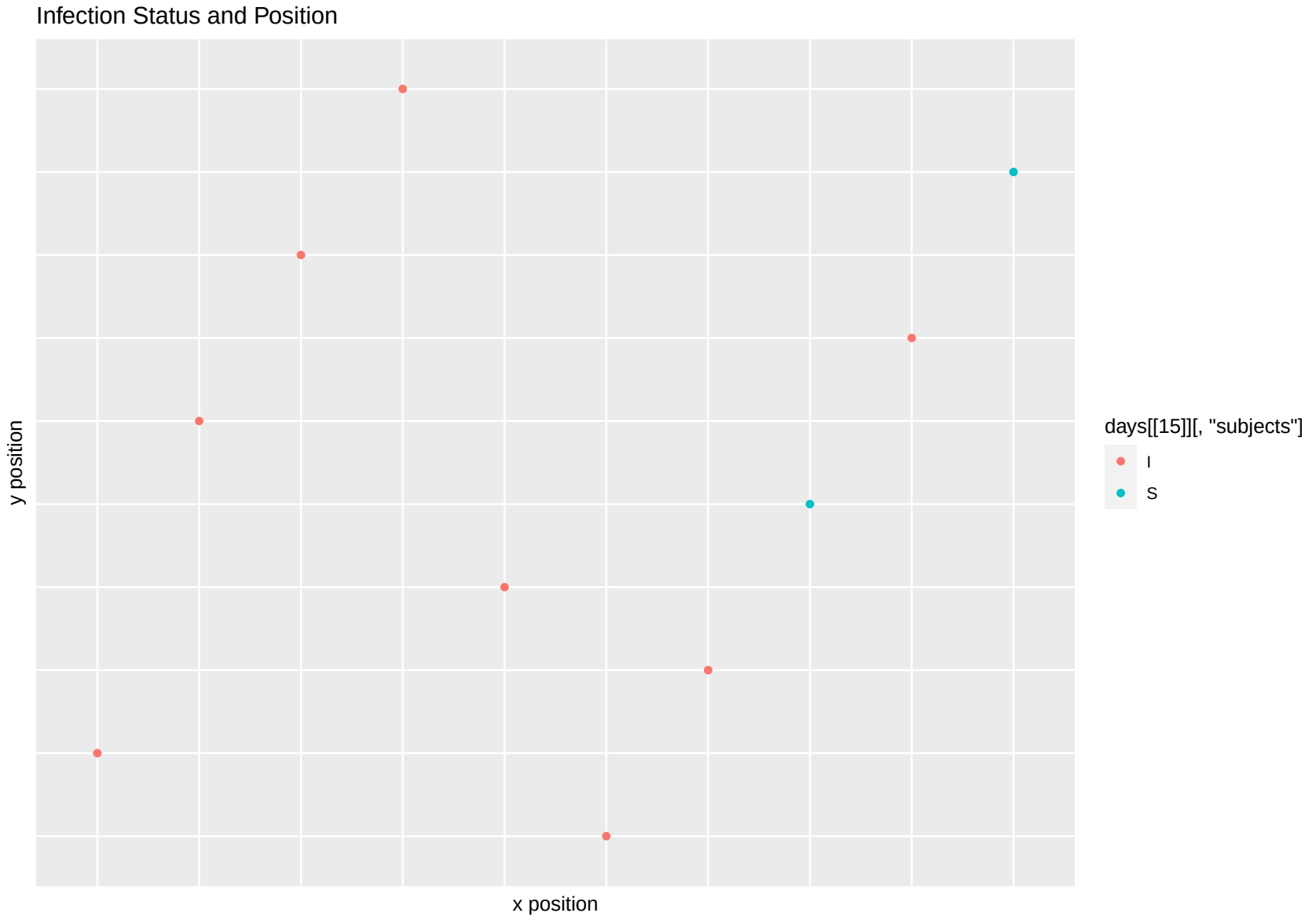
Demonstration Day 5



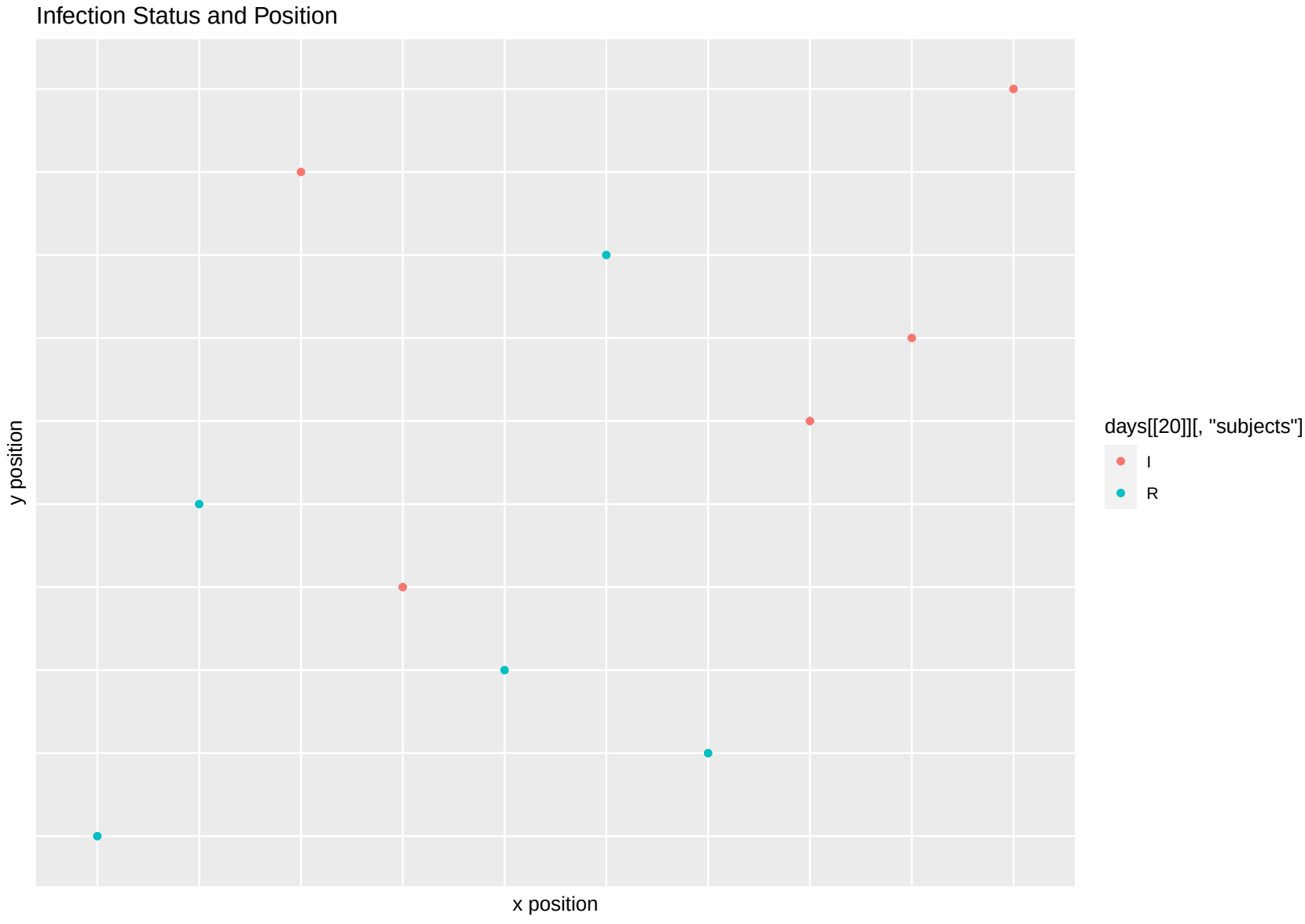
Demonstration Day 10



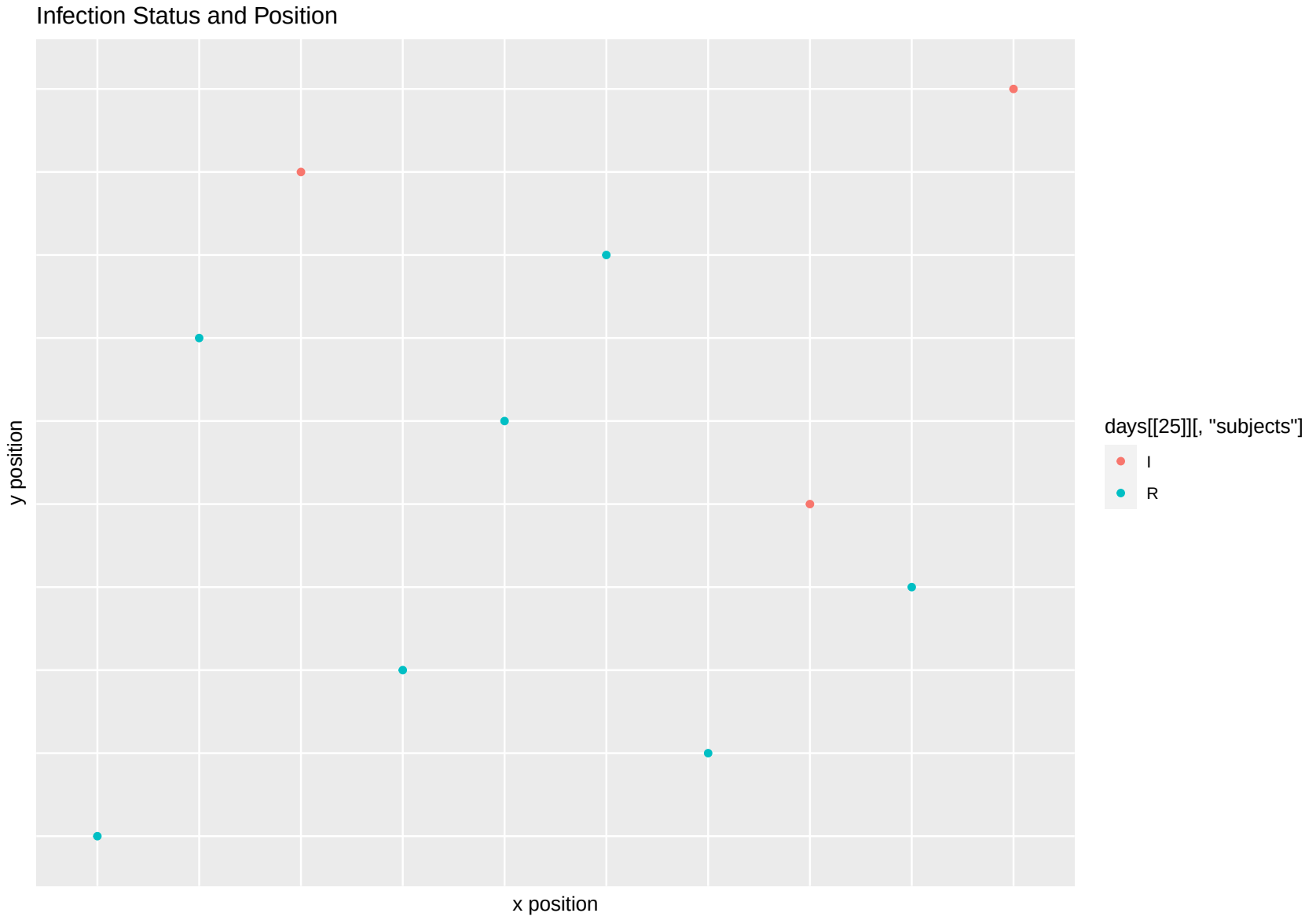
Demonstration Day 15



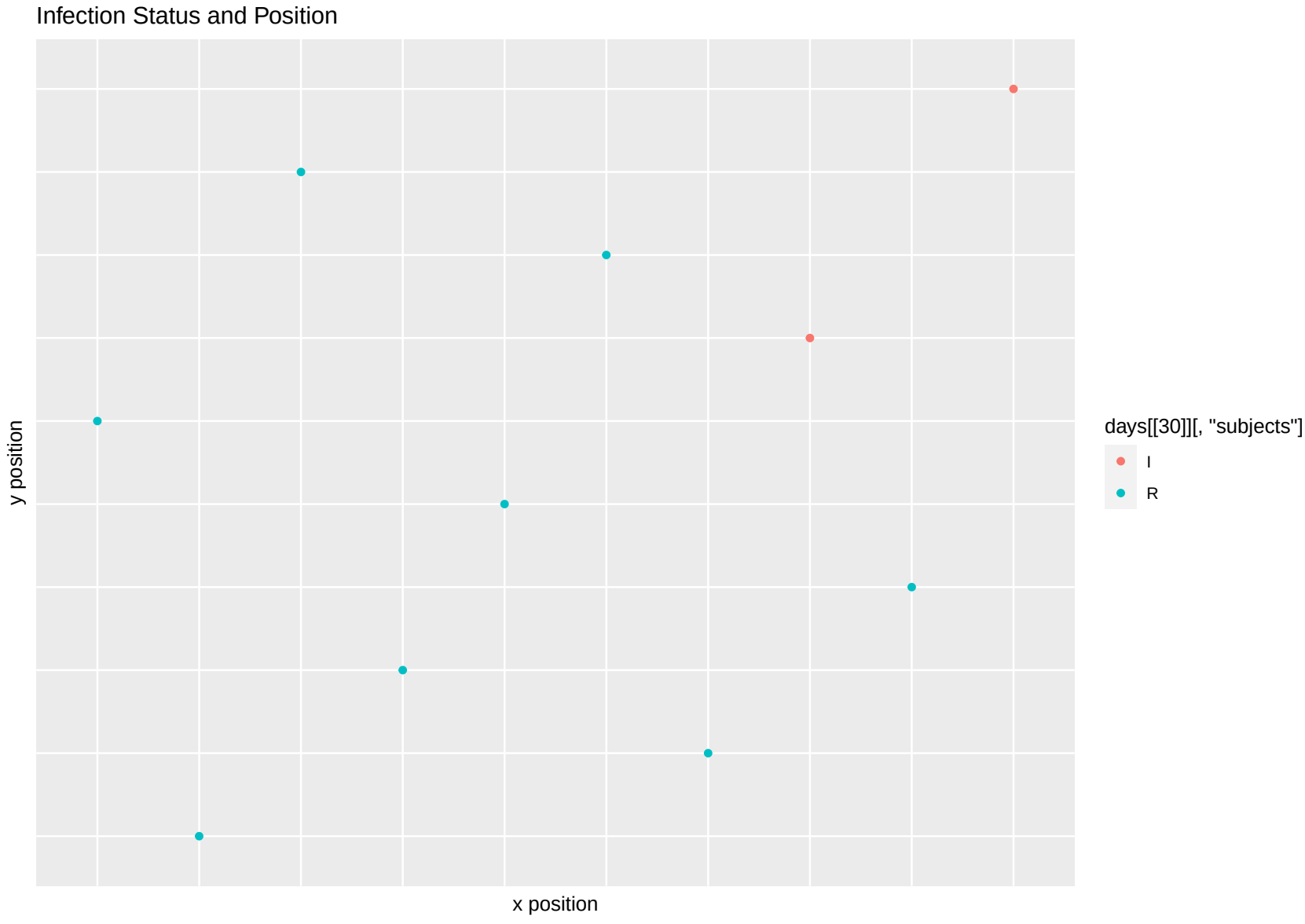
