Hw8

Joe Stoica, Corey Maxedon, Austin Lesh

12/3/2019

Background

Contagion models describe the spread of an attribute in a network: diseases in contact networks, behaviors in friend networks, ideas in discussion networks, and so on. We will look at data from a study on how doctors adopted a new drug. Coleman, Katz, and Menzel 1 collected data on doctors in several cities in Illinois, including the first date on which they prescribed tetracycline (a new drug at the time) and which other doctors they went to for advice or discussion of treatment options. One of the theories investigated in the paper was that doctors who knew other doctors who had already prescribed the drug adopted it more quickly than those who didn't. This behavior can be described as a contagion model. In this assignment, we will simulate from a contagion model and use the simulations to obtain an approximate posterior distribution of the contagion parameter using approximate Bayesian computation.

The contagion model we will consider is the susceptible-infected (SI) contagion model. In this model, each of the nodes in a network can be either susceptible to infection or already infected. At each time step, each infected node picks one of its neighbors uniformly at random. If the chosen neighbor is not infected, it becomes infected with probability p. For us, each of the nodes is a doctor, an "infected" node is a doctor who has prescribed tetracycline, and the links in the network are doctors who discuss cases with each other.

1

```
attributes <- read_csv("attributes.csv")
attributes$ID <- 1:117

network <- read_csv("network.csv")
```

2

```
#' Simulating from the contagion model
#'
#' @param p Infection probability p
#' @param nodes A set of initially infected nodes
#' @param network The network given by the data
#' @param steps The number of steps to simulate
#'
#' @return A vector giving the time point at which each node was infected
contagion_sim <- function(p, network, nodes, steps) {

  n = network
  # this will be the vector that contains the time of infection for each doctor
  ans = rep(NA, 117)
  # add the initial infected
  ans[nodes] = 1
```

```
# keep track of the nodes that are infected
infected_nodes = nodes

# This is a helper function to check to see if the doctor is infected or not
prob_helper = function(x){
  ifelse(runif(1) <= p, TRUE, FALSE)
}

# This is a function to fix when the length of x = 1, which can happen if a
# doctor has spoken to only one other doctor
sample_fix <- function(x) {
  ifelse(length(x) <= 1, x, sample(x, 1))
}

# subset the phone book for the infected
master = alply(network, 1, function(x){which(x == 1, arr.ind = FALSE)})

for (step in 2:steps) {

  # Get the infected doctors and who they talked to
  # Note: Doctor i can list doctor j if they have talked to them, but doctor j
  # might not have included i. Therefore, we can remove the j doctors
  # (the j doctors all have integer(0) as their entry in the list)
  conversation_list <- Filter(length, na.omit(master[infected_nodes]))

  # Select a doctor from conversation list randomly
  random_doctor = sapply(conversation_list, function(x){sample_fix(x)})

  # Use p to see if they're infected
  infected_result = sapply(random_doctor, prob_helper)

  # This is super messy, but basically it finds the doctors who got infected,
  # adds the time the doctor was infected, and updates the infected list
  infected_df = data.frame(random_doctor, infected_result) %>%
    filter(infected_result == TRUE) %>%
    distinct()

  # Enter this loop if infected_df has 1+ rows, which is the only time
  # it's interesting.
  if(dim(infected_df)[1] > 0) {

    # for every row in the infected data frame
    for(i in 1:nrow(infected_df)){

      # If the ith doctor wasn't previously infected (which happens when the
      # value isn't equal to NA),
      if (is.na(ans[infected_df[i, 1]])){

        # set time = step
        ans[infected_df[i, 1]] = step
      }
    }
  }
```

```
    # update infected
    infected_nodes = c(infected_nodes, infected_df$random_doctor)

  }

  return(ans)
}
```

3

Simulate from the model given the real data parameters for the network (the doctor discussion network), the initially infected nodes (the doctors who prescribed tetracycline in month1), and the number of time steps (18 months). Do one simulation each for p = 0.10, 0.50, 0.90, and show what one realization of the simulation looks like.

```
probs <- c(0.1, 0.5, 0.9)
initially_infected_nodes <- attributes[attributes$adoption_date == 1,]$ID
steps = 18
```

```
results <- lapply(probs,
                  contagion_sim,
                  network = network,
                  nodes = initially_infected_nodes,
                  steps = steps)
```

One realization of the simulation, for p = 0.50, looks like this:

```
## [[1]]
##   [1]  1 NA  9 NA NA 15 NA NA NA NA  4 NA 12 NA NA NA NA NA NA 11  8  6  6
##  [24] 12 NA NA  1  8  8 NA 13 NA NA NA 11  5  8 NA NA 17  2  8  4 14  9  5
##  [47] 16 NA  7 NA 11  5 NA  8  4  9 NA NA NA NA NA NA NA NA NA NA  8 NA NA 10
##  [70] NA NA NA NA NA  1 NA  4  1 NA NA  3 13 NA 13 NA  8  7 NA  9  3 NA  6
##  [93]  1  7  9 11 NA  4  6  2  5 11 NA NA NA NA NA NA  7  3 NA NA 10 15 NA
## [116] NA 14
```

It is important to note that it is possible that not every time point will be present in the output, and this case occurs when no doctors are infected at that time.

