Approximate Bayesian Computation for Disease Outbreaks

Ethan Weiland, Yanming Kuang, Yi Liu

December 15, 2022

Objectives

The goal of this project is to select a model for the spread of influenza using the Approximate Bayesian Computation (ABC) method. Two parameters in the model are important for a disease outbreak: the probability that an individual escapes infection from their community (q_c) and the probability that an individual escapes infection from within their household (q_h). Given multiple outbreaks, we investigated whether a two-parameter model (one with identical values of q_c and q_h for outbreaks) is better than a four-parameter model (one with different values of q_c and q_h for each outbreak). These multiple outbreaks are divided into two categories: (1) different spreads of the same strain and (2) different spreads of different strains. We examined the parameter posterior distributions by recreating Figure 3a.) and Figure 3c.) in Toni and Stumpf (2010).

Data

There are two datasets provided by Toni and Stumpf (2010). The first dataset (Table 2) describes two outbreaks of the same strain of influenza (Influenza A) and the second dataset (Table 3) describes two outbreaks of different strains of influenza (Influenza A and B). The rows represent the number of infected individuals in a household. The columns represent the total number of susceptibles in a household.

Table 2: Influenza A (H3N2) infection in 1977-78 (middle column) and 1980-81 (right column) epidemics, Tecumseh, Michigan [1].

Nr. infected individuals	1	2	3	4	5	1	2	3	4	5
0	66	87	25	22	4	44	62	47	38	9
1	13	14	15	9	4	10	13	8	11	5
2		4	4	9	1		9	2	7	3
3			4	3	1			3	5	1
4				1	1				1	0
5					0					1

Table 3: Influenza B infection in 1975-76 epidemic (middle column) and influenza A (H1N1) infection in 1978-79 epidemic (right column), Seattle, Washington [2].

Nr. infected individuals	1	2	3	4	5	1	2	3
0	9	12	18	9	4	15	12	4
1	1	6	6	4	3	11	17	4
2		2	3	4	0		21	4
3			1	3	2			5
4				0	0			
5					0			

Code and Design Decisions

The general idea in using Approximate Bayesian Computation is the following. First, we draw parameter particles from a prior distribution. Then we simulate data according to a probability model given the parameter particles. Next, we compute the similarity between the simulated data and the observed data. Lastly, we make a decision to keep or discard the sample based on the similarity. Our code consists of one first-level function and three second-level functions. We will now discuss the second-level functions before turning our attention to the first-level function.

The first second-level function is named "generate_probability_matrix". It takes three arguments: q_c (the probability that an individual escapes infection from their community), q_h (the probability that an individual escapes infection from their household), and the observed data. The probability that j out of s susceptibles in a household become infected is calculated based on the following formula:

$$w_{js} = \binom{s}{j} w_{jj} (q_c q_h^j)^{s-j},$$

where $w_{0s}=q_c s$, $s=0, 1, 2,\ldots$ and $w_{jj}=1-\sum_{i=0}^{j-1} w_{ij}$. The output of this function is a probability matrix that is used to simulate data.The observed data is used to make sure the probability matrix has the same size as the observed data. To check our work, we implement a test that ensures every probability (i.e., every non-NA value) is between 0 and 1, as probabilities should be. We generate a few sample probability matrices, and we write a function that counts the number of values that fall outside of [0,1]. Then we test if this function's output is equal to 0. The test passed. In other words, we are confident that every probability is between 0 and 1.

The next second-level function is named "generate_data". This function takes two arguments: the previously constructed probability matrix and the observed data. The output of "generate_data" is a simulated dataset. To ensure this dataset is correct, we test that each element on the diagonal is a real (i.e., non-NA) value. The diagonal should contain real values because the diagonal does not represent scenarios where the number of infected individuals exceeds the total number of individuals in a household. We generate a function that counts the number of NA values on the diagonal of a matrix. Using the same generated sample matrices, we test if this functions' output is equal to 0. The test passed. In other words, we ensure that the diagonal of every probability matrix does not contain NA values.

The final second-level function is named "generate_distance". This function calculates the summary statistic used in ABC and takes four arguments: the observed data from the first outbreak of the same virus strain (i.e., influenza A (H3N2) in 1977-78), the simulated data for the second outbreak of the same virus strain (i.e., influenza A (H3N2) in 1980-81), the observed data from the first outbreak of the different virus strains (i.e.,influenza B in 1975-76), and the simulated data for the second outbreak of the different virus strains (i.e.,influenza A (H1N1) in

1978-79) . This function outputs a distance measure between the simulated data and the observed data according to instructions in Toni and Stumpf (2010). We generate a few sample matrices to test the distance function. The distance function is given by Toni and Stumpf (2010) as the following:

$$d(D_0, D^*) = \frac{1}{2}(||D_1 - D^*(q_{h1}, q_{c1})||_F + ||D_2 - D^*(q_{h2}, q_{c2})||_F)$$

Using tests, we find the distance matches between the actual value and the expected value. In other words, the distance function is working as intended.

The first-level function is named "generate_abc_sample". It takes six arguments: observed_data1 and observed_data2 are either the values reported in Table 2 or Table 3 provided by Toni and Stumpf (2010). The third argument summary_statistic is the second-level function "generate_distance". The forth argument prior_distribution is the uniform distribution with the range [0, 1] for the parameters (i.e., probabilities q_h, q_c) that we want to estimate. The fifth argument data_generating_function is the second-level function named "generate_data". The sixth argument epsilon is the tolerance for the distance between observed data and simulated data and is taken as the decision criterion for the sampled parameter particles. Within this function, four parameters are randomly sampled from their prior distributions. After that, two probability matrices are generated. Next, two simulated dataset are generated based on these probability matrices. Then the observed data and simulated data are passed into the summary_statistic to compute the similarity. The similarity value is compared with the epsilon. If the similarity value is less than the epsilon, we will accept these parameter particles and the function will return them as a vector. If the similarity value is greater than the epsilon, the whole procedure will repeat until accepted ones occur. Last, we repeat the ABC algorithm 1000 times for outbreaks by the function replicate() to generate enough posterior parameter samples.