Demonstration Day 20



Demonstration Day 25



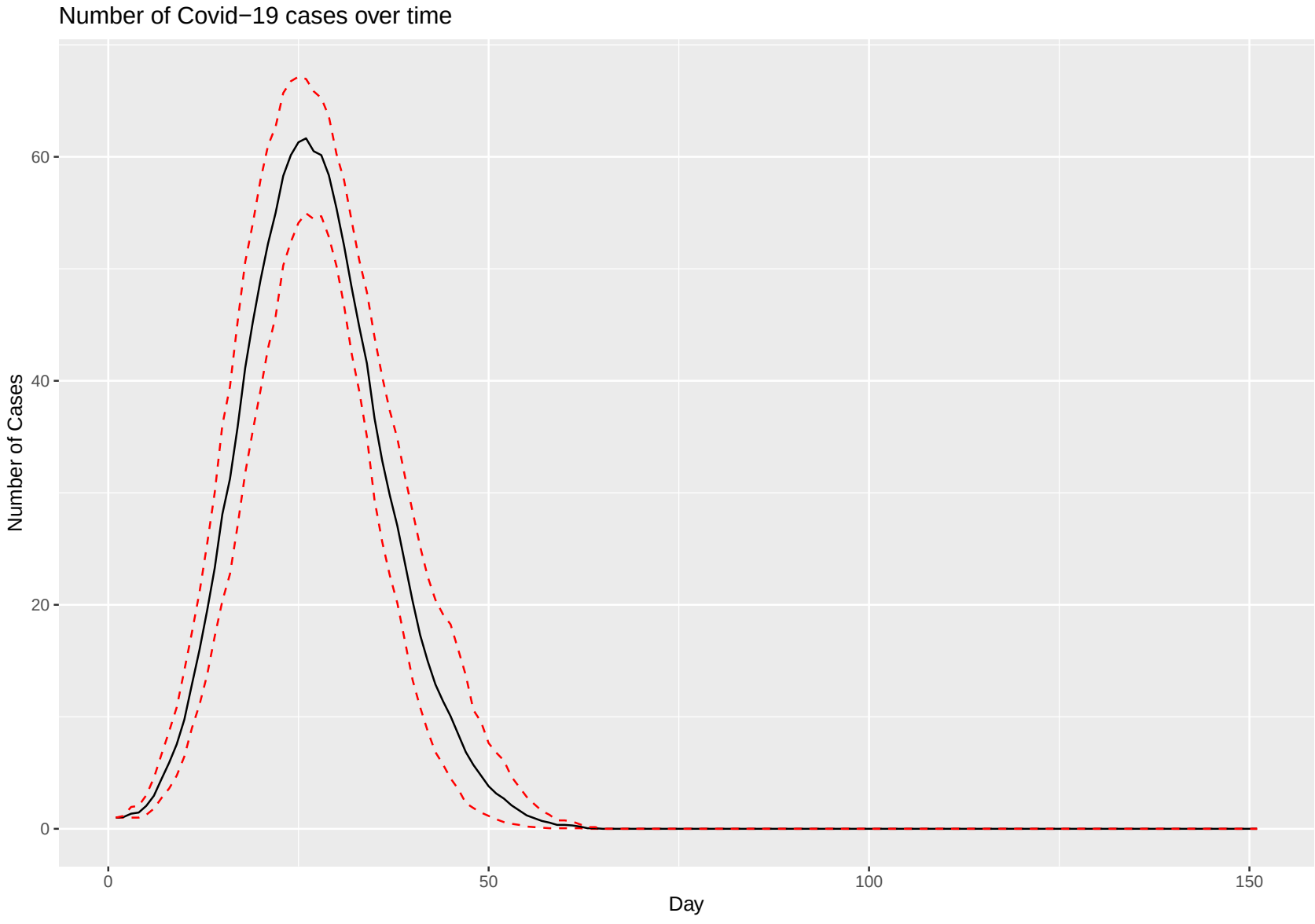
Demonstration Day 30



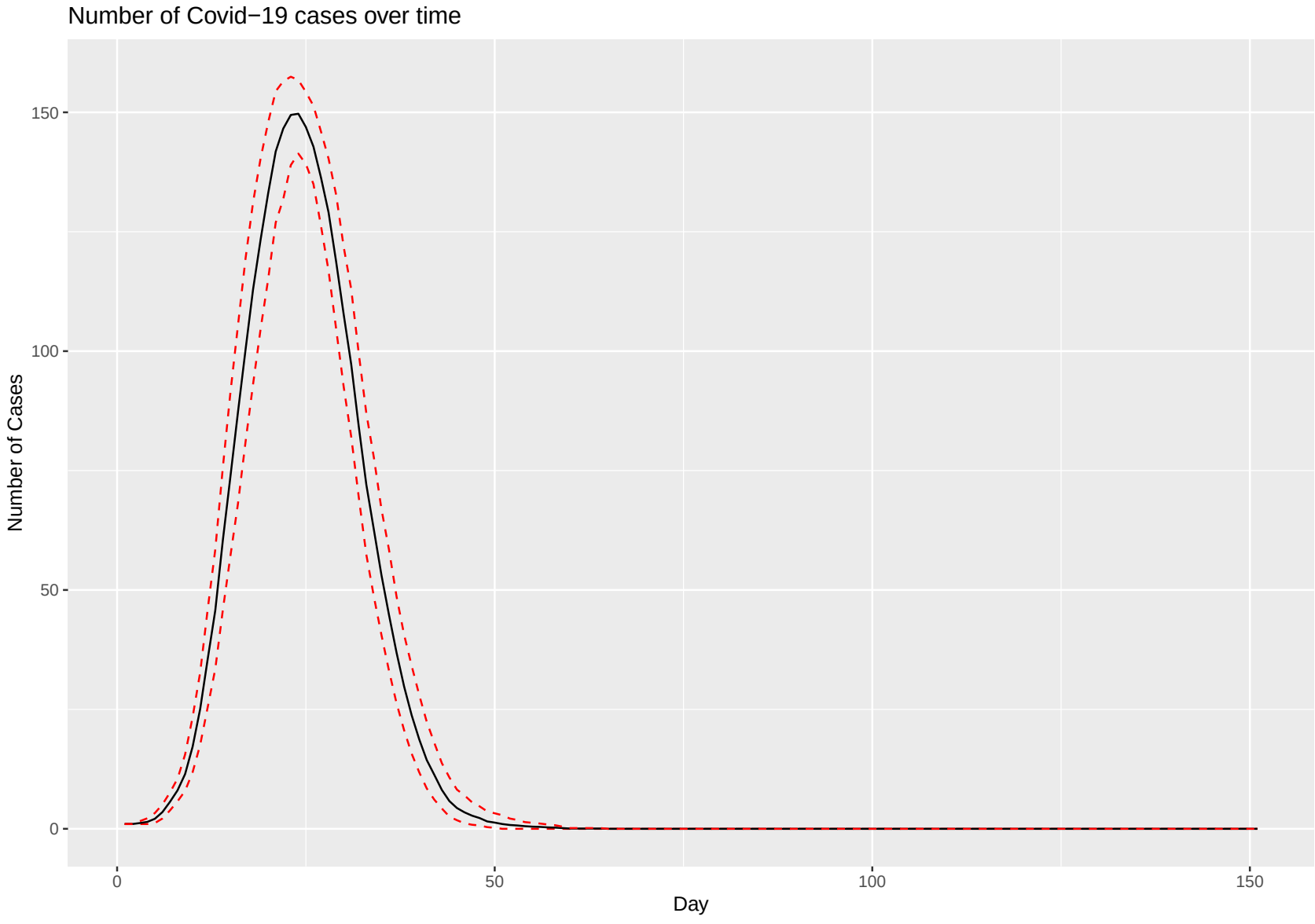
Bootstrap Confidence Intervals

```
mean_func <- function(daytuh, idx) {