4

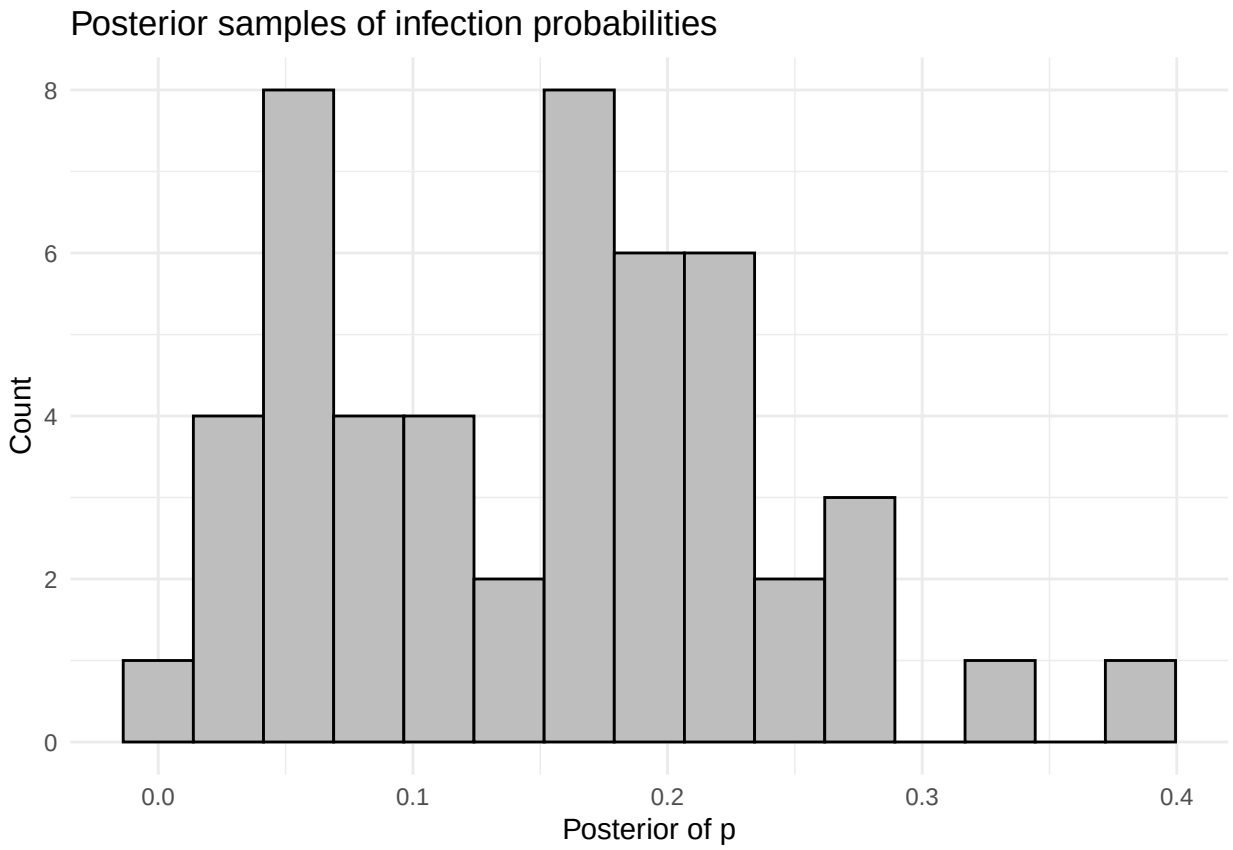
```
generate_abc_sample = function(observed_data,
                               summary_statistic,
                               prior_distribution,
                               data_generating_function,
                               epsilon) {
  theta = prior_distribution()
  y = data_generating_function(theta)
  y[is.na(y)] = 99
  resid = summary_statistic(y, observed_data)
  return(c(theta, resid))
}
```

```
n_samples = dim(network)[1]
prior_distribution = function() runif(1)
data_generating_function = function(x) {
  contagion_sim(p = x, network = network,
                nodes = initially_infected_nodes,
                steps = 18)
}
observed_data = attributes$adoption_date
summary_statistic = function(x, y) sum(abs(x - y))

N = 1000
epsilon = .05

posterior_samples = replicate(n = N,
                              generate_abc_sample(observed_data,
                                                  summary_statistic,
                                                  prior_distribution,
                                                  data_generating_function,
                                                  epsilon))
posterior_samples = t(posterior_samples)
posterior_samples = posterior_samples[order(posterior_samples[,2]),]
best_posterior_samples = posterior_samples[1:(N*epsilon),]
```

```
data.frame(b1 = best_posterior_samples[, 1],
           b2 = best_posterior_samples[, 2]) %>%
  ggplot(aes(b1))+
  geom_histogram(color ="black",
                 fill = "grey",
                 bins = 15)+
  theme_minimal()+
  labs(x = "Posterior of p",
       y = "Count",
       title = "Posterior samples of infection probabilities")
```



Here is the histogram for the posterior samples generated from our abc algorithm.

```
mean_post = mean(best_posterior_samples[,1])
quant = quantile(best_posterior_samples[,1], probs = c(0.025, 0.975))
```

The mean of the posterior samples is 0.1480409. The 0.025 quantile of the posterior samples is 0.0206883. The 0.975 quantile of the posterior samples is 0.3289312.