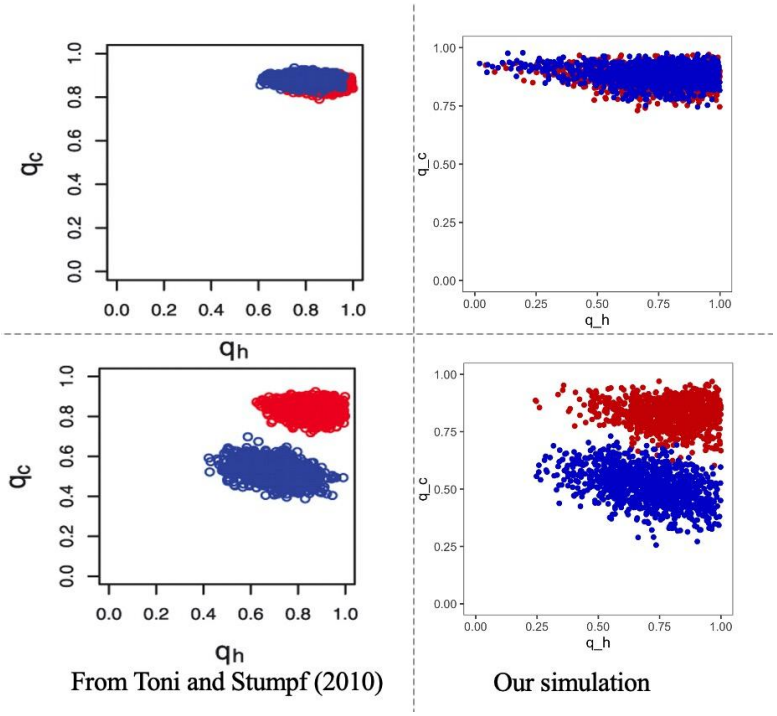
Since this function outputs our best attempt at replicating Figure 3a) and Figure 3c), the best way to ensure our function runs correctly is to compare our replicated figures to the actual figures. We will do so in the Results section.

Some design choices we made include: deciding upon the output of our first-level function, choosing a tolerance level, picking the number of accepted samples to calculate, and deciding to break up Table 2 and Table 3. At first, our first-level function was named "generate_figure" and performed two steps: the ABC algorithm and plotting the accepted samples. However, this output prevented us from seeing more detail on the accepted samples outside of the plots. So we decided to break up this function and perform ABC separately from plotting. The output of our first-level function then became an accepted sample. Toni and Stumpf (2010) use tolerance schedules in their paper, rather than a single tolerance level. In picking our tolerance level, we chose the median of these tolerance schedules (=25 for the data in Table 2 and =10 for the data in Table 3). We used the same number of accepted samples as the article (=1000). We could have used a greater number of accepted samples, at the cost of greater computing time, but since we are attempting a replication we decided to mimic the original

article as closely as possible. Another design choice we made was breaking up Table 2 and Table 3 into their respective outbreaks. Doing so means we have to input two observed datasets (e.g., the first outbreak in Table 2 and the second outbreak in Table 2) rather than just one. But this made coding easier when building the second-level functions. In Table 2 and Table 3, the column numbers behind the ideas in the second outbreak do not make sense. For example, a household size of one in the second outbreak occurs in the sixth column. Breaking Table 2 and Table 3 down into their respective outbreaks allows a household size of one to occur in the first column for both the first and second outbreaks, which makes more sense and is easier to code.

Results

Below are the comparisons of the parameter posterior distribution between our result and that in Toni and Stumpf (2010). In this figure, panels on the left column are from Toni and Stumpf (2010), and panels on the right column are our result. Panels on the first row are the parameter distributions for the different outbreaks of the same strain. Panels on the second row are the parameter distribution for the different outbreaks of different virus strains. From the panels on the first row, we can tell that we get a consistent result that the distributions of qc_1 and qh_1 overlap with the distributions of qc_2 and qh_2. This suggests that we can characterize different outbreaks of the same strain with a two-parameter model. They share the same parameters. From the panels on the second row, we can observe different distributions of these four parameters. This means that the outbreaks of different virus strains have different characteristics and a model with four- parameters is required.



Conclusion

Overall, what is the better model for explaining disease spread? A model that has the same values for the probability an individual escapes infection from their community and the probability an individual escapes infection from their household, or a model that allows these probabilities to vary across outbreaks? Figure 3a) in Toni and Stumpf (2010), and successfully replicated here, shows large overlap between the two parameters for two different outbreaks of the same strain of influenza. This suggests that the same parameters can explain different outbreaks of the same strain of influenza (i.e., this suggests the two-parameter model). Figure 3b), also successfully replicated here, shows distinct clustering of parameters for two different outbreaks of differing strains of influenza. This suggests that different parameters are better at explaining different outbreaks of different strains of influenza (i.e., this suggests the four-parameter model). Overall, whether a two-parameter or four-parameter model is best at explaining disease spread depends on if the outbreaks are of the same or different strains.

GitHub Link: https://github.com/ethanphilipweiland/STAT-S-610-Final-Project