  mean(daytuh[idx])
}

get_bootstrap_cis <- function(days, num_subjects, x_dim, y_dim, I_0, inf_dist, sample_size, ci_type='perc'
  #Create data frame to store samples
  samps <- matrix(data=NA, nrow=sample_size, ncol=days + 1)
  samps <- data.frame(samps)

  #Create data frame to store confidence intervals
  ci_df <- matrix(data=NA, nrow=days + 1, ncol=3)
  ci_df <- data.frame(ci_df)

  for(i in 1:sample_size){
    samps[i, ] <- simulator(days, num_subjects, x_dim, y_dim, I_0, inf_dist)
  }

  for(i in 1:ncol(samps)){
    #Make sure column has more than one unique value
    check <- (length(unique(samps[ ,i])) != 1)
    if(check){
    boot_obj <- boot(samps[ ,i], mean_func, 2000)
    ci_obj <- boot.ci(boot_obj, conf = .95, type = ci_type)
    ci_df[i, 1] <- ci_obj$percent[4]
    ci_df[i, 2] <- ci_obj$t0
    ci_df[i, 3] <- ci_obj$percent[5]
    }else{ci_df[i, ] <- c(samps[1, i], samps[1, i], samps[1, i])}
  }
return(ci_df)
}
```

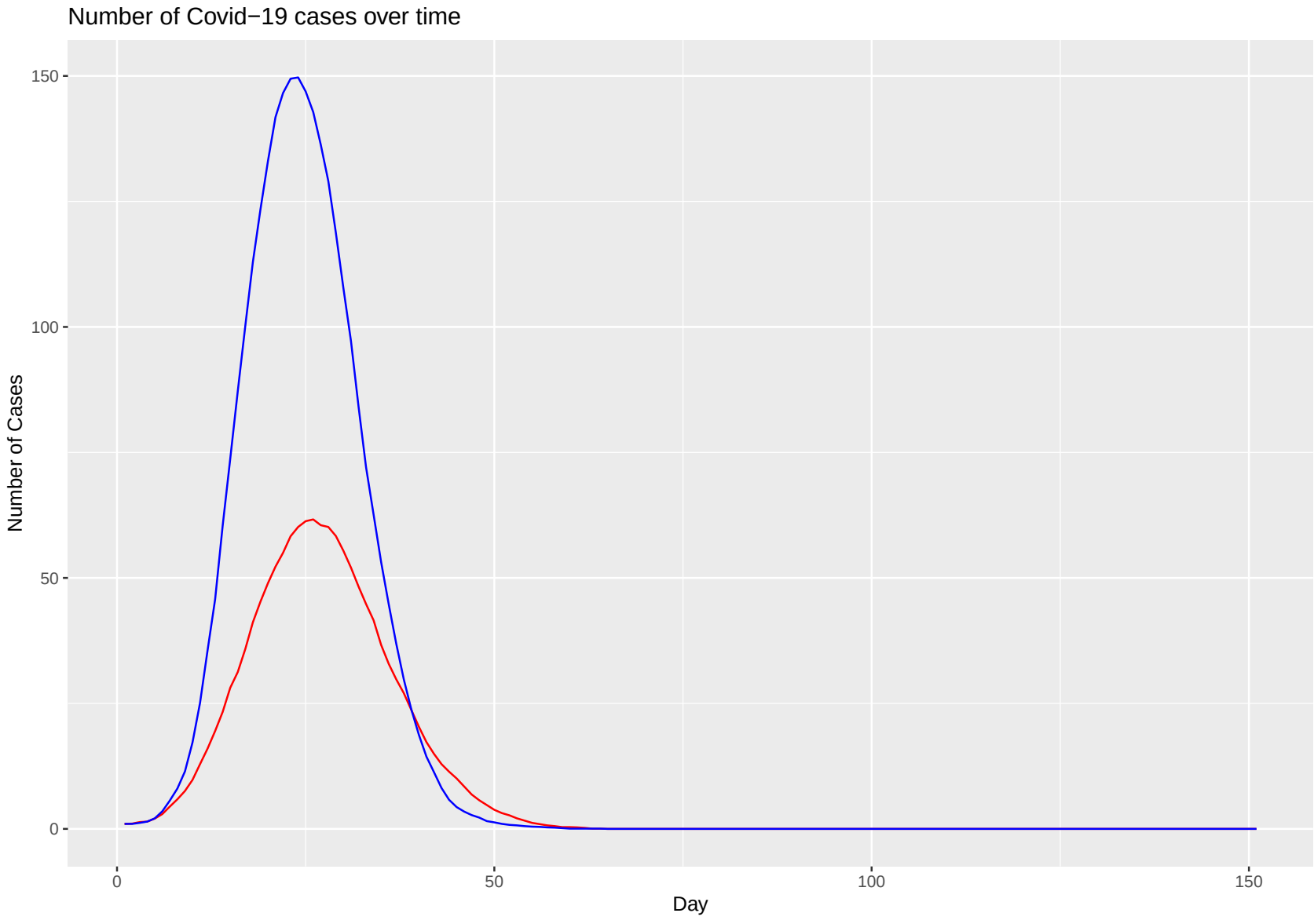
Plot With Bootstrap 95% CI, 100 subjects



Plot With Bootstrap 95% CI, 100 subjects



‘Flatten the Curve’ - Pseudo Vaccination Effect



Pseudo Mask Effect